

A Project Report On

# CLOUD SERVICE SELECTION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirement for 8<sup>th</sup> semester

**Bachelor of Engineering**

in

Computer Science and Engineering

**DAYANANDA SAGAR COLLEGE OF ENGINEERING**

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with A grade

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078



*Submitted By*

**Navneeth Krishna M 1DS17CS727**

**Nishchala M 1DS17CS733**

**Oindrila Chakraborti 1DS17CS734**

*Under the guidance of*

**Prof. Poornima A B**

Asst. Professor, CSE , DSCE

**2020 - 2021**

Department of Computer Science and Engineering

**DAYANANDA SAGAR COLLEGE OF ENGINEERING**

Bangalore - 560078

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

## Dayananda Sagar College of Engineering

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with A grade

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078

## Department of Computer Science & Engineering



### CERTIFICATE

This is to certify that the project titled **Cloud Service Selection using Machine Learning** is a bonafide work carried out by **Navneeth Krishna M [1DS17CS727]**, **Nishchala M [1DS17CS733]**, and **Oindrila Chakraborti [1DS17CS34]** in partial fulfillment of 8th semester, Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during the year 2020-21.

**Prof. Poornima A B**

(Internal Guide)

Asst. Prof. CSE, DSCE

**Dr. Ramesh Babu D R**

Vice Principal & HOD

CSE, DSCE

**Dr. C P S Prakash**

Principal

DSCE

Signature:.....

Signature:.....

Signature:.....

Name of the Examiners:

1.....

2.....

Signature with date:

.....

.....

## Acknowledgement

We are pleased to have successfully completed the project **Cloud Service Selection using Machine Learning**. We thoroughly enjoyed the process of working on this project and gained a lot of knowledge doing so.

We would like to take this opportunity to express our gratitude to **Dr. C P S Prakash**, Principal of DSCE, for permitting us to utilize all the necessary facilities of the institution.

We also thank our respected Vice Principal, HOD of Computer Science & Engineering, DSCE, Bangalore, **Dr. Ramesh Babu D R**, for his support and encouragement throughout the process.

We are immensely grateful to our respected and learned guide, **Prof. Poornima A B**, Asst. Professor CSE, DSCE, for the valuable help and guidance. We are indebted to her for guiding us throughout the process and her useful inputs at all stages of the process.

We also thank all the faculty and support staff of Department of Computer Science, DSCE. Without their support over the years, this work would not have been possible.

Lastly, we would like to express our deep appreciation towards our classmates and our family for providing us with constant moral support and encouragement. They have stood by us in the most difficult of times.

**Navneeth Krishna M 1DS17CS727**

**Nishchala M 1DS17CS733**

**Oindrila Chakraborti 1DS17CS734**

# Cloud Service Selection using Machine Learning

Navneeth Krishna M, Nishchala M, Oindrila Chakraborti

June 6, 2021

## Abstract

Cloud Service Selection has progressed as a cloud computing paradigm of entrusting good faith in a Cloud Service Provider (CSP) for the services with specifications that are legally agreed upon by the involved parties. The procedure of selection narrows down to multiple factors that vary based on a users needs, the environment(s) in use, the volatility of the service(s) required, and the scale at which the transactions are made. Due to the dynamic and fast-moving nature of a CSPs operability, keeping track of all the factors involved in choosing a reliable cloud service becomes tedious, resource-intensive, and expensive to maintain consistency in the predictions about reliability. However, limitations such as biases in datasets, lack of standard tests to assess trustworthiness, and minimal flavors of Cloud Service Providers pose a challenge in establishing these parameters which in turn are necessary for assisting Cloud Users to choose the service that is most feasible for their requirements. Since such detailed moderation is rendered unfeasible, predicting the reciprocation of commitment to the service agreement narrows down to the most important quality of this paradigm known as Cloud Service Trustworthiness.

In this project, we optimize the selection of a cloud service by analyzing user requirements and testing the cloud services performance. Utilizing Analytic Hierarchy Process, the Cloud Service Optimizer renders statistical data through Dashboards along with insights of cloud services. Our model aims to build a comprehensive analysis system that can derive powerful results through the processing of user queries and historical cloud service performance. Additionally, we aim to utilize optimal algorithms that prioritize the requirements of the Cloud Service Customers while considering the ground truths of Cloud Service Providers.

# Contents

1	<b>1 Introduction</b>	7
5	1.1 Cloud Service Selection . . . . .	7
10	1.1.1 How to choose a cloud service? . . . . .	7
15	1.1.2 Cloud Services that are in use today . . . . .	9
20	1.1.3 SLA Compliance and Industry Standards . . . . .	10
25	1.2 Machine Learning . . . . .	12
30	1.2.1 Linear Regression . . . . .	12
35	1.2.2 Linking ML with Cloud Service Selection . . . . .	13
40	1.3 AHP/MCDA . . . . .	13
45	1.4 Real World-Application . . . . .	15
50	1.5 Organization of the Project Report . . . . .	15
55	<b>2 Problem Statement and Proposed Solution</b>	17
60	2.1 Problem Statement . . . . .	17
65	2.2 Existing Systems . . . . .	17
70	2.3 Proposed Solution . . . . .	17
75	2.3.1 Cloud Service Optimization . . . . .	17
80	2.3.2 AHP/Multiple Criteria Decision Analysis . . . . .	20
85	2.3.3 Optimal Rank Decision . . . . .	21
90	2.3.4 Graph Theory . . . . .	22
95	2.3.5 JMeter Log Collection . . . . .	22
100	2.3.6 Analysis of Historical Data . . . . .	23
105	2.3.7 Interactive User Interface . . . . .	25
110	2.4 System Requirements . . . . .	27
115	<b>3 Literature Survey</b>	29
120	3.1 Predict Trustworthiness of Cloud Services Using Linear Regression Model . . . . .	29
125	3.2 "Cloud service trustworthiness assessment based on cloud controls matrix" . . . . .	29
130	3.3 "Fuzzy logic based cloud service trustworthiness model" . . . . .	30
135	3.4 "A Trustworthiness Evaluation Framework in Cloud Computing for Service Selection" . . . . .	31
140	3.5 "A Cloud Service Selection Method Based on Trust and User Preference Clustering" . . . . .	31

<b>4 Architecture and Design</b>	<b>33</b>
4.1 System Overview . . . . .	33
4.2 Software Architecture . . . . .	35
4.2.1 System Block Diagram . . . . .	35
4.2.2 Data Flow Diagram . . . . .	36
<b>5 Implementation</b>	<b>38</b>
5.1 Implementation Platform . . . . .	38
5.1.1 Hardware . . . . .	38
5.1.2 Software . . . . .	38
5.2 Implementation Details . . . . .	39
5.2.1 Organization of implementation files . . . . .	39
5.2.2 AHP Model . . . . .	40
5.2.3 Front-end component . . . . .	44
5.2.4 JMeter Performance Testing . . . . .	49
5.2.5 Database Structure . . . . .	50
5.2.6 Machine Learning . . . . .	52
5.2.7 Back-end Routes . . . . .	53
5.3 Datasets Collection . . . . .	54
<b>6 Testing</b>	<b>58</b>
<b>50 7 Experimentation and Results</b>	<b>60</b>
<b>8 Conclusion</b>	<b>70</b>
<b>9 References</b>	<b>72</b>
<b>10 Appendix</b>	<b>73</b>

# 1 Introduction

## 55 1.1 Cloud Service Selection

Cloud Service Selection has progressed as a cloud computing paradigm of entrusting good faith in a Cloud Service Provider (CSP) for the services with specifications that are legally agreed upon by the involved parties. The abundance of seemingly similar Cloud Services and the enormous range of customized user preferences and requirements have led to difficulty in choosing the optimal 60 CSP. Some of the main issues faced by CC are lack of trust in service providers with respect to availability, reliability and efficiency of services at their time of need, ambiguity in SLAs, non-compliance with SLA, absence of diversified trust elements, and Quality-of-Service guarantee. Although a number of cloud service e-marketplaces are present, users find it very difficult to manually compare the services of different CSPs: especially new users who must go through each 65 features description separately during Cloud Service Provider selection. This is where the model comes into the picture to make the job quick, easy and efficient.

### 1.1.1 How to choose a cloud service?

Currently, in addition to the absence of a common framework that can be used to judge the trustworthiness quotient of a particular CSP, and the various dissimilarities present between two 70 CSPs which render them as completely different from each other, there is no solid way to judge the aforementioned trustworthiness metric of any CSP. With the current industrial standards that are set, for example, audit scores, reviews, and SLAs, the trustworthiness metric can be easily tampered with to favour the CSP and not provide accurate results to the end-user.

A few pointers to be kept in mind while choosing a best-suited CSP are listed below:

- 75 • **The Current Competitors:** Even though the cloud industry has been experiencing an exponential increase with respect to its usage, there are several competitors that already exist in the market. It may include giants present in the cloud game for a very long time, or even a selected set of newer, smaller-scale competitors..
- 80 • **Cloud Security:** The user needs to analyse and predict the security measures that are applicable to their requirements and map them to the security measures provided by each of the CSPs. In addition to this, the user needs to make themselves aware of the mechanisms through which their data is preserved, and the services which are provided freely as well as the services which come under the category of pay-as-you-go category.

85 • **Cloud Compliance:** The user needs to ensure that the services provided by the CSP meet the industry standards and comply with the listed terms and conditions. The user needs to be well aware of their responsibilities and the features of the CSP which can be used to check off their compliance standards.

90 • **Architecture:** While choosing a CSP, another thing to keep in mind the way in which the cloud infrastructure will be incorporated into the working of the users/ organizations workflows, with respect to both current and future scenarios. The different cloud storage architectures also need to be kept in mind, i.e., the type of multilevel storages, archival storages provided, etc.

95 • **Manageability:** The aspects which fall under the responsibility of the user to manage while integrating the services with the respective CSPs should also be taken care of. The time and effort estimates which the user has to spend can adobe a critical deciding factor.

100 • **Service Levels:** Service Level Agreements (SLAs) hold a large amount of weightage when it comes to choosing a CSP. It is essential when organisations / businesses / individuals have strict requirements in terms of the facilities they are offered, for example, availability, reliability, scalability, security, etc. The SLAs act as an agreement between the CSP and customer to ensure that the threshold for the quality-of-service is met. In case there is a scenario that is played out where the end-user / customer does not receive up to the mark service, the CSP has to take measures to make it up to the customer as well.

105 • **Support:** Another critical aspect to be taken into consideration is the amount of support that the particular CSP will provide when required. On one hand, the mode of support provided can be in the form of chat support which might not be acceptable by certain users, whereas in case of the different CSP, there might be a dedicated resource available solely for support which might fit well with many users.

110 • **Costs:** The amount that goes into migrating to a particular cloud service might not be the most important factor that needs to be considered while choosing a CSP, but it does have significant weightage. There are various cost plans available that range from pay-as-you-go basis, reserved methods, up to volume discounts as the service usage increases. The user can assess which plan suits their requirement best.

### 1.1.2 Cloud Services that are in use today

There are several Cloud Service Providers in the market and have transformed into the primary requirement when users want to use the facilities provided by them to host their applications on the internet. As each day passes, there are more budding CSPs coming into existence, but there are a few giants that have made their mark for quite some time in the cloud market. Here are a few of the dominating and established CSPs in the market currently.

- **Amazon Web Services (AWS):** AWS has been one of the longest standing competitors in the field of cloud computing and specialises in database, serverless deployments and artificial intelligence. It provides an array of services such as Elastic Compute Cloud (EC2), Amazon Simple Storage Service (S3), Amazon Relational Database Service (RDS), etc.
- **Microsoft Azure:** Microsoft's Azure cloud along with Microsoft's provisions of software-as-a-service makes this particular CSP rank itself in the second position in the list of most preferred cloud service providers. It focuses on scaling, artificial intelligence, and providing cheap (if not free) services as a public cloud infrastructure.
- **Google Cloud Platform (GCP):** This particular CSP is ranked third on the aforementioned list. GCP, although having faced hiccups in the past, is now focusing on scaling out, building partnerships with established enterprises such as SAP, VMWare, etc., and combining its G Suite and Google Cloud sales efforts.
- **Hewlett Packard Enterprise:** The HP cloud services offered are hybrid in nature and focuses on connecting cloud with edge computing. The strategy revolving around this particular cloud involves its hardware stack, the various edge compute devices (via Aruba), storage and networking facilities, and multiple software platforms like Greenlake, Synergy, etc. This particular CSP prefers to call itself a hybrid IT setup rather than a multi-cloud environment. HPE has established partnerships with enterprises such as Red Hat, VMware, and integrated and converged systems with cloud providers.
- **Cisco Systems:** This particular CSP is a structured hybrid cloud through a network and API prism. A network-centric approach has been taken while implementing the services offered by the cloud. The headliner with respect to this cloud is ACI, short for an architecture known as Application Centric Infrastructure. This architecture along with the network-centric policy focuses on management of the services, policies related to the services offered,

and operations which are reserved for applications that are deployed across different / multiple cloud environments.

#### <sup>145</sup> 1.1.3 SLA Compliance and Industry Standards

**Service Level Agreement (SLA):** Most CC and CSP use a written document as a basis for all services to be provided as well as the quality that is expected from those services. Detailed and formal SLAs are very important to establish between both parties since companies and individuals infrastructure can have serious impacts if something goes wrong with the services that they or <sup>150</sup> their customers are using. These impacts can include Security breaches, unavailability of services, Personally Identifiable Information (PII) theft, etc. SLAs between CCs IT teams and CSP differ depending on factors such as type of services and CCs business needs. Although, most SLAs cover basic attributes such as Speed, Availability, Security, Service Uptime, etc. SLAs also come handy in monitoring cloud governance and compliance. The Service Level Objectives (SLO) are <sup>155</sup> decided based on Key Performance Indicators (KPI) provided by the customers after negotiation with the CSP. There are integrated monitoring services with Cloud Services which can monitor SLA compliance by the processes and alert the users if SLA is violated. CSP usually have generic pre-made SLA but the IT department of the client companies should review the SLA along with their legal counsel before deciding to go ahead with the services. Quality of Services (QoS) <sup>160</sup> must be assessed carefully and decided upon in the SLA. SLA has precise quantifiable values assigned to various attributes to show their performance metrics. SLAs should also include remediation for failing agreements. The items which are usually focused on in the SLA are as follows:

- **Availability and Uptime:** Availability and Uptime are two of the most important attributes <sup>165</sup> that CC focus on. These are not the same as a resource might be up but not accessible by the user which means that its not available. The SLA must provide in percentage the Availability and Uptime of various services.
- **Security and Privacy:** The CC are ultimately responsible for their clients data privacy. Hence, how the CSP handle data security is of paramount importance. Strong user authentication must be used for data encryption, antivirus and malware, etc. The CSP should have <sup>170</sup> intrusion detection activated at all times.
- **Performance:** Generally, Response time is considered as the attribute to determine performance. Hence, metrics must be decided on areas such as service volumes, different demands

at different time intervals, QoS, workloads and peak times.

- 175 • **Network Changes:** Dynamic scaling of equipment by CSP might lead to users facing down-time. Major changes must be conveyed to the customers in advance.
- **Support Agreements:** Most CSP have round-the-clock Help Center Support for CC but Customers might provide some support on their part for their clients. This ratio of Support services provided by the IT team of companies and CSP must be decided in SLA.
- 180 • **Exit Strategy:** Appropriate steps must be included in case any dispute arises and exit strategy has to be devised for CC to migrate to another service provider.

**Consensus Assessments Initiative Questionnaire (CAIQ):** The CAIQ offers an industry-accepted way to record what security controls exist in different services like SaaS, IaaS and PaaS. CAIQ is answered by all CSP as set of Yes/No answers to questions which CC and cloud auditor 185 may wish to ask of a cloud provider to assess their compliance to the Cloud Controls Matrix (CCM). It enables CC to determine the security provisions provided by CSP and decide whether its safe to migrate to the particular CSP.

**Cloud Controls Matrix (CCM):** The CCM is Cybersecurity Framework for Cloud which 190 contains 197 control objectives that can be fulfilled over 17 domains covering key areas in cloud technology. This is used as an objective based assessment of CSP by CC to determine the extent of security features provided during implementation.

**National Institute of Standards and Technology (NIST):** The NIST cloud computing definition is accepted worldwide to provide a clear picture of cloud computing technologies 195 and services. The Cloud computing reference architecture is a conceptual model which enables and empowers discussing the requirements, structures, and operations of cloud computing. The model can be used with any service vendor.

200 **International Organization for Standardization/International Electrotechnical Commission (ISO/IEC):** ISO/IEC industry standards define many attributes of cloud computing technology like Interoperability and Security. On the basis of these defined formal levels of features, users can judge Cloud Services. This helps in avoiding confusion and introducing transparency between CC and CSP.

**European Network and Information Security Agency (ENISA):** ENISA comprises a group of subject experts and representatives in the fields of Industrial, Academic and Governmental organisations. Recent works by ENISA include an assessment which determines the emerging and future risks and security failures which CSP might encounter. It also outlines the  
210 key benefits and features of using Cloud Services.

## 1.2 Machine Learning

Along the norms of every product offered as a service, ML sides along as an enhancement and learning mechanism. As more and more technologies evolve with data output and deviation in results, the application of predictability unfolds in higher magnitudes. As a fitting result, MLs  
215 introduction to technological industries and products offer lucrative foundations. Among such important introductions, ML plays a pivotal role via algorithms that understand patterns and trends in collected data through viable resources. These days, Machine Learning is no longer computation-intensive and can be utilized with little compromise in performance via household desktops and laptops. In the domain of Cloud Service Selection, Machine Learning is a popular  
220 choice of primary algorithmic calculation.

### 1.2.1 Linear Regression

A simple formula  $z = aw + g$  emulates the relationship between two kinds of data popularly known as independent and dependent. Through a line of fit, one can estimate data as mostly a pattern that forms a line along the axes. One Independent and One Dependent variable constitute  
225 Single Linear Regression whereas more than one dependent variable constitutes a Multiple Linear Regression. In its representation,  $z = a_1w_1 + a_2w_2 + \dots + g$  emulates a perfect estimate of weights distributed over a set of variables that together answer an independent variable.

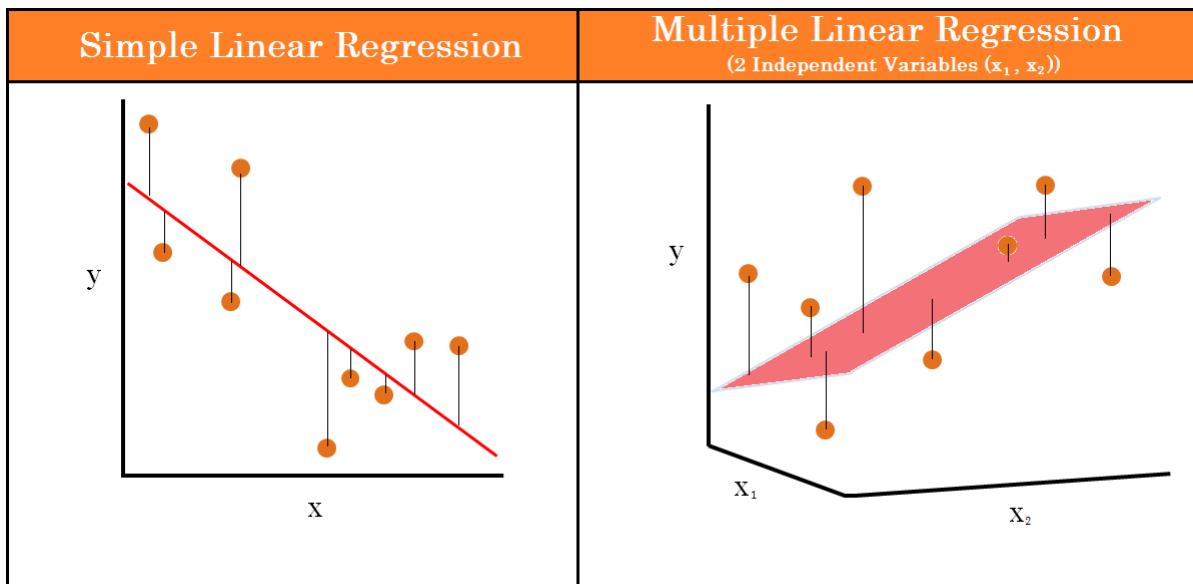


Figure 1.1: Linear Regression Common Types

<sup>230</sup> **1.2.2 Linking ML with Cloud Service Selection**

Specific to Cloud Service Selection, the vast potential of Machine Learning can be incorporated in order to yield many solutions ranging from instance type finalization to service migration. As we delve deep into gaining insights from cloud customer data that is made public by Cloud Service Providers, we notice several patterns that can be exploited for product-side and technical upgrades.

<sup>235</sup> Significantly, when utilizing user inputs and feedback as training models, engineers can predict Cloud Service features like availability, reliability, throughput, etc. Thus, the relationship between Cloud Service Selection and Machine Learning is immensely rich with various applications. For instance, Regression can learn to predict the expected value of Connectivity based on the available values of Throughput and Standard Deviation.

<sup>240</sup> **1.3 AHP/MCDA**

For making the right business decision, there is a need to utilize advanced analytical decision-making algorithms. This is classified as a domain of Operations Research. Under this, Multiple Criteria Decision Making (MCDM) is an integral tool used to evaluate numerous criteria that are conflicting or are different relative to one another. By organizing decision-making problems <sup>245</sup> into structures of relative importance, enterprises can expect uniformity and consistency in their decision finalization. Using certain decision-making software, MCDMs various types such as Ranking based on optimal points, Weighted Sum Model, Weighted Product Model, and Analytic Hierarchy Process are developed. We expand our projects scope of finalizing instance types of a given service type by utilizing Analytic Hierarchy Process.

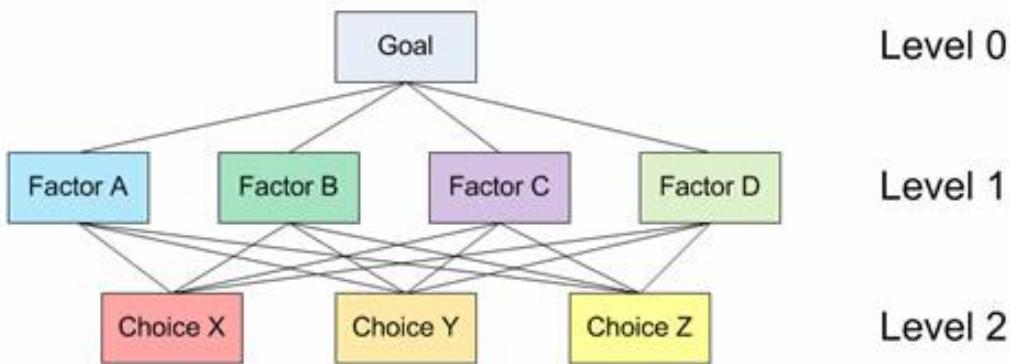


Figure 1.2: Sample AHP Tree

Analytical Hierarchy Process is an advanced decision-making algorithm that solves complex mathematical problems. To reach an overall goal, a tree-like structure is envisioned by constructing a model. The leaf nodes of the tree begin with the inputs to AHP. These are relative importance indicators of all the inputs. For example, if someone wants to know the overall strength of teams that play in the English Premier League. As the leaf nodes, we input the following parameters: Attacking Strength, Defense Strength, Midfield Strength, and Goalkeeping Strength.

In these nodes, we define the relative importance of each team per parameter. For example, take Attacking Strength. It is defined as:

attacking\_strength = (Chelsea, Arsenal): 1.2, (Chelsea, Manchester United): 1.4, (Manchester City, Arsenal): 1.3, (Arsenal, Manchester City): 0.7 ... and so on. The values defined are the relative comparison of two teams based on an individual strength.

Such dictionaries are prepared and are fed as an entire structure in a leaf node. To determine the values of the nodes, we apply several techniques specific to the datasets. Take the first entry: (Chelsea, Arsenal): 1.2. Here, it means that Chelsea is 1.2 times better than Arsenal in terms of Attacking Strength. Similarly, (Arsenal, Manchester City): 0.7 is an entry that defines Arsenals attacking strength to be 0.7 times that of Manchester City. With such values, one can draw insights on individual importance to each criterion. Once all leaf nodes are described, the hierarchy becomes the most important step. Hierarchy is a set of priorities or ranks that define the levels of the nodes in the tree. In this specific example, suppose we want to place the next highest importance on Match Performance. Further, all the leaf nodes are added to a set of match\_performance nodes based on a ranking mechanism. Among the leaf nodes, the

relative importance between each leaf node (Attacking strength, midfield strength and others) is established.

Assuming that we define Attacking strength to be the most important, the match\_performance node will be defined as:

280 match\_performance = (Attacking, Midfield): 1.2, (Attacking, Goalkeeping): 1.4, (Defense, Attacking): 1.3, (Midfield, Defense): 0.7 ... and so on.

As we further describe each of the levels, we finally create a Comparison Matrix for each node at each level. The root node will be the overall result after consideration of each level of the 285 AHP tree. Using the Compare Matrices and establishing relative importance, criterion weights can be output as follows: Arsenal: 0.2, Chelsea 0.45, Manchester United 0.3 and so on. The overall sum of the weights are 1 and these instances are described relative to one another. The best teams are those which are higher in magnitude relative to the other teams.

## 1.4 Real World-Application

290 The Cloud Service Selection model employs Multi Criteria Decision Analysis (MCDA)/Analytic Hierarchy Process (AHP) to assess and evaluate several instances of service types. Similarly the same techniques can be applied to solve various other problems in todays world.

- Knowledge Based Economics: MCDA is used to evaluate decision systems involving Budgets, Expansion of Business, Expenses, etc.
- Resource Provisioning: Resources can be provisioned effectively while keeping relevance, urgency and budget as criteria in MCDA model.
- Assessing competitiveness of teams: Competitive factors can be assessed based on which teams can be ranked in any Competitions
- End Product sales in market: Decision analysis models can be used in taking in attributes like 300 various dynamically changing factors of customers, boundary conditions like budget, location, etc. and analysing previous sales data to find suitable target groups and locations to sell the product.

## 1.5 Organization of the Project Report

The project report is organized as given below:

305 In Chapter (2), we discuss the problem statement and our solution to the problem. The same

chapter also deals with the other existing technology. The Chapter that follows i.e. chapter (3) consists of the details on the literature survey of the papers which align to the problem statement and the proposed solution. In Chapter (4), we present the System Architecture and Design from User point of view as well as through Sub-systems divided in the entire structure of  
310 model. In this chapter, the Data Flow Diagram makes it easier to understand the overall control and sequence of the system. The next chapter, chapter (5) gives the requirements and details about the implementation of the system along with Directory Structure. Chapter (6) deals with the testing of the product. In Chapter (7), we discuss the end results and factors influencing the results of the system. The same chapter deals with establishing the optimal parameters  
315 for the system. Chapter (8) concludes the paper along with diving into some of the Future Enhancements. Chapter (9) mentions the references used during the development of the system. The other supporting information and the source code are gathered in the last section named Appendix.

## 2 Problem Statement and Proposed Solution

### 320 2.1 Problem Statement

To develop a sophisticated cloud service selection system that constitutes user requirements elicitation and the logic that derives the equation of fit that serves the need to present heuristically-accurate results to the cloud user utilizing historical cloud data.

### 2.2 Existing Systems

#### 325 CUELOGIC Cloud Optimization

Related to Private, Public, and Hybrid Clouds, CUELOGIC deals with Cloud Optimization for enterprises and other customers to choose the right cloud product for them. They offer consulting for choosing cloud services and assess the value of a cloud. In addition, they take responsibility for administering cloud infrastructure for enterprises. To build an ecosystem that sustains reasonable

330 prices and valuable cloud services.

#### DENSIFY SOLUTIONS

This product offers cloud control for enterprises along with optimization features. Its unique Cost Management tool caters to allocating reasonable pricing to cloud services and amplifies the quantity output that can be expected from a cloud service. Its Infra as a Cloud feature stands out 335 in offering Code Optimization where cloud services are commissioned at the lowest price ranges by automating the process of cloud selection.

#### OPSANI

OPSANI is an autonomous group that offers continuous cloud optimization as a service. It extends the workability of Artificial Intelligence in the domain of Cloud Computing. This AI brings closer 340 the ability to predict where investments can be saved and where they can be maximized in the cloud. Main objectives of this product are not limited to improving cloud performance and adaptable performance tuning.

### 2.3 Proposed Solution

#### 2.3.1 Cloud Service Optimization

345 The process of determining the right amounts of cloud service resources such as allocation of storage, provisioning of CPUs, configuring memory requirements, etc. while administering judicious spend on cloud services in terms of quality and quantity can be defined as Cloud Service Optimization.

Cloud Service Optimization is generally done to cut costs that can ramp up the overall bill and

350 to reduce wastage of allocated resources. It is achieved through paid software as described in Existing Systems that usually generate optimal findings with respect to choosing the right cloud service based on resources, expected statistics, and other common metrics.

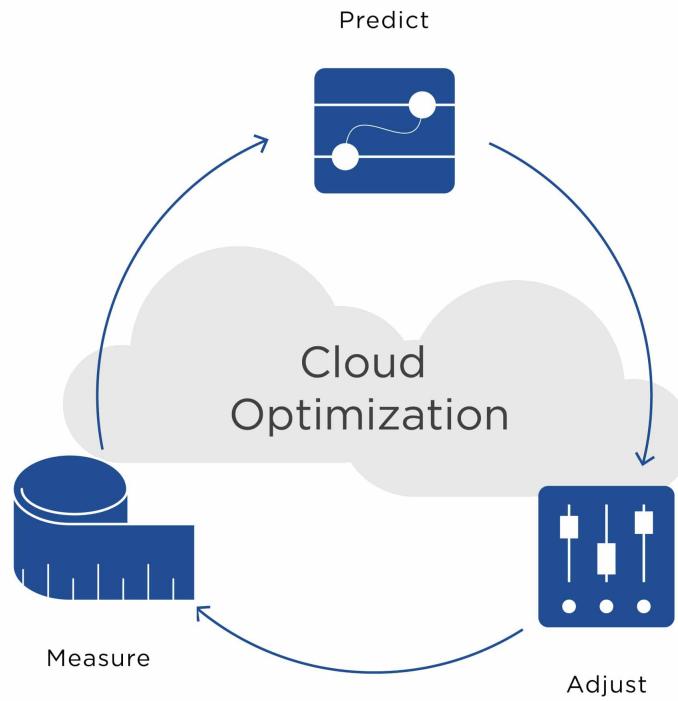


Figure 2.1: Cloud Optimization Common Metrics

355 In our proposed solution, we utilize three important metrics: (i) Filters: Filters are content areas that describe the fundamental features of a cloud service. Ex: Throughput, Uptime, Downtime, etc. In our proposed solution, these are the following filters that are accepted for the services:

SERVICE TYPE: AWS EC2

1. Consistency
2. Error Rate
3. Latency
4. Elapsed Time
5. Connection Time
6. Throughput
7. CPU Utilization
8. Memory

**SERVICE TYPE: AWS RDS**

1. Consistency
2. Error Rate
3. Latency
4. Elapsed Time
5. Connection Time
6. Throughput

**SERVICE TYPE: AWS S3**

- 375 1. Consistency
2. Error Rate
3. Latency
4. Elapsed Time
5. Connection Time
- 380 6. Throughput

The user chooses their desired filters and also allocates priorities to them to rank them overall. For services without instance types, we utilize the prediction model to estimate the expected values of one of the above parameters by considering inputs to the other values.

**(ii) Instance Types**

385 Instance Types of Cloud Service Types are types of a given service that vary in the resources offered, support extended, and metrics available. The ranking that is done in our solution is to optimize choosing the best instance type of a cloud service type among a set of alternatives. In our proposed solution, we offer instances of AWS EC2 and AWS RDS. With these instance types, we establish results (the best instance type) based on our findings from rigorous testing through  
390 JMeter.

**(iii) Rankings/Priorities**

For each of the opted Cloud Service Type, we generate rankings as a type of priority-based comparison matrices created in the Back-end. The weighted averages of all criteria are decided after the user enters rankings. For cloud services with no instance type, we utilize the ML algorithm  
395 to predict the expected value of a content area given the input values for the rest of the content areas. These input values to AHP are defined as follows: The inputs are defined in a structured way in the user interface. They are non-negative integers/decimals that signify the quality of the respective filter.

- Consistency (stdDev): 100 to 1000. Here, 100 indicates a low level of Consistency and 1000 indicates a high level of Consistency. The standard deviation value of the dataset under scope informs us about the scattering / clustering of all the data points present in the data set around the obtained average. The smaller the value of standard deviation obtained, the higher is the consistency of data.
- Error Rate (errorRate): 0 to 100. Here, 0 refers to low error rate and 100 refers to high error rate. It is the percentage value of errors caused due to various reasons when connecting to the Back-end hosted by a service type. Total percentage of errors found for a particular sample request. 0.0 percent shows that all requests completed successfully. Total equals the percentage of error samples in all samples.
- Latency (Latency): 0 to 300. Here, 0 refers to a low level of Latency and 300 refers to a high level of Latency. application client.
- Elapsed Time (elapsed): 0 to 300. Here, 0 refers to low Elapsed time and 300 refers to high Elapsed time.
- Connection Time (Connect): 0 to 300. Here, 0 refers to the low level of Connection Time and 300 refers to the high level of Connection time.
- Throughput (Throughput): 0 to 5000. Here, 0 refers to a low level of Throughput and 5000 refers to a high level of Throughput. Hits/sec, or total number of requests per unit of time (sec, mins, hr) sent successfully to server during test.

Once all of the above-mentioned parameters are received by the back-end, appropriate algorithms are called and run through the inputs.

#### 2.3.2 AHP/Multiple Criteria Decision Analysis

In our AHP model, the leaf nodes are relative values of each instance type that are derived by calculation from JMeter logs. These relative values of each instance type of a given service type are stored in the DB and are retrieved when required. After defining each of the user-requested content areas with our relative representation of research-output log values, we determine the ranks or priorities that will become the input to the parent/root node. Here, the relative ranks are defined based on The Inverse Law. If a content area is good with increasing values obtained by research, then the ranks are as defined by the user. If a content area is bad with increasing values obtained by research, then the inverse of the ranks are fed as input relative ranks.

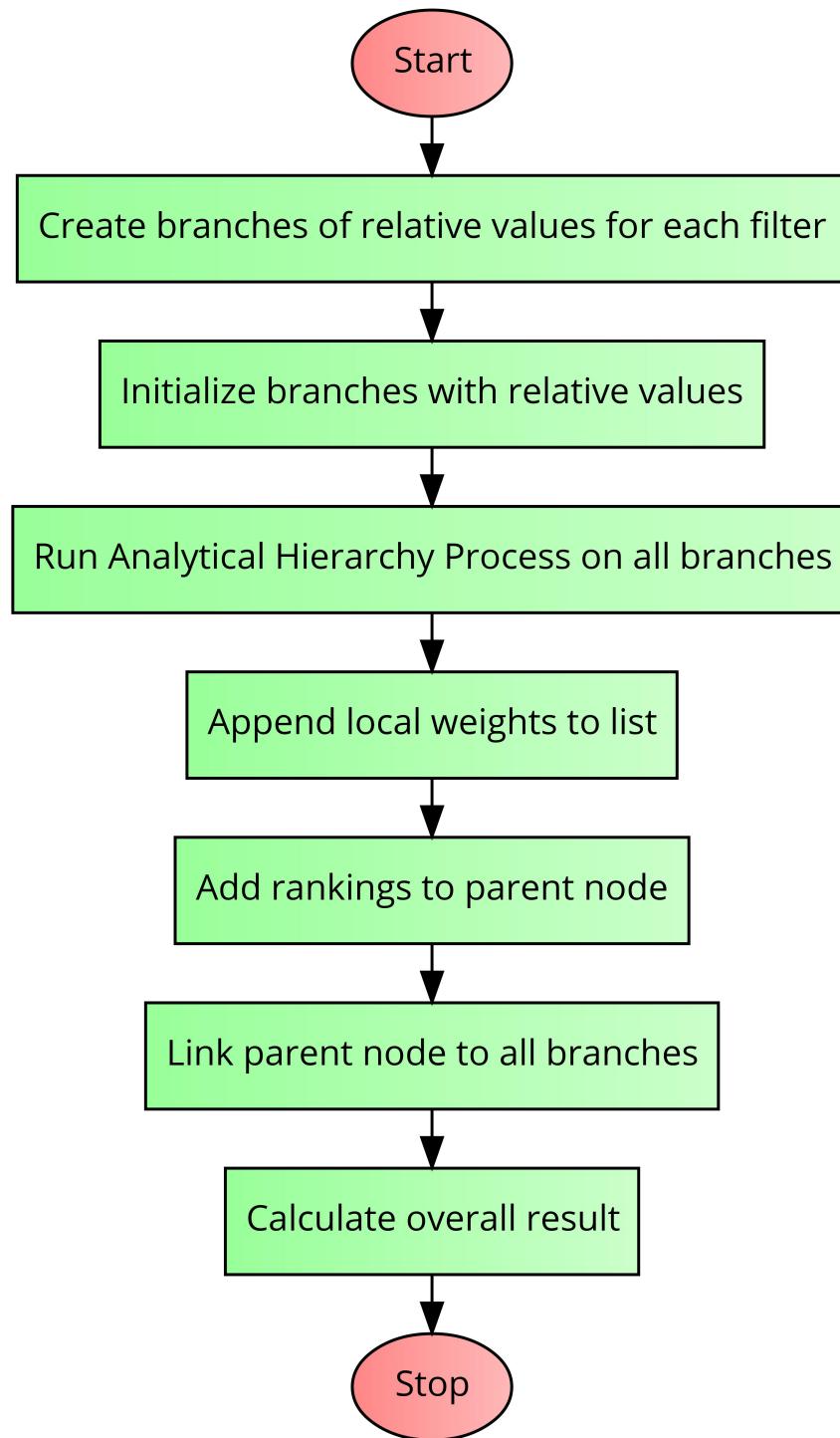


Figure 2.2: AHP Algorithm

### 2.3.3 Optimal Rank Decision

We use the Inverse Law to decide the optimal rank. For content areas that have quality inversely proportional to magnitude, the inputs for the highest level of AHP are taken as the inverse of the input ranks. For content areas that have quality directly proportional to magnitude, the inputs for the highest level of AHP are fed directly after calculating relative rankings. For the scenario when

the same priorities are entered for all the content areas, all values are defaulted to 1. With this ranking system, efficient Analytic Hierarchy weights are generated through numerous iterations of Comparison matrices.

### 2.3.4 Graph Theory

Graph Theory is an old Mathematical concept with a number of modern applications in todays world like modelling Network Topologies, resource allocation and scheduling, etc. Graph Theory is a very popular technique in the field of Computer Science to model pairwise relations between nodes. Here, Graph Theory is used to analyse and model CSP, Cloud SLA and Security Controls of Cloud. Graph Theory was found to be optimal in defining a network structure and representing information about relationships between Nodes. Graph Theory defines relationships between Nodes such as Industry Standards, Organisations that form these standards, Cloud Security Controls and Cloud Controls Matrix (CCM) given by Cloud Security Alliance (CSA) and CSP (e.g., Amazon, Google, Microsoft). The SLA standards defined by EC, ENISA and ISO/IEC have a set of 12 content areas and 82 components. The CAIQ proves to be an efficient method for collection of data and building graphs from it. The CAIQ is the document of Questions and their Yes/No answers as provided by CSP.

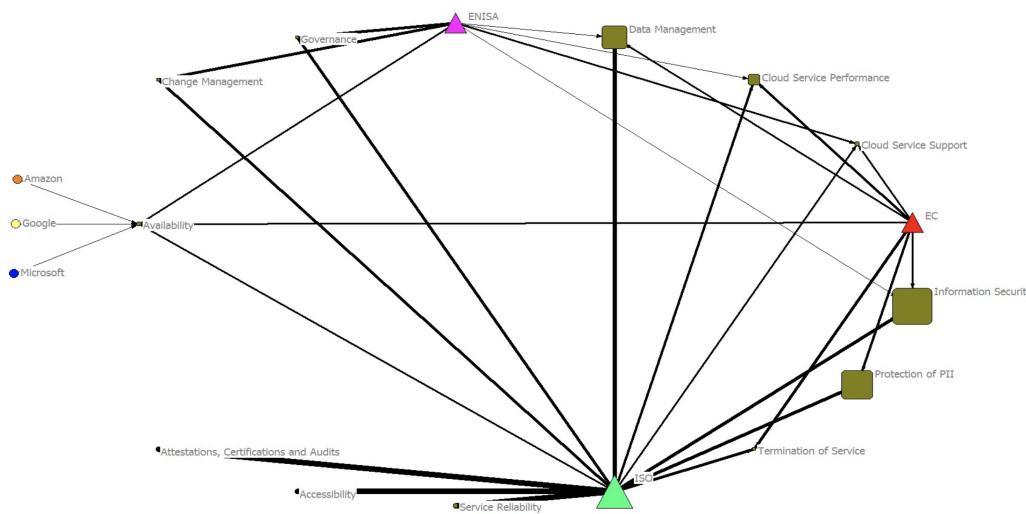


Figure 2.3: Graph Theory

Once the graph is constructed, the importance and relevance of each SLA content area is determined and quantified. The entire network structure formed between these entities along with their relative importance are then provided as input to AHP.

### 2.3.5 JMeter Log Collection

In order to establish trust between CSP and CC, the services must be tested against a wide range of parameters and attributes to check the efficiency of each service. These attributes in-

460 include CPU utilisation, Memory utilisation, Downtime, Uptime, Availability, Throughput, Latency, Consistency, etc. Although CSP have their own monitoring services integrated with the services which are provided, users tend to rely on external sources which conduct unbiased assessment of performance. Hence, to facilitate independent testing of Cloud resources, Jmeter tool is used for testing and analysing performance of services. Various instances of AWS Elastic Cloud Compute  
465 (EC2) and Relational Database Storage (RDS) as well as Simple Storage Service (S3) are tested and logs are collected every day for more than thirty days with varying factors (which might affect the performance such as Number of user requests invoked, Timestamp, Storage intensive and Computation intensive processes, etc.) are set. The EC2 instances and S3 services are tested by invoking HTTP requests through Jmeter while RDS instances are tested by sending JDBC  
470 requests to the tables created in the database.

### 2.3.6 Analysis of Historical Data

Once the logs obtained through JMeter are received by the Back-end, they are prepared to be fed into the Analytic Hierarchy Process and the Multiple Linear Regression models whenever the user interface requests a particular processing. To prepare the JMeter logs, we perform the  
475 following steps:

- Attempt login/signup
- Generate login session
- Traverse through the logs file(s)
- Obtain the required columns data
- Calculate averages of all data
- Upload the log files to the Back-end

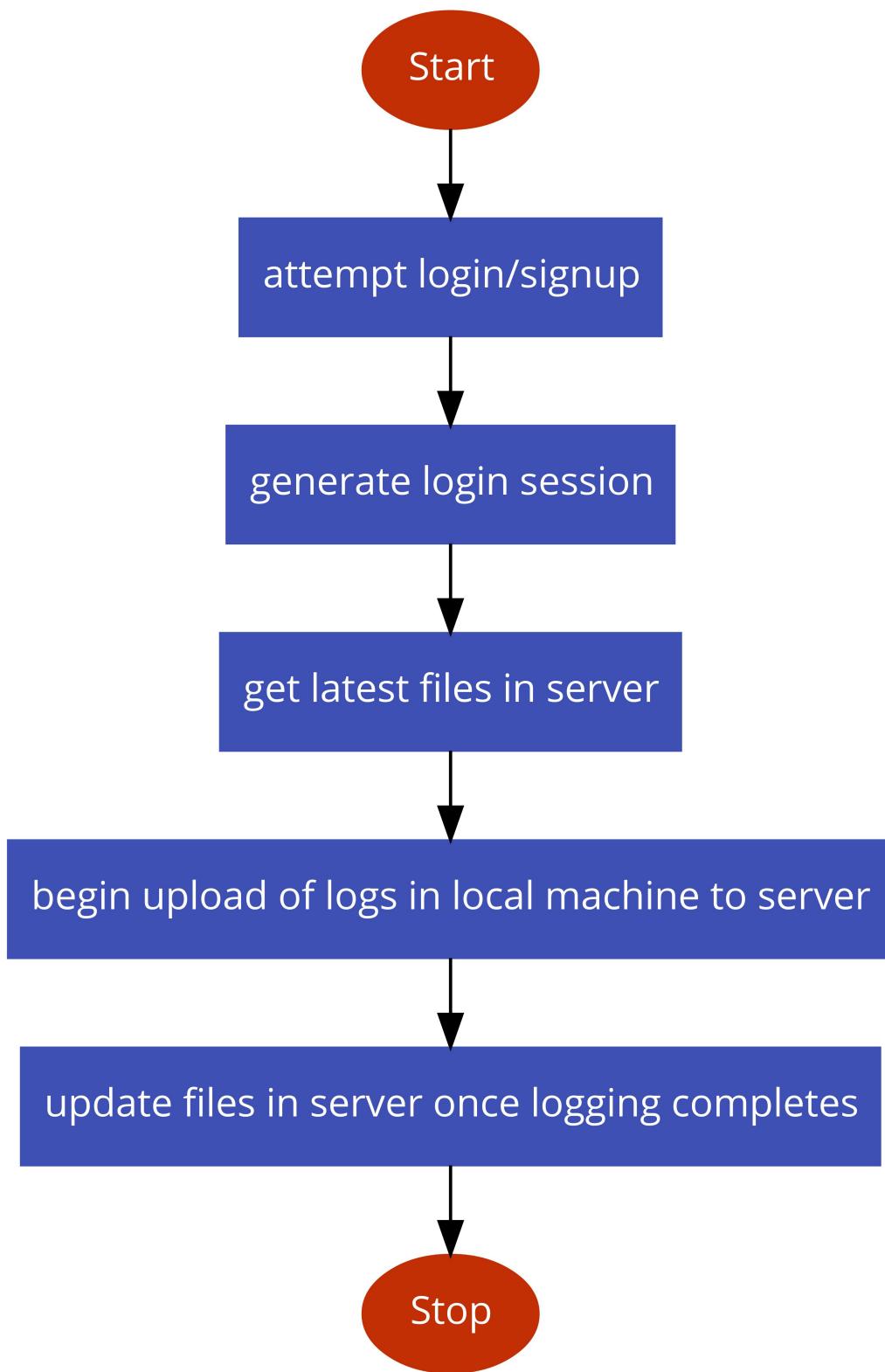


Figure 2.4: Endpoints Algorithm

Calculating Throughput, Consistency (Standard Deviation)

- Assume that the value above from a logging session summary is  $v$

- loggingSessionValue = numberofSamples collected in a logging session \* v
- totalValue = Sum of (loggingSessionValue) for all logging sessions / Sum of (numberofSamples) for all logging sessions

Calculating Elapsed time (connection + response), connection time, and latency

- 490
- Assume that the value above from a logging session is v
  - totalValue = Sum of all (v) / total number of logging sessions With the obtained total values for each of the content areas, we store all of the calculated total values in the database for further use.

### 2.3.7 Interactive User Interface

495     The proposed solution involves the end product web application (created using the React JS Framework) hosted on the free hosting platform - Heroku. It consists of an interactive User Interface with simple steps that are clearly defined to ensure a smooth usage experience with respect to the user. Incase of a first-time user, there is a detailed user manual specified under the FAQs hyperlink. The Contact Us hyperlink consists of the email IDs of the creators. They can 500 be contacted incase of any discrepancies. The Dashboard hyperlink is the section where the user inputs are taken to render graphical data which provide useful insights into the trustworthiness of the services selected by the user. Below is a detailed explanation of the functionality of the web application in a question answer format.

- For Amazon EC2 and Amazon RDS:

- 505     1. What AWS Service are you looking for?

Clicking on the 'Select Service' Button will display the services which are available for comparison and calculation of trustworthiness metric. Select the service type as per the requirement.

- 510     2. What performance metric do you want us to focus on?

Select from a wide range of available Filters and assign priority based on how important each attribute is for your business requirement. Checking the 'Assign same priority to all filters' will assign same priority to all attributes. Don't forget to select the filters that you want to apply after this step. Individual priorities can also be assigned by removing the check 515 the above checkbox, clicking on each filter and filling the priority numbers (Integer type in

range [1, n]: 1 being the highest priority, n (= Total number of Filters available) in the text boxes which appear beside them. To deselect any filter, simply click again on the selected one.

520 3. What instance types of the above listed AWS service do you want?

Select the instance types that you want to compare between and click on Save. If no instances are selected all of the shown instances will be automatically chosen for Trustworthiness Evaluation. After successfully uploading the above data, click on Show Results to assess the trustworthiness value and graphical results of the comparison.

525 • For Amazon S3:

530 1. What AWS Service are you looking for?

Clicking on the 'Select Service' Button will display the services which are available for comparison and calculation of trustworthiness metric. Select S3.

535 2. What metric should ML Prediction be applied to?

The 'Please Assign different values to each filter' must remain unchecked for S3. For S3, Users must choose n-1 (n being the total number of attributes) filters and must provide the values for each of the selected filters. The value of the only 1 remaining filter is predicted. Here, priorities can be positive Integers or Decimal values.

540 Allowed values for the selected filters:

Consistency (stdDev): 100 to 1000. Here, 100 indicates a low level of Consistency and 1000 indicates a high level of Consistency. The standard deviation value of the dataset under scope informs us about the scattering / clustering of all the data points present in the data set around the obtained average. The smaller the value of standard deviation obtained, the higher is the consistency of data.

Error Rate (errorRate): 0 to 100. Here, 0 refers to low error rate and 100 refers to high error rate. It is the percentage value of errors caused due to various reasons when connecting to the Back-end hosted by a service type. Total percentage of errors found for a particular sample request. 0.0 percent shows that all requests completed successfully. Total equals the percentage of error samples in all samples.

Latency (Latency): 0 to 300. Here, 0 refers to a low level of Latency and 300 refers to a high level of Latency. application client.

Elapsed Time (elapsed): 0 to 300. Here, 0 refers to low Elapsed time and 300 refers to high

Elapsed time.

Connection Time (Connect): 0 to 300. Here, 0 refers to the low level of Connection Time and 300 refers to the high level of Connection time.  
550

Throughput (Throughput): 0 to 5000. Here, 0 refers to a low level of Throughput and 5000 refers to a high level of Throughput. Hits/sec, or total number of requests per unit of time (sec, mins, hr) sent successfully to server during test.

After selecting the filters, press Save to upload the data. Click on Show Results to assess the trustworthiness value and graphical results of the comparison.  
555

## 2.4 System Requirements

The Model makes the job of all Cloud Customers hassle-free in selection of suitable services according to how important the services attributes are for their business requirements. The system consists of three modules: Front-end, Endpoints and Back-end. The Front-end incorporates an interactive User Interface where the users can enter their preferences regarding Services, Filters and Priorities and Instances, if applicable. These inputs are saved and sent to the Back-end where AHP technique is applied over various logs and data collected from Endpoints to rank and assess the performance of services according to Users preferences of Attributes.  
560

The System Requirements sequentially are as follows: 1. When the users open the webapp, the homepage displays an overview of Cloud Service Selection and its importance. 2. Three tabs which are available are : Dashboard, FAQs and Contact Us. Dashboard is where user preferences are recorded and the trustworthiness metrics are calculated. FAQs section has all instructions about how to use the Dashboard and rank services, Contact Us section has Email IDs of developers. 3. The Dashboard section asks Users questions regarding the Service Type they want to choose, 570 Filters and Priorities that should be assigned to them and Instance types of Services, in case of EC2 and RDS. 4. Various checks are included to avoid input of incomplete or wrong data like One Service must be chosen, Priority must be assigned, more than one Instances must be chosen for comparison, etc. When all these conditions are satisfied, users can click Save and the data is sent to Backend. 5. Backend employs MCDA/ AHP technique to assign weights to all attributes according to user preferences and logs sent from Endpoints. 6. The AHP technique helps in assessing performance as well as ranking of services. The Graph data is sent to Frontend which displays all information of trustworthiness in a graphical and user-friendly format. 7. The logs which are collected from Endpoints consist of performance records of various services against parameters such as Availability, CPU and Memory Utilisation, Latency, Response Time, etc.  
575

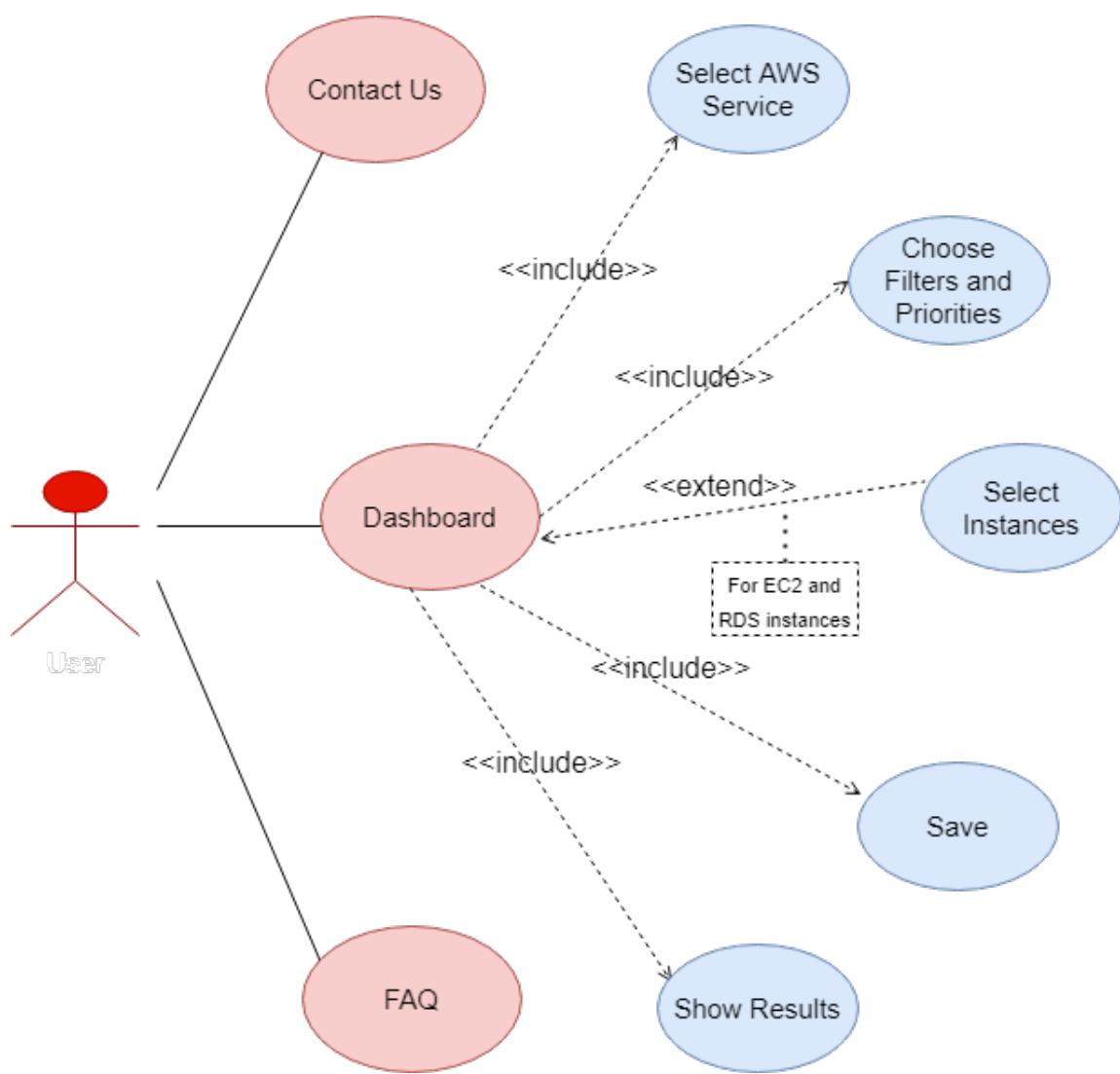


Figure 2.5: Use Case Diagram

### 3 Literature Survey

#### 3.1 Predict Trustworthiness of Cloud Services Using Linear Regression Model

585 The pay-as-you-go feature of many CSPs has enabled enhanced scalability in providing resources based on the requirements of Cloud Customers (CC). Incidentally, scalability introduces a more challenging trust relationship between CSPs and CC as users must depend on the availability of services at the time of need. In addition to scalability, users need an effective system to rank CSPs on their distinct QoS parameters based on specific user purposes. Mahesh K et al. suggest a Linear  
590 Regression Model to assess trustworthiness and choose the most optimal candidate among several Cloud Services based on user criteria. Initially, Linear Discriminate Analysis (LDA) technique was used for clustering and ranking similar services according to their QoS scores. LDA aims to filter out the services which are not along the lines of QoS attributes specified by calculating the similarity degree between service request and service. After the initial confidence is established  
595 using LDA, CC should verify confidence further by observing the adherence to SLA on the grounds of transparency, integrity, compliance, and competency in the past. If SLA verification is carried out by a Cloud Broker (CB) who acts as an intermediary entity between CC and CSPs, then the trust quotient partly depends on confidence between CC and CB. This is known as a reputation-based trust mechanism system. The model analyses different services based on QoS parameters  
600 Response Time, Throughput, Availability, Success ability, and Price in the first step. In the next step, further confidence is checked by computing the trust score and ranking of services out of nine CSPs in the evaluation process, the best service scored 0.4740 while the worst service scored 0.2724.

#### 3.2 "Cloud service trustworthiness assessment based on cloud controls matrix"

605 CCM is an initiative by CSA that deals with describing control guidelines for a cloud services provision of security. Under the control guidelines, there are control domains that are further divided into a number of controls. In CCM v3.0.1, we observe 133 controls. According to Kanpariyasoontorn & Senivongse, it is pertinent to analyze the features of each CCM control  
610 in order to produce a trustworthiness metric. The method proposes a Cloud Service Trustworthiness Assessment Method which initially defines the attributes and sub-attributes of Security and Dependability as Trustworthiness is defined to be a result of them. Sub-characterization

of Security & Dependability ensues and the new attributes map associations with the following standards: (1) The NIST SP800-53 Control Standard and (2) AICPA Control Standard both of which map to the CCM. Next, we define a degree of association between every CCM control domain and sub-attribute with a matrix called Association Matrix. To illustrate, a control domain like Application & Interface Security has degrees of 0.75, 1, 0.75, and 0 for the sub-attributes Confidentiality, Integrity, Availability, and Reliability respectively. The assurance weights for the control domains must be calculated because security assurance is one of the two components that influence trustworthiness. With the mathematical product of the assurance weights and the degree of compliance of a service with a given control domain, we establish the capability of a service P. A service capability matrix is constructed by merging all the service capabilities and is multiplied with the Association Matrix to yield the Service Trustworthiness. This technique presented a self-assessment mechanism which clearly raises questions about biases and the validity of the service trustworthiness. A germane observation is that CSA is an esteemed organization whose reputation holds value enough to treat Kanpariyasoontorn & Senivongse as a promising methodology for assessing trustworthiness.

### 3.3 "Fuzzy logic based cloud service trustworthiness model"

Pandey & Daniel make use of Fuzzy Logic for building a Cloud Service Trustworthiness Model (CSTM) whose goal is to provide a fair evaluation of the trustworthiness of cloud services. The design focuses on agents for Requestor and Provider that are stakeholders which participate in the smooth functioning of a cloud service. As and when these stakeholders request or register for a new service, common and special parameters that constitute trustworthiness are involved in a weighting and positioning process. The listed parameters are defined with first and second grade indexes before determining their weight distribution for both indexes. A Level is ascribed to different parameters based on their quality and an evaluation Fuzzy Matrix is constructed for the first-grade index. It denotes the membership degree of a given parameter of a cloud service in specified trustworthiness levels. Processing through the Fuzzy Comprehensive Evaluation Model is a cardinal phase. Here, the overall trustworthiness membership degree of a cloud service is solved for on different levels. The final result brings about an adjective like high and low for trustworthiness levels of individual parameters. This stakeholder-based topology not only evaluates the overall trustworthiness of a cloud service but also the trustworthiness of individual parameters such as Security, Maintainability, and Usability.

### 3.4 "A Trustworthiness Evaluation Framework in Cloud Computing for Service Selection"

CCs and CSPs often don't have clear information about each other for cloud service selection. A trustworthy relationship should be established between CCs and CSPs on the basis of the reputation of services as well as suitable trust factors. Due to the unavailability of uniform trust metrics for different kinds of services, L. Wang and Z. Wu. propose a trustworthiness evaluation framework which is made of five sections. The Pre-processing Section collects trust factors (quantitative and qualitative) of CSPs which are categorized and converted to a unified scale using the Trust Factor Management Section. Trust coordinate is used to further categorize them into Subjective trust, Objective trust, Direct trust, Indirect trust, Inflow trust, and Outflow trust. Entropy value is used to define trustworthiness metrics (value between 1 and -1 where 1 is completely trustworthy and -1 is untrustworthy) to measure the uncertainty in trust relationships. These factors are then sent to the Trust Factor Processing Section for further processing like updating of common factors, dealing with special factors with trust ontology alignment approach to formalize the meaning of trust for a clearer understanding for user expectations, and assigning weights to all factors which are then employed in the multi-criteria analysis approach by the Trustworthiness Decision Making Section to measure, rank or sort cloud service candidates. Trust Computation using MCDA is carried out by Support Factor Clustering (based on Support Vector Machines and clustering as per CC requirements), Trust Factor Aggregation (functions of factor addition and multiplication are used to aggregate trust factors of services), and Minimum Polyhedron (finds the minimum polyhedron that covers all trust factors in trust coordinate system). To store the result for referencing it later, the Trustworthiness Record Section is utilized. The accuracy of the results is consistent after three-fold verification is carried out by comparing the outcome with feedback, data of revenue, and client growth rate as well as reviews from experts of third-party organizations.

### 3.5 "A Cloud Service Selection Method Based on Trust and User Preference Clustering"

Existing models for Cloud Service Selection do not take into account the similarity in user preferences which seem personalized and unique for each user. To consider the context information and specific user requirements, as well as interaction, requirement personalization and similarity, QoS diversity, complexity and trustworthiness between service providers and users, Y. Wang et al.

propose a Cloud Service Selection method based on Trust and User Preference Clustering which follows a six-step procedure to produce the final measure of trustworthiness. Firstly, all CSPs register to the cloud service registration center. The Cloud Customers (CCs) requirements are sent to the framework as personalized preferences. Condensed hierarchical clustering method is used to cluster requirements and the distance between these clusters is measured as per user service weight preference similarity, user service attribute score similarity, and user evaluation similarity in the third step. Threshold-based hierarchical clustering algorithm clusters repeatedly based on the similarity of preferences until K clusters are present and their distances are measured. The optimal cluster is obtained by assessing the cluster structure degree. The comprehensive trust, sum of direct trust, and recommended trust are evaluated in the next step. Direct trust is calculated with the help of attribute weights (objective weights are calculated using improved entropy weighting method and subjective weights are calculated using AHP), attenuation time, transaction amount, and penalty factor. In order to prevent false reporting of direct trust by malicious users affecting the CSPs reputation in the market, recommended trust (intra-domain recommended trust and extra-domain recommended trust) is also calculated. Recommended trust is the measure between a user and an intra-domain/extr-domain recommender of the CSP. Lastly, CSPs are ranked based on their trustworthiness. Based on the users choice the weights are assigned to direct trust and recommended trust and the final comprehensive trust is computed. This trustworthiness is the deciding factor of the service that is chosen for particular user requests. The improved condensed hierarchical method proves to be better than traditional methods with respect to running time of clustering, iterations taken to cluster, and convergence speed. This method has the lowest Mean Absolute Error and Root Mean Square Error as well as the highest Success Rate for different numbers of transactions and interactions which vouch for the feasibility and effectiveness of the technique and the capability to identify malicious services and feedback.

## 4 Architecture and Design

### 700 4.1 System Overview

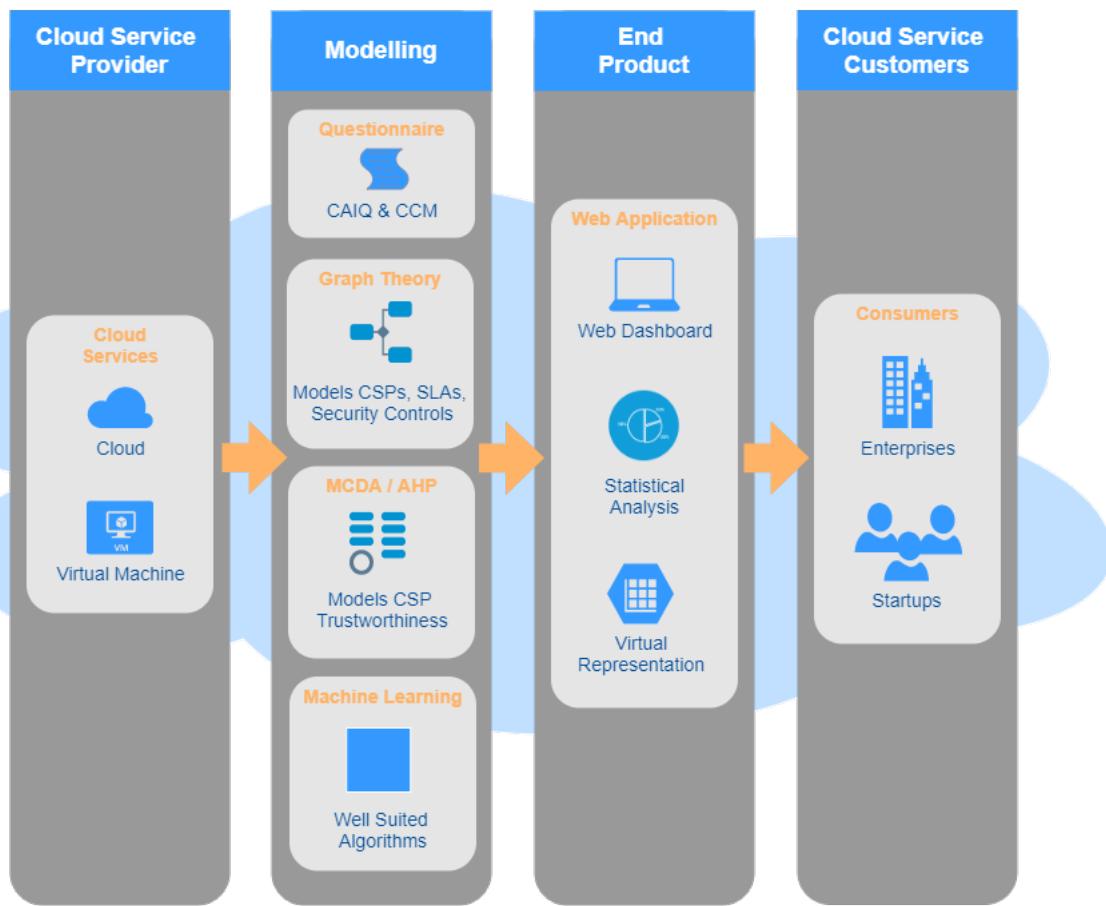


Figure 4.1: User Point-of-View System Diagram

From the end user's point of view, there are four important sectors that are involved in the model,  
705 the Cloud Service Optimizer.

- **Cloud Service Provider:** This is the Cloud Service Provider under scope. AWS, for instance, offers a set of cloud services that the end user utilizes but does not have enough data whether to trust them or not due to the currently existing metrics for establishing trustworthiness not being solid / reliable enough to provide accurate values.
- **Modelling:** This section deals with the product's configuration. It contains several questionnaires that test the standards of these Providers and these answers are mapped using Graph

Theory. We also use historical data generated from JMeter, a performance testing tool to model trustworthiness. AHP and Machine Learning are used to retrieve a Trustworthiness Level based on relative importance of each content type.

715

- **End Product:** It is a system that contains dashboards and visual aids that help the user effectively choose the right CSP service type for their needs. Here, their considerations are met and are included to calculate a result. All the inputs obtained by the user are accurately validated to ensure that the user is only providing the right inputs, are then accumulated into a collective object and are sent to the backend for processing. Once the processing is complete, the results are rendered in the form of graphical structures showcasing statistics. They serve to be visually interactive making it easy for the user to read and comprehend the results generated.

720

- **Consumers:** The consumers extend not only to enterprises but also to startups at large. It may include individuals who are doubtful with respect to migrating their applications to the cloud and might require some additional inputs in the form of accurate trustworthiness quotients.

## 4.2 Software Architecture

### 4.2.1 System Block Diagram

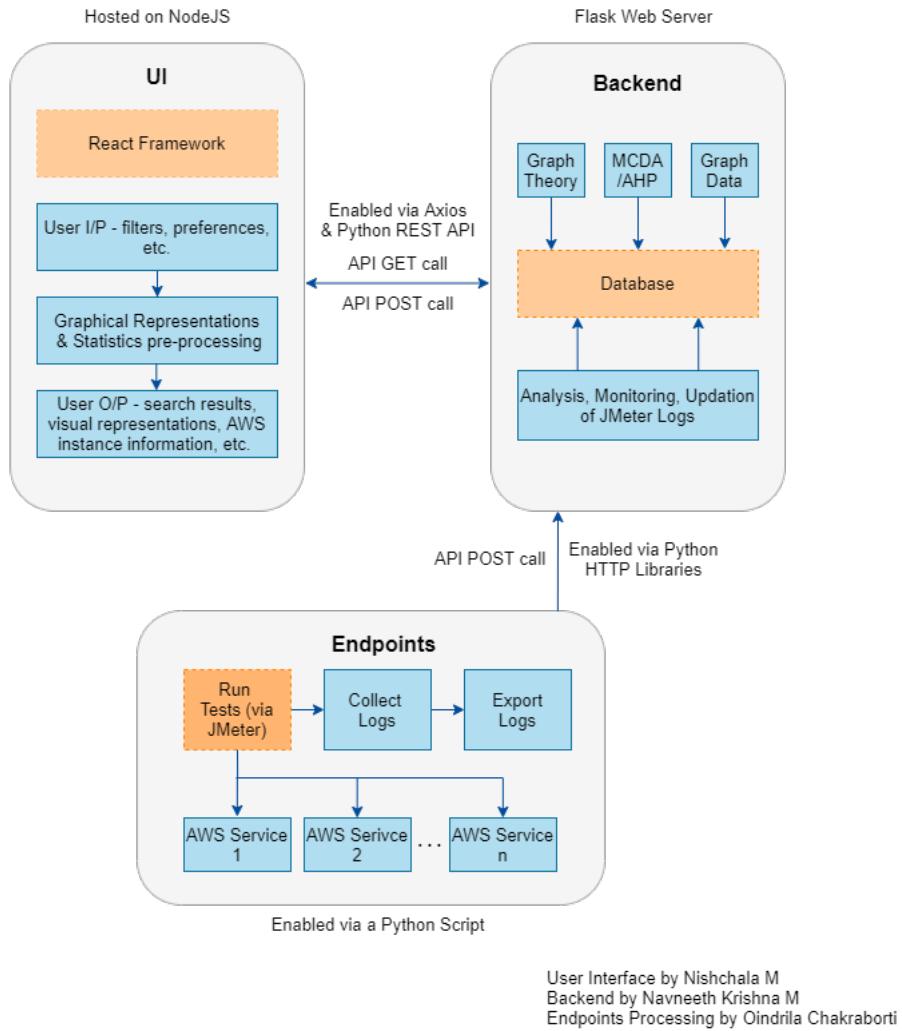


Figure 4.2: End-to-End System Diagram

From a technical perspective, the user interface is a web application hosted on Heroku using React framework. Here, the user input is taken for filters and rankings and after communicating with the backend, an appropriate result is rendered through graphs and charts. Each instance type of a content area is ranked individually along with an Overall result. For services without instance types, like S3, we use Machine Learning to predict the results that can be expected from the service.

Endpoints consist of research that is related to running tests and collecting logs on these various instances of Cloud Services. Here, JMeter is used as a tool along with various plugins like Perfmon and JDBC Driver to get reliable data and we perform multiple levels and intensities of

740 testing. We have collected about 50L rows of logging data that is built as the fundamental training set in the project. We test various content areas like CPU and Memory utilisation Latency, Throughput, Availability,etc. and these are processed and sent to the backend via a Python script.

745 Next, the backend deals with several components that interact with the front-end as well as the logging endpoints. Here, the user input is processed via Analytical Hierarchy Process, a decision-making algorithm used when multiple criterias must be ranked relatively. The backend is a Flask Python backend hosted on Heroku that uses REST APIs to communicate. The database used is a relational MySQL server from DigitalOcean. Logs provide cloud data that are constantly received, analyzed, and updated to the database. Graph related data is sent to the 750 front-end when the user requests such data. For ML, multiple linear regression is used.

#### 4.2.2 Data Flow Diagram

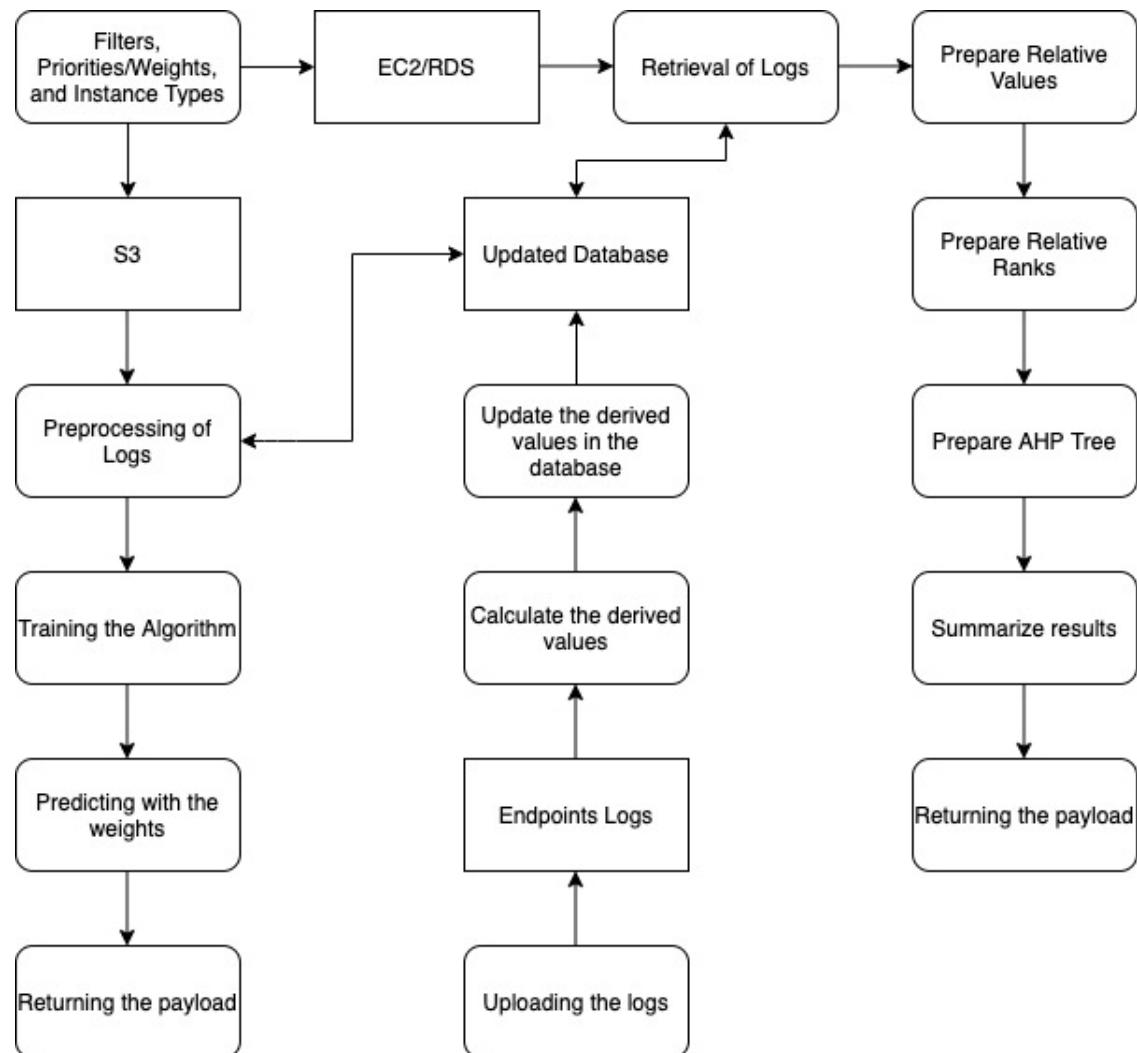


Figure 4.3: Data Flow Diagram

In the above Data Flow Diagram, we obtain an overall understanding of the flow of information

755 from each point in the system. To elaborate on the flow step-by-step, here is an algorithm:

1. Obtain the Filters, Priorities, and Instance Types
2. Navigate according to the instance type
3. If EC2 or RDS, retrieve the logs
4. Then prepare relative values and ranks
- 760 5. Prepare the AHP tree and summarize results and return payload
6. If S3, Preprocess the logs
7. Train the Multiple Linear Regression Algorithm
8. Fit and Predict for the input weights and return payload
9. For updating the database, upload the logs
- 765 10. Calculate the derived values
11. Store the derived values in the database

## 5 Implementation

### 5.1 Implementation Platform

#### 5.1.1 Hardware

- 770 • Processor: Intel Octal Core

- RAM:8 GB

- GPU:NVIDIA GTX1080

#### 5.1.2 Software

- Operating System: Windows 10 (64 bit) and MacOS

- 775 • Software Used: Flask, ReactJS, NodeJS, Heroku, JMeter, MySQLWorkbench, DigitalOcean

- Programming Languages: Python, JavaScript, SQL, Java

- Server: NodeJS, Gunicorn

## 5.2 Implementation Details

### 5.2.1 Organization of implementation files

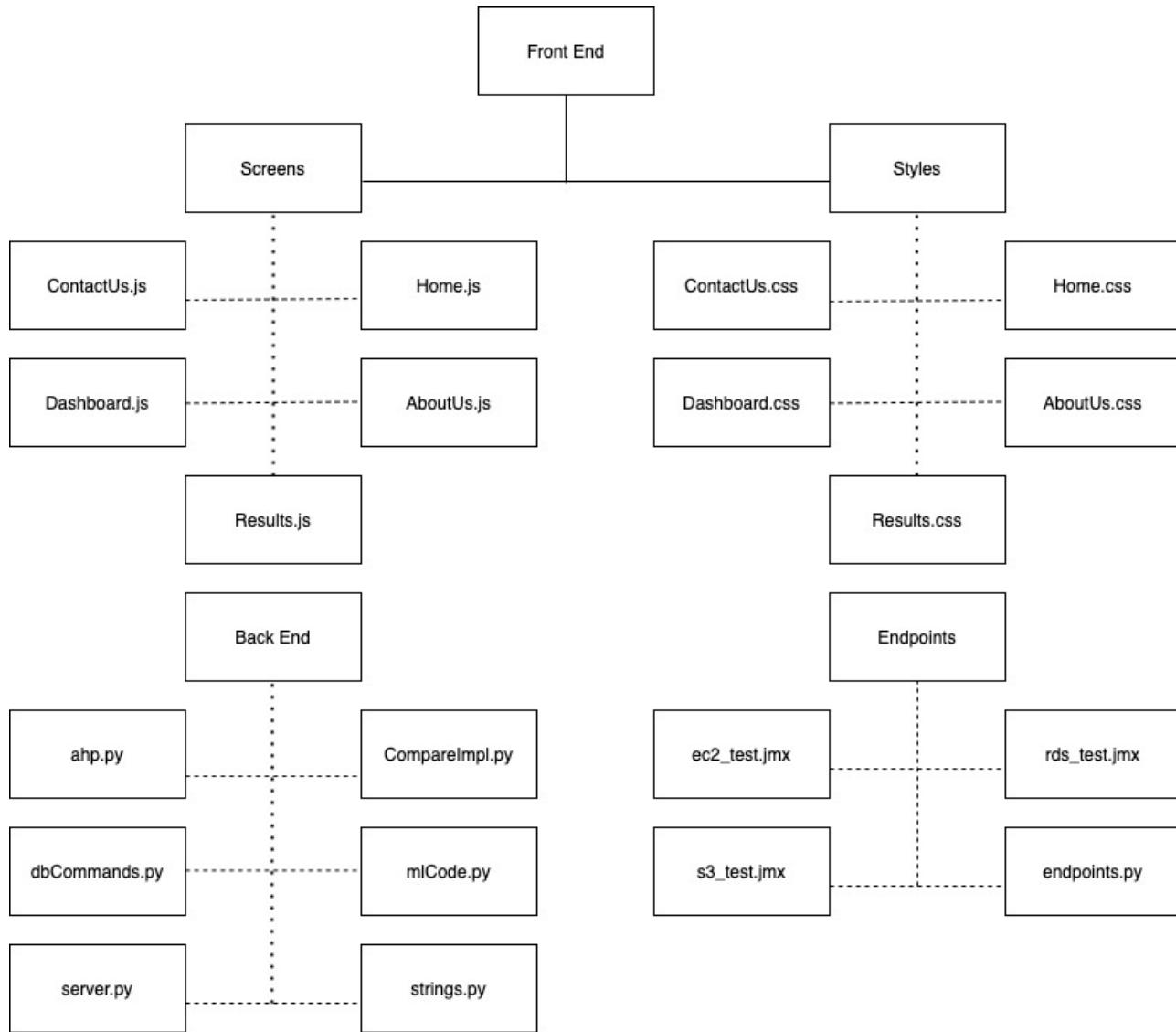


Figure 5.1: Directory Structure

The file structure of each component of the project is as seen above. In the front-end, we have two primary directories for Screens and Styles. Under them, we have the respective JavaScript and Cascading Style Sheets code files for the five pages in the UI: Home, About Us, Contact Us, Dashboard, and Results. In the Back-end of the project, we have ahp.py that runs the main AHP algorithm, CompareImpl.py which contains the implementation of generation of Comparison Matrices, dbCommands.py to facilitate interactions with the database which is a relational database hosted on DigitalOcean, server.py that handles all routing and immediate actions performed by the server, and strings.py that deals with the construction of the payload text and returns the string. The Endpoints contain three testing files that are responsible for the collection of JMeter

logs and an endpoints.py python script to upload all the logs in the local system to the Back-end.

### 5.2.2 AHP Model

As described in the structure of AHP, the algorithm to preprocess the relative ranks and relative values in order to determine the AHP tree is as shown below:

795

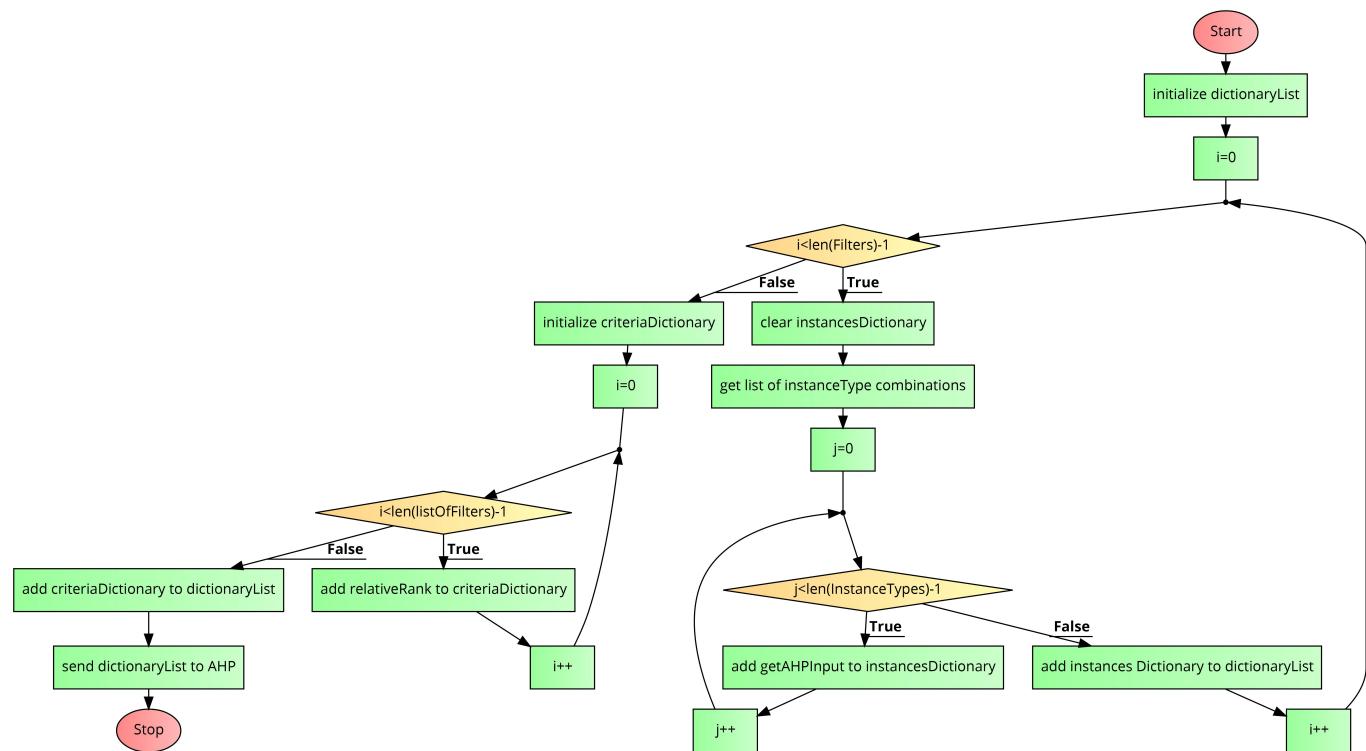


Figure 5.2: Preprocessing Algorithm for AHP

The dictionary list contains a list of inputs for both the levels of AHP.

800

- First, to determine the relative values, we run through the list of input instance types.
- Using the list of instance types, we determine all possible combinations to define the relative value.
- For each combination, we process and obtain a relative value.
- This relative value is determined by the processed output of JMeter tests.
- We add the relative value to the instancesDictionary.
- We repeat for all instance types.

805

- We add the relative values list (instancesDictionary) to the dictionary list.

Similarly, to determine the criteria ranking:

- 810
- First, to determine the relative rankings, we run through the list of input content areas.
  - Using the list of content areas, we determine all possible combinations to define the relative ranking.
  - For each combination, we process and obtain a relative ranking.
  - This relative ranking is obtained via The Inverse Law.
- 815
- We add the relative rank to the criteriaDictionary.
  - We repeat for all content areas.
  - We add the relative ranks list (criteriaDictionary) to the dictionary list.

Once we have established the overall weights, we process them via a payload as follows:

```

821 {
2   "payload": {
3     "Connect": {
4       "data": {
5         "db.t2.medium": 0.332,
826       "db.t2.micro": 0.332,
7       "db.t2.small": 0.336
8     },
9       "displayName": "Connection Time",
10      "filterCompare": {
831        "db.t2.medium > db.t2.medium": 0.0,
12        "db.t2.small > db.t2.medium": 1.2048192771084347
13      }
14    },
15      "Latency": {
846        "data": {
17        "db.t2.medium": 0.332,
18        "db.t2.micro": 0.332,
19        "db.t2.small": 0.336
20      },
851      "displayName": "Latency",

```

```
22 "filterCompare": {
23   "db.t2.medium > db.t2.medium": 0.0,
24   "db.t2.small > db.t2.medium": 1.2048192771084347
25 }
26 },
27 "Throughput": {
28   "data": {
29     "db.t2.medium": 0.335,
30     "db.t2.micro": 0.333,
31     "db.t2.small": 0.332
32   },
33   "displayName": "Throughput",
34   "filterCompare": {
35     "db.t2.medium > db.t2.micro": 0.600600600600601,
36     "db.t2.medium > db.t2.small": 0.9036144578313261,
37     "db.t2.micro > db.t2.small": 0.3012048192771087
38   }
39 },
40 "elapsed": {
41   "data": {
42     "db.t2.medium": 0.332,
43     "db.t2.micro": 0.332,
44     "db.t2.small": 0.336
45   },
46   "displayName": "Elapsed Time",
47   "filterCompare": {
48     "db.t2.medium > db.t2.medium": 0.0,
49     "db.t2.small > db.t2.medium": 1.2048192771084347
50   }
51 },
52 "errorRate": {
53   "data": {
54     "db.t2.medium": 0.198,
55     "db.t2.micro": 0.398,
56     "db.t2.small": 0.404
57   },
58   "displayName": "Error Rate",
59   "filterCompare": {
60     "db.t2.micro > db.t2.medium": 101.010101010101,
```

```

61 "db.t2.small > db.t2.medium": 104.04040404040404,
62 "db.t2.small > db.t2.micro": 1.5075376884422123
63 }
64 },
65 "overall": {
66   "data": {
67     "db.t2.medium": 0.317,
68     "db.t2.micro": 0.339,
69     "db.t2.small": 0.344
70   },
71   "displayName": "Overall",
72   "filterCompare": {
73     "db.t2.micro > db.t2.medium": 6.9400630914826555,
74     "db.t2.small > db.t2.medium": 8.517350157728696,
75     "db.t2.small > db.t2.micro": 1.4749262536873005
76   }
77 },
78   "stdDev": {
79     "data": {
80       "db.t2.medium": 0.328,
81       "db.t2.micro": 0.332,
82       "db.t2.small": 0.34
83     },
84     "displayName": "Consistency",
85     "filterCompare": {
86       "db.t2.micro > db.t2.medium": 1.2195121951219523,
87       "db.t2.small > db.t2.medium": 3.658536585365857,
88       "db.t2.small > db.t2.micro": 2.4096385542168695
89     }
90   }
91 },
92   "processingNumber": "pBjcFAB0gD",
93   "status": "success"
94 }

```

915 As observed above, the filterCompare parameter in each of the content areas in the payload contains data suitable to represent in graphical format. This is a sample processing output sent

to the front-end via the front-end/get-results API call.

### 5.2.3 Front-end component

The web application uses two packages for rendering graphical statistics. Npm provides several packages to render various charts ranging from line graphs, bar charts, pie charts, etc. The two packages used for modelling this section are - chart.js and react-chartjs-2 (the latter being a wrapper for the former). In order to provide variety and the opportunity of better understanding for the end-user, both pie charts and bar charts are used. The aforementioned packages can be installed using simple commands such as:

```
925 npm install chart.js --save npm install react-chartjs-2 --save
```

Both the Bar and Doughnut (Pie Chart) are handy libraries that can be fed various input parameters like dataset, legend, title, and various other options. In the flow of result display, the JSON object received from the Back-end via Axios API calls is broken down into several key-value pairs (due to the nested structure of the JSON object received). The key value pairs are accessed one by one, and converted into a JavaScript object and fed into the data parameter for each of these charts. The data will then be rendered in graphical form on the User Interface. The graphs rendered are very interactive in nature and can be collapsed and re-rendered back again.

