Project X Prototype Documentation



Developers: Ryan Ben & Glen Chiridza

Contents

[Chapter 1: Introduction 3](#_Toc95895398)

[1.1 Purpose of the TD (Technical Document) 3](#_Toc95895399)

[1.2 TD Audience 3](#_Toc95895400)

[1.3 Executive Summary 3](#_Toc95895401)

[1.4 Aims and Objectives 4](#_Toc95895402)

[1.5 Methodologies and Instruments 4](#_Toc95895403)

[1.5 Expected Results 4](#_Toc95895404)

[1.6 Timelines 5](#_Toc95895405)

[Chapter 2: Analysis Phase 6](#_Toc95895406)

[2.1 Analysis of Study 6](#_Toc95895407)

[2.1.1 Analyzing AWS Cloud Trail 6](#_Toc95895408)

[2.1.2 Analyzing Claspin 7](#_Toc95895409)

[2.2 Gaps Identified 9](#_Toc95895410)

[2.3 Conclusion 9](#_Toc95895411)

[Chapter 3: Feasibility Analysis 10](#_Toc95895412)

[3.1 Rationale of proposed system 10](#_Toc95895413)

[3.2 Feasibility Analysis 10](#_Toc95895414)

[3.3 Requirements 10](#_Toc95895415)

[Chapter 4: Design Phase 12](#_Toc95895416)

# Chapter 1: Introduction

**Empower Bank** our client requires application to log all activity within their network and application stack, for applications that are running on premise. This which is a co-operate requirement and is required as early as possible. Before Sesfont can receive the tender, a prototype is required within 2 weeks for assessment.

## 1.1 Purpose of the TD (Technical Document)

This document is meant to tackle the following

* Understand the client requirements.
* Outline the research gathered to provide the solution.
* Outline the constraints of the system.
* Provide a detailed design of the system.

## 1.2 TD Audience

The intended audience is the Sesfont Development Team, Project Management Team and Empower Bank Operations Team.

* Sesfont Project Manager.
* Sesfont Developers.
* Network Engineers.
* DevOps Engineers.
* Software Architects.

## 1.3 Executive Summary

Sesfont has entered a cooperate agreement with Empower Bank to provide a functional prototype within two weeks. Empower Bank a new startup which may potentially offer more tenders in the future as it grows.

Project Goals Include:

1. Provide logging over Empower Bank existing infrastructure.
2. Provide logging over Empower Bank existing architecture.
3. Provide an interface to query trail logs.
4. Provide real time network and application logs.
5. Provide a functional protype with n features within 2 weeks.

## 1.4 Aims and Objectives

The following are the functional aims of the IAAS

* Log inbound traffic from outside the network.
* Log outbound traffic from inside the network.
* Log internal traffic.
* Log traffic at server network level.
* Log requests associated with applications internal and external.
* Allow permitted users to query logs.
* Allow permitted users to define the type of traffic to collect.
* Allow permitted users to export traffic logs.
* Real time logs.

## 1.5 Methodologies and Instruments

The following are the technologies to achieve the intended IAAS.

* Angular (Frontend)
* Django (Backend)
* Python (Services)
* Nginx (Reverse Proxy)
* Microsoft SQL Server (Databases)
* Rsyslog (Logging)
* Redis (Cache)
* Django Channels (Socket Connections)

## 1.5 Expected Results

For the prototype 4 or more functional requirements are required. Priority on

* Logging Application Traffic
* Logging server Traffic
* Query traffic
* Real time updates

## 1.6 Timelines



# Chapter 2: Analysis Phase

In this chapter we analyze the existing solutions to infrastructure and architecture monitoring, and present the necessary research information to build the IAAS.

## 2.1 Analysis of Study

The most popular service for network and application traffic monitoring is AWS Cloud Trail, which is capable of monitoring everything from the VPC traffic and application activities within the VPC i.e. launching EC2 instances. As far as we are concerned, this is the most ideal solution as it provides the service to all AWS cloud developers.

Other solutions include Claspin which is used by Meta to monitor traffic for their applications Instagram, Facebook e.t.c, but this is product specific and internal rather than being provided as a service.