Listed below are a few snapshots of the User-Interface:

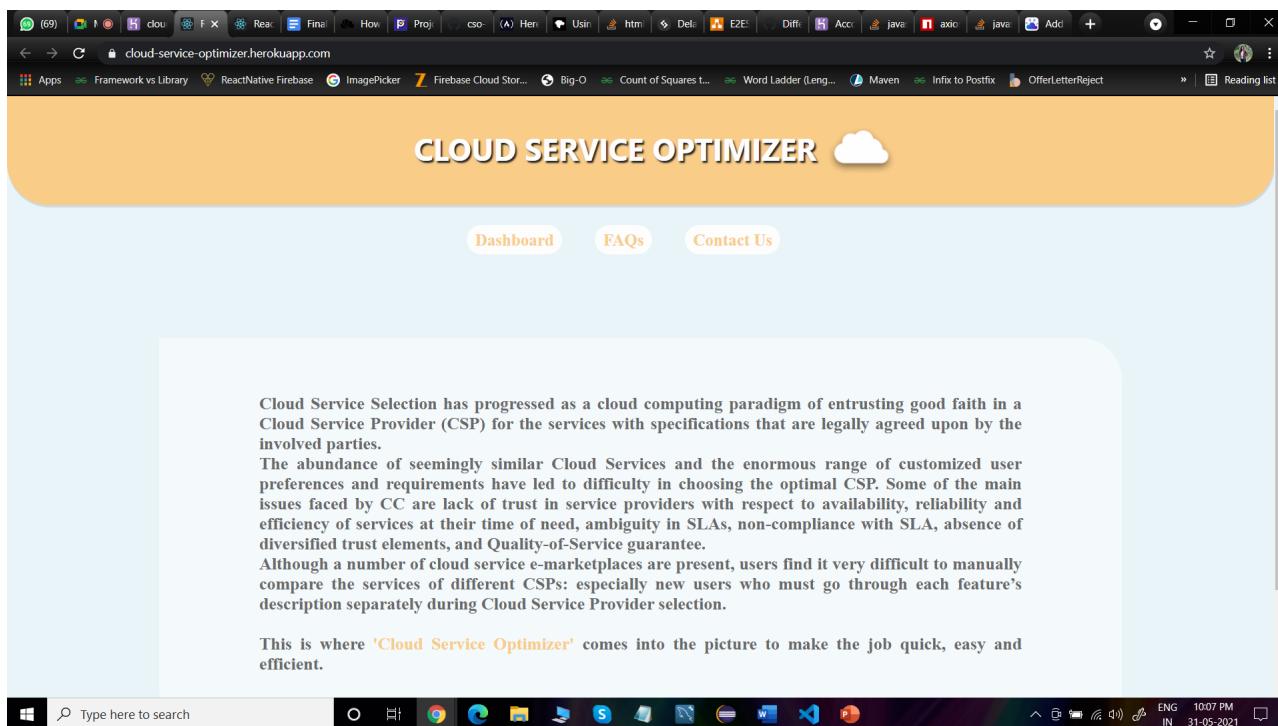


Figure 5.3: Home Page

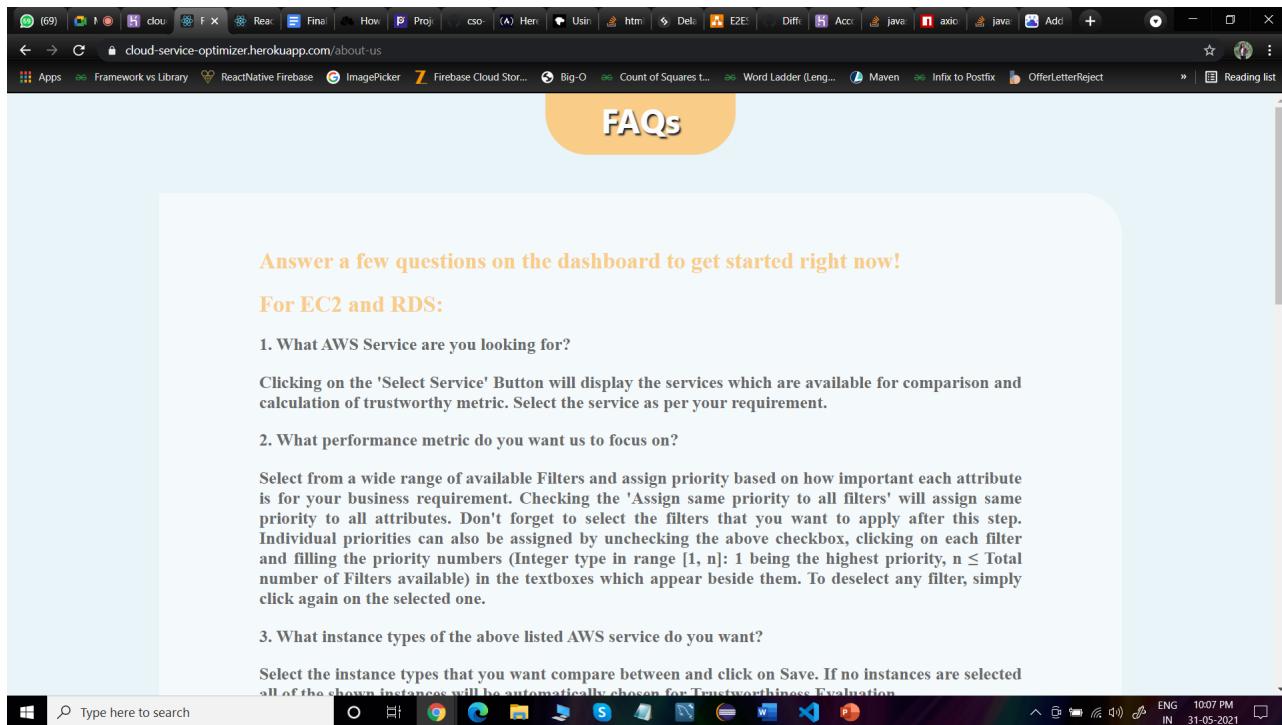


Figure 5.4: FAQs Page

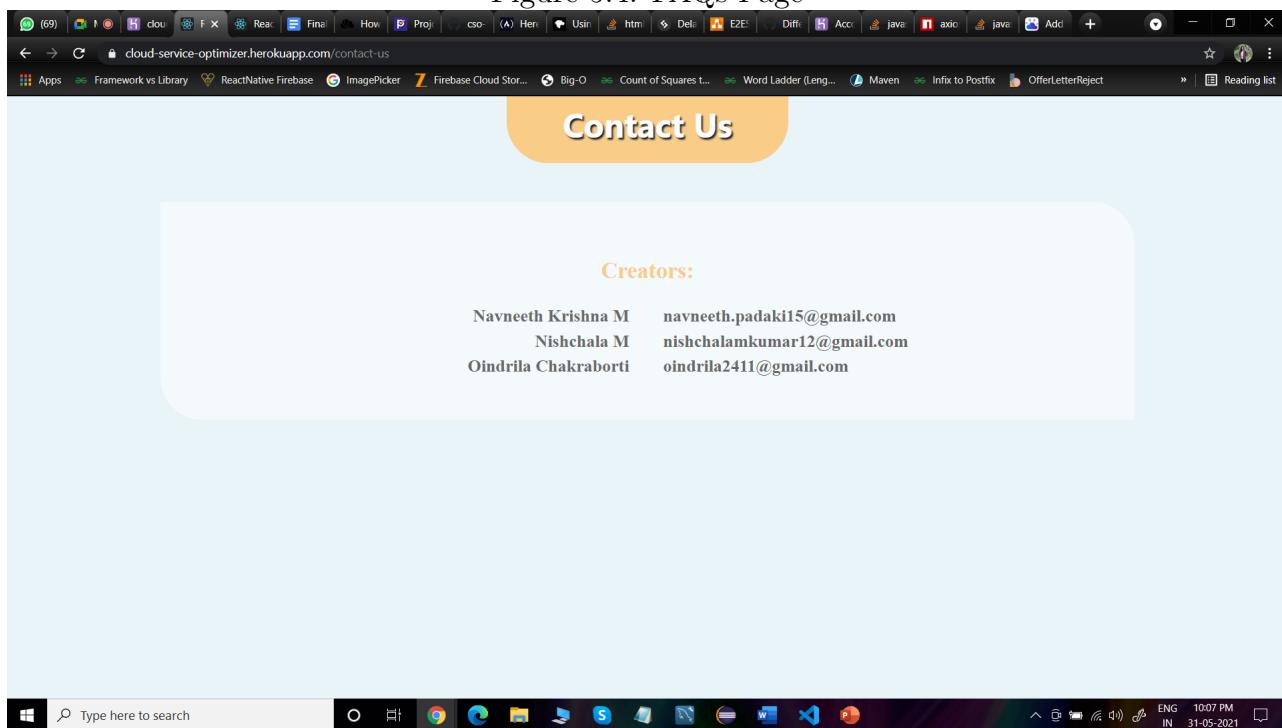
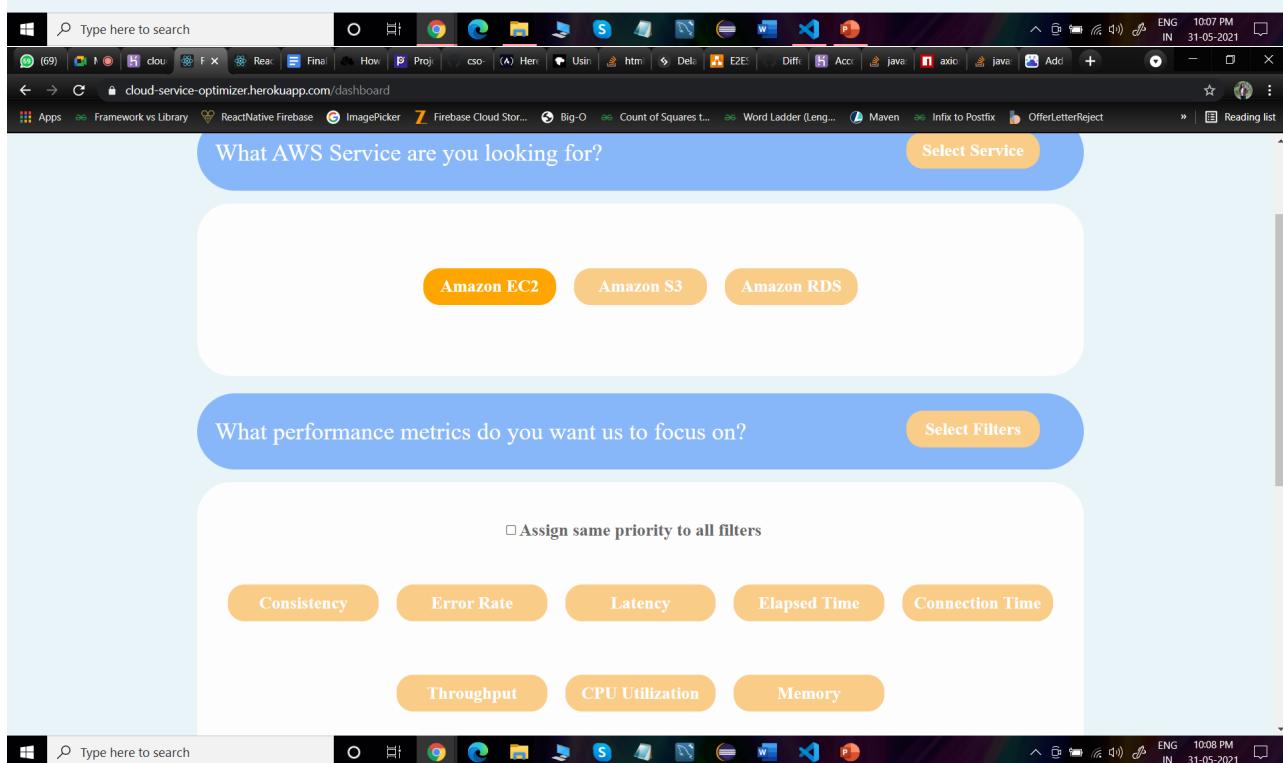
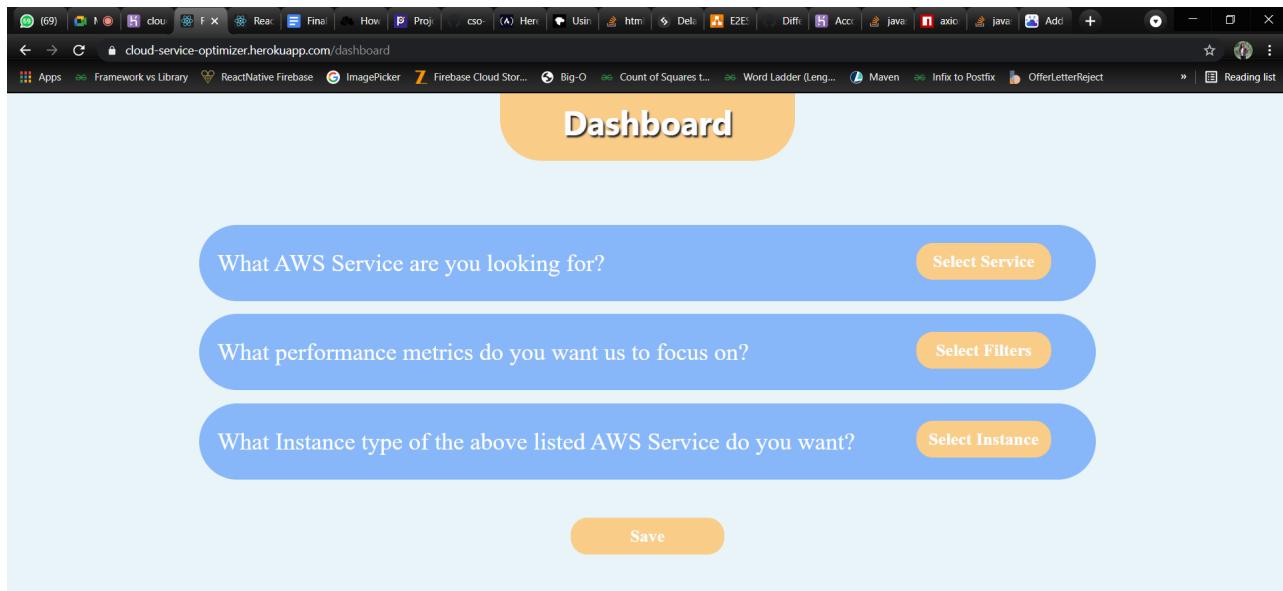


Figure 5.5: Contact Us Page



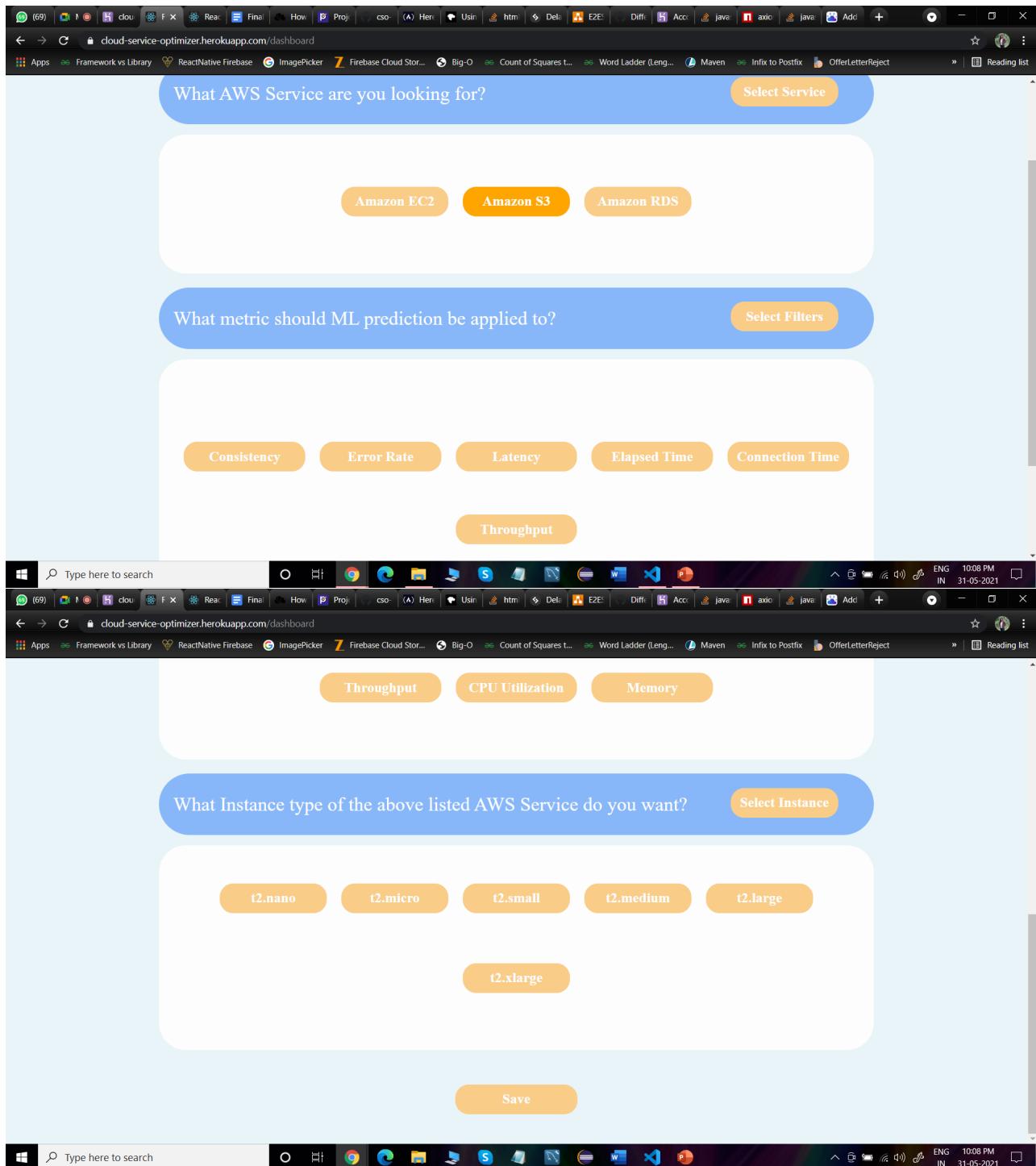


Figure 5.6-5.9: Dashboard Page

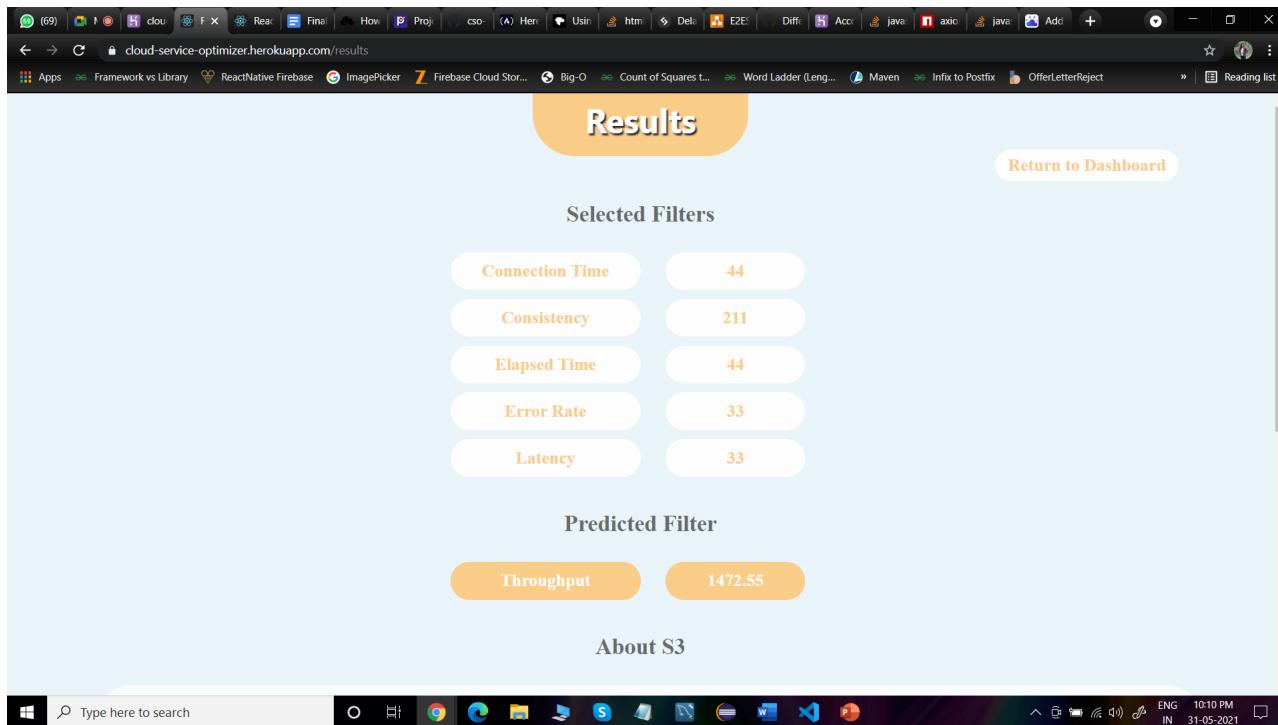
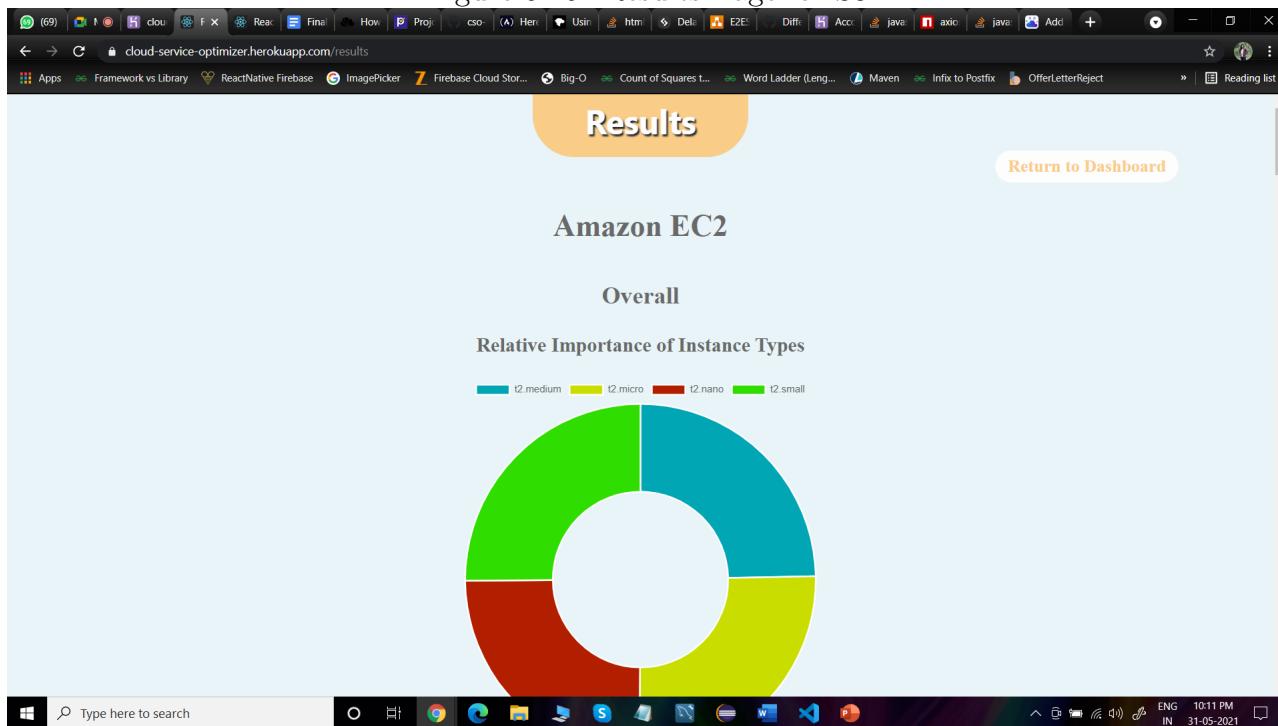


Figure 5.10: Results Page for S3



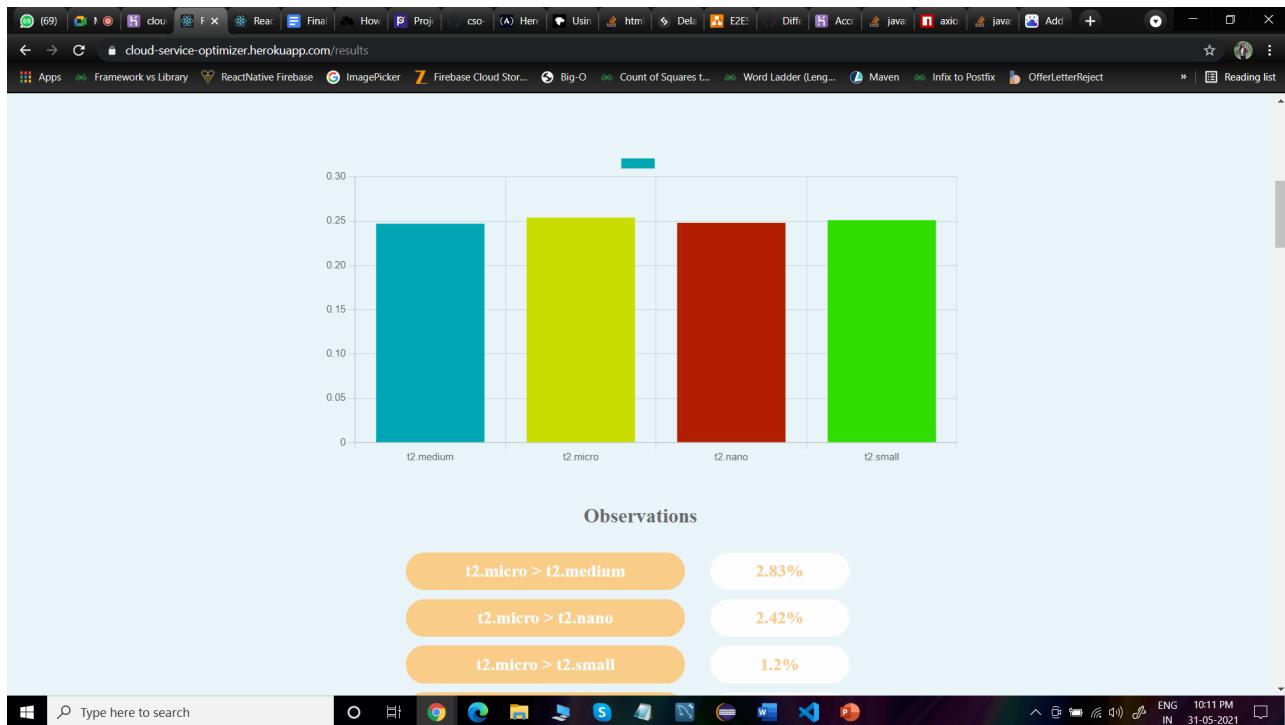


Figure 5.11-5.12: Results Page for EC2/RDS

#### 950 5.2.4 JMeter Performance Testing

CPU and Memory utilisation are important parameters which indicate the health of a server. CPU utilization is a server's usage of processing resources i.e. CPU, or the amount of work handled by CPU in real time. CPU usage varies depending on the task which the CPU is handling. Some computation intensive tasks require more CPU time while others might need less because of swapping of Jobs to other resources like I/O. Memory utilisation is the amount of RAM in use by the running applications. Free Memory is always needed for tasks in execution to exhibit better responsiveness. Hence, ideally Memory usage should be as low as possible although, the minimum required amount of memory is always used for running applications.

In local systems, both CPU and Memory usage can be easily analysed on Task Manager, however to assess these attributes on remote servers such as EC2 instance servers, Perfmon Plugin must be used in addition to the Jmeter tool in local system as well as the Perfmon Server Agent which must be installed on the remote server. For the plugin to work on a system - local or remote - Java Development Kit/ Java Runtime Environment must be installed.

965 The Client agent can be easily installed from the official website, unzipped and copied into the plugins folder of Jmeter. However, to install Server Agent on EC2 instance user data must be provided which executes at the time of System boot and carries out necessary

installations. The commands are as follows (for LINUX instance):

```
970 sudo yum install -y java-1.8.0-openjdk.x86_64  
wget https://github.com/undera/perfmon-agent/releases/download/  
2.2.3/ServerAgent-2.2.3.zip  
unzip ServerAgent-2.2.3.zip  
cd ServerAgent-2.2.3  
975 sh startAgent.sh
```

Once the Server Agent starts and it is integrated with Client Perfmon in local Jmeter, communication flows between these agents and various performance parameters can be logged like CPU, Memory, Disk I/O, Swap, TCP, Network I/O, etc. can be tested and logged in .csv  
980 format as well as real time graphical representation is shown for the attributes.

### 5.2.5 Database Structure

The database being used is a MySQL relational database that is hosted on DigitalOcean. The specifications of the hosted database cluster are: 1 GB RAM / 1vCPU / 10 GB Disk / Primary only / MySQL 8. There are 18 tables in the database and are as defined:

- 985 1. ahpResults: A table used to log all the output results of AHP.
2. APILogs: A table used to store all the API calls and log them.
3. derivedCalculationLogs: A table used to keep track of all updates to derived values.
4. derivedValues: A table used to store all the derived values of each content areas for each instance type.
- 990 5. ec2logs: A table used to store all the log files of EC2 service type.
6. ec2performance: A table used to store all the performance metrics of EC2 service type.
7. ec2summary: A table used to store the summary results of logging while testing an EC2 service type.
8. endpointAccess: A table used to define credentials for permitted logging from registered endpoints.
- 995 9. endpointData: A table that contains information about the logging user.

10. endpointLogs: A table that stores the list of files sent by the user to the database.

11. loginLogs: A table that stores all login information for each login by a user.

12. processingRequests: A table that stores the processing requests generated from the front-end  
1000 and officially stored in the database.

13. saveec2: A table used to store retrieved average values of each instance type of EC2 uploaded  
for all logging sessions.

14. saves3: A table used to store retrieved average values of each S3 logs file uploaded for all  
logging sessions.

1005 15. saverds: A table used to store retrieved average values of each instance type of RDS uploaded  
for all logging sessions.

16. sessionDetails: A table used to store a particular login session details.

17. staticContent: A table used to retrieve static data from the Back-end to display in the front-  
end.

1010 18. taskStatus: A table used to store the status of a processing number as Pending or Complete.

### 5.2.6 Machine Learning

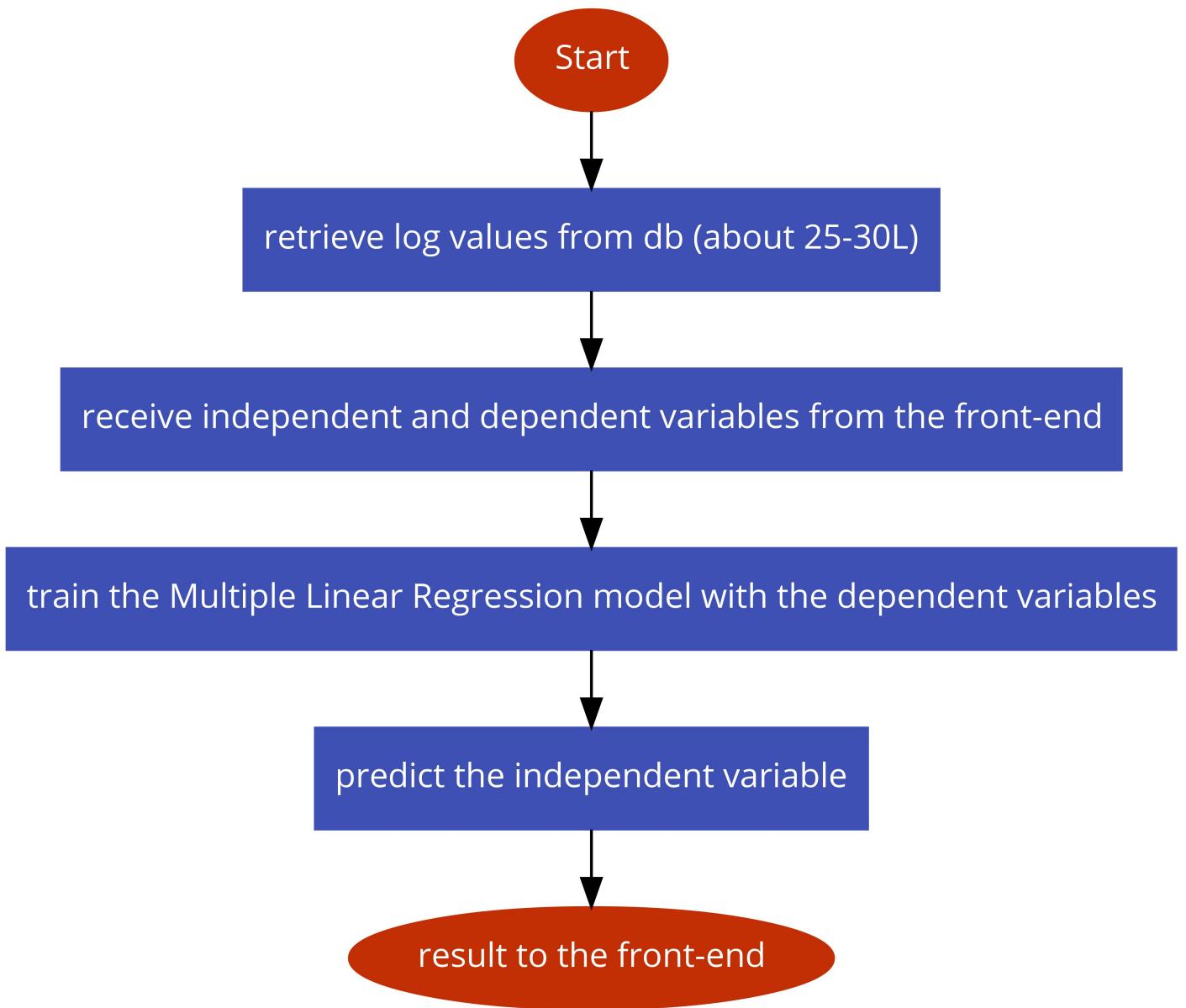


Figure 5.3: Multiple Linear Regression Algorithm

1015 The Machine Learning component is fundamentally a Multiple Linear Regression algorithm that trains about 25-30L log entries that are related to AWS S3. These form rows to the dataset and are split into training and testing just to test the accuracy of the predictability of the model. Once fit and tested, we saw a 90-95 percent accuracy as we gradually trained the model with more and more logs from JMeter. These logs help predict the independent variable using the dependent variables as mentioned in the input. Finally, we return the result to the user interface with a payload.

### 5.2.7 Back-end Routes

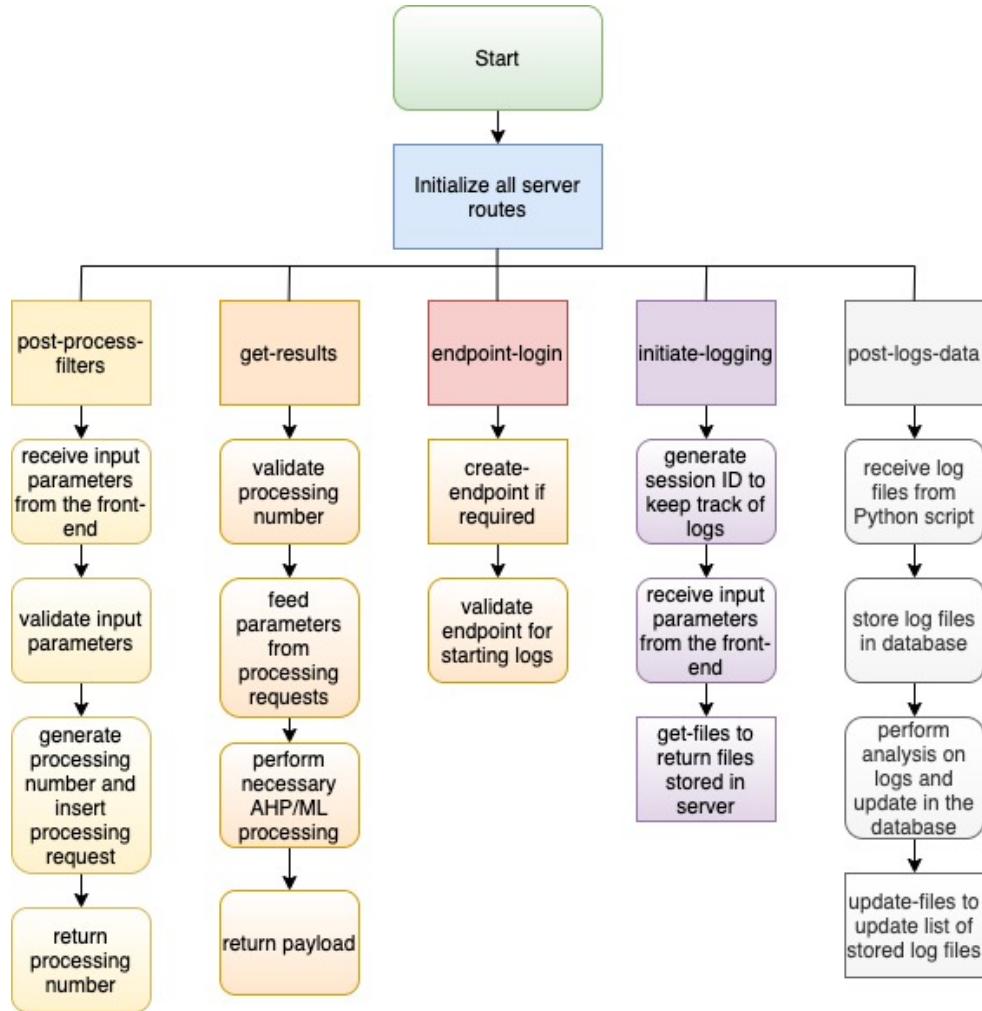


Figure 5.4: Back-end Routes

1025 In the Back-end, there are five primary routes with sub-routes that handle both front-end and endpoints API calls.

1. post-process-filters: This route sends the user request with filters, instance types (if any), service type, and priorities. Once these are received and validated, a processing number is mapped to this request and is returned to the user interface.
2. get-results: This route matches an entered processing number and performs necessary AHP/ML processing. Once the processing is done, a payload is returned to the user interface.
3. endpoint-login: For a registered device to begin uploading logs to the Back-end, this route handles the requests. Login/Signup is handled and the user can register their device.
4. Initiate-logging: For a successful login to the system, the Back-end identifies the files that

were previously uploaded and returns to the endpoint through this route. Once it is retrieved, only new files are logged to the Back-end.

- 1035 5. post-logs-data: The logs to be uploaded are processed by the endpoints and are sent via this route. Once all the new files are uploaded, the files list is updated so that no same logs are sent to the Back-end again.

### 1040 5.3 Datasets Collection

Various instances of EC2 and RDS and S3 are tested against several parameters with varying configuration settings for each run of the test script like 50 - 200 User Requests per second, 1 - 3600s of ramp-up time (time interval in which all user requests are invoked non-uniformly) , 30minutes - 1 hour 30 minutes Duration of tests, Loopcount (number of times the same loop of 1045 Threads (User Requests) will be invoked. The following steps are followed to create a successful Testing Script :

For EC2 :

- Launch new instances of EC2 from AWS Console and specify the below user data and add Security Roles of SSH (default settings), HTTP (Port 80) and TCP (port 4444).

```
1050
1 #!/bin/bash
2 yum update -y
3 yum install -y docker
4 service docker start
5 docker pull bkimminich/juice-shop
6 docker run -d -p 80:3000 bkimminich/juice-shop
7 sudo yum install -y java-1.8.0-openjdk.x86_64
8 wget https://github.com/undera/perfmon-agent/releases/download/2.2.3/
9     ServerAgent-2.2.3.zip
10 unzip ServerAgent-2.2.3.zip
11 cd ServerAgent-2.2.3
12 sh startAgent.sh
```

1065 The OWASP- Juice Shop is used as the sample website.

- Create a new test project by clicking on File, then New.

- Add a Thread Group, enter necessary details about number of requests, ramp-up period, duration, etc.
- Add an HTTP Request Sampler under the above Thread Group. Specify the URL and path of the website which has to be tested. Select GET in the drop down menu in HTTP Request. Add any parameters if necessary.
- Add View Results in a Table, Summary Report, and Perfmon Metrics Collector Listeners in the sampler.
- The three different listeners will save logs for different parameters.

1075 For RDS :

- Launch RDS instances from AWS Console and grant them public access during configuration before launch.
- Launch an EC2 instance to connect to the databases. Follow the previous steps to launch the instance. Connect to the instance using SSH (on CLI or Putty manager depending on OS in the local system). Execute the following commands:

```

1 sudo yum update
2 sudo yum install mysql
3 mysql -h <DB_Host> -P <Port> -u <Username> -p <Password>
4
5 create database RDS_Test;
6
7 use RDS_Test
8
9 create table student (USN varchar(10),
10 StudentName varchar(30),
11 Age int,
12 Branch varchar(5),
13 Gender varchar(1),
14 Semester int,
15 PRIMARY KEY(USN));
16
17 insert into student values ('1DS17CS734', 'Tanusha', 22, 'CSE', 'F', 8);
18

```

19 | (Populate the table using sample values)

- Download and add the MySQL Connector file in the Lib folder inside the Apache JMeter directory.
- In Jmeter, create a new project.

- 1105
- Add JDBC Configuration file and enter the Database hostname, port, username and password. Mention JDBC Driver as the Driver File in configuration.
  - Add a Thread Group with all necessary details and add JDBC Request Sampler with the SQL query that has to be invoked.

- 1110
- Add View Results in table and Summary Report Listeners.

For S3:

- Create the same HTTP Request Test on Jmeter as was done for EC2 instances.
- Create a new bucket and add the source files of the website.
- The index.html file is generally the opening file of the website.
- Click on the opening file, copy the URL and enter it on HTTP Request Sampler in Jmeter.
- Add View Results in table and Summary Report Listeners.

All the listeners together collect logs (used for training the AHP model in Back-end) for the following attributes:

- 1120
- CPU Utilisation: One of the most important Health Parameters is CPU Usage which denotes the amount of work handled by CPU by running or background applications on the Host Server.
  - Memory Utilisation: Memory Utilisation is the RAM occupied by the applications at any given point in time on the Host Server.
  - Latency: Latency denotes the the amount of time the user request waits for the resources to be delivered through HTTP or TCP Protocols

- Connection Time: This attribute denotes the amount of time required to establish a connection between Client and Server in TCP connection.
- Response Time: This value indicates the amount of time required before the first response is received from the Host Server.
- Standard Deviation: The consistency of the server is denoted by this attribute, it shows the deviation of Response Time of individual requests from the average Response Time.
- Error Percentage: This value denotes the percentage of all threads which do not connect with the host. In other words, the percentage of times when the resource is inaccessible to the Client.

## 1135 6 Testing

In our project, we have imbibed the principles of Functional Testing while carrying out each phases of Acceptance, System, Unit, and Integration Testing.

In the Unit Testing phase, while working with individual components, these were the timelines:

- Check functionality of graphical representation.
- Check the unit that retrieves graphs from the Back-end and displays the same.
- Check the functionality of the various sections. involved in the UI that catered to static design templates.
- Check the functionality of the servers API routing.
- Check the functionality of the Back-ends data movement between one structure to another and also verify data handling.
- Check the integration of the database elements with the end-to-end system.
- Verify necessary primary key and foreign key requirements and ensure creation of the required tables.
- Conduct logging activities with ample testing resources within controlled time intervals.
- Extrapolate logs analysis via Endpoints and assimilate required data for processing.

In the Integration Testing, these were the goals defined:

- Describe the connectivity between both the ends of the front-end/post-process-filters API call
- Ensure smooth upload of log files between both the ends of the endpoints.post-logs-data API call
- Sustain network connectivity between yielding the processing results between both the ends of the front-end/get-results API call

In the System Testing, a thorough end-to-end round of testing was carried out for 5-7 days to fix routine bugs and common mistakes. We described scenarios and impasse situations and rigorously tested the applications limits. Finally, in the Acceptance Testing, we formulated all

1160 memory, storage, hosting, security, and data reliability factors once the application was released to

the world at large. Only after an acceptable range of results were obtained was the entire project deployed to Heroku. We configured dynos for memory, expanded database requirements, made suitable tracks for migration purposes, and prepared routine backups and integration checks on the stored data.

## 1165 7 Experimentation and Results

When we tested the requests for Analytical Hierarchy Process, we observed the following:

- Users are more interested in Overall rankings than individual areas.
- When asked to prioritize content areas, throughput was the most important content area among available options.
- 1170 • Since at least 2 filters had to be chosen for processing, users were sometimes spoilt for choice in choosing the best two content areas if not more.
- While consideration of instance types available as an option was deemed useful, users mostly preferred a default AHP ranking of all the content areas.
- 1175 • The processing for AHP with a two-level tree is a complicated procedure. So, with more and more nodes loaded into the tree, the processing increases by two powers.
- Each time user preferences are sent, traversal through the list of combinations is tedious and expensive. However, it is convenient to transact stored data of all filters with one database command and consume values as required by the list of combinations.
- 1180 • As future scope, we seek to derive all values in one search and add relative values by searching through a stored list. Currently, we are searching for each value in the database. It is computation-intensive and time-intensive to individually run SQL SELECT statements to fetch data monotonously.

While observing the results of Multiple Linear Regression in the model, we could draw the following observations and results:

- 1185 • Storing 28-30 L logs and counting is not feasible to happen in real-time.
- To track such huge data through unique numbers is also difficult thus rendering primary keys to be ineffective.
- Training huge datasets in limited time is a herculean task. However, there are batch processing and Map-Reduce solutions in the mix.
- 1190 • An alternative would be to run the training every time after the logs are uploaded and to save and replace the regression model in a stored file.

- With this alternative, preprocessing and training may not occur while user requests for processing.
- Many times, the memory usage exceeded 500 MB just from retrieval and processing of 30 L logs every time an S3 request was made.
- To avoid such leaks, we moved the training component to the endpoints and made the rendering of the model quite feasible.
- The time gap before timeout for a HTTP GET/POST request to receive a response is 30 seconds. Within this timeframe, it proved quite challenging to include training, testing, and prediction of the ML model for a user request.
- Once the training was complete, we returned a payload of the predicted result in graph-compatible format for the user interface.

**Endpoints Results** The logs collected gave insights about the performance of different AWS Services and Instances being subjected to different factors which might affect them like Timestamp, Type of computing - storage intensive, computation intensive, etc.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
-	i-0832dd07599548626	Running	t2.micro	2/2 checks passed	No alarms	+ ap-south-1a	ec2-3-108-41-250.ap-s...	3.108.41.250
-	i-0a73cc87b648a27b	Running	t2.small	2/2 checks passed	No alarms	+ ap-south-1a	ec2-13-234-232-37.ap...	13.234.232.37
-	i-01becb4894e9a43a5	Running	t2.nano	2/2 checks passed	No alarms	+ ap-south-1a	ec2-13-126-114-7.ap-s...	13.126.114.7
-	i-03967392a39e2af93	Running	t2.xlarge	2/2 checks passed	No alarms	+ ap-south-1a	ec2-65-2-31-151.ap-so...	65.2.31.151
-	i-d549d29f1b677d72	Running	t2.medium	2/2 checks passed	No alarms	+ ap-south-1b	ec2-15-207-112-245.ap...	15.207.112.245
-	i-04e601a9902343434	Running	t2.large	2/2 checks passed	No alarms	+ ap-south-1b	ec2-52-66-145-121.ap...	52.66.145.121

Figure 7.1: EC2 instances are launched with Perfmon Server Agent installed

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity
test-database	Instance	MySQL Community	ap-south-1a	db.t2.micro	Available	2.41%	0 Connect
test-database-medium	Instance	MySQL Community	ap-south-1a	db.t2.medium	Available	1.67%	0 Sessions
test-database-small	Instance	MySQL Community	ap-south-1b	db.t2.small	Available	2.67%	0 Connect

Figure 7.2: RDS instances launched

Each RDS Database instance contains a Student Table (with Attributes: USN, Student Name, Branch, Gender) populated with sample values. This table is queried while testing on Jmeter.

1215 S3 Bucket creation:

S3 bucket is created with boilerplate code of a website. Any sample website can be used. Public access must be given for the bucket to be tested externally through Jmeter.

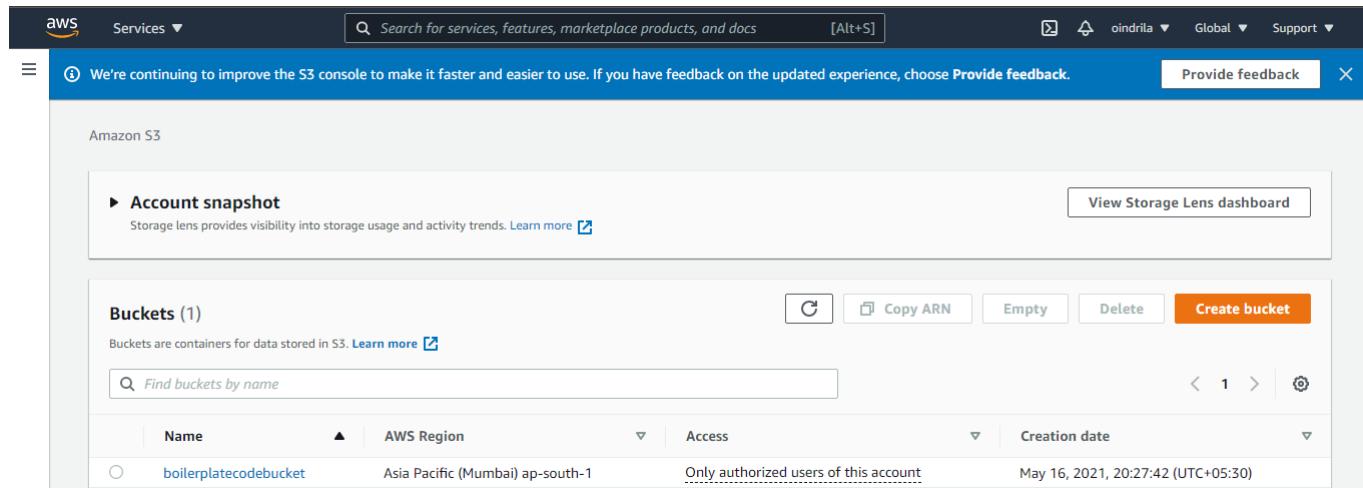


Figure 7.3: S3 Bucket

The public URL of S3 is given as the URL of the website in Jmeters HTTP Request Sampler.  
JMeter:

Testing EC2 instances:

1225 In the test shown below, the following configurations are used :

- Number of Threads : 100
- Ramp-up period : 1s
- Duration : 3600s
- Startup Delay : 1s
- HTTP Request Type : GET Request
- Hostname : ec2-3-108-41-250.ap-south-1.compute.amazonaws.com (for micro, hostname will vary from instance to instance)

- Path : /#

- 1235
- In View results in table listener, File name can be mentioned in .csv or .txt format to save the results in that file

- Summary Report can be saved after the test is complete by clicking on Save Data at the bottom of the window.

- Perfmon Port : 3306

- Hostname/ IP : 3.108.41.250 (for this specific instance)

- 1240
- Mention the metrics to be computed in Perfmon Listener (here, CPU and Memory)

Running ec2.test.jmx :

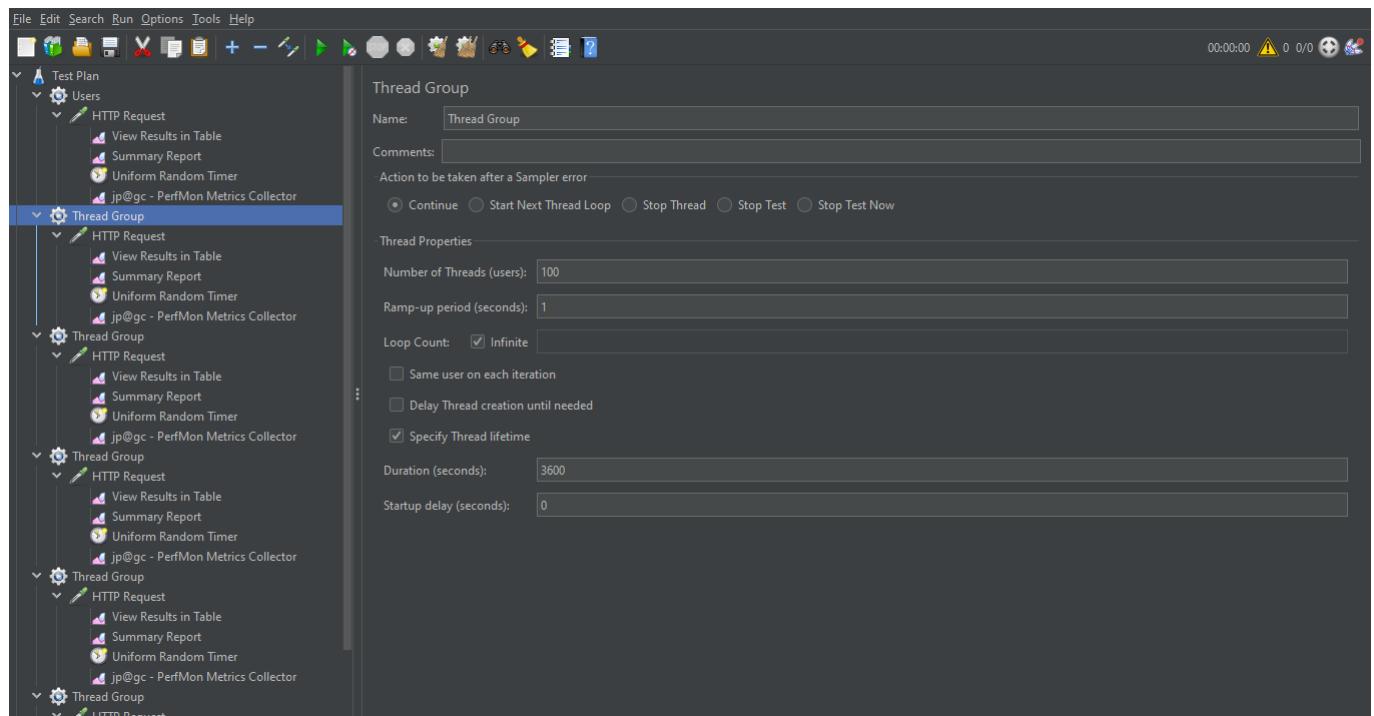


Figure 7.4: Thread Group Configuration

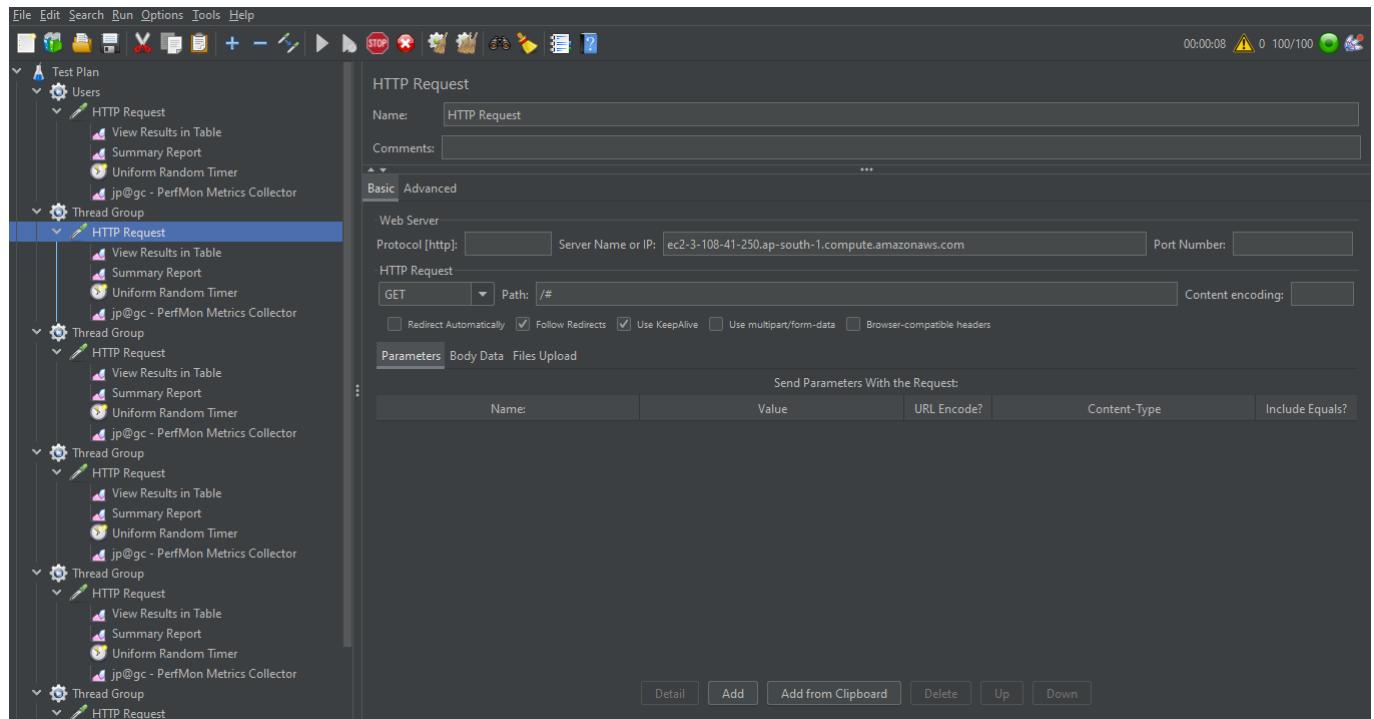


Figure 7.5: HTTP Request Sampler

The screenshot shows the JMeter Test Plan interface with the 'View Results in Table' listener selected. The central panel displays a table of results. The columns are: Sample #, Start Time, Thread Name, Label, Sample Time(...), Status, Bytes, Sent Bytes, Latency, and Connect Time(...). The table contains 21 rows of data. At the bottom of the table, there are buttons for 'Configure', 'Log/Display Only', and checkboxes for 'Errors', 'Successes'. The status bar at the top right shows '00:01:23', a warning icon, '0 0/100', and other icons.

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	09:52:48.195	Thread Group ...	HTTP Request	109	✓	2367	151	101	63
2	09:52:47.863	Thread Group ...	HTTP Request	441	✓	2367	151	433	396
3	09:52:48.180	Thread Group ...	HTTP Request	124	✓	2367	151	118	80
4	09:52:48.208	Thread Group ...	HTTP Request	96	✓	2367	151	91	53
5	09:52:48.168	Thread Group ...	HTTP Request	139	✓	2367	151	128	91
6	09:52:48.167	Thread Group ...	HTTP Request	143	✓	2367	151	143	95
7	09:52:48.186	Thread Group ...	HTTP Request	126	✓	2367	151	126	76
8	09:52:48.203	Thread Group ...	HTTP Request	110	✓	2367	151	110	58
9	09:52:48.064	Thread Group ...	HTTP Request	249	✓	2367	151	249	199
10	09:52:48.148	Thread Group ...	HTTP Request	166	✓	2367	151	165	114
11	09:52:48.035	Thread Group ...	HTTP Request	283	✓	2367	151	283	229
12	09:52:48.145	Thread Group ...	HTTP Request	173	✓	2367	151	173	119
13	09:52:48.160	Thread Group ...	HTTP Request	159	✓	2367	151	159	103
14	09:52:48.134	Thread Group ...	HTTP Request	185	✓	2367	151	185	129
15	09:52:48.113	Thread Group ...	HTTP Request	207	✓	2367	151	207	152
16	09:52:48.044	Thread Group ...	HTTP Request	284	✓	2367	151	284	221
17	09:52:48.104	Thread Group ...	HTTP Request	224	✓	2367	151	224	161
18	09:52:48.163	Thread Group ...	HTTP Request	165	✓	2367	151	165	102
19	09:52:48.111	Thread Group ...	HTTP Request	219	✓	2367	151	219	156
20	09:52:47.927	Thread Group ...	HTTP Request	408	✓	2367	151	408	340
21	09:52:48.076	Thread Group ...	HTTP Request	260	✓	2367	151	260	190

Figure 7.6: View Results in Table (.csv file saved as Log File)

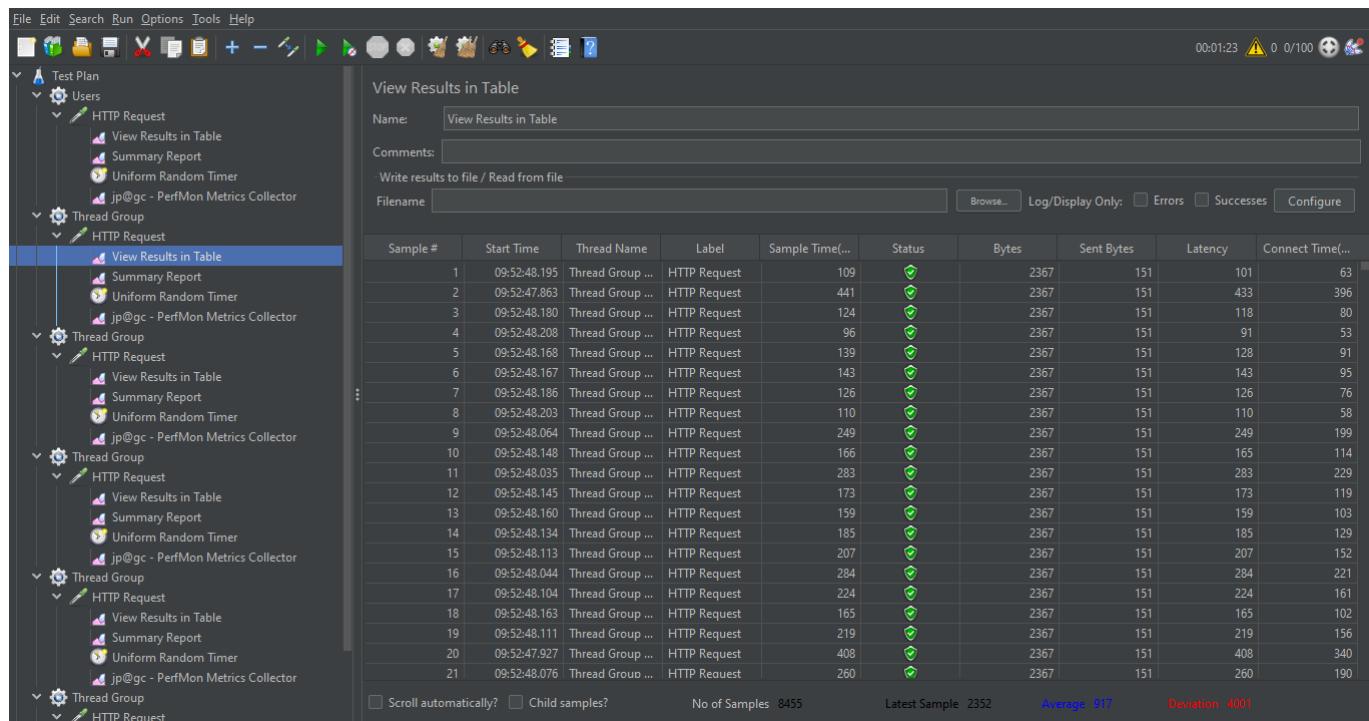


Figure 7.7: Summary Report (.csv file to be saved as Summary File)

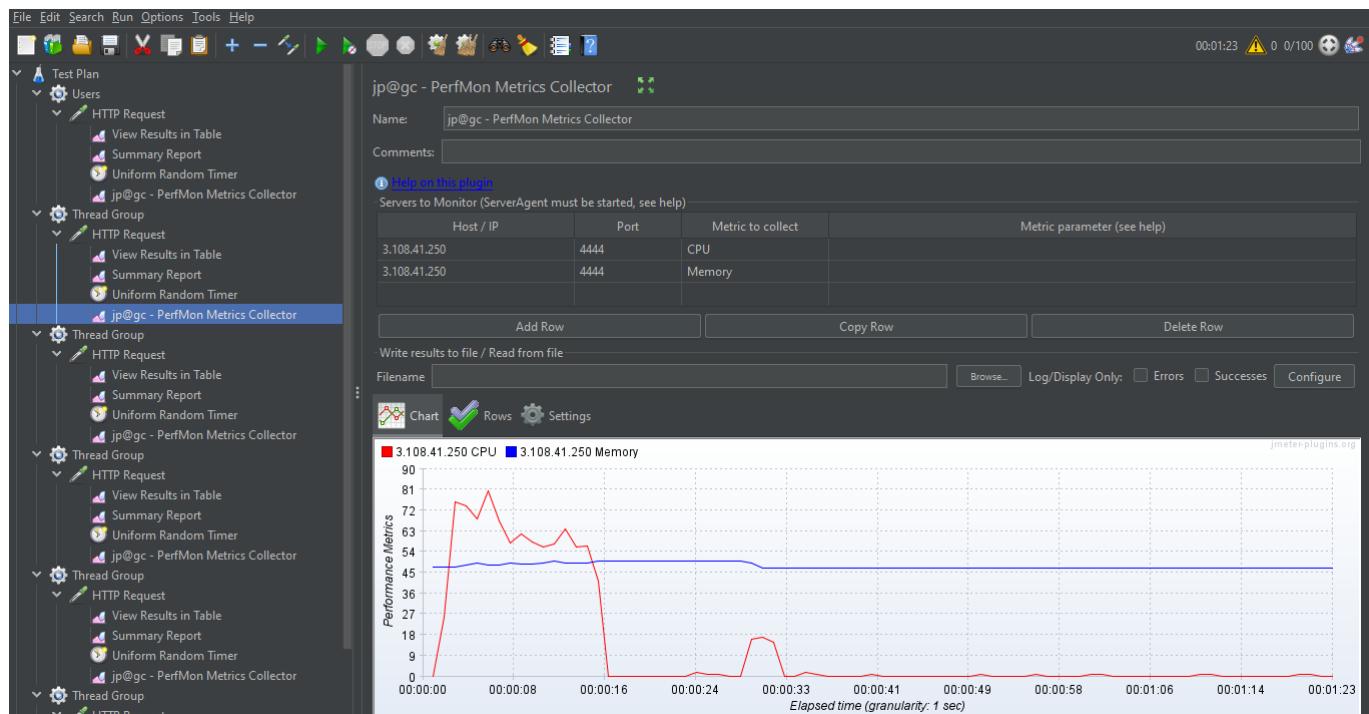


Figure 7.8: Perfmon Metrics Collector (saved as Performance file in .csv format)

Testing RDS instances:

The configurations are as follows:

- 1260 • Max number of Connection : 50

- Max Wait : 10000 ms

- Time between Eviction Runs : 6000 ms

- Database URL :

jdbc:mysql://test-database.cjamazw1gpow.ap-south-1.rds.amazonaws.com:3306/RDS\_Test

- 1265 • Port : 3306

- Thread groups : 5000

- Ramp-up Time : 1s

- Duration : 600s

- Startup Delay : 0s

- 1270 • Query Type

Running rds-test.jmx :

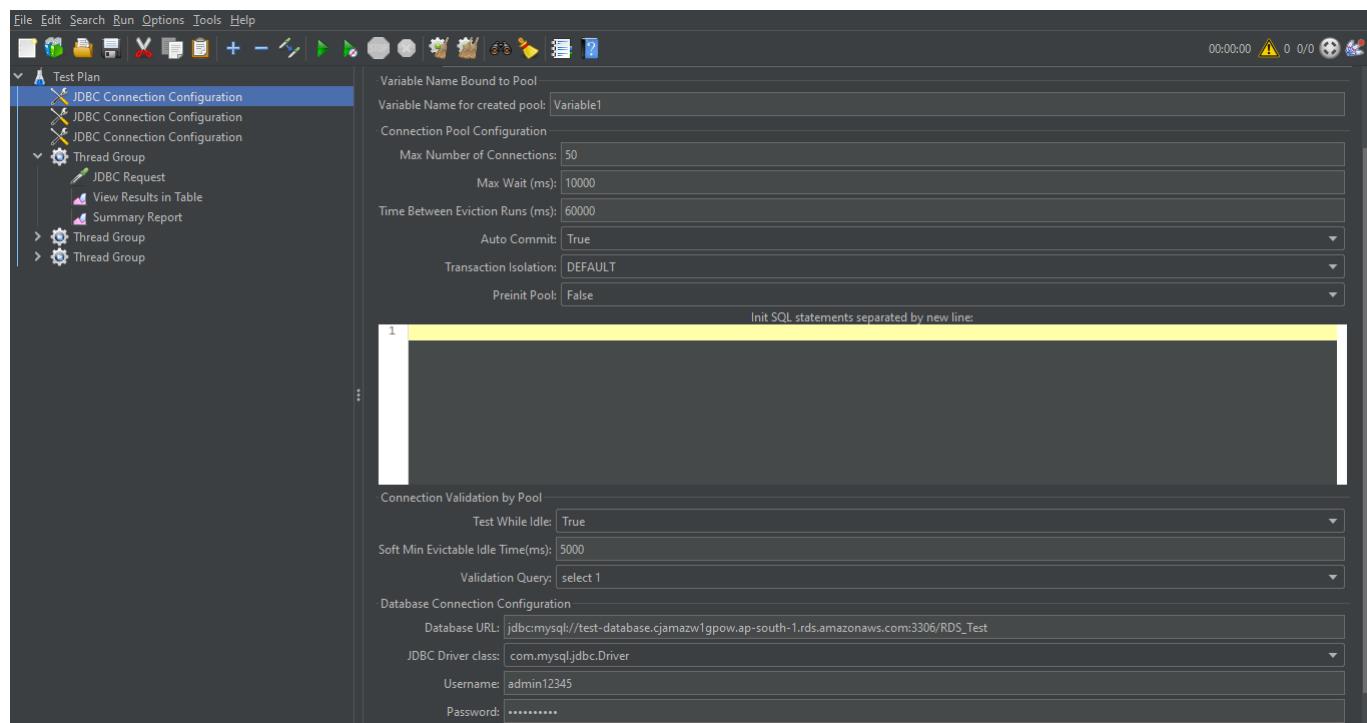


Figure 7.9: JDBC Connection Configuration

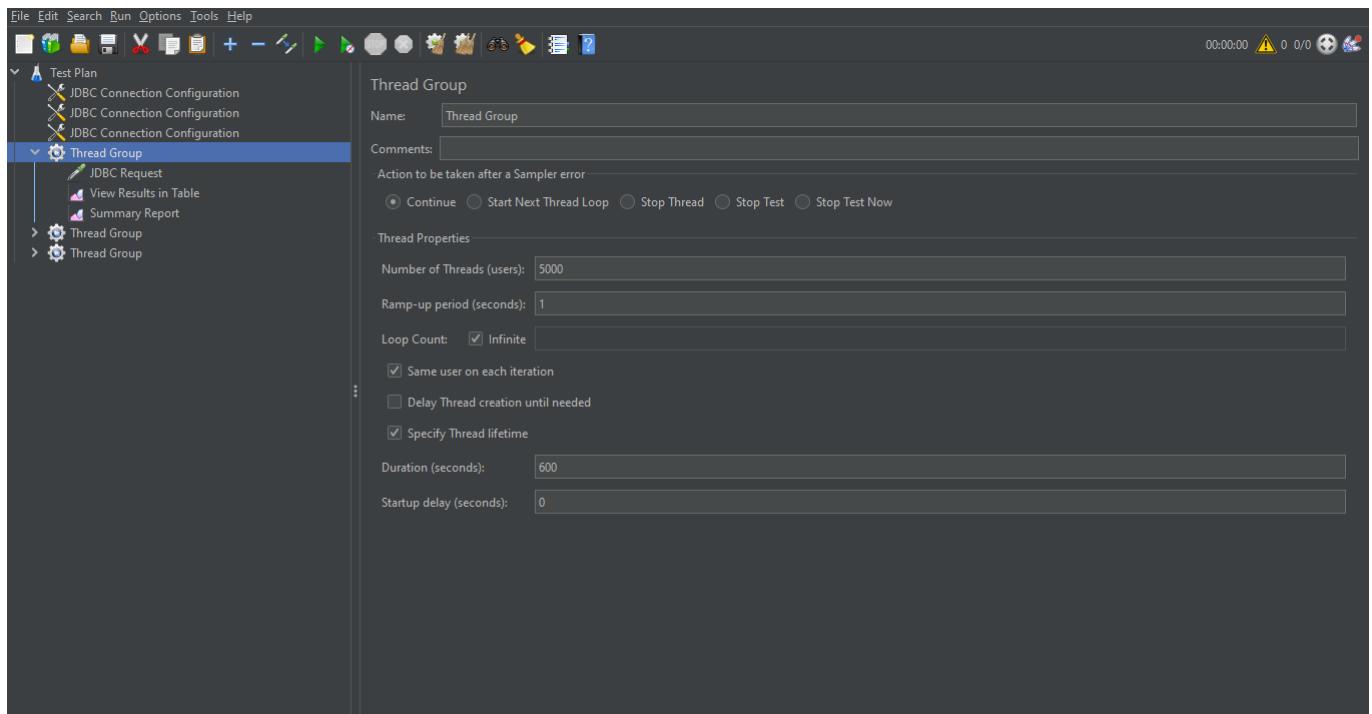


Figure 7.10: Thread Group

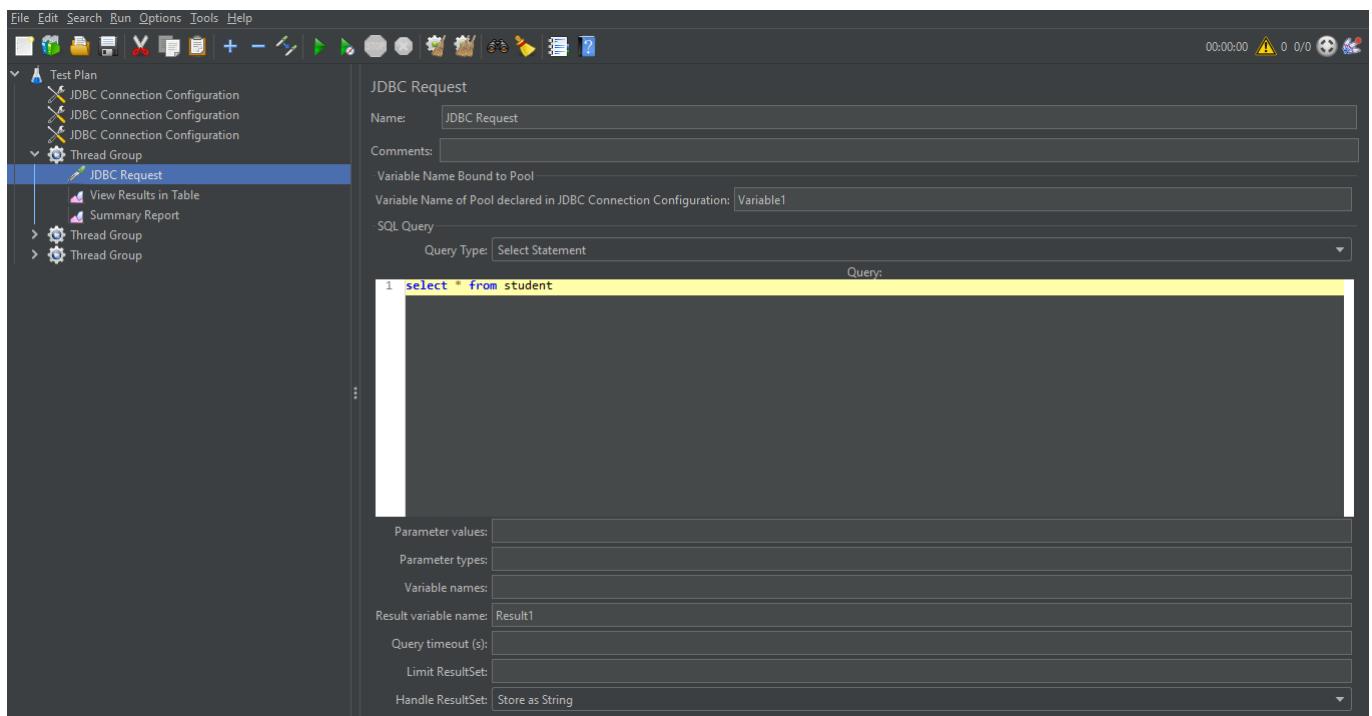


Figure 7.11: JDBC Request

The Results are stored in Log Files and Summary Files like we did for EC2 tests.

Testing S3:

1285 In the test shown below, the following configurations are used :

- Number of Threads : 100
  - Ramp-up period : 1s
  - Duration : 1800s
  - Startup Delay : 1s
- 1290
- HTTP Request Type : GET Request
  - Hostname : boilerplatecodebucket.s3.ap-south-1.amazonaws.com
  - Path : /index.html
  - In View results in table listener, File name can be mentioned in .csv or .txt format to save the results in that file
  - Summary Report can be saved after the test is complete by clicking on Save Data at the bottom of the window.

1295 Running S3 test:

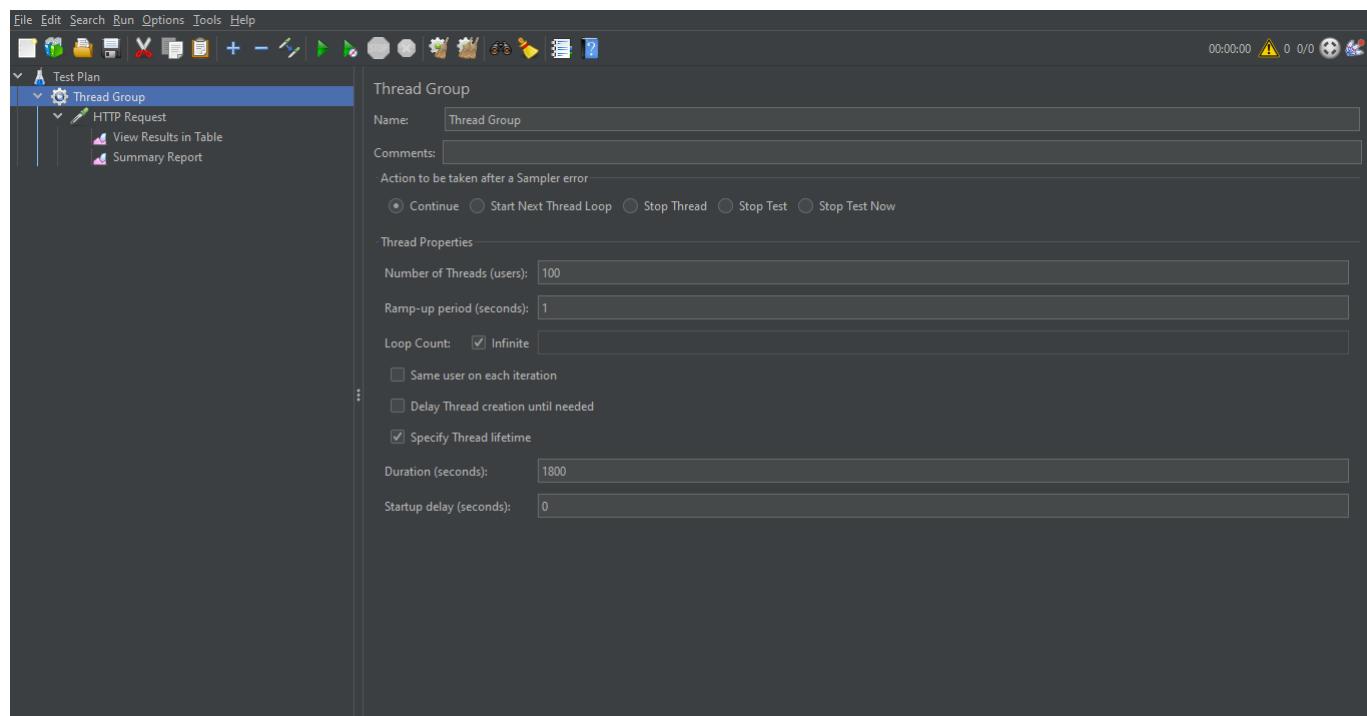


Figure 7.12: Thread Group

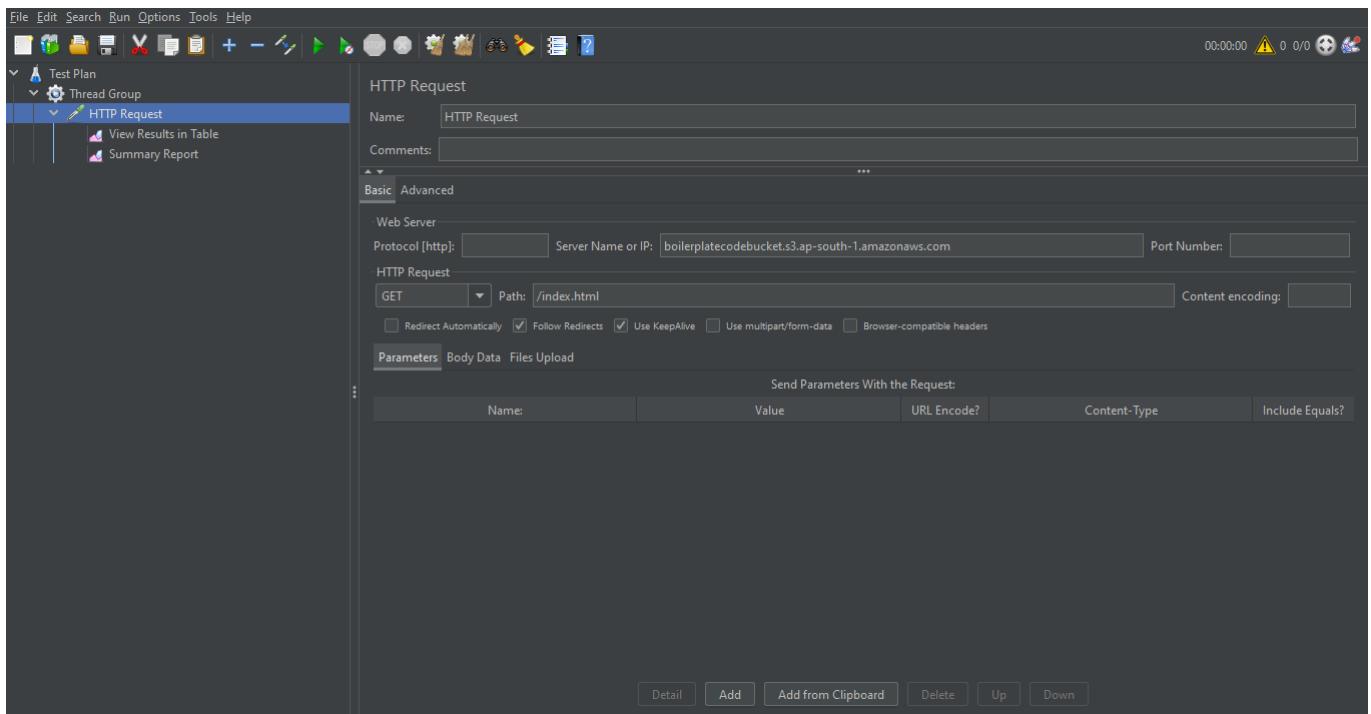


Figure 7.13: HTTP Request

```

Press 1 for logging in. Press 0 for
creating/registering an endpoint:
1

Enter endpoint ID:
2

Enter password:
[REDACTED]

Checking login credentials...
Successfully logged in to Navneeth's
Laptop!

If Windows, press 1. If non-Windows,
press 0.0

Enter the path where the logs are
stored:../../logs/
Your login session ID is:sXeImxQdsQ
Going through files...
No new files to be uploaded.

```

Figure 7.14: Endpoints Logs Upload

The results are stored in Log Files and Summary Files like we did for EC2 tests. These are further saved in .csv format in local directories. The parent directory is sent to the Back-end Database using Python Script.

**1310 8 Conclusion**

The rapid growth and dependency of the cloud industry will influence risk exposure and potential for impact (e.g. availability, performance, etc.).

- Effective management of service levels (e.g. availability, reliability, performance, security) and financial risk to CSCs is dependent on the CSPs capability and trustworthiness.
- The primary objective of our project will be analysing cloud SLAs and cloud security requirements, assessing and defining CSP trustworthiness levels, and establishing a model for predicting cloud service and SLA availability based on multiple factors (e.g. CSP trust levels, cloud service historic performance and cloud service characteristics).
- Evolving industry standards (e.g. NIST, ISO/IEC, etc) for cloud SLAs and the current research and findings related to work regarding CSP trustworthiness will be used as an addition to analyze CSP cloud computing services, SLA performance and CSP trustworthiness.
- Future enhancements will include an accurate and thorough comparison of several cloud service providers instead of just one.
- Another noteworthy point is to ensure that the logs collected in order for computing the trustworthiness quotient is prodigious. This ensures a comprehensive evaluation of the CSP under scope.
- The list of services offered by each CSP will also be increased so that the user has been given the opportunity to choose from a vast array of choices and specifics.

**Future Works**

- The scope of future works in Cloud Service Optimizer will only grow as testing will expand to more instance types of different service types.
- Our team is currently expanding the performance testing on various benchmarks in order to test the application in more areas. The goal is to develop a holistic and all-round analysis of a given cloud service.
- We aim to bring more CSPs into the mix so that more users can get trustworthy data from the standpoint of a cloud user.

- It is a tedious process to train and fit ever-increasing logs. Thus, our model can only sustain as a self-learning model that trains in the background threads. Therefore, this increases the processing efficiency by producing more output in fewer time consumed.
- We learn user inputs and study the type of requests being made with respect to the content areas, the instance types, and the service types. This helps gain insight about the most desired content areas in the cloud marketplace and performance testing can be prioritized on those areas which are trending in order to provide more accurate and reliable results.

## 9 References

- 1345 [1] Mahesh K, Dr. M. Laxmaiah, Dr. Yogesh Kumar Sharma, Predict Trustworthiness of Cloud Services Using Linear Regression Model, IJAST, vol. 29, no. 04, 2020, pp. 5737-5745.
- [2] J. Kanpariyasoontorn and T. Senivongse, "Cloud service trustworthiness assessment based on cloud controls matrix," 2017 19th International Conference on Advanced Communication Technology (ICACT), Bongpyeong, 2017, pp. 291-297.
- 1350 [3] S. Pandey and A. K. Daniel, "Fuzzy logic based cloud service trustworthiness model," 2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, 2016, pp. 73-78.
- [4] L. Wang and Z. Wu, "A Trustworthiness Evaluation Framework in Cloud Computing for Service Selection," in 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), Singapore, Singapore, 2014, pp. 101-106.
- 1355 [5] Y. Wang, J. Wen, W. Zhou, B. Tao, Q. Wu and Z. Tao, "A Cloud Service Selection Method Based on Trust and User Preference Clustering," in IEEE Access, vol. 7, 2019, pp. 110279-110292.
- [6] C. Jatoh, G. Gangadharan, U. Fiore, and R. Buyya, Selcloud: a hybrid multi-criteria decision-making model for selection of cloud services, Soft Computing, pp. 115, 2018.
- 1360 [7] C. Yiming and Z. Yiwei, Saas vendor selection basing on analytic hierarchy process, in 2011 Fourth International Joint Conference on Computational Sciences and Optimization. IEEE, 2011, pp. 511515.
- [8] T. L. Saaty, Decision making with the analytic hierarchy process, International journal of services sciences, vol. 1, no. 1, pp. 8398, 2008.
- 1365 [9] Z. ur Rehman, F. K. Hussain, and O. K. Hussain, Towards multicriteria cloud service selection, in 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. Ieee, 2011, pp. 4448.

## 10 Appendix

1370      Source code

### FRONT-END

#### Home.js

```
1371
2 import { Component } from 'react';
3 import {Link} from 'react-router-dom'
4 import './styles/Home.css'
5 import Cloud from '../resources/cloud.png'

1386
7 class Home extends Component {
8     render() {
9         return(
10             <div style = {{height: '100%', width: '100%'}}>
11                 <div className = 'headerDivStyle'>
12                     <p className = 'headerStyle'>CLOUD SERVICE OPTIMIZER</p>
13                     <img src = { Cloud } alt = "Error" className = 'logoStyle'>
14             />
15                 </div>
16                 <div className = 'linkDivStyle'>
17                     <Link class = 'linkStyle' to = "/dashboard">Dashboard</Link>
18
19                     <Link className = 'linkStyle' to = "/about-us">FAQs</Link>
20                     <Link className = 'linkStyle' to = "/contact-us">Contact Us
21             </Link>
22
23                 <div className = 'introDivStyle'>
24                     <p className = 'introTextStyle'>Cloud Service Selection has
25                         progressed as a cloud computing paradigm of entrusting good faith in a
26                         Cloud Service Provider (CSP) for the services with specifications that are
27                         legally agreed upon by the involved parties.<br/> The abundance of
28                         seemingly similar Cloud Services and the enormous range of customized user
29                         preferences and requirements have led to difficulty in choosing the optimal
30                         CSP. Some of the main issues faced by CC are lack of trust in service
31                         providers with respect to availability, reliability and efficiency of
32                         services at their time of need, ambiguity in SLAs, non-compliance with SLA,
```

```

absence of diversified trust elements , and Quality-of-Service guarantee.<br/> Although a number of cloud service e-marketplaces are present , users find it very difficult to manually compare the services of different CSPs: especially new users who must go through each feature s description separately during Cloud Service Provider selection.<br/><br/> This is where <span style = {{color: '#F9CC88'}}>'Cloud Service Optimizer' </span>comes into the picture to make the job quick , easy and efficient.</p>
22           </div>
23           </div>
24     );
25   }
26
27 export default Home;
1420

```

## Home.css

```

1 .linkStyle {
2   color: #F9CC88 ;
3   background-color: #fdfdfd ;
4   border-radius: 50px ;
5   text-align: center ;
6   font-size: 20px ;
7   text-decoration: none ;
8   font-weight: bold ;
9   margin: 2% ;
10  padding-left: 1% ;
11  padding-right: 1% ;
12  padding-top: 0.7% ;
13  padding-bottom: 0.7% ;
14  font-family: 'Arial Narrow Bold' ;
15  margin-top: 10% ;
16 }
17
18 a:hover {
19   color: orange ;
20 }
21

```

```
122 .headerStyle {  
123     margin-top: 2%;  
124     color: #FDFDFD;  
125     font-size: 230%;  
126     text-align: center;  
127     font-weight: bold;  
128     height: 10%;  
129     text-shadow: 2px 2px 2px #222;  
130     margin-left: 3%;  
131 }  
132  
133 .headerDivStyle {  
134     height: 10%;  
135     align-items: center;  
136     background-color: #F9CC88;  
137     box-shadow: 1px 3px 0.5px #9E9E9E50;  
138     justify-content: center;  
139     display: flex;  
140     border-bottom-left-radius: 50px;  
141     border-bottom-right-radius: 50px;  
142     margin-bottom: 2%;  
143 }  
144  
145 .linkDivStyle {  
146     margin-left: 35%;  
147     margin-top: 1%;  
148 }  
149  
150 .logoStyle {  
151     color: #FDFDFD;  
152     height: 7%;  
153     width: 7%;  
154     margin-bottom: 1%;  
155     margin-top: 2%;  
156 }  
157  
158 .introDivStyle {  
159     width: 70%;  
160     text-align: justify;
```

```
61     margin-left: auto;
62     margin-right: auto;
63     margin-top: 7%;
64     border-bottom-left-radius: 50px;
65     border-top-right-radius: 50px;
66     background-color: #F9CC88;
67     padding: 3%;
68     background: rgba(255,255,255,0.5);
69     margin-bottom: 7%;
70 }
71
72 .introTextStyle {
73     color: #696969;
74     font-family: 'Arial Narrow Bold';
75     font-weight: bold;
76     font-size: 130%;
77     margin-left: 7%;
78     margin-right: 7%;
79     margin-top: 2%;
80     margin-bottom: 2%;
81 }
```

## AboutUs.js

```
1 import { Component } from 'react';
2 import './styles/AboutUs.css'
3
4 class AboutUs extends Component {
5
6     render() {
7         return (
8             <div>
9                 <div className = 'titleDivStyle'>
10                     <h1 className = 'titleStyle'>FAQs</h1>
11                 </div>
12                 <div className = 'qDivStyle'>
```

```

14           <p className = 'qTitleStyle' style = {{color: '#F9CC88'}}>
Answer a few questions on the dashboard to get started right now!</p>
15           <p className = 'qTitleStyle' style = {{color: '#F9CC88'}}>
For EC2 and RDS:</p>
16           <p className = 'qTextStyle'>1. What AWS Service are you
looking for?</p>
17           <p className = 'qTextStyle'>Clicking on the 'Select Service
' Button will display the services which are available for comparison and
1525 calculation of trustworthy metric. Select the service as per your
requirement.</p>
18           <p className = 'qTextStyle'>2. What performance metric do
you want us to focus on?</p>
19           <p className = 'qTextStyle'>Select from a wide range of
available Filters and assign priority based on how important each attribute
is for your business requirement. Checking the 'Assign same priority to
all filters' will assign same priority to all attributes. Don't forget to
select the filters that you want to apply after this step. Individual
priorities can also be assigned by unchecking the above checkbox, clicking
on each filter and filling the priority numbers (Integer type in range [1,
n]: 1 being the highest priority, n      Total number of Filters available)
in the textboxes which appear beside them. To deselect any filter, simply
click again on the selected one.</p>
20           <p className = 'qTextStyle'>3. What instance types of the
above listed AWS service do you want? </p>
21           <p className = 'qTextStyle'>Select the instance types that
you want compare between and click on Save. If no instances are selected
all of the shown instances will be automatically chosen for Trustworthiness
Evaluation.</p>
22           <p className = 'qTextStyle'>After successfully uploading
the above data, click on Show Results to assess the trustworthiness value
and graphical results of the comparison.</p>
23           <p className = 'qTitleStyle' style = {{color: '#F9CC88'}}>
For S3:</p>
24           <p className = 'qTextStyle'>1. What AWS Service are you
looking for?</p>
25           <p className = 'qTextStyle'>Clicking on the 'Select Service
' Button will display the services which are available for comparison and
calculation of trustworthy metric. Select S3.</p>
26           <p className = 'qTextStyle'>2. What metric should ML

```

Prediction be applied to?</p>

27                   <p className = 'qTextStyle'>For S3, Users must choose n-1 (n being the total number of attributes) filters and must provide the values for each of the selected filters. The value of the only 1 remaining filter is predicted. Here, priorities can be positive Integers or Decimal values.</p>

28                   <p className = 'qTextStyle'>Allowed values for the selected filters:</p>

29                   <p className = 'qTextStyle'>Consistency (stdDev): 100 to 1570 1000. Here, 100 indicates a low level of Consistency and 1000 indicates a high level of Consistency. Knowing the standard deviation of your data set tells you how densely the data points are clustered around the mean. The smaller the standard deviation, the more consistent the data. </p>

30                   <p className = 'qTextStyle'>Error Rate (errorRate): 0 to 1575 100. Here, 0 refers to low error rate and 100 refers to high error rate. It is the percentage value of errors caused due to various reasons when connecting to the backend hosted by a service type. Total percentage of errors found for a particular sample request. 0.0% shows that all requests completed successfully. Total equals the percentage of error samples in all samples.</p>

31                   <p className = 'qTextStyle'>Latency (Latency): 0 to 300. 1580 Here, 0 refers to a low level of Latency and 300 refers to a high level of Latency. JMeter measures the latency from just before sending the request to just after the first response has been received. Thus, the time includes all the processing needed to assemble the request as well as assembling the first part of the response, which in general will be longer than one byte. Protocol analysers (such as Wireshark) measure the time when bytes are actually sent/received over the interface. The JMeter time should be closer to that which is experienced by a browser or other application client.</p>

32                   <p className = 'qTextStyle'>Elapsed Time (elapsed): 0 to 1590 300. Here, 0 refers to low Elapsed time and 300 refers to high Elapsed time. JMeter measures the elapsed time from just before sending the request to just after the last response has been received. JMeter does not include the time needed to render the response, nor does JMeter process any client code, for example JavaScript.</p>

33                   <p className = 'qTextStyle'>Connection Time (Connect): 0 to 1595 300. Here, 0 refers to the low level of Connection Time and 300 refers to the high level of Connection time. JMeter measures the time it took to

```

1600 establish the connection, including SSL handshake. Note that connect time
is not automatically subtracted from latency. In case of connection error,
the metric will be equal to the time it took to face the error, for example
in case of Timeout, it should be equal to connection timeout.</p>
34             <p className = 'qTextStyle'>Throughput (Throughput): 0 to
1605 5000. Here, 0 refers to a low level of Throughput and 5000 refers to a high
level of Throughput. Hits/sec, or total number of requests per unit of
time (sec, mins, hr) sent successfully to server during test.</p>
35             <p className = 'qTextStyle' style = {{color: '#F9CC88'}}>
After selecting the filters, press Save to upload the data. Click on Show
1610 Results to assess the trustworthiness value and graphical results of the
comparison.</p>
36         </div>
37     </div>
38 )
39 };
40 }
41
42
43 export default AboutUs;
1620

```

## AboutUs.css

```

1 body {
1622     width: 100%;
3     height: 100%;
4     background-color: #e8f4f8;
5 }
6
1637 .titleStyle {
8     margin-top: 0%;
9     font-size: 250%;
10    color: #FDFDFD;
11    text-shadow: 2px 2px 2px #222;
12    text-align: center;
13    padding: 2%;
14    margin-bottom: 2%;

```

```
15 }
16
147 .titleDivStyle {
17   background-color: #F9CC88;
18   width: 15%;
19   margin-left: auto;
20   margin-right: auto;
21
22   border-bottom-left-radius: 50px;
23   border-bottom-right-radius: 50px;
24   padding-bottom: 0.4%;
25 }
26
127 .qDivStyle {
27   width: 70%;
28   text-align: justify;
29   margin-left: auto;
30   margin-right: auto;
31
32   margin-top: 3%;
33   border-bottom-left-radius: 50px;
34   border-top-right-radius: 50px;
35   background-color: #F9CC88;
36   padding: 3%;
37   background: rgba(255,255,255,0.5);
38   margin-bottom: 3%;
39 }
40
41 .qTitleStyle {
42   color: #696969;
43   font-family: 'Arial Narrow Bold';
44   font-weight: bold;
45   font-size: 170%;
46   margin-left: 7%;
47   margin-right: 7%;
48   margin-top: 2%;
49   margin-bottom: 2%;
50 }
51
152 .qTextStyle {
53   color: #696969;
```

```

54     font-family: 'Arial Narrow Bold';
55     font-weight: bold;
56     font-size: 130%;
57     margin-left: 7%;
58     margin-right: 7%;
59     margin-top: 2%;
60     margin-bottom: 2%;
61 }

```

## ContactUs.js

```

1 import { Component } from 'react';
1692 import './styles/ContactUs.css'
3
4 class ContactUs extends Component {
5     render() {
6         return(
1697
8             <div>
9                 <div className = 'titleDivStyle1'>
10                    <h1 className = 'titleStyle1'>Contact Us</h1>
11                </div>
142                <div className = 'qDivOverallStyle' >
13                    <p className = 'qTitleStyle1' style = {{color: '#F9CC88',
16                     display: 'block'}}>Creators:</p>
14                    <div className = 'qDivStyle1'>
15                        <div style = {{width: '27%'}}>
16                            <p className = 'qTextStyle1' style = {{textAlign: 'right'}}>Navneeth Krishna M</p>
17                            <p className = 'qTextStyle1' style = {{textAlign: 'right'}}>Nishchala M</p>
18                            <p className = 'qTextStyle1' style = {{textAlign: 'right'}}>Oindrila Chakraborti</p>
19                        </div>
20                    <div>
21                        <p className = 'qTextStyle1'>navneeth.

```

```
122             <p className = 'qTextStyle1'>
123                 nishchalamkumar12@gmail.com</p>
124             <p className = 'qTextStyle1'>oindrila2411@gmail.com
125         </p>
126     </div>
127     </div>
128     )};
129 }
130
131 export default ContactUs;
```

## ContactUs.css

```
1730
1 .titleStyle1 {
2     margin-top: 0%;
3     font-size: 250%;
4     color: #FDFDFD;
5     text-shadow: 2px 2px 2px #222;
6     text-align: center;
7     padding: 2%;
8     margin-bottom: 2%;
9 }
10
11 .titleDivStyle1 {
12     background-color: #F9CC88;
13     width: 22%;
14     margin-left: auto;
15     margin-right: auto;
16     border-bottom-left-radius: 50px;
17     border-bottom-right-radius: 50px;
18     padding-bottom: 0.4%;
19 }
20
21 .qDivStyle1 {
22     display: flex;
```

```
23     text-align: center;
24     justify-content: center;
25     align-items: center;
26 }
27
28 .qTitleStyle1 {
29     color: #696969;
30     font-family: 'Arial Narrow Bold';
31     font-weight: bold;
32     font-size: 170%;
33     margin-left: 7%;
34     margin-right: 7%;
35     margin-top: 2%;
36     margin-bottom: 2%;
37     text-align: center;
38 }
39
40 .qTextStyle1 {
41     color: #696969;
42     font-family: 'Arial Narrow Bold';
43     font-weight: bold;
44     font-size: 130%;
45     margin-left: 7%;
46     margin-right: 7%;
47     margin-top: 2%;
48     margin-bottom: 2%;
49     text-align: left;
50 }
51
52 .qDivOverallStyle {
53     width: 70%;
54     text-align: center;
55     justify-content: center;
56     align-items: center;
57     margin-left: auto;
58     margin-right: auto;
59     margin-top: 3%;
60     border-bottom-left-radius: 50px;
61     border-top-right-radius: 50px;
```

```

62     background-color: #F9CC88;
63     padding: 3%;
64     background: rgba(255,255,255,0.5);
65     margin-bottom: 3%;
66 }

```

## Dashboard.js

```

1800
1 import { Component } from 'react';
2 import axios from 'axios'
3 import { Route } from 'react-router-dom'
4 import './styles/Dashboard.css'

1805
5
6 class Dashboard extends Component {
7
8     constructor() {
9         super();
10        var ec2 = "Amazon EC2";
11        var s3 = "Amazon S3";
12        var rds = "Amazon RDS";
13
14        var defaultJson = {}
15        defaultJson[ec2] = {
16            "overall": {
17                "filterCompare": {
18                    "t2.xlarge > t2.large": "72.02",
19                    "t2.xlarge > t2.medium": "95.27",
20                    "t2.xlarge > t2.small": "112.5",
21                    "t2.xlarge > t2.micro": "115.67",
22                    "t2.xlarge > t2.nano": "131.2",
23                    "t2.large > t2.medium": "13.51",
24                    "t2.large > t2.small": "23.53",
25                    "t2.large > t2.micro": "25.37",
26                    "t2.large > t2.nano": "34.4",
27                    "t2.medium > t2.small": "8.82",
28                    "t2.medium > t2.micro": "10.45",
29                    "t2.medium > t2.nano": "18.4",

```

```
130          "t2.small > t2.micro": "1.49",
131          "t2.small > t2.nano": "8.8",
132          "t2.micro > t2.nano": "7.2"
133      },
134      "data": {
135          "t2.xlarge": 0.289,
136          "t2.large": 0.168,
137          "t2.medium": 0.148,
138          "t2.small": 0.136,
139          "t2.micro": 0.134,
140          "t2.nano": 0.125
141      },
142      "displayName": "Overall"
143  },
144  "stdDev": {
145      "filterCompare": {
146          "t2.xlarge > t2.micro": "88.59",
147          "t2.xlarge > t2.large": "93.79",
148          "t2.xlarge > t2.small": "96.5",
149          "t2.xlarge > t2.medium": "99.29",
150          "t2.micro > t2.large": "2.76",
151          "t2.micro > t2.small": "4.2",
152          "t2.micro > t2.medium": "5.67",
153          "t2.large > t2.small": "1.4",
154          "t2.large > t2.medium": "2.84",
155          "t2.small > t2.medium": "1.42",
156          "t2.medium > t2.large": "0.0"
157      },
158      "data": {
159          "t2.xlarge": 0.281,
160          "t2.micro": 0.149,
161          "t2.large": 0.145,
162          "t2.small": 0.143,
163          "t2.nano": 0.141,
164          "t2.medium": 0.141
165      },
166      "displayName": "Consistency"
167  },
168  "errorRate": {
```

```
69      "filterCompare": {
70          "t2.micro > t2.small": "3.55",
71          "t2.micro > t2.medium": "4.17",
72          "t2.micro > t2.xlarge": "14.38",
73          "t2.small > t2.medium": "0.6",
74          "t2.small > t2.xlarge": "10.46",
75          "t2.medium > t2.medium": "0.0",
76          "t2.medium > t2.xlarge": "9.8"
77      },
78      "data": {
79          "t2.micro": 0.175,
80          "t2.small": 0.169,
81          "t2.nano": 0.168,
82          "t2.large": 0.168,
83          "t2.medium": 0.168,
84          "t2.xlarge": 0.153
85      },
86      "displayName": "Error Rate"
87  },
88  "Throughput": {
89      "filterCompare": {
90          "t2.xlarge > t2.large": "53.16",
91          "t2.xlarge > t2.small": "56.13",
92          "t2.xlarge > t2.medium": "61.33",
93          "t2.xlarge > t2.micro": "62.42",
94          "t2.xlarge > t2.nano": "65.75",
95          "t2.large > t2.small": "1.94",
96          "t2.large > t2.medium": "5.33",
97          "t2.large > t2.micro": "6.04",
98          "t2.large > t2.nano": "8.22",
99          "t2.small > t2.medium": "3.33",
100         "t2.small > t2.micro": "4.03",
101         "t2.small > t2.nano": "6.16",
102         "t2.medium > t2.micro": "0.67",
103         "t2.medium > t2.nano": "2.74",
104         "t2.micro > t2.nano": "2.05"
105     },
106     "data": {
107         "t2.xlarge": 0.242,
```

```
108         "t2.large": 0.158,
109         "t2.small": 0.155,
110         "t2.medium": 0.15,
111         "t2.micro": 0.149,
112         "t2.nano": 0.146
113     },
114     "displayName": "Throughput"
115 },
116     "elapsed": {
117         "filterCompare": {
118             "t2.xlarge > t2.micro": "64.71",
119             "t2.xlarge > t2.small": "65.79",
120             "t2.xlarge > t2.medium": "69.13",
121             "t2.xlarge > t2.nano": "73.79",
122             "t2.micro > t2.small": "0.66",
123             "t2.micro > t2.medium": "2.68",
124             "t2.micro > t2.nano": "5.52",
125             "t2.small > t2.medium": "2.01",
126             "t2.small > t2.nano": "4.83",
127             "t2.medium > t2.medium": "0.0",
128             "t2.medium > t2.nano": "2.76"
129         },
130         "data": {
131             "t2.xlarge": 0.252,
132             "t2.micro": 0.153,
133             "t2.small": 0.152,
134             "t2.large": 0.149,
135             "t2.medium": 0.149,
136             "t2.nano": 0.145
137         },
138         "displayName": "Elapsed Time"
139     },
140     "Latency": {
141         "filterCompare": {
142             "t2.xlarge > t2.nano": "111.97",
143             "t2.xlarge > t2.small": "115.0",
144             "t2.xlarge > t2.large": "116.55",
145             "t2.xlarge > t2.medium": "118.12",
146             "t2.nano > t2.small": "1.43",
```

```
147          "t2.nano > t2.large": "2.16",
148          "t2.nano > t2.medium": "2.9",
149          "t2.small > t2.small": "0.0",
150          "t2.small > t2.large": "0.72",
151          "t2.small > t2.medium": "1.45",
152          "t2.large > t2.medium": "0.72"
153      },
154      "data": {
155          "t2.xlarge": 0.301,
156          "t2.nano": 0.142,
157          "t2.micro": 0.14,
158          "t2.small": 0.14,
159          "t2.large": 0.139,
160          "t2.medium": 0.138
161      },
162      "displayName": "Latency"
163  },
164  "Connect": {
165      "filterCompare": {
166          "t2.xlarge > t2.micro": "61.69",
167          "t2.xlarge > t2.small": "63.82",
168          "t2.xlarge > t2.medium": "67.11",
169          "t2.xlarge > t2.nano": "69.39",
170          "t2.micro > t2.small": "1.32",
171          "t2.micro > t2.medium": "3.36",
172          "t2.micro > t2.nano": "4.76",
173          "t2.small > t2.medium": "2.01",
174          "t2.small > t2.nano": "3.4",
175          "t2.medium > t2.medium": "0.0",
176          "t2.medium > t2.nano": "1.36"
177      },
178      "data": {
179          "t2.xlarge": 0.249,
180          "t2.micro": 0.154,
181          "t2.small": 0.152,
182          "t2.large": 0.149,
183          "t2.medium": 0.149,
184          "t2.nano": 0.147
185      },
```

```
186         "displayName": "Connection Time"
187     },
188     "CPU": {
189         "filterCompare": {
190             "t2.xlarge > t2.large": "83.25",
191             "t2.xlarge > t2.medium": "114.72",
192             "t2.xlarge > t2.small": "221.1",
193             "t2.xlarge > t2.micro": "243.14",
194             "t2.xlarge > t2.nano": "306.98",
195             "t2.large > t2.medium": "17.18",
196             "t2.large > t2.small": "75.23",
197             "t2.large > t2.micro": "87.25",
198             "t2.large > t2.nano": "122.09",
199             "t2.medium > t2.small": "49.54",
200             "t2.medium > t2.micro": "59.8",
201             "t2.medium > t2.nano": "89.53",
202             "t2.small > t2.micro": "6.86",
203             "t2.small > t2.nano": "26.74",
204             "t2.micro > t2.nano": "18.6"
205         },
206         "data": {
207             "t2.xlarge": 0.35,
208             "t2.large": 0.191,
209             "t2.medium": 0.163,
210             "t2.small": 0.109,
211             "t2.micro": 0.102,
212             "t2.nano": 0.086
213         },
214         "displayName": "CPU Utilization"
215     },
216     "memory": {
217         "filterCompare": {
218             "t2.xlarge > t2.large": "96.37",
219             "t2.xlarge > t2.medium": "289.6",
220             "t2.xlarge > t2.small": "616.18",
221             "t2.xlarge > t2.micro": "958.7",
222             "t2.xlarge > t2.nano": "1773.08",
223             "t2.large > t2.medium": "98.4",
224             "t2.large > t2.small": "264.71",
```

```

225         "t2.large > t2.micro": "439.13",
226         "t2.large > t2.nano": "853.85",
227         "t2.medium > t2.small": "83.82",
228         "t2.medium > t2.micro": "171.74",
229         "t2.medium > t2.nano": "380.77",
230         "t2.small > t2.micro": "47.83",
231         "t2.small > t2.nano": "161.54",
232         "t2.micro > t2.nano": "76.92"
233     },
234     "data": {
235         "t2.xlarge": 0.487,
236         "t2.large": 0.248,
237         "t2.medium": 0.125,
238         "t2.small": 0.068,
239         "t2.micro": 0.046,
240         "t2.nano": 0.026
241     },
242     "displayName": "memory"
243 }
244 };
245 defaultJson[s3] = {
246     "predictedValues": {
247         "Consistency": "425.82"
248     },
249     "selectedValues": {
250         "Connection Time": 24.0,
251         "Elapsed Time": 2116.0,
252         "Error Rate": 83.0,
253         "Latency": 28.0,
254         "Throughput": 36.0
255     }
256 };
257 defaultJson[rds] = {
258     "Connect": {
259         "data": {
260             "db.t2.medium": 0.34,
261             "db.t2.micro": 0.32,
262             "db.t2.small": 0.339
263         },

```

```
264         "displayName": "Connection Time",
265         "filterCompare": {
266             "db.t2.medium > db.t2.micro": "6.25",
267             "db.t2.medium > db.t2.small": "0.29",
268             "db.t2.small > db.t2.micro": "5.94"
269         }
270     },
271     "Latency": {
272         "data": {
273             "db.t2.medium": 0.311,
274             "db.t2.micro": 0.392,
275             "db.t2.small": 0.297
276         },
277         "displayName": "Latency",
278         "filterCompare": {
279             "db.t2.medium > db.t2.small": "4.71",
280             "db.t2.micro > db.t2.medium": "26.05",
281             "db.t2.micro > db.t2.small": "31.99"
282         }
283     },
284     "Throughput": {
285         "data": {
286             "db.t2.medium": 0.325,
287             "db.t2.micro": 0.345,
288             "db.t2.small": 0.329
289         },
290         "displayName": "Throughput",
291         "filterCompare": {
292             "db.t2.micro > db.t2.medium": "6.15",
293             "db.t2.micro > db.t2.small": "4.86",
294             "db.t2.small > db.t2.medium": "1.23"
295         }
296     },
297     "elapsed": {
298         "data": {
299             "db.t2.medium": 0.341,
300             "db.t2.micro": 0.321,
301             "db.t2.small": 0.338
302         },
303     }
```

```
303     "displayName": "Elapsed Time",
304     "filterCompare": {
305         "db.t2.medium > db.t2.micro": "6.23",
306         "db.t2.medium > db.t2.small": "0.89",
307         "db.t2.small > db.t2.micro": "5.3"
308     }
309 },
310 "errorRate": {
311     "data": {
312         "db.t2.medium": 0.284,
313         "db.t2.micro": 0.403,
314         "db.t2.small": 0.313
315     },
316     "displayName": "Error Rate",
317     "filterCompare": {
318         "db.t2.micro > db.t2.medium": "41.9",
319         "db.t2.micro > db.t2.small": "28.75",
320         "db.t2.small > db.t2.medium": "10.21"
321     }
322 },
323 "overall": {
324     "data": {
325         "db.t2.medium": 0.315,
326         "db.t2.micro": 0.361,
327         "db.t2.small": 0.325
328     },
329     "displayName": "Overall",
330     "filterCompare": {
331         "db.t2.micro > db.t2.medium": "14.6",
332         "db.t2.micro > db.t2.small": "11.08",
333         "db.t2.small > db.t2.medium": "3.17"
334     }
335 },
336 "stdDev": {
337     "data": {
338         "db.t2.medium": 0.284,
339         "db.t2.micro": 0.383,
340         "db.t2.small": 0.333
341     },
342 }
```

```
342         "displayName": "Consistency",
343         "filterCompare": {
344             "db.t2.micro > db.t2.medium": "34.86",
345             "db.t2.micro > db.t2.small": "15.02",
346             "db.t2.small > db.t2.medium": "17.25"
347         }
348     }
349 }
350
351 this.ec2 = ec2;
352 this.s3 = s3;
353 this.rds = rds;
354 this.defaultResponse = defaultJson;
355
356
357 this.state = {
358     showHideServices: false, showHideFilters: false, showHideInstances:
359     false,
360     selectEC2: false, selectS3: false, selectRDS: false,
361     selectConsistency: false, selectErrorRate: false, selectLatency:
362     false, selectElapsed Time: false, selectConnectionTime: false,
363     selectThroughput: false, selectCpu: false, selectMemory: false,
364     checkbox: true, filterAlert: "", serviceAlert: "", instanceAlert: "",
365     selectT2Micro: false, selectT2Nano: false, selectT2Small: false,
366     selectT2Medium: false, selectT2Large: false, selectT2XLarge: false,
367     calculateState: 0, response: []
368 };
369
370 this.hideComponent = this.hideComponent.bind(this);
371 this.handleClick = this.handleClick.bind(this);
372 this.handleCalculate = this.handleCalculate.bind(this);
373 this.calculateButton = this.calculateButton.bind(this);
374 }
375
376 calculateButton() {
377     switch(this.state.calculateState) {
378         case 0:
379             return (
380                 <div className = 'calculateDivStyle'>
```

```
377             <button className = 'calculateButtonStyle' onClick={()>
378               => {
379                 this.handleCalculate();
380               }>Save</button>
381           );
382       case 1:
383         return (
384           <div className = 'calculateDivStyle'>
385             <p className='alertStyle'>Processing...</p>
386           </div>
387         );
388       case 2:
389         return (
390           <Route render={({ history }) => (
391             <div className = 'calculateDivStyle'>
392               <button className = 'calculateButtonStyle' onClick
393                 ={() => {
394                   history.push({
395                     pathname: './results',
396                     state: { data: this.state.response, service
397 : this.getService() }
398                   })
399                 }>Show Results</button>
400               </div>
401             )}>
402         );
403       case 3:
404         return (
405           <div className = 'calculateDivStyle'>
406             <button className = 'calculateButtonStyle' onClick={()>
407               => {
408                 this.handleCalculate();
409               }>Save</button>
410               <p className='lastAlertStyle'>Processing failed. Try
again.</p>
411             </div>
412         );
413       default: console.log("Invalid calculate button state");
414     }
415   }
416 }
```

```
421
412     }
413 }
414
415 getService() {
416     if(this.state.selectEC2 === true)
417         return this.ec2;
418     else if(this.state.selectS3 === true)
419         return this.s3;
420     else if(this.state.selectRDS === true)
421         return this.rds;
422 }
423
424
425 validateEntry(checkboxVal, lowerLimit, upperLimit) {
426     var reg = new RegExp('^\d+$');
427     var regS3 = new RegExp('^\d+(\.\d+)?$');
428     var otherLimit;
429
430     if(this.state.selectEC2 === true)
431         otherLimit = 8;
432     else if(this.state.selectRDS === true)
433         otherLimit = 6;
434     if(this.state.selectS3 === true)
435         return regS3.test(checkboxVal) && parseFloat(checkboxVal) >=
lowerLimit && parseFloat(checkboxVal) <= upperLimit;
436     else
437         return reg.test(checkboxVal) && parseInt(checkboxVal) > 0 &&
parseInt(checkboxVal) <= otherLimit;
438 }
439
440 hideComponent(name) {
441     switch (name) {
442         case "showHideServices":
443             this.setState({ showHideServices: !this.state.showHideServices
});
444             break;
445         case "showHideFilters":
446             this.setState({ showHideFilters: !this.state.showHideFilters })
```

```
        ;  
447            break;  
448        case "showHideInstances":  
449            this.setState({ showHideInstances: !this.state.  
showHideInstances });  
450            break;  
451        default: console.log("Invalid show and hide scenario");  
452    }  
453}  
454  
455 handleClick(name) {  
456    switch(name) {  
457        case "EC2":  
458            this.setState({  
459                selectEC2: !this.state.selectEC2, selectS3: false,  
selectRDS: false});  
460            break;  
461        case "S3":  
462            this.setState({ selectS3: !this.state.selectS3, selectEC2:  
false, selectRDS: false });  
463            break;  
464        case "RDS":  
465            this.setState({ selectRDS: !this.state.selectRDS, selectEC2:  
false, selectS3: false });  
466            break;  
467        case "Consistency":  
468            this.setState({ selectConsistency: !this.state.  
selectConsistency });  
469            break;  
470        case "Error Rate":  
471            this.setState({ selectErrorRate: !this.state.selectErrorRate })  
472        ;  
473            break;  
474        case "Latency":  
475            this.setState({ selectLatency: !this.state.selectLatency });  
476            break;  
477        case "Elapsed Time":  
478            this.setState({ selectElapsedTime: !this.state.  
selectElapsedTime });  
479}
```

```
478         break;
479     case "Connection Time":
480         this.setState({ selectConnectionTime: !this.state.
481 selectConnectionTime });
482         break;
483     case "Throughput":
484         this.setState({ selectThroughput: !this.state.selectThroughput
485 });
486         break;
487     case "CPU Utilization":
488         this.setState({ selectCpu: !this.state.selectCpu });
489         break;
490     case "Memory":
491         this.setState({ selectMemory: !this.state.selectMemory });
492         break;
493     case "T2Nano":
494         this.setState({ selectT2Nano: !this.state.selectT2Nano });
495         break;
496     case "T2Micro":
497         this.setState({ selectT2Micro: !this.state.selectT2Micro });
498         break;
499     case "T2Small":
500         this.setState({ selectT2Small: !this.state.selectT2Small });
501         break;
502     case "T2Medium":
503         this.setState({ selectT2Medium: !this.state.selectT2Medium });
504         break;
505     case "T2Large":
506         this.setState({ selectT2Large: !this.state.selectT2Large });
507         break;
508     case "T2XLarge":
509         this.setState({ selectT2XLarge: !this.state.selectT2XLarge });
510         break;
511     default: console.log("null");
512   }
513
514   handleCalculate() {
```

```

515     this.setState({calculateState: 1});
516
517     var timestamp;
518     var filters = "";
519     var priorities = "";
520     var serviceType = "";
521     var instanceType = "";
522     if(this.state.selectS3!==true && this.state.selectRDS!==true && this.state.selectEC2!==true)
523     {
524         this.setState({ serviceAlert: "Please select at least one service.", calculateState: 0 });
525         return;
526     }
527     else
528         this.setState({ serviceAlert: "Service type noted!" });
529     if(this.state.selectEC2==true)
530     {
531         var ec2 = 0;
532         serviceType = "ec2";
533         if(this.state.selectT2Micro==true)
534             { instanceType += "t2.micro,"; ec2 = ec2 + 1; }
535         if(this.state.selectT2Nano==true)
536             { instanceType += "t2.nano,"; ec2 = ec2 + 1; }
537         if(this.state.selectT2Small==true)
538             { instanceType += "t2.small,"; ec2 = ec2 + 1; }
539         if(this.state.selectT2Medium==true)
540             { instanceType += "t2.medium,"; ec2 = ec2 + 1; }
541         if(this.state.selectT2Large==true)
542             { instanceType += "t2.large,"; ec2 = ec2 + 1; }
543         if(this.state.selectT2XLarge==true)
544             { instanceType += "t2.xlarge,"; ec2 = ec2 + 1; }
545         if(this.state.selectT2Micro==false && this.state.selectT2Nano==false && this.state.selectT2Small==false && this.state.selectT2Medium==false && this.state.selectT2Large==false && this.state.selectT2XLarge==false)
546         {
547             instanceType = "t2.nano,t2.micro,t2.small,t2.medium,t2.large,t2.xlarge,";
548             ec2 = 0;
549             this.setState({ instanceAlert: "" });
550         }
551         if(ec2 === 1)
552             this.setState({ instanceAlert: "Please select two or more services." });
553     }

```

```
instance types for comparison.", calculateState: 0 });

549     return;
550 }
551 else
552     this.setState({ instanceAlert: "Instance types noted!" });
553 }
554 else if(this.state.selectS3==true) {
555     this.setState({ selectCpu: false, selectMemory: false });
556     serviceType = "s3";
557     instanceType = "na,";
558 }
559 else if(this.state.selectRDS==true)
560 {
561     this.setState({ selectCpu: false, selectMemory: false,
562 selectT2Large: false, selectT2Nano: false, selectT2XLarge: false });
563     var rds = 0;
564     serviceType = "rds";
565     if(this.state.selectT2Micro==true)
566     { instanceType += "db.t2.micro,"; rds = rds + 1; }
567     if(this.state.selectT2Small==true)
568     { instanceType += "db.t2.small,"; rds = rds + 1; }
569     if(this.state.selectT2Medium==true)
570     { instanceType += "db.t2.medium,"; rds = rds + 1; }
571     if(this.state.selectT2Micro==false && this.state.selectT2Small==
572 false&& this.state.selectT2Medium==false)
573     {
574         instanceType = "db.t2.micro,db.t2.small,db.t2.medium,";
575         rds = 0;
576         this.setState({ instanceAlert: "" });
577     }
578     if(rds === 1) {
579         this.setState({ instanceAlert: "Please select two or more
580 instance types for comparison.", calculateState: 0 });
581         return;
582     }
583     else
584         this.setState({ instanceAlert: "Instance types noted!" });
585 }
```

```
584
585     //Filter processing begins
586
587     if(this.state.selectS3==true)
588     {
589         if(this.state.selectConsistency==true && this.state.
590             selectErrorRate==true && this.state.selectLatency==true
591             && this.state.selectElapsedTime==true && this.state.
592             selectConnectionTime==true && this.state.selectThroughput==true)
593             {
594                 this.setState({ filterAlert: "Please select all filters except
595                 the one that needs to be predicted.", calculateState: 0 });
596                 return;
597             }
598         if((document.getElementById("checkbox").checked==true || document.
599             getElementById("checkbox").checked==false) &&
600             (this.state.selectConsistency==false && this.state.selectErrorRate==
601             false && this.state.selectLatency==false
602                 && this.state.selectElapsedTime==false && this.state.
603                 selectConnectionTime==false && this.state.selectThroughput==false
604                 && this.state.selectCpu==false && this.state.selectMemory==false)
605         )
606         {
607             if(this.state.selectS3==true)
608             {
609                 this.setState({ filterAlert: "Please select all filters except
610                 the one that needs to be predicted.", calculateState: 0 });
611                 return;
612             }
613         }
614     else if (document.getElementById("checkbox").checked==true && (this.
615         state.selectConsistency==true
```

```

613           || this.state.selectErrorRate==true || this.state.selectLatency==
2455 true || this.state.selectElapsedTime==true
614           || this.state.selectConnectionTime==true || this.state.
615 selectThroughput==true || this.state.selectCpu==true
616           || this.state.selectMemory==true))
617 {
618     if (this.state.selectConsistency==true) { filters += "stdDev,";
priorities += "1," }
619     if (this.state.selectErrorRate==true) { filters += "errorRate,";
priorities += "1," }
620     if (this.state.selectLatency==true) { filters += "Latency,";
priorities += "1," }
621     if (this.state.selectElapsedTime==true) { filters += "elapsed,";
priorities += "1," }
622     if (this.state.selectConnectionTime==true) { filters += "Connect,";
priorities += "1," }
623     if (this.state.selectThroughput==true) { filters += "Throughput,";
priorities += "1," }
624     if (this.state.selectCpu==true) { filters += "CPU,"; priorities +=
"1," }
625     if (this.state.selectMemory==true) { filters += "Memory,";
priorities += "1," }
626     this.setState({ filterAlert: "Filters noted!" });
627   }
628   else if (document.getElementById("checkbox").checked==false && (this.
state.selectConsistency==true
629           || this.state.selectErrorRate==true || this.state.selectLatency==
true || this.state.selectElapsedTime==true
630           || this.state.selectConnectionTime==true || this.state.
selectThroughput==true || this.state.selectCpu==true
631           || this.state.selectMemory==true))
632   {
633     var successText = "Filters with priorities noted!";
634     var failureText = "For Amazon EC2, priorities should range from 1 -
8 and for Amazon RDS, priorities should range from 1 - 6 only.";
635     var failureTextS3 = "For Amazon S3, priorities can be double /
integer values ranging from ";
636     var consistencyLimit = 1000, errorLimit = 100, latencyLimit = 300,
elapsedLimit = 300, connectionLimit = 300, throughputLimit = 5000,

```

```
636     consistencyLowerLimit = 100, lowerLimit = 0;
637     var s3FilterCount = 0;
638
639     if (this.state.selectConsistency === true) {
640         if (this.validateEntry(document.getElementById("PrConsistency").value, consistencyLowerLimit, consistencyLimit)) {
641             filters += "stdDev,";
642             s3FilterCount++;
643             priorities += document.getElementById("PrConsistency").value + ",";
644         }
645     } else {
646         if (this.state.selectS3 === true)
647             this.setState({ filterAlert: failureTextS3 +
648 consistencyLowerLimit + " - " + consistencyLimit + " only for Consistency.",
649 calculateState: 0 });
650         else
651             this.setState({ filterAlert: failureText,
652 calculateState: 0 });
653     }
654     return;
655 }
656 }
657 if (this.state.selectErrorRate === true) {
658     if (this.validateEntry(document.getElementById("PrError").value, lowerLimit, errorLimit)) {
659         filters += "errorRate,";
660         s3FilterCount++;
661         priorities += document.getElementById("PrError").value + ",";
662     }
663     else {
664         if (this.state.selectS3 === true)
665             this.setState({ filterAlert: failureTextS3 + lowerLimit +
666 " - " + errorLimit + " only for Error Rate.", calculateState: 0 });
667         else
668             this.setState({ filterAlert: failureText,
669 calculateState: 0 });
670     }
671 }
```

```
667         if (this.state.selectLatency === true) {
668             if(this.validateEntry(document.getElementById("PrLatency") .
669             value, lowerLimit, latencyLimit)) {
670                 filters += "Latency,";
671                 s3FilterCount++;
672                 priorities += document.getElementById("PrLatency").value + ,
673                 ";
674             }
675             else {
676                 if(this.state.selectS3 === true)
677                     this.setState({ filterAlert: failureTextS3 + lowerLimit
678 + " - " + latencyLimit + " only for Latency.", calculateState: 0 });
679                 else
680                     this.setState({ filterAlert: failureText,
681 calculateState: 0 });
682                 return;
683             }
684         }
685         if (this.state.selectElapsedTime === true) {
686             if(this.validateEntry(document.getElementById("PrElapsed") .
687             value, lowerLimit, elapsedLimit)) {
688                 filters += "elapsed,";
689                 s3FilterCount++;
690                 priorities += document.getElementById("PrElapsed").value + ,
691                 ";
692             } else {
693                 if(this.state.selectS3 === true)
694                     this.setState({ filterAlert: failureTextS3 + lowerLimit
695 + " - " + elapsedLimit + " only for Elapsed Time.", calculateState: 0 });
696                 else
697                     this.setState({ filterAlert: failureText,
698 calculateState: 0 });
699                 return;
700             }
701         }
702         if (this.state.selectConnectionTime === true) {
703             if(this.validateEntry(document.getElementById("PrConnection") .
704             value, lowerLimit, connectionLimit)) {
705                 filters += "Connect,";
```

```
697             s3FilterCount++;
698             priorities += document.getElementById("PrConnection").value
699             +" , ";
700         }
701         else {
702             if(this.state.selectS3 === true)
703                 this.setState({ filterAlert: failureTextS3 + lowerLimit
704 + " - " + connectionLimit + " only for Connection Time.", calculateState:
705 0 });
706             else
707                 this.setState({ filterAlert: failureText ,
708 calculateState: 0 });
709             return;
710         }
711     }
712     if (this.state.selectThroughput==true) {
713         if(this.validateEntry(document.getElementById("PrThroughput").value,
714 lowerLimit, throughputLimit)) {
715             filters += "Throughput , ";
716             s3FilterCount++;
717             priorities += document.getElementById("PrThroughput").value
718             +" , ";
719         }
720         else {
721             if(this.state.selectS3 === true)
722                 this.setState({ filterAlert: failureTextS3 + lowerLimit
723 + " - " + throughputLimit + " only for Throughput.", calculateState: 0 });
724             else
725                 this.setState({ filterAlert: failureText ,
726 calculateState: 0 });
727             return;
728         }
729     }
730     if (this.state.selectCpu==true) {
731         if(this.validateEntry(document.getElementById("PrCpu").value,
732 0, 0)) {
733             filters += "CPU , ";
734             priorities += document.getElementById("PrCpu").value+" , ";
735         }
736     }
737 }
```

```
727         else {
728             this.setState({ filterAlert: failureText, calculateState: 0
729         });
730         return;
731     }
732     if (this.state.selectMemory === true) {
733         if(this.validateEntry(document.getElementById("PrMemory")).value
734 , 0, 0)) {
735             filters += "Memory ,";
736             priorities += document.getElementById("PrMemory").value+","
737 ;
738         }
739     else {
740         this.setState({ filterAlert: failureText, calculateState: 0
741 });
742         return;
743     }
744     if(this.state.selectS3 === true)
745     {
746         if(s3FilterCount !== 5)
747         {
748             this.setState({ filterAlert: "Please select all filters
749 except the one that needs to be predicted.", calculateState: 0 });
750             return;
751         }
752         this.setState({ filterAlert: successText });
753     }
754     var moment = require('moment');
755     timestamp = moment(Date.now()).format("DD-MM-YYYY h:mm:ss");
756
757     var frontendData = {
758         'timestamp': timestamp,
759         'filters': filters.substring(0, filters.length - 1),
760         'priorities': priorities.substring(0, priorities.length - 1),
```

```
761         'instanceType': instanceType.substring(0, instanceType.length - 1),  
762         'serviceType': serviceType  
763     }  
764  
765     frontendData = JSON.stringify(frontendData);  
766     console.log("FrontEndData", frontendData);  
767  
768     this.getResults(frontendData, 3, 3000)  
769 }  
770  
771 getResults(frontendData, retries, backoff) {  
772  
773     const postDataUrl = "https://cloud-service-optimizer-dev.herokuapp.com/  
front-end/post-process-filters/";  
774     const getDataUrl = "https://cloud-service-optimizer-dev.herokuapp.com/  
front-end/get-results?processingNumber=";  
775  
776     axios({  
777         method: 'post',  
778         url: postDataUrl,  
779         headers: {  
780             'Content-type': 'application/json',  
781         },  
782         data: frontendData  
783     })  
784     .then(postJsonData => {  
785         console.log("pNo. obtained via POST", postJsonData.data);  
786  
787         var postResponse = postJsonData.data  
788  
789         setTimeout(() => {  
790             console.log("Initiating a delay of:", backoff);  
791             axios.get(getDataUrl + postResponse.message)  
792             .then(getJsonData => {  
793                 console.log("GET Response (payload)", getJsonData);  
794  
795                 var getResponse = getJsonData.data;  
796  
797                 if(getResponse.status === "success") {  
                     this.setState({calculateState: 2, response: getResponse}
```

```
    .payload})
  }
  else {
    this.setState({calculateState: 2, response: this.
defaultResponse[this.getService()]})
    // this.setState({calculateState: 3})
  }
}
.catch(error => {
  console.log(error);
  if(retries > 0) {
    var retriesLeft = retries - 1;
    console.log("Failure in GET, no. of retries left:",
retriesLeft);
    this.getResults(frontendData, retriesLeft, backoff +
500);
  }
  else {
    this.setState({calculateState: 2, response: this.
defaultResponse[this.getService()]})
    // this.setState({calculateState: 3})
  }
})
}, backoff)
})
.catch(error => {
  console.log(error);
  if(retries > 0) {
    var retriesLeft = retries - 1;
    console.log("Failure in POST, no. of retries left:",
retriesLeft);
    this.getResults(frontendData, retriesLeft, backoff + 500);
  }
  else {
    this.setState({calculateState: 2, response: this.
defaultResponse[this.getService()]})
    // this.setState({calculateState: 3})
  }
})
```

```
830     }
831
832
833     render() {
834         const { showHideServices, showHideFilters, showHideInstances, selectEC2
835 , selectS3, selectRDS,
836             selectConsistency, selectErrorRate, selectLatency,
837             selectElapsedTime, selectConnectionTime, selectThroughput, selectCpu,
838             selectMemory,
839             filterAlert, serviceAlert, instanceAlert,
840             selectT2Micro, selectT2Nano, selectT2Small, selectT2Medium,
841             selectT2Large, selectT2XLarge } = this.state;
842
843         return(
844
845             <div>
846                 <div className = 'titleDivStyle3'>
847                     <h1 className = 'titleStyle3'>Dashboard</h1>
848                 </div>
849                 <div>
850                     <div className = 'questionDivStyle'>
851                         <p className = 'questionStyle'>What AWS Service are you
852             looking for?</p>
853             <button className = 'selectButtonStyle' onClick={() =>
854             this.hideComponent("showHideServices")}>Select Service</button>
855             </div>
856             { showHideServices && (
857                 <div className = 'serviceDivStyle'>
858                     <button className={selectEC2 ? "ec2StyleTrue": "
859             ec2StyleFalse} onClick={() => this.handleClick("EC2")}>{ this.ec2 }</
860             button>
861
862                     <button className={selectS3 ? "s3StyleTrue": "
863             s3StyleFalse} onClick={() => this.handleClick("S3")}>{ this.s3 }</button>
864
865                     <button className={selectRDS ? "rdsStyleTrue": "
866             rdsStyleFalse} onClick={() => this.handleClick("RDS")}>{ this.rds }</
867             button>
868
869                     <div>
870                         <p className='alertStyle'>{serviceAlert}</p>
871                     </div>
872                 </div> ) }
```

```

858         </div>
859
860         <div>
861             <div className = 'questionDivStyle'>
862                 <p className = 'questionStyle'>{ !selectS3 ? 'What
2770 performance metrics do you want us to focus on?' : 'What metric should ML
863 prediction be applied to?' }</p>
864                 <button className = 'selectButtonStyle' onClick={() => this
865 .hideComponent("showHideFilters")}>Select Filters</button>
866             </div>
867             { showHideFilters && (
868                 <div className = 'serviceDivStyle'>
869                     <div>
870                         <input type = { selectS3 ? "hidden" : "checkbox"
871 " } id = "checkbox"/>
872                         <label className='checkboxStyle'>{ selectS3 ?
873                             "" : "Assign same priority to all filters" }</label><br/>
874                     </div>
875                     <div style={{display: 'inline'}}>
876                         <div style={{display: 'inline'}}>
877                             <button className={selectConsistency ? "
878 consistencyStyleTrue": "consistencyStyleFalse"} onClick={() => this.
879 handleClick("Consistency")}>Consistency</button>
880                             { (selectConsistency && document.getElementById(
881 "checkbox").checked !== true &&
882                                 <input className='priorityTextStyle' type="text"
883 id="PrConsistency" placeholder="Enter Priority" required/>
884                             )} </div>
885                     </div>
886                     <div style={{display: 'inline'}}>
887                         <button className={selectErrorRate ? "
888 errorStyleTrue": "errorStyleFalse"} onClick={() => this.handleClick("Error
889 Rate")}>Error Rate</button>
890                         { (selectErrorRate && document.getElementById("checkbox")
891 checked !== true &&
892                                 <input className='priorityTextStyle' type="text"
893 id="PrError" placeholder="Enter Priority" required/>
894                         )} </div>
895                     </div>
896                     <div style={{display: 'inline'}}>

```

```
884             <button className={selectLatency ? "latencyStyleTrue": "latencyStyleFalse"} onClick={() => this.handleClick("Latency")}>Latency</button>
885             { (selectLatency && document.getElementById("checkbox").checked !== true) &&
886               <input className='priorityTextStyle' type="text" id="PrLatency" placeholder="Enter Priority" required/>
887             )
888           </div>
889           <div style={{display: 'inline'}} >
890             <button className={selectElapsedTime ? "elapsedStyleTrue": "elapsedStyleFalse"} onClick={() => this.handleClick("Elapsed Time")}>Elapsed Time</button>
891             { (selectElapsedTime && document.getElementById("checkbox").checked !== true) &&
892               <input className='priorityTextStyle' type="text" id="PrElapsed" placeholder="Enter Priority" required/>
893             )
894           </div>
895
896           <div style={{display: 'inline'}}>
897             <button className={selectConnectionTime ? "connectionStyleTrue": "connectionStyleFalse"} onClick={() => this.handleClick("Connection Time")}>Connection Time</button>
898             { (selectConnectionTime && document.
899               getElementById("checkbox").checked !== true) &&
900               <input className='priorityTextStyle' type="text" id="PrConnection" placeholder="Enter Priority" required/>
901             )
902           </div>
903           <div style={{display: 'inline'}}>
904             <button className={selectThroughput ? "throughputStyleTrue": "throughputStyleFalse"} onClick={() => this.handleClick("Throughput")}>Throughput</button>
905             { (selectThroughput && document.getElementById("checkbox").checked !== true) &&
906               <input className='priorityTextStyle' type="text" id="PrThroughput" placeholder="Enter Priority" required/>
907             )
908           </div>
```

```
907             </div>
908             { selectEC2 && (
909                 <div style={{display: 'inline'}}>
910                     <button className={selectCpu ? "cpuStyleTrue": "cpuStyleFalse"} onClick={() => this.handleClick("CPU Utilization")}>CPU
911                         Utilization</button>
912                         {(selectCpu && document.getElementById("checkbox").checked!==true &&
913                             <input className='priorityTextStyle' type="text" id="PrCpu" placeholder="Enter Priority" required/>
914                         )}>
915                     </div>
916             )}>
917             { selectEC2 && (
918                 <div style={{display: 'inline'}}>
919                     <button className={selectMemory ? "memoryStyleTrue": "memoryStyleFalse"} onClick={() => this.handleClick("Memory")}>Memory</button>
920                         {(selectMemory && document.getElementById("checkbox").checked!==true &&
921                             <input className='priorityTextStyle' type="text" id="PrMemory" placeholder="Enter Priority" required/>
922                         )}>
923                     </div>
924             )}>
925             <div>
926                 <p className='alertStyle'>{filterAlert}</p>
927             </div>
928         </div> )
929     </div>
930     { !selectS3 && (
931         <div>
932             <div className = 'questionDivStyle'>
933                 <p className = 'questionStyle'>What Instance type of
934                     the above listed AWS Service do you want?</p>
935                     <button className = 'selectButtonStyle' onClick={() =>
this.hideComponent("showHideInstances")}>Select Instance</button>
936             </div>
```

```

936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
    {
        selectEC2 && showHideInstances && (
            <div className = 'serviceDivStyle'>
                <button className={selectT2Nano ? "t2NanoStyleTrue"
: "t2NanoStyleFalse"} onClick={() => this.handleClick("T2Nano")}>t2.nano</
button>

                <button className={selectT2Micro ? "
t2MicroStyleTrue": "t2MicroStyleFalse"} onClick={() => this.handleClick("
T2Micro")}>t2.micro</button>

                <button className={selectT2Small ? "
t2SmallStyleTrue": "t2SmallStyleFalse"} onClick={() => this.handleClick("
T2Small")}>t2.small</button>

                <button className={selectT2Medium ? "
t2MediumStyleTrue": "t2MediumStyleFalse"} onClick={() => this.handleClick("
T2Medium")}>t2.medium</button>

                <button className={selectT2Large ? "
t2LargeStyleTrue": "t2LargeStyleFalse"} onClick={() => this.handleClick("
T2Large")}>t2.large</button>

                <button className={selectT2XLarge ? "
t2XLargeStyleTrue": "t2XLargeStyleFalse"} onClick={() => this.handleClick("
T2XLarge")}>t2.xlarge</button>

                <p className='alertStyle'>{instanceAlert}</p>
            </div>
        )
    }

    {
        selectRDS && showHideInstances && (
            <div className = 'serviceDivStyle'>
                <button className={selectT2Micro ? "
t2MicroStyleTrue": "t2MicroStyleFalse"} onClick={() => this.handleClick("
T2Micro")}>db.t2.micro</button>

                <button className={selectT2Small ? "
t2SmallStyleTrue": "t2SmallStyleFalse"} onClick={() => this.handleClick("
T2Small")}>db.t2.small</button>

                <button className={selectT2Medium ? "
t2MediumStyleTrue": "t2MediumStyleFalse"} onClick={() => this.handleClick("
T2Medium")}>db.t2.medium</button>

                <p className='alertStyle'>{instanceAlert}</p>
            </div>
        )
    }

```

```
957             </div>
958         )
959     }
960     </div>
961   )} {
962     { this.calculateButton() }
963   </div>
964
965 );
966 }
967
968 export default Dashboard;
```

2935 **Dashboard.css**

```
1 .titleStyle3 {
2   margin-top: 0%;
2943   font-size: 250%;
4   color: #FDFDFD;
5   text-shadow: 2px 2px 2px #222;
6   text-align: center;
7   padding: 2%;
2948   margin-bottom: 2%;
8 }
9
10
11 .titleDivStyle3 {
12   background-color: #F9CC88;
2953   width: 23%;
14   margin-left: auto;
15   margin-right: auto;
16   border-bottom-left-radius: 50px;
17   border-bottom-right-radius: 50px;
2958   padding-bottom: 0.4%;
19   margin-bottom: 5%;
20 }
21
22 .questionStyle {
```

```
23     font-size: 180%;  
24     font-family: 'Arial Narrow Bold';  
25     width: 78%;  
26     text-align: left;  
27     margin-left: 2%;  
28     color: #FDFDFD;  
29 }  
30  
31 .questionDivStyle {  
32     align-self: center;  
33     display: flex;  
34     background-color: #88B7F9;  
35     margin-bottom: 1%;  
36     width: 70%;  
37     border-radius: 50px;  
38     height: 3%;  
39     margin-left: auto;  
40     margin-right: auto;  
41 }  
42  
43 .selectButtonStyle {  
44     height: 20%;  
45     width: 15%;  
46     padding-top: 10px;  
47     padding-bottom: 10px;  
48     border-width: 0px;  
49     border-radius: 20px;  
50     margin-top: 2%;  
51     font-family: 'Arial Narrow Bold' ;  
52     font-weight: bold;  
53     font-size: 130%;  
54     color: #FDFDFD;  
55     background-color: #F9CC88 ;  
56  
57 }  
58  
59 .serviceDivStyle {  
60     background-color: #FDFDFD;  
61     width: 70%;
```

```
62     margin-left: auto;
63     margin-right: auto;
64     margin-bottom: 1%;
65     border-radius: 40px;
66     text-align: center;
67     padding-top: 3%;
68 }
69
70 .s3StyleTrue {
71     background-color: #FFA500;
72     color: #FDFDFD;
73     padding-top: 10px;
74     padding-bottom: 10px;
75     border-width: 0px;
76     border-radius: 20px;
77     width: 15%;
78     margin: 1%;
79     font-family: 'Arial Narrow Bold' ;
80     font-weight: bold;
81     font-size: 130%;
82     color: #FDFDFD;
83     margin-top: 3%;
84 }
85
86 .s3StyleFalse {
87     background-color: #F9CC88;
88     color: #FDFDFD;
89     padding-top: 10px;
90     padding-bottom: 10px;
91     border-width: 0px;
92     border-radius: 20px;
93     width: 15%;
94     margin: 1%;
95     font-family: 'Arial Narrow Bold' ;
96     font-weight: bold;
97     font-size: 130%;
98     color: #FDFDFD;
99     margin-top: 3%;
100 }
```

```
101
102 .rdsStyleTrue {
103     background-color: #FFA500;
104     color: #FDFDFD;
105     padding-top: 10px;
106     padding-bottom: 10px;
107     border-width: 0px;
108     border-radius: 20px;
109     width: 15%;
110     margin: 1%;
111     font-family: 'Arial Narrow Bold' ;
112     font-weight: bold;
113     font-size: 130%;
114     color: #FDFDFD;
115     margin-top: 3%;
116 }
117
118 .rdsStyleFalse {
119     background-color: #F9CC88;
120     color: #FDFDFD;
121     padding-top: 10px;
122     padding-bottom: 10px;
123     border-width: 0px;
124     border-radius: 20px;
125     width: 15%;
126     margin: 1%;
127     font-family: 'Arial Narrow Bold' ;
128     font-weight: bold;
129     font-size: 130%;
130     color: #FDFDFD;
131     margin-top: 3%;
132 }
133
134 .ec2StyleTrue {
135     background-color: #FFA500;
136     color: #FDFDFD;
137     padding-top: 10px;
138     padding-bottom: 10px;
139     border-width: 0px;
```

```
140     border-radius: 20px;  
141     width: 15%;  
142     margin: 1%;  
143     font-family: 'Arial Narrow Bold' ;  
144     font-weight: bold;  
145     font-size: 130%;  
146     color: #FDFDFD;  
147     margin-top: 3%;  
148 }  
149  
150 .ec2StyleFalse {  
151     background-color: #F9CC88;  
152     color: #FDFDFD;  
153     padding-top: 10px;  
154     padding-bottom: 10px;  
155     border-width: 0px;  
156     border-radius: 20px;  
157     width: 15%;  
158     margin: 1%;  
159     font-family: 'Arial Narrow Bold' ;  
160     font-weight: bold;  
161     font-size: 130%;  
162     color: #FDFDFD;  
163     margin-top: 3%;  
164 }  
165  
166 .consistencyStyleTrue {  
167     background-color: #FFA500;  
168     color: #FDFDFD;  
169     padding-top: 10px;  
170     padding-bottom: 10px;  
171     border-width: 0px;  
172     border-radius: 20px;  
173     width: 17%;  
174     margin: 1%;  
175     font-family: 'Arial Narrow Bold' ;  
176     font-weight: bold;  
177     font-size: 130%;  
178     color: #FDFDFD;
```

```
179     margin-top: 2%;  
180  
181 }  
182  
183 .consistencyStyleFalse {  
184     background-color: #F9CC88;  
185     color: #FDFDFD;  
186     padding-top: 10px;  
187     padding-bottom: 10px;  
188     border-width: 0px;  
189     border-radius: 20px;  
190     width: 17%;  
191     margin: 1%;  
192     font-family: 'Arial Narrow Bold' ;  
193     font-weight: bold;  
194     font-size: 130%;  
195     color: #FDFDFD;  
196     margin-top: 2%;  
197  
198 }  
199  
200 .errorStyleTrue {  
201     background-color: #FFA500;  
202     color: #FDFDFD;  
203     padding-top: 10px;  
204     padding-bottom: 10px;  
205     border-width: 0px;  
206     border-radius: 20px;  
207     width: 17%;  
208     margin: 1%;  
209     font-family: 'Arial Narrow Bold' ;  
210     font-weight: bold;  
211     font-size: 130%;  
212     color: #FDFDFD;  
213     margin-top: 2%;  
214 }  
215  
216 .errorStyleFalse {  
217     background-color: #F9CC88;
```

```
218     color: #FDFDFD;  
219     padding-top: 10px;  
220     padding-bottom: 10px;  
221     border-width: 0px;  
222     border-radius: 20px;  
223     width: 17%;  
224     margin: 1%;  
225     font-family: 'Arial Narrow Bold' ;  
226     font-weight: bold;  
227     font-size: 130%;  
228     color: #FDFDFD;  
229     margin-top: 2%;  
230 }  
231  
232 .latencyStyleTrue {  
233     background-color: #FFA500;  
234     color: #FDFDFD;  
235     padding-top: 10px;  
236     padding-bottom: 10px;  
237     border-width: 0px;  
238     border-radius: 20px;  
239     width: 17%;  
240     margin: 1%;  
241     font-family: 'Arial Narrow Bold' ;  
242     font-weight: bold;  
243     font-size: 130%;  
244     color: #FDFDFD;  
245     margin-top: 2%;  
246 }  
247  
248 .latencyStyleFalse {  
249     background-color: #F9CC88;  
250     color: #FDFDFD;  
251     padding-top: 10px;  
252     padding-bottom: 10px;  
253     border-width: 0px;  
254     border-radius: 20px;  
255     width: 17%;  
256     margin: 1%;
```

```
257     font-family: 'Arial Narrow Bold' ;
258     font-weight: bold;
259     font-size: 130%;
260     color: #FDFDFD;
261     margin-top: 2%;
262 }
263
264 .elapsedStyleTrue {
265     background-color: #FFA500;
266     color: #FDFDFD;
267     padding-top: 10px;
268     padding-bottom: 10px;
269     border-width: 0px;
270     border-radius: 20px;
271     width: 17%;
272     margin: 1%;
273     font-family: 'Arial Narrow Bold' ;
274     font-weight: bold;
275     font-size: 130%;
276     color: #FDFDFD;
277     margin-top: 2%;
278 }
279
280 .elapsedStyleFalse {
281     background-color: #F9CC88;
282     color: #FDFDFD;
283     padding-top: 10px;
284     padding-bottom: 10px;
285     border-width: 0px;
286     border-radius: 20px;
287     width: 17%;
288     margin: 1%;
289     font-family: 'Arial Narrow Bold' ;
290     font-weight: bold;
291     font-size: 130%;
292     color: #FDFDFD;
293     margin-top: 2%;
294 }
```

```
296 .connectionStyleTrue {  
297     background-color: #FFA500;  
298     color: #FDFDFD;  
299     padding-top: 10px;  
300     padding-bottom: 10px;  
301     border-width: 0px;  
302     border-radius: 20px;  
303     width: 17%;  
304     margin: 1%;  
305     font-family: 'Arial Narrow Bold' ;  
306     font-weight: bold;  
307     font-size: 130%;  
308     color: #FDFDFD;  
309     margin-top: 5%;  
310 }  
311  
312 .connectionStyleFalse {  
313     background-color: #F9CC88;  
314     color: #FDFDFD;  
315     padding-top: 10px;  
316     padding-bottom: 10px;  
317     border-width: 0px;  
318     border-radius: 20px;  
319     width: 17%;  
320     margin: 1%;  
321     font-family: 'Arial Narrow Bold' ;  
322     font-weight: bold;  
323     font-size: 130%;  
324     color: #FDFDFD;  
325     margin-top: 5%;  
326 }  
327  
328  
329 .throughputStyleTrue {  
330     background-color: #FFA500;  
331     color: #FDFDFD;  
332     padding-top: 10px;  
333     padding-bottom: 10px;  
334     border-width: 0px;
```

```
335     border-radius: 20px;  
336     width: 17%;  
337     margin: 1%;  
338     font-family: 'Arial Narrow Bold' ;  
339     font-weight: bold;  
340     font-size: 130%;  
341     color: #FDFDFD;  
342     margin-top: 5%;  
343 }  
344  
345 .throughputStyleFalse {  
346     background-color: #F9CC88;  
347     color: #FDFDFD;  
348     padding-top: 10px;  
349     padding-bottom: 10px;  
350     border-width: 0px;  
351     border-radius: 20px;  
352     width: 17%;  
353     margin: 1%;  
354     font-family: 'Arial Narrow Bold' ;  
355     font-weight: bold;  
356     font-size: 130%;  
357     color: #FDFDFD;  
358     margin-top: 5%;  
359 }  
360  
361 .cpuStyleTrue {  
362     background-color: #FFA500;  
363     color: #FDFDFD;  
364     padding-top: 10px;  
365     padding-bottom: 10px;  
366     border-width: 0px;  
367     border-radius: 20px;  
368     width: 17%;  
369     margin: 1%;  
370     font-family: 'Arial Narrow Bold' ;  
371     font-weight: bold;  
372     font-size: 130%;  
373     color: #FDFDFD;
```

```
374     margin-top: 2%;  
375 }  
376  
377 .cpuStyleFalse {  
378     background-color: #F9CC88;  
379     color: #FDFDFD;  
380     padding-top: 10px;  
381     padding-bottom: 10px;  
382     border-width: 0px;  
383     border-radius: 20px;  
384     width: 17%;  
385     margin: 1%;  
386     font-family: 'Arial Narrow Bold' ;  
387     font-weight: bold;  
388     font-size: 130%;  
389     color: #FDFDFD;  
390     margin-top: 2%;  
391 }  
392  
393 .memoryStyleTrue {  
394     background-color: #FFA500;  
395     color: #FDFDFD;  
396     padding-top: 10px;  
397     padding-bottom: 10px;  
398     border-width: 0px;  
399     border-radius: 20px;  
400     width: 17%;  
401     margin: 1%;  
402     font-family: 'Arial Narrow Bold' ;  
403     font-weight: bold;  
404     font-size: 130%;  
405     color: #FDFDFD;  
406     margin-top: 2%;  
407 }  
408  
409 .memoryStyleFalse {  
410     background-color: #F9CC88;  
411     color: #FDFDFD;  
412     padding-top: 10px;
```

```
413     padding-bottom: 10px;  
414     border-width: 0px;  
415     border-radius: 20px;  
416     width: 17%;  
417     margin: 1%;  
418     font-family: 'Arial Narrow Bold' ;  
419     font-weight: bold;  
420     font-size: 130%;  
421     color: #FDFDFD;  
422     margin-top: 2%;  
423 }  
424  
425 .t2NanoStyleTrue {  
426     background-color: #FFA500;  
427     color: #FDFDFD;  
428     padding-top: 10px;  
429     padding-bottom: 10px;  
430     border-width: 0px;  
431     border-radius: 20px;  
432     width: 15%;  
433     margin: 1%;  
434     font-family: 'Arial Narrow Bold' ;  
435     font-weight: bold;  
436     font-size: 130%;  
437     color: #FDFDFD;  
438 }  
439 }  
440  
441 .t2NanoStyleFalse {  
442     background-color: #F9CC88;  
443     color: #FDFDFD;  
444     padding-top: 10px;  
445     padding-bottom: 10px;  
446     border-width: 0px;  
447     border-radius: 20px;  
448     width: 15%;  
449     margin: 1%;  
450     font-family: 'Arial Narrow Bold' ;  
451     font-weight: bold;
```

```
452     font-size: 130%;  
453     color: #FDFDFD;  
454  
455 }  
456  
457 .t2MicroStyleTrue {  
458     background-color: #FFA500;  
459     color: #FDFDFD;  
460     padding-top: 10px;  
461     padding-bottom: 10px;  
462     border-width: 0px;  
463     border-radius: 20px;  
464     width: 15%;  
465     margin: 1%;  
466     font-family: 'Arial Narrow Bold' ;  
467     font-weight: bold;  
468     font-size: 130%;  
469     color: #FDFDFD;  
470  
471 }  
472  
473 .t2MicroStyleFalse {  
474     background-color: #F9CC88;  
475     color: #FDFDFD;  
476     padding-top: 10px;  
477     padding-bottom: 10px;  
478     border-width: 0px;  
479     border-radius: 20px;  
480     width: 15%;  
481     margin: 1%;  
482     font-family: 'Arial Narrow Bold' ;  
483     font-weight: bold;  
484     font-size: 130%;  
485     color: #FDFDFD;  
486  
487 }  
488  
489 .t2SmallStyleTrue {  
490     background-color: #FFA500;
```

```
491     color: #FDFDFD;  
492     padding-top: 10px;  
493     padding-bottom: 10px;  
494     border-width: 0px;  
495     border-radius: 20px;  
496     width: 15%;  
497     margin: 1%;  
498     font-family: 'Arial Narrow Bold' ;  
499     font-weight: bold;  
500     font-size: 130%;  
501     color: #FDFDFD;  
502  
503 }  
504  
505 .t2SmallStyleFalse {  
506     background-color: #F9CC88;  
507     color: #FDFDFD;  
508     padding-top: 10px;  
509     padding-bottom: 10px;  
510     border-width: 0px;  
511     border-radius: 20px;  
512     width: 15%;  
513     margin: 1%;  
514     font-family: 'Arial Narrow Bold' ;  
515     font-weight: bold;  
516     font-size: 130%;  
517     color: #FDFDFD;  
518  
519 }  
520  
521 .t2MediumStyleTrue {  
522     background-color: #FFA500;  
523     color: #FDFDFD;  
524     padding-top: 10px;  
525     padding-bottom: 10px;  
526     border-width: 0px;  
527     border-radius: 20px;  
528     width: 15%;  
529     margin: 1%;
```

```
530     font-family: 'Arial Narrow Bold' ;
531     font-weight: bold;
532     font-size: 130%;
533     color: #FDFDFD;
534
535 }
536
537 .t2MediumStyleFalse {
538     background-color: #F9CC88;
539     color: #FDFDFD;
540     padding-top: 10px;
541     padding-bottom: 10px;
542     border-width: 0px;
543     border-radius: 20px;
544     width: 15%;
545     margin: 1%;
546     font-family: 'Arial Narrow Bold' ;
547     font-weight: bold;
548     font-size: 130%;
549     color: #FDFDFD;
550 }
551
552 .t2LargeStyleTrue {
553     background-color: #FFA500;
554     color: #FDFDFD;
555     padding-top: 10px;
556     padding-bottom: 10px;
557     border-width: 0px;
558     border-radius: 20px;
559     width: 15%;
560     margin: 1%;
561     font-family: 'Arial Narrow Bold' ;
562     font-weight: bold;
563     font-size: 130%;
564     color: #FDFDFD;
565     margin-bottom: 5%;
566 }
567
568 .t2LargeStyleFalse {
```

```
569     background-color: #F9CC88;  
570     color: #FDFDFD;  
571     padding-top: 10px;  
572     padding-bottom: 10px;  
573     border-width: 0px;  
574     border-radius: 20px;  
575     width: 15%;  
576     margin: 1%;  
577     font-family: 'Arial Narrow Bold' ;  
578     font-weight: bold;  
579     font-size: 130%;  
580     color: #FDFDFD;  
581     margin-bottom: 5%;  
582 }  
  
583  
584 .t2XLargeStyleTrue {  
585     background-color: #FFA500;  
586     color: #FDFDFD;  
587     padding-top: 10px;  
588     padding-bottom: 10px;  
589     border-width: 0px;  
590     border-radius: 20px;  
591     width: 15%;  
592     margin: 1%;  
593     font-family: 'Arial Narrow Bold' ;  
594     font-weight: bold;  
595     font-size: 130%;  
596     color: #FDFDFD;  
597     margin-top: 2%;  
598 }  
  
599  
600 .t2XLargeStyleFalse {  
601     background-color: #F9CC88;  
602     color: #FDFDFD;  
603     padding-top: 10px;  
604     padding-bottom: 10px;  
605     border-width: 0px;  
606     border-radius: 20px;  
607     width: 15%;
```

```
608     margin: 1%;  
609     font-family: 'Arial Narrow Bold' ;  
610     font-weight: bold;  
611     font-size: 130%;  
612     color: #FDFDFD;  
613     margin-top: 2%;  
614 }  
615  
616 .calculateDivStyle {  
617     text-align: center;  
618 }  
619  
620 .calculateButtonStyle {  
621     width: 12%;  
622     padding-top: 10px;  
623     padding-bottom: 10px;  
624     border-width: 0px;  
625     border-radius: 20px;  
626     margin-top: 2%;  
627     font-family: 'Arial Narrow Bold' ;  
628     font-weight: bold;  
629     font-size: 130%;  
630     background-color: #F9CC88 ;  
631     color: #FDFDFD;  
632     margin-bottom: 3%;  
633 }  
634  
635 .checkboxStyle {  
636     color: #696969;  
637     text-align: left;  
638     font-family: 'Arial Narrow Bold' ;  
639     font-weight: bold;  
640     font-size: 130%;  
641     margin-right: 20px;  
642     margin-bottom: 20px;  
643 }  
644  
645 .submitButtonStyle {  
646     margin-top: 4%;
```

```
647     border-width: 0px;
648     border-radius: 20px;
649     width: 10%;
650     background-color: #88B7F9;
651     color: #FDFDFD;
652     padding-top: 10px;
653     padding-bottom: 10px;
654     font-family: 'Arial Narrow Bold' ;
655     font-weight: bold;
656     font-size: 130%;
657 }
658
659 .alertStyle {
660     color: #696969;
661     text-align: left;
662     font-family: 'Arial Narrow Bold';
663     font-weight: bold;
664     font-size: 130%;
665     text-align: center;
666     margin-top: 3%;
667     padding: 2%;
668 }
669
670 .lastAlertStyle {
671     color: #696969;
672     text-align: left;
673     font-family: 'Arial Narrow Bold';
674     font-weight: bold;
675     font-size: 130%;
676     text-align: center;
677     margin-top: 0.7%;
678     padding: 2%;
679 }
680
681 .priorityTextStyle {
682     background-color: #D3D3D3;
683     padding-top: 10px;
684     padding-bottom: 10px;
685     border-width: 0px;
```

```
686     border-radius: 20px;  
687     width: 10%;  
688     font-family: 'Arial Narrow Bold';  
689     font-weight: bold;  
690     text-align: center;  
691     font-size: 100%;  
692 }  
3630
```

## Results.js

```
1 import { Component } from 'react';  
3632  
3 import {Link} from 'react-router-dom'  
4 import { Doughnut, Bar } from 'react-chartjs-2';  
5 import './styles/Results.css'  
6  
6 class Results extends Component {  
3640  
8     textUtil(filterCompare) {  
9         var components = [];  
10        components.push(  
11            <p className = 'filterHeadingStyle'>Observations</p>  
3642        );  
13        Object.entries(filterCompare).forEach(item => {  
14            var instanceName = item[0];  
15            var instanceValue = item[1];  
16            components.push(  
3657                <div className = 's3DivStyle'>  
18                    <div className = 'instanceNameStyle'>{ instanceName }</div>  
19                    <div className = 'instanceValueStyle'>{ instanceValue + "%" }  
3660                </div>  
20                    </div>  
3664                );  
21        })  
22        return components;  
23    }  
24  
3666    s3TextUtil(payload) {
```

```
27     var components = [];
28     components.push(
29       <p className = 's3FilterHeadingStyle'>Selected Filters</p>
30     );
31     Object.entries(payload.selectedValues).forEach(item => {
32       var filterName = item[0];
33       var filterValue = item[1];
34       components.push(
35         <div className = 's3DivStyle'>
36           <div className = 'selectedS3FilterNameStyle'>{ filterName
37           }</div>
38           <div className = 'selectedS3FilterValueStyle'>{ filterValue
39           }</div>
40         </div>
41     );
42     components.push(
43       <p className = 's3FilterHeadingStyle'>Predicted Filter</p>
44     );
45     Object.entries(payload.predictedValues).forEach(item => {
46       var filterName = item[0];
47       var filterValue = item[1];
48       components.push(
49         <div className = 's3DivStyle'>
50           <div className = 'predictedS3FilterNameStyle'>{ filterName
51           }</div>
52           <div className = 'predictedS3FilterValueStyle'>{
53             filterValue }</div>
54         </div>
55     );
56   }
57
58   createData(filterName, filterData) {
59     const bgColorMap = {
60       "t2.nano": "#B21F00",
61       "t2.micro": "#C9DE00",
62       "t2.small": "#2FDE00",
63     }
64   }
65 }
```

```
362     "t2.medium": "#00A6B4",
363     "t2.large": "#6800B4",
364     "t2.xlarge": "#FF4500",
365     "db.t2.micro": "#C9DE00",
366     "db.t2.small": "#2FDE00",
367     "db.t2.medium": "#00A6B4",
368
369   };
370
371   const hvColorMap = {
372     "t2.nano": "#501800",
373     "t2.micro": "#4B5000",
374     "t2.small": "#175000",
375     "t2.medium": "#003350",
376     "t2.large": "#35014F",
377     "t2.xlarge": "#991900",
378     "db.t2.micro": "#4B5000",
379     "db.t2.small": "#175000",
380     "db.t2.medium": "#003350",
381   };
382
383
384   var graphLabels = [];
385   var graphData = [];
386   var bgColor = [];
387   var hvColor = [];
388
389   Object.entries(filterData).forEach(item => {
390     var instanceType = item[0];
391     var instanceValue = item[1];
392
393     graphLabels.push(instanceType);
394     graphData.push(instanceValue);
395     bgColor.push(bgColorMap[instanceType]);
396     hvColor.push(hvColorMap[instanceType]);
397   })
398
399   const data = {
400     labels: graphLabels,
401     datasets: [
```

```
101      {
102        label: "",
103        backgroundColor: bgColor,
104        hoverBackgroundColor: hvColor,
105        data: graphData
106      }
107    ]
108  }
109
110  return data;
111}
112
113 createCharts(data, filterCompare, displayName) {
114  return(
115    <div>
116      <div className = 'doughnutGraphStyle'>
117        <p className = 'filterStyle'>{ displayName }</p>
118        <p className = 'filterRelativeStyle'>Relative Importance of
Instance Types</p>
119        <Doughnut
120          data = { data }
121          options = {{
122            title: {
123              display: true,
124              text: displayName,
125              fontSize: 20
126            },
127            legend: {
128              display: true,
129              position: 'right'
130            }
131          }}
132        />
133      </div>
134      <div className = 'barGraphStyle'>
135        <Bar
136          data={ data }
137          options={{{
138            title:{
```

```
139             display: false ,
140             text: displayName ,
141             fontSize: 20
142         },
143         legend: {
144             display: false
145         },
146         tooltips: {
147             enabled: false
148         }
149     })
150     />
151 </div>
152 { this.textUtil(filterCompare) }
153 <br/>
154 <hr className = 'hrStyle' />
155 </div>
156 );
157 }
158
159 renderGraphs(data) {
160     var components = [];
161     Object.entries(data).forEach(item => {
162         if(item[0] !== "overall") {
163             var collectiveData = item[1];
164             var filterData = collectiveData.data;
165             var filterCompare = collectiveData.filterCompare;
166             var displayName = collectiveData.displayName;
167
168             var graphData = this.createData(displayName, filterData);
169             var graphs = this.createCharts(graphData, filterCompare,
170             displayName);
171             components.push(graphs);
172         }
173     })
174
175     return components;
176 }
```

```
177    renderOverall(data) {
178        var collectiveData = data.overall;
179        var filterData = collectiveData.data;
180        var filterCompare = collectiveData.filterCompare;
181        var displayName = collectiveData.displayName;
182
183        var graphData = this.createData(displayName, filterData);
184        var graph = this.createCharts(graphData, filterCompare, displayName);
185
186        return graph;
187    }
188
189    render() {
190        var service = this.props.location.state.service;
191        var data = this.props.location.state.data;
192        console.log(data);
193        if(service === "Amazon S3") {
194            return (
195                <div>
196                    <div className = 'titleDivStyle2'>
197                        <h1 className = 'titleStyle2'>Results</h1>
198                    </div>
199                    <Link className = 'linkReturnStyle' to = "/dashboard">
3840                Return to Dashboard</Link>
200                    <div className = 'graphDivStyleCharts'>
201                        { this.s3TextUtil(data) }
202                        <p className = 's3FilterHeadingStyle'>About S3</p>
203                        <div className = 'aboutS3DivStyle'>
204
205                            <p className = 'aboutS3TextStyle'>S3 is an Object
storage built to store and retrieve any amount of data from anywhere.
206                            S3 has no instance types.<br/><br/> Amazon Simple
Storage Service (Amazon S3) is an object storage service that offers
3850 industry-leading
207                            scalability, data availability, security, and
performance.<br/><br/>This means customers of all sizes and industries can
use it to store
208                            and protect any amount of data for a range of use
cases, such as data lakes, websites, mobile applications, backup and
3855
```

```

209                     restore, archive, enterprise applications, IoT
210 devices, and big data analytics.<br/><br/> Amazon S3 provides easy-to-use
211 management features so you can organize your data
212 and configure finely-tuned access controls to meet your specific business,
213 organizational, and compliance requirements. <br/><
214 br/>Amazon S3 is designed for 99.99999999% (11 9's) of durability, and
215 stores
216
217                     data for millions of applications for companies all
218 around the world.<br/><br/><span style = {{color: '#F9CC88'}}>Cloud
219 Service Optimizer</span> runs several consistent
220 tests on S3's performance.</p>
221
222                 </div>
223             </div>
224         </div>
225     );
226
227 }
228
229
230
231
232
233
234 }
235
236
237 export default Results;

```

**Results.css**

```
3895 1 .titleStyle2 {  
2     margin-top: 0%;  
3     font-size: 250%;  
4     color: #FDFDFD;  
5     text-shadow: 2px 2px 2px #222;  
6     text-align: center;  
7     padding: 2%;  
8     margin-bottom: 2%;  
9 }  
3900  
10  
11 .titleDivStyle2 {  
12     background-color: #F9CC88;  
13     width: 17%;  
14     margin-left: auto;  
15     margin-right: auto;  
16     border-bottom-left-radius: 50px;  
17     border-bottom-right-radius: 50px;  
18     padding-bottom: 0.4%;  
19 }  
3905  
20  
21 .filterStyle {  
22     color: #696969;  
23     font-family: 'Arial Narrow Bold';  
24     font-weight: bold;  
25     font-size: 180%;  
26     text-align: center;  
27 }  
28  
29 .filterRelativeStyle {  
30     color: #696969;  
31     font-family: 'Arial Narrow Bold';  
32     font-weight: bold;  
33     font-size: 150%;  
34     text-align: center;  
35 }  
36  
37 .filterTextStyle {  
38     color: #696969;
```

```
39     font-family: 'Arial Narrow Bold';
39b    font-weight: bold;
41     font-size: 130%;
42     text-align: center;
43     margin-left: 7%;
44     margin-right: 7%;
44b    margin-top: 2%;
45     margin-bottom: 2%;
46 }
47 }

48

49 .serviceTextStyle {
50     color: #696969;
51     font-family: 'Arial Narrow Bold';
52     font-weight: bold;
53     font-size: 230%;
54     text-align: center;
55 }

56

57 .doughnutGraphStyle {
58     width: 30%;
59     height: 40%;
60     margin-top: 3%;
61     text-align: center;
62     margin-bottom: 3%;
63     margin-left: 35%;
64 }

65

66 .barGraphStyle {
67     width: 50%;
68     height: 40%;
69     margin-top: 5%;
70     text-align: center;
71     margin-bottom: 3%;
72     margin-left: 25%;
73 }

74

75 .graphDivStyleCharts {
76     width: 100%;
77     height: 100%;
```

```
78     text-align: center;
79     margin-right: 30%;
80 }
81
82 .hrStyle {
83     border: 1.5px solid grey;
84 }
85
86 .linkReturnStyle {
87     color: #F9CC88;
88     background-color: #fdfdfd;
89     border-radius: 50px;
90     text-align: left;
91     font-size: 130%;
92     text-decoration: none;
93     font-weight: bold;
94     margin: 5%;
95     margin-left: 78%;
96     padding-left: 1%;
97     padding-right: 1%;
98     padding-top: 0.5%;
99     padding-bottom: 0.5%;
100    font-family: 'Arial Narrow Bold';
101 }
102
103 a:hover {
104     color: orange;
105 }
106
107 .s3DivStyle {
108     display: flex;
109     margin-left: auto;
110     margin-right: auto;
111     text-align: justify;
112     justify-content: center;
113     align-items: center;
114 }
115
116 .s3FilterHeadingStyle {
```

```
117     color: #696969;  
118     font-family: 'Arial Narrow Bold';  
119     font-weight: bold;  
120     font-size: 170%;  
121     margin-top: 2%;  
122     margin-bottom: 2%;  
123 }  
124  
125 .filterHeadingStyle {  
126     color: #696969;  
127     font-family: 'Arial Narrow Bold';  
128     font-weight: bold;  
129     font-size: 150%;  
130     margin-top: 2%;  
131     margin-bottom: 2%;  
132 }  
133  
134 .selectedS3FilterNameStyle {  
135     color: #F9CC88;  
136     background-color: #FDFDFD;  
137     border-radius: 50px;  
138     padding: 0.7%;  
139     margin-right: 2%;  
140     padding-right: 2%;  
141     padding-left: 2%;  
142     font-family: 'Arial Narrow Bold';  
143     font-weight: bold;  
144     font-size: 130%;  
145     margin-bottom: 0.7%;  
146     width: 11%;  
147     text-align: center;  
148 }  
149  
150 .selectedS3FilterValueStyle {  
151     color: #F9CC88;  
152     background-color: #FDFDFD;  
153     border-radius: 50px;  
154     padding: 0.7%;  
155     margin-right: 2%;
```

```
156     padding-right: 2%;  
157     padding-left: 2%;  
158     font-family: 'Arial Narrow Bold';  
159     font-weight: bold;  
160     font-size: 130%;  
161     margin-bottom: 0.7%;  
162     justify-content: flex;  
163     text-align: center;  
164     width: 7%;  
165 }  
166  
167 .predictedS3FilterNameStyle {  
168     color: #FDFDFD;  
169     background-color: #F9CC88;  
170     border-radius: 50px;  
171     padding: 0.7%;  
172     margin-right: 2%;  
173     padding-right: 2%;  
174     padding-left: 2%;  
175     font-family: 'Arial Narrow Bold';  
176     font-weight: bold;  
177     font-size: 130%;  
178     margin-bottom: 0.7%;  
179     width: 11%;  
180     text-align: center;  
181 }  
182  
183 .predictedS3FilterValueStyle {  
184     color: #FDFDFD;  
185     background-color: #F9CC88;  
186     border-radius: 50px;  
187     padding: 0.7%;  
188     margin-right: 2%;  
189     padding-right: 2%;  
190     padding-left: 2%;  
191     font-family: 'Arial Narrow Bold';  
192     font-weight: bold;  
193     font-size: 130%;  
194     margin-bottom: 0.7%;
```

```
195     text-align: center;
196     width: 7%;
197 }
198
199 .aboutS3DivStyle {
200     background-color: #FDFDFD;
201     border-radius: 50px;
202     padding: 4%;
203     width: 80%;
204     margin-bottom: 3%;
205     margin-left: 6.7%;
206     background: rgba(255,255,255,0.7);
207 }
208
209 .aboutS3TextStyle {
210     color: #696969;
211     font-family: 'Arial Narrow Bold';
212     font-weight: bold;
213     font-size: 130%;
214     margin-left: 2%;
215     margin-right: 2%;
216     margin-top: 2%;
217     margin-bottom: 2%;
218     text-align: justify;
219 }
220
221 .instanceNameStyle {
222     color: #FDFDFD;
223     background-color: #F9CC88;
224     border-radius: 50px;
225     padding: 0.7%;
226     margin-right: 2%;
227     padding-right: 2%;
228     padding-left: 2%;
229     font-family: 'Arial Narrow Bold';
230     font-weight: bold;
231     font-size: 130%;
232     margin-bottom: 0.7%;
233     width: 18%;
```

```
234     text-align: center;
235 }
236
237 .instanceValueStyle {
238     color: #F9CC88;
239     background-color: #FDFDFD;
240     border-radius: 50px;
241     padding: 0.7%;
242     margin-right: 2%;
243     padding-right: 2%;
244     padding-left: 2%;
245     font-family: 'Arial Narrow Bold';
246     font-weight: bold;
247     font-size: 130%;
248     margin-bottom: 0.7%;
249     text-align: center;
250     width: 7%;
251 }
```