### 2.1.1 Analyzing AWS Cloud Trail

From the source [AWS](https://aws.amazon.com/cloudtrail/faqs/#:~:text=AWS%20CloudTrail%20enables%20auditing%2C%20security,%2C%20analysis%2C%20and%20remediation%20actions.) , CloudTrail enables auditing, security monitoring, and operational troubleshooting by tracking user activity and API usage. CloudTrail logs, continuously monitors, and retains account activity related to actions across your AWS infrastructure, giving you control over storage, analysis, and remediation actions.

Too which it has the following benefits:

CloudTrail helps you prove compliance, improve security posture, and consolidate activity records across regions and accounts. CloudTrail provides visibility into user activity by recording actions taken on your account. CloudTrail records important information about each action, including who made the request, the services used, the actions performed, parameters for the actions, and the response elements returned by the AWS service. This information helps you track changes made to your AWS resources and troubleshoot operational issues. CloudTrail makes it easier to ensure compliance with internal policies and regulatory standards. For more details, refer to the AWS compliance white paper “[Security at scale: Logging in AWS](https://d1.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf)”.

**The advantage AWS has over Cloud Trail is that it monitors Amazon’s internal applications, whilst we are meant to monitor applications like Pastel which is an off the shelf application and don’t have the source code too.**

It unfortunate that Cloud Trail is a closed source application we do not have access too, hence all need in this analysis is to understand the workings of Cloud Trail.

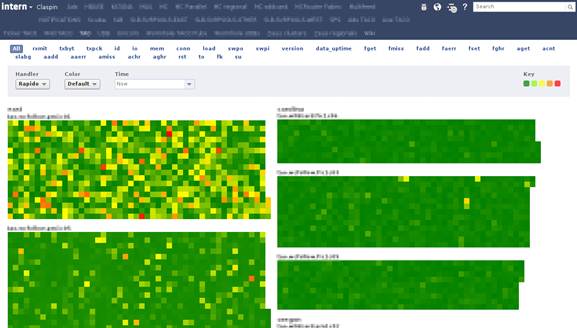
### 2.1.2 Analyzing Claspin

Source from Klint Finley’s blog **“**[***A Peek At Facebook’s Insanely Awesome Monitoring Tool Claspin – TechCrunch***](https://techcrunch.com/2012/09/19/a-peak-at-facebooks-insanely-awesome-monitoring-tool-claspin/#:~:text=So%20he%20built%20Claspin%2C%20a,representing%20servers%20grouped%20by%20rack.)***”.***

*Facebook engineer Sean Lynch has built a mind-blowing, custom server-monitoring tool for the company that uses heat mapping to keep tabs on a huge number of servers at a glance.*

*Lynch is part of the cache performance team at Facebook. When things go wrong, he needs to know quickly whether problems are being caused by caching or something else. Off the shelf monitoring tools just weren’t good enough.*

*So, he built Claspin, a server-monitoring system named after a protein that monitors cells for DNA damage. And today he’s giving curious readers a tour of the system*[*in a post on the Facebook Engineering Blog*](https://www.facebook.com/notes/facebook-engineering/monitoring-cache-with-claspin/10151076705703920)*.*

**

*Claspin displays grid-like maps representing servers grouped by rack. Each cell of the grid represents one server, and its color depends on the health of that particular server. Green for good, red for bad, yellow for in-between and black if it’s missing a stat (which means it’s probably down). This visualization approach enables Facebook engineers to check the status of a huge number of servers at once.*

*“On a 30″ screen we could easily fit 10,000 hosts at the same time, with 30 or more stats contributing to their color, updated in real time — usually in a matter of seconds or minutes,” Lynch writes.*

*“When I first deployed Claspin, the view above had a lot redder in it,” he writes. “By making it easier for more people to spot server issues quickly, Claspin has allowed us to catch more ‘yellows’ and prevent more ‘reds.'”*

*As to how Claspin determines the health of a system, Lynch writes: “I settled on coloring a host by its ‘hottest’ statistic, with hotness computed from predefined thresholds. It’s dirt simple, but it gives us a way to encode tribal knowledge about what values are ‘bad’ into the view.”*