## BACK-END

4150 server.py

```
1 from flask import Flask, render_template, jsonify, request
2 from flaskext.mysql import MySQL
4153 from flask_cors import CORS, cross_origin
4 import random
5 import string
6 import datetime
7 import json
4168 import ahp
9 import os
10 import time
11 from threading import Thread
12 from uwsgi_tasks import task, TaskExecutor
4168 import threading
14 import databaseCommands as dbCommands
```

```
15 import mlCode as mlp
16
17 app = Flask(__name__)
418 CORS(app, support_credentials=True)
19 app.debug = True
20
21 def updateFilesInDB(endpointID, filesString):
22     dbCommands.updateFiles(endpointID, filesString)
423
24 def logAPICall(name):
25     timestamp = datetime.datetime.now()
26     dbCommands.logAPICall(name, str(timestamp))
27
428 def generatePNumber():
29     randomNo = 'p'
30     randomNoChars = ''.join([random.choice(string.ascii_letters + string.digits
) for n in range(9)])
31     randomNo = randomNo + randomNoChars
432     print(randomNo)
33     return randomNo
34
35 def generateSNumber():
36     randomNo = 's'
437     randomNoChars = ''.join([random.choice(string.ascii_letters + string.digits
) for n in range(9)])
38     randomNo = randomNo + randomNoChars
39     print(randomNo)
40     return randomNo
41
42 def getValFromDb(text):
43     #this function will use the entered text and search for statically
44     #stored content in a table
45     #if it matches a certain string, it will return it
46     #else, it will say not found
47     statement = "select content from staticContent where name = '" + text + "'"
48     result = dbCommands.dbSelect(statement)
49     if(result == []):
50         return "not found"
```

```
51     else:
52         return result
53
54 def retrieveValuesFromDB():
55     return dbCommands.retrieveValues()
56
57 def startProcessing(filters, serviceType, instanceType, priorities):
58     filtersList = filters.split(",")
59     instancesList = instanceType.split(",")
60     prioritiesList = priorities.split(",")
61     result = dbCommands.ahpCode(serviceType, instancesList, filtersList,
62     prioritiesList)
63     return result
64
65 def bgTask(pNo):
66     print('Started thread:')
67     val = getResultsFromProcessingNumber(pNo)
68
69 def processFilters(timestamp, filters, priorities, serviceType, instanceTypes):
70     #it uses the filters as input and generates ahp and ml-based calculations
71     errorMessage = ""
72     filtersList = filters.split(",")
73     prioritiesList = priorities.split(",")
74     if (filters == ""):
75         errorMessage+="Filters is missing."
76     if (timestamp == ""):
77         errorMessage+=" Timestamp is missing."
78     if (serviceType == ""):
79         errorMessage+=" ServiceType is missing."
80     if (instanceTypes == ""):
81         errorMessage+=" InstanceTypes is missing."
82     if (priorities == ""):
83         errorMessage+=" Priorities is missing."
84     if (not len(filtersList) == len(prioritiesList)):
85         errorMessage+=" Filters and Priorities length do not match."
86     if errorMessage != "":
87         return ("failure", errorMessage)
88     else:
89         pNo = generatePNumber()
```

```
489     while(dbCommands.isDuplicatePNumber(pNo)):
490         pNo = generatePNumber()
491         dbCommands.insertPNumber(pNo)
492         dbCommands.insertPRequest(str(pNo), timestamp, filters, priorities,
493         serviceType, instanceTypes)
494         return ("success", str(pNo))
495
496 def updateResult(result, pNo):
497     timestamp = datetime.datetime.now()
498     dbCommands.updateResults(result, pNo, str(timestamp))
499
500 def checkStatus(pNo):
501     if(dbCommands.checkProcessingStatus(pNo)):
502         return 1
503     else:
504         return 0
505
506 def checkIfProcessingIsDone(pNo):
507     if(dbCommands.checkIfProcessingIsDone(pNo)):
508         return 1
509     else:
510         return 0
511
512 def isAS3Request(pNo):
513     if(dbCommands.isAS3Request(pNo)):
514         return 1
515     else:
516         return 0
517
518 def getResultsFromPNo(text):
519     if(not dbCommands.isDuplicatePNumber(text)):
520         return (0, "Invalid PNo.")
521     elif(checkIfProcessingIsDone(text)):
522         return (0, "Processing for this pNo is still pending... Try again")
523     else:
524         result = dbCommands.getResultFromAHP(text)
525         return result
526
527 def getResultsFromProcessingNumber(text):
```

```
127     if(not dbCommands.isDuplicatePNumber(text)):
128         return (0, "Invalid PNo.")
129     elif(checkStatus(text)):
130         return (0, "Processing for this pNo is complete!")
131     else:
132         filters = dbCommands.getFilters(text)
133         serviceType = dbCommands.getServiceType(text)
134         priorities = dbCommands.getPriorities(text)
135         if(isAS3Request(text)):
136             result = mlp.runML(serviceType, 'na', priorities, filters)
137             updateResult(result, text)
138             return result
139         else:
140             instanceType = dbCommands.getInstanceType(text)
141             result = startProcessing(filters, serviceType, instanceType,
142 priorities)
143             updateResult(result, text)
144             return result
145
146 def getLatestTimestamp(endpointID):
147     filesString = dbCommands.GetFiles(endpointID)
148     return filesString
149
150 def initiateLogging(endpointID):
151     sNo = generateSNumber()
152     while(dbCommands.isDuplicateSNumber(sNo)):
153         sNo = generateSNumber()
154     dbCommands.insertSNumber(sNo, endpointID)
155     return str(sNo)
156
157 def updateSessionInfo(sessionID):
158     dbCommands.updateSession(sessionID)
159
160 def saveToEC2Summary(index, valuesList, sessionID, endpointID, instanceType,
161 logsSummary, timestamp):
162     label = valuesList[0]
163     numberofSamples = valuesList[1]
```

```
164     maxVal = valuesList[4]
165     stdDev = valuesList[5]
166     errorRate = valuesList[6]
167     throughput = valuesList[7]
168     receivedKBPS = valuesList[8]
169     sentKBPS = valuesList[9]
170     avgBytes = valuesList[10]
171     dbCommands.insertEC2Summary(index, label, numberOfSamples, average, minVal,
172     maxVal, stdDev, errorRate, throughput, receivedKBPS, sentKBPS, avgBytes,
173     endpointID, sessionID, instanceType, timestamp)
174
175 def saveSummaryInEC2(index, sessionID, endpointID, instanceType, logsSummary,
176 timestamp):
177     summary_dict = json.loads(logsSummary)
178     valuesList = []
179     print(summary_dict.values())
180     for i in summary_dict.values():
181         valuesList.append(i)
182     print(valuesList)
183     saveToEC2Summary(index, valuesList, sessionID, endpointID, instanceType,
184     logsSummary, timestamp)
185
186
187 def saveSummaryDataInEC2(index, sessionID, endpointID, instanceType,
188 logsSummary, timestamp):
189     summary_dict = json.loads(logsSummary)
190     valuesList = []
191
192
193 def saveToEC2Logs(index, valuesList, instanceType, sessionID, endpointID,
194 timestamp):
195     elapsed = valuesList[0]
196     responseCode = valuesList[1]
197     responseMessage = valuesList[2]
198     success = valuesList[3]
199     latency = valuesList[4]
200     connect = valuesList[5]
201     dbCommands.insertEC2Logs(index, elapsed, responseCode, responseMessage,
202     success, latency, connect, instanceType, endpointID, sessionID, timestamp)
```

```
196 def saveToEC2Performance(index, valuesList, instanceType, sessionID, endpointID
197     , timestamp):
198     cpu = valuesList[0]
199     memory = valuesList[1]
200     dbCommands.insertEC2Performance(index, cpu, memory, instanceType,
201     endpointID, sessionID, timestamp)
202
203
204 def saveLogsInEC2(sessionID, endpointID, instanceType, logs, timestamp):
205     logs_dict = json.loads(logs)
206     print(len(logs_dict))
207     print("Logs Dict['elapsed']:")
208     print(len(logs_dict.values()))
209     index = dbCommands.getLatestCount('ec2logs')
210     for j in range(1, len(logs_dict['elapsed'])):
211         valuesList = []
212         for i in logs_dict.values():
213             valuesList.append(i.get(str(j)))
214         saveToEC2Logs(index, valuesList, instanceType, sessionID, endpointID,
215 timestamp)
216         index = index + 1
217
218
219 def savePerformanceInEC2(sessionID, endpointID, instanceType, logsPerformance,
220 timestamp):
221     logs_dict = json.loads(logsPerformance)
222     print(len(logs_dict))
223     print("LogsPerformance Dict['CPU']:")
224     print(len(logs_dict.values()))
225     index = dbCommands.getLatestCount('ec2performance')
226     for j in range(1, len(logs_dict['CPU'])):
227         valuesList = []
228         for i in logs_dict.values():
229             valuesList.append(i.get(str(j)))
230         saveToEC2Performance(index, valuesList, instanceType, sessionID,
231 endpointID, timestamp)
232         index = index + 1
233
234
235 def saveSummary(sessionID, endpointID, instanceType, serviceType, logsSummary,
236 timestamp):
237     if(serviceType=='ec2'):
```

```
229     index = dbCommands.getLatestCount('ec2summary')
230     saveSummaryInEC2(index, sessionID, endpointID, instanceType,
231     logsSummary, timestamp)
232     return 1
233
234     if(endpointID == "hello"):
235         return 0
236
237 def saveSummaryData(sessionID, endpointID, instanceType, serviceType,
238     logsSummary, timestamp):
239
240     if(serviceType=='ec2'):
241         saveSummaryDataInEC2(index, sessionID, endpointID, instanceType,
242         logsSummary, timestamp)
243
244     return 1
245
246     if(endpointID == "hello"):
247         return 0
248
249     else:
250         return 0
251
252
253 def saveLogs(sessionID, endpointID, instanceType, serviceType, logs, timestamp)
254 :
255
256     if(serviceType=='ec2'):
257         saveLogsInEC2(sessionID, endpointID, instanceType, logs, timestamp)
258
259     return 1
260
261     if(endpointID == "hello"):
262         return 0
263
264
265 def savePerformance(sessionID, endpointID, instanceType, serviceType,
266     logsPerformance, timestamp):
267
268     if(serviceType=='ec2'):
269         savePerformanceInEC2(sessionID, endpointID, instanceType,
270         logsPerformance, timestamp)
271
272     return 1
273
274     if(endpointID == "hello"):
275         return 0
276
277     else:
```

```
262         return 0
263
264 def checkPassword(endpointID, password):
265     if(dbCommands.checkPassword(endpointID, password)):
266         return 1
267     else:
268         return 0
269
270 def logLogin(endpointID, timestamp):
271     dbCommands.logLogin(endpointID, timestamp)
272
273 def attemptLogin(endpointID, password, timestamp):
274     if(checkPassword(endpointID, password)):
275         logLogin(endpointID, timestamp)
276         return "success"
277     else:
278         return "failure"
279
280 def updateDerivedValues(sessionID, endpointID, instanceType, serviceType,
281 timestamp):
282     elapsed = dbCommands.getElapsed(instanceType, serviceType)
283     latency = dbCommands.getLatency(instanceType, serviceType)
284     connect = dbCommands.getConnect(instanceType, serviceType)
285     cpu = dbCommands.getCPU(instanceType, serviceType)
286     memory = dbCommands.getMemory(instanceType, serviceType)
287     standardDeviation = dbCommands.getOverallStandardDeviation(instanceType,
288 serviceType)
289     errorRate = dbCommands.getOverallErrorRate(instanceType, serviceType)
290     throughput = dbCommands.getOverallThroughput(instanceType, serviceType)
291     cpu = dbCommands.getCPU(instanceType, serviceType)
292     memory = dbCommands.getMemory(instanceType, serviceType)
293     dbCommands.updateDerivedValues(elapsed, latency, connect, cpu, memory,
294 standardDeviation, errorRate, throughput, sessionID, endpointID,
295 instanceType, serviceType, timestamp)
296
297     dbCommands.updateDerivedCalculationLogs(endpointID, sessionID, serviceType,
298 instanceType, timestamp)
299
300 def getNameFromEndpointID(endpointID):
301     name = dbCommands.getNameFromEndpointID(endpointID)
```

```
296     return name
297
298 def createEndpoint(name, password):
299     endpointID = dbCommands.createEndpoint(name, password)
300     return str(endpointID)
301
302 def updateDerValues(sessionID, endpointID, instanceType, serviceType, timestamp):
303     elapsed = dbCommands.getStoredElapsed(instanceType, serviceType)
304     latency = dbCommands.getStoredLatency(instanceType, serviceType)
305     connect = dbCommands.getStoredConnect(instanceType, serviceType)
306     cpu = dbCommands.getStoredCPU(instanceType, serviceType)
307     memory = dbCommands.getStoredMemory(instanceType, serviceType)
308     standardDeviation = dbCommands.getOverallStandardDeviation(instanceType,
309     serviceType)
310     errorRate = dbCommands.getOverallErrorRate(instanceType, serviceType)
311     throughput = dbCommands.getOverallThroughput(instanceType, serviceType)
312     dbCommands.updateDerivedValues(elapsed, latency, connect, cpu, memory,
313     standardDeviation, errorRate, throughput, sessionID, endpointID,
314     instanceType, serviceType, timestamp)
315     dbCommands.updateDerivedCalculationLogs(endpointID, sessionID, serviceType,
316     instanceType, timestamp)
317
318 def saveEC2(index, sessionID, endpointID, instanceType, serviceType, elapsed,
319     latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
320     cpu, memory, performanceSize):
321     dbCommands.saveEC2(index, sessionID, endpointID, instanceType, serviceType,
322     elapsed, latency, connect, stdDev, errorRate, throughput, summarySize,
323     timestamp, cpu, memory, performanceSize)
324
325 def saveRDS(index, sessionID, endpointID, instanceType, serviceType, elapsed,
326     latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
327     cpu, memory, performanceSize):
328     dbCommands.saveRDS(index, sessionID, endpointID, instanceType, serviceType,
329     elapsed, latency, connect, stdDev, errorRate, throughput, summarySize,
330     timestamp, cpu, memory, performanceSize)
331
332 def saveS3(index, sessionID, endpointID, instanceType, serviceType, elapsed,
333     latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
```

```
cpu, memory, performanceSize):
    dbCommands.saveS3(index, sessionID, endpointID, instanceType, serviceType,
elapsed, latency, connect, stdDev, errorRate, throughput, summarySize,
timestamp, cpu, memory, performanceSize)

322
323 def saveLogData(sessionID, endpointID, instanceType, serviceType, elapsed,
latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
isPerformanceIncluded, cpu, memory, performanceSize):
324     if(serviceType == 'ec2'):
325         index = dbCommands.getLatestCount('saveec2')
326         if(isPerformanceIncluded == '1'):
327             saveEC2(index, sessionID, endpointID, instanceType, serviceType,
elapsed, latency, connect, stdDev, errorRate, throughput, summarySize,
timestamp, cpu, memory, performanceSize)
328         else:
329             saveEC2(index, sessionID, endpointID, instanceType, serviceType,
elapsed, latency, connect, stdDev, errorRate, throughput, summarySize,
timestamp, 0, 0, 0)
330         return 1
331     elif(serviceType == 'rds'):
332         #do rds
333         index = dbCommands.getLatestCount('saverds')
334         saveRDS(index, sessionID, endpointID, instanceType, serviceType,
elapsed, latency, connect, stdDev, errorRate, throughput, summarySize,
timestamp, 0, 0, 0)
335         return 1
336     elif(serviceType == 's3'):
337         #do s3
338         index = dbCommands.getLatestCount('saves3')
339         saveS3(index, sessionID, endpointID, instanceType, serviceType, elapsed,
latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
0, 0, 0)
340         return 1
341     else:
342         return 0
343
344 @app.route('/', methods = ['GET'])
345 def hello():
346     return "Hello! Welcome to the server of Cloud Service Optimizer"
```

```
347
348 @app.route('/front-end/get-content/', methods=['GET', 'OPTIONS'])
349 def front_end_get_content():
350
351     logAPICall('f/get-content')
352     val = ""
353     text = ""
354     text = request.args.get("string")
355     val = getValFromDb(text)
356     if(val=="not found"):
357         value = jsonify({
358             "status": "failure",
359             "message": ""
360         })
361         value.headers.add('Access-Control-Allow-Origin', '*')
362         value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
363         return value
364     else:
365         value = jsonify({
366             "status": "success",
367             "message": val
368         })
369         value.headers.add('Access-Control-Allow-Origin', '*')
370         value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
371         return value
372
373
374 @app.route('/front-end/post-process-filters/', methods=['POST'])
375 def front_end_post_process_filters():
376     logAPICall('f/get-process-filters')
377     request_data = request.get_json()
378     serviceType = request_data['serviceType']
379     timestamp = request_data['timestamp']
380     filters = request_data['filters']
381     priorities = request_data['priorities']
382     instanceType = request_data['instanceType']
383     val = ()
384     val = processFilters(timestamp, filters, priorities, serviceType,
4595     instanceType)
```

```
385     value = jsonify({
386         "status": val[0],
387         "message": val[1]
388     })
389     value.headers.add('Access-Control-Allow-Origin', '*')
390     value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
391     return value
392
393 @app.route('/front-end/get-results/', methods=['GET', 'OPTIONS'])
394 def front_end_get_results():
395     val = ""
396     text = ""
397     text = request.args.get("processingNumber")
398     val = getResultsFromProcessingNumber(text)
399     if(val[0]==0):
400         value = jsonify({
401             "status": "failure",
402             "data": val[1],
403             "processingNumber": text
404         })
405         value.headers.add('Access-Control-Allow-Origin', '*')
406         value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
407         return value
408     else:
409         val1 = json.loads(val)
410         value = jsonify({
411             "status": "success",
412             "payload": val1,
413             "processingNumber": text
414         })
415         value.headers.add('Access-Control-Allow-Origin', '*')
416         value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
417         return value
418
419 @app.route('/endpoints/get-files/', methods=['POST'])
420 def endpoints_get_latest_timestamp():
421     logAPICall('e/get-files')
422     request_data = request.get_json()
423     endpointID = request_data['endpointID']
```

```
424     val = getLatestTimestamp(endpointID)
425     return jsonify({
426         "status": "success",
427         "message": val,
428         "endpointID": endpointID
429     })
430
431 @app.route('/endpoints/update-files/', methods=['POST'])
432 def endpoints_update_files():
433     logAPICall('e/update_files')
434     request_data = request.get_json()
435     endpointID = request_data['endpointID']
436     sessionID = request_data['sessionID']
437     filesString = request_data['filesString']
438     print(filesString)
439     updateFilesInDB(endpointID, filesString)
440     updateSessionInfo(sessionID)
441     return jsonify({
442         "status": "success",
443         "message": "Files list updated.",
444         "endpointID": endpointID,
445         "filesString": filesString
446     })
447
448 @app.route('/endpoints/initiate-logging/', methods=['GET'])
449 def endpoints_initiate_logging():
450     logAPICall('e/initiate-logging')
451     val = ""
452     endpointID = ""
453     endpointID = request.args.get("endpointID")
454     val = initiateLogging(endpointID)
455     return jsonify({
456         "status": "success",
457         "message": val
458     })
459
460
461 @app.route('/endpoints/post-logs/', methods=['POST'])
462 def endpoints_post_summary():
```

```
463     logAPICall('e/post-logs')
464     request_data = request.get_json()
465     sessionID = request_data['sessionID']
466     endpointID = request_data['endpointID']
467     instanceType = request_data['instanceType']
468     serviceType = request_data['serviceType']
469     logsSummary = request_data['logsSummary']
470     logs = request_data['logs']
471     timestamp = request_data['timestamp']
472     isPerformanceIncluded = request_data['isPerformanceIncluded']
473     logsPerformance = request_data['logsPerformance']
474     val = saveSummary(sessionID, endpointID, instanceType, serviceType,
475     logsSummary, timestamp)
476     val2 = saveLogs(sessionID, endpointID, instanceType, serviceType, logs,
477     timestamp)
478     val3 = 1
479     if(isPerformanceIncluded == '1'):
480         val3 = savePerformance(sessionID, endpointID, instanceType, serviceType
481         , logsPerformance, timestamp)
482     if(val and val2 and val3):
483         print("In update derived values")
484         updateDerivedValues(sessionID, endpointID, instanceType, serviceType,
485         timestamp)
486     if(not val or not val2 or not val3):
487         return jsonify({
488             "status": "failure",
489             "message": val,
490             "sessionID": sessionID,
491             "endpointID": endpointID,
492             "serviceType": serviceType,
493             "instanceType": instanceType
494         })
495     else:
496         return jsonify({
497             "status": "success",
498             "message": val,
499             "sessionID": sessionID,
500             "endpointID": endpointID,
501             "serviceType": serviceType,
```

```
498     "instanceType": instanceType
499 )
500
501 @app.route('/endpoints/post-log-data/', methods=['POST'])
502 def endpoints_post_log_data():
503     logAPICall('e/post-log-data')
504     request_data = request.get_json()
505     sessionID = request_data['sessionID']
506     endpointID = request_data['endpointID']
507     instanceType = request_data['instanceType']
508     serviceType = request_data['serviceType']
509     elapsed = request_data['elapsed']
510     latency = request_data['latency']
511     connect = request_data['connect']
512     stdDev = request_data['stdDev']
513     errorRate = request_data['errorRate']
514     throughput = request_data['throughput']
515     summarySize = request_data['summarySize']
516     timestamp = request_data['timestamp']
517     isPerformanceIncluded = request_data['isPerformanceIncluded']
518     cpu = request_data['cpu']
519     memory = request_data['memory']
520     performanceSize = request_data['performanceSize']
521     val = saveLogData(sessionID, endpointID, instanceType, serviceType, elapsed,
522     , latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
523     isPerformanceIncluded, cpu, memory, performanceSize)
524     if(val):
525         print("In update derived values")
526         updateDerValues(sessionID, endpointID, instanceType, serviceType,
527         timestamp)
528     if(not val):
529         return jsonify({
530             "status": "failure",
531             "message": val,
532             "sessionID": sessionID,
533             "endpointID": endpointID,
534             "serviceType": serviceType,
535             "instanceType": instanceType
536         })
```

```
534     else:
535
536         return jsonify({
537             "status": "success",
538             "message": val,
539             "sessionID": sessionID,
540             "endpointID": endpointID,
541             "serviceType": serviceType,
542             "instanceType": instanceType
543         })
544
545 @app.route('/endpoints/endpoint-login/', methods=['POST'])
546 def endpoints_endpoint_login():
547     logAPICall('e/endpoint_login')
548     request_data = request.get_json()
549     endpointID = request_data['endpointID']
550     password = request_data['password']
551     timestamp = request_data['timestamp']
552     result = attemptLogin(endpointID, password, timestamp)
553     if(result == "success"):
554         name = getNameFromEndpointID(endpointID)
555         return jsonify({
556             "status": "success",
557             "message": name,
558             "endpointID": endpointID
559         })
560     else:
561         return jsonify({
562             "status": "failure",
563             "message": "Invalid credentials.",
564             "endpointID": endpointID
565         })
566
567 @app.route('/endpoints/create-endpoint/', methods=['POST'])
568 def endpoints_create_endpoint():
569     logAPICall('e/create-endpoint')
570     request_data = request.get_json()
571     name = request_data['name']
572     password = request_data['password']
573     result = createEndpoint(name, password)
```

```
573     return jsonify({
574         "status": "success",
575         "message": result,
576         "name": name
577     })
578
579 if __name__ == '__main__':
580     app.run()
```

## 4800 dbCommands.py

```
1 import mysql.connector
2 import ahp
3 import time
4 from itertools import combinations
5
6 db =mysql.connector.connect(host='db-mysql-blr1-22358-do-user-9282953-0.b.db.
7     ondigitalocean.com',
8     database='defaultdb', user='doadmin', password='***', port='25060', auth_plugin
9     ='mysql_native_password')
10 mycursor = db.cursor()
11
12 def dbInsert(statement, data):
13     mycursor.execute(statement, data)
14
15 def dbSelect(statement):
16     mycursor.execute(statement)
17     result = mycursor.fetchall()
18     return result
19
20 def isDuplicatePNumber(pno):
21     mycursor.execute('select * from taskStatus where pno=\''+pno+'\'')
22     result = mycursor.fetchall()
23     print(result)
24     if result == []:
25         return False
26     else:
```

```
25         return True
426
27 def isDuplicateSNumber(sno):
28     mycursor.execute('select * from sessionDetails where sno=\''+sno+'\';')
29     result = mycursor.fetchall()
30     if result == []:
31         return False
32     else:
33         return True
34
35 def insertPNumber(pno):
36     mycursor.execute('insert into taskStatus values(\''+pno+'\', \'Pending\');')
37     db.commit()
38
39 def insertSNumber(sno, endpointID):
40     mycursor.execute('insert into sessionDetails values(\''+sno+'\', \'Pending\',
41     \''+endpointID+'\');')
42     db.commit()
43
44 def logAPICall(name, timestamp):
45     mycursor.execute('insert into APILogs values(\''+name+'\', \''+timestamp+'\
46     \');')
47     db.commit()
48
49 def getFiles(endpointID):
50     mycursor.execute('select files from endpointLogs where endpointID=\''+str(
51     endpointID)+'\';')
52     result = mycursor.fetchall()
53     return result
54
55
56 def updateFiles(endpointID, filesString):
57     mycursor.execute('update endpointLogs set files = \''+str(filesString)+\
58     '\', where endpointID = \''+str(endpointID)+'\';')
59     db.commit()
60
61
62 def updateSession(sessionID):
63     mycursor.execute('update sessionDetails set status = \'Complete\' where sno
64     = \''+str(sessionID)+'\';')
```

```

58     db.commit()
59
460 def insertEC2Logs(index, elapsed, responseCode, responseMessage, success,
61     latency, connect, instanceType, endpointID, sessionID, timestamp):
62     mycursor.execute('insert into ec2logs values(\''+str(index)+'\', \''+str(
63         elapsed)+'\', \''+str(responseCode)+'\', \''+str(responseMessage)+'\', \''+
64         str(success)+'\', \''+str(latency)+'\', \''+str(connect)+'\', \''+str(
4875       instanceType)+'\', \''+
66       +str(endpointID)+'\', \''+str(sessionID)+'\', \''+str(timestamp)+'\');')
67
68     db.commit()
69
70 def insertEC2Performance(index, cpu, memory, instanceType, endpointID,
4880   sessionID, timestamp):
71     mycursor.execute('insert into ec2performance values(\''+str(index)+'\', \''+
72       +str(cpu)+'\', \''+str(memory)+'\', \''+str(instanceType)+'\', \''+
73       +str(endpointID)+'\', \''+str(sessionID)+'\', \''+str(timestamp)+'\');')
74
75 def insertEC2Summary(index, label, numberOfWorkSamples, average, minVal, maxVal,
4890   stdDev, errorRate, throughput, receivedKBPS, sentKBPS, avgBytes, endpointID,
76   sessionID, instanceType, timestamp):
77     mycursor.execute('insert into ec2summary values(\''+str(index)+'\', \''+
78       +str(label)+'\', \''+str(numberOfWorkSamples)+'\', \''+str(average)+'\', \''+
79       +str(minVal)+'\', \''+str(maxVal)+'\', \''+str(stdDev)+'\', \''+str(errorRate)+'\',
80       \', \''+
81       +str(throughput)+'\', \''+str(receivedKBPS)+'\', \''+str(sentKBPS)+'\', \''+
82       +str(avgBytes)+'\', \''+str(endpointID)+'\', \''+str(sessionID)+'\', \''+
4895     +str(instanceType)+'\', \''+str(timestamp)+'\');')
83
84     db.commit()
85
75 def checkPassword(endpointID, password):
76     mycursor.execute('select password from endpointAccess where endpointID=\''+
4900   str(endpointID)+'\';')
77
78     result = mycursor.fetchall()
79
80     result = str(result)
81
82     result = result[3:-4]
83
84     if(password==result):
85
86         return 1
87
88     else:

```

```
83         return 0
84
85 def logLogin(endpointID, timestamp):
86     mycursor.execute('insert into loginLogs values(\''+str(endpointID)+'\', \'
87     '+str(timestamp)+'\')')
88     db.commit()
89
90 def getNoOfSamples(instanceType, serviceType):
91     mycursor.execute('select summarySize from save'+str(serviceType)+'
92     where instanceType = \''+str(instanceType)+'\';')
93     noResult = mycursor.fetchall()
94     noResultList = []
95     for i in noResult:
96         noResultList.append(str(i)[2:-3])
97     noResultList = [float(ele) for ele in noResultList]
98     return noResultList
99
100 def getNoOfPerformanceSamples(instanceType, serviceType):
101     mycursor.execute('select performanceSize from save'+str(serviceType)+'
102     where instanceType = \''+str(instanceType)+'\';')
103     noResult = mycursor.fetchall()
104     noResultList = []
105     for i in noResult:
106         noResultList.append(str(i)[2:-3])
107     noResultList = [float(ele) for ele in noResultList]
108     return noResultList
109
110 def getOverallThroughput(instanceType, serviceType):
111     tpResultList = getThroughputs(instanceType, serviceType)
112     noResultList = getNoOfSamples(instanceType, serviceType)
113     tpProductsList = []
114     for i in range(0, len(tpResultList)):
115         tpProductsList.append(tpResultList[i]*noResultList[i])
116     totalNumberOfSamples = 0
117     tpProductsSum = 0
118     for ele in range(0, len(noResultList)):
119         totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
120         tpProductsSum = tpProductsSum + tpProductsList[ele]
121     if(len(noResultList) == 0):
```

```
119         overallThroughput = 0
120     else:
121         overallThroughput = tpProductsSum / totalNumberOfSamples
122     return str(overallThroughput)
123
124 def getNumberOfSamples(instanceType , serviceType):
125     mycursor.execute('select numberofSamples from '+str(serviceType)+ 'summary
126 where instanceType = \''+str(instanceType)+'\';')
127     noResult = mycursor.fetchall()
128     noResultList = []
129     for i in noResult:
130         noResultList.append(str(i)[2:-3])
131     noResultList = [float(ele) for ele in noResultList]
132     return noResultList
133
134 def getThroughputs(instanceType , serviceType):
135     mycursor.execute('select throughput from save'+str(serviceType)+', where
136 instanceType = \''+str(instanceType)+'\';')
137     tpResult = mycursor.fetchall()
138     tpResultList = []
139     for i in tpResult:
140         tpResultList.append(str(i)[2:-3])
141     tpResultList = [float(ele) for ele in tpResultList]
142     return tpResultList
143
144 def getLatency(instanceType , serviceType):
145     mycursor.execute('select Latency from '+str(serviceType)+ 'logs where
146 instanceType = \''+str(instanceType)+'\';')
147     result = mycursor.fetchall()
148     resultList = []
149     for i in result:
150         resultList.append(str(i)[2:-3])
151
152     resultList = [float(ele) for ele in resultList]
153     if(len(resultList)==0):
154         avg = 0
155     else:
156         avg = sum(resultList)/len(resultList)
157     return str(avg)
```

```
155
156 def getErrorRates(instanceType, serviceType):
157     mycursor.execute('select errorRate from save'+str(serviceType)+', where
158 instanceType = \''+str(instanceType)+'\';')
159     erResult = mycursor.fetchall()
160     erResultList = []
161     for i in erResult:
162         erResultList.append(str(i)[2:-4])
163     erResultList = [float(ele) for ele in erResultList]
164     return erResultList
165
166 def getOverallErrorRate(instanceType, serviceType):
167     erResultList = getErrorRates(instanceType, serviceType)
168     noResultList = getNoOfSamples(instanceType, serviceType)
169     erProductsList = []
170     for i in range(0, len(erResultList)):
171         erProductsList.append(erResultList[i]*noResultList[i])
172     totalNumberOfSamples = 0
173     erProductsSum = 0
174     for ele in range(0, len(noResultList)):
175         totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
176         erProductsSum = erProductsSum + erProductsList[ele]
177     if(len(noResultList) == 0):
178         overallErrorRate = 0
179     else:
180         overallErrorRate = erProductsSum / totalNumberOfSamples
181     return str(overallErrorRate)
182
183 def getElapsed(instanceType, serviceType):
184     mycursor.execute('select elapsed from '+str(serviceType)+', logs where
185 instanceType = \''+str(instanceType)+'\';')
186     result = mycursor.fetchall()
187     resultList = []
188     for i in result:
189         resultList.append(str(i)[2:-3])
190     resultList = [float(ele) for ele in resultList]
191     if(len(resultList)==0):
192         avg = 0
193     else:
```

```
192         avg = sum(resultList)/len(resultList)
193     return str(avg)
194
195 def getElapsedValues(instanceType , serviceType):
196     mycursor.execute('select elapsed from save'+str(serviceType)+', where
197 instanceType = \''+str(instanceType)+'\';')
198     sdResult = mycursor.fetchall()
199     sdResultList = []
200     for i in sdResult:
201         sdResultList.append(str(i)[2:-3])
202     sdResultList = [float(ele) for ele in sdResultList]
203     return sdResultList
204
205 def getLatencyValues(instanceType , serviceType):
206     mycursor.execute('select Latency from save'+str(serviceType)+', where
207 instanceType = \''+str(instanceType)+'\';')
208     sdResult = mycursor.fetchall()
209     sdResultList = []
210     for i in sdResult:
211         sdResultList.append(str(i)[2:-3])
212     sdResultList = [float(ele) for ele in sdResultList]
213     return sdResultList
214
215 def getConnectValues(instanceType , serviceType):
216     mycursor.execute('select Connect from save'+str(serviceType)+', where
217 instanceType = \''+str(instanceType)+'\';')
218     sdResult = mycursor.fetchall()
219     sdResultList = []
220     for i in sdResult:
221         sdResultList.append(str(i)[2:-3])
222     sdResultList = [float(ele) for ele in sdResultList]
223     return sdResultList
224
225 def getCPUValues(instanceType , serviceType):
226     mycursor.execute('select cpu from save'+str(serviceType)+', where
227 instanceType = \''+str(instanceType)+'\';')
228     sdResult = mycursor.fetchall()
229     sdResultList = []
230     for i in sdResult:
```

```
227     sdResultList.append(str(i)[2:-3])
228     sdResultList = [float(ele) for ele in sdResultList]
229     return sdResultList
230
231 def getMemoryValues(instanceType, serviceType):
232     mycursor.execute('select memory from save'+str(serviceType)+', where
233 instanceType = \''+str(instanceType)+'\';')
234     sdResult = mycursor.fetchall()
235     sdResultList = []
236     for i in sdResult:
237         sdResultList.append(str(i)[2:-3])
238     sdResultList = [float(ele) for ele in sdResultList]
239     return sdResultList
240
241 def getStoredElapsed(instanceType, serviceType):
242     sdResultList = getElapsedValues(instanceType, serviceType)
243     noResultList = getNoOfSamples(instanceType, serviceType)
244     sdProductsList = []
245     for i in range(0, len(sdResultList)):
246         sdProductsList.append(sdResultList[i]*noResultList[i])
247     totalNumberOfSamples = 0
248     sdProductsSum = 0
249     for ele in range(0, len(noResultList)):
250         totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
251         sdProductsSum = sdProductsSum + sdProductsList[ele]
252     if(len(noResultList) == 0):
253         overallElapsed = 0
254     else:
255         overallElapsed = sdProductsSum / totalNumberOfSamples
256     return str(overallElapsed)
257
258 def getStoredLatency(instanceType, serviceType):
259     sdResultList = getLatencyValues(instanceType, serviceType)
260     noResultList = getNoOfSamples(instanceType, serviceType)
261     sdProductsList = []
262     for i in range(0, len(sdResultList)):
263         sdProductsList.append(sdResultList[i]*noResultList[i])
264     totalNumberOfSamples = 0
265     sdProductsSum = 0
```

```
265     for ele in range(0, len(noResultList)):
266         totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
267         sdProductsSum = sdProductsSum + sdProductsList[ele]
268     if(len(noResultList) == 0):
269         overallLatency = 0
270     else:
271         overallLatency = sdProductsSum / totalNumberOfSamples
272     return str(overallLatency)
273
274 def getStoredConnect(instanceType, serviceType):
275     sdResultList = getConnectValues(instanceType, serviceType)
276     noResultList = getNoOfSamples(instanceType, serviceType)
277     sdProductsList = []
278     for i in range(0, len(sdResultList)):
279         sdProductsList.append(sdResultList[i]*noResultList[i])
280     totalNumberOfSamples = 0
281     sdProductsSum = 0
282     for ele in range(0, len(noResultList)):
283         totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
284         sdProductsSum = sdProductsSum + sdProductsList[ele]
285     if(len(noResultList) == 0):
286         overallConnect = 0
287     else:
288         overallConnect = sdProductsSum / totalNumberOfSamples
289     return str(overallConnect)
290
291 def getStoredCPU(instanceType, serviceType):
292     sdResultList = getCPUValues(instanceType, serviceType)
293     noResultList = getNoOfPerformanceSamples(instanceType, serviceType)
294     sdProductsList = []
295     for i in range(0, len(sdResultList)):
296         sdProductsList.append(sdResultList[i]*noResultList[i])
297     totalNumberOfSamples = 0
298     sdProductsSum = 0
299     for ele in range(0, len(noResultList)):
300         totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
301         sdProductsSum = sdProductsSum + sdProductsList[ele]
302     if(len(noResultList) == 0 or totalNumberOfSamples == 0):
303         overallCPU = 0
```

```
304     else:
305         overallCPU = sdProductsSum / totalNumberOfSamples
306     return str(overallCPU)
307
308 def getStoredMemory(instanceType, serviceType):
309     sdResultList = getMemoryValues(instanceType, serviceType)
310     noResultList = getNoOfPerformanceSamples(instanceType, serviceType)
311     sdProductsList = []
312     for i in range(0, len(sdResultList)):
313         sdProductsList.append(sdResultList[i]*noResultList[i])
314     totalNumberOfSamples = 0
315     sdProductsSum = 0
316     for ele in range(0, len(noResultList)):
317         totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
318         sdProductsSum = sdProductsSum + sdProductsList[ele]
319     if(len(noResultList) == 0 or totalNumberOfSamples == 0):
320         overallMemory = 0
321     else:
322         overallMemory = sdProductsSum / totalNumberOfSamples
323     return str(overallMemory)
324
325 def getStandardDeviations(instanceType, serviceType):
326     mycursor.execute('select stdDev from save'+str(serviceType)+', where
327 instanceType = \''+str(instanceType)+'\';')
328     sdResult = mycursor.fetchall()
329     sdResultList = []
330     for i in sdResult:
331         sdResultList.append(str(i)[2:-3])
332     sdResultList = [float(ele) for ele in sdResultList]
333     return sdResultList
334
335 def getOverallStandardDeviation(instanceType, serviceType):
336     sdResultList = getStandardDeviations(instanceType, serviceType)
337     noResultList = getNoOfSamples(instanceType, serviceType)
338     sdProductsList = []
339     for i in range(0, len(sdResultList)):
340         sdProductsList.append(sdResultList[i]*noResultList[i])
341     totalNumberOfSamples = 0
342     sdProductsSum = 0
```

```
342     for ele in range(0, len(noResultList)):
343         totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
344         sdProductsSum = sdProductsSum + sdProductsList[ele]
345     if(len(noResultList) == 0):
346         overallStandardDeviation = 0
347     else:
348         overallStandardDeviation = sdProductsSum / totalNumberOfSamples
349     return str(overallStandardDeviation)
350
351 def getConnect(instanceType, serviceType):
352     mycursor.execute('select Connect from '+str(serviceType)+',logs where
353     instanceType = \''+str(instanceType)+'\';')
354     result = mycursor.fetchall()
355     resultList = []
356     for i in result:
357         resultList.append(str(i)[2:-3])
358     resultList = [float(ele) for ele in resultList]
359     if(len(resultList)==0):
360         avg = 0
361     else:
362         avg = sum(resultList)/len(resultList)
363     return str(avg)
364
365 def getCPU(instanceType, serviceType):
366     mycursor.execute('select cpu from '+str(serviceType)+',performance where
367     instanceType = \''+str(instanceType)+'\';')
368     result = mycursor.fetchall()
369     resultList = []
370     for i in result:
371         resultList.append(str(i)[2:-3])
372     resultList = [float(ele) for ele in resultList]
373     if(len(resultList)==0):
374         avg = 0
375     else:
376         avg = sum(resultList)/len(resultList)
377     return str(avg)
378
379 def getMemory(instanceType, serviceType):
380     mycursor.execute('select memory from '+str(serviceType)+',performance where
```

```

370     instanceType = \''+str(instanceType)+'\';')
371
372     result = mycursor.fetchall()
373
374     resultList = []
375
376     for i in result:
377
378         resultList.append(str(i)[2:-3])
379
380     resultList = [float(ele) for ele in resultList]
381
382     if(len(resultList)==0):
383
384         avg = 0
385
386     else:
387
388         avg = sum(resultList)/len(resultList)
389
390     return str(avg)
391
392
393 def isServiceTypeNotLoaded(serviceType):
394
395     mycursor.execute('select * from derivedValues where serviceType=\''+str(
396     serviceType)+'\';')
397
398     result = mycursor.fetchall()
399
400     if result == []:
401
402         return True
403
404     else:
405
406         return False
407
408
409 def isInstanceTypeNotLoaded(instanceType):
410
411     mycursor.execute('select * from derivedValues where instanceType=\''+str(
412     instanceType)+'\';')
413
414     result = mycursor.fetchall()
415
416     if result == []:
417
418         return True
419
420     else:
421
422         return False
423
424
425 def insertDerivedValues(elapsed, latency, connect, cpu, memory,
426
427     standardDeviation, errorRate, throughput, sessionID, endpointID,
428
429     instanceType, serviceType, timestamp):
430
431     mycursor.execute('insert into derivedValues values(\''+str(elapsed)+'\', \''
432     '+str(latency)+'\', \''+str(connect)+'\', \''+str(cpu)+'\', \''+str(memory)
433     +'\', \''+str(standardDeviation)+'\', \''+str(errorRate)+'\', \''+str(
434     throughput)+'\', \''+str(serviceType)+'\', \''
435     +str(instanceType)+'\', \''+str(timestamp)+'\', \''+str(sessionID)+'\', \''
436     +str(endpointID)+'\');')
437
438

```

```

409     db.commit()
410
411 def updateExistingDerivedValues(elapsed, latency, connect, cpu, memory,
412     standardDeviation, errorRate, throughput, sessionID, endpointID,
413     instanceType, serviceType, timestamp):
414     mycursor.execute('update derivedValues set elapsed = \''+str(elapsed)+'\',
415     Latency = \''+str(latency)+'\', Connect = \''+
416     str(connect)+'\', cpu = \''+str(cpu)+'\', memory = \''+str(memory)+'\',
417     stdDev = \''+str(standardDeviation)+'\', errorRate = \''+str(errorRate)+'
418     \', Throughput = \''+str(throughput)+'
419     '\', lastUpdated = \''+str(timestamp)+'\', sessionID = \''+str(sessionID)+'
420     '\', endpointID = \''+str(endpointID)+'
421     '\', where instanceType = \''+str(instanceType)+'\', and serviceType = \''+str
422     (serviceType)+'\';')
423     db.commit()
424
425 def updateDerivedValues(elapsed, latency, connect, cpu, memory,
426     standardDeviation, errorRate, throughput, sessionID, endpointID,
427     instanceType, serviceType, timestamp):
428     serviceTypeFlag = 0
429     instanceTypeFlag = 0
430     if(isServiceTypeNotLoaded(serviceType)):
431         serviceTypeFlag = 1
432     if(isInstanceTypeNotLoaded(instanceType)):
433         instanceTypeFlag = 1
434     if(serviceTypeFlag or instanceTypeFlag):
435         insertDerivedValues(elapsed, latency, connect, cpu, memory,
436     standardDeviation, errorRate, throughput, sessionID, endpointID,
437     instanceType, serviceType, timestamp)
438     else:
439         updateExistingDerivedValues(elapsed, latency, connect, cpu, memory,
440     standardDeviation, errorRate, throughput, sessionID, endpointID,
441     instanceType, serviceType, timestamp)
442
443 def updateDerivedCalculationLogs(endpointID, sessionID, serviceType,
444     instanceType, timestamp):
445     mycursor.execute('insert into derivedCalculationLogs values(\''+str(
446     endpointID)+'\', \''+str(sessionID)+'\', \''+str(serviceType)+'\', \''+str(
447     instanceType)+'\', \''+str(timestamp)+'\');')

```

```
432     db.commit()  
433  
434 def getNameFromEndpointID(endpointID):  
435     mycursor.execute('select user from endpointData where endpointID=\''+str(  
    endpointID)+'\';')  
436     result = mycursor.fetchall()  
437     result = str(result)[3:-4]  
438     return result  
439  
440 def getLatestID():  
441     mycursor.execute('select endpointID from endpointData where endpointID = (  
        select max(endpointid) from endpointData)')  
442     result = mycursor.fetchall()  
443     if(result == []):  
444         return 2  
445     else:  
446         result = str(result)[3]  
447         return int(result)+1  
448  
449 def getLatestCount(tableName):  
450     mycursor.execute('select count(*) from '+str(tableName))  
451     result = mycursor.fetchall()  
452     result = str(result)[2:-3]  
453     if(result == '0'):  
454         return 0  
455     else:  
456         return int(result)+1  
457  
458 def updateEndpointData(id, name):  
459     mycursor.execute('insert into endpointData values (\''+str(id)+'\', \''+str(  
    name)+'\');')  
460     db.commit()  
461  
462 def updateEndpointAccess(id, password):  
463     mycursor.execute('insert into endpointAccess values (\''+str(id)+'\', \''+  
    str(password)+'\');')  
464     db.commit()  
465  
466 def initEndpointLogs(id):
```

```
467     mycursor.execute('insert into endpointLogs values (\''+str(id)+'\', files =\n        \'\\') ;')
468     db.commit()
469
470 def createEndpoint(name, password):
471     id = getLatestID()
472     updateEndpointData(id, name)
473     updateEndpointAccess(id, password)
474     initEndpointLogs(id)
475     return id
476
477 def getRetrievedValue(fieldName, serviceType, instanceType):
478     mycursor.execute('select '+str(fieldName)+' from derivedValues where\n        serviceType=\''+str(serviceType)+'\'' and instanceType = \''+str(\n        instanceType)+'\';')
479     result = mycursor.fetchall()
480     result = str(result)[3:-4]
481     end_time = time.time()
482     return result
483
484
485 def insertPRequest(pNo, timestamp, filters, priorities, serviceType,
486     instanceType):
487     mycursor.execute('insert into processingRequests values (\''+str(pNo)+'\',
488         \''+str(filters)+'\', \''+str(priorities)+'\', \''+
489         str(serviceType)+'\', \''+str(instanceType)+'\', \''+str(timestamp)+'\');')
490     db.commit()
491
492 def getRelativeValue(serviceType, firstParam, secondParam, fieldName):
493     reversableValues = ['elapsed', 'stdDev', 'Latency', 'errorRate', 'Connect',
494         'cpu', 'memory', 'CPU']
495     firstVal = getRetrievedValue(fieldName, serviceType, firstParam)
496     secondVal = getRetrievedValue(fieldName, serviceType, secondParam)
497     result = float(firstVal) / float(secondVal)
498     if(fieldName in reversableValues):
499         result = 1/result
500     return result
```

```

500 def getRelativeRank(firstParam, secondParam, listOfFilters, priorities):
501     firstParamIndex = listOfFilters.index(firstParam)
502     firstValue = priorities[firstParamIndex]
503     firstValue = 1 / float(firstValue)
504     secondParamIndex = listOfFilters.index(secondParam)
505     secondValue = priorities[secondParamIndex]
506     secondValue = 1 / float(secondValue)
507     relativeRank = firstValue / secondValue
508     return relativeRank
509
510 def ahpCode(serviceType, listOfInstanceTypes, listOfFilters, priorities):
511     start_time = time.time()
512     comb = combinations(listOfInstanceTypes, 2)
513     instancesDict = {}
514     dictionaryList = []
515     for j in listOfFilters:
516         instancesDict.clear()
517         comb1 = combinations(listOfInstanceTypes, 2)
518         for i in list(comb1):
519             instancesDict[i] = getRelativeValue(serviceType, str(i[0]), str(i
5395 [1]), str(j))
520             dictionaryList.append(instancesDict.copy())
521     comb2 = combinations(listOfFilters, 2)
522     criteriaDict = {}
523     for i in list(comb2):
524         criteriaDict[i] = getRelativeRank(str(i[0]), str(i[1]), listOfFilters,
priorities)
525     end_time = time.time()
526     print("processing for ahp code done in:")
527     print("--- %s seconds ---" % (end_time - start_time))
528     result = ahp.ahpProcessing(dictionaryList, criteriaDict, listOfFilters,
listOfInstanceTypes)
529     return result
530
531 def getFilters(pno):
532     mycursor.execute('select filters from processingRequests where pno=' +str(
pno)+';')
533     result = mycursor.fetchall()
534     result = str(result)[3:-4]

```

```
535     return result
536
537 def getServiceType(pno):
538     mycursor.execute('select serviceType from processingRequests where pno=' +
539                      str(pno)+';')
540     result = mycursor.fetchall()
541     result = str(result)[3:-4]
542     return result
543
544 def getPriorities(pno):
545     mycursor.execute('select priorities from processingRequests where pno=' +
546                      str(pno)+';')
547     result = mycursor.fetchall()
548     result = str(result)[3:-4]
549     return result
550
551
552 def getInstanceType(pno):
553
554     mycursor.execute('select instanceType from processingRequests where pno=' +
555                      str(pno)+';')
556     result = mycursor.fetchall()
557     result = str(result)[3:-4]
558     return result
559
560
561 def updateTaskStatus(pno):
562     mycursor.execute('update taskStatus set status = \'Complete\' where pno = ' +
563                      str(pno)+';')
564     db.commit()
565
566
567 def insertAHPResult(result, pno, timestamp):
568     weights = result
569     resultStr = ''
570     weights = str(weights)
571     resultStr = resultStr + weights
572     mycursor.execute('insert into ahpResults values ('+str(pno)+', ' +
573                      resultStr+', '+str(timestamp)+');')
574     db.commit()
```

```
569 def updateResults(result, pNo, timestamp):
570     updateTaskStatus(pNo)
571     if(not isAS3Request(pNo)):
572         insertAHPResult(result, pNo, timestamp)
573
574 def checkProcessingStatus(pNo):
575     mycursor.execute('select status from taskStatus where pno=\''+str(pNo)+'\';')
576     result = mycursor.fetchall()
577     result = str(result)[3:-4]
578     if result == 'Complete':
579         return 1
580     else:
581         return 0
582
583 def checkIfProcessingIsDone(pNo):
584     mycursor.execute('select status from taskStatus where pno=\''+str(pNo)+'\';')
585     result = mycursor.fetchall()
586     result = str(result)[3:-4]
587     if result == 'Pending':
588         return 1
589     else:
590         return 0
591
592 def isAS3Request(pNo):
593     mycursor.execute('select serviceType from processingRequests where pno=\''+
594     str(pNo)+'\';')
595     result = mycursor.fetchall()
596     result = str(result)[3:-4]
597     if result == 's3':
598         return 1
599     else:
600         return 0
601
602 def saveEC2(index, sessionID, endpointID, instanceType, serviceType, elapsed,
603 latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
604 cpu, memory, performanceSize):
605     mycursor.execute('insert into saveec2 values(\''+str(index)+'\', \''+str(
```

```

    elapsed)+'\', \''+str(latency)+'\', \''+str(connect)+'\', \''+str(
    summarySize)+'\', \''+str(stdDev)+'\', \''+str(errorRate)+'\', \''+str(
    throughput)+'\', \'',
603   +str(performanceSize)+'\', \''+str(cpu)+'\', \''+str(memory)+'\', \''+str(
    instanceType)+'\', \''+str(endpointID)+'\', \''+str(sessionID)+'\', \''+str(
    timestamp)+'\');')
604   db.commit()
605
606 def saveRDS(index, sessionID, endpointID, instanceType, serviceType, elapsed,
607   latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
608   cpu, memory, performanceSize):
609   mycursor.execute('insert into saverds values(\''+str(index)+'\', \''+str(
610     elapsed)+'\', \''+str(latency)+'\', \''+str(connect)+'\', \''+str(
611     summarySize)+'\', \''+str(stdDev)+'\', \''+str(errorRate)+'\', \''+str(
612     throughput)+'\', \'',
613     +str(performanceSize)+'\', \''+str(cpu)+'\', \''+str(memory)+'\', \''+str(
614     instanceType)+'\', \''+str(endpointID)+'\', \''+str(sessionID)+'\', \''+str(
615     timestamp)+'\');')
616   db.commit()
617
618 def saveS3(index, sessionID, endpointID, instanceType, serviceType, elapsed,
619   latency, connect, stdDev, errorRate, throughput, summarySize, timestamp,
620   cpu, memory, performanceSize):
621   mycursor.execute('insert into saves3 values(\''+str(index)+'\', \''+str(
622     elapsed)+'\', \''+str(latency)+'\', \''+str(connect)+'\', \''+str(
623     summarySize)+'\', \''+str(stdDev)+'\', \''+str(errorRate)+'\', \''+str(
624     throughput)+'\', \'',
625     +str(performanceSize)+'\', \''+str(cpu)+'\', \''+str(memory)+'\', \''+str(
626     instanceType)+'\', \''+str(endpointID)+'\', \''+str(sessionID)+'\', \''+str(
627     timestamp)+'\');')
628   db.commit()
629
630 def getElapsedForML(instanceType, serviceType):
631   sdResultList = getElapsedValues(instanceType, serviceType)
632   noResultList = getNoOfSamples(instanceType, serviceType)
633   sdProductsList = []
634   for i in range(0, len(sdResultList)):
635     for j in range(1, int(noResultList[i])):
636       sdProductsList.append(sdResultList[i])

```

```
623     return sdProductsList
624
625 def getLatencyForML(instanceType, serviceType):
626     sdResultList = getLatencyValues(instanceType, serviceType)
627     noResultList = getNoOfSamples(instanceType, serviceType)
628     sdProductsList = []
629     for i in range(0, len(sdResultList)):
630         for j in range(1, int(noResultList[i])):
631             sdProductsList.append(sdResultList[i])
632     return sdProductsList
633
634 def getConnectForML(instanceType, serviceType):
635     sdResultList = getConnectValues(instanceType, serviceType)
636     noResultList = getNoOfSamples(instanceType, serviceType)
637     sdProductsList = []
638     for i in range(0, len(sdResultList)):
639         for j in range(1, int(noResultList[i])):
640             sdProductsList.append(sdResultList[i])
641     return sdProductsList
642
643 def getStandardDeviationForML(instanceType, serviceType):
644     sdResultList = getStandardDeviations(instanceType, serviceType)
645     noResultList = getNoOfSamples(instanceType, serviceType)
646     sdProductsList = []
647     for i in range(0, len(sdResultList)):
648         for j in range(1, int(noResultList[i])):
649             sdProductsList.append(sdResultList[i])
650     return sdProductsList
651
652 def getErrorRateForML(instanceType, serviceType):
653     erResultList = getErrorRates(instanceType, serviceType)
654     noResultList = getNoOfSamples(instanceType, serviceType)
655     erProductsList = []
656     for i in range(0, len(erResultList)):
657         for j in range(1, int(noResultList[i])):
658             erProductsList.append(erResultList[i])
659     return erProductsList
660
661 def getThroughputForML(instanceType, serviceType):
```

```

662     tpResultList = getThroughputs(instanceType, serviceType)
663     noResultList = getNoOfSamples(instanceType, serviceType)
664     tpProductsList = []
665     for i in range(0, len(tpResultList)):
666         for j in range(1, int(noResultList[i])):
667             tpProductsList.append(tpResultList[i])
668     return tpProductsList
669
670 def getResultFromAHP(processingNumber):
671     mycursor.execute('select results from ahpResults where pno=' +str(
672     processingNumber)+';')
673     result = mycursor.fetchall()
674     result = str(result)[3:-4]
675     print(result)
676     return result
677
678
679
680
681
682
683
684
685

```

**mlCode.py**

```

1 import databaseCommands as db
5592 from pandas import DataFrame
3 from sklearn import linear_model
4 import numpy as np
5 import strings
6
5597 def startProcessing(serviceType, instanceType, predictorValues, predictorNames):
7     :
8     inputList = []
9     print('predictorValues')
10    print(predictorValues)
5601    print('predictorNames')
12    print(predictorNames)
13    predictorNames = predictorNames.split(", ")
14    print(predictorNames)
15    elapsed = db.getElapsedForML(instanceType, serviceType)
16    if('elapsed' in predictorNames):
17        inputList.insert(predictorNames.index('elapsed'), elapsed)
18    else:

```

```
19         lastElement = elapsed
20         lastElementName = 'elapsed'
21     throughput = db.getThroughputForML(instanceType, serviceType)
22     if('Throughput' in predictorNames):
23         inputList.insert(predictorNames.index('Throughput'), throughput)
24     else:
25         lastElement = throughput
26         lastElementName = 'Throughput'
27     latency = db.getLatencyForML(instanceType, serviceType)
28     if('Latency' in predictorNames):
29         inputList.insert(predictorNames.index('Latency'), latency)
30     else:
31         lastElement = latency
32         lastElementName = 'Latency'
33     connect = db.getConnectForML(instanceType, serviceType)
34     if('Connect' in predictorNames):
35         inputList.insert(predictorNames.index('Connect'), connect)
36     else:
37         lastElement = connect
38         lastElementName = 'Connect'
39     stdDev = db.getStandardDeviationForML(instanceType, serviceType)
40     if('stdDev' in predictorNames):
41         inputList.insert(predictorNames.index('stdDev'), stdDev)
42     else:
43         lastElement = stdDev
44         lastElementName = 'stdDev'
45     errorRate = db.getErrorRateForML(instanceType, serviceType)
46     if('errorRate' in predictorNames):
47         inputList.insert(predictorNames.index('errorRate'), errorRate)
48     else:
49         lastElement = errorRate
50         lastElementName = 'errorRate'
51     inputList.append(lastElement)
52     return (inputList, lastElementName)
53
54 def runML(serviceType, instanceType, predictorValues, predictorNames):
55     res = startProcessing(serviceType, instanceType, predictorValues,
5645 predictorNames)
56     ipList = res[0]
```

```
57     lastElement = res[1]
58     predictorValues = predictorValues.split(", ")
59     predictorNames = predictorNames.split(", ")
60     predictorValues = [float(i) for i in predictorValues]
61     X = [ipList[:-1]]
62     xarray = np.array(ipList[:-1], 'float')
63     xarray = np.transpose(xarray)
64     y = [ipList[-1]]
65     yarray = np.array(ipList[-1], 'float')
66     yarray = np.transpose(yarray)
67     regr = linear_model.LinearRegression()
68     regr.fit(xarray, yarray)
69     predictedValue = regr.predict([predictorValues])
70     print(predictedValue)
71     predictedValueStr = 'For the entered values of '
72     selectedValuesDict = {}
73     finalDict = {}
74     for i in range(0, len(predictorValues)):
75         selectedValuesDict[strings.getDisplayName(str(predictorNames[i]))] =
76             round(float(predictorValues[i]), 2)
77     selectedValuesDict = str(selectedValuesDict)[1:-1]
78     finalSVDict = {}
79     finalSVDict["selectedValues"] = {selectedValuesDict}
80     predictedValueDict = {}
81     finalPVDict = {}
82     pv = float(str(predictedValue)[1:-1])
83     pvNo = str(round(pv, 2))
84     predictedValueDict[strings.getDisplayName(str(lastElement))] = pvNo
85     predictedValueDict = str(predictedValueDict)[1:-1]
86     finalPVDict["predictedValues"] = {predictedValueDict}
87     predictedValueRes = str(finalPVDict)[1:-1]
88     selectedValuesRes = str(finalSVDict)[1:-1]
89     finalData = selectedValuesRes + ', ' + predictedValueRes
90     finalData = str(finalData)
91     finalData = finalData.replace("'", '')
92     finalData = finalData.replace("''", '""')
93     finalData = '{' + finalData + '}'
94
95     return finalData
```

**ahp.py**

```
1 #!/usr/bin/env python3
5692 # -*- coding: utf-8 -*-
3
4 from CompareImpl import Compare
5 import json
6 import strings
5697 import time
8
9 def ahpProcessing(comparisonsList, criterionList, listOfFilters,
10     listOfInstanceTypes):
11     start_time = time.time()
12     ahpList = []
13     for i in range(0, len(comparisonsList)):
14         contentArea = Compare(listOfFilters[i], comparisonsList[i], precision
15 =3, random_index='saaty')
16         ahpList.append(contentArea)
17     end_time = time.time()
18     print("first for loop done in:")
19     print("--- %s seconds ---" % (end_time - start_time))
20     localWeights = []
21     localWeightsDict = {}
22     j = 0
23     subFilterTextDict = {}
24     finalDict = {}
25     subDisplayNameTextDict = {}
26     criterionWeights = []
27     criterionWeightsDict = {}
28     filterTextDict = {}
29     displayNameTextDict = {}
30     criteria = Compare('Ranks', criterionList, precision = 3, random_index=
31     'saaty')
32     criteria.add_children(ahpList)
33     criterionWeights.append(criteria.target_weights)
34     criterionWeightsDict["data"] = criteria.target_weights
```

```

32     filterTextDict["filterCompare"] = strings.getFilterText(criteria.
33         target_weights, "Overall")
34     displayNameTextDict["displayName"] = "Overall"
35     localWeightsRes = str(localWeightsDict)[-1]
36     criterionWeightsRes = str(criterionWeightsDict)[1:-1]
37     filterTextRes = str(filterTextDict)[1:-1]
38     displayNameRes = str(displayNameTextDict)[1:-1]
39     finalData = filterTextRes + ', ' + criterionWeightsRes + ', ' +
40         displayNameRes
41     finalDict["overall"] = {finalData[:]}
42     start_time=time.time()
43     for i in ahpList:
44         localWeightsDict["data"] = i.local_weights
45         subFilterTextDict["filterCompare"] = strings.getFilterText(i.
46             local_weights, strings.getDisplayName(listOfFilters[j]))
47         subDisplayNameTextDict["displayName"] = strings.getDisplayName(
48             listOfFilters[j])
49         localWeightsRes = str(localWeightsDict)[-1]
50         subFilterTextRes = str(subFilterTextDict)[-1]
51         subDisplayNameRes = str(subDisplayNameTextDict)[-1]
52         finalSubData = subFilterTextRes + ', ' + localWeightsRes + ', ' +
53         subDisplayNameRes
54         finalDict[listOfFilters[j]] = {finalSubData[:]}
55         j = j + 1
56     end_time = time.time()
57     print("second for loop done in:")
58     print("--- %s seconds ---" % (end_time - start_time))
59     finalDict = str(finalDict)
60     finalDict = finalDict.replace('\'', '\"')
61     finalDict = finalDict.replace('\"', '\"')
62     return finalDict

```

## strings.py

```

1 from itertools import combinations
2
3 def getFilterText(criterionWeights, filter):

```

```

4     weightsList = []
5     weightsDict = {}
6     for i in criterionWeights:
7         weightsList.append(criterionWeights[i])
8         weightsDict[criterionWeights[i]] = i
9     comb = combinations(weightsList, 2)
10    finalDict = {}
11    for i in list(comb):
12        firstValue = float(i[0])
13        secondValue = float(i[1])
14        res = (firstValue - secondValue) * 100 / secondValue
15        if(firstValue > secondValue):
16            message = str(weightsDict[i[0]]) + ' > ' + str(weightsDict[i[1]])
17        elif(firstValue < secondValue):
18            message = str(weightsDict[i[1]]) + ' > ' + str(weightsDict[i[0]])
19        else:
20            continue
21        finalDict[message] = str(round(res, 2))
22    return finalDict
23
24 def getDisplayName(filter):
25     if(filter == 'stdDev'):
26         return 'Consistency'
27     elif(filter == 'errorRate'):
28         return 'Error Rate'
29     elif(filter == 'elapsed'):
30         return 'Elapsed Time'
31     elif(filter == 'Connect'):
32         return 'Connection Time'
33     elif(filter == 'CPU'):
34         return 'CPU Utilization'
35     else:
36         return filter

```

5795

**DATABASE**

```

1 create table processingRequests(pno varchar(10) primary key, filters varchar
(500), priorities varchar(500), serviceType varchar(500), instanceType
varchar(500), timestamp varchar(100));

```

```

5802  create table ahpResults(pno varchar(10) primary key, results varchar(10000),
      timestamp varchar(100));
3   create table derivedCalculationLogs(endpointID varchar(30), sessionID varchar
      (30), serviceType varchar(30), instanceType varchar(30), timestamp varchar
      (30) primary key);
5804  create table derivedValues(elapsed varchar(30), Latency varchar(30), Connect
      varchar(30), cpu varchar(30), memory varchar(30), stdDev varchar(30),
      errorRate varchar(30), Throughput varchar(30), serviceType varchar(30),
      instanceType varchar(30), lastUpdated varchar(30) primary key, sessionID
      varchar(30), endpointID varchar(30));
5815  create table loginLogs(endpointID varchar(10), timestamp varchar(50) primary
      key);
6   create table endpointAccess(endpointID varchar(10) primary key, password
      varchar(50));
7   create table sessionDetails(sno varchar(15) primary key, status varchar(15),
      endpointID varchar(5));
5815
8   create table APILogs(callName varchar(30), timestamp varchar(30) primary key);
9   create table endpointLogs(endpointID varchar(10) primary key, files varchar
      (10000));
10
10  create table ec2summary(entryID varchar(50) primary key, label varchar(15),
      numberOfWorks varchar(15), average varchar(15), min varchar(10), max
      varchar(10), stdDev varchar(10), errorRate varchar(30), Throughput varchar
      (10), receivedKBps varchar(10), sentKBps varchar(10), averageBytes varchar
      (15), endpointID varchar(30), sessionID varchar(10), instanceType varchar
      (30), timestamp varchar(30));
5820
5821  create table ec2logs(entryID varchar(50) primary key, elapsed varchar(10),
      responseCode varchar(300), responseMessage varchar(300), success varchar
      (10), Latency varchar(10), Connect varchar(10), instanceType varchar(15),
      endpointID varchar(10), sessionID varchar(10), timestamp varchar(30));
12
12  create table ec2performance(entryID varchar(50) primary key, cpu varchar(10),
      memory varchar(10), instanceType varchar(15), endpointID varchar(10),
      sessionID varchar(10), timestamp varchar(30));
5830
13  create table endpointData(endpointID varchar(15) primary key, user varchar(30))
      ;
14  create table staticContent (name varchar(50) primary key, content varchar(500))
      ;
5835
15  create table taskStatus(pno varchar(15) primary key, status varchar(15));
16  insert into staticContent values('about-us', 'about-us');
17  create table saveec2(entryID varchar(50) primary key, elapsed varchar(30),
      
```

```

5840      Latency varchar(30), Connect varchar(30), summarySize varchar(30), stdDev
      varchar(30), errorRate varchar(30), Throughput varchar(30), performanceSize
      varchar(30), cpu varchar(30), memory varchar(30), instanceType varchar(30)
      , endpointID varchar(30), sessionID varchar(10), timestamp varchar(30));
18   create table saverds(entryID varchar(50) primary key, elapsed varchar(30),
      Latency varchar(30), Connect varchar(30), summarySize varchar(30), stdDev
      varchar(30), errorRate varchar(30), Throughput varchar(30), performanceSize
      varchar(30), cpu varchar(30), memory varchar(30), instanceType varchar(30)
      , endpointID varchar(30), sessionID varchar(10), timestamp varchar(30));
5845   create table saves3(entryID varchar(50) primary key, elapsed varchar(30),
      Latency varchar(30), Connect varchar(30), summarySize varchar(30), stdDev
      varchar(30), errorRate varchar(30), Throughput varchar(30), performanceSize
      varchar(30), cpu varchar(30), memory varchar(30), instanceType varchar(30)
      , endpointID varchar(30), sessionID varchar(10), timestamp varchar(30));
19   create table saves3(entryID varchar(50) primary key, elapsed varchar(30),
      Latency varchar(30), Connect varchar(30), summarySize varchar(30), stdDev
      varchar(30), errorRate varchar(30), Throughput varchar(30), performanceSize
      varchar(30), cpu varchar(30), memory varchar(30), instanceType varchar(30)
      , endpointID varchar(30), sessionID varchar(10), timestamp varchar(30));
5850

```

## ENDPOINTS

### EC2\_Test.jmx

```

1 <?xml version="1.0" encoding="UTF-8"?>
5862 <jmeterTestPlan version="1.2" properties="5.0" jmeter="5.4.1">
3   <hashTree>
4     <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan"
      " enabled="true">
5       <stringProp name="TestPlan.comments" />
5866     <boolProp name="TestPlan.functional_mode">false</boolProp>
7       <boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
8       <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
9       <elementProp name="TestPlan.user_defined_variables" elementType="

Arguments" guiclass="ArgumentsPanel" testclass="Arguments" testname="User
5870 Defined Variables" enabled="true">
10      <collectionProp name="Arguments.arguments" />
11    </elementProp>
12    <stringProp name="TestPlan.user_define_classpath" />
13  </TestPlan>
5874  <hashTree>
15    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"

```

```

16      testname="Users" enabled="true">
17          <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
18      <elementProp name="ThreadGroup.main_controller" elementType="LoopController" guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
19          <boolProp name="LoopController.continue_forever">false</boolProp>
20      <intProp name="LoopController.loops">-1</intProp>
21      </elementProp>
22      <stringProp name="ThreadGroup.num_threads">200</stringProp>
23      <stringProp name="ThreadGroup.ramp_time">1</stringProp>
24      <boolProp name="ThreadGroup.scheduler">true</boolProp>
25      <stringProp name="ThreadGroup.duration">3600</stringProp>
26      <stringProp name="ThreadGroup.delay">0</stringProp>
27      <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
28  </ThreadGroup>
29  <hashTree>
30      <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="HTTP Request" enabled="true">
31          <elementProp name="HTTPSampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
32              <collectionProp name="Arguments.arguments" />
33              <stringProp name="HTTPSampler.domain">ec2-15-206-185-135.ap-south-1.compute.amazonaws.com</stringProp>
34              <stringProp name="HTTPSampler.port" />
35              <stringProp name="HTTPSampler.protocol" />
36              <stringProp name="HTTPSampler.contentEncoding" />
37              <stringProp name="HTTPSampler.path">/#</stringProp>
38              <stringProp name="HTTPSampler.method">GET</stringProp>
39              <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
40              <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
41              <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
42              <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
43              <stringProp name="HTTPSampler.embedded_url_re" />
44              <stringProp name="HTTPSampler.connect_timeout" />

```

```
45      <stringProp name="HTTPSampler.response_timeout" />
46    </HTTPSamplerProxy>
47    <hashTree>
48      <ResultCollector guiclass="TableVisualizer" testclass=""
5920 ResultCollector" testname="View Results in Table" enabled="true">
49        <boolProp name="ResultCollector.error_logging">false</
50      boolProp>
51      <objProp>
52        <name>saveConfig</name>
53        <value class="SampleSaveConfiguration">
54          <time>true</time>
55          <latency>true</latency>
56          <timestampl>true</timestampl>
57          <success>true</success>
58          <label>true</label>
59          <code>true</code>
60          <message>true</message>
61          <threadName>true</threadName>
62          <dataType>true</dataType>
63          <encoding>false</encoding>
64          <assertions>true</assertions>
65          <subresults>true</subresults>
66          <responseData>false</responseData>
67          <samplerData>false</samplerData>
68          <xml>false</xml>
69          <fieldNames>true</fieldNames>
70          <responseHeaders>false</responseHeaders>
71          <requestHeaders>false</requestHeaders>
72          <responseDataOnError>false</responseDataOnError>
542          <saveAssertionResultsFailureMessage>true</
73        saveAssertionResultsFailureMessage>
74          <assertionsResultsToSave>0</assertionsResultsToSave>
75          <bytes>true</bytes>
76          <url>true</url>
77          <threadCounts>true</threadCounts>
78          <sampleCount>true</sampleCount>
79          <idleTime>true</idleTime>
80          <connectTime>true</connectTime>
81        </value>
```

```
581      </objProp>
582      <stringProp name="filename">D:\Jmeter Logs\logs\ec2\t2.nano\
583      ec2_t2.nano_2705212200_1.csv</stringProp>
584      </ResultCollector>
585      <hashTree />
586      <ResultCollector guiclass="SummaryReport" testclass="" testname="Summary Report" enabled="true">
587          <boolProp name="ResultCollector.error_logging">false</
588          boolProp>
589          <objProp>
590              <name>saveConfig</name>
591              <value class="SampleSaveConfiguration">
592                  <time>true</time>
593                  <latency>true</latency>
594                  <timestampl>true</timestampl>
595                  <success>true</success>
596                  <label>true</label>
597                  <code>true</code>
598                  <message>true</message>
599                  <threadName>true</threadName>
600                  <dataType>true</dataType>
601                  <encoding>false</encoding>
602                  <assertions>true</assertions>
603                  <subresults>true</subresults>
604                  <responseData>false</responseData>
605                  <samplerData>false</samplerData>
606                  <xml>false</xml>
607                  <fieldNames>true</fieldNames>
608                  <responseHeaders>false</responseHeaders>
609                  <requestHeaders>false</requestHeaders>
610                  <responseDataOnError>false</responseDataOnError>
611                  <saveAssertionResultsFailureMessage>true</
612                  saveAssertionResultsFailureMessage>
613                      <assertionsResultsToSave>0</assertionsResultsToSave>
614                      <bytes>true</bytes>
615                      <sentBytes>true</sentBytes>
616                      <url>true</url>
617                      <threadCounts>true</threadCounts>
618                      <idleTime>true</idleTime>
```

```
116          <connectTime>true</connectTime>
117      </value>
118  </objProp>
119  <stringProp name="filename" />
120 </ResultCollector>
121 <hashTree />
122 <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass=
123 UniformRandomTimer" testname="Uniform Random Timer" enabled="true">
124     <stringProp name="ConstantTimer.delay">5</stringProp>
125     <stringProp name="RandomTimer.range">100.0</stringProp>
126 </UniformRandomTimer>
127 <hashTree />
128 <kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.
129 vizualizers.PerfMonGui" testclass="kg.apc.jmeter.perfmon.PerfMonCollector"
130 testname="jp@gc - PerfMon Metrics Collector" enabled="true">
131     <boolProp name="ResultCollector.error_logging">false</
132 boolProp>
133 <objProp>
134     <name>saveConfig</name>
135     <value class="SampleSaveConfiguration">
136         <time>true</time>
137         <latency>true</latency>
138         <timestamp>true</timestamp>
139         <success>true</success>
140         <label>true</label>
141         <code>true</code>
142         <message>true</message>
143         <threadName>true</threadName>
144         <dataType>true</dataType>
145         <encoding>false</encoding>
146         <assertions>true</assertions>
147         <subresults>true</subresults>
148         <responseData>false</responseData>
149         <samplerData>false</samplerData>
150         <xml>false</xml>
151         <fieldNames>true</fieldNames>
152         <responseHeaders>false</responseHeaders>
153         <requestHeaders>false</requestHeaders>
154         <responseDataOnError>false</responseDataOnError>
```

```
151                     <saveAssertionResultsFailureMessage>true</
152             saveAssertionResultsFailureMessage>
153                 <assertionsResultsToSave>0</assertionsResultsToSave>
154                 <bytes>true</bytes>
155                 <sentBytes>true</sentBytes>
156                 <url>true</url>
157                 <threadCounts>true</threadCounts>
158                 <idleTime>true</idleTime>
159                 <connectTime>true</connectTime>
160             </value>
161         </objProp>
162         <stringProp name="filename" />
163         <longProp name="interval_grouping">1000</longProp>
164         <boolProp name="graph_aggregated">false</boolProp>
165         <stringProp name="include_sample_labels" />
166         <stringProp name="exclude_sample_labels" />
167         <stringProp name="start_offset" />
168         <stringProp name="end_offset" />
169         <boolProp name="include_checkbox_state">false</boolProp>
170         <boolProp name="exclude_checkbox_state">false</boolProp>
171         <collectionProp name="metricConnections">
172             <collectionProp name="413873611">
173                 <stringProp name="1417052243">15.206.185.135</
174             stringProp>
175                 <stringProp name="1600768">4444</stringProp>
176                 <stringProp name="66952">CPU</stringProp>
177                 <stringProp name="0" />
178             </collectionProp>
179             <collectionProp name="-1886840710">
180                 <stringProp name="1417052243">15.206.185.135</
181             stringProp>
182                 <stringProp name="1600768">4444</stringProp>
183                 <stringProp name="-1993889503">Memory</stringProp>
184                 <stringProp name="0" />
185             </collectionProp>
186             </collectionProp>
187         </kg.apc.jmeter.perfmon.PerfMonCollector>
188         <hashTree />
189     </hashTree >
```

```

187      </hashTree>
188
189      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
190      testname="Thread Group" enabled="true">
191          <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
192
193          <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
194          guiclass="LoopControlPanel" testclass="LoopController"
195          testname="Loop Controller" enabled="true">
196              <boolProp name="LoopController.continue_forever">false</boolProp>
197
198              <intProp name="LoopController.loops">-1</intProp>
199
200          </elementProp>
201
202          <stringProp name="ThreadGroup.num_threads">200</stringProp>
203
204          <stringProp name="ThreadGroup.ramp_time">1</stringProp>
205
206          <boolProp name="ThreadGroup.scheduler">true</boolProp>
207
208          <stringProp name="ThreadGroup.duration">3600</stringProp>
209
210          <stringProp name="ThreadGroup.delay">0</stringProp>
211
212          <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
213
214      </ThreadGroup>
215
216      <hashTree>
217
218          <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
219          testname="HTTP Request" enabled="true">
220
221              <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
222              " guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User
223              Defined Variables" enabled="true">
224
225                  <collectionProp name="Arguments.arguments" />
226
227              </elementProp>
228
229              <stringProp name="HTTPSampler.domain">ec2-52-66-210-87.ap-south
230              -1.compute.amazonaws.com</stringProp>
231
232              <stringProp name="HTTPSampler.port" />
233
234              <stringProp name="HTTPSampler.protocol" />
235
236              <stringProp name="HTTPSampler.contentEncoding" />
237
238              <stringProp name="HTTPSampler.path">/#</stringProp>
239
240              <stringProp name="HTTPSampler.method">GET</stringProp>
241
242              <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
243
244              <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
245
246              <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
247
248              <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>

```

```
216      <stringProp name="HTTPSampler.embedded_url_re" />
217      <stringProp name="HTTPSampler.connect_timeout" />
218      <stringProp name="HTTPSampler.response_timeout" />
219  </HTTPSamplerProxy>
220  <hashTree>
221      <ResultCollector guiclass="TableVisualizer" testclass=""
222 ResultCollector" testname="View Results in Table" enabled="true">
223          <boolProp name="ResultCollector.error_logging">false</
224 boolProp>
225          <objProp>
226              <name>saveConfig</name>
227              <value class="SampleSaveConfiguration">
228                  <time>true</time>
229                  <latency>true</latency>
230                  <timestamp>true</timestamp>
231                  <success>true</success>
232                  <label>true</label>
233                  <code>true</code>
234                  <message>true</message>
235                  <threadName>true</threadName>
236                  <dataType>true</dataType>
237                  <encoding>false</encoding>
238                  <assertions>true</assertions>
239                  <subresults>true</subresults>
240                  <responseData>false</responseData>
241                  <samplerData>false</samplerData>
242                  <xml>false</xml>
243                  <fieldNames>true</fieldNames>
244                  <responseHeaders>false</responseHeaders>
245                  <requestHeaders>false</requestHeaders>
246                  <responseDataOnError>false</responseDataOnError>
247                  <saveAssertionResultsFailureMessage>true</
248 saveAssertionResultsFailureMessage>
249                  <assertionsResultsToSave>0</assertionsResultsToSave>
250                  <bytes>true</bytes>
251                  <sentBytes>true</sentBytes>
252                  <url>true</url>
253                  <threadCounts>true</threadCounts>
254                  <sampleCount>true</sampleCount>
```

```
252             <idleTime>true</idleTime>
253             <connectTime>true</connectTime>
254         </value>
255     </objProp>
256     <stringProp name="filename">D:\Jmeter Logs\logs\ec2\t2.micro\
6155 ec2_t2.micro_2705212200_1.csv</stringProp>
257     </ResultCollector>
258     <hashTree />
259     <ResultCollector guiclass="SummaryReport" testclass=""
ResultCollector" testname="Summary Report" enabled="true">
260         <boolProp name="ResultCollector.error_logging">false</
boolProp>
261         <objProp>
262             <name>saveConfig </name>
263             <value class="SampleSaveConfiguration">
264                 <time>true</time>
265                 <latency>true</latency>
266                 <timestamp>true</timestamp>
267                 <success>true</success>
268                 <label>true</label>
269                 <code>true</code>
270                 <message>true</message>
271                 <threadName>true</threadName>
272                 <dataType>true</dataType>
273                 <encoding>false</encoding>
274                 <assertions>true</assertions>
275                 <subresults>true</subresults>
276                 <responseData>false</responseData>
277                 <samplerData>false</samplerData>
278                 <xml>false</xml>
279                 <fieldNames>true</fieldNames>
280                 <responseHeaders>false</responseHeaders>
281                 <requestHeaders>false</requestHeaders>
282                 <responseDataOnError>false</responseDataOnError>
283                 <saveAssertionResultsFailureMessage>true</
6185 saveAssertionResultsFailureMessage>
284                 <assertionsResultsToSave>0</assertionsResultsToSave>
285                 <bytes>true</bytes>
286                 <sentBytes>true</sentBytes>
```

```
287             <url>true</url>
288             <threadCounts>true</threadCounts>
289             <idleTime>true</idleTime>
290             <connectTime>true</connectTime>
291         </value>
292     </objProp>
293     <stringProp name="filename" />
294 </ResultCollector>
295 <hashTree />
296 <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer" testname="Uniform Random Timer" enabled="true">
297     <stringProp name="ConstantTimer.delay">5</stringProp>
298     <stringProp name="RandomTimer.range">100.0</stringProp>
299 </UniformRandomTimer>
300 <hashTree />
301 <kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.vizualizers.PerfMonGui" testclass="kg.apc.jmeter.perfmon.PerfMonCollector" testname="jp@gc - PerfMon Metrics Collector" enabled="true">
302     <boolProp name="ResultCollector.error_logging">false</boolProp>
303 <objProp>
304     <name>saveConfig</name>
305     <value class="SampleSaveConfiguration">
306         <time>true</time>
307         <latency>true</latency>
308         <timestamp>true</timestamp>
309         <success>true</success>
310         <label>true</label>
311         <code>true</code>
312         <message>true</message>
313         <threadName>true</threadName>
314         <dataType>true</dataType>
315         <encoding>false</encoding>
316         <assertions>true</assertions>
317         <subresults>true</subresults>
318         <responseData>false</responseData>
319         <samplerData>false</samplerData>
320         <xml>false</xml>
321         <fieldNames>true</fieldNames>
```

```
322             <responseHeaders>false</responseHeaders>
323             <requestHeaders>false</requestHeaders>
324             <responseDataOnError>false</responseDataOnError>
325             <saveAssertionResultsFailureMessage>true</
326             saveAssertionResultsFailureMessage>
327                 <assertionsResultsToSave>0</assertionsResultsToSave>
328                 <bytes>true</bytes>
329                 <sentBytes>true</sentBytes>
330                 <url>true</url>
331                 <threadCounts>true</threadCounts>
332                 <idleTime>true</idleTime>
333                 <connectTime>true</connectTime>
334             </value>
335         </objProp>
336         <stringProp name="filename" />
337         <longProp name="interval_grouping">1000</longProp>
338         <boolProp name="graph_aggregated">false</boolProp>
339         <stringProp name="include_sample_labels" />
340         <stringProp name="exclude_sample_labels" />
341         <stringProp name="start_offset" />
342         <stringProp name="end_offset" />
343         <boolProp name="include_checkbox_state">false</boolProp>
344         <boolProp name="exclude_checkbox_state">false</boolProp>
345         <collectionProp name="metricConnections">
346             <collectionProp name="-869074490">
347                 <stringProp name="-1693625063">52.66.210.87</stringProp>
348             >
349                 <stringProp name="1600768">4444</stringProp>
350                 <stringProp name="66952">CPU</stringProp>
351                 <stringProp name="0" />
352             </collectionProp>
353             <collectionProp name="1125178485">
354                 <stringProp name="-1693625063">52.66.210.87</stringProp>
355             >
356                 <stringProp name="1600768">4444</stringProp>
357                 <stringProp name="-1993889503">Memory</stringProp>
358                 <stringProp name="0" />
359             </collectionProp>
360         </collectionProp>
```

```

358         </kg.apc.jmeter.perfmon.PerfMonCollector>
359         <hashTree />
360     </hashTree>
361   </hashTree>
362   <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
363   testname="Thread Group" enabled="true">
364     <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
365     <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
366     guiclass="LoopControlPanel" testclass="LoopController"
367     testname="Loop Controller" enabled="true">
368       <boolProp name="LoopController.continue_forever">false</boolProp>
369       <intProp name="LoopController.loops">-1</intProp>
370     </elementProp>
371     <stringProp name="ThreadGroup.num_threads">200</stringProp>
372     <stringProp name="ThreadGroup.ramp_time">1</stringProp>
373     <boolProp name="ThreadGroup.scheduler">true</boolProp>
374     <stringProp name="ThreadGroup.duration">3600</stringProp>
375     <stringProp name="ThreadGroup.delay">0</stringProp>
376     <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
377   </ThreadGroup>
378   <hashTree>
379     <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
380     testname="HTTP Request" enabled="true">
381       <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
382       " guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User
383       Defined Variables" enabled="true">
384         <collectionProp name="Arguments.arguments" />
385       </elementProp>
386       <stringProp name="HTTPSampler.domain">ec2-13-127-208-47.ap-south
387       -1.compute.amazonaws.com</stringProp>
388       <stringProp name="HTTPSampler.port" />
389       <stringProp name="HTTPSampler.protocol" />
390       <stringProp name="HTTPSampler.contentEncoding" />
391       <stringProp name="HTTPSampler.path">/#</stringProp>
392       <stringProp name="HTTPSampler.method">GET</stringProp>
393       <boolProp name="HTTPSampler.follow_redirects">true</boolProp>

```

```
387      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
388      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
389      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
390      <stringProp name="HTTPSampler.embedded_url_re" />
391      <stringProp name="HTTPSampler.connect_timeout" />
392      <stringProp name="HTTPSampler.response_timeout" />
393    </HTTPSamplerProxy>
394    <hashTree>
395      <ResultCollector guiclass="TableVisualizer" testclass=""
6315  ResultCollector" testname="View Results in Table" enabled="true">
396        <boolProp name="ResultCollector.error_logging">false</
397          boolProp>
398        <objProp>
399          <name>saveConfig </name>
400          <value class="SampleSaveConfiguration">
401            <time>true</time>
402            <latency>true</latency>
403            <timestamp>true</timestamp>
404            <success>true</success>
405            <label>true</label>
406            <code>true</code>
407            <message>true</message>
408            <threadName>true</threadName>
409            <dataType>true</dataType>
410            <encoding>false</encoding>
411            <assertions>true</assertions>
412            <subresults>true</subresults>
413            <responseData>false</responseData>
414            <samplerData>false</samplerData>
415            <xml>false</xml>
416            <fieldNames>true</fieldNames>
417            <responseHeaders>false</responseHeaders>
418            <requestHeaders>false</requestHeaders>
419            <responseDataOnError>false</responseDataOnError>
420            <saveAssertionResultsFailureMessage>true</
421              saveAssertionResultsFailureMessage>
422              <assertionsResultsToSave>0</assertionsResultsToSave>
423              <bytes>true</bytes>
424              <sentBytes>true</sentBytes>
```

```
423             <url>true</url>
424             <threadCounts>true</threadCounts>
425             <sampleCount>true</sampleCount>
426             <idleTime>true</idleTime>
427             <connectTime>true</connectTime>
428         </value>
429     </objProp>
430     <stringProp name="filename">D:\Jmeter Logs\logs\ec2\t2.medium
431 \ec2_t2.medium_2705212200_l.csv</stringProp>
432     </ResultCollector>
433     <hashTree />
434     <ResultCollector guiclass="SummaryReport" testclass="
435 ResultCollector" testname="Summary Report" enabled="true">
436         <boolProp name="ResultCollector.error_logging">false</
437 boolProp>
438     <objProp>
439         <name>saveConfig</name>
440         <value class="SampleSaveConfiguration">
441             <time>true</time>
442             <latency>true</latency>
443             <timestampl>true</timestampl>
444             <success>true</success>
445             <label>true</label>
446             <code>true</code>
447             <message>true</message>
448             <threadName>true</threadName>
449             <dataType>true</dataType>
450             <encoding>false</encoding>
451             <assertions>true</assertions>
452             <subresults>true</subresults>
453             <responseData>false</responseData>
454             <samplerData>false</samplerData>
455             <xml>false</xml>
456             <fieldNames>true</fieldNames>
457             <responseHeaders>false</responseHeaders>
458             <requestHeaders>false</requestHeaders>
459             <responseDataOnError>false</responseDataOnError>
460             <saveAssertionResultsFailureMessage>true</
461 saveAssertionResultsFailureMessage>
```

```
458             <assertionsResultsToSave>0</assertionsResultsToSave>
459             <bytes>true</bytes>
460             <sentBytes>true</sentBytes>
461             <url>true</url>
462             <threadCounts>true</threadCounts>
463             <idleTime>true</idleTime>
464             <connectTime>true</connectTime>
465         </value>
466     </objProp>
467     <stringProp name="filename" />
468 </ResultCollector>
469 <hashTree />
470 <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass=
471 UniformRandomTimer" testname="Uniform Random Timer" enabled="true">
472     <stringProp name="ConstantTimer.delay">5</stringProp>
473     <stringProp name="RandomTimer.range">100.0</stringProp>
474 </UniformRandomTimer>
475 <hashTree />
476 <kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.
477 vizualizers.PerfMonGui" testclass="kg.apc.jmeter.perfmon.PerfMonCollector"
478 testname="jp@gc - PerfMon Metrics Collector" enabled="true">
479     <boolProp name="ResultCollector.error_logging">false</
480 boolProp>
481     <objProp>
482         <name>saveConfig</name>
483         <value class="SampleSaveConfiguration">
484             <time>true</time>
485             <latency>true</latency>
486             <timestamp>true</timestamp>
487             <success>true</success>
488             <label>true</label>
489             <code>true</code>
490             <message>true</message>
491             <threadName>true</threadName>
492             <dataType>true</dataType>
493             <encoding>false</encoding>
494             <assertions>true</assertions>
495             <subresults>true</subresults>
496             <responseData>false</responseData>
```

```
493             <samplerData>false</samplerData>
494             <xml>false</xml>
495             <fieldNames>true</fieldNames>
496             <responseHeaders>false</responseHeaders>
497             <requestHeaders>false</requestHeaders>
498             <responseDataOnError>false</responseDataOnError>
499             <saveAssertionResultsFailureMessage>true</
6430         saveAssertionResultsFailureMessage>
500             <assertionsResultsToSave>0</assertionsResultsToSave>
501             <bytes>true</bytes>
502             <sentBytes>true</sentBytes>
503             <url>true</url>
504             <threadCounts>true</threadCounts>
505             <idleTime>true</idleTime>
506             <connectTime>true</connectTime>
507             </value>
508         </objProp>
509         <stringProp name="filename" />
510         <longProp name="interval_grouping">1000</longProp>
511         <boolProp name="graph_aggregated">false</boolProp>
512         <stringProp name="include_sample_labels" />
513         <stringProp name="exclude_sample_labels" />
514         <stringProp name="start_offset" />
515         <stringProp name="end_offset" />
516         <boolProp name="include_checkbox_state">false</boolProp>
517         <boolProp name="exclude_checkbox_state">false</boolProp>
518         <collectionProp name="metricConnections" >
519             <collectionProp name="1718702701">
520                 <stringProp name="-758330053">13.127.208.47</stringProp>
521             >
522                 <stringProp name="1600768">4444</stringProp>
523                 <stringProp name="66952">CPU</stringProp>
524                 <stringProp name="0" />
525             </collectionProp>
526             <collectionProp name="-582011620">
527                 <stringProp name="-758330053">13.127.208.47</stringProp>
528             >
529                 <stringProp name="1600768">4444</stringProp>
530                 <stringProp name="-1993889503">Memory</stringProp>
```

```

529             <stringProp name="0" />
530         </collectionProp>
531     </collectionProp>
532     </kg.apc.jmeter.perfmon.PerfMonCollector>
533     <hashTree />
534   </hashTree>
535 </hashTree>
536 <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
537 testname="Thread Group" enabled="true">
538     <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
539     <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
540 guiclass="LoopControlPanel" testclass="LoopController"
541 testname="Loop Controller" enabled="true">
542         <boolProp name="LoopController.continue_forever">false</boolProp>
543         <intProp name="LoopController.loops">-1</intProp>
544     </elementProp>
545     <stringProp name="ThreadGroup.num_threads">200</stringProp>
546     <stringProp name="ThreadGroup.ramp_time">1</stringProp>
547     <boolProp name="ThreadGroup.scheduler">true</boolProp>
548     <stringProp name="ThreadGroup.duration">3600</stringProp>
549     <stringProp name="ThreadGroup.delay">0</stringProp>
550     <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
551   </ThreadGroup>
552   <hashTree>
553     <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
554 testname="HTTP Request" enabled="true">
555         <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
556 " guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User
557 Defined Variables" enabled="true">
558             <collectionProp name="Arguments.arguments" />
559         </elementProp>
560         <stringProp name="HTTPSampler.domain">ec2-65-2-126-58.ap-south
561 -1.compute.amazonaws.com</stringProp>
562         <stringProp name="HTTPSampler.port" />
563         <stringProp name="HTTPSampler.protocol" />
564         <stringProp name="HTTPSampler.contentEncoding" />

```

```

558     <stringProp name="HTTPSampler.path">/#</stringProp>
559     <stringProp name="HTTPSampler.method">GET</stringProp>
560     <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
561     <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
562     <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
563     <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
564     <stringProp name="HTTPSampler.embedded_url_re" />
565     <stringProp name="HTTPSampler.connect_timeout" />
566     <stringProp name="HTTPSampler.response_timeout" />
567 </HTTPSamplerProxy>
568 <hashTree>
569     <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table" enabled="true">
570         <boolProp name="ResultCollector.error_logging">false</boolProp>
571         <objProp>
572             <name>saveConfig</name>
573             <value class="SampleSaveConfiguration">
574                 <time>true</time>
575                 <latency>true</latency>
576                 <timestampl>true</timestampl>
577                 <success>true</success>
578                 <label>true</label>
579                 <code>true</code>
580                 <message>true</message>
581                 <threadName>true</threadName>
582                 <dataType>true</dataType>
583                 <encoding>false</encoding>
584                 <assertions>true</assertions>
585                 <subresults>true</subresults>
586                 <responseData>false</responseData>
587                 <samplerData>false</samplerData>
588                 <xml>false</xml>
589                 <fieldNames>true</fieldNames>
590                 <responseHeaders>false</responseHeaders>
591                 <requestHeaders>false</requestHeaders>
592                 <responseDataOnError>false</responseDataOnError>
593                 <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>

```

```
594         <assertionsResultsToSave>0</assertionsResultsToSave>
595         <bytes>true</bytes>
596         <sentBytes>true</sentBytes>
597         <url>true</url>
598         <threadCounts>true</threadCounts>
599         <sampleCount>true</sampleCount>
600         <idleTime>true</idleTime>
601         <connectTime>true</connectTime>
602         </value>
603     </objProp>
604     <stringProp name="filename">D:\Jmeter Logs\logs\ec2\t2.large\
605     ec2_t2.large_2705212200_1.csv</stringProp>
606     </ResultCollector>
607     <hashTree />
608     <ResultCollector guiclass="SummaryReport" testclass=""
609     ResultCollector" testname="Summary Report" enabled="true">
610         <boolProp name="ResultCollector.error_logging">false</
611     boolProp>
612         <objProp>
613             <name>saveConfig</name>
614             <value class="SampleSaveConfiguration">
615                 <time>true</time>
616                 <latency>true</latency>
617                 <timestamp>true</timestamp>
618                 <success>true</success>
619                 <label>true</label>
620                 <code>true</code>
621                 <message>true</message>
622                 <threadName>true</threadName>
623                 <dataType>true</dataType>
624                 <encoding>false</encoding>
625                 <assertions>true</assertions>
626                 <subresults>true</subresults>
627                 <responseData>false</responseData>
628                 <samplerData>false</samplerData>
629                 <xml>false</xml>
630                 <fieldNames>true</fieldNames>
631                 <responseHeaders>false</responseHeaders>
632                 <requestHeaders>false</requestHeaders>
```

```

630          <responseDataOnError>false</responseDataOnError>
631          <saveAssertionResultsFailureMessage>true</
632      saveAssertionResultsFailureMessage>
633          <assertionsResultsToSave>0</assertionsResultsToSave>
634          <bytes>true</bytes>
635          <sentBytes>true</sentBytes>
636          <url>true</url>
637          <threadCounts>true</threadCounts>
638          <idleTime>true</idleTime>
639          <connectTime>true</connectTime>
640      </value>
641      <stringProp name="filename" />
642  </ResultCollector>
643  <hashTree />
644  <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass=""
645  UniformRandomTimer" testname="Uniform Random Timer" enabled="true">
646      <stringProp name="ConstantTimer.delay">5</stringProp>
647      <stringProp name="RandomTimer.range">100.0</stringProp>
648  </UniformRandomTimer>
649  <hashTree />
650  <kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.
651  vizualizers.PerfMonGui" testclass="kg.apc.jmeter.perfmon.PerfMonCollector"
652  testname="jp@gc - PerfMon Metrics Collector" enabled="true">
653      <boolProp name="ResultCollector.error_logging">false</
654  boolProp>
655      <objProp>
656          <name>saveConfig</name>
657          <value class="SampleSaveConfiguration">
658              <time>true</time>
659              <latency>true</latency>
660              <timestamp>true</timestamp>
661              <success>true</success>
662              <label>true</label>
663              <code>true</code>
664              <message>true</message>
665              <threadName>true</threadName>
666              <dataType>true</dataType>
667              <encoding>false</encoding>

```

```
664             <assertions>true</assertions>
665             <subresults>true</subresults>
666             <responseData>false</responseData>
667             <samplerData>false</samplerData>
668             <xml>false</xml>
669             <fieldNames>true</fieldNames>
670             <responseHeaders>false</responseHeaders>
671             <requestHeaders>false</requestHeaders>
672             <responseDataOnError>false</responseDataOnError>
673             <saveAssertionResultsFailureMessage>true</
674             saveAssertionResultsFailureMessage>
675                 <assertionsResultsToSave>0</assertionsResultsToSave>
676                 <bytes>true</bytes>
677                 <sentBytes>true</sentBytes>
678                 <url>true</url>
679                 <threadCounts>true</threadCounts>
680                 <idleTime>true</idleTime>
681                 <connectTime>true</connectTime>
682                     </value>
683             </objProp>
684             <stringProp name="filename" />
685             <longProp name="interval_grouping">1000</longProp>
686             <boolProp name="graph_aggregated">false</boolProp>
687             <stringProp name="include_sample_labels" />
688             <stringProp name="exclude_sample_labels" />
689             <stringProp name="start_offset" />
690             <stringProp name="end_offset" />
691             <boolProp name="include_checkbox_state">false</boolProp>
692             <boolProp name="exclude_checkbox_state">false</boolProp>
693             <collectionProp name="metricConnections">
694                 <collectionProp name="908413826">
695                     <stringProp name="-1047607161">65.2.126.58</stringProp>
696                     <stringProp name="1600768">4444</stringProp>
697                     <stringProp name="66952">CPU</stringProp>
698                     <stringProp name="0" />
699                 </collectionProp>
700                 <collectionProp name="-1392300495">
701                     <stringProp name="-1047607161">65.2.126.58</stringProp>
702                     <stringProp name="1600768">4444</stringProp>
```

```

702             <stringProp name="-1993889503">Memory</stringProp>
703             <stringProp name="0" />
704         </collectionProp>
705     </collectionProp>
706     </kg.apc.jmeter.perfmon.PerfMonCollector>
707     <hashTree />
708   </hashTree>
709 </hashTree>
710   <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
711 testname="Thread Group" enabled="true">
712     <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
713     <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
714 guiclass="LoopControlPanel" testclass="LoopController"
715 testname="Loop Controller" enabled="true">
716       <boolProp name="LoopController.continue_forever">false</boolProp>
717     <intProp name="LoopController.loops">-1</intProp>
718   </elementProp>
719   <stringProp name="ThreadGroup.num_threads">200</stringProp>
720   <stringProp name="ThreadGroup.ramp_time">1</stringProp>
721   <boolProp name="ThreadGroup.scheduler">true</boolProp>
722   <stringProp name="ThreadGroup.duration">3600</stringProp>
723   <stringProp name="ThreadGroup.delay">0</stringProp>
724   <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
725 </ThreadGroup>
726   <hashTree>
727     <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
728 testname="HTTP Request" enabled="true">
729       <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
730 " guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User
731 Defined Variables" enabled="true">
732         <collectionProp name="Arguments.arguments" />
733       </elementProp>
734       <stringProp name="HTTPSampler.domain">ec2-13-232-6-88.ap-south
735 -1.compute.amazonaws.com</stringProp>
736       <stringProp name="HTTPSampler.port" />
737       <stringProp name="HTTPSampler.protocol" />

```

```

731      <stringProp name="HTTPSampler.contentEncoding" />
732      <stringProp name="HTTPSampler.path">/#</stringProp>
733      <stringProp name="HTTPSampler.method">GET</stringProp>
734      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
735      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
736      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
737      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
738      <stringProp name="HTTPSampler.embedded_url_re" />
739      <stringProp name="HTTPSampler.connect_timeout" />
740      <stringProp name="HTTPSampler.response_timeout" />
741  </HTTPSamplerProxy>
742  <hashTree>
743      <ResultCollector guiclass="TableVisualizer" testclass=""
    ResultCollector" testname="View Results in Table" enabled="true">
744          <boolProp name="ResultCollector.error_logging">false</
    boolProp>
745          <objProp>
746              <name>saveConfig</name>
747              <value class="SampleSaveConfiguration">
748                  <time>true</time>
749                  <latency>true</latency>
750                  <timestamp>true</timestamp>
751                  <success>true</success>
752                  <label>true</label>
753                  <code>true</code>
754                  <message>true</message>
755                  <threadName>true</threadName>
756                  <dataType>true</dataType>
757                  <encoding>false</encoding>
758                  <assertions>true</assertions>
759                  <subresults>true</subresults>
760                  <responseData>false</responseData>
761                  <samplerData>false</samplerData>
762                  <xml>false</xml>
763                  <fieldNames>true</fieldNames>
764                  <responseHeaders>false</responseHeaders>
765                  <requestHeaders>false</requestHeaders>
766                  <responseDataOnError>false</responseDataOnError>
767                  <saveAssertionResultsFailureMessage>true</

```

```
6735     saveAssertionResultsFailureMessage >
6736         <assertionsResultsToSave>0</assertionsResultsToSave>
6737         <bytes>true</bytes>
6738         <sentBytes>true</sentBytes>
6739         <url>true</url>
6740         <threadCounts>true</threadCounts>
6741         <sampleCount>true</sampleCount>
6742         <idleTime>true</idleTime>
6743         <connectTime>true</connectTime>
6744         </value>
6745     </objProp>
6746     <stringProp name="filename">D:\Jmeter Logs\logs\ec2\t2.small\
6747 ec2_t2.small_2705212200_1.csv</stringProp>
6748     </ResultCollector>
6749     <hashTree />
6750     <ResultCollector guiclass="SummaryReport" testclass=""
6751 ResultCollector" testname="Summary Report" enabled="true">
6752         <boolProp name="ResultCollector.error_logging">false</
6753 boolProp>
6754     <objProp>
6755         <name>saveConfig </name>
6756         <value class="SampleSaveConfiguration">
6757             <time>true</time>
6758             <latency>true</latency>
6759             <timestamp>true</timestamp>
6760             <success>true</success>
6761             <label>true</label>
6762             <code>true</code>
6763             <message>true</message>
6764             <threadName>true</threadName>
6765             <dataType>true</dataType>
6766             <encoding>false</encoding>
6767             <assertions>true</assertions>
6768             <subresults>true</subresults>
6769             <responseData>false</responseData>
6770             <samplerData>false</samplerData>
6771             <xml>false</xml>
6772             <fieldNames>true</fieldNames>
6773             <responseHeaders>false</responseHeaders>
```

```

803             <requestHeaders>false</requestHeaders>
804             <responseDataOnError>false</responseDataOnError>
805             <saveAssertionResultsFailureMessage>true</
806             saveAssertionResultsFailureMessage>
807                 <assertionsResultsToSave>0</assertionsResultsToSave>
808                 <bytes>true</bytes>
809                 <sentBytes>true</sentBytes>
810                 <url>true</url>
811                 <threadCounts>true</threadCounts>
812                 <idleTime>true</idleTime>
813                 <connectTime>true</connectTime>
814             </value>
815         </objProp>
816         <stringProp name="filename" />
817     </ResultCollector>
818     <hashTree />
819     <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass=
820 "UniformRandomTimer" testname="Uniform Random Timer" enabled="true">
821         <stringProp name="ConstantTimer.delay">5</stringProp>
822         <stringProp name="RandomTimer.range">100.0</stringProp>
823     </UniformRandomTimer>
824     <hashTree />
825     <kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.
826 vizualizers.PerfMonGui" testclass="kg.apc.jmeter.perfmon.PerfMonCollector"
827 testname="jp@gc - PerfMon Metrics Collector" enabled="true">
828         <boolProp name="ResultCollector.error_logging">false</
829     boolProp>
830         <objProp>
831             <name>saveConfig</name>
832             <value class="SampleSaveConfiguration">
833                 <time>true</time>
834                 <latency>true</latency>
835                 <timestamp>true</timestamp>
836                 <success>true</success>
837                 <label>true</label>
838                 <code>true</code>
839                 <message>true</message>
840                 <threadName>true</threadName>
841                 <dataType>true</dataType>

```

```
837      <encoding>false</encoding>
838      <assertions>true</assertions>
839      <subresults>true</subresults>
840      <responseData>false</responseData>
841      <samplerData>false</samplerData>
842      <xml>false</xml>
843      <fieldNames>true</fieldNames>
844      <responseHeaders>false</responseHeaders>
845      <requestHeaders>false</requestHeaders>
846      <responseDataOnError>false</responseDataOnError>
847      <saveAssertionResultsFailureMessage>true</
848      saveAssertionResultsFailureMessage>
849          <assertionsResultsToSave>0</assertionsResultsToSave>
850          <bytes>true</bytes>
851          <sentBytes>true</sentBytes>
852          <url>true</url>
853          <threadCounts>true</threadCounts>
854          <idleTime>true</idleTime>
855          <connectTime>true</connectTime>
856      </value>
857  </objProp>
858  <stringProp name="filename" />
859  <longProp name="interval_grouping">1000</longProp>
860  <boolProp name="graph_aggregated">false</boolProp>
861  <stringProp name="include_sample_labels" />
862  <stringProp name="exclude_sample_labels" />
863  <stringProp name="start_offset" />
864  <stringProp name="end_offset" />
865  <boolProp name="include_checkbox_state">false</boolProp>
866  <boolProp name="exclude_checkbox_state">false</boolProp>
867  <collectionProp name="metricConnections">
868      <collectionProp name="-88133052">
869          <stringProp name="-92271679">13.232.6.88</stringProp>
870          <stringProp name="1600768">4444</stringProp>
871          <stringProp name="66952">CPU</stringProp>
872          <stringProp name="0" />
873      </collectionProp>
874      <collectionProp name="1906119923">
875          <stringProp name="-92271679">13.232.6.88</stringProp>
```

```

875             <stringProp name="1600768">4444</stringProp>
876             <stringProp name="-1993889503">Memory</stringProp>
877             <stringProp name="0" />
878         </collectionProp>
879     </collectionProp>
880     </kg.apc.jmeter.perfmon.PerfMonCollector>
881     <hashTree />
882   </hashTree>
883 </hashTree>
884 <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
885 testname="Thread Group" enabled="true">
886     <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
887     <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
888      guiclass="LoopControlPanel" testclass="LoopController"
889      testname="Loop Controller" enabled="true">
890         <boolProp name="LoopController.continue_forever">false</boolProp>
891         <intProp name="LoopController.loops">-1</intProp>
892     </elementProp>
893     <stringProp name="ThreadGroup.num_threads">200</stringProp>
894     <stringProp name="ThreadGroup.ramp_time">1</stringProp>
895     <boolProp name="ThreadGroup.scheduler">true</boolProp>
896     <stringProp name="ThreadGroup.duration">3600</stringProp>
897     <stringProp name="ThreadGroup.delay">0</stringProp>
898     <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
899   </ThreadGroup>
900   <hashTree>
901     <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
902     testname="HTTP Request" enabled="true">
903       <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
904       " guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User
905 Defined Variables" enabled="true">
906         <collectionProp name="Arguments.arguments" />
907       </elementProp>
908       <stringProp name="HTTPSampler.domain">ec2-13-126-213-152.ap-
909 south-1.compute.amazonaws.com</stringProp>
910       <stringProp name="HTTPSampler.port" />

```

```

904      <stringProp name="HTTPSampler.protocol" />
905      <stringProp name="HTTPSampler.contentEncoding" />
906      <stringProp name="HTTPSampler.path">/#</stringProp>
907      <stringProp name="HTTPSampler.method">GET</stringProp>
908      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
909      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
910      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
911      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
912      <stringProp name="HTTPSampler.embedded_url_re" />
913      <stringProp name="HTTPSampler.connect_timeout" />
914      <stringProp name="HTTPSampler.response_timeout" />
915    </HTTPSamplerProxy>
916    <hashTree>
917      <ResultCollector guiclass="TableVisualizer" testclass=""
905 ResultCollector" testname="View Results in Table" enabled="true">
918        <boolProp name="ResultCollector.error_logging">false</
919      boolProp>
920      <objProp>
921        <name>saveConfig</name>
922        <value class="SampleSaveConfiguration">
923          <time>true</time>
924          <latency>true</latency>
925          <timestamp>true</timestamp>
926          <success>true</success>
927          <label>true</label>
928          <code>true</code>
929          <message>true</message>
930          <threadName>true</threadName>
931          <dataType>true</dataType>
932          <encoding>false</encoding>
933          <assertions>true</assertions>
934          <subresults>true</subresults>
935          <responseData>false</responseData>
936          <samplerData>false</samplerData>
937          <xml>false</xml>
938          <fieldNames>true</fieldNames>
939          <responseHeaders>false</responseHeaders>
940          <requestHeaders>false</requestHeaders>
941          <responseDataOnError>false</responseDataOnError>

```

```
941             <saveAssertionResultsFailureMessage>true</
942             saveAssertionResultsFailureMessage>
943                 <assertionsResultsToSave>0</assertionsResultsToSave>
944                 <bytes>true</bytes>
945                 <sentBytes>true</sentBytes>
946                 <url>true</url>
947                 <threadCounts>true</threadCounts>
948                 <sampleCount>true</sampleCount>
949                 <idleTime>true</idleTime>
950                 <connectTime>true</connectTime>
951             </value>
952         </objProp>
953         <stringProp name="filename">D:\Jmeter Logs\logs\ec2\t2.xlarge
954 \ec2_t2.xlarge_2705212200_1.csv</stringProp>
955     </ResultCollector>
956     <hashTree />
957     <ResultCollector guiclass="SummaryReport" testclass=""
958 ResultCollector" testname="Summary Report" enabled="true">
959         <boolProp name="ResultCollector.error_logging">false</
960         boolProp>
961         <objProp>
962             <name>saveConfig </name>
963             <value class="SampleSaveConfiguration">
964                 <time>true</time>
965                 <latency>true</latency>
966                 <timestamp>true</timestamp>
967                 <success>true</success>
968                 <label>true</label>
969                 <code>true</code>
970                 <message>true</message>
971                 <threadName>true</threadName>
972                 <dataType>true</dataType>
973                 <encoding>false</encoding>
974                 <assertions>true</assertions>
975                 <subresults>true</subresults>
976                 <responseData>false</responseData>
977                 <samplerData>false</samplerData>
978                 <xml>false</xml>
979                 <fieldNames>true</fieldNames>
```

```

976          <responseHeaders>false</responseHeaders>
977          <requestHeaders>false</requestHeaders>
978          <responseDataOnError>false</responseDataOnError>
979          <saveAssertionResultsFailureMessage>true</
980          saveAssertionResultsFailureMessage>
981          <assertionsResultsToSave>0</assertionsResultsToSave>
982          <bytes>true</bytes>
983          <sentBytes>true</sentBytes>
984          <url>true</url>
985          <threadCounts>true</threadCounts>
986          <idleTime>true</idleTime>
987          <connectTime>true</connectTime>
988          </value>
989          <stringProp name="filename" />
990        </ResultCollector>
991        <hashTree />
992        <UniformRandomTimer guiclass="UniformRandomTimerGui" testclass=""
993 UniformRandomTimer" testname="Uniform Random Timer" enabled="true">
994          <stringProp name="ConstantTimer.delay">5</stringProp>
995          <stringProp name="RandomTimer.range">100.0</stringProp>
996        </UniformRandomTimer>
997        <hashTree />
998        <kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.
999 vizualizers.PerfMonGui" testclass="kg.apc.jmeter.perfmon.PerfMonCollector"
1000 testname="jp@gc - PerfMon Metrics Collector" enabled="true">
1001          <boolProp name="ResultCollector.error_logging">false</
1002          boolProp>
1003          <objProp>
1004            <name>saveConfig</name>
1005            <value class="SampleSaveConfiguration">
1006              <time>true</time>
1007              <latency>true</latency>
1008              <timestamp>true</timestamp>
1009              <success>true</success>
              <label>true</label>
              <code>true</code>
              <message>true</message>
              <threadName>true</threadName>

```

```

1010          <dataType>true</dataType>
1011          <encoding>false</encoding>
1012          <assertions>true</assertions>
1013          <subresults>true</subresults>
1014          <responseData>false</responseData>
1015          <samplerData>false</samplerData>
1016          <xml>false</xml>
1017          <fieldNames>true</fieldNames>
1018          <responseHeaders>false</responseHeaders>
1019          <requestHeaders>false</requestHeaders>
1020          <responseDataOnError>false</responseDataOnError>
1021          <saveAssertionResultsFailureMessage>true</
720      saveAssertionResultsFailureMessage>
1022          <assertionsResultsToSave>0</assertionsResultsToSave>
1023          <bytes>true</bytes>
1024          <sentBytes>true</sentBytes>
1025          <url>true</url>
1026          <threadCounts>true</threadCounts>
1027          <idleTime>true</idleTime>
1028          <connectTime>true</connectTime>
1029          </value>
1030      </objProp>
1031      <stringProp name="filename" />
1032      <longProp name="interval_grouping">1000</longProp>
1033      <boolProp name="graph_aggregated">false</boolProp>
1034      <stringProp name="include_sample_labels" />
1035      <stringProp name="exclude_sample_labels" />
1036      <stringProp name="start_offset" />
1037      <stringProp name="end_offset" />
1038      <boolProp name="include_checkbox_state">false</boolProp>
1039      <boolProp name="exclude_checkbox_state">false</boolProp>
1040      <collectionProp name="metricConnections">
1041          <collectionProp name="1168561953">
1042              <stringProp name="-201932049">13.126.213.152</
    stringProp>
1043              <stringProp name="1600768">4444</stringProp>
1044              <stringProp name="66952">CPU</stringProp>
1045              <stringProp name="0" />
1046          </collectionProp>

```

```

1047         <collectionProp name="-1132152368">
1048             <stringProp name="-201932049">13.126.213.152</
1049             stringProp>
1050                 <stringProp name="1600768">4444</stringProp>
1051                 <stringProp name="-1993889503">Memory</stringProp>
1052                 <stringProp name="0" />
1053             </collectionProp>
1054         </collectionProp>
1055     <hashTree />
1056 </hashTree>
1057 </hashTree>
1058 </hashTree>
1059 </hashTree>
1060 </jmeterTestPlan>

```

**RDS\_Test.jmx**

```

7065
1 <?xml version="1.0" encoding="UTF-8"?>
2 <jmeterTestPlan version="1.2" properties="5.0" jmeter="5.4.1">
3     <hashTree>
4         <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan
7070 " enabled="true">
5             <stringProp name="TestPlan.comments" />
6             <boolProp name="TestPlan.functional_mode">false</boolProp>
7             <boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
8             <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
7079             <elementProp name="TestPlan.user_defined_variables" elementType="

Arguments" guiclass="ArgumentsPanel" testclass="Arguments" testname="User
Defined Variables" enabled="true">
9                 <collectionProp name="Arguments.arguments" />
10                <elementProp>
11                    <stringProp name="TestPlan.user_define_classpath" />
12                </elementProp>
13            </TestPlan>
14        <hashTree>
15            <JDBCDataSource guiclass="TestBeanGUI" testclass="JDBCDataSource"
testname=" JDBC Connection Configuration" enabled="true">

```

```

716     <stringProp name="dataSource">Variable1</stringProp>
717     <stringProp name="poolMax">50</stringProp>
718     <stringProp name="timeout">10000</stringProp>
719     <stringProp name="trimInterval">60000</stringProp>
720     <boolProp name="autocommit">true</boolProp>
721     <stringProp name="transactionIsolation">DEFAULT</stringProp>
722     <boolProp name="preinit">false</boolProp>
723     <stringProp name="initQuery" />
724     <boolProp name="keepAlive">true</boolProp>
725     <stringProp name="connectionAge">5000</stringProp>
726     <stringProp name="checkQuery">select 1</stringProp>
727     <stringProp name="dbUrl">jdbc:mysql://test-database.cjamazw1gpow.ap
-south-1.rds.amazonaws.com:3306/RDS_Test</stringProp>
728     <stringProp name="driver">com.mysql.jdbc.Driver</stringProp>
729     <stringProp name="username">admin12345</stringProp>
730     <stringProp name="password">admin12345</stringProp>
731     <stringProp name="connectionProperties" />
732   </JDBCDataSource>
733   <hashTree />
734   <JDBCDataSource guiclass="TestBeanGUI" testclass="JDBCDataSource"
735   testname="JDBC Connection Configuration" enabled="true">
736     <stringProp name="dataSource">Variable2</stringProp>
737     <stringProp name="poolMax">50</stringProp>
738     <stringProp name="timeout">10000</stringProp>
739     <stringProp name="trimInterval">60000</stringProp>
740     <boolProp name="autocommit">true</boolProp>
741     <stringProp name="transactionIsolation">DEFAULT</stringProp>
742     <boolProp name="preinit">false</boolProp>
743     <stringProp name="initQuery" />
744     <boolProp name="keepAlive">true</boolProp>
745     <stringProp name="connectionAge">5000</stringProp>
746     <stringProp name="checkQuery">select 1</stringProp>
747     <stringProp name="dbUrl">jdbc:mysql://test-database-small.
748     cjamazw1gpow.ap-south-1.rds.amazonaws.com:3306/RDS_Test_Small</stringProp>
749     <stringProp name="driver">com.mysql.jdbc.Driver</stringProp>
750     <stringProp name="username">admin12345</stringProp>
751     <stringProp name="password">admin12345</stringProp>
752     <stringProp name="connectionProperties" />
753   </JDBCDataSource>

```

```

52      <hashTree />
53
54      <JDBCDataSource guiclass="TestBeanGUI" testclass="JDBCDataSource"
55      testname="JDBC Connection Configuration" enabled="true">
56          <stringProp name="dataSource">Variable3</stringProp>
57          <stringProp name="poolMax">50</stringProp>
58          <stringProp name="timeout">10000</stringProp>
59          <stringProp name="trimInterval">60000</stringProp>
60          <boolProp name="autocommit">true</boolProp>
61          <stringProp name="transactionIsolation">DEFAULT</stringProp>
62          <boolProp name="preinit">false</boolProp>
63          <stringProp name="initQuery" />
64          <boolProp name="keepAlive">true</boolProp>
65          <stringProp name="connectionAge">5000</stringProp>
66          <stringProp name="checkQuery">select 1</stringProp>
67          <stringProp name="dbUrl">jdbc:mysql://test-database-medium.
68          cjamazwigpow.ap-south-1.rds.amazonaws.com:3306/RDS_Test_Medium</stringProp>
69          <stringProp name="driver">com.mysql.jdbc.Driver</stringProp>
70          <stringProp name="username">admin12345</stringProp>
71          <stringProp name="password">admin12345</stringProp>
72          <stringProp name="connectionProperties" />
73      </JDBCDataSource>
74
75      <hashTree />
76
77      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
78      testname="Thread Group" enabled="true">
79          <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
80
81          <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
82          guiclass="LoopControlPanel" testclass="LoopController"
83          testname="Loop Controller" enabled="true">
84              <boolProp name="LoopController.continue_forever">false</boolProp>
85
86              <intProp name="LoopController.loops">-1</intProp>
87          </elementProp>
88
89          <stringProp name="ThreadGroup.num_threads">5000</stringProp>
90
91          <stringProp name="ThreadGroup.ramp_time">1</stringProp>
92
93          <boolProp name="ThreadGroup.scheduler">true</boolProp>
94
95          <stringProp name="ThreadGroup.duration">600</stringProp>
96
97          <stringProp name="ThreadGroup.delay">0</stringProp>
98
99          <boolProp name="ThreadGroup.same_user_on_next_iteration">true</

```

```

    boolProp>
84      </ThreadGroup>
785      <hashTree>
86          <JDBCSSampler guiclass="TestBeanGUI" testclass="JDBCSSampler"
testname="JDBC Request" enabled="true">
87              <stringProp name="dataSource">Variable1</stringProp>
88              <stringProp name="queryType">Select Statement</stringProp>
789              <stringProp name="query">select * from student</stringProp>
90              <stringProp name="queryArguments" />
91              <stringProp name="queryArgumentsTypes" />
92              <stringProp name="variableNames" />
93              <stringProp name="resultVariable">Result1</stringProp>
794              <stringProp name="queryTimeout" />
95              <stringProp name="resultSetMaxRows" />
96              <stringProp name="resultSetHandler">Store as String</stringProp>
97          </JDBCSSampler>
98          <hashTree />
799          <ResultCollector guiclass="TableVisualizer" testclass="
ResultCollector" testname="View Results in Table" enabled="true">
100              <boolProp name="ResultCollector.error_logging">false</boolProp>
101              <objProp>
102                  <name>saveConfig</name>
103                  <value class="SampleSaveConfiguration">
104                      <time>true</time>
105                      <latency>true</latency>
106                      <timestampl>true</timestampl>
107                      <success>true</success>
108                      <label>true</label>
109                      <code>true</code>
110                      <message>true</message>
111                      <threadName>true</threadName>
112                      <dataType>true</dataType>
113                      <encoding>false</encoding>
114                      <assertions>true</assertions>
115                      <subresults>true</subresults>
116                      <responseData>false</responseData>
117                      <samplerData>false</samplerData>
118                      <xml>false</xml>
119                      <fieldNames>true</fieldNames>

```

```
120      <responseHeaders>false</responseHeaders>
121      <requestHeaders>false</requestHeaders>
122      <responseDataOnError>false</responseDataOnError>
123      <saveAssertionResultsFailureMessage>true</
124          saveAssertionResultsFailureMessage>
125              <assertionsResultsToSave>0</assertionsResultsToSave>
126              <bytes>true</bytes>
127              <sentBytes>true</sentBytes>
128              <url>true</url>
129              <threadCounts>true</threadCounts>
130              <idleTime>true</idleTime>
131              <connectTime>true</connectTime>
132          </value>
133      </objProp>
134      <stringProp name="filename">D:\Jmeter Logs\logs\rds\db.t2.micro\
135 rds_db.t2.micro_2705212330_1.csv</stringProp>
136      </ResultCollector>
137      <hashTree />
138      <ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report" enabled="true">
139          <boolProp name="ResultCollector.error_logging">false</boolProp>
140          <objProp>
141              <name>saveConfig</name>
142              <value class="SampleSaveConfiguration">
143                  <time>true</time>
144                  <latency>true</latency>
145                  <timestamp>true</timestamp>
146                  <success>true</success>
147                  <label>true</label>
148                  <code>true</code>
149                  <message>true</message>
150                  <threadName>true</threadName>
151                  <dataType>true</dataType>
152                  <encoding>false</encoding>
153                  <assertions>true</assertions>
154                  <subresults>true</subresults>
155                  <responseData>false</responseData>
156                  <samplerData>false</samplerData>
157                  <xml>false</xml>
```

```
156      <fieldNames>true</fieldNames>
157      <responseHeaders>false</responseHeaders>
158      <requestHeaders>false</requestHeaders>
159      <responseDataOnError>false</responseDataOnError>
160      <saveAssertionResultsFailureMessage>true</
161      saveAssertionResultsFailureMessage>
162      <assertionsResultsToSave>0</assertionsResultsToSave>
163      <bytes>true</bytes>
164      <sentBytes>true</sentBytes>
165      <url>true</url>
166      <threadCounts>true</threadCounts>
167      <idleTime>true</idleTime>
168      <connectTime>true</connectTime>
169      </value>
170      <stringProp name="filename" />
171      </ResultCollector>
172      <hashTree />
173      </hashTree>
174      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
175      testname="Thread Group" enabled="true">
176          <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
177          <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
178          guiclass="LoopControlPanel" testclass="LoopController"
179          testname="Loop Controller" enabled="true">
180              <boolProp name="LoopController.continue_forever">false</boolProp>
181              <intProp name="LoopController.loops">-1</intProp>
182              <elementProp>
183                  <stringProp name="ThreadGroup.num_threads">5000</stringProp>
184                  <stringProp name="ThreadGroup.ramp_time">1</stringProp>
185                  <boolProp name="ThreadGroup.scheduler">true</boolProp>
186                  <stringProp name="ThreadGroup.duration">600</stringProp>
187                  <stringProp name="ThreadGroup.delay">0</stringProp>
188                  <boolProp name="ThreadGroup.same_user_on_next_iteration">true</
189                  boolProp>
190          </ThreadGroup>
191          <hashTree>
```

```
188      <JDBCSSampler guiclass="TestBeanGUI" testclass="JDBCSSampler"
189      testname=" JDBC Request" enabled="true">
190          <stringProp name="dataSource">Variable2</stringProp>
191          <stringProp name="queryType">Select Statement</stringProp>
192          <stringProp name="query">select * from student</stringProp>
193          <stringProp name="queryArguments" />
194          <stringProp name="queryArgumentsTypes" />
195          <stringProp name="resultVariable">Result2</stringProp>
196          <stringProp name="queryTimeout" />
197          <stringProp name="resultSetMaxRows" />
198          <stringProp name="resultSetHandler">Store as String</stringProp>
199      </JDBCSSampler>
200      <hashTree />
201      <ResultCollector guiclass="TableVisualizer" testclass="
202 ResultCollector" testname="View Results in Table" enabled="true">
203          <boolProp name="ResultCollector.error_logging">false</boolProp>
204          <objProp>
205              <name>saveConfig</name>
206              <value class="SampleSaveConfiguration">
207                  <time>true</time>
208                  <latency>true</latency>
209                  <timestamp>true</timestamp>
210                  <success>true</success>
211                  <label>true</label>
212                  <code>true</code>
213                  <message>true</message>
214                  <threadName>true</threadName>
215                  <dataType>true</dataType>
216                  <encoding>false</encoding>
217                  <assertions>true</assertions>
218                  <subresults>true</subresults>
219                  <responseData>false</responseData>
220                  <samplerData>false</samplerData>
221                  <xml>false</xml>
222                  <fieldNames>true</fieldNames>
223                  <responseHeaders>false</responseHeaders>
224                  <requestHeaders>false</requestHeaders>
225                  <responseDataOnError>false</responseDataOnError>
```

```
225          <saveAssertionResultsFailureMessage>true</
7320      saveAssertionResultsFailureMessage>
226          <assertionsResultsToSave>0</assertionsResultsToSave>
227          <bytes>true</bytes>
228          <sentBytes>true</sentBytes>
229          <url>true</url>
230          <threadCounts>true</threadCounts>
231          <idleTime>true</idleTime>
232          <connectTime>true</connectTime>
233      </value>
234  </objProp>
235      <stringProp name="filename">D:\Jmeter Logs\logs\rds\db.t2.small\
rds_db.t2.small_2705212330_1.csv</stringProp>
236  </ResultCollector>
237  <hashTree />
238  <ResultCollector guiclass="SummaryReport" testclass="
7335 ResultCollector" testname="Summary Report" enabled="true">
239      <boolProp name="ResultCollector.error_logging">false</boolProp>
240  <objProp>
241      <name>saveConfig</name>
242      <value class="SampleSaveConfiguration">
243          <time>true</time>
244          <latency>true</latency>
245          <timestampl>true</timestampl>
246          <success>true</success>
247          <label>true</label>
248          <code>true</code>
249          <message>true</message>
250          <threadName>true</threadName>
251          <dataType>true</dataType>
252          <encoding>false</encoding>
253          <assertions>true</assertions>
254          <subresults>true</subresults>
255          <responseData>false</responseData>
256          <samplerData>false</samplerData>
257          <xml>false</xml>
258          <fieldNames>true</fieldNames>
259          <responseHeaders>false</responseHeaders>
260          <requestHeaders>false</requestHeaders>
```

```
261          <responseDataOnError>false</responseDataOnError>
262          <saveAssertionResultsFailureMessage>true</
7360      saveAssertionResultsFailureMessage>
263          <assertionsResultsToSave>0</assertionsResultsToSave>
264          <bytes>true</bytes>
265          <sentBytes>true</sentBytes>
266          <url>true</url>
267          <threadCounts>true</threadCounts>
268          <idleTime>true</idleTime>
269          <connectTime>true</connectTime>
270      </value>
271      </objProp>
272      <stringProp name="filename" />
273  </ResultCollector>
274  <hashTree />
275 </hashTree>
276 <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
7375 testname="Thread Group" enabled="true">
277      <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
278      <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
7380 guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
279          <boolProp name="LoopController.continue_forever">false</boolProp>
280          <intProp name="LoopController.loops">-1</intProp>
281      </elementProp>
282      <stringProp name="ThreadGroup.num_threads">5000</stringProp>
283      <stringProp name="ThreadGroup.ramp_time">1</stringProp>
284      <boolProp name="ThreadGroup.scheduler">true</boolProp>
285      <stringProp name="ThreadGroup.duration">600</stringProp>
286      <stringProp name="ThreadGroup.delay">0</stringProp>
287      <boolProp name="ThreadGroup.same_user_on_next_iteration">true</
288      boolProp>
289  </ThreadGroup>
290  <hashTree>
291      <JDBCSSampler guiclass="TestBeanGUI" testclass="JDBCSSampler"
7395 testname="JDBC Request" enabled="true">
292          <stringProp name="dataSource">Variable3</stringProp>
```

```
292      <stringProp name="queryType">Select Statement</stringProp>
293      <stringProp name="query">select * from student</stringProp>
294      <stringProp name="queryArguments" />
295      <stringProp name="queryArgumentsTypes" />
296      <stringProp name="variableNames" />
297      <stringProp name="resultVariable">Result3</stringProp>
298      <stringProp name="queryTimeout" />
299      <stringProp name="resultSetMaxRows" />
300      <stringProp name="resultSetHandler">Store as String</stringProp>
301  </JDBC Sampler>
302  <hashTree />
303  <ResultCollector guiclass="TableVisualizer" testclass=
ResultCollector" testname="View Results in Table" enabled="true">
304      <boolProp name="ResultCollector.error_logging">false</boolProp>
305      <objProp>
306          <name>saveConfig</name>
307          <value class="SampleSaveConfiguration">
308              <time>true</time>
309              <latency>true</latency>
310              <timestamp>true</timestamp>
311              <success>true</success>
312              <label>true</label>
313              <code>true</code>
314              <message>true</message>
315              <threadName>true</threadName>
316              <dataType>true</dataType>
317              <encoding>false</encoding>
318              <assertions>true</assertions>
319              <subresults>true</subresults>
320              <responseData>false</responseData>
321              <samplerData>false</samplerData>
322              <xml>false</xml>
323              <fieldNames>true</fieldNames>
324              <responseHeaders>false</responseHeaders>
325              <requestHeaders>false</requestHeaders>
326              <responseDataOnError>false</responseDataOnError>
327              <saveAssertionResultsFailureMessage>true</
328      saveAssertionResultsFailureMessage>
329              <assertionsResultsToSave>0</assertionsResultsToSave>
```

```
329          <bytes>true</bytes>
330          <sentBytes>true</sentBytes>
331          <url>true</url>
332          <threadCounts>true</threadCounts>
333          <idleTime>true</idleTime>
334          <connectTime>true</connectTime>
335          </value>
336      </objProp>
337      <stringProp name="filename">D:\Jmeter Logs\logs\rds\db.t2.medium
7445 \rds_db.t2.medium_2705212330_1.csv</stringProp>
338      </ResultCollector>
339      <hashTree />
340      <ResultCollector guiclass="SummaryReport" testclass="
ResultCollector" testname="Summary Report" enabled="true">
341          <boolProp name="ResultCollector.error_logging">false</boolProp>
342          <objProp>
343              <name>saveConfig</name>
344              <value class="SampleSaveConfiguration">
345                  <time>true</time>
346                  <latency>true</latency>
347                  <timestamp>true</timestamp>
348                  <success>true</success>
349                  <label>true</label>
350                  <code>true</code>
351                  <message>true</message>
352                  <threadName>true</threadName>
353                  <dataType>true</dataType>
354                  <encoding>false</encoding>
355                  <assertions>true</assertions>
356                  <subresults>true</subresults>
357                  <responseData>false</responseData>
358                  <samplerData>false</samplerData>
359                  <xml>false</xml>
360                  <fieldNames>true</fieldNames>
361                  <responseHeaders>false</responseHeaders>
362                  <requestHeaders>false</requestHeaders>
363                  <responseDataOnError>false</responseDataOnError>
364                  <saveAssertionResultsFailureMessage>true</
saveAssertionResultsFailureMessage>
```

```

365           <assertionsResultsToSave>0</assertionsResultsToSave>
366           <bytes>true</bytes>
367           <sentBytes>true</sentBytes>
368           <url>true</url>
369           <threadCounts>true</threadCounts>
370           <idleTime>true</idleTime>
371           <connectTime>true</connectTime>
372       </value>
373     </objProp>
374     <stringProp name="filename" />
375   </ResultCollector>
376   <hashTree />
377 </hashTree>
378 </hashTree>
379 </hashTree>
380 </jmeterTestPlan>

```

**S3\_Test.jmx**

```

7491 <?xml version="1.0" encoding="UTF-8"?>
2 <jmeterTestPlan version="1.2" properties="5.0" jmeter="5.4.1">
3   <hashTree>
4     <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan"
" enabled="true">
5       <stringProp name="TestPlan.comments" />
6       <boolProp name="TestPlan.functional_mode">false</boolProp>
7       <boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
8       <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
9       <elementProp name="TestPlan.user_defined_variables" elementType="
7505 Arguments" guiclass="ArgumentsPanel" testclass="Arguments" testname="User
Defined Variables" enabled="true">
10      <collectionProp name="Arguments.arguments" />
11    </elementProp>
12    <stringProp name="TestPlan.user_define_classpath" />
7518  </TestPlan>
14  <hashTree>
15    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" /

```

```

    testname="Thread Group" enabled="true">
        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
7515    >
        <elementProp name="ThreadGroup.main_controller" elementType="LoopController" guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
            <boolProp name="LoopController.continue_forever">false</boolProp>
7520    >
            <intProp name="LoopController.loops">-1</intProp>
20        </elementProp>
21        <stringProp name="ThreadGroup.num_threads">10</stringProp>
22        <stringProp name="ThreadGroup.ramp_time">1</stringProp>
723        <boolProp name="ThreadGroup.scheduler">true</boolProp>
24        <stringProp name="ThreadGroup.duration">1800</stringProp>
25        <stringProp name="ThreadGroup.delay">0</stringProp>
26        <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
727    </ThreadGroup>
28    <hashTree>
29        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="HTTP Request" enabled="true">
30            <elementProp name="HTTPSampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
31                <collectionProp name="Arguments.arguments" />
32            </elementProp>
33            <stringProp name="HTTPSampler.domain">boilerplatecodebucket.s3.ap-south-1.amazonaws.com</stringProp>
7540            <stringProp name="HTTPSampler.port" />
34            <stringProp name="HTTPSampler.protocol" />
35            <stringProp name="HTTPSampler.contentEncoding" />
36            <stringProp name="HTTPSampler.path">/index.html</stringProp>
37            <stringProp name="HTTPSampler.method">GET</stringProp>
7545            <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
40            <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
41            <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
42            <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
7550            <stringProp name="HTTPSampler.embedded_url_re" />
44            <stringProp name="HTTPSampler.connect_timeout" />

```

```
45      <stringProp name="HTTPSampler.response_timeout" />
46    </HTTPSamplerProxy>
47
48    <hashTree>
49      <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table" enabled="true">
50        <boolProp name="ResultCollector.error_logging">false</boolProp>
51        <objProp>
52          <name>saveConfig</name>
53          <value class="SampleSaveConfiguration">
54            <time>true</time>
55            <latency>true</latency>
56            <timestampl>true</timestampl>
57            <success>true</success>
58            <label>true</label>
59            <code>true</code>
60            <message>true</message>
61            <threadName>true</threadName>
62            <dataType>true</dataType>
63            <encoding>false</encoding>
64            <assertions>true</assertions>
65            <subresults>true</subresults>
66            <responseData>false</responseData>
67            <samplerData>false</samplerData>
68            <xml>false</xml>
69            <fieldNames>true</fieldNames>
70            <responseHeaders>false</responseHeaders>
71            <requestHeaders>false</requestHeaders>
72            <responseDataOnError>false</responseDataOnError>
73            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
74              <assertionsResultsToSave>0</assertionsResultsToSave>
75              <bytes>true</bytes>
76              <sentBytes>true</sentBytes>
77              <url>true</url>
78              <threadCounts>true</threadCounts>
79              <idleTime>true</idleTime>
80              <connectTime>true</connectTime>
81            </value>
```

```
81          </objProp>
82          <stringProp name="filename">D:\Jmeter Logs\logs\s3\
83          s3_2705211845_1.csv</stringProp>
84      </ResultCollector>
85      <hashTree />
86      <ResultCollector guiclass="SummaryReport" testclass=""
87      ResultCollector" testname="Summary Report" enabled="true">
88          <boolProp name="ResultCollector.error_logging">false</
89          boolProp>
90          <objProp>
91              <name>saveConfig</name>
92              <value class="SampleSaveConfiguration">
93                  <time>true</time>
94                  <latency>true</latency>
95                  <timestampl>true</timestampl>
96                  <success>true</success>
97                  <label>true</label>
98                  <code>true</code>
99                  <message>true</message>
100                 <threadName>true</threadName>
101                 <dataType>true</dataType>
102                 <encoding>false</encoding>
103                 <assertions>true</assertions>
104                 <subresults>true</subresults>
105                 <responseData>false</responseData>
106                 <samplerData>false</samplerData>
107                 <xml>false</xml>
108                 <fieldNames>true</fieldNames>
109                 <responseHeaders>false</responseHeaders>
110                 <requestHeaders>false</requestHeaders>
111                 <responseDataOnError>false</responseDataOnError>
112                 <saveAssertionResultsFailureMessage>true</
113                 saveAssertionResultsFailureMessage>
114                     <assertionsResultsToSave>0</assertionsResultsToSave>
115                     <bytes>true</bytes>
116                     <sentBytes>true</sentBytes>
117                     <url>true</url>
118                     <threadCounts>true</threadCounts>
119                     <idleTime>true</idleTime>
```

```

146          <connectTime>true</connectTime>
117      </value>
118  </objProp>
119  <stringProp name="filename" />
120 </ResultCollector>
121 <hashTree />
122 </hashTree>
123 </hashTree>
124 </hashTree>
125 </hashTree>
126 </jmeterTestPlan>

```

**endpoints.py**

```

7641 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3
4 import csv
5 import pandas as pd
7656 import json
7 import os
8 import requests
9 import datetime
10 import time
7661 fileName = 'data/Log_11052021.csv'
12 path = './../bigLogsDir/'
13 endpointID = 2
14 separator = '/'
15 url = 'http://cloud-service-optimizer-dev.herokuapp.com/'
766
17
18 def attemptLogin(eID, pwd):
19     headersPost = {'Content-type': 'application/json'}
20     timestamp = datetime.datetime.now()
7671 postData = {'endpointID': str(eID), 'password': str(pwd),
22         'timestamp': str(timestamp)}
23 postData = json.dumps(postData)

```

```
24     responseFile = requests.post(url + 'endpoints/endpoint-login/',
25                                     data=postData, headers=headersPost)
26
27     responseFile = json.loads(responseFile.text)
28
29     if responseFile['status'] == 'success':
30         return (1, responseFile['message'])
31
32
33 def attemptSignup(name, password):
34
35     headersPost = {'Content-type': 'application/json'}
36
37     postData = {'name': str(name), 'password': str(password)}
38
39     postData = json.dumps(postData)
40
41     responseFile = requests.post(url + 'endpoints/create-endpoint/',
42                                   data=postData, headers=headersPost)
43
44     responseFile = json.loads(responseFile.text)
45
46     if responseFile['status'] == 'success':
47         return (1, responseFile['message'], responseFile['name'])
48
49     else:
50
51         return (0, responseFile['message'], responseFile['name'])
52
53
54 loginFailure = 1
55
56 while loginFailure:
57
58     inputFailure = 1
59
60     while inputFailure:
61
62         loginOrSignup = \
63             input('Press 1 for logging in. Press 0 for creating/registering an
64               endpoint:\n')
65
66         if not loginOrSignup == '1' and not loginOrSignup == '0':
67             print 'Wrong input. Please try again.'
68
69         else:
70
71             inputFailure = 0
72
73     if loginOrSignup == '1':
74
75         eID = input('Enter endpoint ID:\n')
76
77         pwd = input('Enter password:\n')
78
79         print 'Checking login credentials...'
```

```
62     loginResult = attemptLogin(eID, pwd)
63
64     if loginResult[0]:
65         print 'Successfully logged in to ' + str(loginResult[1]) \
66             + '!\\n'
67
68         pathFailure = 1
69
70         while pathFailure:
71             separatorCheckFailure = 1
72
73             while separatorCheckFailure:
74                 separatorCheck = \
75                     input('If Windows, press 1. If non-Windows, press 0. ')
76
77                 if separatorCheck == '0' or separatorCheck == '1':
78                     separatorCheckFailure = 0
79
80                 else:
81                     print 'Invalid entry. Please try again.'
82
83                 if separatorCheck == '1':
84                     separator = '\\\\' + '\\\\'
85
86                 else:
87                     separator = '/'
88
89                 enteredPath = \
90                     input('Enter the path where the logs are stored:')
91
92                 separatorValid = '/'
93
94                 if separatorCheck == '1':
95                     separatorValid = '\\\\'
96
97                 if not enteredPath[-1] == separatorValid:
98                     enteredPath = enteredPath + separator
99
100                if os.path.isdir(enteredPath):
101
102                    pathFailure = 0
103
104                else:
105
106                    print 'The entered directory does not exist. Please try
107 again.'
108
109                loginFailure = 0
110
111            else:
112
113                print 'Invalid credentials. Try again.'
114
115        else:
116
117            name = input('Please enter your name:\\n')
118
119            passwordsDoNotMatch = 1
120
121            while passwordsDoNotMatch:
122
123                password = input('Please create your password:\\n')
```

```
100     confirmPassword = input('Please confirm your password:\n')
101     if password == confirmPassword:
102         passwordsDoNotMatch = 0
103     else:
104         print 'Passwords do not match. Try again.'
105     signupResult = attemptSignup(name, password)
106     if signupResult[0]:
107         print 'Successfully signed up user ' + str(signupResult[2])
108         print 'Your endpoint ID to log in is: ' \
109             + str(signupResult[1])
110     print 'Login with created credentials:'
111     eID = input('Enter endpoint ID:\n')
112     pwd = input('Enter password:\n')
113     print 'Checking login credentials...'
114     loginResult = attemptLogin(eID, pwd)
115     if loginResult[0]:
116         print 'Successfully logged in to ' + str(loginResult[1]) \
117             + '!\n'
118         pathFailure = 1
119         while pathFailure:
120             separatorCheckFailure = 1
121             while separatorCheckFailure:
122                 separatorCheck = \
123                     input('If Windows, press 1. If non-Windows, press 0.')
124                     )
125                     if separatorCheck == '0' or separatorCheck == '1':
126                         separatorCheckFailure = 0
127                     else:
128                         print 'Invalid entry. Please try again.'
129                     if separatorCheck == '1':
130                         separator = '\\\\' + '\\\\'
131                     else:
132                         separator = '/'
133                     enteredPath = \
134                         input('Enter the path where the logs are stored:')
135                     separatorValid = '/'
136                     if separatorCheck == '1':
137                         separatorValid = '\\\\'
138                     if not enteredPath[-1] == separatorValid:
```

```
139         enteredPath = enteredPath + separator
140
141         if os.path.isdir(enteredPath):
142             pathFailure = 0
143
144         else:
145
146             print 'The entered directory does not exist. Please try
7790 again.'
147             loginFailure = 0
148
149         else:
150
151     def getFilesInServer(endpointID):
152
153         headersPost = {'Content-type': 'application/json'}
154
155         postData = {'endpointID': str(endpointID)}
156
157         postData = json.dumps(postData)
158
159         responseFile = requests.post(url + 'endpoints/get-files/',
160                                         data=postData, headers=headersPost)
161
162         responseFile = json.loads(responseFile.text)
163
164         if responseFile['status'] == 'success':
165
166             strFiles = str(responseFile['message'])
167
168             strFiles = strFiles[3:]
169
170             strFiles = strFiles[:-3]
171
172             strFiles = strFiles[::]
173
174             return strFiles
175
176         else:
177
178             return ''
```

```
177 filesInServer = getFilesInServer(endpointID)
178 sessionID = initiateLogging(endpointID)
179 print 'Your login session ID is:' + str(sessionID)
180
181
182 def uploadLogFile(
183     isPerformanceIncluded,
184     performanceJSON,
185     logsJSON,
186     summaryJSON,
187     sessionID,
188     endpointID,
189     serviceType,
190     instanceType,
191 ):
192
193     headersPost = {'Content-type': 'application/json'}
194     timestamp = datetime.datetime.now()
195     postData = {}
196     if isPerformanceIncluded:
197         postData = {
198             'sessionID': sessionID,
199             'endpointID': endpointID,
200             'serviceType': serviceType,
201             'instanceType': instanceType,
202             'elapsed': str(logsJSON[0]),
203             'latency': str(logsJSON[1]),
204             'connect': str(logsJSON[2]),
205             'stdDev': str(summaryJSON[0]),
206             'errorRate': str(summaryJSON[1]),
207             'throughput': str(summaryJSON[2]),
208             'summarySize': str(summaryJSON[3]),
209             'timestamp': str(timestamp),
210             'isPerformanceIncluded': '1',
211             'cpu': str(performanceJSON[0]),
212             'memory': str(performanceJSON[1]),
213             'performanceSize': str(performanceJSON[2]),
214         }
215     else:
```

```
216     postData = {  
217         'sessionID': sessionID,  
218         'endpointID': endpointID,  
219         'serviceType': serviceType,  
220         'instanceType': instanceType,  
221         'elapsed': str(logsJSON[0]),  
222         'latency': str(logsJSON[1]),  
223         'connect': str(logsJSON[2]),  
224         'stdDev': str(summaryJSON[0]),  
225         'errorRate': str(summaryJSON[1]),  
226         'throughput': str(summaryJSON[2]),  
227         'summarySize': str(summaryJSON[3]),  
228         'timestamp': str(timestamp),  
229         'isPerformanceIncluded': '0',  
230         'cpu': '',  
231         'memory': '',  
232         'performanceSize': '',  
233     }  
234     postData = json.dumps(postData)  
235     start_time = time.time()  
236     response = requests.post(url + 'endpoints/post-log-data/',  
237                             data=postData, headers=headersPost)  
238     end_time = time.time()  
239     print 'Upload done in:'  
240     print '--- %s seconds ---' % (end_time - start_time)  
241  
242  
243     def getElapsed(df):  
244         return df.mean()  
245  
246  
247     def getLatency(df):  
248         return df.mean()  
249  
250  
251     def getConnect(df):  
252         return df.mean()  
253  
254
```

```
255 def readLogFile(filePath):
256     df = pd.read_csv(filePath, usecols=[
257         'elapsed',
258         'responseCode',
259         'responseMessage',
260         'success',
261         'Latency',
262         'Connect',
263     ])
264     df = df[1:]
265     elapsed = getElapsed(df['elapsed'])
266     latency = getLatency(df['Latency'])
267     connect = getConnect(df['Connect'])
268     return (elapsed, latency, connect)
269     json_str = df.to_json()
270
271
272
273 def readSummaryFile(filePath):
274     df = pd.read_csv(filePath, usecols=[
275         'Label',
276         '# Samples',
277         'Average',
278         'Min',
279         'Max',
280         'Std. Dev.',
281         'Error %',
282         'Throughput',
283         'Received KB/sec',
284         'Sent KB/sec',
285         'Avg. Bytes',
286     ])
287     df = df.iloc[0]
288     sizeOfFile = df['# Samples']
289     stdDev = df['Std. Dev.']
290     errorRate = df['Error %']
291     throughput = df['Throughput']
292     return (stdDev, errorRate, throughput, sizeOfFile)
293     json_str = df.to_json()
```

```
294     return json_str
295
296
297 def readPerformanceFile(filePath):
298     df = pd.read_csv(filePath, usecols=['CPU', 'Memory'])
299     df = df[1:]
300     sizeOfFile = len(df.index)
301     cpu = df['CPU'].mean()
302     memory = df['Memory'].mean()
303     return (cpu, memory, sizeOfFile)
304
305
306 def updateFilesListInServer(filesList, endpointID, sessionID):
307     headersPost = {'Content-type': 'application/json'}
308     postData = {'sessionID': sessionID, 'endpointID': endpointID,
309                 'filesString': filesList}
310     postData = json.dumps(postData)
311     response = requests.post(url + 'endpoints/update-files/',
312                               data=postData, headers=headersPost)
313
314
315 filesList = filesInServer
316
317 flag = 0
318 print 'Going through files...'
319 for serviceType in os.listdir(path):
320     if os.path.isdir(path + serviceType) and (serviceType == 'ec2'
321         or serviceType == 'rds'):
322         subpath = path + serviceType
323         for instanceType in os.listdir(subpath):
324             if os.path.isdir(subpath + separator + instanceType):
325                 suppPath = subpath + separator + instanceType
326                 for filename in os.listdir(suppPath):
327                     if os.path.isfile(suppPath + separator + filename) \
328                         and filename.endswith('_l.csv'):
329                         filePath = suppPath + separator + filename
330                         summaryFile = filename[:-5] + 's.csv'
331                         performanceFile = filename[:-5] + 'p.csv'
332                         summaryPath = suppPath + separator + summaryFile
```

```
333         performancePath = suppath + separator \
334             + performanceFile
335         if filesList.find(filePath) == -1:
336             print 'Currently uploading file:'
337             print filePath
338             logsJSON = readLogFile(filePath)
339             summaryJSON = readSummaryFile(summaryPath)
340             if os.path.exists(performancePath):
341                 performanceJSON = \
342                     readPerformanceFile(performancePath)
343             uploadLogFile(
344                 1,
345                 performanceJSON,
346                 logsJSON,
347                 summaryJSON,
348                 sessionID,
349                 endpointID,
350                 serviceType,
351                 instanceType,
352             )
353         else:
354             uploadLogFile(
355                 0,
356                 '',
357                 logsJSON,
358                 summaryJSON,
359                 sessionID,
360                 endpointID,
361                 serviceType,
362                 instanceType,
363             )
364             print 'File uploaded successfully!'
365             filesList = filesList + filePath + ', '
366             flag = 1
367     elif os.path.isdir(path + serviceType) and serviceType == 's3':
368         subpath = path + serviceType
369         for filename in os.listdir(subpath):
370             if os.path.isfile(subpath + separator + filename) \
371                 and filename.endswith('_l.csv'):
```

```
372         filePath = subpath + separator + filename
373         summaryFile = filename[:-5] + 's.csv'
374         summaryPath = subpath + separator + summaryFile
375         if filesList.find(filePath) == -1:
376             print 'Currently uploading s3 file:'
377             print filePath
378             logsJSON = readLogFile(filePath)
379             summaryJSON = readSummaryFile(summaryPath)
380             uploadLogFile(
381                 0,
382                 '',
383                 logsJSON,
384                 summaryJSON,
385                 sessionID,
386                 endpointID,
387                 serviceType,
388                 'na',
389             )
390             print 'File uploaded successfully!'
391             filesList = filesList + filePath + ', '
392             flag = 1
393             print 'Read the s3 file'
394 if flag:
395     filesList = filesList[:-2]
396     print 'Uploaded files list:' + filesList
397 else:
398     print 'No new files to be uploaded.'
399 updateFilesListInServer(filesList, endpointID, sessionID)
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
```

# Report-Poornima B

*by Poornima B*

---

**Submission date:** 05-Jun-2021 09:29AM (UTC+0530)

**Submission ID:** 1600752742

**File name:** S6ProjectReport.docx (3.37M)

**Word count:** 28676

**Character count:** 248258

## Introduction

81

### 1.1 Cloud Service Selection

Cloud Service Selection has progressed as a cloud computing paradigm of entrusting good faith in a Cloud Service Provider (CSP) for the services with specifications that are legally agreed upon by the involved parties. The abundance of seemingly similar Cloud Services and the enormous range of customized user preferences and requirements have led to difficulty in choosing the optimal CSP. Some of the main issues faced by CC are lack of trust in service providers with respect to availability, reliability and efficiency of services at their time of need, ambiguity in SLAs, non-compliance with SLA, absence of diversified trust elements, and Quality-of-Service guarantee. Although a number of cloud service e-marketplaces are present, users find it very difficult to manually compare the services of different CSPs: especially new users who must go through each feature's description separately during Cloud Service Provider selection. This is where the model comes into the picture to make the job quick, easy and efficient.

#### 1.1.1 How to choose a cloud service

Currently, in addition to the absence of a common framework that can be used to judge the trustworthiness quotient of a particular CSP, and the various dissimilarities present between two CSPs which render them as completely different from each other, there is no solid way to judge the aforementioned trustworthiness metric of any CSP. With the current industrial standards that are set, for example, audit scores, reviews, and SLAs, the trustworthiness metric can be easily tampered with to favour the CSP and not provide accurate results to the end-user. A few pointers to be kept in mind while choosing a best-suited CSP are listed below:

- **The Current Competitors:** Even though the cloud industry has been experiencing an exponential increase with respect to its usage, there are several competitors that already exist in the market. It may include giants present in the cloud game for a very long time, or even a selected set of newer, smaller-scale competitors.
- **Cloud Security:** The user needs to analyse and predict the security measures that are applicable to their requirements and map them to the security measures provided by each of the CSPs. In addition to this, the user needs to make themselves aware of the mechanisms through which their data is preserved, and the services which are provided freely as well as the services which come under the category of pay-as-you-go category.
- **Cloud Compliance:** The user needs to ensure that the services provided by the CSP meet the industry standards and comply with the listed terms and conditions. The user needs to be well aware of their responsibilities and the features of the CSP which can be used to check off their compliance standards.
- **Architecture:** While choosing a CSP, another thing to keep in mind the way in which the cloud infrastructure will be incorporated into the working of the user's/ organization's workflows, with respect to both current and future scenarios. The different cloud storage architectures also need to be kept in mind, i.e., the type of multilevel storages, archival storages provided, etc.
- **Manageability:** The aspects which fall under the responsibility of the user to manage while integrating the services with the respective CSPs should also be taken care of. The time and effort estimates which the user has to spend can be a critical deciding factor.
- **Service Levels:** Service Level Agreements (SLAs) hold a large amount of weightage when it comes to choosing a CSP. It is essential when organisations / businesses / individuals have strict requirements in terms of the facilities they are offered, for example, availability, reliability, scalability, security, etc. The SLAs act as an agreement between the CSP and customer to ensure that the threshold for the quality-of-service is met. In case there is a scenario that is played out where the end-user / customer does not receive up to the mark service, the CSP has to take measures to make it up to the customer as well.

- **Support:** Another critical aspect to be taken into consideration is the amount of support that the particular CSP will provide when required. On one hand, the mode of support provided can be in the form of chat support which might not be acceptable by certain users, whereas in case of the different CSP, there might be a dedicated resource available solely for support which might fit well with many users.
- **Costs:** The amount that goes into migrating to a particular cloud service might not be the most important factor that needs to be considered while choosing a CSP, but it does have significant weightage. There are various cost plans available that range from pay-as-you-go basis, reserved methods, up to volume discounts as the service usage increases. The user can assess which plan suits their requirement best.

### 1.1.2 Cloud Services that are in use today

There are several Cloud Service Providers in the market and have transformed into the primary requirement when users want to use the facilities provided by them to host their applications on the internet. As each day passes, there are more budding CSPs coming into existence, but there are a few giants that have made their mark for quite some time in the cloud market. Here are a few of the dominating and established CSPs in the market currently.

- **Amazon Web Services (AWS):** AWS has been one of the longest standing competitors in the field of cloud computing and specializes in database, serverless deployments and artificial intelligence. It provides an array of services such as Elastic Compute Cloud (EC2), Amazon Simple Storage Service (S3), Amazon Relational Database Service (RDS), etc.
- **Microsoft Azure:** Microsoft's Azure cloud along with Microsoft's provisions of software-as-a-service makes this particular CSP rank itself in the second position in the list of most preferred cloud service providers. It focuses on scaling, artificial intelligence, and providing cheap (if not free) services as a public cloud infrastructure.
- **Google Cloud Platform (GCP):** This particular CSP is ranked third on the aforementioned list. GCP, although having faced hiccups in the past, is now focusing on scaling out, building partnerships with established enterprises such as SAP, VMWare, etc., and combining its G Suite and Google Cloud sales efforts.
- **Hewlett Packard Enterprise:** The HP cloud services offered are hybrid in nature and focuses on connecting cloud with edge computing. The strategy revolving around this particular cloud involves its hardware stack, the various edge compute devices (via Aruba), storage and networking facilities, and multiple software platforms like Greenlake, Synergy, etc. This particular CSP prefers to call itself a hybrid setup rather than a multi-cloud environment. HPE has established partnerships with enterprises such as Red Hat, VMWare, and integrated and converged systems with cloud providers.
- **Cisco Systems:** This particular CSP is a structured hybrid cloud through a network and API prism. A network-centric approach has been taken while implementing the services offered by the cloud. The headliner with respect to this cloud is ACI, short for an architecture known as Application Centric Infrastructure. This architecture along with the network-centric policy focuses on management of the services, policies related to the services offered, and operations which are reserved for applications that are deployed across different / multiple cloud environments.

### 1.1.3 SLA Compliance and Industry Standards

#### Service Level Agreement (SLA)

Most CC and CSP use a written document as a basis for all services to be provided as well as the quality that is expected from those services. Detailed and formal SLAs are very important to establish between both parties since companies' and individuals' infrastructure can have serious impacts if something goes wrong with the services that they or their customers are using. These impacts can include Security breaches, unavailability of services,

Personally Identifiable Information (PII) theft, etc. SLAs between CC's IT teams and CSP differ depending on factors such as type of services and CC's business needs. Although, most SLAs cover basic attributes such as Speed, Availability, Security, Service Uptime, etc. SLAs also come handy in monitoring cloud governance and compliance. The Service Level Objectives (SLO) are decided based on Key Performance Indicators (KPI) provided by the customers after negotiation with the CSP. There are integrated monitoring services with Cloud Services which can monitor SLA compliance by the processes and alert the users if SLA is violated. CSP usually have generic pre-made SLA but the IT department of the client companies should review the SLA along with their legal counsel before deciding to go ahead with the services. Quality of Services (QoS) must be assessed carefully and decided upon in the SLA.

SLA has precise quantifiable values assigned to various attributes to show their performance metrics. SLAs should also include remediation for failing agreements. The items which are usually focused on in the SLA are as follows:

- **Availability and Uptime:** Availability and Uptime are two of the most important attributes that CC focus on. These are not the same as a resource might be 'up' but not accessible by the user which means that it's not available. The SLA must provide in percentage the Availability and Uptime of various services.
- **Security and Privacy:** The CC are ultimately responsible for their clients' data privacy. Hence, how the CSP handle data security is of paramount importance. Strong user authentication must be used for data encryption, antivirus and malware, etc. The CSP should have intrusion detection activated at all times.
- **Performance:** Generally, Response time is considered as the attribute to determine performance. Hence, metrics must be decided on areas such as service volumes, different demands at different time intervals, QoS, workloads and peak times.
- **Network Changes:** Dynamic scaling of equipment by CSP might lead to users facing downtime. Major changes must be conveyed to the customers in advance.
- **Support Agreements:** Most CSP have round-the-clock Help Center Support for CC but Customers might provide some support on their part for their clients. This ratio of Support services provided by the IT team of companies and CSP must be decided in SLA.
- **Exit Strategy:** Appropriate steps must be included in case any dispute arises and exit strategy has to be used for CC to migrate to another service provider.

#### Consensus Assessments Initiative Questionnaire (CAIQ)

The CAIQ offers an industry-accepted way to record what security controls exist in different services like SaaS, IaaS and PaaS. CAIQ is answered by all CSP as set of Yes/No answers to questions which CC and cloud auditor may wish to ask of a cloud provider to assess their compliance to the Cloud Controls Matrix (CCM).

It enables CC to determine the security provisions provided by CSP and decide whether it's safe to migrate to the particular CSP.

27

#### Cloud Controls Matrix (CCM)

The CCM is Cybersecurity Framework for Cloud which contains 197 control objectives that can be fulfilled over 17 domains covering key areas in cloud technology. This is used as an objective based assessment of CSP by CC to determine the extent of security features provided during implementation.

65

#### National Institute of Standards and Technology (NIST)

27

The NIST cloud computing definition is accepted worldwide to provide a clear picture of cloud computing technologies and services. The Cloud computing reference architecture is a conceptual model which enables and empowers discussing the requirements, structures, and operations of cloud computing. The model can be used with any service vendor.

66

#### International Organization for Standardization/International Electrotechnical Commission (ISO/IEC)

**ISO/IEC** industry standards define many attributes of cloud computing technology like Interoperability and Security. On the basis of these defined formal levels of features, users can judge Cloud Services. This helps in avoiding confusion and introducing transparency between CC and CSP.

27

### European Network and Information Security Agency (ENISA)

ENISA comprises a group of subject experts and representatives in the fields of Industrial, Academic and Governmental organisations. Recent works by ENISA include an assessment which determines the emerging and future risks and security failures which CSP might encounter. It also outlines the key benefits and features of using Cloud Services.

## 1.2 Machine Learning

Along the norms of every product offered as a service, ML sides along as an enhancement and learning mechanism. As more and more technologies evolve with data output and deviation in results, the application of predictability unfolds in higher magnitudes. As a fitting result, ML's introduction to technological industries and products offer lucrative foundations. Among such important introductions, ML plays a pivotal role via algorithms that understand patterns and trends in collected data through viable resources. These days, Machine Learning is no longer computation-intensive and can be utilized with little compromise in performance via household desktops and laptops. In the domain of Cloud Service Selection, Machine Learning is a popular choice of primary algorithmic calculation.

### 1.2.1 Linear Regression

A simple formula  $z = aw + g$  emulates the relationship between two kinds of data popularly known as independent and dependent. Through a line of fit, one can estimate data as mostly a pattern that forms along the axes. One Independent and One Dependent variable constitute Single Linear Regression whereas **more than one dependent variable** constitutes **a Multiple Linear Regression**. In its representation,  $z = a_1w_1 + a_2w_2 + \dots + g$  emulates a perfect estimate of weights distributed over a set of variables that together answer an independent variable.

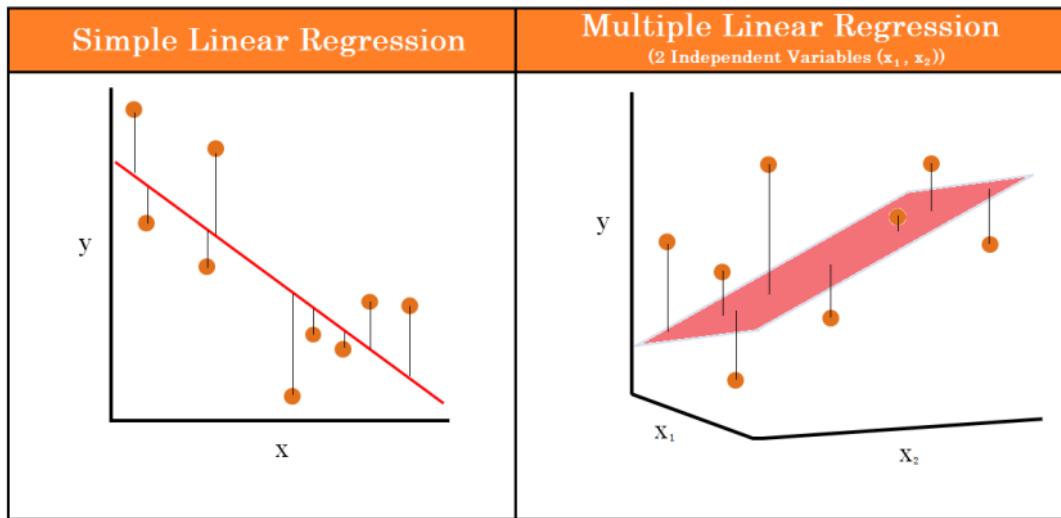


Figure 1.1: Linear Regression Common Types

## 1.2.2 Linking ML with Cloud Service Selection

Specific to Cloud Service Selection, the vast potential of Machine Learning can be incorporated in order to yield many solutions ranging from instance type finalization to service migration. As we delve deep into gaining insights from cloud customer data that is made public by Cloud Service Providers, we notice several patterns that can be exploited for product-side and technical upgrades. Significantly, when utilizing user inputs and feedback as training models, engineers can predict Cloud Service features like availability, reliability, throughput, etc. Thus, the relationship between Cloud Service Selection and Machine Learning is immensely rich with various applications. For instance, Regression can learn to predict the expected value of Connectivity based on the available values of Throughput and Standard Deviation.

## 1.3 AHP/MCDA

For making the right business decision, there is a need to utilize advanced analytical decision-making algorithms. This is classified as a domain of Operations Research. Under this, Multiple Criteria Decision Making (MCDM) is an integral tool used to evaluate numerous criteria that are conflicting or are different relative to one another. By organizing decision-making problems into structures of relative importance, enterprises can expect uniformity and consistency in their decision finding. Using certain decision-making software, MCDM's various types such as Ranking based on optimal points, Weighted Sum Model, Weighted Product Model, and Analytic Hierarchy Process are developed. We expand our project's scope of finalizing instance types of a given service type by utilizing Analytic Hierarchy Process.

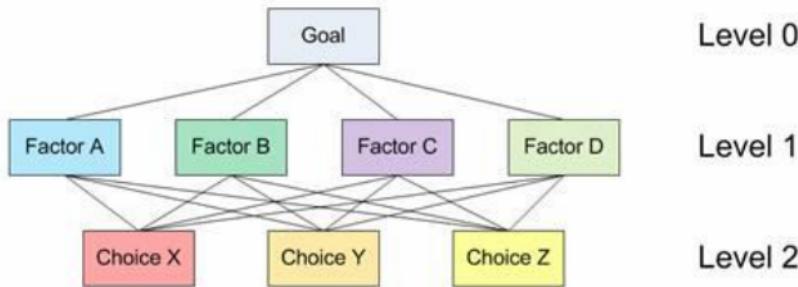


Figure 1.2: Sample AHP Tree

Analytical Hierarchy Process is an advanced decision-making algorithm that solves complex mathematical problems. To reach an overall goal, a tree-like structure is envisioned by constructing a model. The leaf nodes of the tree begin with the inputs to AHP. These are relative importance indicators of all the inputs. For example, if someone wants to know the overall strength of teams that play in the English Premier League. As the leaf nodes, we input the following parameters: Attacking Strength, Defense Strength, Midfield Strength, and Goalkeeping Strength.

In these nodes, we define the relative importance of each team per parameter. For example, take Attacking Strength. It is defined as:

attacking\_strength = {(Chelsea, Arsenal): 1.2, (Chelsea, Manchester United): 1.4, (Manchester City, Arsenal): 1.3, (Arsenal, Manchester City): 0.7 ...} and so on. The values defined are the relative comparison of two teams based on an individual strength.

Such dictionaries are prepared and are fed as an entire structure in a leaf node. To determine the values of the nodes, we apply several techniques specific to the datasets. Take the first entry: (Chelsea, Arsenal): 1.2. Here, it means that

Chelsea is 1.2 times better than Arsenal in terms of Attacking Strength. Similarly, (Arsenal, Manchester City): 0.7 is an entry that defines Arsenal's attacking strength to be 0.7 times that of Manchester City. With such values, one can draw insights on individual importance to each criterion. Once all leaf nodes are described, the hierarchy becomes the most important step. Hierarchy is a set of priorities or ranks that define the levels of the nodes in the tree. In this specific example, suppose we want to place the next highest importance on Match Performance. Further, all the leaf nodes are added to a set of match\_performance nodes based on a ranking mechanism. Among the leaf nodes, the relative importance between each leaf node (Attacking strength, midfield strength and others) is established. Assuming that we define Attacking strength to be the most important, the match\_performance node will be defined as:

match\_performance = {(Attacking, Midfield): 1.2, (Attacking, Goalkeeping): 1.4, (Defense, Attacking): 1.3, (Midfield, Defense): 0.7 ...} and so on.

As we further describe each of the levels, we finally create a Comparison Matrix for each node at each level. The root node will be the overall result after consideration of each level of the AHP tree. Using the Compare Matrices and establishing relative importance, criterion weights can be output as follows: Arsenal: 0.2, Chelsea 0.45, Manchester United 0.3 and so on... The overall sum of the weights are 1 and these instances are described relative to one another. The best teams are those which are higher in magnitude relative to the other teams.

## 1.4 Real World-Application

The Cloud Service Selection model employs **Multi Criteria Decision Analysis (MCDA)/Analytic Hierarchy Process (AHP)** to assess and evaluate several instances of service types. Similarly the same techniques can be applied to solve various other problems in today's world.

- Knowledge Based Economics: MCDA is used to evaluate decision systems involving Budgets, Expansion of Business, Expenses, etc.
- Resource Provisioning: Resources can be provisioned effectively while keeping relevance, urgency and budget as criteria in MCDA model.
- Assessing competitiveness of teams: Competitive factors can be assessed based on which teams can be ranked in any Competitions
- End Product sales in market: Decision analysis models can be used in taking in attributes like various dynamically changing factors of customers, boundary conditions like budget, location, etc. and analysing previous sales data to find suitable target groups and locations to sell the product.

## 1.5 Organization of the Project Report

The project report is organized as given below:

In Chapter (2), we discuss the problem statement and our solution to the problem. The same chapter also deals with the other existing technology. The Chapter that follows i.e. chapter (3) consists of the details on the literature survey of the papers which align to the problem statement and the proposed solution. In Chapter (4), we present the System Architecture and Design from User point of view as well as through Sub-systems divided in the entire structure of model. In this chapter, the Data Flow Diagram makes it easier to understand the overall control and sequence of the system. The next chapter, chapter (5) gives the requirements and details about the implementation of the system along with Directory Structure. Chapter (6) deals with the testing of the product. In Chapter (7), we discuss the end results and factors influencing the results of the system. The same chapter deals with establishing the optimal parameters for the system. Chapter (8) concludes the paper along with diving into some of the Future Enhancements. Chapter (9) mentions the references used during the development of the system. The other supporting information and the source code are gathered in the last section named 'Appendix'.

## Problem Statement and Proposed Solution

### 2.1 Problem Statement

To develop a sophisticated cloud service selection system that constitutes user requirements elicitation & the logic that derives the equation of fit that serves the need to present heuristically-accurate results to the cloud user utilizing historical cloud data.

### 2.2 Existing Systems

#### CUELOGIC Cloud Optimization

Related to Private, Public, and Hybrid Clouds, CUELOGIC deals with Cloud Optimization for enterprises and other customers to choose the right cloud product for them. They offer consulting for choosing cloud services and assess the value of a cloud. In addition, they take responsibility for administering cloud infrastructure for enterprises. To build an ecosystem that sustains reasonable prices and valuable cloud services.

#### DENSIFY SOLUTIONS

This product offers cloud control for enterprises along with optimization features. Its unique Cost Management tool caters to allocating reasonable pricing to cloud services and amplifies the quantity output that can be expected from a cloud service. Its Infra as a Cloud feature stands out in offering Code Optimization where cloud services are commissioned at the lowest price ranges by automating the process of cloud selection.

#### OPSANI

OPSANI is an autonomous group that offers continuous cloud optimization as a service. It extends the workability of Artificial Intelligence in the domain of Cloud Computing. This AI brings closer the ability to predict where investments can be saved and where they can be maximized in the cloud. Main objectives of this product are not limited to improving cloud performance and adaptable performance tuning.

### 2.3 Proposed Solution

#### 2.3.1 Cloud Service Optimization

The process of determining the right amounts of cloud service resources such as allocation of storage, provisioning of CPUs, configuring memory requirements, etc. while administering judicious spend on cloud services in terms of quality and quantity can be defined as Cloud Service Optimization.

Cloud Service Optimization is generally done to cut costs that can ramp up the overall bill and to reduce wastage of allocated resources. It is achieved through paid software as described in Existing Systems that usually generate optimal findings with respect to choosing the right cloud service based on resources, expected statistics, and other common metrics.

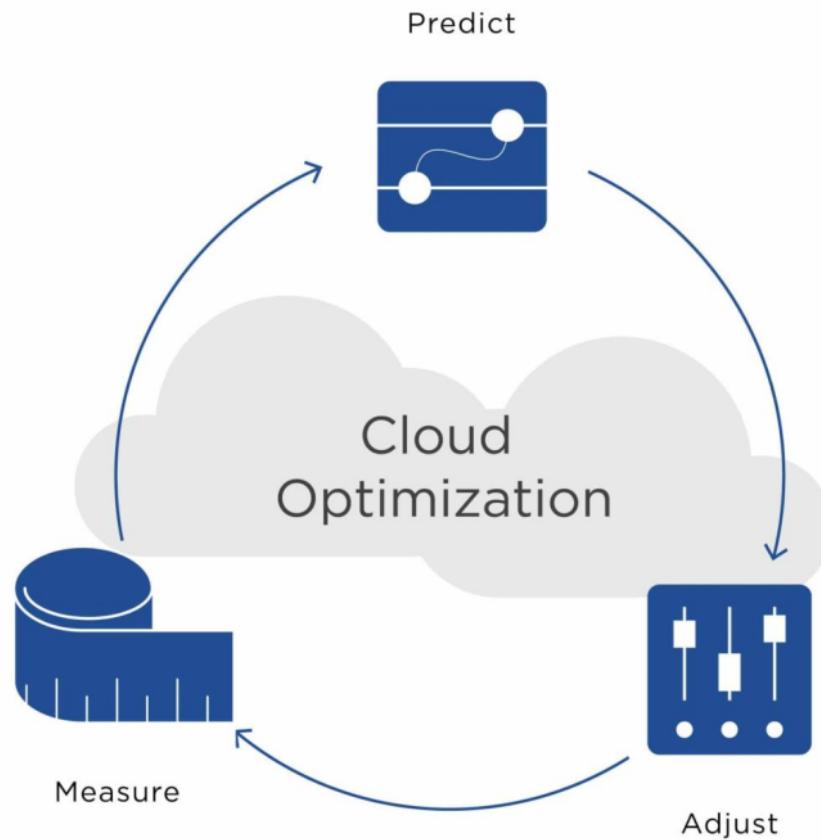


Figure 2.1: Cloud Optimization Common Metrics

In our proposed solution, we utilize three important metrics:

- (i) Filters: Filters are content areas that describe the fundamental features of a cloud service. Ex: Throughput, Uptime, Downtime, etc.

In our proposed solution, these are the following filters that are accepted for the services:

SERVICE TYPE: AWS EC2

1. Consistency
2. Error Rate
3. Latency
4. Elapsed Time
5. Connection Time
6. Throughput

7. CPU Utilization
8. Memory

SERVICE TYPE: AWS RDS

1. Consistency
2. Error Rate
3. Latency
4. Elapsed Time
5. Connection Time
6. Throughput

SERVICE TYPE: AWS S3

1. Consistency
2. Error Rate
3. Latency
4. Elapsed Time
5. Connection Time
6. Throughput

The user chooses their desired filters and also allocates priorities to them to rank them overall. For services without instance types, we utilize the prediction model to estimate the expected values of one of the above parameters by considering inputs to the other values.

(ii) Instance Types

Instance Types of Cloud Service Types are types of a given service that vary in the resources offered, support extended, and metrics available. The ranking that is done in our solution is to optimize choosing the best instance type of a cloud service type among a set of alternatives.

In our proposed solution, we offer instances of AWS EC2 and AWS RDS. With these instance types, we establish results (the best instance type) based on our findings from rigorous testing through JMeter.

(iii) Rankings/Priorities

For each of the opted Cloud Service Type, we generate rankings as a type of priority-based comparison matrices created in the backend. The weighted averages of all criteria are decided after the user enters rankings. For cloud services with no instance type, we utilize the ML algorithm to predict the expected value of a content area given the input values for the rest of the content areas. These input values to AHP are defined as follows:

The inputs are defined in a structured way in the user interface. They are non-negative integers/decimals that signify the quality of the respective filter.

- Consistency (stdDev): 100 to 1000. Here, 100 indicates a low level of Consistency and 1000 indicates a high level of Consistency. The standard deviation value of the dataset under scope informs us about the scattering / clustering of all the data points present in the data set around the obtained average. The smaller the value of standard deviation obtained, the higher is the consistency of data.
- Error Rate (errorRate): 0 to 100. Here, 0 refers to low error rate and 100 refers to high error rate. It is the percentage value of errors caused due to various reasons when connecting to the backend hosted by a service type. Total percentage of errors found for a particular sample request. 0.0% shows that all requests completed successfully. Total equals the percentage of error samples in all samples.
- Latency (Latency): 0 to 300. Here, 0 refers to a low level of Latency and 300 refers to a high level of Latency. application client.
- Elapsed Time (elapsed): 0 to 300. Here, 0 refers to low Elapsed time and 300 refers to high Elapsed time.
- Connection Time (Connect): 0 to 300. Here, 0 refers to the low level of Connection Time and 300 refers to the high level of Connection time.

- Throughput (Throughput): 0 to 5000. Here, 0 refers to a low level of Throughput and 5000 refers to a high level of Throughput. Hits/sec, or total number of requests per unit of time (sec, mins, hr) sent successfully to server during test.

Once all of the above-mentioned parameters are received by the backend, appropriate algorithms are called and run through the inputs.

### 2.3.2 AHP/Multiple Criteria Decision Analysis

In our AHP model, the leaf nodes are relative values of each instance type that are derived by calculation from JMeter logs. These relative values of each instance type of a given service type are stored in the DB and are retrieved when required. After defining each of the user-requested content areas with our relative representation of research-output log values, we determine the ranks or priorities that will become the input to the parent/root node. Here, the relative ranks are defined based on The Inverse Law. If a content area is “good” with increasing values obtained by research, then the ranks are as defined by the user. If a content area is “bad” with increasing values obtained by research, then the inverse of the ranks are fed as input relative ranks.

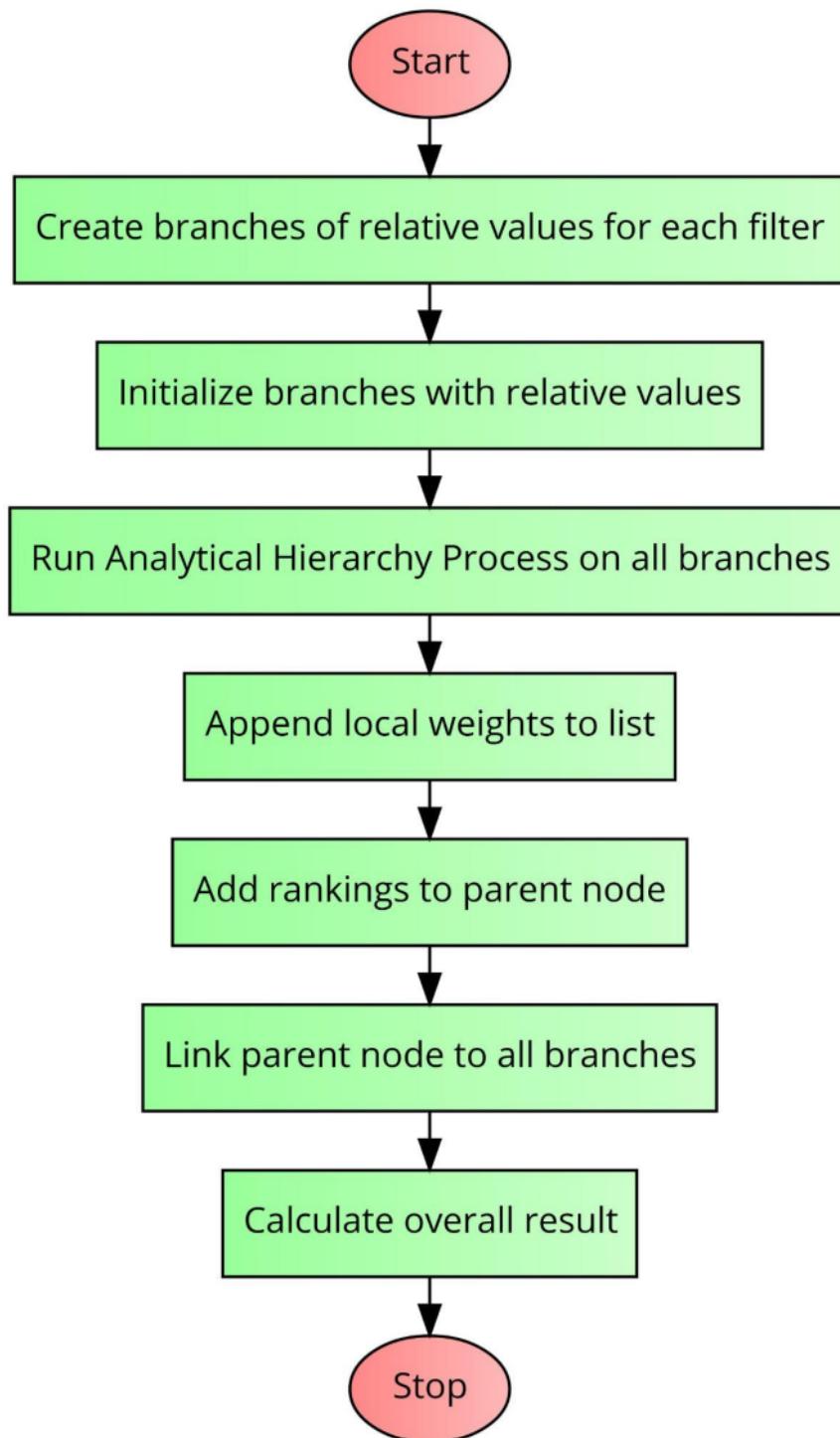


Figure 2.2: AHP Algorithm

### 2.3.3 Optimal Rank Decision

We use the Inverse Law to decide the optimal rank. For content areas that have quality inversely proportional to magnitude, the inputs for the highest level of AHP are taken as the inverse of the input ranks. For content areas that have quality directly proportional to magnitude, the inputs for the highest level of AHP are fed directly after calculating relative rankings. For the scenario when the same priorities are entered for all the content areas, all values are defaulted to 1. With this ranking system, efficient Analytic Hierarchy weights are generated through numerous iterations of Comparison matrices.

### 2.3.4 Graph Theory

Graph Theory is an old Mathematical concept with a number of modern applications in today's world like modelling Network Topologies, resource allocation and scheduling, etc. Graph Theory is a very popular technique in the field [32] Computer Science to model pairwise relations between nodes. Here, Graph Theory is used to analyse [8] and model CSP, Cloud SLA and Security Controls of Cloud. Graph Theory was found to be optimal in defining a network structure and representing information about relationships between Nodes. Graph Theory defines relationships [67] between Nodes such as Industry Standards, Organisations that form these standard [32] Cloud Security Controls and Cloud Controls Matrix (CCM) given by [32] Cloud Security Alliance (CSA) and CSP (e.g., Amazon, Google, Microsoft). The SLA standards defined by EC, ENISA and ISO/IEC have a set of 12 content areas and 82 components. The CAIQ proves to be an efficient method for collection of data and building graphs from it. The CAIQ is the document of Questions and their Yes/No answers as provided by CSP.

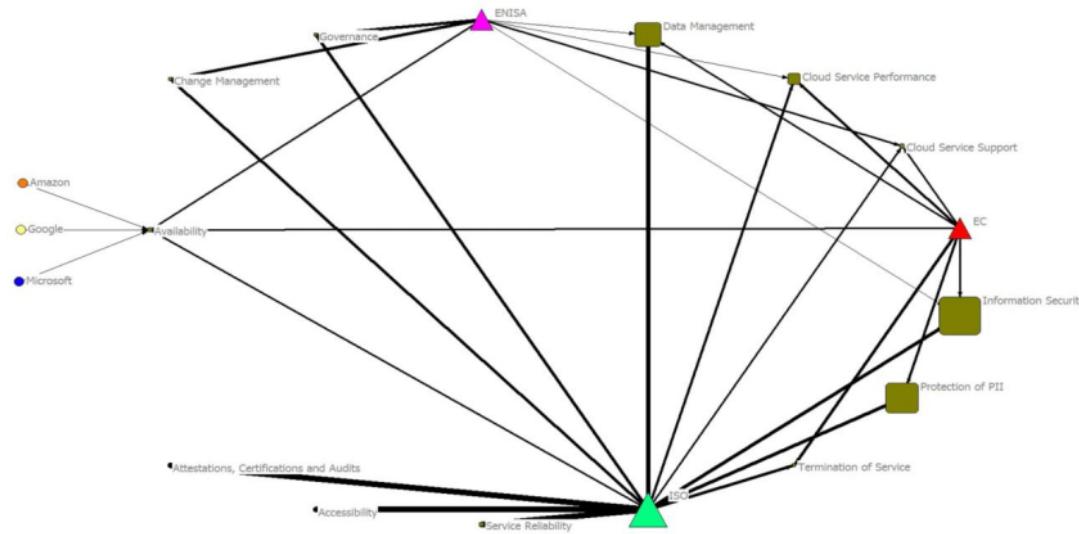


Figure 2.3: Graph Theory

Once the graph is constructed, the importance and relevance of each SLA content area is determined and quantified. The entire network structure formed between these entities along with their relative importance are then provided as input to AHP.

### 2.3.5 JMeter Log Collection

In order to establish trust between CSP and CC, the services must be tested against a wide range of parameters and attributes to check the efficiency of each service. These attributes include CPU utilisation, Memory utilisation, Downtime, Uptime, Availability, Throughput, Latency, Consistency, etc. Although CSP have their own monitoring services integrated with the services which are provided, users tend to rely on external sources which conduct unbiased assessment of performance. Hence, to facilitate independent testing of Cloud resources, Jmeter tool is used for testing and analysing performance of services. Various instances of AWS Elastic Cloud Compute (EC2) and Relational Database Storage (RDS) as well as Simple Storage Service (S3) are tested and logs are collected every day for more than thirty days with varying factors (which might affect the performance such as Number of user requests invoked, Timestamp, Storage intensive and Computation intensive processes, etc.) are set. The EC2 instances and S3 services are tested by invoking HTTP requests through Jmeter while RDS instances are tested by sending JDBC requests to the tables created in the database.

### 2.3.6 Analysis of Historical Data

Once the logs obtained through JMeter are received by the backend, they are prepared to be fed into the Analytic Hierarchy Process and the Multiple Linear Regression models whenever the user interface requests a particular processing. To prepare the JMeter logs, we perform the following steps:

- Attempt login/signup
- Generate login session
- Traverse through the logs file(s)
- Obtain the required columns data
- Calculate averages of all data
- Upload the log files to the backend

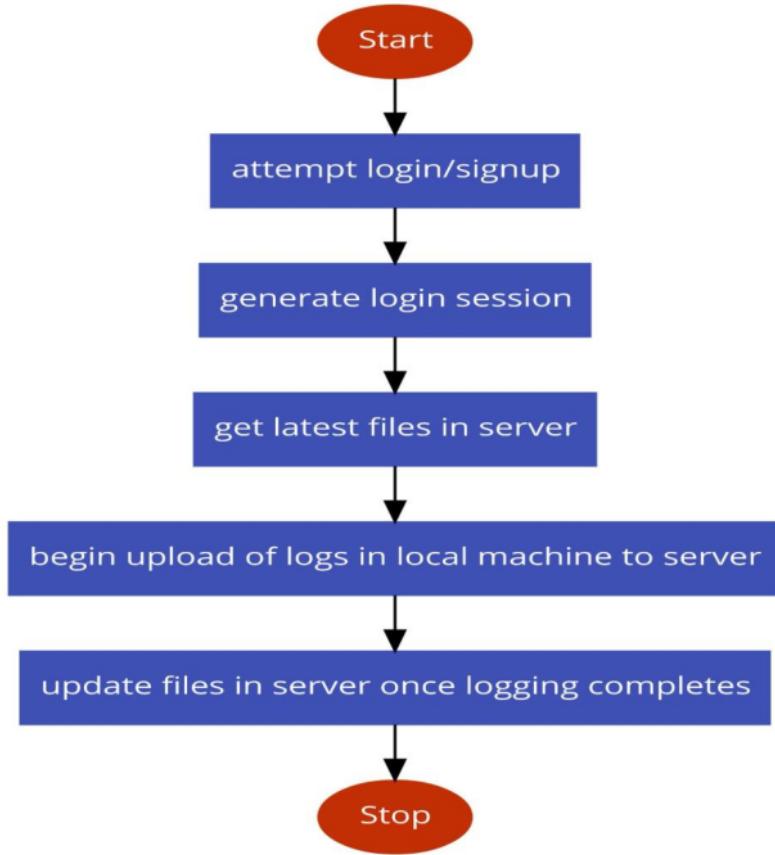


Figure 2.4: Endpoints algorithm

#### Calculating Throughput, Consistency (Standard Deviation)

- Assume that the value above from a logging session summary is 'v'
- $\text{loggingSessionValue} = \# \text{ofSamples collected in a logging session} * v$
- $\text{totalValue} = \text{Sum of (loggingSessionValue) for all logging sessions} / \text{Sum of (\#OfSamples) for all logging sessions}$

#### Calculating Elapsed time (connection + response), connection time, and latency

- Assume that the value above from a logging session is 'v'
- $\text{totalValue} = \text{Sum of all (v)} / \text{total number of logging sessions}$

With the obtained total values for each of the content areas, we store all of the calculated total values in the database for further use.

#### 2.3.7 Interactive User Interface

The proposed solution involves the end product web application (created using the React JS Framework) hosted on the free hosting platform - Heroku.

It consists of an interactive User Interface with simple steps that are clearly defined to ensure a smooth usage experience with respect to the user.

Incase of a first-time user, there is a detailed user manual specified under the 'FAQs' hyperlink. The 'Contact Us' hyperlink consists of the email IDs of the creators. They can be contacted incase of any discrepancies. The 'Dashboard' hyperlink is the section where the user inputs are taken to render graphical data which provide useful insights into the trustworthiness of the services selected by the user.