*Claspin provides a tabbed interface so that Lynch can toggle between different views. He can also change which stats affect the color of the cells. “Mousing over a host draws an outline around its rack and pops up a tooltip with the hostname, rack number, and all the stats Claspin is looking at for that host, with the values colored based on Claspin’s thresholds for that stat,” he writes.*

*The interface is entirely browser-based. Coloring is done with JavaScript and the heatmaps are drawn with SVG. You can find out more and see a couple more screenshots*[*on Lynch’s blog post*](https://www.facebook.com/notes/facebook-engineering/monitoring-cache-with-claspin/10151076705703920)*.*

*Facebook engineers have*[*talked about Claspin before*](http://www.infoworld.com/d/application-development/facebook-we-need-create-our-own-dev-and-management-tools-196437)*in interviews, but I think this is the first time we’ve gotten a peek behind the curtain.*

*It doesn’t look like the company is open sourcing this project quite yet. “We always try to open-source tools like this, so it’s something we’ll consider with Claspin,” a Facebook spokesperson told me. “But it’s possible that it’s so tightly integrated with our infrastructure that it wouldn’t be broadly useful.”*

*Facebook has*[*open sourced a lot of its custom-built development and operations software*](https://developers.facebook.com/opensource/)*, including the NoSQL database*[*Apache Cassandra*](http://cassandra.apache.org/)*and its PHP to C++ transformer*[*HipHop*](https://github.com/facebook/hiphop-php)*. It’s even gone so far as to open source its*[*data center infrastructure plans*](https://beta.techcrunch.com/2011/04/07/open-compute-project/)*. So don’t be surprised to see this*[*hit GitHub in the future*](https://github.com/facebook)*.*

*In the meantime, who’s going to be the first to clone it?*

**This even though explaining how it works, it does not give an in site of how Meta actually built their monitoring infrastructure. But health checks are a must to implement, as they are essential for monitoring activity.**

**Claspin also does not seem to log application traffic, but a post by Jonah Kowal explains that it is both an integration of an internal tool and customizing code. Jonah Kowal says:**

*Facebook, Google, Netflix, and many others use a customized and in-house written stack of tools which combine some open source with lots of custom code. They do not typically follow the standard mix of commercial and open source you see in most companies. Their entire runtimes from the OS to the application server, to the application itself is customized open source, so the monitoring must also follow the same design pattern.*

## 2.2 Gaps Identified

* Cloud Trails is product specific.
* Although the service if offered by Cloud Watch, Cloud Trail does not perform health checks on server.
* Claspin does not log application data.

## 2.3 Conclusion

Not much can be gathered about how the existing solutions are built, but we definitely know how they work. Narrowing our requirements, which are to build a service that can perform health checks on servers and both network and application logs in the network.

# Chapter 3: Feasibility Analysis

[Product Name] is infrastructure as service that monitors traffic to individual server and applications, as well as the internal network traffic. It prossvides an interface to query each of these logs, as well as provide the ability to specify the type of logs to collect i.e., Log HTTP and FTP traffic only. All this done without flooding the entire network.

## 3.1 Rationale of proposed system

* It is co-operated requirements to log application and network activity.
* There is currently no service for this for application and network running on premise.
* It makes it easier to track a problem if it occurs.
* Empower Bank is able to make predictions and also be aware when there are threats or problems to clients outside the network.

## 3.2 Feasibility Analysis

Technical Feasibility

* A router capable of receiving logs is not available at Sesfont Technologies. Hence a solution to monitoring network activity is to be done at the later stage.
* However, software to build the service is present.

Social & Economic Feasibility

* Empower bank can now trace application and network events.
* Empower bank can visualize their application and network traffic.