Below is a detailed explanation of the functionality of the web application in a question answer format.

- For Amazon EC2 and Amazon RDS:

1. What AWS Service are you looking for?

Clicking on the 'Select Service' Button will display the services which are available for comparison and calculation of trustworthiness metric. Select the service type as per the requirement.

2. What performance metric do you want us to focus on?

Select from a wide range of available Filters and assign priority based on how important each attribute is for your business requirement. Checking the 'Assign same priority to all filters' will assign same priority to all attributes.

Don't forget to select the filters that you want to apply after this step. Individual priorities can also be assigned by unchecking the above checkbox, clicking on each filter and filling the priority numbers (Integer type in range [1, n]: 1 being the highest priority,  $n \leq$  Total number of Filters available) in the textboxes which appear beside them. To deselect any filter, simply click again on the selected one.

3. What instance types of the above listed AWS service do you want?

Select the instance types that you want to compare between and click on Save. If no instances are selected all of the shown instances will be automatically chosen for Trustworthiness Evaluation.

After successfully uploading the above data, click on Show Results to assess the trustworthiness value and graphical results of the comparison.

- For Amazon S3:

1. What AWS Service are you looking for?

Clicking on the 'Select Service' Button will display the services which are available for comparison and calculation of trustworthiness metric. Select S3.

2. What metric should ML Prediction be applied to?

37: 'Please Assign different values to each filter' must remain unchecked for S3. For S3, Users must choose  $n-1$  ( $n$  being the total number of attributes) filters and must provide the values for each of the selected filters. The value of the only 1 remaining filter is predicted. Here, priorities can be positive Integers or Decimal values.

Allowed values for the selected filters:

Consistency (stdDev): 100 to 1000. Here, 100 indicates a low level of Consistency and 1000 indicates a high level of Consistency. The standard deviation value of the dataset under scope informs us about the scattering / clustering of all the data points present in the data set around the obtained average. The smaller the value of standard deviation obtained, the higher is the consistency of data.

Error Rate (errorRate): 0 to 100. Here, 0 refers to low error rate and 100 refers to high error rate. It is the percentage value of errors caused due to various reasons when connecting to the backend hosted by a service type. Total percentage of errors found for a particular sample request. 0.0% shows that all requests completed successfully. Total equals the percentage of error samples in all samples.

Latency (Latency): 0 to 300. Here, 0 refers to a low level of Latency and 300 refers to a high level of Latency. application client.

Elapsed Time (elapsed): 0 to 300. Here, 0 refers to low Elapsed time and 300 refers to high Elapsed time.

Connection Time (Connect): 0 to 300. Here, 0 refers to the low level of Connection Time and 300 refers to the high level of Connection time.

Throughput (Throughput): 0 to 5000. Here, 0 refers to a low level of Throughput and 5000 refers to a high level of Throughput. Hits/sec, or total number of requests per unit of time (sec, mins, hr) sent successfully to server during test.

After selecting the filters, press Save to upload the data. Click on Show Results to assess the trustworthiness value and graphical results of the comparison.

## 2.4 System Requirements

The Model makes the job of all Cloud Customers hassle-free in selection of suitable services according to how important the services' attributes are for their business requirements. The system consists of three modules: Frontend, Endpoints and Backend. The Frontend incorporates an interactive User Interface where the users can enter their preferences regarding Services, Filters and Priorities and Instances, if applicable. These inputs are saved and sent to the Backend where AHP technique is applied over various logs and data collected from Endpoints to rank and assess the performance of services according to Users' preferences of Attributes.

The System Requirements sequentially are as follows :

1. When the users open the webapp, the homepage displays an overview of Cloud Service Selection and its importance.
2. Three tabs which are available are : Dashboard, FAQs and Contact Us. Dashboard is where user preferences are recorded and the trustworthiness metrics are calculated. FAQs section has all instructions about how to use the Dashboard and rank services, Contact Us section has Email IDs of developers.
3. The Dashboard section asks Users questions regarding the Service Type they want to choose, Filters and Priorities that should be assigned to them and Instance types of Services, in case of EC2 and RDS.
4. Various checks are included to avoid input of incomplete or wrong data like One Service must be chosen, Priority must be assigned, more than one Instances must be chosen for comparison, etc. When all these conditions are satisfied, users can click Save and the data is sent to Backend.
5. Backend employs MCDA/ AHP technique to assign weights to all attributes according to user preferences and logs sent from Endpoints.
6. The AHP technique helps in assessing performance as well as ranking of services. The Graph data is sent to Frontend which displays all information of trustworthiness in a graphical and user-friendly format.
7. The logs which are collected from Endpoints consist of performance records of various services against parameters such as Availability, CPU and Memory Utilisation, Latency, Response Time, etc.

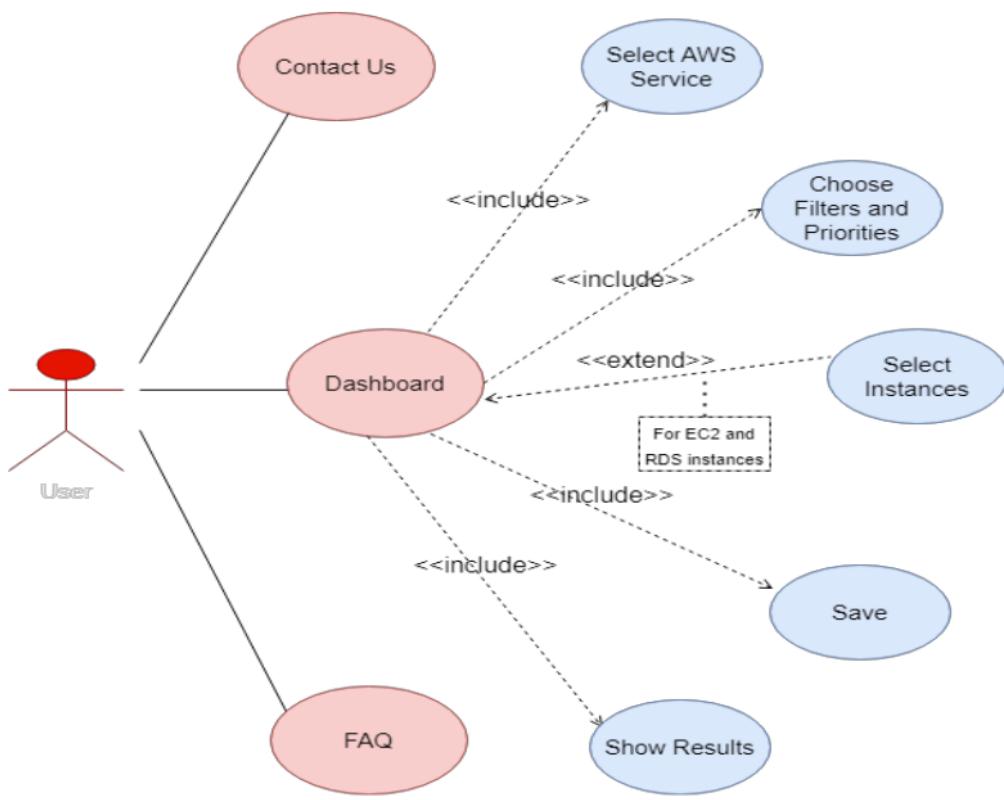


Figure 2.5: Use Case Diagram

## Literature Survey

### 3.1 "Predict Trustworthiness of Cloud Services Using Linear Regression Model"

The pay-as-you-go feature of many CSPs has enabled enhanced scalability in providing resources based on the requirements of Cloud Customers (CC). Incidentally, scalability introduces a more challenging trust relationship between CSPs and CC as users must depend on the availability of services at the time of need. In addition to scalability, users need an effective system to rank CSPs on their distinct QoS parameters based on specific user purposes. Mahesh K et al. suggest a Linear Regression Model to assess trustworthiness and choose the most optimal candidate among several Cloud Services based on user criteria. Initially, Linear Discriminate Analysis (LDA) technique was used for clustering and ranking similar services according to their QoS scores. LDA aims to filter out the services which are not along the lines of QoS attributes specified by calculating the similarity degree between service request and service. After the initial confidence is established using LDA, CC should verify confidence further by observing the adherence to SLA on the grounds of transparency, integrity, compliance, and competency in the past. If SLA verification is carried out by a Cloud Broker (CB) who acts as an intermediary entity between CC and CSPs, then the trust quotient partly depends on confidence between CC and CB. This is known as a reputation-based trust mechanism system. The model analyses different services based on QoS parameters – Response Time, Throughput, Availability, Success ability, and Price in the first step. In the next step, further confidence is checked by computing the trust score and ranking of services – out of nine CSPs in the evaluation process, the best service scored 0.4740 while the worst service scored 0.2724.

### 3.2 "Cloud service trustworthiness assessment based on cloud controls matrix"

CCM is an initiative by CSA that deals with describing control guidelines for a cloud service's provision of security. Under the control guidelines, there are control domains that are further divided into a number of controls. In CCM v3.0.1, we observe 133 controls. According to Kanpariyasoontorn & Senivongse, it is pertinent to analyze the features of each CCM control in order to produce a trustworthiness metric. The method proposes a Cloud Service Trustworthiness Assessment Method which initially defines the attributes & sub-attributes of Security and Dependability as Trustworthiness is defined to be a result of them. Sub-characterization of Security & Dependability ensues and the new attributes map associations with the following standards: (1) The NIST SP800-53 Control Standard and (2) AICPA Control Standard both of which map to the CCM. Next, we define a degree of association between every CCM control domain and sub-attribute with a matrix called 'Association Matrix'. To illustrate, a control domain like Application & Interface Security has degrees of 0.75, 1, 0.75, and 0 for the sub-attributes Confidentiality, Integrity, Availability, and Reliability respectively. The assurance weights for the control domains must be calculated because security assurance is one of the two components that influence trustworthiness. With the mathematical product of the assurance weights and the degree of compliance of a service with a given control domain, we establish the capability of a service 'P'. A service capability matrix is constructed by merging all the service capabilities and is multiplied with the Association Matrix to yield the Service Trustworthiness. This technique presented a self-assessment mechanism which clearly raises questions about biases and the validity of the service trustworthiness. A germane observation is that CSA is an esteemed organization whose reputation holds value enough to treat Kanpariyasoontorn & Senivongse as a promising methodology for assessing trustworthiness.

### 3.3 "Fuzzy logic based cloud service trustworthiness model"

Pandey & Danic<sup>23</sup> make use of Fuzzy Logic for building a Cloud Service Trustworthiness Model (CSTM) whose goal is to provide a fair evaluation of the trustworthiness of cloud services. The design focuses on agents for 'Requestor' and 'Provider' that are stakeholders which participate in the smooth functioning of a cloud service. As and when these stakeholders request or register for a new service, common and special parameters that constitute trustworthiness are involved in a weighting and positioning process. The listed parameters are defined with first and second grade indexes before determining their weight distribution for both indexes. A Level is ascribed to different parameters based on their quality and an evaluation Fuzzy Matrix is constructed for the first-grade index. It denotes the membership degree of a given parameter of a cloud service in specific trustworthiness levels. Processing through the Fuzzy Comprehensive Evaluation Model is a cardinal phase. Here, the overall trustworthiness membership degree of a cloud service is solved for on different levels. The final result brings about an adjective like 'high' and 'low' for trustworthiness levels of individual parameters. This stakeholder-based topology not only evaluates the overall trustworthiness of a cloud service but also the trustworthiness of individual parameters such as Security, Maintainability, and Usability.

### 3.4 "A Trustworthiness Evaluation Framework in Cloud Computing for Service Selection"

CCs and CSPs often don't have clear information about each other for cloud service selection. A trustworthy relationship should be established between CCs and CSPs on the basis of the reputation of services<sup>14</sup> as well as suitable trust factors. Due to the unavailability of uniform trust metrics for different kinds of services, L. Wang and Z. Wu<sup>30</sup> propose a trustworthiness evaluation framework which is made of five sections. The Pre-processing Section collects trust factors (quantitative and qualitative) of CSPs which are categorized and converted to a unified scale using the Trust Factor Management Section. Trust coordinate is used to further categorize them into Subjective trust, Objective trust, Direct trust, Indirect trust, Inflow trust, and Outflow trust. Entropy value is used to define trustworthiness metrics (value between 1 and -1 where 1 is completely trustworthy and -1 is untrustworthy) to measure the uncertainty in trust relationships. These factors are then sent to the Trust Factor Processing Section for further processing like updating of common factors, dealing with special factors with trust ontology alignment approach to formalize the meaning of trust for a clearer understanding for user expectations, and assigning weights to all factors which are then employed in the multi-criteria analysis approach by the Trustworthiness Decision Making Section to measure, rank or sort cloud service candidates. Trust Computation using MCDA is carried out by Support Factor Clustering (based on Support Vector Machines and clustering as per CC requirements), Trust Factor Aggregation (functions of factor addition and multiplication are used to aggregate trust factors of services), and Minimum Polyhedron (finds the minimum polyhedron that covers all trust factors in trust coordinate system). To store the result for referencing it later, the Trustworthiness Record Section is utilized. The accuracy of the results is consistent after three-fold verification is carried out by comparing the outcome with feedback, data of revenue, and client growth rate as well as reviews from experts of third-party organizations.

### 3.5 "A Cloud Service Selection Method Based on Trust and User Preference Clustering"

Existing models for Cloud Service Selection do not take into account the similarity in user preferences which seem personalized and unique for each user. To consider the context information and specific user requirements, as well as interaction, requirement personalization<sup>14</sup> and similarity, QoS diversity, complexity and trustworthiness between service providers and users, Y. Wang et al. propose a Cloud Service Selection method based on Trust and User Preference Clustering<sup>14</sup> which follows a six-step procedure to produce the final measure of trustworthiness. Firstly, all CSPs register to the cloud service registration center. The Cloud Customers' (CCs) requirements are sent to the framework as personalized preferences. Condensed hierarchical clustering method is used to cluster requirements and the distance between these clusters is measured as per user service weight preference<sup>14</sup>, similarity, user service attribute score similarity, and user evaluation similarity in the third step. Threshold-based hierarchical clustering algorithm

clusters repeatedly based on [14] similarity of preferences until 'K' clusters are present and their distances are measured. The optimal cluster is obtained by assessing the cluster structure degree. The comprehensive trust [56] of direct trust, and recommended trust are evaluated in the next step. Direct trust is calculated with the help of attribute weights (objective weights are calculated using improved entropy weighting method and subjective weights are calculated using AHP), attenuation time, transaction amount, and penalty factor. In order to prevent false reporting of direct trust by malicious users affecting the CSPs' reputation in the market, recommended trust (intra-domain recommended trust and extra-domain recommended trust) is also calculated. Recommended trust is the measure between a user and an intra-domain/extr-domain recommender of the CSP. Lastly, CSPs are ranked based on their trustworthiness. Based on the user's choice the weights are assigned to direct trust and recommended trust and the final comprehensive trust is computed. This trustworthiness is the deciding factor of the service that is chosen for particular user requests. The improved condensed hierarchical method proves to be better than traditional method [64] with respect to running time of clustering, iterations taken to cluster, and convergence speed. This method has the lowest Mean Absolute Error and Root Mean Square Error as well as the highest Success Rate for different numbers of transactions and interactions which vouch for the feasibility and effectiveness of the technique and the capability to identify malicious services and feedback.

# Architecture and Design

## 4.1 System Overview

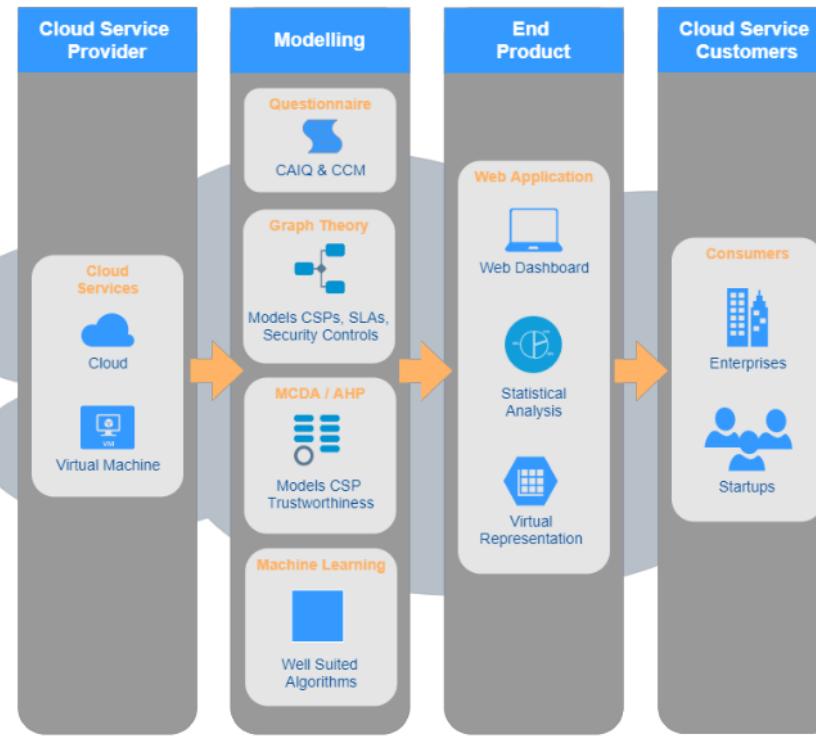


Figure 4.1: User Point-of-View System Diagram

74

From the end user's point of view, there are four important sectors that are involved in the model, the Cloud Service Optimizer.

55

- **Cloud Service Provider:** This is the Cloud Service Provider under scope. AWS, for instance, offers a set of cloud services that the end user utilizes but does not have enough data whether to trust them or not due to the currently existing metrics for establishing trustworthiness not being solid / reliable enough to provide accurate values.
- **Modelling:** This section deals with the product's configuration. It contains several questionnaires that test the standards of these Providers and these answers are mapped using Graph Theory. We also use historical data generated from JMeter, a performance testing tool to model trustworthiness. AHP and Machine Learning are used to retrieve a Trustworthiness Level based on relative importance of each content type.
- **End Product:** It is a system that contains dashboards and visual aids that help the user effectively choose the right CSP service type for their needs. Here, their considerations are included to calculate a result. All the inputs obtained by the user are accurately validated to ensure that the user is only providing the right inputs, are then accumulated into a collective object and are sent to the backend for processing. Once the processing is complete, the results are rendered in the form of graphical structures showcasing

statistics. They serve to be visually interactive making it easy for the user to read and comprehend the results generated.

- **Consumers:** The consumers extend not only to enterprises but also to startups at large. It may include individuals who are doubtful with respect to migrating their applications to the cloud and might require some additional inputs in the form of accurate trustworthiness quotients.

## 4.2 Software Architecture

### 4.2.1 System Block Diagram

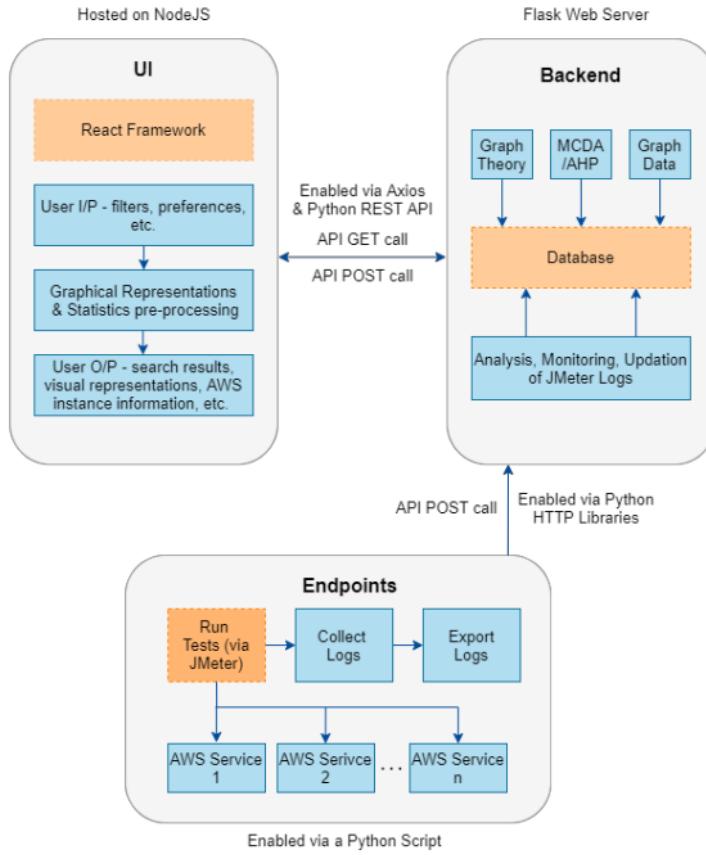


Figure 4.2: End-to-End System Diagram

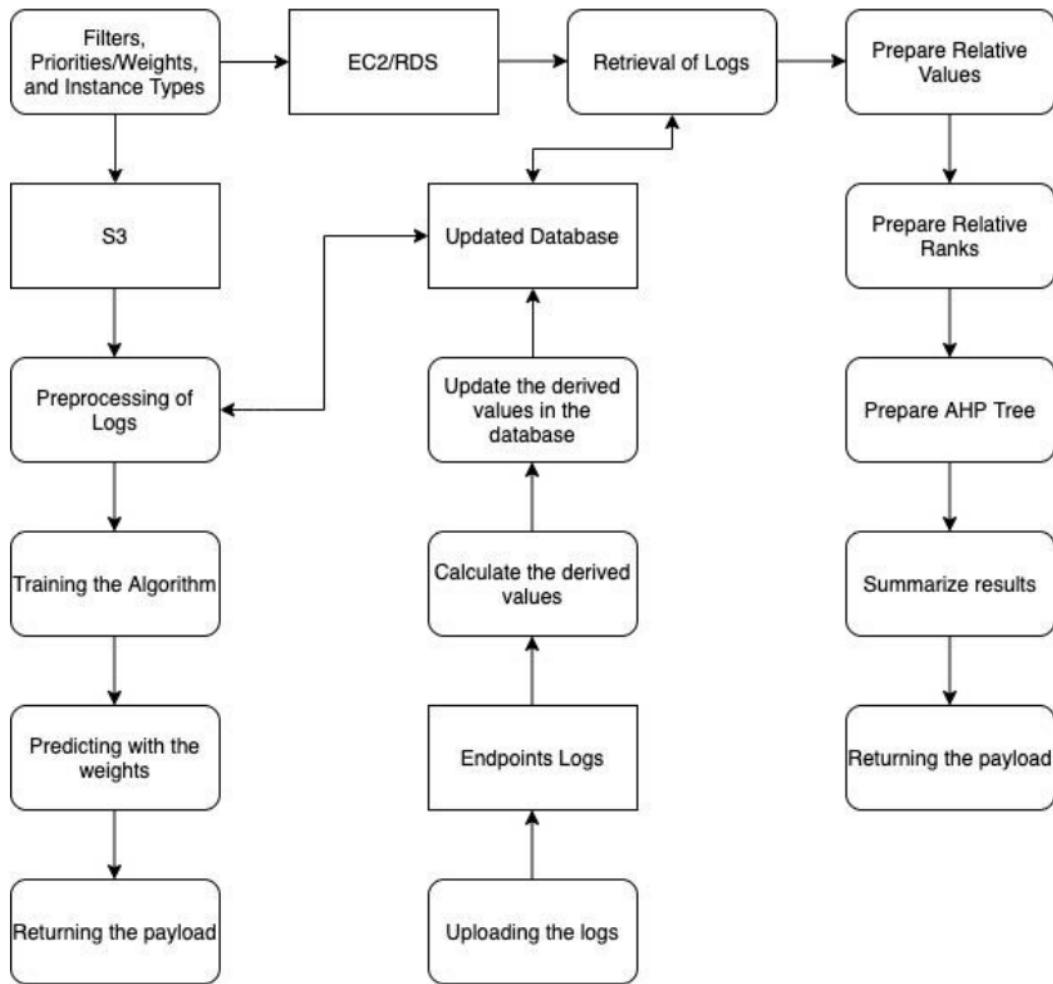
From a technical perspective, the user interface is a web application hosted on Heroku using React framework. Here, the user input is taken for filters and rankings and after communicating with the backend, an appropriate result is rendered through graphs and charts. Each instance type of a content area is ranked individually along with an Overall result. For services without instance types, like S3, we use Machine Learning to predict the results that can be expected from the service.

Endpoints consist of research that is related to running tests and collecting logs on these various instances of Cloud Services. Here, JMeter is used as a tool along with various plugins like Perfmon and JDBC Driver to get reliable data

and we perform multiple levels and intensities of testing. We have collected about 50L rows of logging data that is built as the fundamental training set in the project. We test various content areas like CPU and Memory utilisation Latency, Throughput, Availability,etc. and these are processed and sent to the backend via a Python script.

Next, the backend deals with several components that interact with the front-end as well as the logging endpoints. Here, the user input is processed via Analytical Hierarchy Process, a decision-making algorithm used when multiple criterias must be ranked relatively. The backend is a Flask Python backend hosted on Heroku that uses REST APIs to communicate. The database used is a relational MySQL server from DigitalOcean. Logs provide cloud data that are constantly received, analyzed, and updated to the database. Graph related data is sent to the front-end when the user requests such data. For ML, multiple linear regression is used.

#### 4.2.2 Data Flow Diagram



In the above **Data Flow Diagram**, we obtain an overall understanding of the flow of information from each point in the system. To elaborate on the flow step-by-step, here is an algorithm:

1. Obtain the Filters, Priorities, and Instance Types
2. Navigate according to the instance type
3. If EC2 or RDS, retrieve the logs
4. Then prepare relative values and ranks
5. Prepare the AHP tree and summarize results and return payload
6. If S3, Preprocess the logs
7. Train the Multiple Linear Regression Algorithm
8. Fit and Predict for the input weights and return payload
9. For updating the database, upload the logs
10. Calculate the derived values
11. Store the derived values in the database.

5

## Implementation

### 5.1 Implementation Platform

#### 5.1.1 Hardware

- Processor: Intel Octal Core
- RAM:8 GB
- GPU:NVIDIA GTX1080

#### 5.1.2 Software

- Operating System: Windows 10 (64 bit) and Mac OS
- Software Used: Flask , ReactJS, NodeJS, Heroku, JMeter, MySQLWorkbench, DigitalOcean
- Programming Languages: Python, JavaScript, SQL, Java
- Server: NodeJS, Gunicorn

### 5.2 Implementation Details

#### 5.2.1 Organization of implementation files

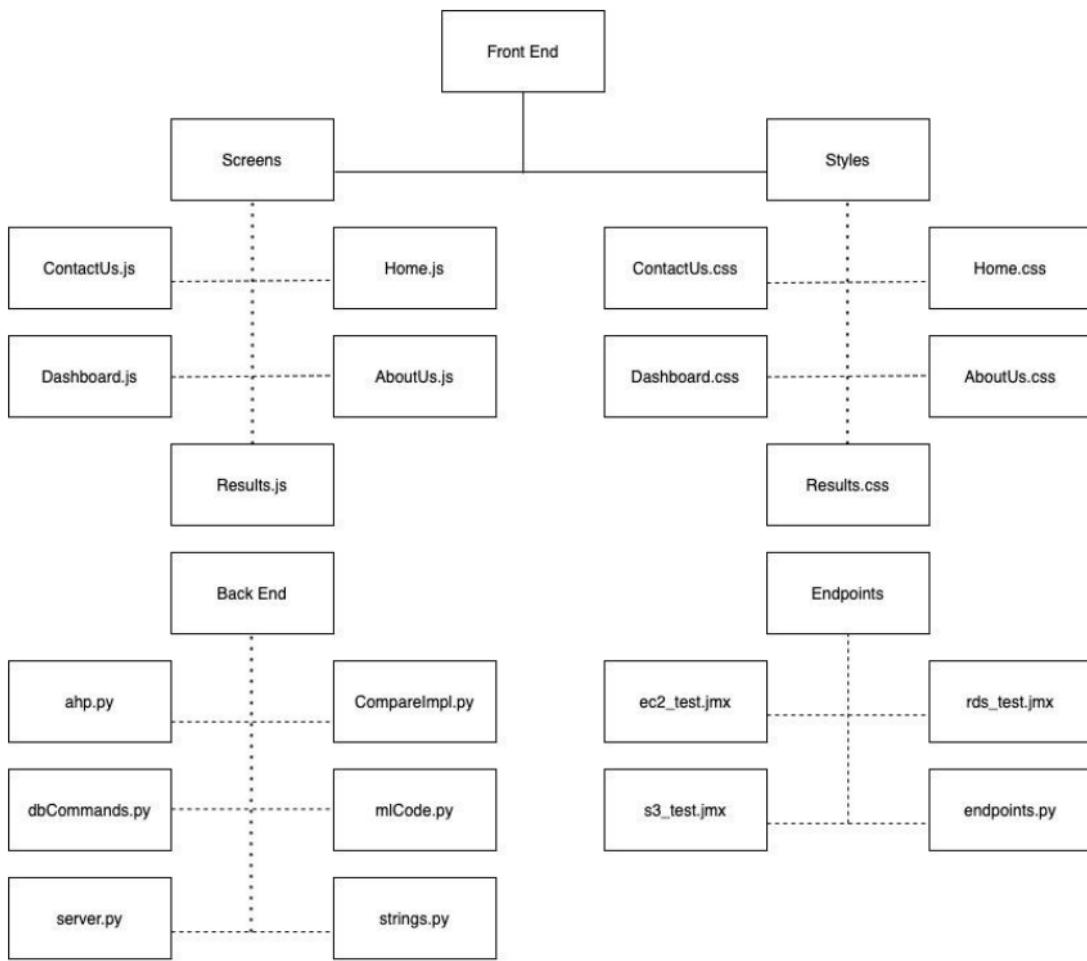


Figure 5.1: Directory Structure

The file structure of each component of the project is as seen above. In the front-end, we have two primary directories for Screens and Styles. Under them, we have the respective JavaScript and Cascading Style Sheets code files for the five pages in the UI: Home, About Us, Contact Us, Dashboard, and Results. In the backend of the project, we have ahp.py that runs the main AHP algorithm, CompareImpl.py which contains the implementation of generation of Comparison Matrices, dbCommands.py to facilitate interactions with the database which is a relational database hosted on DigitalOcean, server.py that handles all routing and immediate actions performed by the server, and strings.py that deals with the construction of the payload text and returns the string. The Endpoints contain three testing files that are responsible for the collection of JMeter logs and an endpoints.py python script to upload all the logs in the local system to the backend.

### 5.2.2 AHP Model

As described in the structure of AHP, the algorithm to preprocess the relative ranks and relative values in order to determine the AHP tree is as shown below:

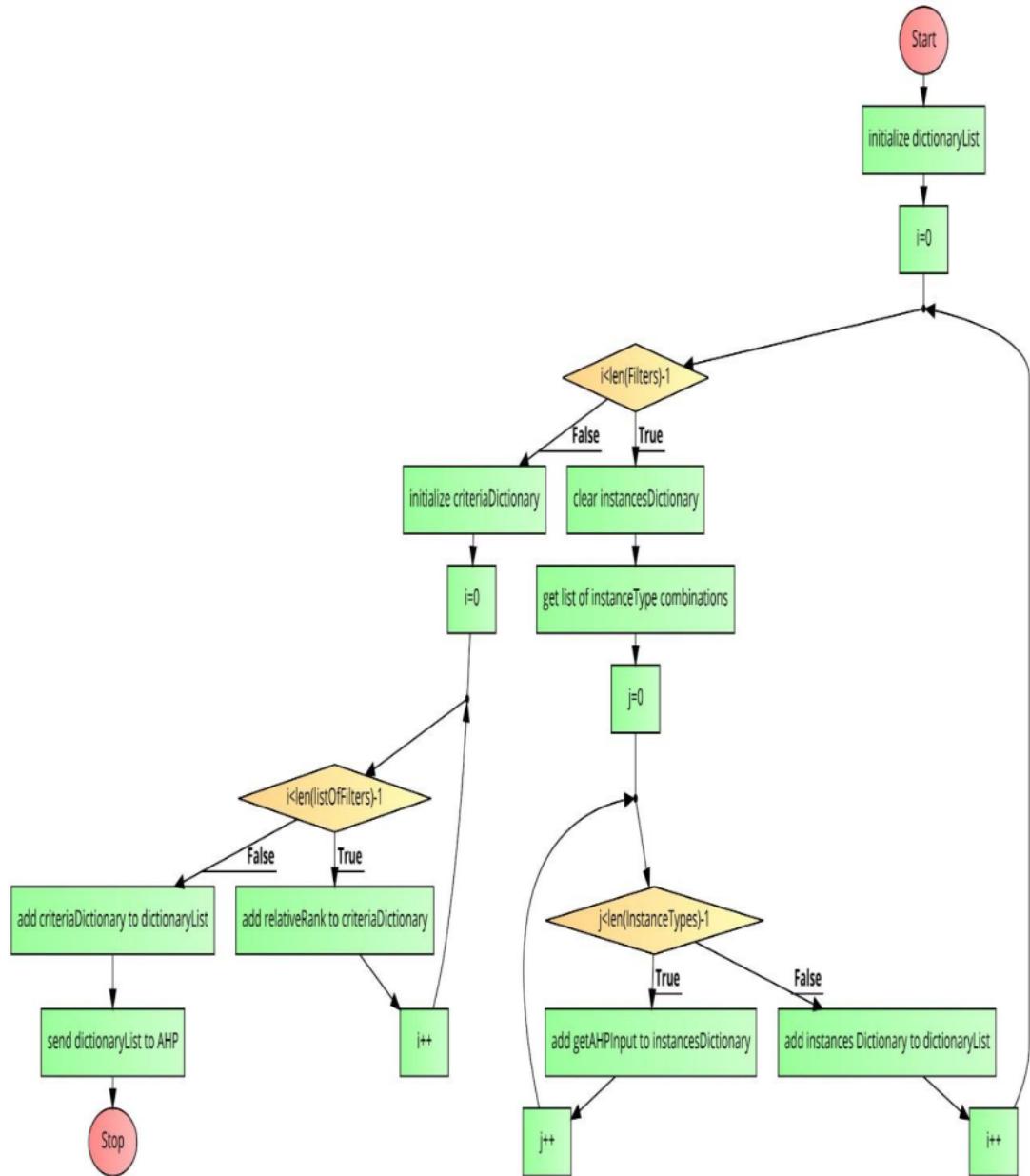


Figure 5.2: Preprocessing Algorithm for AHP

The dictionary list contains a list of inputs for both the levels of AHP.

- First, to determine the relative values, we run through the list of input instance types.
- Using the list of instance types, we determine all possible combinations to define the relative value.
- For each combination, we process and obtain a relative value.
- This relative value is determined by the processed output of JMeter tests.

- We add the relative value to the instancesDictionary.
- We repeat for all instance types.
- We add the relative values list (instancesDictionary) to the dictionary list.

Similarly, to determine the criteria ranking:

- First, to determine the relative rankings, we run through the list of input content areas.
- Using the list of content areas, we determine all possible combinations to define the relative ranking.
- For each combination, we process and obtain a relative ranking.
- This relative ranking is obtained via The Inverse Law.
- We add the relative rank to the criteriaDictionary.
- We repeat for all content areas.
- We add the relative ranks list (criteriaDictionary) to the dictionary list.

Once we have established the overall weights, we process them via a payload as follows:

```
{
  "payload": {
    "Connect": {
      "data": {
        "db.t2.medium": 0.332,
        "db.t2.micro": 0.332,
        "db.t2.small": 0.336
      },
      "displayName": "Connection Time",
      "filterCompare": {
        "db.t2.medium > db.t2.medium": 0.0,
        "db.t2.small > db.t2.medium": 1.2048192771084347
      }
    },
    "Latency": {
      "data": {
        "db.t2.medium": 0.332,
        "db.t2.micro": 0.332,
        "db.t2.small": 0.336
      },
      "displayName": "Latency",
      "filterCompare": {
        "db.t2.medium > db.t2.medium": 0.0,
        "db.t2.small > db.t2.medium": 1.2048192771084347
      }
    },
    "Throughput": {
      "data": {

```

```
        "db.t2.medium": 0.335,
        "db.t2.micro": 0.333,
        "db.t2.small": 0.332
    },
    "displayName": "Throughput",
    "filterCompare": {
        "db.t2.medium > db.t2.micro": 0.600600600600601,
        "db.t2.medium > db.t2.small": 0.9036144578313261,
92        "db.t2.micro > db.t2.small": 0.3012048192771087
    }
},
"elapsed": {
    "data": {
        "db.t2.medium": 0.332,
        "db.t2.micro": 0.332,
        "db.t2.small": 0.336
    },
    "displayName": "Elapsed Time",
    "filterCompare": {
        "db.t2.medium > db.t2.medium": 0.0,
        "db.t2.small > db.t2.medium": 1.2048192771084347
    }
},
"errorRate": {
    "data": {
        "db.t2.medium": 0.198,
        "db.t2.micro": 0.398,
        "db.t2.small": 0.404
    },
    "displayName": "Error Rate",
    "filterCompare": {
        "db.t2.micro > db.t2.medium": 101.01010101010101,
        "db.t2.small > db.t2.medium": 104.04040404040404,
        "db.t2.small > db.t2.micro": 1.5075376884422123
    }
},
"overall": {
```

```

    "data": {
        "db.t2.medium": 0.317,
        "db.t2.micro": 0.339,
        "db.t2.small": 0.344
    },
    "displayName": "Overall",
    "filterCompare": {
        "db.t2.micro > db.t2.medium": 6.9400630914826555,
        "db.t2.small > db.t2.medium": 8.517350157728696,
        "db.t2.small > db.t2.micro": 1.4749262536873005
    }
},
"stdDev": {
    "data": {
        "db.t2.medium": 0.328,
        "db.t2.micro": 0.332,
        "db.t2.small": 0.34
    },
    "displayName": "Consistency",
    "filterCompare": {
        "db.t2.micro > db.t2.medium": 1.2195121951219523,
        "db.t2.small > db.t2.medium": 3.658536585365857,
        "db.t2.small > db.t2.micro": 2.4096385542168695
    }
},
"processingNumber": "pBjcFAB0gD",
"status": "success"
}

```

As observed above, the filterCompare parameter in each of the content areas in the payload contains data suitable to represent in graphical format. This is a sample processing output sent to the front-end via the front-end/get-results API call.

### 5.2.3 Front-end component

The web application uses two packages for rendering graphical statistics. Npm provides several packages to render various charts ranging from line graphs, bar charts, pie charts, etc. The two packages used for modelling this section are - chart.js and react-chartjs-2 (the latter being a wrapper for the former). In order to provide variety and the opportunity of better understanding for the end-user, both pie charts and bar charts are used.

The aforementioned packages can be installed using simple commands such as

```
npm install chart.js --save  
npm install react-chart-js --save
```

Both the Bar and Doughnut (Pie Chart) are handy libraries that can be fed various input parameters like dataset, legend, title, and various other options. In the flow of result display, the JSON object received from the backend via Axios API calls is broken down into several key-value pairs (due to the nested structure of the JSON object received). The key value pairs are accessed one by one, and converted into a JavaScript object and fed into the data parameter for each of these charts. The data will then be rendered in graphical form on the User Interface. The graphs rendered are very interactive in nature and can be collapsed and re-rendered back again.

#### 5.2.4 JMeter Performance Testing

CPU and Memory utilisation are important parameters which indicate the health of a server. CPU utilization is a server's usage of processing resources i.e. CPU, or the amount of work handled by CPU in real time. CPU usage varies depending on the task which the CPU is handling. Some computation intensive tasks require more CPU time while others might need less because of swapping of Jobs to other resources like I/O. Memory utilisation is the amount of RAM in use by the running applications. Free Memory is always needed for tasks in execution to exhibit better responsiveness. Hence ideally Memory usage should be as low as possible although, the minimum required amount of memory is always used for running applications.

In local systems, both CPU and Memory usage can be easily analysed on Task Manager, however to assess these attributes on remote servers such as EC2 instance servers, Perfmon Plugin must be used in addition to the Jmeter tool in local system as well as the Perfmon Server Agent which must be installed on the remote server. For the plugin to work on a system - local or remote - Java Development Kit/ Java Runtime Environment must be installed.

The Client agent can be easily installed from the official website, unzipped and copied into the plugins folder of Jmeter. However, to install Server Agent on EC2 instance user data must be provided which executes at the time of System boot and carries out necessary installations.

The commands are as follows (for LINUX instance):

```
45  
26 o yum install -y java-1.8.0-openjdk.x86_64  
wget https://github.com/undera/perfmon-agent/releases/download/2.2.3/ServerAgent-2.2.3.zip  
unzip ServerAgent-2.2.3.zip  
cd ServerAgent-2.2.3  
sh startAgent.sh
```

Once the Server Agent starts and it is integrated with Client Perfmon<sup>51</sup> local Jmeter, communication flows between these agents and various performance parameters can be logged like CPU, Memory, Disk I/O, Swap, TCP, Network I/O, etc. can be tested and logged in .csv format as well as real time graphical representation is shown for the attributes.

#### 5.2.5 Database Structure

The database being used is a MySQL relational database that is hosted on DigitalOcean. The specifications of the hosted database cluster are: 1 GB RAM / 1vCPU / 10 GB Disk / Primary only / MySQL 8.

There are 18 tables in the database and are as defined:

1. ahpResults: A table used to log all the output results of AHP.
2. APILogs: A table used to store all the API calls and log them.

3. derivedCalculationLogs: A table used to keep track of all updates to derived values.
4. derivedValues: A table used to store all the derived values of each content areas for each instance type.
5. ec2logs: A table used to store all the log files of EC2 service type.
6. ec2performance: A table used to store all the performance metrics of EC2 service type.
7. ec2summary: A table used to store the summary results of logging while testing an EC2 service type.
8. endpointAccess: A table used to define credentials for permitted logging from registered endpoints.
9. endpointData: A table that contains information about the logging user.
10. endpointLogs: A table that stores the list of files sent by the user to the database.
11. loginLogs: A table that stores all login information for each login by a user.
12. processingRequests: A table that stores the processing requests generated from the front-end and officially stored in the database.
13. saveec2: A table used to store retrieved average values of each instance type of EC2 uploaded for all logging sessions.
14. saves3: A table used to store retrieved average values of each S3 logs file uploaded for all logging sessions.
15. saverds: A table used to store retrieved average values of each instance type of RDS uploaded for all logging sessions.
16. sessionDetails: A table used to store a particular login session details.
17. staticContent: A table used to retrieve static data from the backend to display in the front-end.
18. taskStatus: A table used to store the status of a processing number as Pending or Complete.

#### 5.2.6 Machine Learning

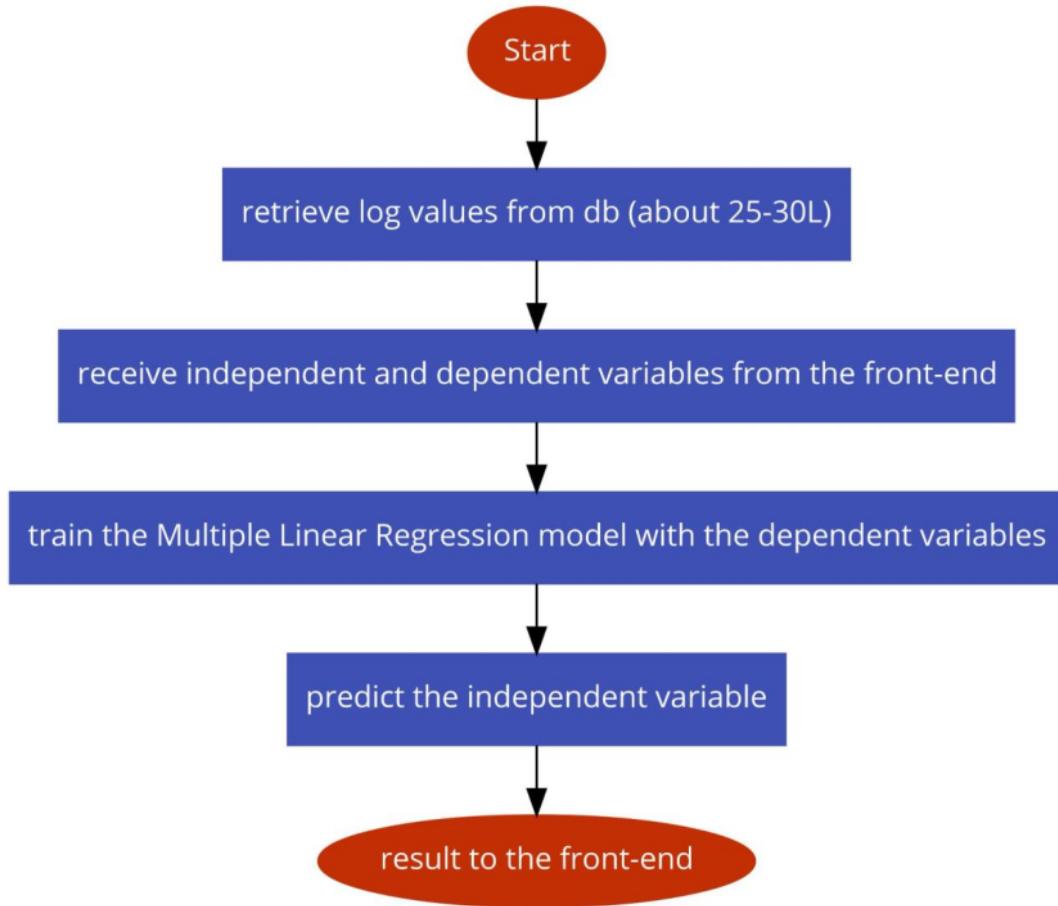


Figure 5.3: Multiple Linear Regression Algorithm

The Machine Learning component is fundamentally a Multiple Linear Regression algorithm that trains about 25-30L log entries that are related to AWS S3. These form rows to the dataset and are split into training and testing just to test the accuracy of the predictability of the model.

Once fit and tested, we saw a 90-95% accuracy as we gradually trained the model with more and more logs from JMeter. These logs help predict the independent variable using the dependent variables as mentioned in the input. Finally, we return the result to the user interface with a payload.

#### 5.2.7 Backend Routes

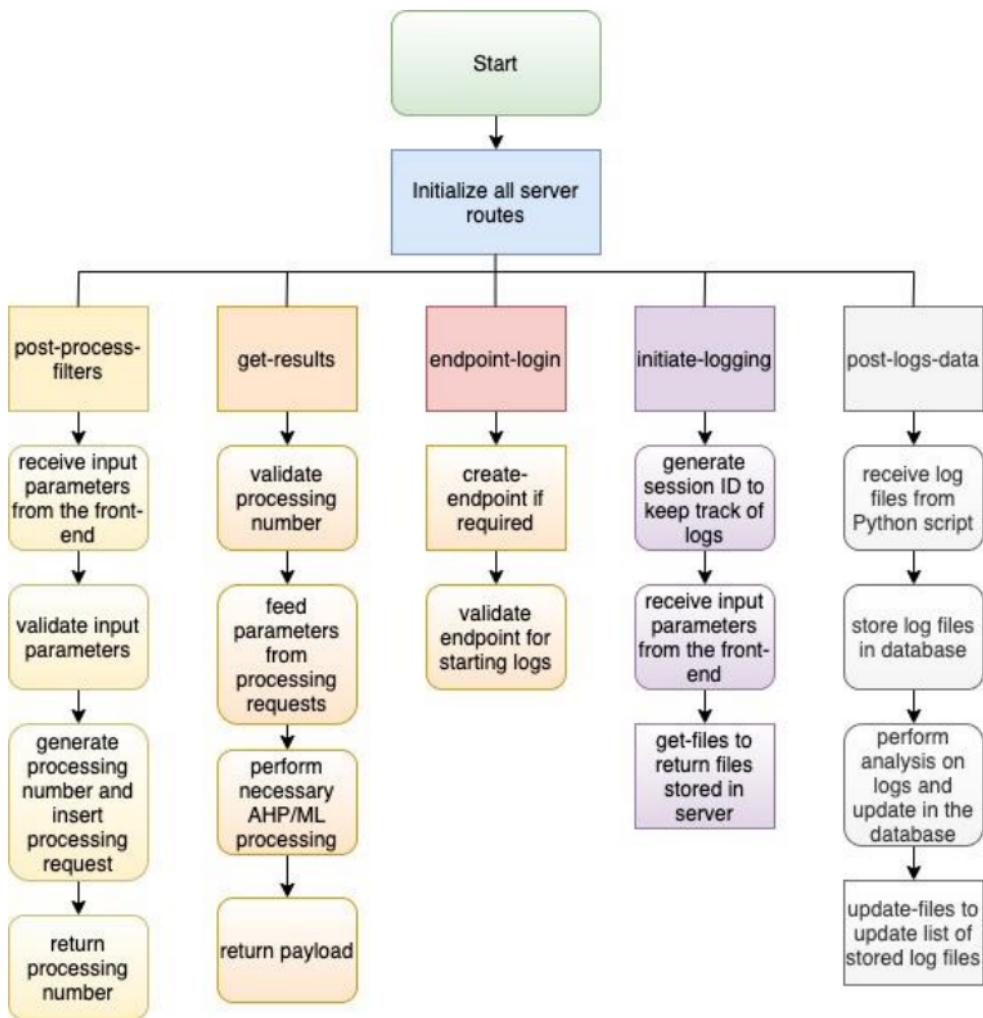


Figure 5.4: Backend Routes

In the backend, there are five primary routes with sub-routes that handle both front-end and endpoints API calls.

1. **post-process-filters:** This route sends the user request with filters, instance types (if any), service type, and priorities. Once these are received and validated, a processing number is mapped to this request and is returned to the user interface.
2. **get-results:** This route matches an entered processing number and performs necessary AHP/ML processing. Once the processing is done, a payload is returned to the user interface.
3. **endpoint-login:** For a registered device to begin uploading logs to the backend, this route handles the requests. Login/Signup is handled and the user can register their device.
4. **Initiate-logging:** For a successful login to the system, the backend identifies the files that were previously uploaded and returns to the endpoint through this route. Once it is retrieved, only new files are logged to the backend.

- post-logs-data: The logs to be uploaded are processed by the endpoints and are sent via this route. Once all the new files are uploaded, the files list is updated so that no same logs are sent to the backend again.

### 5.3 Datasets Collection

Various instances of EC2 and RDS and S3 are tested against several parameters with varying configuration settings for each run of the test script like 50 - 200 User Requests per second, 1 - 3600s of ramp-up time (time interval in which all user requests are invoked non-uniformly) , 30minutes - 1 hour 30 minutes Duration of tests, Loopcount (number of times the same loop of Threads (User Requests) will be invoked. The following steps are followed to create a successful Testing Script :

For EC2 :

- Launch new instances of EC2 from AWS Console and specify the below user data and add Security Roles of SSH (default settings), HTTP (Port 80) and TCP (port 4444).

```
26
#!/bin/bash
yum update -y
yum install -y docker
service docker start
docker pull bkimminich/juice-shop
docke45un -d -p 80:3000 bkimminich/juice-shop
26o yum install -y java-1.8.0-openjdk.x86_64
wget https://github.com/undera/perfmon-agent/releases/download/2.2.3/ServerAgent-2.2.3.zip
unzip ServerAgent-2.2.3.zip
cd ServerAgent-2.2.3
sh startAgent.sh
```

The OWASP- Juice Shop is used as the sample website.

- Create a new test project by clicking on File > New.
- Add a Thread Group, enter necessary details about number of requests, ramp-up period, duration, etc.
- Add an HTTP Request Sampler under the above Thread Group. Specify the URL and path of the website which has to be tested. Select GET in the drop down menu in HTTP Request. Add any parameters if necessary.
- Add View Results in a Table, Summary Report, and Perfmon Metrics Collector Listeners in the sampler.
- The three different listeners will save logs for different parameters.

For RDS :

- Launch RDS instances from AWS Console and grant them public access during configuration before launch.
- <sup>34</sup>Launch an EC2 instance to connect to the databases. Follow the previous steps to launch the instance. Connect to the <sup>34</sup>instance using SSH (on CLI or Putty manager depending on OS in the local system). Execute the following commands:

```
sudo yum update
sudo yum install mysql 34
mysql -h <DB_Host> -P <Port> -u <Username> -p <Password>

create database RDS_Test;
```

```

use RDS_Test

create table student (USN varchar(10),
StudentName varchar(30),
Age int,
Branch varchar(5),
Gender varchar(1),
Semester int,
PRIMARY KEY(USN));

insert into student values ('1DS17CS734', 'Tanusha', 22, 'CSE','F',8);

```

(Populate the table using sample values)

- Download and add the MySQL Connector file in the Lib folder inside the Apache JMeter directory.
- In Jmeter, create a new project.
- Add JDBC Configuration file and enter the Database hostname, port, username and password. Mention JDBC Driver as the Driver File in configuration.
- Add a Thread Group with all necessary details and add JDBC Request Sampler with the SQL query that has to be invoked.
- Add View Results in table and Summary Report Listeners.

For S3:

- ~~86~~ Create the same HTTP Request Test on Jmeter as was done for EC2 instances.
- ~~Create a new bucket and add the source files of the website.~~
- The index.html file is generally the opening file of the website.
- Click ~~51~~ the opening file, copy the URL and enter it on HTTP Request Sampler in Jmeter.
- Add ~~View Results in table~~ and Summary Report ~~Listeners~~.

All ~~the~~ listeners together collect logs (used for training the AHP model in backend) for the following attributes:

- CPU Utilisation: One of the most important Health Parameters is CPU Usage which denotes the amount of work handled by CPU by running or background applications on the Host Server.
- Memory Utilisation: Memory Utilisation is the RAM occupied by the applications at any given point in time on the Host Server.
- Latency: Latency denotes the the amount of time the user request waits for the resources to be delivered through HTTP or TCP Protocols
- Connection Time: This attribute denotes the amount of time required to establish a connection between Client and Server in TCP connection.
- Response Time: This value indicates the amount of time required before the first response is received from the Host Server.
- Standard Deviation: The consistency of the server is denoted by this attribute, it shows the deviation of Response Time of individual requests from the average Response Time.
- Error Percentage: This value denotes the percentage of all threads which do not connect with the host. In other words, the percentage of times when the resource is inaccessible to the Client.

## Testing

In our project, we have imbibed the principles of Functional Testing while carrying out each phases of Acceptance, System, Unit, and Integration Testing.

In the Unit Testing phase, while working with individual components, these were the timelines:

- Check functionality of graphical representation
- Check the unit that retrieves graphs from the backend and displays the same
- Check the functionality of the various sections involved in the UI that catered to static design templates
- Check the functionality of the server's API routing
- Check the functionality of the backend's data movement between one structure to another and also verify data handling
- Check the integration of the database elements with the end-to-end system
- Verify necessary primary key and foreign key requirements and ensure creation of the required tables
- Conduct logging activities with ample testing resources within controlled time intervals
- Extrapolate logs analysis via Endpoints and assimilate required data for processing

In the Integration Testing, these were the goals defined:

- Describe the connectivity between both the ends of the front-end/post-process-filters API call
- Ensure smooth upload of log files between both the ends of the endpoints.post-logs-data API call
- Sustain network connectivity between yielding the processing results between both the ends of the front-end/get-results API call

In the System Testing, a thorough end-to-end round of testing was carried out for 5-7 days to fix routine bugs and common mistakes. We described scenarios and impasse situations and rigorously tested the application's limits.

Finally, in the Acceptance Testing, we formulated all memory, storage, hosting, security, and data reliability factors once the application was released to the world at large. Only after an acceptable range of results were obtained was the entire project deployed to Heroku. We configured dynos for memory, expanded database requirements, made suitable tracks for migration purposes, and prepared routine backups and integration checks on the stored data.

## Experimentation and Results

When we tested the requests for Analytical Hierarchy Process, we observed the following:

- Users are more interested in Overall rankings than individual areas.
- When asked to prioritize content areas, throughput was the most important content area among available options.
- Since at least 2 filters had to be chosen for processing, users were sometimes spoilt for choice in choosing the best two content areas if not more.
- While consideration of instance types available as an option was deemed useful, users mostly preferred a default AHP ranking of all the content areas.
- The processing for AHP with a two-level tree is a complicated procedure. So, with more and more nodes loaded into the tree, the processing increases by two powers.
- Each time user preferences are sent, traversal through the list of combinations is tedious and expensive. However, it is convenient to transact stored data of all filters with one database command and consume values as required by the list of combinations.
- As future scope, we seek to derive all values in one search and add relative values by searching through a stored list. Currently, we are searching for each value in the database. It is computation-intensive and time-intensive to individually run SQL SELECT statements to fetch data monotonously.

While observing the results of Multiple Linear Regression in the model, we could draw the following observations and results:

- Storing 28-30 L logs and counting is not feasible to happen in real-time.
- To track such huge data through unique numbers is also difficult thus rendering primary keys to be ineffective.
- Training huge datasets in limited time is a herculean task. However, there are batch processing and Map-Reduce solutions in the mix.
- An alternative would be to run the training every time after the logs are uploaded and to save and replace the regression model in a stored file.
- With this alternative, preprocessing and training may not occur while user requests for processing.
- Many times, the memory usage exceeded 500 MB just from retrieval and processing of 30 L logs every time an S3 request was made.
- To avoid such leaks, we moved the training component to the endpoints and made the rendering of the model quite feasible.
- The time gap before timeout for a HTTP GET/POST request to receive a response is 30 seconds. Within this timeframe, it proved quite challenging to include training, testing, and prediction of the ML model for a user request.
- Once the training was complete, we returned a payload of the predicted result in graph-compatible format for the user interface.

### Endpoints Results

The logs collected gave insights about the performance of different AWS Services and Instances being subjected to different factors which might affect them like Timestamp, Type of computing - storage intensive, computation intensive, etc.

The screenshot shows the AWS EC2 Instances page with the following details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
-	i-0832dd07599548626	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	ec2-3-108-41-250.ap-s...	3.108.41.250
-	i-0a73cc8c7b648a27b	Running	t2.small	2/2 checks passed	No alarms	ap-south-1a	ec2-13-234-232-37.ap...	13.234.232.37
-	i-01becb4894e9a43a5	Running	t2.nano	2/2 checks passed	No alarms	ap-south-1a	ec2-13-126-114-7.ap...	13.126.114.7
-	i-03967392a39e2af93	Running	t2.xlarge	2/2 checks passed	No alarms	ap-south-1a	ec2-65-2-31-151.ap-so...	65.2.31.151
-	i-0d549d29f1b677d72	Running	t2.medium	2/2 checks passed	No alarms	ap-south-1b	ec2-15-207-112-245.ap...	15.207.112.245
-	i-04e601a9902343434	Running	t2.large	2/2 checks passed	No alarms	ap-south-1b	ec2-52-66-145-121.ap...	52.66.145.121

Figure 7.1: EC2 instances are launched with Perfmon Server Agent installed

The screenshot shows the AWS RDS Databases page with the following details:

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity
test-database	Instance	MySQL Community	ap-south-1a	db.t2.micro	Available	2.41%	0 Connect
test-database-medium	Instance	MySQL Community	ap-south-1a	db.t2.medium	Available	1.67%	0 Sessions
test-database-small	Instance	MySQL Community	ap-south-1b	db.t2.small	Available	2.67%	0 Connect

Figure 7.2: RDS instances launched

Each RDS Database instance contains a Student Table (with Attributes: USN, Student Name, Branch, Gender) populated with sample values. This table is queried while testing on Jmeter.

S3 Bucket created:

S3 bucket is created with boilerplate code of a website. Any sample website can be used. Public access must be given for the bucket to be tested externally through Jmeter.

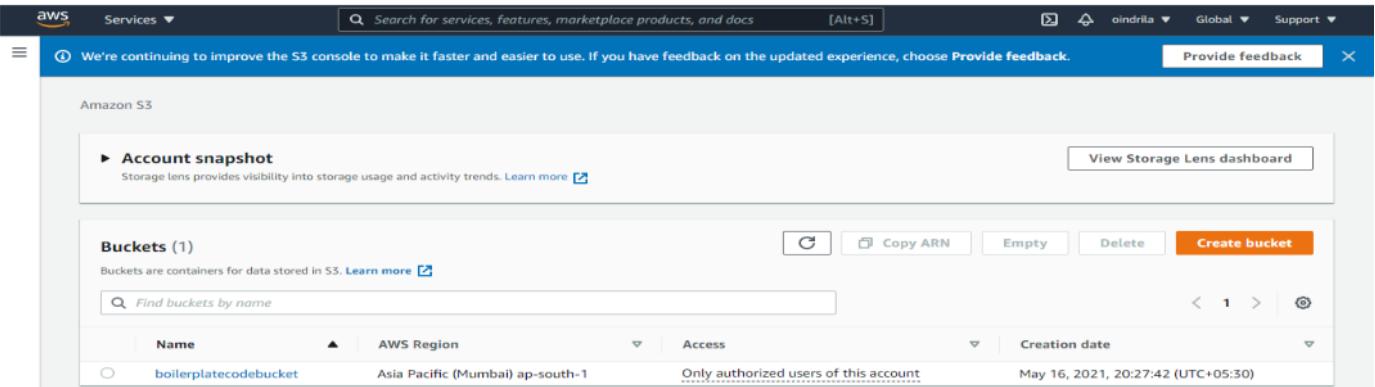


Figure 7.3: S3 Bucket

The public URL of S3 is given as the URL of the website in Jmeter's HTTP Request Sampler.

JMeter:

Testing EC2 instances.

In the test shown below, the following configurations are used :

- Number of Threads : 100
- Ramp-up period : 1s
- Duration : 3600s
- Startup Delay : 1s
- HTTP Request Type : GET Request
- Hostname : ec2-3-108-41-250.ap-south-1.compute.amazonaws.com (for micro, hostname will vary from instance to instance)
- Path : /#
- In View results in table listener, File name can be mentioned in .csv or .txt format to save the results in that file
- Summary Report can be saved after the test is complete by clicking on 'Save Data' at the bottom of the window.
- Perfmon Port : 3306
- Hostname/ IP : 3.108.41.250 (for this specific instance)
- Mention the metrics to be computed in Perfmon Listener (here, CPU and Memory)

Running ec2\_test.jmx :

Figure 7.4: Thread Group Configuration

Figure 7.5: HTTP Request Sampler

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	09:52:48.195	Thread Group ...	HTTP Request	109	✓	2367	151	101	63
2	09:52:47.863	Thread Group ...	HTTP Request	441	✓	2367	151	433	396
3	09:52:48.180	Thread Group ...	HTTP Request	124	✓	2367	151	118	80
4	09:52:48.200	Thread Group ...	HTTP Request	96	✓	2367	151	91	53
5	09:52:48.168	Thread Group ...	HTTP Request	139	✓	2367	151	128	91
6	09:52:48.167	Thread Group ...	HTTP Request	143	✓	2367	151	143	95
7	09:52:48.186	Thread Group ...	HTTP Request	126	✓	2367	151	126	76
8	09:52:48.203	Thread Group ...	HTTP Request	110	✓	2367	151	110	58
9	09:52:48.064	Thread Group ...	HTTP Request	249	✓	2367	151	249	199
10	09:52:48.148	Thread Group ...	HTTP Request	166	✓	2367	151	165	114
11	09:52:48.035	Thread Group ...	HTTP Request	283	✓	2367	151	283	229
12	09:52:48.145	Thread Group ...	HTTP Request	173	✓	2367	151	173	119
13	09:52:48.160	Thread Group ...	HTTP Request	159	✓	2367	151	159	103
14	09:52:48.134	Thread Group ...	HTTP Request	185	✓	2367	151	185	129
15	09:52:48.113	Thread Group ...	HTTP Request	207	✓	2367	151	207	152
16	09:52:48.044	Thread Group ...	HTTP Request	284	✓	2367	151	284	221
17	09:52:48.104	Thread Group ...	HTTP Request	224	✓	2367	151	224	161
18	09:52:48.163	Thread Group ...	HTTP Request	165	✓	2367	151	165	102
19	09:52:48.111	Thread Group ...	HTTP Request	219	✓	2367	151	219	156
20	09:52:47.927	Thread Group ...	HTTP Request	408	✓	2367	151	408	340
21	09:52:48.076	Thread Group ...	HTTP Request	260	✓	2367	151	260	190

Figure 7.6: View Results in Table (.csv file saved as Log File)

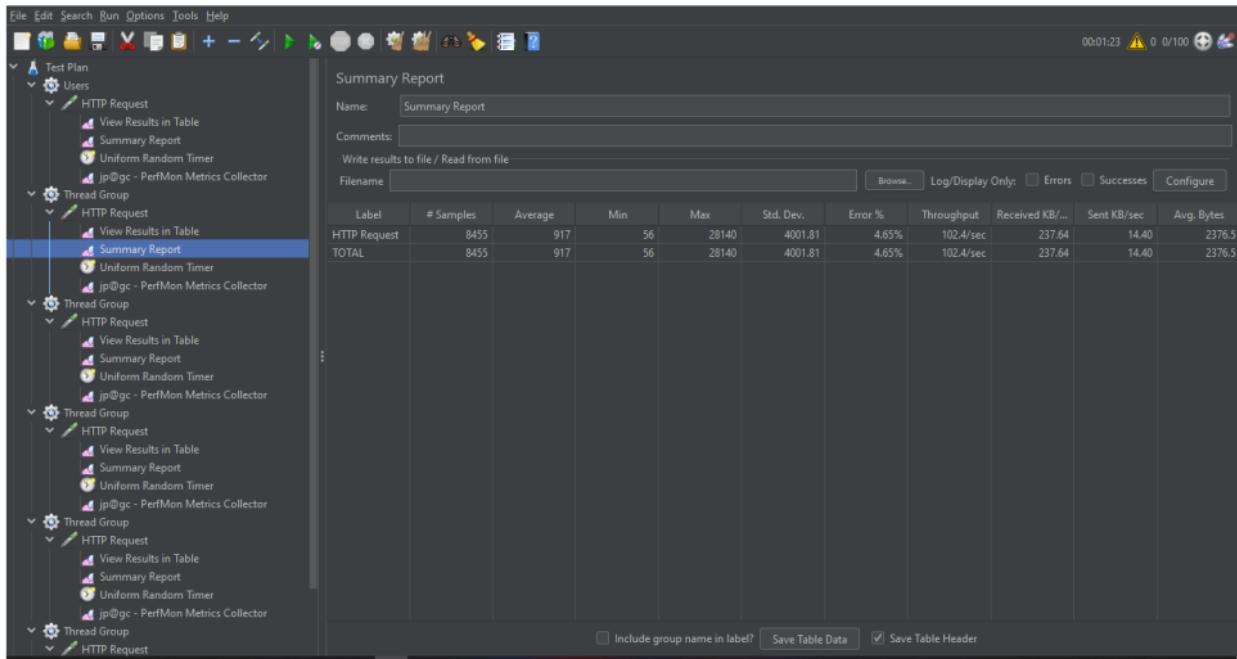


Figure 7.7: Summary Report (.csv file to be saved as Summary File)

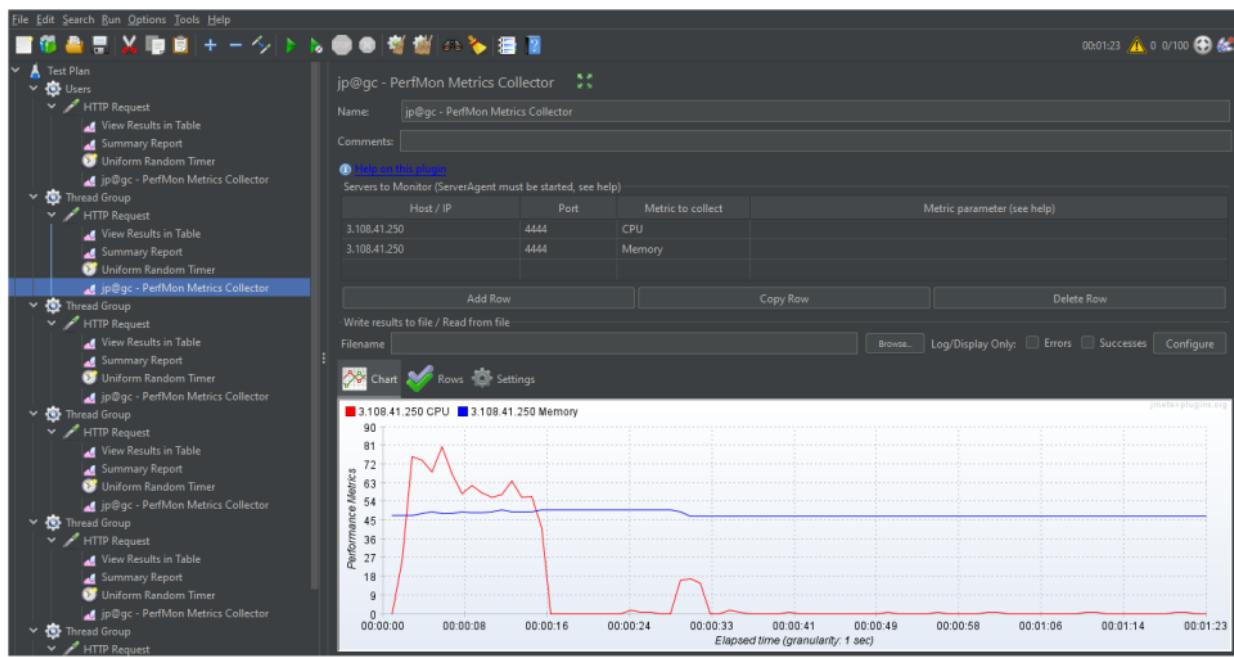


Figure 7.8: Perfmon Metrics Collector (saved as Performance file in .csv format)

Testing RDS instances:

The configurations are as follows:

- Max number of Connection : 50
- Max Wait : 10000 ms
- Time between Eviction Runs: 6000 ms
- Database URL : jdbc:mysql://test-database.cjamazw1gpow.ap-south-1.rds.amazonaws.com:3306/RDS\_Test
- Port : 3306
- Thread groups : 5000
- Ramp-up Time : 1s
- Duration : 600s
- Startup Delay : 0s
- Query Type

Running rds\_test.jmx :

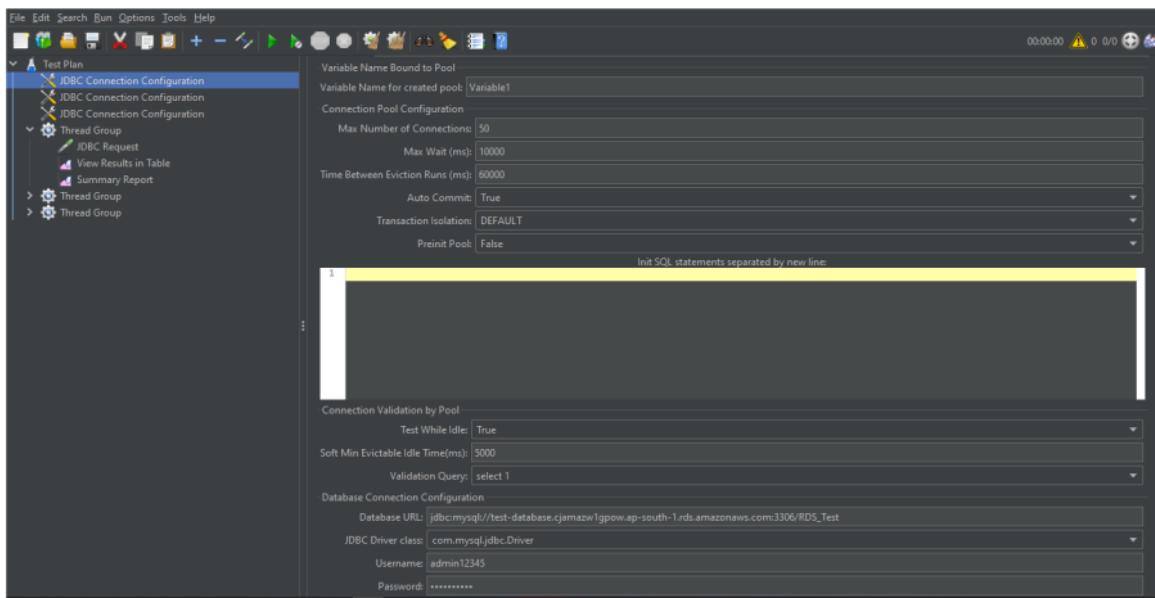


Figure 7.9: JDBC Connection Configuration

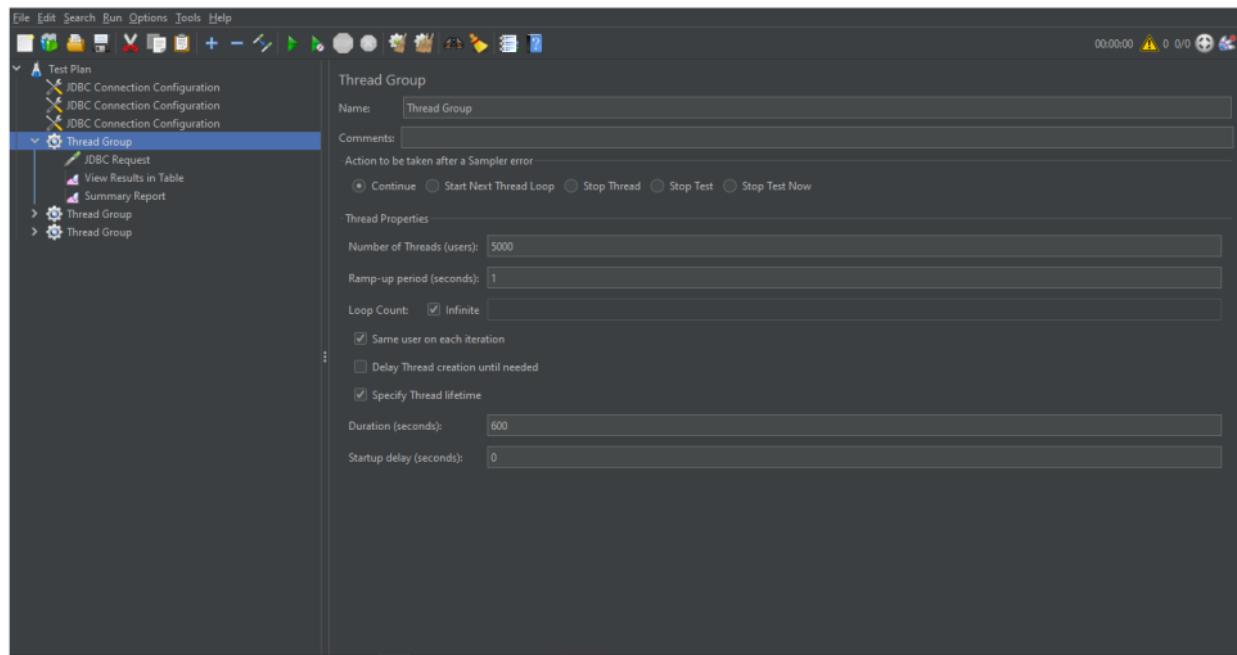


Figure 7.10: Thread Group

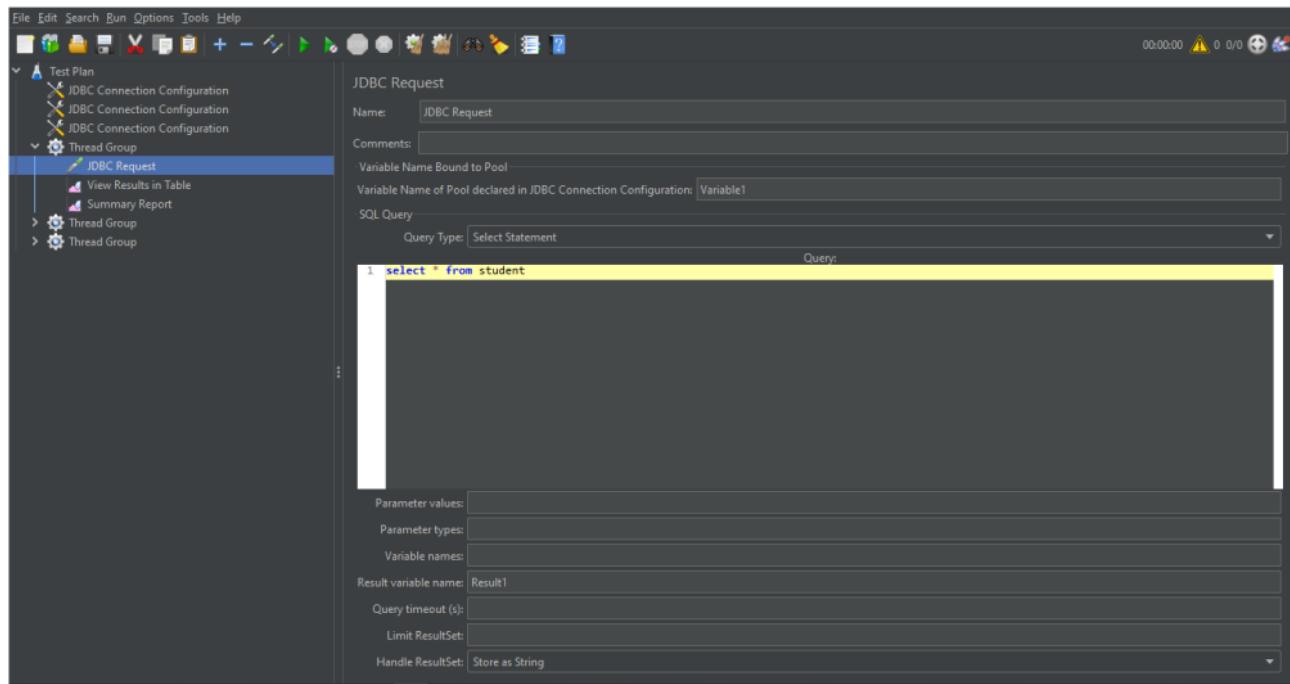


Figure 7.11: JDBC Request

The Results are stored in Log Files and Summary Files like we did for EC2 tests.

Testing S3.

In the test shown below, the following configurations are used :

- Number of Threads : 100
- Ramp-up period : 1s
- Duration : 1800s
- Startup Delay : 1s
- HTTP Request Type : GET Request
- Hostname : boilerplatecodebucket.s3.ap-south-1.amazonaws.com
- Path : /index.html
- In View results in table listener, File name can be mentioned in .csv or .txt format to save the results in that file
- Summary Report can be saved after the test is complete by clicking on ‘Save Data’ at the bottom of the window.

Running S3 test:

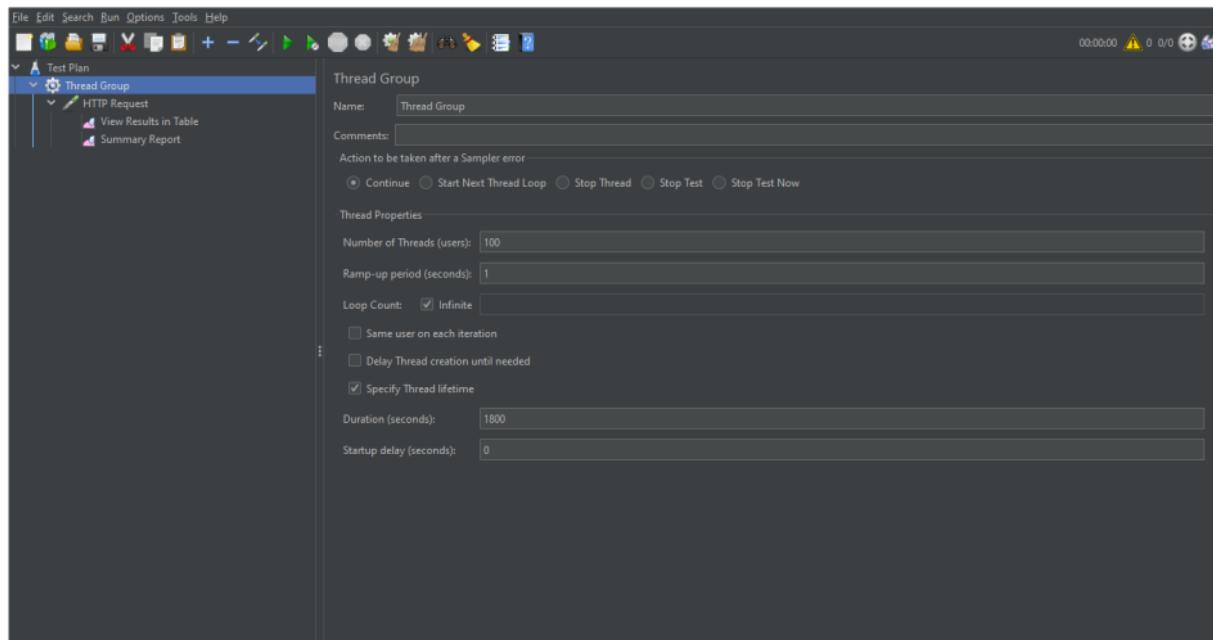


Figure 7.12: Thread Group

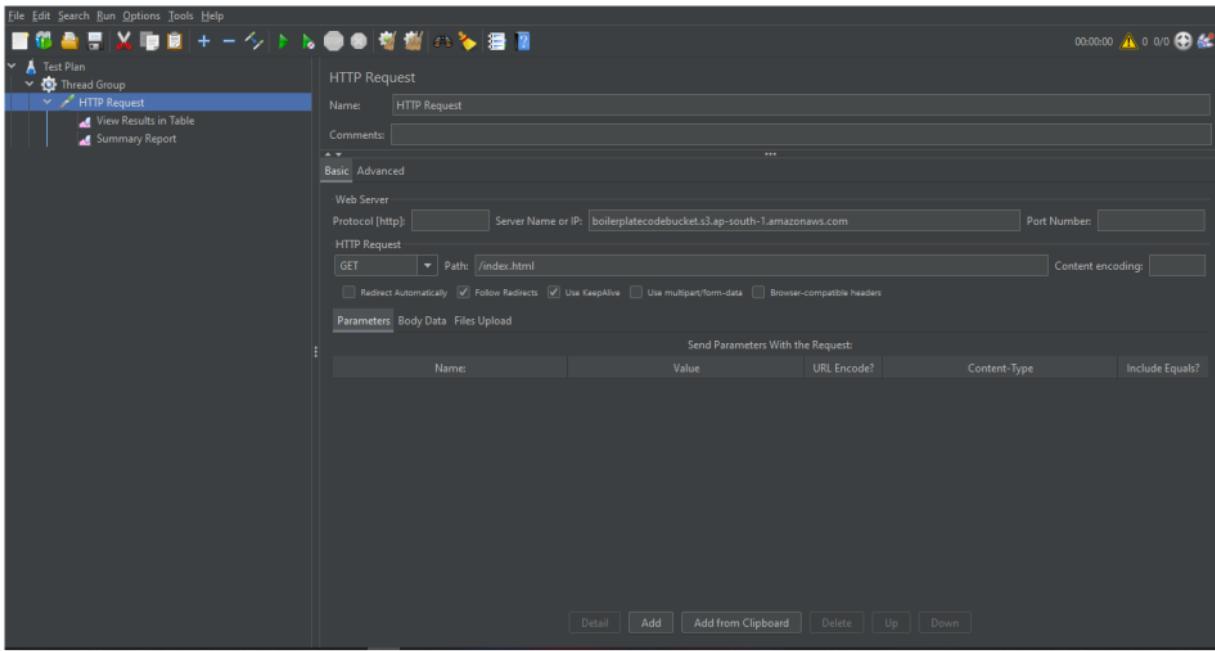


Figure 7.13: HTTP Request

The results are stored in Log Files and Summary Files like we did for EC2 tests. These are further saved in .csv format in local directories. The parent directory is sent to the Backend Database using Python Script.

```
Press 1 for logging in. Press 0 for  
creating/registering an endpoint:
```

```
1
```

```
Enter endpoint ID:
```

```
2
```

```
Enter password:
```

```
[REDACTED]  
Checking login credentials...
```

```
Successfully logged in to Navneeth's  
Laptop!
```

```
If Windows, press 1. If non-Windows,  
press 0.0
```

```
Enter the path where the logs are  
stored:../../logs/
```

```
Your login session ID is:sXeImxQdsQ
```

```
Going through files...
```

```
No new files to be uploaded.
```

Figure 7.14: Endpoints Logs Upload

## Conclusion

- The rapid growth and dependency of the cloud industry will influence risk exposure and potential for impact (e.g. availability, performance, etc.).
- Effective management of service levels (e.g. availability, reliability, performance, security) and financial risk to CSCs is dependent on the CSP's capability and trustworthiness.
- The primary objective of our project will be analysing cloud SLAs and cloud security requirements, assessing and defining CSP trustworthiness levels, and establishing a model for predicting cloud service and SLA availability based on multiple factors (e.g. CSP trust levels, cloud service historic performance and cloud service characteristics).
- Evolving industry standards (e.g. NIST, ISO/IEC, etc) for cloud SLAs and the current research and findings related to work regarding CSP trustworthiness will be used as an addition to analyze CSP cloud computing services, SLA performance and CSP trustworthiness.
- Future enhancements will include an accurate and thorough comparison of several cloud service providers instead of just one.
- Another noteworthy point is to ensure that the logs collected in order for computing the trustworthiness quotient is prodigious. This ensures a comprehensive evaluation of the CSP under scope.
- The list of services offered by each CSP will also be increased so that the user has been given the opportunity to choose from a vast array of choices and specifics.

### Future Works

- The scope of future works in Cloud Service Optimizer will only grow as testing will expand to more instance types of different service types.
- Our team is currently expanding the performance testing on various benchmarks in order to test the application in more areas. The goal is to develop a holistic and all-round analysis of a given cloud service.
- We aim to bring more CSPs into the mix so that more users can get trustworthy data from the standpoint of a cloud user.
- It is a tedious process to train and fit ever-increasing logs. Thus, our model can only sustain as a self-learning model that trains in the background threads. Therefore, this increases the processing efficiency by producing more output in fewer time consumed.
- We learn user inputs and study the type of requests being made with respect to the content areas, the instance types, and the service types. This helps gain insight about the most desired content areas in the cloud marketplace and performance testing can be prioritized on those areas which are trending in order to provide more accurate and reliable results.

## References

- [1] Mahesh K, Dr. M. Laxmaiah, Dr. Yogesh Kumar Sharma, "Predict Trustworthiness of Cloud Services Using Linear Regression Model", IJAST, vol. 29, no. 04, 2020, pp. 5737-5745.
- [2] J. Kanpariyasoontorn and T. Senivongse, "Cloud service trustworthiness assessment based on cloud controls matrix," 2017 19th International Conference on Advanced Communication Technology (ICACT), Bongpyeong, 2017, pp. 291-297.
- [3] S. Pandey and A. K. Daniel, "Fuzzy logic based cloud service trustworthiness model," 2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, 2016, pp. 73-78.
- [4] L. Wang and Z. Wu, "A Trustworthiness Evaluation Framework in Cloud Computing for Service Selection," in 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), Singapore, Singapore, 2014, pp. 101-106.
- [5] Y. Wang, J. Wen, W. Zhou, B. Tao, Q. Wu and Z. Tao, "A Cloud Service Selection Method Based on Trust and User Preference Clustering," in IEEE Access, vol. 7, 2019, pp. 110279-110292.
- [6] C. Jatoh, G. Gangadharan, U. Fiore, and R. Buyya, "Selcloud: a hybrid multi-criteria decision-making model for selection of cloud services, Soft Computing, pp. 1-15, 2018.
- [7] C. Yiming and Z. Yiwei, "Saas vendor selection basing on analytic hierarchy process," in 2011 Fourth International Joint Conference on Computational Sciences and Optimization. IEEE, 2011, pp. 511–515.
- [8] T. L. Saaty, "Decision making with the analytic hierarchy process," International journal of services sciences, vol. 1, no. 1, pp. 83–98, 2008.
- [9] Z. ur Rehman, F. K. Hussain, and O. K. Hussain, "Towards multicriteria cloud service selection," in 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. Ieee, 2011, pp. 44–48.