## 3.3 Functional Requirements

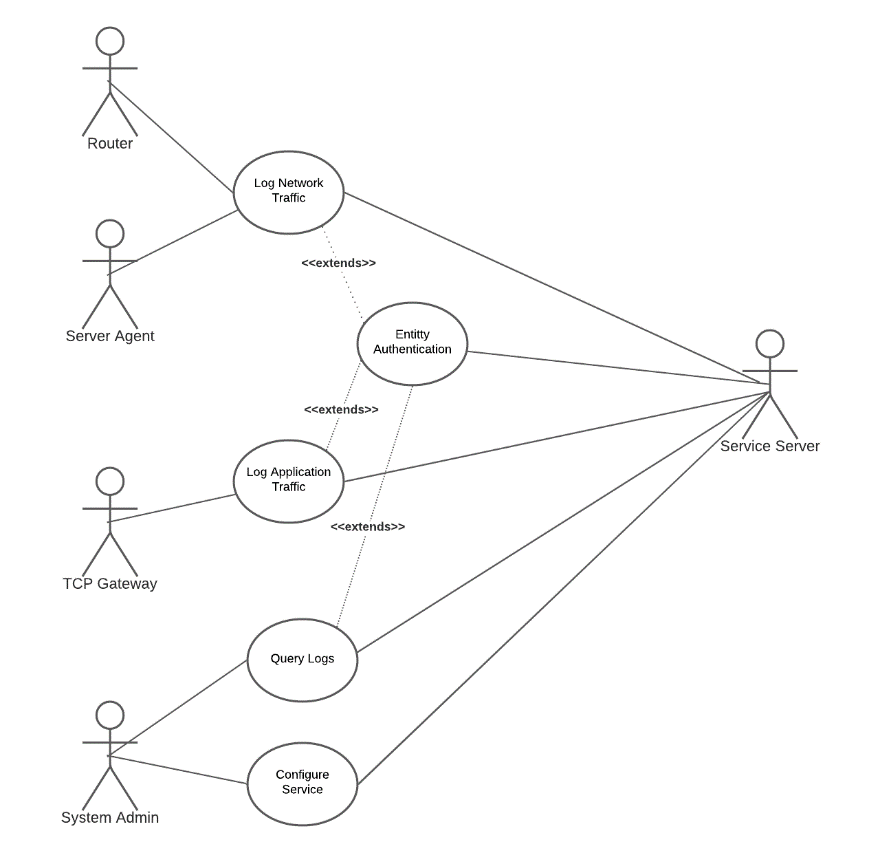
As mentioned in 1.4 the following are the set of functional requirements

* Log inbound traffic from outside the network.
* Log outbound traffic from inside the network.
* Log internal traffic.
* Log traffic at server network level.
* Log requests associated with applications internal and external.
* Allow permitted users to query logs.
* Allow permitted users to define the type of traffic to collect.
* Allow permitted users to export traffic logs.
* Real time logs.

## 3.4 Non-Functional Requirements

* A Premium User Interface (Modern Material Design & Micro interactions)
* Little network load for the service. We do not want to flood the network as it has its initial purpose.
* Close to bug free code.
* Provide SHA256 security for user data.
* Clean code practices during development.
* Code reviews during development.
* Low latency for retrieving and processing data.
* Code reusability (Modular Approach).
* Easy installation.

## 3.5 Use Case Diagram

Below is the use case diagram of the IAAS.

# Chapter 4: Design Phase

The design at this stage is limited to the functional requirements of the prototype, which are part of the full version of the service. This is because Empower bank are not yet clear on their infrastructure, as they outsource their development abroad.

## 3.4 Architectural Design

Below is a list of the entities that make up the IAAS. Also with it is the reason why the entity exists and its configurations.

**Router (Physical)**

The only device capable of monitoring the network traffic is the router itself, as it is the backbone of the network. Its purpose if to send its logs to the Application Server, to which we can store and present that information to the client.

**Server Agent (Application)**

This is an application that runs on the server hosting an instance of an application. It collects packets from the server’s physical socket. Unpack the packet and send the logs to the application server.

**TCP Gateway (Application)**

A combination of Nginx and Rsyslog, the TCP send TCP logs to the application server for processing. This sits at the entry point of the network or possibly the server itself if Empower Bank is running Nginx on their servers.

**Redis**

In memory database to support Django Channels and also reduce latency by caching some information.