## Appendix

Source code

### **FRONTEND**

#### **Home.js**

```

20
import { Component } from 'react';
import {Link} from 'react-router-dom'
import './styles/Home.css'
import Cloud from './resources/cloud.png'

29
class Home extends Component {
  render() {
    return(
      <div style = {{height: '100%', width: '100%'}}>
        <div className = 'headerDivStyle'>
          <p className = 'headerStyle'>CLOUD SERVICE OPTIMIZER</p>
          <img src = { Cloud } alt = "Error" className = 'logoStyle' />
        </div>
        <div className = 'linkDivStyle'>
          <Link class = 'linkStyle' to = "/dashboard">Dashboard</Link>
          <Link class = 'linkStyle' to = "/about-us">FAQs</Link>
          <Link class = 'linkStyle' to = "/contact-us">Contact Us</Link>
        </div>
        <div className = 'introDivStyle'>
          <p className = 'introTextStyle'>Cloud Service Selection has progressed as a cloud computing
          paradigm of entrusting good faith in a Cloud Service Provider (CSP) for the services with specifications that are
          legally agreed upon by the involved parties.<br/>The abundance of seemingly similar Cloud Services and the
          enormous range of customized user preferences and requirements have led to difficulty in choosing the optimal CSP.
          Some of the main issues faced by CC are lack of trust in service providers with respect to availability, reliability and
          efficiency of services at their time of need, ambiguity in SLAs, non-compliance with SLA, absence of diversified
          trust elements, and Quality-of-Service guarantee.<br/> Although a number of cloud service e-marketplaces are
          present, users find it very difficult to manually compare the services of different CSPs: especially new users who
          must go through each feature's description separately during Cloud Service Provider selection.<br/><br/> This is
          where <span style = {{color: '#F9CC88'}}>'Cloud Service Optimizer' </span>comes into the picture to make the job
          quick, easy and efficient.</p>
        </div>
      </div>
    );
  }
}

export default Home;

```

#### **Home.css**

```
.linkStyle {
```

```
color: #F9CC88;
background-color: #fdfdfd;
border-radius: 50px;
text-align: center;
font-size: 20px;
text-decoration: none;
font-weight: bold;
margin: 2%;
padding-left: 1%;
padding-right: 1%;
padding-top: 0.7%;
padding-bottom: 0.7%;
font-family: 'Arial Narrow Bold';
margin-top: 10%;
}

a:hover {
    color: orange;
}

.headerStyle {
    margin-top: 2%;
    color: #FDFDFD;
    font-size: 230%;
    align: center;
    font-weight: bold;
    height: 10%;
    text-shadow: 2px 2px 2px #222;
    margin-left: 3%;
}

.headerDivStyle {
    height: 10%;
    align-items: center;
    background-color: #F9CC88;
    box-shadow: 1px 3px 0.5px #9E9E9E50;
    justify-content: center;
    display: flex;
    border-bottom-left-radius: 50px;
    border-bottom-right-radius: 50px;
    margin-bottom: 2%;
}

.li53ivStyle {
    margin-left: 35%;
    margin-top: 1%;
}

.logoStyle {
```

```

color: #FDFDFD;
height: 7%;
width: 7%;
margin-bottom: 1%;
margin-top: 2%;
}

.introDivStyle {
  width: 70%;
  text-align: justify;
  margin-left: auto;
  margin-right: auto;
  margin-top: 7%;  

border-bottom-left-radius: 50px;
border-top-right-radius: 50px;
background-color: #F9CC88;
padding: 3%;  

background: rgba(255,255,255,0.5);
margin-bottom: 7%;
}

.introTextStyle {
  color: #696969;
  font-family: 'Arial Narrow Bold';
  font-weight: bold;
  font-size: 130%;
  margin-left: 7%;
  margin-right: 7%;
  margin-top: 2%;
  margin-bottom: 2%;
}

```

46

## AboutUs.js

```

import { Component } from 'react';
import './styles/AboutUs.css'

class AboutUs extends Component {

  render() {
    return(  

      <div>
        <div className = 'titleDivStyle'>
          <h1 className = 'titleStyle'>FAQs</h1>
        </div>
        <div className = 'qDivStyle'>
          <p className = 'qTitleStyle' style = {{color: '#F9CC88'}}>Answer a few questions on the dashboard to  

get started right now!</p>

```

<p className = 'qTextStyle' style = {{color: '#F9CC88'}}>For EC2 and RDS:</p>  
<p className = 'qTextStyle'>1. What AWS Service are you looking for?</p>  
<p className = 'qTextStyle'>Clicking on the 'Select Service' Button will display the services which are available for comparison and calculation of trustworthy metric. Select the service as per your requirement.</p>  
<p className = 'qTextStyle'>2. What performance metric do you want us to focus on?</p>  
    <p className = 'qTextStyle'>Select from a wide range of available Filters and assign priority based on how important each attribute is for your business requirement. Checking the 'Assign same priority to all filters' will assign same priority to all attributes. Don't forget to select the filters that you want to apply after this step. Individual priorities can also be assigned by unchecking the above checkbox, clicking on each filter and filling the priority numbers (Integer type in range [1, n]: 1 being the highest priority.  $n \leq$  Total number of Filters available) in the textboxes which appear beside them. To deselect any filter, simply click again on the selected one.</p>  
    <p className = 'qTextStyle'>3. What instance types of the above listed AWS service do you want?</p>  
    <p className = 'qTextStyle'>Select the instance types that you want compare between and click on Save. If no instances are selected all of the shown instances will be automatically chosen for Trustworthiness Evaluation.</p>  
    <p className = 'qTextStyle'>After successfully uploading the above data, click on Show Results to assess the trustworthiness value and graphical results of the comparison.</p>  
    <p className = 'qTextStyle' style = {{color: '#F9CC88'}}>For S3:</p>  
    <p className = 'qTextStyle'>1. What AWS Service are you looking for?</p>  
    <p className = 'qTextStyle'>Clicking on the 'Select Service' Button will display the services which are available for comparison and calculation of trustworthy metric. Select S3.</p>  
    <p className = 'qTextStyle'>2. What metric should ML 37 diction be applied to?</p>  
    <p className = 'qTextStyle'>For S3, Users must choose n-1 ( $n$  being the total number of attributes) filters and must provide the values for each of the selected filters. The value of the only 1 remaining filter is predicted. Here, priorities can be positive Integers or Decimal values.</p>  
    <p className = 'qTextStyle'>Allowed values for the selected filters:</p>  
    <p className = 'qTextStyle'>Consistency (stdD 31): 100 to 1000. Here, 100 indicates a low level of Consistency and 1000 indicates a high level of Consistency. Knowing the standard deviation of your data set tells you how densely the data points are clustered around the mean. The smaller the standard deviation, the more consistent the data. </p>  
    <p className = 'qTextStyle'>Error Rate (errorRate): 0 to 100. Here, 0 refers to low error rate and 100 refers to high error rate. It is the percentage value of errors caused due to various reasons when connecting to the backend hosted by a service type. Total percentage of errors found for a particular sample request. 0.0% shows that all requests completed successfully. Total equals the percentage of error samples in all samples.</p>  
    <p className = 'qTextStyle'>Latency (Latency): 0 to 300. Here, 0 refers to a low level of Latency and 300 refers to a high level of Latency. JMeter measures the latency from just before sending the request to just after the first response has been received. Thus, the time includes all the processing needed to assemble the request as well as assembling the first part of the response, which in general will be longer than one byte. Protocol analysers (such as Wireshark) measure the time when bytes are actually sent/received over the interface. The JMeter time should be closer to that which is experienced by a browser or other application client.</p>  
    <p className = 'qTextStyle'>Elapsed Time (elapsed): 0 to 300. Here, 0 refers to low Elapsed time and 300 refers to high Elapsed time. JMeter measures the elapsed time from just before sending the request to just after

the last response has been received. JMeter does not include the time needed to render the response, nor does JMeter process any client code, for example JavaScript.</p>

<p className = 'qTextStyle'>Connection Time (Connect): 0 to 300. Here, 0 refers to the low level of Connection Time and 300 refers to the high level of Connection time. JMeter measures the time it took to establish the connection, including SSL handshake. Note that connect time is not automatically subtracted from latency. In case of connection error, the metric will be equal to the time it took to face the error, for example in case of Timeout, it should be equal to connection timeout.</p>

<p className = 'qTextStyle'>Throughput (Throughput): 0 to 5000. Here, 0 refers to a low level of Throughput and 5000 refers to a high level of Throughput. Hits/sec, or total number of requests per unit of time (sec, mins, hr) sent successfully to server during test.</p>

<p className = 'qTextStyle' style = {{color: '#F9CC88'}}>After selecting the filters, press Save to upload the data. Click on Show Results to assess the trustworthiness value and graphical results of the comparison.</p>

```
    </div>
    </div>
)
};

}
```

```
export default AboutUs;
```

68

### AboutUs.css

```
body {
  width: 100%;
  height: 100%;
  background-color: #e8f4f8;
}

.titleStyle {
  margin-top: 0%;
  font-size: 250%;
  color: #FDFDFD;
  [40]t-shadow: 2px 2px 2px #222;
  text-align: center;
  padding: 2%;
  margin-bottom: 2%;
}

.ti[22]DivStyle {
  background-color: #F9CC88;
  width: 15%;
  margin-left: auto;
  margin-right: auto;
  border-bottom-left-radius: 50px;
  border-bottom-right-radius: 50px;
  padding-bottom: 0.4%;
}
```

```
.qDivStyle {  
    width: 70%;  
    text-align: justify;  
    margin-left: auto;  
    margin-right: auto;  
    margin-top: 3%;  
    border-bottom-left-radius: 50px;  
    border-top-right-radius: 50px;  
    background-color: #F9CC88;  
    padding: 3%;  
    background: rgba(255,255,255,0.5);  
    margin-bottom: 3%;  
}
```

```
.qTitleStyle {  
    color: #696969;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 170%;  
    margin-left: 7%;  
    margin-right: 7%;  
    margin-top: 2%;  
    margin-bottom: 2%;  
}
```

```
.qTextStyle {  
    color: #696969;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    margin-left: 7%;  
    margin-right: 7%;  
    margin-top: 2%;  
    margin-bottom: 2%;  
}
```

## Contact<sup>29</sup>'s

```
import { Component } from 'react';  
import './styles/ContactUs.css'  
  
class ContactUs extends Component {  
    render() {  
        return(  
  
            <div>  
                <div className = 'titleDivStyle1'>  
                    <h1 className = 'titleStyle1'>Contact Us</h1>  
                </div>
```

```

<div className = 'qDivOverallStyle' >
  <p className = 'qTitleStyle1' style = {{color: '#F9CC88', display: 'block'}}>Creators:</p>
  <div className = 'qDivStyle1'>
    <div style = {{width: '27%'}}>
      <p className = 'qTextStyle1' style = {{textAlign: 'right'}}>Navneeth Krishna M</p>
      <p className = 'qTextStyle1' style = {{textAlign: 'right'}}>Nishchala M</p>
      <p className = 'qTextStyle1' style = {{textAlign: 'right'}}>Oindrila Chakraborti</p>
    </div>
    <div>
      <p className = 'qTextStyle1'>navneeth.padaki15@gmail.com</p>
      <p className = 'qTextStyle1'>nishchalamkumar12@gmail.com</p>
      <p className = 'qTextStyle1'>oindrila2411@gmail.com</p>
    </div>
    </div>
  </div>
)>;
}

```

export default ContactUs;

### ContactUs.css

```

.title1 {
  margin-top: 0%;
  font-size: 250%;
  color: #FDFDFD;
  -webkit-box-shadow: 2px 2px 2px #222;
  text-align: center;
  padding: 2%;
  margin-bottom: 2%;
}

.divStyle1 {
  background-color: #F9CC88;
  width: 22%;
  margin-left: auto;
  margin-right: auto;
  border-bottom-left-radius: 50px;
  border-bottom-right-radius: 50px;
  padding-bottom: 0.4%;
}

.qDivStyle1 {
  display: flex;
  text-align: center;
  justify-content: center;
  align-items: center;
}

```

```
.qTitleStyle1 {  
    color: #696969;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 170%;  
    margin-left: 7%;  
    margin-right: 7%;  
    margin-top: 2%;  
    margin-bottom: 2%;  
    text-align: center;  
}  
  
.qTextStyle1 {  
    color: #696969;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    margin-left: 7%;  
    margin-right: 7%;  
    margin-top: 2%;  
    margin-bottom: 2%;  
    text-align: left;  
}  
  
.qDivOverallStyle {  
    width: 70%;  
    text-align: center;  
    justify-content: center;  
    min-items: center;  
    margin-left: auto;  
    margin-right: auto;  
    margin-top: 3%;  
    border-bottom-left-radius: 50px;  
    border-top-right-radius: 50px;  
    background-color: #F9CC88;  
    padding: 3%;  
    background: rgba(255,255,255,0.5);  
    margin-bottom: 3%;  
}
```

20

```
Dashboard.js  
import { Component } from 'react';  
import axios from 'axios'  
import { Route } from 'react-router-dom'  
import './styles/Dashboard.css'  
  
class Dashboard extends Component {
```

```
constructor() {
    super();
    var ec2 = "Amazon EC2";
    var s3 = "Amazon S3";
    var rds = "Amazon RDS";

    var defaultJson = {}
    defaultJson[ec2] = {
        "overall": {
            "filterCompare": {
                "t2.xlarge > t2.large": "72.02",
                "t2.xlarge > t2.medium": "95.27",
                "t2.xlarge > t2.small": "112.5",
                "t2.xlarge > t2.micro": "115.67",
                "t2.xlarge > t2.nano": "131.2",
                "t2.large > t2.medium": "13.51",
                "t2.large > t2.small": "23.53",
                "t2.large > t2.micro": "25.37",
                "t2.large > t2.nano": "34.4",
                "t2.medium > t2.small": "8.82",
                "t2.medium > t2.micro": "10.45",
                "t2.medium > t2.nano": "18.4",
                "t2.small > t2.micro": "1.49",
                "t2.small > t2.nano": "8.8",
                "t2.micro > t2.nano": "7.2"
            },
            "data": {
                "t2.xlarge": 0.289,
                "t2.large": 0.168,
                "t2.medium": 0.148,
                "t2.small": 0.136,
                "t2.micro": 0.134,
                "t2.nano": 0.125
            },
            "displayName": "Overall"
        },
        "stdDev": {
            "filterCompare": {
                "t2.xlarge > t2.micro": "88.59",
                "t2.xlarge > t2.large": "93.79",
                "t2.xlarge > t2.small": "96.5",
                "t2.xlarge > t2.medium": "99.29",
                "t2.micro > t2.large": "2.76",
                "t2.micro > t2.small": "4.2",
                "t2.micro > t2.medium": "5.67",
                "t2.large > t2.small": "1.4",
                "t2.large > t2.medium": "2.84",
                "t2.small > t2.medium": "1.42",
                "t2.medium > t2.medium": "0.0"
            }
        }
    }
}
```

```
        },
        "data": {
            "t2.xlarge": 0.281,
            "t2.micro": 0.149,
            "t2.large": 0.145,
            "t2.small": 0.143,
            "t2.nano": 0.141,
            "t2.medium": 0.141
        },
        "displayName": "Consistency"
    },
    "errorRate": {
        "filterCompare": {
            "t2.micro > t2.small": "3.55",
            "t2.micro > t2.medium": "4.17",
            "t2.micro > t2.xlarge": "14.38",
            "t2.small > t2.medium": "0.6",
            "t2.small > t2.xlarge": "10.46",
            "t2.medium > t2.medium": "0.0",
            "t2.medium > t2.xlarge": "9.8"
        },
        "data": {
            "t2.micro": 0.175,
            "t2.small": 0.169,
            "t2.nano": 0.168,
            "t2.large": 0.168,
            "t2.medium": 0.168,
            "t2.xlarge": 0.153
        },
        "displayName": "Error Rate"
    },
    "Throughput": {
        "filterCompare": {
            "t2.xlarge > t2.large": "53.16",
            "t2.xlarge > t2.small": "56.13",
            "t2.xlarge > t2.medium": "61.33",
            "t2.xlarge > t2.micro": "62.42",
            "t2.xlarge > t2.nano": "65.75",
            "t2.large > t2.small": "1.94",
            "t2.large > t2.medium": "5.33",
            "t2.large > t2.micro": "6.04",
            "t2.large > t2.nano": "8.22",
            "t2.small > t2.medium": "3.33",
            "t2.small > t2.micro": "4.03",
            "t2.small > t2.nano": "6.16",
            "t2.medium > t2.micro": "0.67",
            "t2.medium > t2.nano": "2.74",
            "t2.micro > t2.nano": "2.05"
        },
    }
}
```

```
"data": {
    "t2.xlarge": 0.242,
    "t2.large": 0.158,
    "t2.small": 0.155,
    "t2.medium": 0.15,
    "t2.micro": 0.149,
    "t2.nano": 0.146
},
"displayName": "Throughput"
},
"elapsed": {
    "filterCompare": {
        "t2.xlarge > t2.micro": "64.71",
        "t2.xlarge > t2.small": "65.79",
        "t2.xlarge > t2.medium": "69.13",
        "t2.xlarge > t2.nano": "73.79",
        "t2.micro > t2.small": "0.66",
        "t2.micro > t2.medium": "2.68",
        "t2.micro > t2.nano": "5.52",
        "t2.small > t2.medium": "2.01",
        "t2.small > t2.nano": "4.83",
        "t2.medium > t2.medium": "0.0",
        "t2.medium > t2.nano": "2.76"
    },
    "data": {
        "t2.xlarge": 0.252,
        "t2.micro": 0.153,
        "t2.small": 0.152,
        "t2.large": 0.149,
        "t2.medium": 0.149,
        "t2.nano": 0.145
    },
    "displayName": "Elapsed Time"
},
"Latency": {
    "filterCompare": {
        "t2.xlarge > t2.nano": "111.97",
        "t2.xlarge > t2.small": "115.0",
        "t2.xlarge > t2.large": "116.55",
        "t2.xlarge > t2.medium": "118.12",
        "t2.nano > t2.small": "1.43",
        "t2.nano > t2.large": "2.16",
        "t2.nano > t2.medium": "2.9",
        "t2.small > t2.small": "0.0",
        "t2.small > t2.large": "0.72",
        "t2.small > t2.medium": "1.45",
        "t2.large > t2.medium": "0.72"
    },
    "data": {
```

```
        "t2.xlarge": 0.301,
        "t2.nano": 0.142,
        "t2.micro": 0.14,
        "t2.small": 0.14,
        "t2.large": 0.139,
        "t2.medium": 0.138
    },
    "displayName": "Latency"
},
"Connect": {
    "filterCompare": {
        "t2.xlarge > t2.micro": "61.69",
        "t2.xlarge > t2.small": "63.82",
        "t2.xlarge > t2.medium": "67.11",
        "t2.xlarge > t2.nano": "69.39",
        "t2.micro > t2.small": "1.32",
        "t2.micro > t2.medium": "3.36",
        "t2.micro > t2.nano": "4.76",
        "t2.small > t2.medium": "2.01",
        "t2.small > t2.nano": "3.4",
        "t2.medium > t2.medium": "0.0",
        "t2.medium > t2.nano": "1.36"
    },
    "data": {
        "t2.xlarge": 0.249,
        "t2.micro": 0.154,
        "t2.small": 0.152,
        "t2.large": 0.149,
        "t2.medium": 0.149,
        "t2.nano": 0.147
    },
    "displayName": "Connection Time"
},
"CPU": {
    "filterCompare": {
        "t2.xlarge > t2.large": "83.25",
        "t2.xlarge > t2.medium": "114.72",
        "t2.xlarge > t2.small": "221.1",
        "t2.xlarge > t2.micro": "243.14",
        "t2.xlarge > t2.nano": "306.98",
        "t2.large > t2.medium": "17.18",
        "t2.large > t2.small": "75.23",
        "t2.large > t2.micro": "87.25",
        "t2.large > t2.nano": "122.09",
        "t2.medium > t2.small": "49.54",
        "t2.medium > t2.micro": "59.8",
        "t2.medium > t2.nano": "89.53",
        "t2.small > t2.micro": "6.86",
        "t2.small > t2.nano": "26.74",
    }
}
```

```
        "t2.micro > t2.nano": "18.6"
    },
    "data": {
        "t2.xlarge": 0.35,
        "t2.large": 0.191,
        "t2.medium": 0.163,
        "t2.small": 0.109,
        "t2.micro": 0.102,
        "t2.nano": 0.086
    },
    "displayName": "CPU Utilization"
},
"memory": {
    "filterCompare": {
        "t2.xlarge > t2.large": "96.37",
        "t2.xlarge > t2.medium": "289.6",
        "t2.xlarge > t2.small": "616.18",
        "t2.xlarge > t2.micro": "958.7",
        "t2.xlarge > t2.nano": "1773.08",
        "t2.large > t2.medium": "98.4",
        "t2.large > t2.small": "264.71",
        "t2.large > t2.micro": "439.13",
        "t2.large > t2.nano": "853.85",
        "t2.medium > t2.small": "83.82",
        "t2.medium > t2.micro": "171.74",
        "t2.medium > t2.nano": "380.77",
        "t2.small > t2.micro": "47.83",
        "t2.small > t2.nano": "161.54",
        "t2.micro > t2.nano": "76.92"
    },
    "data": {
        "t2.xlarge": 0.487,
        "t2.large": 0.248,
        "t2.medium": 0.125,
        "t2.small": 0.068,
        "t2.micro": 0.046,
        "t2.nano": 0.026
    },
    "displayName": "memory"
}
};

defaultJson[s3] = {
    "predictedValues": {
        "Consistency": "425.82"
    },
    "selectedValues": {
        "Connection Time": 24.0,
        "Elapsed Time": 2116.0,
        "Error Rate": 83.0,
        "Latency": 10.0
    }
};
```

```
        "Latency": 28.0,
        "Throughput": 36.0
    }
};

defaultJson[rds] = {
    "Connect": {
        "data": {
            "db.t2.medium": 0.34,
            "db.t2.micro": 0.32,
            "db.t2.small": 0.339
        },
        "displayName": "Connection Time",
        "filterCompare": {
            "db.t2.medium > db.t2.micro": "6.25",
            "db.t2.medium > db.t2.small": "0.29",
            "db.t2.small > db.t2.micro": "5.94"
        }
    },
    "Latency": {
        "data": {
            "db.t2.medium": 0.311,
            "db.t2.micro": 0.392,
            "db.t2.small": 0.297
        },
        "displayName": "Latency",
        "filterCompare": {
            "db.t2.medium > db.t2.small": "4.71",
            "db.t2.micro > db.t2.medium": "26.05",
            "db.t2.micro > db.t2.small": "31.99"
        }
    },
    "Throughput": {
        "data": {
            "db.t2.medium": 0.325,
            "db.t2.micro": 0.345,
            "db.t2.small": 0.329
        },
        "displayName": "Throughput",
        "filterCompare": {
            "db.t2.micro > db.t2.medium": "6.15",
            "db.t2.micro > db.t2.small": "4.86",
            "db.t2.small > db.t2.medium": "1.23"
        }
    },
    "elapsed": {
        "data": {
            "db.t2.medium": 0.341,
            "db.t2.micro": 0.321,
            "db.t2.small": 0.338
        }
    }
};
```

```
        },
        "displayName": "Elapsed Time",
        "filterCompare": {
            "db.t2.medium > db.t2.micro": "6.23",
            "db.t2.medium > db.t2.small": "0.89",
            "db.t2.small > db.t2.micro": "5.3"
        }
    },
    "errorRate": {
        "data": {
            "db.t2.medium": 0.284,
            "db.t2.micro": 0.403,
            "db.t2.small": 0.313
        },
        "displayName": "Error Rate",
        "filterCompare": {
            "db.t2.micro > db.t2.medium": "41.9",
            "db.t2.micro > db.t2.small": "28.75",
            "db.t2.small > db.t2.medium": "10.21"
        }
    },
    "overall": {
        "data": {
            "db.t2.medium": 0.315,
            "db.t2.micro": 0.361,
            "db.t2.small": 0.325
        },
        "displayName": "Overall",
        "filterCompare": {
            "db.t2.micro > db.t2.medium": "14.6",
            "db.t2.micro > db.t2.small": "11.08",
            "db.t2.small > db.t2.medium": "3.17"
        }
    },
    "stdDev": {
        "data": {
            "db.t2.medium": 0.284,
            "db.t2.micro": 0.383,
            "db.t2.small": 0.333
        },
        "displayName": "Consistency",
        "filterCompare": {
            "db.t2.micro > db.t2.medium": "34.86",
            "db.t2.micro > db.t2.small": "15.02",
            "db.t2.small > db.t2.medium": "17.25"
        }
    }
};

};
```

```

this.ec2 = ec2;
this.s3 = s3;
this.rds = rds;
this.defaultResponse = defaultJson;

this.state = {
  showHideServices: false, showHideFilters: false, showHideInstances: false,
  selectEC2: false, selectS3: false, selectRDS: false,
  selectConsistency: false, selectErrorRate: false, selectLatency: false, selectElapsedTime: false,
  selectConnectionTime: false, selectThroughput: false, selectCpu: false, selectMemory: false,
  checkbox: true, filterAlert: "", serviceAlert: "", instanceAlert: "",
  selectT2Micro: false, selectT2Nano: false, selectT2Small: false, selectT2Medium: false, selectT2Large: false,
  selectT2XLarge: false,
  calculateState: 0, response: {}
};

43
this.hideComponent = this.hideComponent.bind(this);
this.handleClick = this.handleClick.bind(this);
this.handleCalculate = this.handleCalculate.bind(this);
this.calculateButton = this.calculateButton.bind(this);
}

calculateButton() {
  switch(this.state.calculateState) {
    case 0:
      return(
        <div className = 'calculateDivStyle'>
          <button className = 'calculateButtonStyle' onClick={() => {
            this.handleCalculate();
          }}>Save</button>
        </div>
      );
    case 1:
      return(
        <div className = 'calculateDivStyle'>
          <p className='alertStyle'>Processing...</p>
        </div>
      );
    case 2:
      return(
        <Route render={({ history }) => (
          <div className = 'calculateDivStyle'>
            <button className = 'calculateButtonStyle' onClick={() => {
              history.push({
                pathname: '/results',
                state: { data: this.state.response, service: this.getService() }
              })
            }}>Show Results</button>
        </div>
      )}
  )
}

```

```

        </div>
    )}/>
);
case 3:
return(
<div className = 'calculateDivStyle'>
<button className = 'calculateButtonStyle' onClick={() => {
    this.handleCalculate();
}}>Save</button>
<p className='lastAlertStyle'>Processing failed. Try again.</p>
</div>
);
default: console.log("Invalid calculate button state");
}

}

getService() {
if(this.state.selectEC2 === true)
    return this.ec2;
else if(this.state.selectS3 === true)
    return this.s3;
else if(this.state.selectRDS === true)
    return this.rds;

}

validateEntry(checkboxVal, lowerLimit, upperLimit) {
var reg = new RegExp('^\d+$');
var regS3 = new RegExp('^\d+(\.\d+)?$');
var otherLimit;

if(this.state.selectEC2 === true)
    otherLimit = 8;
else if(this.state.selectRDS === true)
    otherLimit = 6;
if(this.state.selectS3 === true)
    return regS3.test(checkboxVal) && parseFloat(checkboxVal) >= lowerLimit && parseFloat(checkboxVal)
<= upperLimit;
else
    return reg.test(checkboxVal) && parseInt(checkboxVal) > 0 && parseInt(checkboxVal) <= otherLimit;
}

hideComponent(name) {
switch (name) {
case "showHideServices":
this.setState({ showHideServices: !this.state.showHideServices });
break;
case "showHideFilters":
}
}

```

```

        this.setState({ showHideFilters: !this.state.showHideFilters });
        break;
    case "showHideInstances":
        this.setState({ showHideInstances: !this.state.showHideInstances });
        break;
    default: console.log("Invalid show and hide scenario");
}
}

handleClick(name) {
    switch(name) {
        case "EC2":
            this.setState({
                selectEC2: !this.state.selectEC2, selectS3: false, selectRDS: false});
            break;
        case "S3":
            this.setState({ selectS3: !this.state.selectS3, selectEC2: false, selectRDS: false });
            break;
        case "RDS":
            this.setState({ selectRDS: !this.state.selectRDS, selectEC2: false, selectS3: false });
            break;
        case "Consistency":
            this.setState({ selectConsistency: !this.state.selectConsistency });
            break;
        case "Error Rate":
            this.setState({ selectErrorRate: !this.state.selectErrorRate });
            break;
        case "Latency":
            this.setState({ selectLatency: !this.state.selectLatency });
            break;
        case "Elapsed Time":
            this.setState({ selectElapsedTime: !this.state.selectElapsedTime });
            break;
        case "Connection Time":18
            this.setState({ selectConnectionTime: !this.state.selectConnectionTime });
            break;
        case "Throughput":
            this.setState({ selectThroughput: !this.state.selectThroughput });
            break;
        case "CPU Utilization":18
            this.setState({ selectCpu: !this.state.selectCpu });
            break;
        case "Memory":
            this.setState({ selectMemory: !this.state.selectMemory });
            break;
        case "T2Nano":
            this.setState({ selectT2Nano: !this.state.selectT2Nano });
            break;
        case "T2Micro":
    }
}

```

```

        this.setState({ selectT2Micro: !this.state.selectT2Micro });
        break;
    case "T2Small":18
        this.setState({ selectT2Small: !this.state.selectT2Small });
        break;
    case "T2Medium":
        this.setState({ selectT2Medium: !this.state.selectT2Medium });
        break;
    case "T2Large":
        this.setState({ selectT2Large: !this.state.selectT2Large });
        break;
    case "T2XLarge":
        this.setState({ selectT2XLarge: !this.state.selectT2XLarge });
        break;
    default: console.log("null");
}
}

handleCalculate() {

    this.setState({calculateState: 1});
    var timestamp;
    var filters = "";
    var priorities = "";
    var serviceType = "";
    var instanceType = "";
    if(this.state.selectS3==true && this.state.selectRDS==true && this.state.selectEC2==true)
    { this.setState({ serviceAlert: "Please select at least one service.", calculateState: 0 }); return; }
    else
        this.setState({ serviceAlert: "Service type noted!" });
    if(this.state.selectEC2==true)
    { var ec2 = 0;
        serviceType = "ec2";
        if(this.state.selectT2Micro==true)
            { instanceType += "t2.micro,"; ec2 = ec2 + 1; }
        if(this.state.selectT2Nano==true)
            { instanceType += "t2.nano,"; ec2 = ec2 + 1; }
        if(this.state.selectT2Small==true)
            { instanceType += "t2.small,"; ec2 = ec2 + 1; }
        if(this.state.selectT2Medium==true)
            { instanceType += "t2.medium,"; ec2 = ec2 + 1; }
        if(this.state.selectT2Large==true)
            { instanceType += "t2.large,"; ec2 = ec2 + 1; }
        if(this.state.selectT2XLarge==true)
            { instanceType += "t2.xlarge,"; ec2 = ec2 + 1; }
        if(this.state.selectT2Micro==false && this.state.selectT2Nano==false &&
this.state.selectT2Small==false
            && this.state.selectT2Medium==false && this.state.selectT2Large==false &&
this.state.selectT2XLarge==false)
    }
}

```

```

{
    instanceType = "t2.nano,t2.micro,t2.small,t2.medium,t2.large,t2.xlarge,";
    ec2 = 0;
    this.setState({ instanceAlert: "" });
}
if(ec2 === 1) {
    this.setState({ instanceAlert: "Please select two or more instance types for comparison.", calculateState: 0
});
    return;
}
else
    this.setState({ instanceAlert: "Instance types noted!" });
}
else if(this.state.selectS3==true) {
    this.setState({ selectCpu: false, selectMemory: false });
    serviceType = "s3";
    instanceType = "na,";
}
else if(this.state.selectRDS==true)
{
    this.setState({ selectCpu: false, selectMemory: false, selectT2Large: false, selectT2Nano: false,
selectT2XLarge: false });
    var rds = 0;
    serviceType = "rds";
    if(this.state.selectT2Micro==true)
    { instanceType += "db.t2.micro,"; rds = rds + 1; }
    if(this.state.selectT2Small==true)
    { instanceType += "db.t2.small,"; rds = rds + 1; }
    if(this.state.selectT2Medium==true)
    { instanceType += "db.t2.medium,"; rds = rds + 1; }
    if(this.state.selectT2Micro==false && this.state.selectT2Small==false&&
this.state.selectT2Medium==false)
    {
        instanceType = "db.t2.micro,db.t2.small,db.t2.medium,";
        rds = 0;
        this.setState({ instanceAlert: "" });
    }
    if(rds === 1) {
        this.setState({ instanceAlert: "Please select two or more instance types for comparison.", calculateState: 0
});
        return;
    }
    else
        this.setState({ instanceAlert: "Instance types noted!" });
}
//Filter processing begins

```

```

if(this.state.selectS3===true)
{
    if(this.state.selectConsistency===true && this.state.selectErrorRate===true &&
this.state.selectLatency===true
        && this.state.selectElapsedTime===true && this.state.selectConnectionTime===true &&
this.state.selectThroughput===true)
    {
        this.setState({ filterAlert: "Please select all filters except the one that needs to be predicted.", calculateState: 0 });
        return;
    }
72
    if((document.getElementById("checkbox").checked===true ||
document.getElementById("checkbox").checked===false) &&
        (this.state.selectConsistency===false && this.state.selectErrorRate===false &&
this.state.selectLatency===false
        && this.state.selectElapsedTime===false && this.state.selectConnectionTime===false &&
this.state.selectThroughput===false
        && this.state.selectCpu===false && this.state.selectMemory===false))
    {
        if(this.state.selectS3===true)
        {
            this.setState({ filterAlert: "Please select all filters except the one that needs to be predicted.", calculateState: 0 });
            return;
        }
        else
        {
            this.setState({ filterAlert: "Please select at least one filter.", calculateState: 0 });
            return;
        }
    }
else if (document.getElementById("checkbox").checked===true && (this.state.selectConsistency===true
|| this.state.selectErrorRate===true || this.state.selectLatency===true || this.state.selectElapsedTime===true
|| this.state.selectConnectionTime===true || this.state.selectThroughput===true || this.state.selectCpu===true
|| this.state.selectMemory===true))
{
    if (this.state.selectConsistency===true) { filters += "stdDev,"; priorities += "1," }
    if (this.state.selectErrorRate===true) { filters += "errorRate,"; priorities += "1," }
    if (this.state.selectLatency===true) { filters += "Latency,"; priorities += "1," }
    if (this.state.selectElapsedTime===true) { filters += "elapsed,"; priorities += "1," }
    if (this.state.selectConnectionTime===true) { filters += "Connect,"; priorities += "1," }
    if (this.state.selectThroughput===true) { filters += "Throughput,"; priorities += "1," }
    if (this.state.selectCpu===true) { filters += "CPU,"; priorities += "1," }
    if (this.state.selectMemory===true) { filters += "Memory,"; priorities += "1," }
    this.setState({ filterAlert: "Filters noted!" });
}
else if (document.getElementById("checkbox").checked===false && (this.state.selectConsistency===true
|| this.state.selectErrorRate===true || this.state.selectLatency===true || this.state.selectElapsedTime===true
|| this.state.selectConnectionTime===true || this.state.selectThroughput===true || this.state.selectCpu===true
|| this.state.selectMemory===true))
{
    if (this.state.selectConsistency===true) { filters += "stdDev,"; priorities += "1," }
    if (this.state.selectErrorRate===true) { filters += "errorRate,"; priorities += "1," }
    if (this.state.selectLatency===true) { filters += "Latency,"; priorities += "1," }
    if (this.state.selectElapsedTime===true) { filters += "elapsed,"; priorities += "1," }
    if (this.state.selectConnectionTime===true) { filters += "Connect,"; priorities += "1," }
    if (this.state.selectThroughput===true) { filters += "Throughput,"; priorities += "1," }
    if (this.state.selectCpu===true) { filters += "CPU,"; priorities += "1," }
    if (this.state.selectMemory===true) { filters += "Memory,"; priorities += "1," }
    this.setState({ filterAlert: "Filters noted!" });
}

```

```

    || this.state.selectConnectionTime==true || this.state.selectThroughput==true || this.state.selectCpu==true
    || this.state.selectMemory==true))
{
    var successText = "Filters with priorities noted!";
    var failureText = "For Amazon EC2, priorities should range from 1 - 8 and for Amazon RDS, priorities
should range from 1 - 6 only.";
    var failureTextS3 = "For Amazon S3, priorities can be double / integer values ranging from ";
    var consistencyLimit = 1000, errorLimit = 100, latencyLimit = 300, elapsedLimit = 300, connectionLimit =
300, throughputLimit = 5000,
    consistencyLowerLimit = 100, lowerLimit = 0;
    var s3FilterCount = 0;

    if (this.state.selectConsistency==true) {
        if(this.validateEntry(document.getElementById("PrConsistency").value, consistencyLowerLimit,
consistencyLimit)) {
            filters += "stdDev,";
            s3FilterCount++;
            priorities += document.getElementById("PrConsistency").value+ ",";
        }
        else {
            if(this.state.selectS3 === true)
                this.setState({ filterAlert: failureTextS3 + consistencyLowerLimit + " - " + consistencyLimit + " only
for Consistency.", calculateState: 0 });
            else
                this.setState({ filterAlert: failureText, calculateState: 0 });
            return;
        }
    }
    if (this.state.selectErrorRate==true) {
        if(this.validateEntry(document.getElementById("PrError").value, lowerLimit, errorLimit)) {
            filters += "errorRate,";
            s3FilterCount++;
            priorities += document.getElementById("PrError").value+ ",";
        }
        else {
            if(this.state.selectS3 === true)
                this.setState({ filterAlert: failureTextS3 + lowerLimit + " - " + errorLimit + " only for Error Rate.",,
calculateState: 0 });
            else
                this.setState({ filterAlert: failureText, calculateState: 0 });
            return;
        }
    }
    if (this.state.selectLatency==true) {
        if(this.validateEntry(document.getElementById("PrLatency").value, lowerLimit, latencyLimit)) {
            filters += "Latency,";
            s3FilterCount++;
            priorities += document.getElementById("PrLatency").value+ ",";
        }
    }
}

```

```

        else {
            if(this.state.selectS3 === true)
                this.setState({ filterAlert: failureTextS3 + lowerLimit + " - " + latencyLimit + " only for Latency.", calculateState: 0 });
            else
                this.setState({ filterAlert: failureText, calculateState: 0 });
            return;
        }
    }

    if (this.state.selectElapsedTime === true) {
        if(this.validateEntry(document.getElementById("PrElapsed").value, lowerLimit, elapsedLimit)) {
            filters += "elapsed,";
            s3FilterCount++;
            priorities += document.getElementById("PrElapsed").value + ",";
        } else {
            if(this.state.selectS3 === true)
                this.setState({ filterAlert: failureTextS3 + lowerLimit + " - " + elapsedLimit + " only for Elapsed Time.", calculateState: 0 });
            else
                this.setState({ filterAlert: failureText, calculateState: 0 });
            return;
        }
    }

    if (this.state.selectConnectionTime === true) {
        if(this.validateEntry(document.getElementById("PrConnection").value, lowerLimit, connectionLimit)) {
            filters += "Connect,";
            s3FilterCount++;
            priorities += document.getElementById("PrConnection").value + ",";
        } else {
            if(this.state.selectS3 === true)
                this.setState({ filterAlert: failureTextS3 + lowerLimit + " - " + connectionLimit + " only for Connection Time.", calculateState: 0 });
            else
                this.setState({ filterAlert: failureText, calculateState: 0 });
            return;
        }
    }

    if (this.state.selectThroughput === true) {
        if(this.validateEntry(document.getElementById("PrThroughput").value, lowerLimit, throughputLimit)) {
            filters += "Throughput,";
            s3FilterCount++;
            priorities += document.getElementById("PrThroughput").value + ",";
        } else {
            if(this.state.selectS3 === true)
                this.setState({ filterAlert: failureTextS3 + lowerLimit + " - " + throughputLimit + " only for Throughput.", calculateState: 0 });
            else

```

```

        this.setState({ filterAlert: failureText, calculateState: 0 });
        return;
    }
}
if (this.state.selectCpu==true) {
    if(this.validateEntry(document.getElementById("PrCpu").value, 0, 0)) {
        filters += "CPU,";
        priorities += document.getElementById("PrCpu").value+ ",";
    }
    else {
        this.setState({ filterAlert: failureText, calculateState: 0 });
        return;
    }
}
if (this.state.selectMemory==true) {
    if(this.validateEntry(document.getElementById("PrMemory").value, 0, 0)) {
        filters += "Memory,";
        priorities += document.getElementById("PrMemory").value+ ",";
    }
    else {
        this.setState({ filterAlert: failureText, calculateState: 0 });
        return;
    }
}
if(this.state.selectS3==true)
{
    if(s3FilterCount !== 5)
    {
        this.setState({ filterAlert: "Please select all filters except the one that needs to be predicted.", calculateState: 0 });
        return;
    }
}

this.setState({ filterAlert: successText });
}

var moment = require('moment');
timestamp = moment(Date.now()).format("DD-MM-YYYY h:mm:ss");

var frontendData = {
    'timestamp': timestamp,
    'filters': filters.substring(0, filters.length - 1),
    'priorities': priorities.substring(0, priorities.length - 1),
    'instanceType': instanceType.substring(0, instanceType.length - 1),
    'serviceType': serviceType
}
frontendData = JSON.stringify(frontendData);
console.log("FrontEndData", frontendData);

```

```

        this.getResults(frontendData, 3, 3000)
    }

    getResults(frontendData, retries, backoff) {

        const postDataUrl = "https://cloud-service-optimizer-dev.herokuapp.com/front-end/post-process-filters/";
        const getDataUrl = "https://cloud-service-optimizer-dev.herokuapp.com/front-end/get-
results?processingNumber=";

        axios({
            method: 'post',
            url: postDataUrl,
            headers: {
                'Content-type': 'application/json',
            },
            data: frontendData
        })
        .then(postJsonData => {
            console.log("pNo. obtained via POST", postJsonData.data);

            var postResponse = postJsonData.data

            setTimeout(() => {
                console.log("Initiating a delay of.", backoff);
                axios.get(getDataUrl + postResponse.message)
                .then(getJsonData => {
                    console.log("GET Response (payload)", getJsonData);

                    var getResponse = getJsonData.data;

                    if(getResponse.status === "success") {
                        this.setState({calculateState: 2, response: getResponse.payload})
                    }
                    else {
                        this.setState({calculateState: 2, response: this.defaultResponse[this.getService()]})
                        // this.setState({calculateState: 3})
                    }
                })
                .catch(error => {
                    console.log(error);
                    if(retries > 0) {
                        var retriesLeft = retries - 1;
                        console.log("Failure in GET, no. of retries left:", retriesLeft);
                        this.getResults(frontendData, retriesLeft, backoff + 500);
                    }
                    else {
                        this.setState({calculateState: 2, response: this.defaultResponse[this.getService()]})
                        // this.setState({calculateState: 3})
                    }
                })
            })
        })
    }
}

```

```

        }
    })
}, backoff)
})
.catch(error => {
    console.log(error);
    if(retries > 0) {
        var retriesLeft = retries - 1;
        console.log("Failure in POST, no. of retries left:", retriesLeft);
        this.getResults(frontendData, retriesLeft, backoff + 500);
    }
    else {
        this.setState({calculateState: 2, response: this.defaultResponse[this.getService()]})
        // this.setState({calculateState: 3})
    }
})
}
}

render() {
    const { showHideServices, showHideFilters, showHideInstances, selectEC2, selectS3, selectRDS,
        selectConsistency, selectErrorRate, selectLatency, selectElapsedTime, selectConnectionTime,
        selectThroughput, selectCpu, selectMemory,
        filterAlert, serviceAlert, instanceAlert,
        selectT2Micro, selectT2Nano, selectT2Small, selectT2Medium, selectT2Large, selectT2XLarge } =
this.state;
    return(
        <div>
            <div className = 'titleDivStyle3'>
                <h1 className = 'titleStyle3'>Dashboard</h1>
            </div>
            <div>
                <div className = 'questionDivStyle'>
                    <p className = 'questionStyle'>What AWS Service are you looking for?</p>
                    <button className = 'selectButtonStyle' onClick={() =>
this.hideComponent("showHideServices")}>Select Service</button>
                </div>
                { showHideServices && (
                    <div className = 'serviceDivStyle'>
                        <button className={selectEC2 ? "ec2StyleTrue": "ec2StyleFalse"} onClick={() =>
this.handleClick("EC2")}>{ this.ec2 }</button>
                        <button className={selectS3 ? "s3StyleTrue": "s3StyleFalse"} onClick={() =>
this.handleClick("S3")}>{ this.s3 }</button>
                        <button className={selectRDS ? "rdsStyleTrue": "rdsStyleFalse"} onClick={() =>
this.handleClick("RDS")}>{ this.rds }</button>
                    </div>
                    <p className='alertStyle'>{serviceAlert}</p>
                </div>
            )
        )
    )
}
}

```

```

        </div> )
    </div>
    <div>
        <div className = 'questionDivStyle'>
            <p className = 'questionStyle'>{ !selectS3 ? 'What performance metrics do you want us to focus on?' :
            'What metric should ML prediction be applied to?' }</p>
            <button className = 'selectButtonStyle' onClick={() =>
            this.hideComponent("showHideFilters")}>Select Filters</button>
        </div>
        { showHideFilters && (
            <div className = 'serviceDivStyle'>
                <div>
                    <input type = { selectS3 ? "hidden" : "checkbox" } id = "checkbox"/>
                    <label className='checkboxStyle'>{ selectS3 ? "" : "Assign same priority to all filters"
78
} </label><br/>
                </div>
                <div style={{display: 'inline'}}>
                    <div style={{display: 'inline'}}>
                        <button className={selectConsistency ? "consistencyStyleTrue": "consistencyStyleFalse"} onClick={() => this.handleClick("Consistency")}>Consistency</button>
                        {(selectConsistency && document.getElementById("checkbox").checked!==true &&
                            <input className='priorityTextStyle' type="text" id="PrConsistency" placeholder="Enter
Priority" required/>
                        )}
                    </div>
                    <div style={{display: 'inline'}}>
                        <button className={selectErrorRate ? "errorStyleTrue": "errorStyleFalse"} onClick={() =>
this.handleClick("Error Rate")}>Error Rate</button>
                        {(selectErrorRate && document.getElementById("checkbox").checked!==true &&
                            <input className='priorityTextStyle' type="text" id="PrError" placeholder="Enter Priority"
required/>
                        )}
                    </div>
                    <div style={{display: 'inline'}}>
                        <button className={selectLatency ? "latencyStyleTrue": "latencyStyleFalse"} onClick={() =>
this.handleClick("Latency")}>Latency</button>
                        {(selectLatency && document.getElementById("checkbox").checked!==true &&
                            <input className='priorityTextStyle' type="text" id="PrLatency" placeholder="Enter
Priority" required/>
                        )}
                    </div>
                    <div style={{display: 'inline'}} >
                        <button className={selectElapsedTime ? "elapsedStyleTrue": "elapsedStyleFalse"} onClick={() =>
this.handleClick("Elapsed Time")}>Elapsed Time</button>
                        {(selectElapsedTime && document.getElementById("checkbox").checked!==true &&
                            <input className='priorityTextStyle' type="text" id="PrElapsed" placeholder="Enter
Priority" required/>
                        )}
                    </div>
                </div>
            </div>
        )
    </div>

```

```

<div style={{display: 'inline'}}>
    <button className={selectConnectionTime ? "connectionStyleTrue": "connectionStyleFalse"} onClick={() => this.handleClick("Connection Time")}>Connection Time</button>
        {(selectConnectionTime && document.getElementById("checkbox").checked!==true &&
            <input className='priorityTextStyle' type="text" id="PrConnection" placeholder="Enter Priority" required/>
        )}
    </div>
    <div style={{display: 'inline'}}>
        <button className={selectThroughput ? "throughputStyleTrue": "throughputStyleFalse"} onClick={() => this.handleClick("Throughput")}>Throughput</button>
            {(selectThroughput && document.getElementById("checkbox").checked!==true &&
                <input className='priorityTextStyle' type="text" id="PrThroughput" placeholder="Enter Priority" required/>
            )}
    </div>
    { selectEC2 && (
        <div style={{display: 'inline'}}>
            <button className={selectCpu ? "cpuStyleTrue": "cpuStyleFalse"} onClick={() => this.handleClick("CPU Utilization")}>CPU Utilization</button>
                {(selectCpu && document.getElementById("checkbox").checked!==true &&
                    <input className='priorityTextStyle' type="text" id="PrCpu" placeholder="Enter Priority" required/>
                )}
        </div>
    )}
    { selectEC2 && (
        <div style={{display: 'inline'}}>
            <button className={selectMemory ? "memoryStyleTrue": "memoryStyleFalse"} onClick={() => this.handleClick("Memory")}>Memory</button>
                {(selectMemory && document.getElementById("checkbox").checked!==true &&
                    <input className='priorityTextStyle' type="text" id="PrMemory" placeholder="Enter Priority" required/>
                )}
        </div>
    )}
    <div>
        <p className='alertStyle'>{filterAlert}</p>
    </div>
</div> )
</div>
{ !selectS3 && (
<div>
    <div className = 'questionDivStyle'>
        <p className = 'questionStyle'>What Instance type of the above listed AWS Service do you want?</p>

```

```

        <button className = 'selectButtonStyle' onClick={() =>
this.hideComponent("showHideInstances")}>Select Instance</button>
      </div>

      {

        selectEC2 && showHideInstances && (
          <div className = 'serviceDivStyle'>
            <button className={selectT2Nano ? "t2NanoStyleTrue": "t2NanoStyleFalse"} onClick={() =>
this.handleClick("T2Nano")}>t2.nano</button>
            <button className={selectT2Micro ? "t2MicroStyleTrue": "t2MicroStyleFalse"} onClick={() =>
this.handleClick("T2Micro")}>t2.micro</button>
            <button className={selectT2Small ? "t2SmallStyleTrue": "t2SmallStyleFalse"} onClick={() =>
this.handleClick("T2Small")}>t2.small</button>
            <button className={selectT2Medium ? "t2MediumStyleTrue": "t2MediumStyleFalse"} onClick={() =>
this.handleClick("T2Medium")}>t2.medium</button>
            <button className={selectT2Large ? "t2LargeStyleTrue": "t2LargeStyleFalse"} onClick={() =>
this.handleClick("T2Large")}>t2.large</button>
            <button className={selectT2XLarge ? "t2XLargeStyleTrue": "t2XLargeStyleFalse"} onClick={() =>
this.handleClick("T2XLarge")}>t2.xlarge</button>
            <p className='alertStyle'>{instanceAlert}</p>
          </div>
        )
      }

      { selectRDS && showHideInstances && (
        <div className = 'serviceDivStyle'>
          <button className={selectT2Micro ? "t2MicroStyleTrue": "t2MicroStyleFalse"} onClick={() =>
this.handleClick("T2Micro")}>db.t2.micro</button>
          <button className={selectT2Small ? "t2SmallStyleTrue": "t2SmallStyleFalse"} onClick={() =>
this.handleClick("T2Small")}>db.t2.small</button>
          <button className={selectT2Medium ? "t2MediumStyleTrue": "t2MediumStyleFalse"} onClick={() =>
this.handleClick("T2Medium")}>db.t2.medium</button>
          <p className='alertStyle'>{instanceAlert}</p>
        </div>
      )
    }

    { this.calculateButton() }
  </div>

);
}

export default Dashboard;

```

### Dashboard.css

```
.title42 {
    margin-top: 0%;
    font-size: 250%;
    color: #FDFDFD;
    -webkit-box-shadow: 2px 2px 2px #222;
    text-align: center;
    padding: 2%;
    margin-bottom: 2%;
}

.title22 {
    background-color: #F9CC88;
    width: 23%;
    margin-left: auto;
    margin-right: auto;
    border-bottom-left-radius: 50px;
    border-bottom-right-radius: 50px;
    padding-bottom: 0.4%;
    margin-bottom: 5%;
}

.questionStyle {
    font-size: 180%;
    font-family:'Arial Narrow Bold';
    width: 78%;
    text-align: left;
    margin-left: 2%;
    color: #FDFDFD;
}

.questionDivStyle {
    align-self: center;
    display: flex;
    background-color: #88B7F9;
    margin-bottom: 1%;
    width: 70%;
    border-radius: 50px;
    height: 3%;
    margin-left: auto;
    margin-right: auto;
}

.selectButtonStyle {
    height: 20%;
    width: 15%;
    padding-top: 10px;
    padding-bottom: 10px;
    border-width: 0px;
    border-radius: 20px;
```

```
    margin-top: 2%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    background-color: #F9CC88 ;  
  
}  
  
.serviceDivStyle {  
    background-color: #FDFDFD;  
    width: 70%;  
    margin-left: auto;  
    margin-right: auto;  
    margin-bottom: 1%;  
    border-radius: 40px;  
    text-align: center;  
    padding-top: 3%;  
}  
  
.s3StyleTrue {  
    background-color: #FFA500;  
    color: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 15%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 3%;  
}  
  
.s3StyleFalse {  
    background-color: #F9CC88;  
    color: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 15%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;
```

```
margin-top: 3%;  
}  
  
.rdsStyleTrue {  
background-color: #FFA500;  
color: #FDFDFD;  
padding-top: 10px;  
padding-bottom: 10px;  
border-width: 0px;  
border-radius: 20px;  
width: 15%;  
margin: 1%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
color: #FDFDFD;  
margin-top: 3%;  
}  
  
.rdsStyleFalse {  
background-color: #F9CC88;  
color: #FDFDFD;  
padding-top: 10px;  
padding-bottom: 10px;  
border-width: 0px;  
border-radius: 20px;  
width: 15%;  
margin: 1%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
color: #FDFDFD;  
margin-top: 3%;  
}  
  
.ec2StyleTrue {  
background-color: #FFA500;  
color: #FDFDFD;  
padding-top: 10px;  
padding-bottom: 10px;  
border-width: 0px;  
border-radius: 20px;  
width: 15%;  
margin: 1%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
color: #FDFDFD;  
margin-top: 3%;
```

```
.ec2StyleFalse {  
background-color: #F9CC88;  
color: #FDFDFD;  
padding-top: 10px;  
padding-bottom: 10px;  
border-width: 0px;  
border-radius: 20px;  
width: 15%;  
margin: 1%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
color: #FDFDFD;  
margin-top: 3%;  
}  
  
.consistencyStyleTrue {  
background-color: #FFA500;  
color: #FDFDFD;  
padding-top: 10px;  
padding-bottom: 10px;  
border-width: 0px;  
border-radius: 20px;  
width: 17%;  
margin: 1%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
color: #FDFDFD;  
margin-top: 2%;  
}  
  
.consistencyStyleFalse {  
background-color: #F9CC88;  
color: #FDFDFD;  
padding-top: 10px;  
padding-bottom: 10px;  
border-width: 0px;  
border-radius: 20px;  
width: 17%;  
margin: 1%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
color: #FDFDFD;  
margin-top: 2%;
```

```
}

.errorStyleTrue {
    background-color: #FFA500;
    border: #FDFDFD;
    padding-top: 10px;
    padding-bottom: 10px;
    border-width: 0px;
    border-radius: 20px;
    width: 17%;
    margin: 1%;
    font-family: 'Arial Narrow Bold';
    font-weight: bold;
    font-size: 130%;
    color: #FDFDFD;
    margin-top: 2%;
}

.errorStyleFalse {
    background-color: #F9CC88;
    border: #FDFDFD;
    padding-top: 10px;
    padding-bottom: 10px;
    border-width: 0px;
    border-radius: 20px;
    width: 17%;
    margin: 1%;
    font-family: 'Arial Narrow Bold';
    font-weight: bold;
    font-size: 130%;
    color: #FDFDFD;
    margin-top: 2%;
}

.latencyStyleTrue {
    background-color: #FFA500;
    border: #FDFDFD;
    padding-top: 10px;
    padding-bottom: 10px;
    border-width: 0px;
    border-radius: 20px;
    width: 17%;
    margin: 1%;
    font-family: 'Arial Narrow Bold';
    font-weight: bold;
    font-size: 130%;
    color: #FDFDFD;
    margin-top: 2%;
```

```
}

.latencyStyleFalse {
    background-color: #F9CC88;
    border: #FDFDFD;
    padding-top: 10px;
    padding-bottom: 10px;
    border-width: 0px;
    border-radius: 20px;
    width: 17%;
    margin: 1%;
    font-family: 'Arial Narrow Bold';
    font-weight: bold;
    font-size: 130%;
    color: #FDFDFD;
    margin-top: 2%;
}

.elapsedStyleTrue {
    background-color: #FFA500;
    border: #FDFDFD;
    padding-top: 10px;
    padding-bottom: 10px;
    border-width: 0px;
    border-radius: 20px;
    width: 17%;
    margin: 1%;
    font-family: 'Arial Narrow Bold';
    font-weight: bold;
    font-size: 130%;
    color: #FDFDFD;
    margin-top: 2%;
}

.elapsedStyleFalse {
    background-color: #F9CC88;
    border: #FDFDFD;
    padding-top: 10px;
    padding-bottom: 10px;
    border-width: 0px;
    border-radius: 20px;
    width: 17%;
    margin: 1%;
    font-family: 'Arial Narrow Bold';
    font-weight: bold;
    font-size: 130%;
    color: #FDFDFD;
    margin-top: 2%;
}
```

```
.connectionStyleTrue {  
    background-color: #FFA500;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 17%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 5%;  
}
```

```
.connectionStyleFalse {  
    background-color: #F9CC88;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 17%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 5%;  
}
```

```
.throughputStyleTrue {  
    background-color: #FFA500;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 17%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 5%;  
}
```

```
.throughputStyleFalse {  
    background-color: #F9CC88;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 17%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 5%;  
}  
  
.cpuStyleTrue {  
    background-color: #FFA500;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 17%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 2%;  
}  
  
.cpuStyleFalse {  
    background-color: #F9CC88;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 17%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 2%;  
}
```

```
.memoryStyleTrue {  
    background-color: #FFA500;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 17%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 2%;  
}  
  
.memoryStyleFalse {  
    background-color: #F9CC88;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 17%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
    margin-top: 2%;  
}  
  
.t2NanoStyleTrue {  
    background-color: #FFA500;  
    border: #FDFDFD;  
    padding-top: 10px;  
    padding-bottom: 10px;  
    border-width: 0px;  
    border-radius: 20px;  
    width: 15%;  
    margin: 1%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    color: #FDFDFD;  
}  
  
.t2NanoStyleFalse {
```

```
background-color: #F9CC88;
5    lor: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
10   margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;

}

.t2MicroStyleTrue {
background-color: #FFA500;
5    lor: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
10   margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;

}

.t2MicroStyleFalse {
background-color: #F9CC88;
5    lor: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
10   margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;

}

.t2SmallStyleTrue {
background-color: #FFA500;
```

```
5 5lor: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;

}

.t2SmallStyleFalse {
background-color: #F9CC88;
5 5lor: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;

}

.t2MediumStyleTrue {
background-color: #FFA500;
5 5lor: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;

}

.t2MediumStyleFalse {
background-color: #F9CC88;
color: #FDFDFD;
```

```
5 padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;
}

.t2LargeStyleTrue {
background-color: #FFA500;
5 border: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;
margin-bottom: 5%;
}

.t2LargeStyleFalse {
background-color: #F9CC88;
5 border: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;
margin-bottom: 5%;
}

.t2XLargeStyleTrue {
background-color: #FFA500;
5 border: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
```

```
border-width: 0px;
border-radius: 20px;
width: 15%;
margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;
margin-top: 2%;
}

.t2XLargeStyleFalse {
background-color: #F9CC88;
color: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
width: 15%;
margin: 1%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
color: #FDFDFD;
margin-top: 2%;
}

.calculateDivStyle {
text-align: center;
}

.calculateButtonStyle {
width: 12%;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
margin-top: 2%;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
background-color: #F9CC88 ;
color: #FDFDFD;
margin-bottom: 3%;
}

.checkboxStyle {
color: #696969;
text-align: left;
```

```
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
margin-right: 20px;
margin-bottom: 20px;
}

.submitButtonStyle {
margin-top: 4%;
border-width: 0px;
border-radius: 20px;
width: 10%;
background-color: #88B7F9;
border: #FDFDFD;
padding-top: 10px;
padding-bottom: 10px;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
}

.alertStyle {
color: #696969;
text-align: left;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
text-align: center;
margin-top: 3%;
padding: 2%;
}

.lastAlertStyle {
color: #696969;
text-align: left;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 130%;
text-align: center;
margin-top: 0.7%;
padding: 2%;
}

.priorityTextStyle {
background-color: #D3D3D3;
padding-top: 10px;
padding-bottom: 10px;
border-width: 0px;
border-radius: 20px;
```

```
60 th: 10%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
text-align: center;  
font-size: 100%;  
}
```

### Results.<sup>j29</sup>

```
import { Component } from 'react';  
import {Link} from 'react-router-dom'  
import { Doughnut, Bar } from 'react-chartjs-2';  
import './styles/Results.css'  
  
class Results extends Component {  
  
  textUtil(filterCompare) {  
    var components = [];  
    components.push(  
      <p className = 'filterHeadingStyle'>Observations</p>  
    );  
    Object.entries(filterCompare).forEach(item => {  
      var instanceName = item[0];  
      var instanceValue = item[1];  
      components.push(  
        <div className = 's3DivStyle'>  
          <div className = 'instanceNameStyle'>{ instanceName }</div>  
          <div className = 'instanceValueStyle'>{ instanceValue + "%" }</div>  
        </div>  
      );  
    })  
    return components;  
  }  
  
  s3TextUtil(payload) {  
    var components = [];  
    components.push(  
      <p className = 's3FilterHeadingStyle'>Selected Filters</p>  
    );  
    Object.entries(payload.selectedValues).forEach(item => {  
      var filterName = item[0];  
      var filterValue = item[1];  
      components.push(  
        <div className = 's3DivStyle'>  
          <div className = 'selectedS3FilterNameStyle'>{ filterName }</div>  
          <div className = 'selectedS3FilterValueStyle'>{ filterValue }</div>  
        </div>  
      );  
    })  
    components.push(  
    );  
  }  
}
```

```

<p className = 's3FilterHeadingStyle'>Predicted Filter</p>
);
Object.entries(payload.predictedValues).forEach(item => {
  var filterName = item[0];
  var filterValue = item[1];
  components.push(
    <div className = 's3DivStyle'>
      <div className = 'predictedS3FilterNameStyle'>{ filterName }</div>
      <div className = 'predictedS3FilterValueStyle'>{ filterValue }</div>
    </div>
  );
})
return components;
}

createData(filterName, filterData) {
  const bgColorMap = {
    "t2.nano": "#B21F00",
    "t2.micro": "#C9DE00",
    "t2.small": "#2FDE00",
    "t2.medium": "#00A6B4",
    "t2.large": "#6800B4",
    "t2.xlarge": "#FF4500",
    "db.t2.micro": "#C9DE00",
    "db.t2.small": "#2FDE00",
    "db.t2.medium": "#00A6B4",
  };

  const hvColorMap = {
    "t2.nano": "#501800",
    "t2.micro": "#4B5000",
    "t2.small": "#175000",
    "t2.medium": "#003350",
    "t2.large": "#35014F",
    "t2.xlarge": "#991900",
    "db.t2.micro": "#4B5000",
    "db.t2.small": "#175000",
    "db.t2.medium": "#003350",
  };
}

var graphLabels = [];
var graphData = [];
var bgColor = [];
var hvColor = [];

Object.entries(filterData).forEach(item => {
  var instanceType = item[0];
  var instanceValue = item[1];

```

```

graphLabels.push(instanceType);
graphData.push(instanceValue);
bgColor.push(bgColorMap[instanceType]);
hvColor.push(hvColorMap[instanceType]);
})

const data = {
  labels: graphLabels,
  datasets: [
    {
      label: "",
      backgroundColor: bgColor,
      hoverBackgroundColor: hvColor,
      data: graphData
    }
  ]
}

return data;
}

createCharts(data, filterCompare, displayName) {
  return(
    <div>
      <div className = 'doughnutGraphStyle'>
        <p className = 'filterStyle'>{ displayName }</p>
        <p className = 'filterRelativeStyle'>Relative Importance of Instance Types</p>
        <Doughnut
          data = { data }
          options = {{
            title: {
              display: true,
              text: displayName,
              fontSize: 20
            },
            legend: {
              display: true,
              position: 'right'
            }
          }}
        />
      </div>
      <div className = 'barGraphStyle'>
        <Bar
          data={ data }
          options={{{
            title:{}
            display: false,
          }}}
        />
      </div>
    </div>
  )
}

```

```

        text: displayName,
        fontSize: 20
    },
    legend: {
        display: false
    },
    tooltips: {
        enabled: false
    }
})
/>
</div>
{ this.textUtil(filterCompare) }
<br/>
<hr className = 'hrStyle' />
</div>
);
}
}

renderGraphs(data) {
    var components = [];
    Object.entries(data).forEach(item => {
        if(item[0] !== "overall") {
            var collectiveData = item[1];
            var filterData = collectiveData.data;
            var filterCompare = collectiveData.filterCompare;
            var displayName = collectiveData.displayName;

            var graphData = this.createData(displayName, filterData);
            var graphs = this.createCharts(graphData, filterCompare, displayName);
            components.push(graphs);
        }
    })
    return components;
}

renderOverall(data) {
    var collectiveData = data.overall;
    var filterData = collectiveData.data;
    var filterCompare = collectiveData.filterCompare;
    var displayName = collectiveData.displayName;

    var graphData = this.createData(displayName, filterData);
    var graph = this.createCharts(graphData, filterCompare, displayName);

    return graph;
}

```

```

render() {
  var service = this.props.location.state.service;
  var data = this.props.location.state.data;
  console.log(data);
  if(service === "Amazon S3") {
    return(
      <div>
        <div className = 'titleDivStyle2'>
          <h1 className = 'titleStyle2'>Results</h1>
        </div>
        <Link className = 'linkReturnStyle' to = "/dashboard">Return to Dashboard</Link>
        <div className = 'graphDivStyleCharts'>
          { this.s3TextUtil(data) }
          <p className = 's3FilterHeadingStyle'>About S3</p>
          <div className = 'aboutS3DivStyle'>
            <p className = 'aboutS3TextStyle'>S3 is an Object storage built to store and retrieve any amount
              of data from anywhere. 50
            <p> S3 has no instance types.<br/><br/> Amazon Simple Storage Service (Amazon S3) is an object
              storage service that offers industry-leading
              scalability, data availability, security, and performance.<br/><br/>This means customers of all
              sizes and industries can use it to store
              and protect any amount of data for a range of use cases, such as data lakes, websites, mobile
              applications, backup and
              restore, archive, enterprise applications, IoT devices, and big data analytics.<br/><br/> Amazon S3
              provides easy-to-use
              management features so you can organize your data and configure finely-tuned access controls to
              meet your specific business,
              organizational, and compliance requirements. <br/><br/>Amazon S3 is designed for
              99.999999999% (11 9's) of durability, and stores
              data for millions of applications for companies all around the world.<br/><br/><span style =
              {{color: '#F9CC88'}}>Cloud Service Optimizer</span> runs several consistent
              tests on S3's performance.</p>
            </div>
          </div>
        </div>
      );
    }

    return(
      <div>
        <div className = 'titleDivStyle2'>
          <h1 className = 'titleStyle2'>Results</h1>
        </div>
        <Link className = 'linkReturnStyle' to = "/dashboard">Return to Dashboard</Link>
        <p className = 'serviceTextStyle'>{ service }</p>
        <div className = 'graphDivStyleCharts'>
          { this.renderOverall(data) }
          { this.renderGraphs(data) }
        </div>
      </div>
    );
  }
}

```

```
        </div>
        </div>
    )
};

}

export default Results;
```

**Results.css**

```
.title2 {
    margin-top: 0%;
    font-size: 250%;
    color: #FDFDFD;
    -webkit-box-shadow: 2px 2px 2px #222;
    text-align: center;
    padding: 2%;
    margin-bottom: 2%;
}

.titleDivStyle2 {
    background-color: #F9CC88;
    width: 17%;
    margin-left: auto;
    margin-right: auto; 43
    border-bottom-left-radius: 50px;
    border-bottom-right-radius: 50px;
    padding-bottom: 0.4%;
}

.filterStyle {
    color: #696969;
    font-family: 'Arial Narrow Bold';
    font-weight: bold;
    font-size: 180%;
    text-align: center;
}

.filterRelativeStyle {
    color: #696969;
    font-family: 'Arial Narrow Bold';
    font-weight: bold;
    font-size: 150%;
    text-align: center;
}

.filterTextStyle {
    color: #696969;
    font-family: 'Arial Narrow Bold';
```

```
font-weight: bold;
25 t-size: 130%;
text-align: center;
margin-left: 7%;
margin-right: 7%;
margin-top: 2%;
margin-bottom: 2%;
}

.serviceTextStyle {
4 lor: #696969;
font-family: 'Arial Narrow Bold';
font-weight: bold;
font-size: 230%;
text-align: center;
}

.doughnutGraphStyle {
width: 30%;
height: 40%;
4 margin-top: 3%;
text-align: center;
margin-bottom: 3%;
margin-left: 35%;
}

.barGraphStyle {
width: 50%;
53 ght: 40%;
margin-top: 5%;
text-align: center;
margin-bottom: 3%;
margin-left: 25%;
}

.g85 mDivStyleCharts {
width: 100%;
height: 100%;
text-align: center;
margin-right: 30%;
}

.hrStyle {
border: 1.5px solid grey;
}

.linkReturnStyle {
color: #F9CC88;
background-color: #fdfdfd;
```

```
border-radius: 50px;
text-align: left;
font-size: 130%;
text-decoration: none;
font-weight: bold;
margin: 5%;

margin-left: 78%;
padding-left: 1%;
padding-right: 1%;
padding-top: 0.5%;
padding-bottom: 0.5%;
font-family: 'Arial Narrow Bold';
}

a:hover {
  color: orange;
}

.s248vStyle {
  display: flex;
  margin-left: auto;
  margin-right: auto;
  text-align: justify;
  justify-content: center;
  align-items: center;
}

.s3FilterHeadingStyle {
  color: #696969;
  font-family: 'Arial Narrow Bold';
  font-weight: bold;
  font-size: 170%;
  margin-top: 2%;
  margin-bottom: 2%;
}

.filterHeadingStyle {
  color: #696969;
  font-family: 'Arial Narrow Bold';
  font-weight: bold;
  font-size: 150%;
  margin-top: 2%;
  margin-bottom: 2%;
}

.selectedS3FilterNameStyle {
  color: #F9CC88;
  background-color: #FDFDFD;
  border-radius: 50px;
```

```
padding: 0.7%;  
margin-right: 2%;  
padding-right: 4%;  
padding-left: 2%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
margin-bottom: 0.7%;  
36 th: 11%;  
text-align: center;  
}  
  
.selectedS3FilterValueStyle {
```

```
color: #F9CC88;  
background-color: #FDFDFD;  
border-radius: 50px;  
padding: 0.7%;  
margin-right: 2%;  
padding-right: 4%;  
padding-left: 2%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
margin-bottom: 0.7%;  
justify-content: flex;  
text-align: center;  
width: 7%;  
}
```

```
.predictedS3FilterNameStyle {  
color: #FDFDFD;  
background-color: #F9CC88;  
border-radius: 50px;  
padding: 0.7%;  
margin-right: 2%;  
padding-right: 4%;  
padding-left: 2%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
margin-bottom: 0.7%;  
36 th: 11%;  
text-align: center;  
}
```

```
.predictedS3FilterValueStyle {  
color: #FDFDFD;  
background-color: #F9CC88;  
border-radius: 50px;
```

```
padding: 0.7%;  
margin-right: 2%;  
padding-right: 4%;  
padding-left: 2%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
margin-bottom: 0.7%;  
text-align: center;  
width: 7%;  
}  
  
.aboutS3DivStyle {  
background-color: #FDFDFD;  
border-radius: 50px;  
padding: 4%;  
width: 80%;  
margin-bottom: 3%;  
margin-left: 6.7%;  
background: rgba(255,255,255,0.7);  
}  
  
.aboutS3TextStyle {  
color: #696969;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
margin-left: 2%;  
margin-right: 2%;  
margin-top: 2%;  
margin-bottom: 2%;  
text-align: justify;  
}  
  
.instanceNameStyle {  
color: #FDFDFD;  
background-color: #F9CC88;  
border-radius: 50px;  
padding: 0.7%;  
margin-right: 2%;  
padding-right: 4%;  
padding-left: 2%;  
font-family: 'Arial Narrow Bold';  
font-weight: bold;  
font-size: 130%;  
margin-bottom: 0.7%;  
width: 18%;  
text-align: center;  
}
```

```
.instanceValueStyle {  
    color: #F9CC88;  
    background-color: #FDFDFD;  
    border-radius: 50px;  
    padding: 0.7%;  
    margin-right: 2%;  
    padding-right: 4%;  
    padding-left: 2%;  
    font-family: 'Arial Narrow Bold';  
    font-weight: bold;  
    font-size: 130%;  
    margin-bottom: 0.7%;  
    text-align: center;  
    width: 7%;  
}
```

## BACKEND

10  
**server.py**

```
from flask import Flask, render_template, jsonify, request  
from flaskext.mysql import MySQL  
from flask_cors import CORS, cross_origin  
import random  
import logging  
import datetime  
import json  
import ahp  
import os  
import time  
from threading import Thread  
from uwsgi_tasks import task, TaskExecutor  
import threading  
import databaseCommands as dbCommands  
import mlCode as mlp  
84  
app = Flask(__name__)  
CORS(app, support_credentials=True)  
app.debug = True  
  
def updateFilesInDB(endpointID, filesString):  
    dbCommands.updateFiles(endpointID, filesString)  
  
def logAPICall(name):  
    timestamp = datetime.datetime.now()  
    dbCommands.logAPICall(name, str(timestamp))  
  
def generatePNumber():  
    randomNo = 'p'
```

41

```
randomNoChars = ".join([random.choice(string.ascii_letters + string.digits) for n in range(9)])\nrandomNo = randomNo + randomNoChars\nprint(randomNo)\nreturn randomNo\n\ndef generateSNumber():\n    randomNo = 's' 41\n    randomNoChars = ".join([random.choice(string.ascii_letters + string.digits) for n in range(9)])\n    randomNo = randomNo + randomNoChars\n    print(randomNo)\n    return randomNo\n\ndef getValFromDb(text):\n    #this function will use the entered text and search for statically\n    #stored content in a table\n    #if it matches a certain string, it will return it\n    #else, it will say not found\n    statement = "select content from staticContent where name = \\'" + text + "\';"\n    result = dbCommands.dbSelect(statement)\n    if(result == []):\n        return "not found"\n    else:\n        return result\n\ndef retrieveValuesFromDB():\n    return dbCommands.retrieveValues()\n\ndef startProcessing(filters, serviceType, instanceType, priorities):\n    filtersList = filters.split(",")\n    instancesList = instanceType.split(",")\n    prioritiesList = priorities.split(",")\n    result = dbCommands.ahpCode(serviceType, instancesList, filtersList, prioritiesList)\n    return result\n\ndef bgTask(pNo):\n    print('Started thread: ')\n    val = getResultsFromProcessingNumber(pNo)\n\ndef processFilters(timestamp, filters, priorities, serviceType, instanceTypes):\n    #it uses the filters as input and generates ahp and ml-based calculations\n    errorMessage = ""\n    filtersList = filters.split(",")\n    prioritiesList = priorities.split(",")\n    if (filters == ""):\n        errorMessage+="Filters is missing."'\n    if(timestamp == ""):\n        errorMessage+=" Timestamp is missing."'\n    if(serviceType == ""):\n        errorMessage+=" ServiceType is missing."'
```

```

if(instanceTypes == ""):
    errorMessage+=" InstanceTypes is missing."
if(priorities == ""):
    errorMessage+=" Priorities is missing."
if(not len(filtersList) == len(prioritiesList)):
    errorMessage+=" Filters and Priorities length do not match."
if errorMessage != "":
    return ("failure", errorMessage)
else:
    pNo = generatePNumber()
    while(dbCommands.isDuplicatePNumber(pNo)):
        pNo = generatePNumber()
    dbCommands.insertPNumber(pNo)
    dbCommands.insertPRequest(str(pNo), timestamp, filters, priorities, serviceType, instanceTypes)
    return ("success", str(pNo))

def updateResult(result, pNo):
    timestamp = datetime.datetime.now()
    dbCommands.updateResults(result, pNo, str(timestamp))

def checkStatus(pNo):
    if(dbCommands.checkProcessingStatus(pNo)):
        return 1
    else:
        return 0

def checkIfProcessingIsDone(pNo):
    if(dbCommands.checkIfProcessingIsDone(pNo)):
        return 1
    else:
        return 0

def isAS3Request(pNo):
    if(dbCommands.isAS3Request(pNo)):
        return 1
    else:
        return 0

def getResultsFromPNo(text):
    if(not dbCommands.isDuplicatePNumber(text)):
        return (0, "Invalid PNo.")
    elif(checkIfProcessingIsDone(text)):
        return (0, "Processing for this pNo is still pending... Try again")
    else:
        result = dbCommands.getResultFromAHP(text)
        return result

def getResultsFromProcessingNumber(text):
    if(not dbCommands.isDuplicatePNumber(text)):
```

```

        return (0, "Invalid PNo.")
    elif(checkStatus(text)):
        return (0, "Processing for this pNo is complete!")
    else:
        filters = dbCommands.getFilters(text)
        serviceType = dbCommands.getServiceType(text)
        priorities = dbCommands.getPriorities(text)
        if(isAS3Request(text)):
            result = mlp.runML(serviceType, 'na', priorities, filters)
            updateResult(result, text)
            return result
        else:
            instanceType = dbCommands.getInstanceType(text)
            result = startProcessing(filters, serviceType, instanceType, priorities)
            updateResult(result, text)
            return result

def getLatestTimestamp(endpointID):
    filesString = dbCommands.GetFiles(endpointID)
    return filesString

def initiateLogging(endpointID):
    sNo = generateSNumber()
    while(dbCommands.isDuplicateSNumber(sNo)):
        sNo = generateSNumber()
    dbCommands.insertSNumber(sNo, endpointID)
    return str(sNo)

def updateSessionInfo(sessionID):
    dbCommands.updateSession(sessionID)

def saveToEC2Summary(index, valuesList, sessionID, endpointID, instanceType, logsSummary, timestamp):
    label = valuesList[0]
    numberOfSamples = valuesList[1]
    average = valuesList[2]
    minVal = valuesList[3]
    maxVal = valuesList[4]
    stdDev = valuesList[5]
    errorRate = valuesList[6]
    throughput = valuesList[7]
    receivedKBPS = valuesList[8]
    sentKBPS = valuesList[9]
    avgBytes = valuesList[10]
    dbCommands.insertEC2Summary(index, label, numberOfSamples, average, minVal, maxVal, stdDev, errorRate,
    throughput, receivedKBPS, sentKBPS, avgBytes, endpointID, sessionID, instanceType, timestamp)

def saveSummaryInEC2(index, sessionID, endpointID, instanceType, logsSummary, timestamp):
    summary_dict = json.loads(logsSummary)
    valuesList = []

```

```

print(summary_dict.values())
for i in summary_dict.values():
    valuesList.append(i)
print(valuesList)
saveToEC2Summary(index, valuesList, sessionID, endpointID, instanceType, logsSummary, timestamp)

def saveSummaryDataInEC2(index, sessionID, endpointID, instanceType, logsSummary, timestamp):
    summary_dict = json.loads(logsSummary)
    valuesList = []

def saveToEC2Logs(index, valuesList, instanceType, sessionID, endpointID, timestamp):
    elapsed = valuesList[0]
    responseCode = valuesList[1]
    responseMessage = valuesList[2]
    success = valuesList[3]
    latency = valuesList[4]
    connect = valuesList[5]
    dbCommands.insertEC2Logs(index, elapsed, responseCode, responseMessage, success, latency, connect,
    instanceType, endpointID, sessionID, timestamp)

def saveToEC2Performance(index, valuesList, instanceType, sessionID, endpointID, timestamp):
    cpu = valuesList[0]
    memory = valuesList[1]
    dbCommands.insertEC2Performance(index, cpu, memory, instanceType, endpointID, sessionID, timestamp)

def saveLogsInEC2(sessionID, endpointID, instanceType, logs, timestamp):
    logs_dict = json.loads(logs)
    print(len(logs_dict))
    print("Logs Dict['elapsed']:")
    print(len(logs_dict.values()))
    index = dbCommands.getLatestCount('ec2logs')
    for j in range(1, len(logs_dict['elapsed'])):
        valuesList = []
        for i in logs_dict.values():
            valuesList.append(i.get(str(j)))
        saveToEC2Logs(index, valuesList, instanceType, sessionID, endpointID, timestamp)
        index = index + 1

def savePerformanceInEC2(sessionID, endpointID, instanceType, logsPerformance, timestamp):
    logs_dict = json.loads(logsPerformance)
    print(len(logs_dict))
    print("LogsPerformance Dict['CPU']:")
    print(len(logs_dict.values()))
    index = dbCommands.getLatestCount('ec2performance')
    for j in range(1, len(logs_dict['CPU'])):
        valuesList = []
        for i in logs_dict.values():
            valuesList.append(i.get(str(j)))

```

```

        saveToEC2Performance(index, valuesList, instanceType, sessionID, endpointID, timestamp)
        index = index + 1

def saveSummary(sessionID, endpointID, instanceType, serviceType, logsSummary, timestamp):
    if(serviceType=='ec2'):
        index = dbCommands.getLatestCount('ec2summary')
        saveSummaryInEC2(index, sessionID, endpointID, instanceType, logsSummary, timestamp)
        return 1
    if(endpointID == "hello"):
        return 0
    else:
        return 0

def saveSummaryData(sessionID, endpointID, instanceType, serviceType, logsSummary, timestamp):
    if(serviceType=='ec2'):
        saveSummaryDataInEC2(index, sessionID, endpointID, instanceType, logsSummary, timestamp)
        return 1
    if(endpointID == "hello"):
        return 0
    else:
        return 0

def saveLogs(sessionID, endpointID, instanceType, serviceType, logs, timestamp):
    if(serviceType=='ec2'):
        saveLogsInEC2(sessionID, endpointID, instanceType, logs, timestamp)
        return 1
    if(endpointID == "hello"):
        return 0
    else:
        return 0

def savePerformance(sessionID, endpointID, instanceType, serviceType, logsPerformance, timestamp):
    if(serviceType=='ec2'):
        savePerformanceInEC2(sessionID, endpointID, instanceType, logsPerformance, timestamp)
        return 1
    if(endpointID == "hello"):
        return 0
    else:
        return 0

def checkPassword(endpointID, password):
    if(dbCommands.checkPassword(endpointID, password)):
        return 1
    else:
        return 0

def logLogin(endpointID, timestamp):
    dbCommands.logLogin(endpointID, timestamp)

```

```

def attemptLogin(endpointID, password, timestamp):
    if(checkPassword(endpointID, password)):
        logLogin(endpointID, timestamp)
        return "success"
    else:
        return "failure"

def updateDerivedValues(sessionID, endpointID, instanceType, serviceType, timestamp):
    elapsed = dbCommands.getElapsed(instanceType, serviceType)
    latency = dbCommands.getLatency(instanceType, serviceType)
    connect = dbCommands.getConnect(instanceType, serviceType)
    cpu = dbCommands.getCPU(instanceType, serviceType)
    memory = dbCommands.getMemory(instanceType, serviceType)
    standardDeviation = dbCommands.getOverallStandardDeviation(instanceType, serviceType)
    errorRate = dbCommands.getOverallErrorRate(instanceType, serviceType)
    throughput = dbCommands.getOverallThroughput(instanceType, serviceType)
    cpu = dbCommands.getCPU(instanceType, serviceType)
    memory = dbCommands.getMemory(instanceType, serviceType)
    dbCommands.updateDerivedValues(elapsed, latency, connect, cpu, memory, standardDeviation, errorRate,
    throughput, sessionID, endpointID, instanceType, serviceType, timestamp)
    dbCommands.updateDerivedCalculationLogs(endpointID, sessionID, serviceType, instanceType, timestamp)

def getNameFromEndpointID(endpointID):
    name = dbCommands.getNameFromEndpointID(endpointID)
    return name

def createEndpoint(name, password):
    endpointID = dbCommands.createEndpoint(name, password)
    return str(endpointID)

def updateDerValues(sessionID, endpointID, instanceType, serviceType, timestamp):
    elapsed = dbCommands.getStoredElapsed(instanceType, serviceType)
    latency = dbCommands.getStoredLatency(instanceType, serviceType)
    connect = dbCommands.getStoredConnect(instanceType, serviceType)
    cpu = dbCommands.getStoredCPU(instanceType, serviceType)
    memory = dbCommands.getStoredMemory(instanceType, serviceType)
    standardDeviation = dbCommands.getOverallStandardDeviation(instanceType, serviceType)
    errorRate = dbCommands.getOverallErrorRate(instanceType, serviceType)
    throughput = dbCommands.getOverallThroughput(instanceType, serviceType)
    dbCommands.updateDerivedValues(elapsed, latency, connect, cpu, memory, standardDeviation, errorRate,
    throughput, sessionID, endpointID, instanceType, serviceType, timestamp)
    dbCommands.updateDerivedCalculationLogs(endpointID, sessionID, serviceType, instanceType, timestamp)

def saveEC2(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev, errorRate,
throughput, summarySize, timestamp, cpu, memory, performanceSize):
    dbCommands.saveEC2(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect,
    stdDev, errorRate, throughput, summarySize, timestamp, cpu, memory, performanceSize)

```

```

def saveRDS(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev,
errorRate, throughput, summarySize, timestamp, cpu, memory, performanceSize):
    dbCommands.saveRDS(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect,
stdDev, errorRate, throughput, summarySize, timestamp, cpu, memory, performanceSize)

def saveS3(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev, errorRate,
throughput, summarySize, timestamp, cpu, memory, performanceSize):
    dbCommands.saveS3(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect,
stdDev, errorRate, throughput, summarySize, timestamp, cpu, memory, performanceSize)

def saveLogData(sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev, errorRate,
throughput, summarySize, timestamp, isPerformanceIncluded, cpu, memory, performanceSize):
    if(serviceType == 'ec2'):
        index = dbCommands.getLatestCount('saveec2')
        if(isPerformanceIncluded == '1'):
            saveEC2(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev,
errorRate, throughput, summarySize, timestamp, cpu, memory, performanceSize)
        else:
            saveEC2(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev,
errorRate, throughput, summarySize, timestamp, 0, 0, 0)
        return 1
    elif(serviceType == 'rds'):
        #do rds
        index = dbCommands.getLatestCount('saverds')
        saveRDS(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev,
errorRate, throughput, summarySize, timestamp, 0, 0, 0)
        return 1
    elif(serviceType == 's3'):
        #do s3
        index = dbCommands.getLatestCount('saves3')
        saveS3(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev, errorRate,
throughput, summarySize, timestamp, 0, 0, 0)
        return 1
    else:
        return 0

@app.route('/', methods =['GET'])
def hello():
    return "Hello! Welcome to the server of Cloud Service Optimizer"

@app.route('/front-end/get-content/', methods=['GET', 'OPTIONS'])
def front_end_get_content():

    logAPICall('f/get-content')
    val = ""
    text = ""
    text = request.args.get("string")
    val = getValFromDb(text)
    if(val=="not found"):


```

```

value = jsonify({
    "status": "failure",
    "message": ""
})
20
value.headers.add('Access-Control-Allow-Origin', '*')
value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
return value
else:
    value = jsonify({
        "status": "success",
        "message": val
    })
20
value.headers.add('Access-Control-Allow-Origin', '*')
value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
return value

@app.route('/front-end/post-process-filters/', methods=['POST'])
def front_end_post_process_filters():
    logAPICall('f/get-process-filters')
    request_data = request.get_json()
    serviceType = request_data['serviceType']
    timestamp = request_data['timestamp']
    filters = request_data['filters']
    priorities = request_data['priorities']
    instanceType = request_data['instanceType']
    val = []
    val = processFilters(timestamp, filters, priorities, serviceType, instanceType)
    value = jsonify({
        "status": val[0],
        "message": val[1]
    })
20
value.headers.add('Access-Control-Allow-Origin', '*')
value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
return value

@app.route('/front-end/get-results/', methods=['GET', 'OPTIONS'])
def front_end_get_results():
    val = ""
    text = ""
    text = request.args.get("processingNumber")
    val = getResultsFromProcessingNumber(text)
    if(val[0]==0):
        value = jsonify({
            "status": "failure",
            "data": val[1],
            "processingNumber": text
        })
20
value.headers.add('Access-Control-Allow-Origin', '*')

```

```

        value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
        return value
    else:
        val1 = json.loads(val)
        value = jsonify({
            "status": "success",
            "payload": val1,
            "processingNumber": text
        })
        20
        value.headers.add('Access-Control-Allow-Origin', '*')
        value.headers.add('Access-Control-Allow-Headers', 'Content-Type')
        return value

@app.route('/endpoints/get-files/', methods=['POST'])
def endpoints_get_latest_timestamp():
    logAPICall('e/get-files')
    request_data = request.get_json()
    endpointID = request_data['endpointID']
    val = getLatestTimestamp(endpointID)
    return jsonify({
        "status": "success",
        "message": val,
        "endpointID": endpointID
    })

@app.route('/endpoints/update-files/', methods=['POST'])
def endpoints_update_files():
    logAPICall('e/update_files')
    request_data = request.get_json()
    endpointID = request_data['endpointID']
    sessionID = request_data['sessionID']
    filesString = request_data['filesString']
    print(filesString)
    updateFilesInDB(endpointID, filesString)
    updateSessionInfo(sessionID)
    return jsonify({
        "status": "success",
        "message": "Files list updated.",
        "endpointID": endpointID,
        "filesString": filesString
    })

@app.route('/endpoints/initiate-logging/', methods=['GET'])
def endpoints_initiate_logging():
    logAPICall('e/initiate-logging')
    val = ""
    endpointID = ""
    endpointID = request.args.get("endpointID")
    val = initiateLogging(endpointID)

```

```

return jsonify({
    "status": "success",
    "message": val
})

@app.route('/endpoints/post-logs/', methods=['POST'])
def endpoints_post_summary():
    logAPICall('e/post-logs')
    request_data = request.get_json()
    sessionID = request_data['sessionID']
    endpointID = request_data['endpointID']
    instanceType = request_data['instanceType']
    serviceType = request_data['serviceType']
    logsSummary = request_data['logsSummary']
    logs = request_data['logs']
    timestamp = request_data['timestamp']
    isPerformanceIncluded = request_data['isPerformanceIncluded']
    logsPerformance = request_data['logsPerformance']
    val = saveSummary(sessionID, endpointID, instanceType, serviceType, logsSummary, timestamp)
    val2 = saveLogs(sessionID, endpointID, instanceType, serviceType, logs, timestamp)
    val3 = 1
    if(isPerformanceIncluded == '1'):
        val3 = savePerformance(sessionID, endpointID, instanceType, serviceType, logsPerformance, timestamp)
    if(val and val2 and val3):
        print("In update derived values")
        updateDerivedValues(sessionID, endpointID, instanceType, serviceType, timestamp)
    if(not val or not val2 or not val3):
        return jsonify({
            "status": "failure",
            "message": val,
            "sessionID": sessionID,
            "endpointID": endpointID,
            "serviceType": serviceType,
            "instanceType": instanceType
        })
    else:
        return jsonify({
            "status": "success",
            "message": val,
            "sessionID": sessionID,
            "endpointID": endpointID,
            "serviceType": serviceType,
            "instanceType": instanceType
        })

@app.route('/endpoints/post-log-data/', methods=['POST'])
def endpoints_post_log_data():
    logAPICall('e/post-log-data')

```

```

request_data = request.get_json()
sessionID = request_data['sessionID']
endpointID = request_data['endpointID']
instanceType = request_data['instanceType']
serviceType = request_data['serviceType']
elapsed = request_data['elapsed']
latency = request_data['latency']
connect = request_data['connect']
stdDev = request_data['stdDev']
errorRate = request_data['errorRate']
throughput = request_data['throughput']
summarySize = request_data['summarySize']
timestamp = request_data['timestamp']
isPerformanceIncluded = request_data['isPerformanceIncluded']
cpu = request_data['cpu']
memory = request_data['memory']
performanceSize = request_data['performanceSize']
val = saveLogData(sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev,
errorRate, throughput, summarySize, timestamp, isPerformanceIncluded, cpu, memory, performanceSize)
if(val):
    print("In update derived values")
    updateDerValues(sessionID, endpointID, instanceType, serviceType, timestamp)
if(not val):
    return jsonify({
        "status": "failure",
        "message": val,
        "sessionID": sessionID,
        "endpointID": endpointID,
        "serviceType": serviceType,
        "instanceType": instanceType
    })
else:
    return jsonify({
        "status": "success",
        "message": val,
        "sessionID": sessionID,
        "endpointID": endpointID,
        "serviceType": serviceType,
        "instanceType": instanceType
    })

@app.route('/endpoints/endpoint-login/', methods=['POST'])
def endpoints_endpoint_login():
    logAPICall('e/endpoint_login')
    request_data = request.get_json()
    endpointID = request_data['endpointID']
    password = request_data['password']
    timestamp = request_data['timestamp']
    result = attemptLogin(endpointID, password, timestamp)

```

```

if(result == "success"):
    name = getNameFromEndpointID(endpointID)
    return jsonify({
        "status": "success",
        "message": name,
        "endpointID": endpointID
    })
else:
    return jsonify({
        "status": "failure",
        "message": "Invalid credentials.",
        "endpointID": endpointID
    })

@app.route('/endpoints/create-endpoint/', methods=['POST'])
def endpoints_create_endpoint():
    logAPICall('e/create-endpoint')
    request_data = request.get_json()
    name = request_data['name']
    password = request_data['password']
    result = createEndpoint(name, password)
    return jsonify({
        "status": "success",
        "message": result,
        "name": name
    })

if __name__ == '__main__':
    app.run()

```

### **dbCommands.py**

```

import mysql.connector
import ahp
import time
from itertools import combinations

db =mysql.connector.connect(host='db-mysql-blr1-22358-do-user-9282953-0.b.db.ondigitalocean.com',
database='defaultdb', user='doadmin', password='***', port='25060', auth_plugin='mysql_native_password')
mycursor = db.cursor()

def dbInsert(statement, data):
    mycursor.execute(statement, data)

def dbSelect(statement):
    mycursor.execute(statement)
    result = mycursor.fetchall()
    return result

```

```

def isDuplicatePNumber(pno):
    mycursor.execute('select * from taskStatus where pno=' + pno + ';')
    result = mycursor.fetchall()
    print(result)
    if result == []:
        return False
    else:
        return True

def isDuplicateSNumber(sno):
    mycursor.execute('select * from sessionDetails where sno=' + sno + ';')
    result = mycursor.fetchall()
    if result == []:
        return False
    else:
        return True

def insertPNumber(pno):
    mycursor.execute('insert into taskStatus values(' + pno + ', \'Pending\');')
    db.commit()

def insertSNumber(sno, endpointID):
    mycursor.execute('insert into sessionDetails values(' + sno + ', \'Pending\', \'' + endpointID + '\');')
    db.commit()

def logAPICall(name, timestamp):
    mycursor.execute('insert into APILogs values(' + name + ', \'' + timestamp + '\');')
    db.commit()

def getFiles(endpointID):
    mycursor.execute('select files from endpointLogs where endpointID=' + str(endpointID) + ';')
    result = mycursor.fetchall()
    return result

def updateFiles(endpointID, filesString):
    mycursor.execute('update endpointLogs set files = ' + str(filesString) + ' where endpointID = ' +
                     str(endpointID) + ';')
    db.commit()

def updateSession(sessionID):
    mycursor.execute('update sessionDetails set status = \'Complete\' where sno = ' + str(sessionID) + ';')
    db.commit()

def insertEC2Logs(index, elapsed, responseCode, responseMessage, success, latency, connect, instanceType,
                  endpointID, sessionID, timestamp):
    mycursor.execute('insert into ec2logs values(' + str(index) + ', ' + str(elapsed) + ', ' + str(responseCode) + ', '
                     + str(responseMessage) + ', ' + str(success) + ', ' + str(latency) + ', ' + str(connect) + ', ' + str(instanceType) + ', '
                     + str(endpointID) + ', ' + str(sessionID) + ', ' + str(timestamp) + ');')
    db.commit()