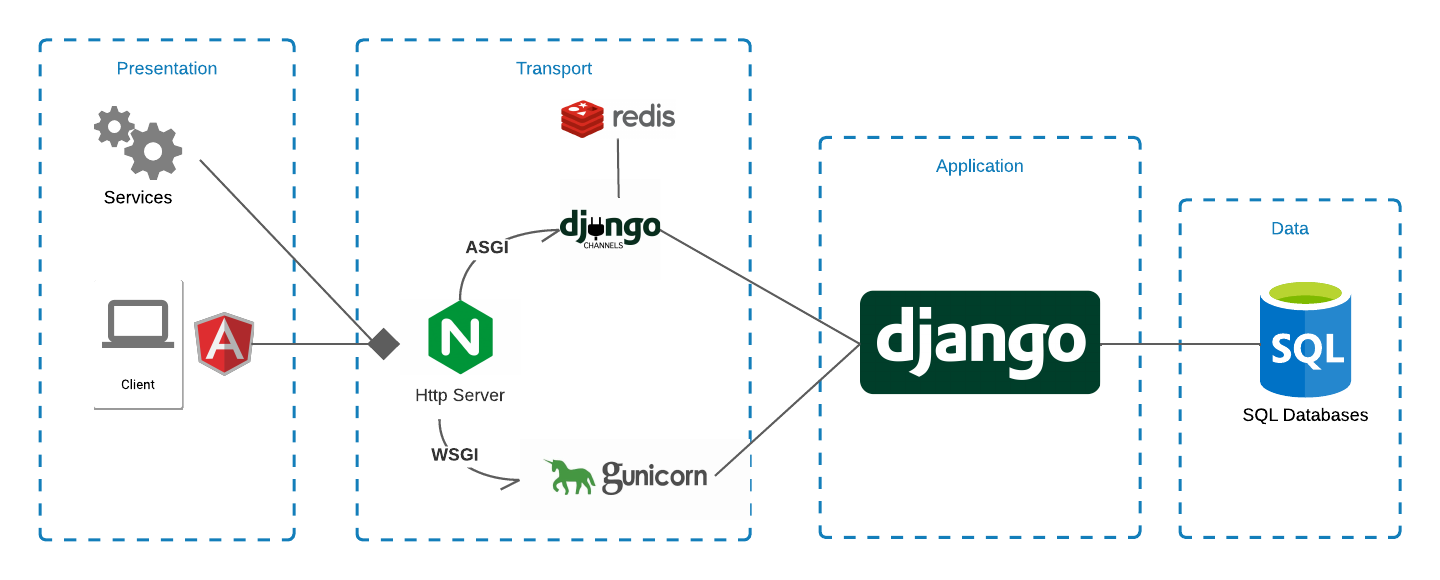
**Django Channels**

To provide real-time logs to the client application.

**Django**

Application server to support the service.

Below is the architectural diagram of the system. Loggers are used to represent data coming from proxies and server agents. And from the diagram, one can note that a Monolithic approach is taken, and this is to reduce clutter. We do not want empower bank worrying about the dev ops of the PAAS instead of their core banking system.