```

```

def insertEC2Performance(index, cpu, memory, instanceType, endpointID, sessionID, timestamp):
    mycursor.execute('insert into ec2performance values('+str(index)+', '+str(cpu)+', '+str(memory)+',
    '+str(instanceType)+', '+
    +str(endpointID)+', '+str(sessionID)+', '+str(timestamp)+');')
    db.commit()

def insertEC2Summary(index, label, number_of_samples, average, min_val, max_val, std_dev, error_rate, throughput,
received_kbps, sent_kbps, avg_bytes, endpoint_id, session_id, instance_type, timestamp):
    mycursor.execute('insert into ec2summary values('+str(index)+', '+str(label)+', '+str(number_of_samples)+',
    '+str(average)+', '+str(min_val)+', '+str(max_val)+', '+str(std_dev)+', '+str(error_rate)+', '+
    +str(throughput)+', '+str(received_kbps)+', '+str(sent_kbps)+', '+str(avg_bytes)+', '+str(endpoint_id)+',
    '+str(session_id)+', '+str(instance_type)+', '+str(timestamp)+');')
    db.commit()

def check_password(endpoint_id, password):
    mycursor.execute('select password from endpointAccess where endpointID='+str(endpoint_id)+';')
    result = mycursor.fetchall()
    result = str(result)
    result = result[3:-4]
    if(password==result):
        return 1
    else:
        return 0

def log_login(endpoint_id, timestamp):
    mycursor.execute('insert into loginLogs values('+str(endpoint_id)+', '+str(timestamp)+');')
    db.commit()

def get_no_of_samples(instance_type, service_type):
    mycursor.execute('select summarySize from save'+str(service_type)+' where instanceType =
    '+str(instance_type)+';')
    no_result = mycursor.fetchall()
    no_result_list = []
    for i in no_result:
        no_result_list.append(str(i)[2:-3])
    no_result_list = [float(ele) for ele in no_result_list]
    return no_result_list

def get_no_of_performance_samples(instance_type, service_type):
    mycursor.execute('select performanceSize from save'+str(service_type)+' where instanceType =
    '+str(instance_type)+';')
    no_result = mycursor.fetchall()
    no_result_list = []
    for i in no_result:
        no_result_list.append(str(i)[2:-3])
    no_result_list = [float(ele) for ele in no_result_list]
    return no_result_list

```

```

def getOverallThroughput(instanceType, serviceType):
    tpResultList = getThroughputs(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(tpResultList)):
        tpProductsList.append(tpResultList[i]*noResultList[i])
    totalNumberOfSamples = 0
    tpProductsSum = 0
    for ele in range(0, len(noResultList)):
        totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
        tpProductsSum = tpProductsSum + tpProductsList[ele]
    if(len(noResultList) == 0):
        overallThroughput = 0
    else:
        overallThroughput = tpProductsSum / totalNumberOfSamples
    return str(overallThroughput)

def getNumberOfSamples(instanceType, serviceType):
    mycursor.execute('select numberofSamples from '+str(serviceType)+'summary where instanceType = \
    "'"+str(instanceType)+"';")
    noResult = mycursor.fetchall()
    noResultList = []
    for i in noResult:
        noResultList.append(str(i)[2:-3])
    noResultList = [float(ele) for ele in noResultList]
    return noResultList

def getThroughputs(instanceType, serviceType):
    mycursor.execute('select throughput from save'+str(serviceType)+'' where instanceType = \
    "'"+str(instanceType)+"';")
    tpResult = mycursor.fetchall()
    tpResultList = []
    for i in tpResult:
        tpResultList.append(str(i)[2:-3])
    tpResultList = [float(ele) for ele in tpResultList]
    return tpResultList

def getLatency(instanceType, serviceType):
    mycursor.execute('select Latency from '+str(serviceType)+'logs where instanceType = "'"+str(instanceType)+"';")
    result = mycursor.fetchall()
    resultList = []
    for i in result:
        resultList.append(str(i)[2:-3])

    resultList = [float(ele) for ele in resultList]
    if(len(resultList)==0):
        avg = 0
    else:
        avg = sum(resultList)/len(resultList)

```

```

return str(avg)

def getErrorRates(instanceType, serviceType):
    mycursor.execute('select errorRate from save'+str(serviceType)+' where instanceType = \\"'+str(instanceType)+'\\";')
    erResult = mycursor.fetchall()
    erResultList = []
    for i in erResult:
        erResultList.append(str(i)[2:-4])
    erResultList = [float(ele) for ele in erResultList]
    return erResultList

def getOverallErrorRate(instanceType, serviceType):
    erResultList = getErrorRates(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    erProductsList = []
    for i in range(0, len(erResultList)):
        erProductsList.append(erResultList[i]*noResultList[i])
    totalNumberOfSamples = 0
    erProductsSum = 0
    for ele in range(0, len(noResultList)):
        totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
        erProductsSum = erProductsSum + erProductsList[ele]
    if(len(noResultList) == 0):
        overallErrorRate = 0
    else:
        overallErrorRate = erProductsSum / totalNumberOfSamples
    return str(overallErrorRate)

def getElapsed(instanceType, serviceType):
    mycursor.execute('select elapsed from '+str(serviceType)+'.logs where instanceType = \\"'+str(instanceType)+'\\";')
    result = mycursor.fetchall()
    resultList = []
    for i in result:
        resultList.append(str(i)[2:-3])
    resultList = [float(ele) for ele in resultList]
    if(len(resultList)==0):
        avg = 0
    else:
        avg = sum(resultList)/len(resultList)
    return str(avg)

def getElapsedValues(instanceType, serviceType):
    mycursor.execute('select elapsed from save'+str(serviceType)+' where instanceType = \\"'+str(instanceType)+'\\";')
    sdResult = mycursor.fetchall()
    sdResultList = []
    for i in sdResult:
        sdResultList.append(str(i)[2:-3])
    sdResultList = [float(ele) for ele in sdResultList]
    return sdResultList

```

```

def getLatencyValues(instanceType, serviceType):
    mycursor.execute('select Latency from save'+str(serviceType)+' where instanceType = \''+str(instanceType)+'\'')
    sdResult = mycursor.fetchall()
    sdResultList = []
    for i in sdResult:
        sdResultList.append(str(i)[2:-3])
    sdResultList = [float(ele) for ele in sdResultList]
    return sdResultList

def getConnectValues(instanceType, serviceType):
    mycursor.execute('select Connect from save'+str(serviceType)+' where instanceType = \''+str(instanceType)+'\'')
    sdResult = mycursor.fetchall()
    sdResultList = []
    for i in sdResult:
        sdResultList.append(str(i)[2:-3])
    sdResultList = [float(ele) for ele in sdResultList]
    return sdResultList

def getCPUValues(instanceType, serviceType):
    mycursor.execute('select cpu from save'+str(serviceType)+' where instanceType = \''+str(instanceType)+'\'')
    sdResult = mycursor.fetchall()
    sdResultList = []
    for i in sdResult:
        sdResultList.append(str(i)[2:-3])
    sdResultList = [float(ele) for ele in sdResultList]
    return sdResultList

def getMemoryValues(instanceType, serviceType):
    mycursor.execute('select memory from save'+str(serviceType)+' where instanceType = \''+str(instanceType)+'\'')
    sdResult = mycursor.fetchall()
    sdResultList = []
    for i in sdResult:
        sdResultList.append(str(i)[2:-3])
    sdResultList = [float(ele) for ele in sdResultList]
    return sdResultList

def getStoredElapsed(instanceType, serviceType):
    sdResultList = getElapsedValues(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        productsList.append(sdResultList[i]*noResultList[i])
    totalNumberOfSamples = 0
    sdProductsSum = 0
    for ele in range(0, len(noResultList)):
        totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
        sdProductsSum = sdProductsSum + productsList[ele]
    if(len(noResultList) == 0):

```

```

overallElapsed = 0
else:
    overallElapsed = sdProductsSum / totalNumberOfSamples
return str(overallElapsed)

def getStoredLatency(instanceType, serviceType):
    sdResultList = getLatencyValues(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        sdProductsList.append(sdResultList[i]*noResultList[i])
    totalNumberOfSamples = 0
    sdProductsSum = 0
    for ele in range(0, len(noResultList)):
        totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
        sdProductsSum = sdProductsSum + sdProductsList[ele]
    if(len(noResultList) == 0):
        overallLatency = 0
    else:
        overallLatency = sdProductsSum / totalNumberOfSamples
    return str(overallLatency)

def getStoredConnect(instanceType, serviceType):
    sdResultList = getConnectValues(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        sdProductsList.append(sdResultList[i]*noResultList[i])
    totalNumberOfSamples = 0
    sdProductsSum = 0
    for ele in range(0, len(noResultList)):
        totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
        sdProductsSum = sdProductsSum + sdProductsList[ele]
    if(len(noResultList) == 0):
        overallConnect = 0
    else:
        overallConnect = sdProductsSum / totalNumberOfSamples
    return str(overallConnect)

def getStoredCPU(instanceType, serviceType):
    sdResultList = getCPUValues(instanceType, serviceType)
    noResultList = getNoOfPerformanceSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        sdProductsList.append(sdResultList[i]*noResultList[i])
    totalNumberOfSamples = 0
    sdProductsSum = 0
    for ele in range(0, len(noResultList)):
        totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]

```

```

sdProductsSum = sdProductsSum + sdProductsList[ele]
if(len(noResultList) == 0 or totalNumberOfSamples == 0):
    overallCPU = 0
else:
    overallCPU = sdProductsSum / totalNumberOfSamples
return str(overallCPU)

def getStoredMemory(instanceType, serviceType):
    sdResultList = getMemoryValues(instanceType, serviceType)
    noResultList = getNoOfPerformanceSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        sdProductsList.append(sdResultList[i]*noResultList[i])
    totalNumberOfSamples = 0
    sdProductsSum = 0
    for ele in range(0, len(noResultList)):
        totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
        sdProductsSum = sdProductsSum + sdProductsList[ele]
    if(len(noResultList) == 0 or totalNumberOfSamples == 0):
        overallMemory = 0
    else:
        overallMemory = sdProductsSum / totalNumberOfSamples
    return str(overallMemory)

def getStandardDeviations(instanceType, serviceType):
    mycursor.execute('select stdDev from save'+str(serviceType)+'' where instanceType = \''+str(instanceType)+'\'')
    sdResult = mycursor.fetchall()
    sdResultList = []
    for i in sdResult:
        sdResultList.append(str(i)[2:-3])
    sdResultList = [float(ele) for ele in sdResultList]
    return sdResultList

def getOverallStandardDeviation(instanceType, serviceType):
    sdResultList = getStandardDeviations(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        sdProductsList.append(sdResultList[i]*noResultList[i])
    totalNumberOfSamples = 0
    sdProductsSum = 0
    for ele in range(0, len(noResultList)):
        totalNumberOfSamples = totalNumberOfSamples + noResultList[ele]
        sdProductsSum = sdProductsSum + sdProductsList[ele]
    if(len(noResultList) == 0):
        overallStandardDeviation = 0
    else:
        overallStandardDeviation = sdProductsSum / totalNumberOfSamples
    return str(overallStandardDeviation)

```

```

def getConnect(instanceType, serviceType):
    mycursor.execute('select Connect from '+str(serviceType)+'logs where instanceType = "'+str(instanceType)+'";')
    result = mycursor.fetchall()
    resultList = []
    for i in result:
        resultList.append(str(i)[2:-3])
    resultList = [float(ele) for ele in resultList]
    if(len(resultList)==0):
        avg = 0
    else:
        avg = sum(resultList)/len(resultList)
    return str(avg)

def getCPU(instanceType, serviceType):
    mycursor.execute('select cpu from '+str(serviceType)+'performance where instanceType = "'+str(instanceType)+'";')
    result = mycursor.fetchall()
    resultList = []
    for i in result:
        resultList.append(str(i)[2:-3])
    resultList = [float(ele) for ele in resultList]
    if(len(resultList)==0):
        avg = 0
    else:
        avg = sum(resultList)/len(resultList)
    return str(avg)

def getMemory(instanceType, serviceType):
    mycursor.execute('select memory from '+str(serviceType)+'performance where instanceType = "'+str(instanceType)+'";')
    result = mycursor.fetchall()
    resultList = []
    for i in result:
        resultList.append(str(i)[2:-3])
    resultList = [float(ele) for ele in resultList]
    if(len(resultList)==0):
        avg = 0
    else:
        avg = sum(resultList)/len(resultList)
    return str(avg)

def isServiceTypeNotLoaded(serviceType):
    mycursor.execute('select * from derivedValues where serviceType='+'\''+str(serviceType)+'\'')
    result = mycursor.fetchall()
    if result == []:
        return True
    else:
        return False

```

```

def isInstanceTypeNotLoaded(instanceType):
    mycursor.execute('select * from derivedValues where instanceType=' + str(instanceType) + ';')
    result = mycursor.fetchall()
    if result == []:
        return True
    else:
        return False

def insertDerivedValues(elapsed, latency, connect, cpu, memory, standardDeviation, errorRate, throughput,
sessionID, endpointID, instanceType, serviceType, timestamp):
    mycursor.execute('insert into derivedValues values(' + str(elapsed) + '\'', '\'' + str(latency) + '\'', '\'' + str(connect) + '\',
'\'' + str(cpu) + '\'', '\'' + str(memory) + '\'', '\'' + str(standardDeviation) + '\'', '\'' + str(errorRate) + '\'', '\'' + str(throughput) + '\',
'\'' + str(serviceType) + '\'', '\'' + str(timestamp) + '\'', '\'' + str(sessionID) + '\'', '\'' + str(endpointID) + '\');')
    db.commit()

def updateExistingDerivedValues(elapsed, latency, connect, cpu, memory, standardDeviation, errorRate,
throughput, sessionID, endpointID, instanceType, serviceType, timestamp):
    mycursor.execute('update derivedValues set elapsed = ' + str(elapsed) + '\'', 'Latency = ' + str(latency) + '\'', 'Connect =
\' +
    str(connect) + '\'', 'cpu = ' + str(cpu) + '\'', 'memory = ' + str(memory) + '\'', 'stdDev = ' + str(standardDeviation) + '\',
'errorRate = ' + str(errorRate) + '\'', 'Throughput = ' + str(throughput) +
    '\'', 'lastUpdated = ' + str(timestamp) + '\'', 'sessionID = ' + str(sessionID) + '\'', 'endpointID = ' + str(endpointID) +
    '\'', 'where instanceType = ' + str(instanceType) + '\'' + ' and serviceType = ' + str(serviceType) + '\'';')
    db.commit()

def updateDerivedValues(elapsed, latency, connect, cpu, memory, standardDeviation, errorRate, throughput,
sessionID, endpointID, instanceType, serviceType, timestamp):
    serviceTypeFlag = 0
    instanceTypeFlag = 0
    if(isServiceTypeNotLoaded(serviceType)):
        serviceTypeFlag = 1
    if(isInstanceTypeNotLoaded(instanceType)):
        instanceTypeFlag = 1
    if(serviceTypeFlag or instanceTypeFlag):
        insertDerivedValues(elapsed, latency, connect, cpu, memory, standardDeviation, errorRate, throughput,
sessionID, endpointID, instanceType, serviceType, timestamp)
    else:
        updateExistingDerivedValues(elapsed, latency, connect, cpu, memory, standardDeviation, errorRate,
throughput, sessionID, endpointID, instanceType, serviceType, timestamp)

def updateDerivedCalculationLogs(endpointID, sessionID, serviceType, instanceType, timestamp):
    mycursor.execute('insert into derivedCalculationLogs values(' + str(endpointID) + '\'', '\'' + str(sessionID) + '\',
'\'' + str(serviceType) + '\'', '\'' + str(instanceType) + '\'', '\'' + str(timestamp) + '\');')
    db.commit()

def getNameFromEndpointID(endpointID):
    mycursor.execute('select user from endpointData where endpointID=' + str(endpointID) + '\';')

```

```

result = mycursor.fetchall()
result = str(result)[3:-4]
return result

def getLatestID():
    mycursor.execute('select endpointID from endpointData where endpointID = (select max(endpointid) from endpointData)')
    result = mycursor.fetchall()
    if(result == []):
        return 2
    else:
        result = str(result)[3]
        return int(result)+1

def getLatestCount(tableName):
    mycursor.execute('select count(*) from '+str(tableName))
    result = mycursor.fetchall()
    result = str(result)[2:-3]
    if(result == '0'):
        return 0
    else:
        return int(result)+1

def updateEndpointData(id, name):
    mycursor.execute('insert into endpointData values ('+str(id)+', "'+str(name)+');')
    db.commit()

def updateEndpointAccess(id, password):
    mycursor.execute('insert into endpointAccess values ('+str(id)+', "'+str(password)+');')
    db.commit()

def initEndpointLogs(id):
    mycursor.execute('insert into endpointLogs values ('+str(id)+', files = \'\');
```

- db.commit()

```

def createEndpoint(name, password):
    id = getLatestID()
    updateEndpointData(id, name)
    updateEndpointAccess(id, password)
    initEndpointLogs(id)
    return id

def getRetrievedValue(fieldName, serviceType, instanceType):
    mycursor.execute('select '+str(fieldName)+' from derivedValues where serviceType='+'+str(serviceType)+'' and
instanceType = '+'+str(instanceType)+';')
    result = mycursor.fetchall()
    result = str(result)[3:-4]
    end_time = time.time()
    return result

```

```

def insertPRequest(pNo, timestamp, filters, priorities, serviceType, instanceType):
    mycursor.execute('insert into processingRequests values ("'+str(pNo)+'","'+str(filters)+'","'+str(priorities)+'","'+str(serviceType)+'","'+str(instanceType)+'","'+str(timestamp)+')')
    db.commit()

def getRelativeValue(serviceType, firstParam, secondParam, fieldName):
    reversibleValues = ['elapsed', 'stdDev', 'Latency', 'errorRate', 'Connect', 'cpu', 'memory', 'CPU']
    firstVal = getRetrievedValue(fieldName, serviceType, firstParam)
    secondVal = getRetrievedValue(fieldName, serviceType, secondParam)
    result = float(firstVal) / float(secondVal)
    if(fieldName in reversibleValues):
        result = 1/result
    return result

def getRelativeRank(firstParam, secondParam, listOfFilters, priorities):
    firstParamIndex = listOfFilters.index(firstParam)
    firstValue = priorities[firstParamIndex]
    firstValue = 1 / float(firstValue)
    secondParamIndex = listOfFilters.index(secondParam)
    secondValue = priorities[secondParamIndex]
    secondValue = 1 / float(secondValue)
    relativeRank = firstValue / secondValue
    return relativeRank

def ahpCode(serviceType, listOfInstanceTypes, listOfFilters, priorities):
    start_time = time.time()
    comb = combinations(listOfInstanceTypes, 2)
    instancesDict = {}
    dictionaryList = []
    for j in listOfFilters:
        instancesDict.clear()
        comb1 = combinations(listOfInstanceTypes, 2)
        for i in list(comb1):
            instancesDict[i] = getRelativeValue(serviceType, str(i[0]), str(i[1]), str(j))
            dictionaryList.append(instancesDict.copy())
    comb2 = combinations(listOfFilters, 2)
    criteriaDict = {}
    for i in list(comb2):
        criteriaDict[i] = getRelativeRank(str(i[0]), str(i[1]), listOfFilters, priorities)
    end_time = time.time()
    print("processing for ahp code done in:")
    print("--- %s seconds ---" % (end_time - start_time))
    result = ahp.ahpProcessing(dictionaryList, criteriaDict, listOfFilters, listOfInstanceTypes)
    return result

def getFilters(pno):

```

```

mycursor.execute('select filters from processingRequests where pno=' + str(pno) + ';')
result = mycursor.fetchall()
result = str(result)[3:-4]
return result

def getServiceType(pno):
    mycursor.execute('select serviceType from processingRequests where pno=' + str(pno) + ';')
    result = mycursor.fetchall()
    result = str(result)[3:-4]
    return result

def getPriorities(pno):
    mycursor.execute('select priorities from processingRequests where pno=' + str(pno) + ';')
    result = mycursor.fetchall()
    result = str(result)[3:-4]
    return result

def getInstanceType(pno):

    mycursor.execute('select instanceType from processingRequests where pno=' + str(pno) + ';')
    result = mycursor.fetchall()
    result = str(result)[3:-4]
    return result

def updateTaskStatus(pno):
    mycursor.execute('update taskStatus set status = \'Complete\' where pno = ' + str(pno) + ';')
    db.commit()

def insertAHPResult(result, pno, timestamp):
    weights = result
    resultStr = ""
    weights = str(weights)
    resultStr = resultStr + weights
    mycursor.execute('insert into ahpResults values (' + str(pno) + ', ' + resultStr + ', ' + str(timestamp) + ');')
    db.commit()

def updateResults(result, pNo, timestamp):
    updateTaskStatus(pNo)
    if(not isAS3Request(pNo)):
        insertAHPResult(result, pNo, timestamp)

def checkProcessingStatus(pNo):
    mycursor.execute('select status from taskStatus where pno=' + str(pNo) + ';')
    result = mycursor.fetchall()
    result = str(result)[3:-4]
    if result == 'Complete':
        return 1
    else:

```

```

    return 0

def checkIfProcessingIsDone(pNo):
    mycursor.execute('select status from taskStatus where pno='+str(pNo)+';')
    result = mycursor.fetchall()
    result = str(result)[3:-4]
    if result == 'Pending':
        return 1
    else:
        return 0

def isAS3Request(pNo):
    mycursor.execute('select serviceType from processingRequests where pno='+str(pNo)+';')
    result = mycursor.fetchall()
    result = str(result)[3:-4]
    if result == 's3':
        return 1
    else:
        return 0

def saveEC2(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev, errorRate,
throughput, summarySize, timestamp, cpu, memory, performanceSize):
    mycursor.execute('insert into saveec2 values('+str(index)+', '+str(elapsed)+', '+str(latency)+',
'+str(connect)+', '+str(summarySize)+', '+str(stdDev)+', '+str(errorRate)+', '+str(throughput)+',
'+str(performanceSize)+', '+str(cpu)+', '+str(memory)+', '+str(instanceType)+', '+str(endpointID)+',
'+str(sessionID)+', '+str(timestamp)+');')
    db.commit()

def saveRDS(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev,
errorRate, throughput, summarySize, timestamp, cpu, memory, performanceSize):
    mycursor.execute('insert into saverds values('+str(index)+', '+str(elapsed)+', '+str(latency)+',
'+str(connect)+', '+str(summarySize)+', '+str(stdDev)+', '+str(errorRate)+', '+str(throughput)+',
'+str(performanceSize)+', '+str(cpu)+', '+str(memory)+', '+str(instanceType)+', '+str(endpointID)+',
'+str(sessionID)+', '+str(timestamp)+');')
    db.commit()

def saveS3(index, sessionID, endpointID, instanceType, serviceType, elapsed, latency, connect, stdDev, errorRate,
throughput, summarySize, timestamp, cpu, memory, performanceSize):
    mycursor.execute('insert into saves3 values('+str(index)+', '+str(elapsed)+', '+str(latency)+',
'+str(connect)+', '+str(summarySize)+', '+str(stdDev)+', '+str(errorRate)+', '+str(throughput)+',
'+str(performanceSize)+', '+str(cpu)+', '+str(memory)+', '+str(instanceType)+', '+str(endpointID)+',
'+str(sessionID)+', '+str(timestamp)+');')
    db.commit()

def getElapsedForML(instanceType, serviceType):
    sdResultList = getElapsedValues(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
```

```

for j in range(1, int(noResultList[i])):
    sdProductsList.append(sdResultList[i])
return sdProductsList

def getLatencyForML(instanceType, serviceType):
    sdResultList = getLatencyValues(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        for j in range(1, int(noResultList[i])):
            sdProductsList.append(sdResultList[i])
    return sdProductsList

def getConnectForML(instanceType, serviceType):
    sdResultList = getConnectValues(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        for j in range(1, int(noResultList[i])):
            sdProductsList.append(sdResultList[i])
    return sdProductsList

def getStandardDeviationForML(instanceType, serviceType):
    sdResultList = getStandardDeviations(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(sdResultList)):
        for j in range(1, int(noResultList[i])):
            sdProductsList.append(sdResultList[i])
    return sdProductsList

def getErrorRateForML(instanceType, serviceType):
    erResultList = getErrorRates(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(erResultList)):
        for j in range(1, int(noResultList[i])):
            erProductsList.append(erResultList[i])
    return erProductsList

def getThroughputForML(instanceType, serviceType):
    tpResultList = getThroughputs(instanceType, serviceType)
    noResultList = getNoOfSamples(instanceType, serviceType)
    productsList = []
    for i in range(0, len(tpResultList)):
        for j in range(1, int(noResultList[i])):
            tpProductsList.append(tpResultList[i])
    return tpProductsList

```

```
def getResultFromAHP(processingNumber):
    mycursor.execute('select results from ahpResults where pno=' + str(processingNumber) + ';')
    result = mycursor.fetchall()
    result = str(result)[3:-4]
    print(result)
    return result
```

### mlCode.py

```
63 import databaseCommands as db
from pandas import DataFrame
from sklearn import linear_model
import numpy as np
import strings

def startProcessing(serviceType, instanceType, predictorValues, predictorNames):
    inputList = []
    print('predictorValues')
    print(predictorValues)
    print('predictorNames')
    print(predictorNames)
    predictorNames = predictorNames.split(',')
    print(predictorNames)
    elapsed = db.getElapsedForML(instanceType, serviceType)
    if('elapsed' in predictorNames):
        inputList.insert(predictorNames.index('elapsed'), elapsed)
    else:
        lastElement = elapsed
        lastElementName = 'elapsed'
    throughput = db.getThroughputForML(instanceType, serviceType)
    if('Throughput' in predictorNames):
        inputList.insert(predictorNames.index('Throughput'), throughput)
    else:
        lastElement = throughput
        lastElementName = 'Throughput'
    latency = db.getLatencyForML(instanceType, serviceType)
    if('Latency' in predictorNames):
        inputList.insert(predictorNames.index('Latency'), latency)
    else:
        lastElement = latency
        lastElementName = 'Latency'
    connect = db.getConnectForML(instanceType, serviceType)
    if('Connect' in predictorNames):
        inputList.insert(predictorNames.index('Connect'), connect)
    else:
        lastElement = connect
        lastElementName = 'Connect'
    stdDev = db.getStandardDeviationForML(instanceType, serviceType)
```

```

if('stdDev' in predictorNames):
    inputList.insert(predictorNames.index('stdDev'), stdDev)
else:
    lastElement = stdDev
    lastElementName = 'stdDev'
errorRate = db.getErrorRateForML(instanceType, serviceType)
if('errorRate' in predictorNames):
    inputList.insert(predictorNames.index('errorRate'), errorRate)
else:
    lastElement = errorRate
    lastElementName = 'errorRate'
inputList.append(lastElement)
return (inputList, lastElementName)

def runML(serviceType, instanceType, predictorValues, predictorNames):
    res = startProcessing(serviceType, instanceType, predictorValues, predictorNames)
    ipList = res[0]
    lastElement = res[1]
    predictorValues = predictorValues.split(",")
    predictorNames = predictorNames.split ","
    predictorValues = [float(i) for i in predictorValues]
    X = [ipList[:-1]]
    xarray = np.array(ipList[:-1], 'float')
    xarray = np.transpose(xarray)
    y = [ipList[-1]]
    yarray = np.array(ipList[-1], 'float')
    yarray = np.transpose(yarray)
    regr = linear_model.LinearRegression()
    regr.fit(xarray, yarray)
    predictedValue = regr.predict([predictorValues])
    print(predictedValue)
    predictedValueStr = 'For the entered values of '
    selectedValuesDict = {}
    finalDict = {}
    for i in range(0, len(predictorValues)):
        selectedValuesDict[strings.getDisplayName(str(predictorNames[i]))] = round(float(predictorValues[i]), 2)
    selectedValuesDict = str(selectedValuesDict)[1:-1]
    finalSVDict = {}
    finalSVDict["selectedValues"] = {selectedValuesDict}
    predictedValueDict = {}
    finalPVDict = {}
    pv = float(str(predictedValue)[1:-1])
    pvNo = str(round(pv, 2))
    predictedValueDict[strings.getDisplayName(str(lastElement))] = pvNo
    predictedValueDict = str(predictedValueDict)[1:-1]
    finalPVDict["predictedValues"] = {predictedValueDict}
    predictedValueRes = str(finalPVDict)[1:-1]
    selectedValuesRes = str(finalSVDict)[1:-1]
    finalData = selectedValuesRes + ',' + predictedValueRes

```

```
finalData = str(finalData)
finalData = finalData.replace("'", "")
finalData = finalData.replace('"', "'")
finalData = '{' + finalData + '}'
return finalData
```

57

## ahp.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from CompareImpl import Compare
import json
import strings
import time

def ahpProcessing(comparisonsList, criterionList, listOfFilters, listOfInstanceTypes):
    start_time = time.time()
    ahpList = []
    for i in range(0, len(comparisonsList)):
        contentArea = Compare(listOfFilters[i], comparisonsList[i], precision=3, random_index='saaty')
        ahpList.append(contentArea)
    end_time = time.time()
    print("first for loop done in:")
    print("--- %s seconds ---" % (end_time - start_time))
    localWeights = []
    localWeightsDict = {}
    j = 0
    subFilterTextDict = {}
    finalDict = {}
    subDisplayNameTextDict = {}
    criterionWeights = []
    criterionWeightsDict = {}
    filterTextDict = {}
    displayNameTextDict = {}
    criteria = Compare('Ranks', criterionList, precision = 3, random_index='saaty')
    criteria.add_children(ahpList)
    criterionWeights.append(criteria.target_weights)
    criterionWeightsDict["data"] = criteria.target_weights
    filterTextDict["filterCompare"] = strings.getFilterText(criteria.target_weights, "Overall")
    displayNameTextDict[ "displayName"] = "Overall"
    localWeightsRes = str(localWeightsDict)[-1]
    criterionWeightsRes = str(criterionWeightsDict)[1:-1]
    filterTextRes = str(filterTextDict)[1:-1]
    displayNameRes = str(displayNameTextDict)[1:-1]
    finalData = filterTextRes + ',' + criterionWeightsRes + ',' + displayNameRes
    finalDict["overall"] = {finalData[:]}
    start_time=time.time()
    for i in ahpList:
```

```

localWeightsDict["data"] = i.local_weights
subFilterTextDict["filterCompare"] = strings.getFilterText(i.local_weights,
strings.getDisplayName(listOfFilters[j]))
subDisplayNameTextDict["displayName"] = strings.getDisplayName(listOfFilters[j])
localWeightsRes = str(localWeightsDict)[1:-1]
subFilterTextRes = str(subFilterTextDict)[1:-1]
subDisplayNameRes = str(subDisplayNameTextDict)[1:-1]
finalSubData = subFilterTextRes + ',' + localWeightsRes + ',' + subDisplayNameRes
finalDict[listOfFilters[j]] = {finalSubData[:]}
j = j + 1
end_time = time.time()
print("second for loop done in:")
print("--- %s seconds ---" % (end_time - start_time))
finalDict = str(finalDict)
finalDict = finalDict.replace('\"', '')
finalDict = finalDict.replace("'", "'")
return finalDict

```

### strings.py

```

from itertools import combinations

def getFilterText(criterionWeights, filter):
    weightsList = []
    weightsDict = {}
    for i in criterionWeights:
        weightsList.append(criterionWeights[i])
        weightsDict[criterionWeights[i]] = i
    comb = combinations(weightsList, 2)
    finalDict = {}
    for i in list(comb):
        firstValue = float(i[0])
        secondValue = float(i[1])
        res = (firstValue - secondValue) * 100 / secondValue
        if(firstValue>secondValue):
            message = str(weightsDict[i[0]]) + '>' + str(weightsDict[i[1]])
        elif(firstValue<secondValue):
            message = str(weightsDict[i[1]]) + '>' + str(weightsDict[i[0]])
        else:
            continue
        finalDict[message] = str(round(res, 2))
    return finalDict

def getDisplayName(filter):
    if(filter == 'stdDev'):
        return 'Consistency'
    elif(filter == 'errorRate'):
        return 'Error Rate'
    elif(filter == 'elapsed'):

```

```

        return 'Elapsed Time'
    elif(filter == 'Connect'):
        return 'Connection Time'
    elif(filter == 'CPU'):
        return 'CPU Utilization'
    else:
        return filter

```

## DATABASE

```

create table processingRequests(pno varchar(10) primary key, filters varchar(500), priorities
varchar(500), serviceType varchar(500), instanceType varchar(500), timestamp varchar(100));
create table ahpResults(pno varchar(10) primary key, results varchar(10000), timestamp
varchar(100));
create table derivedCalculationLogs(endpointID varchar(30), sessionID varchar(30),
serviceType varchar(30), instanceTy13 varchar(30), timestamp varchar(30) primary key);
create table derivedValues(elapsed varchar(30), Latency varchar13(30), Connect varchar(30), cpu
varchar(30), memory varchar(30), stdDev varchar(30), errorRate varchar(30), Throughput
varchar(30), serviceType varchar(30), instanceType varchar(30), lastUpdated varchar(30)
primary key, sessionID varchar(30), endpointID varchar(30));
create table loginLogs(endpointID varchar(10), timestamp varchar(50) primary key);
create table endpointAccess(endpointID varchar(10) primary key, password varchar(50));
create table sessionDetails(sno varchar(15) primary key, status varchar(15), endpointID
varchar(5));
create table APILogs(callName varchar(30), timest24mp varchar(30) primary key);
create table endpointLogs(endpointID varchar(10) primary key, files varchar(10000));
create table ec2summary(entryID varchar(50) primary key, label varchar(15),
numberOfSamples varchar(15), average varchar(15), min varchar(10), max varchar(10), stdDev
varchar(10), errorRate varchar(30), Throughput varchar(10), receivedKBps varchar(10),
sentKBps varchar(10), averageBytes varchar(15), endpointID varchar(30), sessionID
varchar(10), instanceType varchar(30), timestamp varchar(30));
create table ec2logs(entryID varchar(50) primary key, elapsed varchar(10), responseCode
varchar(300), responseMessage varchar(300), success varchar(10), Latency varchar(10),
Connect varchar(10), instanceType varchar(15), endpointID varchar(10), sessionID varchar(10),
timestamp varchar(30));
create table ec2performance(entryID varchar(50) primary key, cpu varchar(10), memory
varchar(10)24, instanceType varchar(15), endpointID varchar(10), sessionID varchar(10),
timestamp varchar(30));
create table endpointData(endpointID varchar24(5) primary key, user varchar(30));
create table staticContent (name varchar(50) primary key, content varchar(500));
create table taskStatus(pno varchar(15) primary key, status varchar(15));
insert into staticContent values('about-us', 'about-us');
create table saveec2(entryID varchar(50) primary key, elapsed varchar(30), Latency
varchar(30), Connect var13ar(30), summarySize varchar(30), stdDev varchar(30), errorRate
varchar(30), Throughput varchar(30), performanceSize varchar(30), cpu varchar(30), memory

```

```

varchar(30), instanceType varchar(30), endpointID varchar(30), sessionID varchar(10),
timestamp varchar(30));
create table saverds(entryID varchar(50) primary key, elapsed varchar(30), Latency
varchar(30), Connect varchar(30), summarySize varchar(30), stdDev varchar(30), errorRate
varchar(30), Throughput varchar(30), performanceSize varchar(30), cpu varchar(30), memory
varchar(30), instanceType varchar(30), endpointID varchar(30), sessionID varchar(10),
timestamp varchar(30));
create table saves3(entryID varchar(50) primary key, elapsed varchar(30), Latency varchar(30),
Connect varchar(30), summarySize varchar(30), stdDev varchar(30), errorRate varchar(30),
Throughput varchar(30), performanceSize varchar(30), cpu varchar(30), memory varchar(30),
instanceType varchar(30), endpointID varchar(30), sessionID varchar(10), timestamp
varchar(30));

```

## ENDPOINTS

### Ec2\_test.jmx

```

1
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.4.1">
<hashTree>
  <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">
    <stringProp name="TestPlan.comments"></stringProp>
    <boolProp name="TestPlan.functional_mode">false</boolProp>
    <boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
    <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
    <elementProp name="TestPlan.user_defined_variables" elementType="Arguments" guiclass="ArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
      <collectionProp name="Arguments.arguments"/>
    </elementProp>
    <stringProp name="TestPlan.user_define_classpath"></stringProp>
  </TestPlan>
<hashTree>
  <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Users" enabled="true">
    <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
    <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
      <boolProp name="LoopController.continue_forever">false</boolProp>
      <intProp name="LoopController.loops">1</intProp>
    </elementProp>
    <stringProp name="ThreadGroup.num_threads">200</stringProp>
    <stringProp name="ThreadGroup.ramp_time">1</stringProp>
    <boolProp name="ThreadGroup.scheduler">true</boolProp>
    <stringProp name="ThreadGroup.duration">3600</stringProp>
    <stringProp name="ThreadGroup.delay">0</stringProp>
    <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
  </ThreadGroup>
<hashTree>

```

```

<HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="HTTP
Request" enabled="true">
    <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
    </elementProp>
    <stringProp name="HTTPSampler.domain">ec2-15-206-185-135.ap-south-
1.compute.amazonaws.com</stringProp>
    <stringProp name="HTTPSampler.port"></stringProp>
    <stringProp name="HTTPSampler.protocol"></stringProp>
    <stringProp name="HTTPSampler.contentEncoding"></stringProp>
    <stringProp name="HTTPSampler.path">/#</stringProp>
    <stringProp name="HTTPSampler.method">GET</stringProp>
    <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
    <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
    <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
    <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
    <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
    <stringProp name="HTTPSampler.connect_timeout"></stringProp>
    <stringProp name="HTTPSampler.response_timeout"></stringProp>
</HTTPSamplerProxy>
<hashTree>
    <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table"
enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
            <name>saveConfig</name>
            <value class="SampleSaveConfiguration">
                <time>true</time>
                <latency>true</latency>
                <timestamp>true</timestamp>
                <success>true</success>
                <label>true</label>
                <code>true</code>
                <message>true</message>
                <threadName>true</threadName>
                <dataType>true</dataType>
                <encoding>false</encoding>
                <assertions>true</assertions>
                <subresults>true</subresults>
                <responseData>false</responseData>
                <samplerData>false</samplerData>
                <xml>false</xml>
                <fieldNames>true</fieldNames>
                <responseHeaders>false</responseHeaders>
                <requestHeaders>false</requestHeaders>
                <responseDataOnError>false</responseDataOnError>
                <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
                <assertionsResultsToSave>0</assertionsResultsToSave>
            </value>
        </objProp>
    </ResultCollector>
</hashTree>

```

```
<bytes>true</bytes>
<url>true</url>
<threadCounts>true</threadCounts>
<sampleCount>true</sampleCount>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">D:\Jmeter 2
Logs\logs\ec2\t2.nano\ec2_t2.nano_2705212200_1.csv</stringProp>
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
```

```
<hashTree/>
<UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
<stringProp name="ConstantTimer.delay">5</stringProp>
<stringProp name="RandomTimer.range">100.0</stringProp>
<UniformRandomTimer>
<hashTree/>
<kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.vizualizers.PerfMonGui"
testclass="kg.apc.jmeter.perfmon.PerfMonCollector" testname="jp@gc - PerfMon Metrics Collector"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">1000</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
```

```

<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
<collectionProp name="metricConnections">
    <collectionProp name="413873611">
        <stringProp name="1417052243">15.206.185.135</stringProp>9
        <stringProp name="1600768">4444</stringProp>
        <stringProp name="66952">CPU</stringProp>
        <stringProp name="0"></stringProp>
    </collectionProp>
    <collectionProp name="-1886840710">
        <stringProp name="1417052243">15.206.185.135</stringProp>11
        <stringProp name="1600768">4444</stringProp>
        <stringProp name="-1993889503">Memory</stringProp>
        <stringProp name="0"></stringProp>
    </collectionProp>
</collectionProp>
</kg.apc.jmeter.perfmon.PerfMonCollector>
<hashTree/>
<hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
            <boolProp name="LoopController.continue_forever">false</boolProp>
            <intProp name="LoopController.loops">-1</intProp>
        </elementProp>
        <stringProp name="ThreadGroup.num_threads">200</stringProp>
        <stringProp name="ThreadGroup.ramp_time">1</stringProp>
        <boolProp name="ThreadGroup.scheduler">true</boolProp>
        <stringProp name="ThreadGroup.duration">3600</stringProp>
        <stringProp name="ThreadGroup.delay">0</stringProp>
        <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
    </ThreadGroup>
</hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="HTTP
Request" enabled="true">
        <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
            <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">ec2-52-66-210-87.ap-south-
1.compute.amazonaws.com</stringProp>
        <stringProp name="HTTPSampler.port"></stringProp>
        <stringProp name="HTTPSampler.protocol"></stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/#</stringProp>

```

```

2
<stringProp name="HTTPSSampler.method">GET</stringProp>
<boolProp name="HTTPSSampler.follow_redirects">true</boolProp>
<boolProp name="HTTPSSampler.auto_redirects">false</boolProp>
<boolProp name="HTTPSSampler.use_keepalive">true</boolProp>
<boolProp name="HTTPSSampler.DO_MULTIPART_POST">false</boolProp>
<stringProp name="HTTPSSampler.embedded_url_re"></stringProp>
<stringProp name="HTTPSSampler.connect_timeout"></stringProp>
<stringProp name="HTTPSSampler.response_timeout"></stringProp>
</HTTPSSamplerProxy>
<hashTree>
    <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table"
enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
            <name>saveConfig</name>
            <value class="SampleSaveConfiguration">
                <time>true</time>
                <latency>true</latency>
                <timestamp>true</timestamp>
                <success>true</success>
                <label>true</label>
                <code>true</code>
                <message>true</message>
                <threadName>true</threadName>
                <dataType>true</dataType>
                <encoding>false</encoding>
                <assertions>true</assertions>
                <subresults>true</subresults>
                <responseData>false</responseData>
                <samplerData>false</samplerData>
                <xml>false</xml>
                <fieldNames>true</fieldNames>
                <responseHeaders>false</responseHeaders>
                <requestHeaders>false</requestHeaders>
                <responseDataOnError>false</responseDataOnError>
                <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
                <assertionsResultsToSave>0</assertionsResultsToSave>
                <bytes>true</bytes>
                <sentBytes>true</sentBytes>
                <url>true</url>
                <threadCounts>true</threadCounts>
                <sampleCount>true</sampleCount>
                <idleTime>true</idleTime>
                <connectTime>true</connectTime>
                </value>
            </objProp>
            <stringProp name="filename">D:\Jmeter
2
Logs\logs\ec2\t2.micro\ec2_t2.micro_2705212200_1.csv</stringProp>
        </ResultCollector>

```

```

<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
<UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
<stringProp name="ConstantTimer.delay">5</stringProp>
<stringProp name="RandomTimer.range">100.0</stringProp>
<3>UniformRandomTimer>
<hashTree/>
<kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.vizualizers.PerfMonGui"
testclass="kg.apc.jmeter.perfmon.PerfMonCollector" testname="jp@gc - PerfMon Metrics Collector"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>

```

```

<objProp>
  <name>saveConfig</name>
  <value class="SampleSaveConfiguration">
    <time>true</time>
    <latency>true</latency>
    <timestamp>true</timestamp>
    <success>true</success>
    <label>true</label>
    <code>true</code>
    <message>true</message>
    <threadName>true</threadName>
    <dataType>true</dataType>
    <encoding>false</encoding>
    <assertions>true</assertions>
    <subresults>true</subresults>
    <responseData>false</responseData>
    <samplerData>false</samplerData>
    <xml>false</xml>
    <fieldNames>true</fieldNames>
    <responseHeaders>false</responseHeaders>
    <requestHeaders>false</requestHeaders>
    <responseDataOnError>false</responseDataOnError>
    <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
    <assertionsResultsToSave>0</assertionsResultsToSave>
    <bytes>true</bytes>
    <sentBytes>true</sentBytes>
    <url>true</url>
    <threadCounts>true</threadCounts>
    <idleTime>true</idleTime>
    <connectTime>true</connectTime>
  </value>
</objProp>
<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">1000</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
<collectionProp name="metricConnections">
  <collectionProp name="-869074490">
    <stringProp name="-1693625063">52.66.210.87</stringProp>
    9
    <stringProp name="1600768">4444</stringProp>
    <stringProp name="66952">CPU</stringProp>
    <stringProp name="0"></stringProp>
  </collectionProp>
<collectionProp name="1125178485">

```

11

```

<stringProp name="-1693625063">52.66.210.87</stringProp>
<stringProp name="1600768">4444</stringProp>
<stringProp name="-1993889503">Memory</stringProp>
<stringProp name="0"></stringProp>
</collectionProp>
</collectionProp>
</kg.apc.jmeter.perfmon.PerfMonCollector>
<hashTree/>
<hashTree>
<ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
<stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
<elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
<boolProp name="LoopController.continue_forever">false</boolProp>
<intProp name="LoopController.loops">-1</intProp>
</elementProp>
<stringProp name="ThreadGroup.num_threads">200</stringProp>
<stringProp name="ThreadGroup.ramp_time">1</stringProp>
<boolProp name="ThreadGroup.scheduler">true</boolProp>
<stringProp name="ThreadGroup.duration">3600</stringProp>
<stringProp name="ThreadGroup.delay">0</stringProp>
<boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
</ThreadGroup>
<hashTree>
<HTTPSSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSSamplerProxy" testname="HTTP
Request" enabled="true">
<elementProp name="HTTPSampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
<collectionProp name="Arguments.arguments"/>
</elementProp>
<stringProp name="HTTPSampler.domain">ec2-13-127-208-47.ap-south-
1.compute.amazonaws.com</stringProp>
<stringProp name="HTTPSampler.port"></stringProp>
<stringProp name="HTTPSampler.protocol"></stringProp>
<stringProp name="HTTPSampler.contentEncoding"></stringProp>
<stringProp name="HTTPSampler.path">/#</stringProp>
<stringProp name="HTTPSampler.method">GET</stringProp>
<boolProp name="HTTPSampler.follow_redirects">true</boolProp>
<boolProp name="HTTPSampler.auto_redirects">false</boolProp>
<boolProp name="HTTPSampler.use_keepalive">true</boolProp>
<boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
<stringProp name="HTTPSampler.embedded_url_re"></stringProp>
<stringProp name="HTTPSampler.connect_timeout"></stringProp>
<stringProp name="HTTPSampler.response_timeout"></stringProp>
</HTTPSSamplerProxy>
<hashTree>

```

```

<ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table"
enabled="true">
    <boolProp name="ResultCollector.error_logging">false</boolProp>
    <objProp>
        <name>saveConfig</name>
        <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true </saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <url>true</url>
            <threadCounts>true</threadCounts>
            <sampleCount>true</sampleCount>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
            </value>
        </objProp>
        <stringProp name="filename">D:\Jmeter
Logs\logs\ec2\t2.medium\ec2_t2.medium_2705212200_1.csv</stringProp>
    </ResultCollector>
    <hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
    <boolProp name="ResultCollector.error_logging">false</boolProp>
    <objProp>
        <name>saveConfig</name>
        <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>

```

```
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
<UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
<stringProp name="ConstantTimer.delay">5</stringProp>
<stringProp name="RandomTimer.range">100.0</stringProp>
<UniformRandomTimer>
<hashTree/>
<kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.vizualizers.PerfMonGui"
testclass="kg.apc.jmeter.perfmon.PerfMonCollector" testname="jp@gc - PerfMon Metrics Collector"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
```

```

<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">1000</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
<collectionProp name="metricConnections">
  <collectionProp name="1718702701">
    <stringProp name="-758330053">13.127.208.47</stringProp>9
    <stringProp name="1600768">4444</stringProp>
    <stringProp name="66952">CPU</stringProp>
    <stringProp name="0"></stringProp>
  </collectionProp>
  <collectionProp name="-582011620">11
    <stringProp name="-758330053">13.127.208.47</stringProp>
    <stringProp name="1600768">4444</stringProp>
    <stringProp name="-1993889503">Memory</stringProp>
    <stringProp name="0"></stringProp>
  </collectionProp>
</collectionProp>
</kg.apc.jmeter.perfmon.PerfMonCollector>
<hashTree/>
<hashTree>1
</hashTree>

```

```
<ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
  <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
  <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
    <boolProp name="LoopController.continue_forever">false</boolProp>
    <intProp name="LoopController.loops">-1</intProp>
  </elementProp>
  <stringProp name="ThreadGroup.num_threads">200</stringProp>
  <stringProp name="ThreadGroup.ramp_time">1</stringProp>
  <boolProp name="ThreadGroup.scheduler">true</boolProp>
  <stringProp name="ThreadGroup.duration">3600</stringProp>
  <stringProp name="ThreadGroup.delay">0</stringProp>
  <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
</ThreadGroup>
<hashTree>
  <HTTPSSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSSamplerProxy" testname="HTTP
Request" enabled="true">
    <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
      <collectionProp name="Arguments.arguments"/>
    </elementProp>
    <stringProp name="HTTPSampler.domain">ec2-65-2-126-58.ap-south-
1.compute.amazonaws.com</stringProp>
      <stringProp name="HTTPSampler.port"></stringProp>
      <stringProp name="HTTPSampler.protocol"></stringProp>
      <stringProp name="HTTPSampler.contentEncoding"></stringProp>
      <stringProp name="HTTPSampler.path">/#</stringProp>
      <stringProp name="HTTPSampler.method">GET</stringProp>
      <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
      <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
      <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
      <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
      <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
      <stringProp name="HTTPSampler.connect_timeout"></stringProp>
      <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSSamplerProxy>
  <hashTree>
    <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table"
enabled="true">
      <boolProp name="ResultCollector.error_logging">false</boolProp>
      <objProp>
        <name>saveConfig</name>
        <value class="SampleSaveConfiguration">
          <time>true</time>
          <latency>true</latency>
          <timestamp>true</timestamp>
          <success>true</success>
          <label>true</label>
        </value>
      </objProp>
    </ResultCollector>
  </hashTree>
</hashTree>
```

```

<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true </saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<sampleCount>true</sampleCount>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">D:\Jmeter 2
Logs\logs\ec2\t2.large\ec2_t2.large_2705212200_1.csv</stringProp>
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>

```

```
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
<UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
<stringProp name="ConstantTimer.delay">5</stringProp>
<stringProp name="RandomTimer.range">100.0</stringProp>
<UniformRandomTimer>
<hashTree/>
<kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.vizualizers.PerfMonGui"
testclass="kg.apc.jmeter.perfmon.PerfMonCollector" testname="jp@gc - PerfMon Metrics Collector"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
```

```

<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
<long Prop name="interval_grouping">1000</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
<collectionProp name="metricConnections">
<collectionProp name="908413826">
    <stringProp name="-1047607161">65.2.126.58</stringProp>
    9
    <stringProp name="1600768">4444</stringProp>
    <stringProp name="66952">CPU</stringProp>
    <stringProp name="0"></stringProp>
</collectionProp>
<collectionProp name="-1392300495">
    <stringProp name="-1047607161">65.2.126.58</stringProp>
    11
    <stringProp name="1600768">4444</stringProp>
    <stringProp name="-1993889503">Memory</stringProp>
    <stringProp name="0"></stringProp>
</collectionProp>
</collectionProp>
</kg.apc.jmeter.perfmon.PerfMonCollector>
<hashTree/>
<hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
            <boolProp name="LoopController.continue_forever">false</boolProp>
            <intProp name="LoopController.loops">-1</intProp>
</elementProp>
        <stringProp name="ThreadGroup.num_threads">200</stringProp>
        <stringProp name="ThreadGroup.ramp_time">1</stringProp>
        <boolProp name="ThreadGroup.scheduler">true</boolProp>
    </ThreadGroup>
</hashTree>

```

```

<stringProp name="ThreadGroup.duration">3600</stringProp>
<stringProp name="ThreadGroup.delay">0</stringProp>
<boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
</ThreadGroup>
<hashTree>
    <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="HTTP Request" enabled="true">
        <elementProp name="HTTPSampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
            <collectionProp name="Arguments.arguments"/>
        </elementProp>
        <stringProp name="HTTPSampler.domain">ec2-13-232-6-88.ap-south-1.compute.amazonaws.com</stringProp>
        <stringProp name="HTTPSampler.port"></stringProp>
        <stringProp name="HTTPSampler.protocol"></stringProp>
        <stringProp name="HTTPSampler.contentEncoding"></stringProp>
        <stringProp name="HTTPSampler.path">/#</stringProp>
        <stringProp name="HTTPSampler.method">GET</stringProp>
        <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
        <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
        <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
        <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
        <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
        <stringProp name="HTTPSampler.connect_timeout"></stringProp>
        <stringProp name="HTTPSampler.response_timeout"></stringProp>
    </HTTPSamplerProxy>
    <hashTree>
        <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table" enabled="true">
            <boolProp name="ResultCollector.error_logging">false</boolProp>
            <objProp>
                <name>saveConfig</name>
                <value class="SampleSaveConfiguration">
                    <time>true</time>
                    <latency>true</latency>
                    <timestamp>true</timestamp>
                    <success>true</success>
                    <label>true</label>
                    <code>true</code>
                    <message>true</message>
                    <threadName>true</threadName>
                    <dataType>true</dataType>
                    <encoding>false</encoding>
                    <assertions>true</assertions>
                    <subresults>true</subresults>
                    <responseData>false</responseData>
                    <samplerData>false</samplerData>
                    <xml>false</xml>
                    <fieldNames>true</fieldNames>
                </value>
            </objProp>
        </ResultCollector>
    </hashTree>
</hashTree>

```

```
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true </saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<sampleCount>true</sampleCount>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">D:\Jmeter 2
Logs\logs\ec2\t2.small\ec2_t2.small_2705212200_1.csv</stringProp>
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true </saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
```

```
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
<UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
<stringProp name="ConstantTimer.delay">5</stringProp>
<stringProp name="RandomTimer.range">100.0</stringProp>
<UniformRandomTimer>
<hashTree/>
<kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.vizualizers.PerfMonGui"
testclass="kg.apc.jmeter.perfmon.PerfMonCollector" testname="jp@gc - PerfMon Metrics Collector"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
```

```

<stringProp name="filename"></stringProp>
<longProp name="interval_grouping">1000</longProp>
<boolProp name="graph_aggregated">false</boolProp>
<stringProp name="include_sample_labels"></stringProp>
<stringProp name="exclude_sample_labels"></stringProp>
<stringProp name="start_offset"></stringProp>
<stringProp name="end_offset"></stringProp>
<boolProp name="include_checkbox_state">false</boolProp>
<boolProp name="exclude_checkbox_state">false</boolProp>
<collectionProp name="metricConnections">
    <collectionProp name="-88133052">          9
        <stringProp name="-92271679">13.232.6.88</stringProp>
        <stringProp name="1600768">4444</stringProp>
        <stringProp name="66952">CPU</stringProp>
        <stringProp name="0"></stringProp>
    </collectionProp>
    <collectionProp name="1906119923">          11
        <stringProp name="-92271679">13.232.6.88</stringProp>
        <stringProp name="1600768">4444</stringProp>
        <stringProp name="-1993889503">Memory</stringProp>
        <stringProp name="0"></stringProp>
    </collectionProp>
</collectionProp>
</kg.apc.jmeter.perfmon.PerfMonCollector>
<hashTree/>
    <hashTree>
        <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
            <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
            <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
                <boolProp name="LoopController.continue_forever">false</boolProp>
                <intProp name="LoopController.loops">-1</intProp>
            </elementProp>
            <stringProp name="ThreadGroup.num_threads">200</stringProp>
            <stringProp name="ThreadGroup.ramp_time">1</stringProp>
            <boolProp name="ThreadGroup.scheduler">true</boolProp>
            <stringProp name="ThreadGroup.duration">3600</stringProp>
            <stringProp name="ThreadGroup.delay">0</stringProp>
            <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
        </ThreadGroup>
        <hashTree>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="HTTP
Request" enabled="true">
                <elementProp name="HTTPSampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
                    <collectionProp name="Arguments.arguments"/>
                </elementProp>

```

```
<stringProp name="HTTPSampler.domain">ec2-13-126-213-152.ap-south-1.compute.amazonaws.com</stringProp>
<stringProp name="HTTPSampler.port"></stringProp>
<stringProp name="HTTPSampler.protocol"></stringProp>
<stringProp name="HTTPSampler.contentEncoding"></stringProp>
<stringProp name="HTTPSampler.path">/#</stringProp>
<stringProp name="HTTPSampler.method">GET</stringProp>
<boolProp name="HTTPSampler.follow_redirects">true</boolProp>
<boolProp name="HTTPSampler.auto_redirects">false</boolProp>
<boolProp name="HTTPSampler.use_keepalive">true</boolProp>
<boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
<stringProp name="HTTPSampler.embedded_url_re"></stringProp>
<stringProp name="HTTPSampler.connect_timeout"></stringProp>
<stringProp name="HTTPSampler.response_timeout"></stringProp>
</HTTPSamplerProxy>
<hashTree>
    <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
            <name>saveConfig</name>
            <value class="SampleSaveConfiguration">
                <time>true</time>
                <latency>true</latency>
                <timestamp>true</timestamp>
                <success>true</success>
                <label>true</label>
                <code>true</code>
                <message>true</message>
                <threadName>true</threadName>
                <dataType>true</dataType>
                <encoding>false</encoding>
                <assertions>true</assertions>
                <subresults>true</subresults>
                <responseData>false</responseData>
                <samplerData>false</samplerData>
                <xml>false</xml>
                <fieldNames>true</fieldNames>
                <responseHeaders>false</responseHeaders>
                <requestHeaders>false</requestHeaders>
                <responseDataOnError>false</responseDataOnError>
                <saveAssertionResultsFailureMessage>true </saveAssertionResultsFailureMessage>
                <assertionsResultsToSave>0</assertionsResultsToSave>
                <bytes>true</bytes>
                <sentBytes>true</sentBytes>
                <url>true</url>
                <threadCounts>true</threadCounts>
                <sampleCount>true</sampleCount>
                <idleTime>true</idleTime>
            
        
    
```

```

<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">D:\Jmeter
Logs\logs\ec2\t2.xlarge\ec2_t2.xlarge_2705212200_1.csv</stringProp>2
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
<UniformRandomTimer guiclass="UniformRandomTimerGui" testclass="UniformRandomTimer"
testname="Uniform Random Timer" enabled="true">
<stringProp name="ConstantTimer.delay">5</stringProp>
<stringProp name="RandomTimer.range">100.0</stringProp>

```

```
</UniformRandomTimer>
<hashTree/>
3 <kg.apc.jmeter.perfmon.PerfMonCollector guiclass="kg.apc.jmeter.vizualizers.PerfMonGui"
testclass="kg.apc.jmeter.perfmon.PerfMonCollector" testname="jp@gc - PerfMon Metrics Collector"
enabled="true">
    <boolProp name="ResultCollector.error_logging">false</boolProp>
    <objProp>
        <name>saveConfig</name>
        <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <url>true</url>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
        </value>
    </objProp>
    <stringProp name="filename"></stringProp>
    <long Prop name="interval_grouping">1000</longProp>
    <boolProp name="graph_aggregated">false</boolProp>
    <stringProp name="include_sample_labels"></stringProp>
    <stringProp name="exclude_sample_labels"></stringProp>
    <stringProp name="start_offset"></stringProp>
    <stringProp name="end_offset"></stringProp>
    <boolProp name="include_checkbox_state">false</boolProp>
    <boolProp name="exclude_checkbox_state">false</boolProp>
    <collectionProp name="metricConnections">
        <collectionProp name="1168561953">
```

```

9
<stringProp name="-201932049">13.126.213.152</stringProp>
<stringProp name="1600768">4444</stringProp>
<stringProp name="66952">CPU</stringProp>
<stringProp name="0"></stringProp>
</collectionProp>
<collectionProp name="-1132152368">
11
<stringProp name="-201932049">13.126.213.152</stringProp>
<stringProp name="1600768">4444</stringProp>
<stringProp name="-1993889503">Memory</stringProp>
<stringProp name="0"></stringProp>
</collectionProp>
</collectionProp>
<!--g.apc.jmeter.perfmon.PerfMonCollector>
<hashTree/>
</hashTree>
</hashTree>
</hashTree>
</hashTree>
</jmeterTestPlan>

```

### Rds\_Test.jmx

```

<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.4.1">
<hashTree>
<TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">
<stringProp name="TestPlan.comments"></stringProp>
<boolProp name="TestPlan.functional_mode">false</boolProp>
<boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
<boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
<elementProp name="TestPlan.user_defined_variables" elementType="Arguments" guiclass="ArgumentsPanel"
testclass="Arguments" testname="User Defined Variables" enabled="true">
<collectionProp name="Arguments.arguments"/>
</elementProp>
<stringProp name="TestPlan.user_define_classpath"></stringProp>
</TestPlan>
<hashTree>
<JDBCDataSource guiclass="TestBeanGUI" testclass="JDBCDataSource" testname="JDBC Connection
Configuration" enabled="true">
<stringProp name="dataSource">Variable1</stringProp>
<stringProp name="poolMax">50</stringProp>
<stringProp name="timeout">10000</stringProp>
<stringProp name="trimInterval">60000</stringProp>
<boolProp name="autocommit">true</boolProp>
<stringProp name="transactionIsolation">DEFAULT</stringProp>
<boolProp name="preinit">false</boolProp>
<stringProp name="initQuery"></stringProp>
<boolProp name="keepAlive">true</boolProp>
<stringProp name="connectionAge">5000</stringProp>

```

71

```
<stringProp name="checkQuery">select 1</stringProp>
<stringProp name="dbUrl">jdbc:mysql://test-database.cjamazw1gpow.ap-south-
1.rds.amazonaws.com:3306/RDS_Test</stringProp> 16
<stringProp name="driver">com.mysql.jdbc.Driver</stringProp>
<stringProp name="username">admin12345</stringProp>
<stringProp name="password">admin12345</stringProp>
<stringProp name="connectionProperties"></stringProp>
</JDBCDataSource>
<hashTree/>
<JDBCDataSource guiclass="TestBeanGUI" testclass="JDBCDataSource" testname="JDBC Connection
Configuration" enabled="true">
<stringProp name="dataSource">Variable2</stringProp>
<stringProp name="poolMax">50</stringProp>
<stringProp name="timeout">10000</stringProp>
<stringProp name="trimInterval">60000</stringProp>
<boolProp name="autocommit">true</boolProp>
<stringProp name="transactionIsolation">DEFAULT</stringProp>
<boolProp name="preinit">false</boolProp>
<stringProp name="initQuery"></stringProp>
<boolProp name="keepAlive">true</boolProp>
<stringProp name="connectionAge">5000</stringProp>
<stringProp name="checkQuery">select 1</stringProp>
<stringProp name="dbUrl">jdbc:mysql://test-database-small.cjamazw1gpow.ap-south-
1.rds.amazonaws.com:3306/RDS_Test_Small</stringProp> 16
<stringProp name="driver">com.mysql.jdbc.Driver</stringProp>
<stringProp name="username">admin12345</stringProp>
<stringProp name="password">admin12345</stringProp>
<stringProp name="connectionProperties"></stringProp>
</JDBCDataSource>
<hashTree/>
<JDBCDataSource guiclass="TestBeanGUI" testclass="JDBCDataSource" testname="JDBC Connection
Configuration" enabled="true">
<stringProp name="dataSource">Variable3</stringProp>
<stringProp name="poolMax">50</stringProp>
<stringProp name="timeout">10000</stringProp>
<stringProp name="trimInterval">60000</stringProp>
<boolProp name="autocommit">true</boolProp>
<stringProp name="transactionIsolation">DEFAULT</stringProp>
<boolProp name="preinit">false</boolProp>
<stringProp name="initQuery"></stringProp>
<boolProp name="keepAlive">true</boolProp>
<stringProp name="connectionAge">5000</stringProp>
<stringProp name="checkQuery">select 1</stringProp>
<stringProp name="dbUrl">jdbc:mysql://test-database-medium.cjamazw1gpow.ap-south-
1.rds.amazonaws.com:3306/RDS_Test_Medium</stringProp>
<stringProp name="driver">com.mysql.jdbc.Driver</stringProp>
<stringProp name="username">admin12345</stringProp>
<stringProp name="password">admin12345</stringProp>
<stringProp name="connectionProperties"></stringProp>
```

```

</JDBCDataSource>
<hashTree/>
1 <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
    <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
    <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <intProp name="LoopController.loops">-1</intProp>
    </elementProp>
    <stringProp name="ThreadGroup.num_threads">5000</stringProp>
    <stringProp name="ThreadGroup.ramp_time">1</stringProp>
    <boolProp name="ThreadGroup.scheduler">true</boolProp>
    <stringProp name="ThreadGroup.duration">600</stringProp>
    <stringProp name="ThreadGroup.delay">0</stringProp>
    <boolProp name="ThreadGroup.same_user_on_next_iteration">true</boolProp>
</ThreadGroup>
<hashTree/>
    <JDBCSSampler guiclass="TestBeanGUI" testclass="JDBCSSampler" testname="JDBC Request"
enabled="true">
        16
        <stringProp name="dataSource">Variable1</stringProp>
        <stringProp name="queryType">Select Statement</stringProp>
        <stringProp name="query">select * from student</stringProp>
        <stringProp name="queryArguments"></stringProp>
        <stringProp name="queryArgumentsTypes"></stringProp>
        <stringProp name="variableNames"></stringProp>33
        <stringProp name="resultVariable">Result1</stringProp>
        <stringProp name="queryTimeout"></stringProp>
        <stringProp name="resultSetMaxRows"></stringProp>
        <stringProp name="resultSetHandler">Store as String</stringProp>
</JDBCSSampler>
<hashTree/>
    <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table"
enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
            <name>saveConfig</name>
            <value class="SampleSaveConfiguration">
                <time>true</time>
                <latency>true</latency>
                <timestamp>true</timestamp>
                <success>true</success>
                <label>true</label>
                <code>true</code>
                <message>true</message>
                <threadName>true</threadName>
                <dataType>true</dataType>
                <encoding>false</encoding>
                <assertions>true</assertions>

```

```
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">D:\Jmeter
Logs\logs\rds\db.t2.micro\rds_db.t2.micro_2705212330_1.csv</stringProp>2
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestampl>true</timestampl>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
```

```

<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
</hashTree>
<ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
    1
    <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
    <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <intProp name="LoopController.loops">1</intProp>
    </elementProp>
    <stringProp name="ThreadGroup.num_threads">5000</stringProp>
    <stringProp name="ThreadGroup.ramp_time">1</stringProp>
    <boolProp name="ThreadGroup.scheduler">true</boolProp>
    <stringProp name="ThreadGroup.duration">600</stringProp>
    <stringProp name="ThreadGroup.delay">0</stringProp>
    <boolProp name="ThreadGroup.same_user_on_next_iteration">true</boolProp>
</ThreadGroup>
<hashTree>
    <JDBCSampler guiclass="TestBeanGUI" testclass="JDBCSampler" testname="JDBC Request"
enabled="true">
        16
        <stringProp name="dataSource">Variable2</stringProp>
        <stringProp name="queryType">Select Statement</stringProp>
        <stringProp name="query">select * from student</stringProp>
        <stringProp name="queryArguments"></stringProp>
        <stringProp name="queryArgumentsTypes"></stringProp>
        <stringProp name="variableNames"></stringProp>
        33
        <stringProp name="resultVariable">Result2</stringProp>
        <stringProp name="queryTimeout"></stringProp>
        <stringProp name="resultSetMaxRows"></stringProp>
        <stringProp name="resultSetHandler">Store as String</stringProp>
    <1 JDBCSampler>
    <hashTree/>
    <ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table"
enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
            <name>saveConfig</name>
            <value class="SampleSaveConfiguration">
                <time>true</time>

```

```
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">D:\Jmeter
Logs\logs\rds\db.t2.small\rds_db.t2.small_2705212330_1.csv</stringProp>
2
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
```

```

<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
</hashTree>
<ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group"
enabled="true">
    <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
    <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller" enabled="true">
        <boolProp name="LoopController.continue_forever">false</boolProp>
        <intProp name="LoopController.loops">-1</intProp>
    </elementProp>
    <stringProp name="ThreadGroup.num_threads">5000</stringProp>
    <stringProp name="ThreadGroup.ramp_time">1</stringProp>
    <boolProp name="ThreadGroup.scheduler">true</boolProp>
    <stringProp name="ThreadGroup.duration">600</stringProp>
    <stringProp name="ThreadGroup.delay">0</stringProp>
    <boolProp name="ThreadGroup.same_user_on_next_iteration">true</boolProp>
</ThreadGroup>
<hashTree>
    <JDBCSampler guiclass="TestBeanGUI" testclass="JDBCSampler" testname="JDBC Request"
enabled="true">
        <stringProp name="dataSource">Variable3</stringProp>
        <stringProp name="queryType">Select Statement</stringProp>
        <stringProp name="query">select * from student</stringProp>
        <stringProp name="queryArguments"></stringProp>
        <stringProp name="queryArgumentsTypes"></stringProp>
        <stringProp name="variableNames"></stringProp>
        <stringProp name="resultVariable">Result3</stringProp>
        <stringProp name="queryTimeout"></stringProp>
        <stringProp name="resultSetMaxRows"></stringProp>
    </JDBCSampler>
</hashTree>

```

```

<stringProp name="resultSetHandler">Store as String</stringProp>
<1><DBCSampler>
<hashTree/>
<ResultCollector guiclass="TableVisualizer" testclass="ResultCollector" testname="View Results in Table"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">D:\Jmeter
Logs\logs\rds\db.t2.medium\rds_db.t2.medium_2705212330_1.csv</stringProp>
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>

```

```

<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
</hashTree>
</hashTree>
</hashTree>
</jmeterTestPlan>

```

### S3\_Test.jmx

```

<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.4.1">
<hashTree>
  <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">
    <stringProp name="TestPlan.comments"></stringProp>
    <boolProp name="TestPlan.functional_mode">false</boolProp>
    <boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
    <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
    <elementProp name="TestPlan.user_defined_variables" elementType="Arguments" guiclass="ArgumentsPanel"
      testclass="Arguments" testname="User Defined Variables" enabled="true">
      <collectionProp name="Arguments.arguments"/>
    
```



```

<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>
<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename">D:\Jmeter Logs\logs\s3\s3_2705211845_1.csv</stringProp>
</ResultCollector>
<hashTree/>
<ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary Report"
enabled="true">
<boolProp name="ResultCollector.error_logging">false</boolProp>
<objProp>
<name>saveConfig</name>
<value class="SampleSaveConfiguration">
<time>true</time>
<latency>true</latency>
<timestamp>true</timestamp>
<success>true</success>
<label>true</label>
<code>true</code>
<message>true</message>
<threadName>true</threadName>
<dataType>true</dataType>
<encoding>false</encoding>
<assertions>true</assertions>
<subresults>true</subresults>

```

```

<responseData>false</responseData>
<samplerData>false</samplerData>
<xml>false</xml>
<fieldNames>true</fieldNames>
<responseHeaders>false</responseHeaders>
<requestHeaders>false</requestHeaders>
<responseDataOnError>false</responseDataOnError>
<saveAssertionResultsFailureMessage>true </saveAssertionResultsFailureMessage>
<assertionsResultsToSave>0</assertionsResultsToSave>
<bytes>true</bytes>
<sentBytes>true</sentBytes>
<url>true</url>
<threadCounts>true</threadCounts>
<idleTime>true</idleTime>
<connectTime>true</connectTime>
</value>
</objProp>
<stringProp name="filename"></stringProp>
</ResultCollector>
<hashTree/>
</hashTree>
</hashTree>
</hashTree>
</hashTree>
</jmeterTestPlan>

```

### Endpoints.py

54

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

```

```

import csv
import pandas as pd
import json
import os
import requests
import datetime
import time
fileName = 'data/Log_11052021.csv'
path = './bigLogsDir/'
endpointID = 2
separator = '/'
url = 'http://cloud-service-optimizer-dev.herokuapp.com/'

def attemptLogin(eID, pwd):
    headersPost = {'Content-type': 'application/json'}
    timestamp = datetime.datetime.now()
    postData = { "endpointID": str(endpointID), "password": str(pwd), "timestamp": str(timestamp)}

```

```

postData = json.dumps(postData)
responseFile = requests.post(url+'endpoints/endpoint-login/', data = postData, headers = headersPost)
responseFile = json.loads(responseFile.text)
if(responseFile['status'] == 'success'):
    return (1, responseFile['message'])
else:
    return (0, responseFile['message'])

def attemptSignup(name, password):
    headersPost = {'Content-type': 'application/json'}
    postData = { "name": str(name), "password": str(password)}
    postData = json.dumps(postData)
    responseFile = requests.post(url+'endpoints/create-endpoint/', data = postData, headers = headersPost)
    responseFile = json.loads(responseFile.text)
    if(responseFile['status'] == 'success'):
        return (1, responseFile['message'], responseFile['name'])
    else:
        return (0, responseFile['message'], responseFile['name'])

loginFailure = 1

while(loginFailure):
    inputFailure = 1
    while(inputFailure):
        loginOrSignup = input('Press 1 for logging in. Press 0 for creating/registering an endpoint:\n')
        if(not loginOrSignup == '1' and not loginOrSignup == '0'):
            print("Wrong input. Please try again.")
        else:
            inputFailure = 0
    if(loginOrSignup == '1'):
        eID = input("Enter endpoint ID:\n")
        pwd = input("Enter password:\n")
        print("Checking login credentials...")
        loginResult = attemptLogin(eID, pwd)
        if(loginResult[0]):
            print("Successfully logged in to "+str(loginResult[1])+"!\n")
            pathFailure = 1
            while(pathFailure):
                separatorCheckFailure = 1
                while(separatorCheckFailure):
                    separatorCheck = input('If Windows, press 1. If non-Windows, press 0.')
                    if(separatorCheck == '0' or separatorCheck == '1'):
                        separatorCheckFailure = 0
                    else:
                        print('Invalid entry. Please try again.')
                if(separatorCheck == '1'):
                    separator = '\\\\' + '\\'
                else:
                    separator = '/'

```

```
enteredPath = input('Enter the path where the logs are stored:')
separatorValid = '/'
if(separatorCheck == '1'):
    separatorValid = '\\'
if(not enteredPath[-1] == separatorValid):
    enteredPath = enteredPath + separator
if(os.path.isdir(enteredPath)):
    pathFailure = 0
else:
    print('The entered directory does not exist. Please try again.')
    loginFailure = 0
else:
    print("Invalid credentials. Try again.")
else:
    name = input('Please enter your name:\n')
    passwordsDoNotMatch = 1
    while(passwordsDoNotMatch):
        password = input('Please create your password:\n')
        confirmPassword = input('Please confirm your password:\n')
        if(password == confirmPassword):
            passwordsDoNotMatch = 0
        else:
            print("Passwords do not match. Try again.")
    signupResult = attemptSignup(name, password)
    if(signupResult[0]):
        print("Successfully signed up user "+str(signupResult[2]))
        print("Your endpoint ID to log in is: "+str(signupResult[1]))
    print("Login with created credentials:")
    eID = input("Enter endpoint ID:\n")
    pwd = input("Enter password:\n")
    print("Checking login credentials...")
    loginResult = attemptLogin(eID, pwd)
    if(loginResult[0]):
        print("Successfully logged in to "+str(loginResult[1])+"!\n")
        pathFailure = 1
        while(pathFailure):
            separatorCheckFailure = 1
            while(separatorCheckFailure):
                separatorCheck = input('If Windows, press 1. If non-Windows, press 0.')
                if(separatorCheck == '0' or separatorCheck == '1'):
                    separatorCheckFailure = 0
                else:
                    print('Invalid entry. Please try again.')
            if(separatorCheck == '1'):
                separator = '\\\\' + '\\\\'
            else:
                separator = '\\'
    enteredPath = input('Enter the path where the logs are stored:')
    separatorValid = '/'
```

```

if(separatorCheck == '1'):
    separatorValid = '\\'
if(not enteredPath[-1] == separatorValid):
    enteredPath = enteredPath + separator
if(os.path.isdir(enteredPath)):
    pathFailure = 0
else:
    print("The entered directory does not exist. Please try again.")
    loginFailure = 0
else:
    print("Invalid credentials. Try again.")
endpointID = eID
path = enteredPath

def getFilesInServer(endpointID):
    headersPost = {'Content-type': 'application/json'}
    postData = { "endpointID": str(endpointID)}
    postData = json.dumps(postData)
    responseFile = requests.post(url+'endpoints/get-files/', data = postData, headers = headersPost)
    responseFile = json.loads(responseFile.text)
    if(responseFile['status'] == 'success'):
        strFiles = str(responseFile['message'])
        strFiles = strFiles[3:]
        strFiles = strFiles[:-3]
        strFiles = strFiles[:]
        return strFiles
    else:
        return ""

def initiateLogging(endpointID):
    ploads = {'endpointID': str(endpointID)}
    responseFile = requests.get(url+'endpoints/initiate-logging', params = ploads)
    responseFile = json.loads(responseFile.text)
    if(responseFile['status'] == "success"):
        return responseFile['message']

filesInServer = getFilesInServer(endpointID)
sessionID = initiateLogging(endpointID)
print("Your login session ID is:"+str(sessionID))

def uploadLogFile(isPerformanceIncluded, performanceJSON, logsJSON, summaryJSON, sessionID, endpointID,
serviceType, instanceType):
    headersPost = {'Content-type': 'application/json'}
    timestamp = datetime.datetime.now()
    postData = {}
    if(isPerformanceIncluded):
        postData = { "sessionID": sessionID, "endpointID": endpointID, "serviceType": serviceType, "instanceType": instanceType, "elapsed": str(logsJSON[0]), "latency": str(logsJSON[1]), "connect": str(logsJSON[2]), "stdDev": str(summaryJSON[0]), "errorRate": str(summaryJSON[1]), "throughput": str(summaryJSON[2]), "summarySize": str(summaryJSON[3])}

```

```

str(summaryJSON[3]), "timestamp": str(timestamp), "isPerformanceIncluded": '1', "cpu": str(performanceJSON[0]),
"memory": str(performanceJSON[1]), "performanceSize": str(performanceJSON[2])}
else:
    postData = { "sessionID": sessionID, "endpointID": endpointID, "serviceType": serviceType, "instanceType": instanceType, "elapsed": str(logsJSON[0]), "latency": str(logsJSON[1]), "connect": str(logsJSON[2]), "stdDev": str(summaryJSON[0]), "errorRate": str(summaryJSON[1]), "throughput": str(summaryJSON[2]), "summarySize": str(summaryJSON[3]), "timestamp": str(timestamp), "isPerformanceIncluded": '0', "cpu": "", "memory": "", "performanceSize": ""}

    postData = json.dumps(postData)
    start_time = time.time()
    response = requests.post(url+endpoints/post-log-data/, data = postData, headers=headersPost)
    end_time = time.time()
    print("Upload done in:")
    print("--- %s seconds ---" % (end_time - start_time))

def getElapsed(df):
    return df.mean()

def getLatency(df):
    return df.mean()

def getConnect(df):
    return df.mean()

def readLogFile(filePath):
    df = pd.read_csv(filePath, usecols = ['elapsed', 'responseCode', 'responseMessage', 'success', 'Latency', 'Connect'])
    df = df[1:]
    elapsed = getElapsed(df["elapsed"])
    latency = getLatency(df["Latency"])
    connect = getConnect(df["Connect"])
    return(elapsed, latency, connect)
    json_str = df.to_json()
    return json_str

def readSummaryFile(filePath): 47
    df = pd.read_csv(filePath, usecols = ['Label', '# Samples', 'Average', 'Min', 'Max', 'Std. Dev.', 'Error %',
'Throughput', 'Received KB/sec', 'Sent KB/sec', 'Avg. Bytes'])
    df = df.iloc[0]
    sizeOfFile = df['# Samples']
    stdDev = df['Std. Dev.']
    errorRate = df['Error %']
    throughput = df['Throughput']
    return (stdDev, errorRate, throughput, sizeOfFile)
    json_str = df.to_json()
    return json_str

def readPerformanceFile(filePath):
    df = pd.read_csv(filePath, usecols = ['CPU', 'Memory'])
    df = df[1:]

```

```

sizeOfFile = len(df.index)
cpu = df["CPU"].mean()
memory = df["Memory"].mean()
return (cpu, memory, sizeOfFile)

def updateFilesListInServer(filesList, endpointID, sessionID):
    headersPost = {'Content-type': 'application/json'}
    postData = { "sessionID": sessionID, "endpointID": endpointID, "filesString": filesList}
    postData = json.dumps(postData)
    response = requests.post(url+endpoints/update-files/, data = postData, headers=headersPost)

filesList = filesInServer

flag = 0
print('Going through files...')
for serviceType in os.listdir(path):
    if os.path.isdir(path+serviceType) and (serviceType == 'ec2' or serviceType == 'rds'):
        subpath = path+serviceType
        for instanceType in os.listdir(subpath):
            if os.path.isdir(subpath + separator + instanceType):
                suppPath = subpath + separator + instanceType
                for filename in os.listdir(suppPath):
                    if os.path.isfile(suppPath+separator+filename) and filename.endswith('_l.csv'):
                        filePath = suppPath+separator+filename
                        summaryFile = filename[:-5] + 's.csv'
                        performanceFile = filename[:-5] + 'p.csv'
                        summaryPath = suppPath + separator + summaryFile
                        performancePath = suppPath + separator + performanceFile
                        if filesList.find(filePath) == -1:
                            print("Currently uploading file:")
                            print(filePath)
                            logsJSON = readLogFile(filePath)
                            summaryJSON = readSummaryFile(summaryPath)
                            if(os.path.exists(performancePath)):
                                performanceJSON = readPerformanceFile(performancePath)
                                uploadLogFile(1, performanceJSON, logsJSON, summaryJSON, sessionID, endpointID,
serviceType, instanceType)
                            else:
                                uploadLogFile(0, "", logsJSON, summaryJSON, sessionID, endpointID, serviceType,
instanceType)
                            print('File uploaded successfully!')
                            filesList = filesList + filePath + ','
                            flag = 1
            elif(os.path.isdir(path+serviceType) and (serviceType == 's3')):
                subpath = path+serviceType
                for filename in os.listdir(subpath):
                    if os.path.isfile(subpath+separator+filename) and filename.endswith('_l.csv'):
                        filePath = subpath+separator+filename

```

```
summaryFile = filename[:-5] + 's.csv'
summaryPath = subpath + separator + summaryFile
if filesList.find(filePath) == -1:
    print("Currently uploading s3 file:")
    print(filePath)
    logsJSON = readLogFile(filePath)
    summaryJSON = readSummaryFile(summaryPath)
    uploadLogFile(0, "", logsJSON, summaryJSON, sessionID, endpointID, serviceType, 'na')
    print('File uploaded successfully!')
    filesList = filesList + filePath + ','
    flag = 1
    print('Read the s3 file')
if(flag):
    filesList = filesList[:-2]
    print('Uploaded files list:' + filesList)
else:
    print("No new files to be uploaded.")
updateFilesListInServer(filesList, endpointID, sessionID)
```

# Report-Poornima B

## ORIGINALITY REPORT



## PRIMARY SOURCES

1	<b>Submitted to The Robert Gordon University</b> Student Paper	<b>5%</b>
2	<b>pastebin.com</b> Internet Source	<b>4%</b>
3	<b>openaccess.maltepe.edu.tr</b> Internet Source	<b>2%</b>
4	<b>www.telegraaf.nl</b> Internet Source	<b>1%</b>
5	<b>fidodesign.net</b> Internet Source	<b>1%</b>
6	<b>svn.apache.org</b> Internet Source	<b>&lt;1%</b>
7	<b>www.johnsnowlabs.com</b> Internet Source	<b>&lt;1%</b>
8	<b>Robert Maeser. "Analyzing CSP Trustworthiness and Predicting Cloud Service Performance", IEEE Open Journal of the Computer Society, 2020</b> Publication	<b>&lt;1%</b>

---

9	gitlab.rm.ingv.it Internet Source	<1 %
10	Submitted to Sim University Student Paper	<1 %
11	www.jmeter-archive.org Internet Source	<1 %
12	stackoverflow.com Internet Source	<1 %
13	Submitted to University of Technology, Sydney Student Paper	<1 %
14	Yubiao Wang, Junhao Wen, Wei Zhou, Bamei Tao, Quanwang Wu, Zhiyong Tao. "A Cloud Service Selection Method Based on Trust and User Preference Clustering", IEEE Access, 2019 Publication	<1 %
15	Ramcharan Kakarla, Sundar Krishnan, Sridhar Alla. "Applied Data Science Using PySpark", Springer Science and Business Media LLC, 2021 Publication	<1 %
16	nsuworks.nova.edu Internet Source	<1 %
17	vibdoc.com Internet Source	<1 %

18	Submitted to CSU, San Jose State University Student Paper	<1 %
19	www.zdnet.com Internet Source	<1 %
20	Nabendu Biswas. "Foundation Gatsby Projects", Springer Science and Business Media LLC, 2021 Publication	<1 %
21	jmeter.apache.org Internet Source	<1 %
22	Submitted to NorthTec Student Paper	<1 %
23	Sarvesh Pandey, A.K. Daniel. "Fuzzy logic based cloud service trustworthiness model", 2016 IEEE International Conference on Engineering and Technology (ICETECH), 2016 Publication	<1 %
24	Submitted to University of Newcastle Student Paper	<1 %
25	mikazo.csid21.com Internet Source	<1 %
26	github.com Internet Source	<1 %
27	cloudsecurityalliance.fr Internet Source	<1 %

28	helovia.net Internet Source	<1 %
29	Pro React, 2015. Publication	<1 %
30	Lifeng Wang, Zhengping Wu. "A Trustworthiness Evaluation Framework in Cloud Computing for Service Selection", 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, 2014 Publication	<1 %
31	charitypigroast.com Internet Source	<1 %
32	patents.google.com Internet Source	<1 %
33	bz.apache.org Internet Source	<1 %
34	docs.aws.amazon.com Internet Source	<1 %
35	Submitted to Strathfield College Pty Ltd Student Paper	<1 %
36	acpuuk.org.uk Internet Source	<1 %
37	"Emerging Technologies in Data Mining and Information Security", Springer Science and Business Media LLC, 2019	<1 %

- 38 Submitted to colorado-technical-university <1 %  
Student Paper
- 
- 39 machine.zive.net <1 %  
Internet Source
- 
- 40 manifold.umn.edu <1 %  
Internet Source
- 
- 41 version.aalto.fi <1 %  
Internet Source
- 
- 42 www.intility.no <1 %  
Internet Source
- 
- 43 Aarthy Raghu, Meenakumari Balaiah, Sridevi Veluswami, Shirley Sundarsingh, Rajkumar Thangarajan, Samson Mani. "Identification of novel somatic cell-free DNA variants by next-generation sequencing in breast cancer patients", International Journal of Molecular & Immuno Oncology, 2020 <1 %  
Publication
- 
- 44 Jirayu Kanpariyasoontorn, Twittie Senivongse. "Cloud service trustworthiness assessment based on cloud controls matrix", 2017 19th International Conference on Advanced Communication Technology (ICACT), 2017 <1 %  
Publication
- 
- 45 blog.claves.me <1 %  
Internet Source

<1 %

- 
- 46 Submitted to The Cornell Education Group <1 %  
Student Paper
- 
- 47 examples.javacodegeeks.com <1 %  
Internet Source
- 
- 48 historienet.no <1 %  
Internet Source
- 
- 49 Adrian W. West. "Practical Web Design for  
Absolute Beginners", Springer Science and  
Business Media LLC, 2016 <1 %  
Publication
- 
- 50 codebrahma.com <1 %  
Internet Source
- 
- 51 Sai Matam, Jagdeep Jain. "Pro Apache JMeter",  
Springer Science and Business Media LLC, <1 %  
2017  
Publication
- 
- 52 nl.wikihow.com <1 %  
Internet Source
- 
- 53 www.wileyplus.com <1 %  
Internet Source
- 
- 54 eportfolio.lib.ksu.edu.tw <1 %  
Internet Source

- 55 "Intelligent and Fuzzy Techniques: Smart and Innovative Solutions", Springer Science and Business Media LLC, 2021 <1 %  
Publication
- 
- 56 Submitted to Amity University <1 %  
Student Paper
- 
- 57 gist.github.com <1 %  
Internet Source
- 
- 58 idcloudhost.com <1 %  
Internet Source
- 
- 59 pt.wikipedia.org <1 %  
Internet Source
- 
- 60 Brian Travis. "Pro Drupal 7 for Windows Developers", Springer Science and Business Media LLC, 2011 <1 %  
Publication
- 
- 61 Journal of Corporate Real Estate, Volume 14, Issue 2 (2012-07-28) <1 %  
Publication
- 
- 62 blouson.indiesj.com <1 %  
Internet Source
- 
- 63 datatofish.com <1 %  
Internet Source
- 
- 64 doaj.org <1 %  
Internet Source

65	repository.londonmet.ac.uk Internet Source	<1 %
66	www.forwiss.tu-muenchen.de Internet Source	<1 %
67	Submitted to George Washington University Student Paper	<1 %
68	aspidan.nu Internet Source	<1 %
69	www.ebah.pt Internet Source	<1 %
70	www.scribd.com Internet Source	<1 %
71	artemisprojects.org Internet Source	<1 %
72	checkbox12.blogspot.com Internet Source	<1 %
73	historiek.net Internet Source	<1 %
74	inspirit.net.in Internet Source	<1 %
75	www.4tix.ch Internet Source	<1 %
76	www.allalliedhealthschools.com Internet Source	<1 %

77	<a href="http://www.interbookie.pl">www.interbookie.pl</a>	<1 %
Internet Source		
78	<a href="http://www.sec.gov">www.sec.gov</a>	<1 %
Internet Source		
79	Moises Belchin, Patricia Juberias. "Web Programming with Dart", Springer Science and Business Media LLC, 2015	<1 %
Publication		
80	Wang, Lifeng, and Zhengping Wu. "A Novel Trustworthiness Measurement Model for Cloud Service", 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014.	<1 %
Publication		
81	<a href="http://journalofcloudcomputing.springeropen.com">journalofcloudcomputing.springeropen.com</a>	<1 %
Internet Source		
82	<a href="http://studentprojectguide.com">studentprojectguide.com</a>	<1 %
Internet Source		
83	<a href="http://www.amazon.com.mx">www.amazon.com.mx</a>	<1 %
Internet Source		
84	<a href="http://www.coursehero.com">www.coursehero.com</a>	<1 %
Internet Source		
85	<a href="http://www.qhaut.de">www.qhaut.de</a>	<1 %
Internet Source		

86	Brett McLaughlin, Sara Perrott. "AWS Certified SysOps Administrator Study Guide, 2E", Wiley, 2020	<1 %
	Publication	
87	Dan Mabbutt, Adam Freeman, Matthew MacDonald. "Pro ASP.NET 4.5 in VB", Springer Science and Business Media LLC, 2013	<1 %
	Publication	
88	<a href="http://www.amazon.fr">www.amazon.fr</a>	<1 %
	Internet Source	
89	<a href="http://www.concernusa.org">www.concernusa.org</a>	<1 %
	Internet Source	
90	<a href="http://www.theguardian.com">www.theguardian.com</a>	<1 %
	Internet Source	
91	<a href="http://www.tuning-gids.nl">www.tuning-gids.nl</a>	<1 %
	Internet Source	
92	Submitted to University of Hong Kong	<1 %
	Student Paper	

Exclude quotes      On  
Exclude bibliography      On

Exclude matches      Off