## 3.5 Design Research

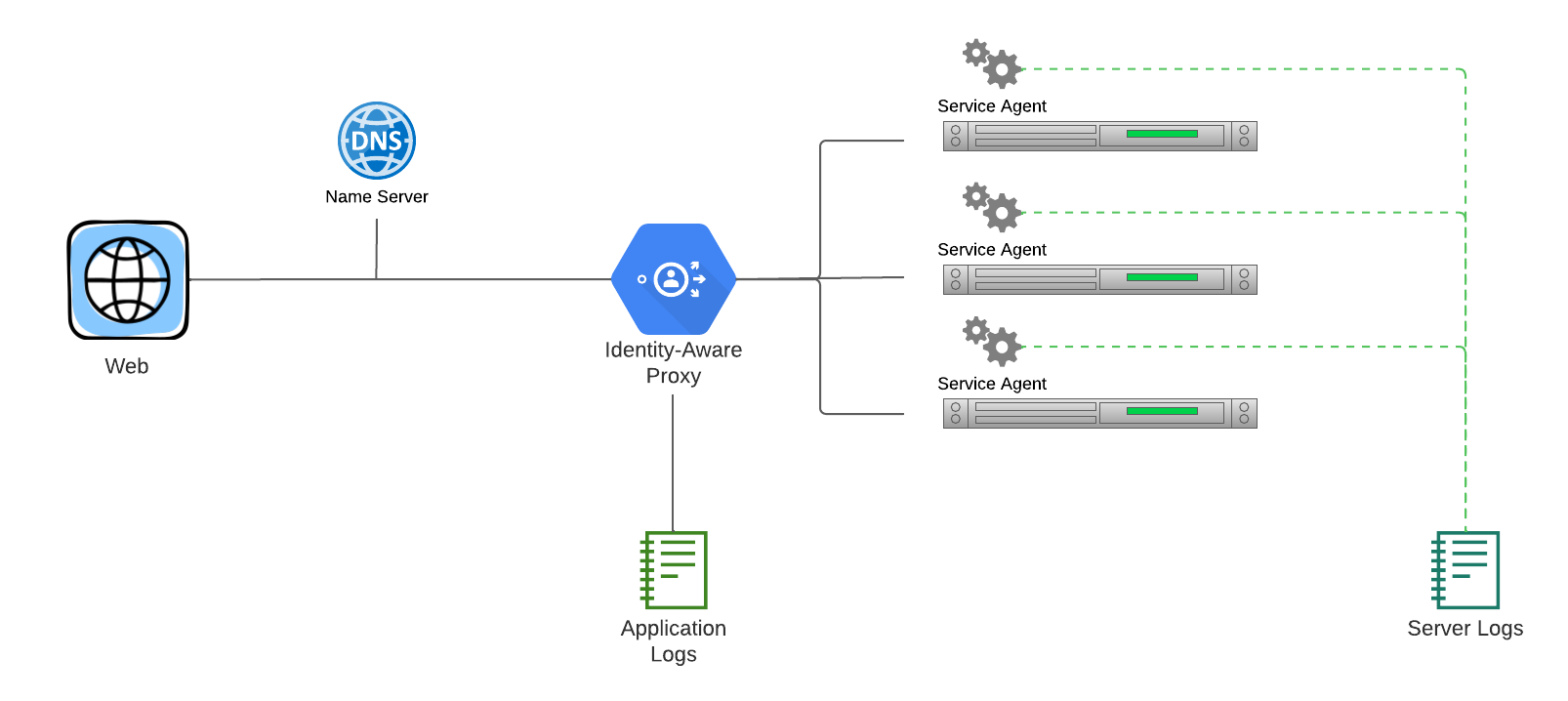
Here is the most important sub topic of the documentation. In this sub topic we look at the research done as to provide a fully running systems. No patents are present for the information below.

### 3.5.1 Design Backbone

In our requirements we want to collect application and network traffic. Starting with the simplest for us **Application Data.**

The main constraint that distinguishes this project from **Logger** is that we cannot directly run our code on the Empower Bank applications. This means we have to setup a tap which we can log incoming traffic into the network. With the knowledge present, we can make use of the Nginx or Apache log entries, to which we send these to the service application for processing with the help of Rsyslog. **It is at this point that the requests are unencrypted** by SSL.

For **Network Data** these are two types of traffic we will collect, Outbound and Inbound at the server level & outbound and inbound at the router. An agent will be installed on the server to watch the socket and send traffic data real time. **For the router, research is still to be carried out as myself… am not a network engineer.**

****

### 3.5.2 Nginx Logging

Source from the Nginx [Documentation](https://docs.nginx.com/nginx/admin-guide/monitoring/logging/).

NGINX writes information about client requests in the access log right after the request is processed. By default, the access log is located at **logs/access.log**, and the information is written to the log in the predefined **combined** format. To override the default setting, use the [log\_format](https://nginx.org/en/docs/http/ngx_http_log_module.html#log_format) directive to change the format of logged messages, as well as the [access\_log](https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log) directive to specify the location of the log and its format. The log format is defined using variables.

The following examples define the log format that extends the predefined **combined** format with the value indicating the ratio of gzip compression of the response. The format is then applied to a virtual server that enables compression.

http {

log\_format compression '$remote\_addr - $remote\_user [$time\_local] '

'"$request" $status $body\_bytes\_sent '

'"$http\_referer" "$http\_user\_agent" "$gzip\_ratio"';

server {

gzip on;

access\_log /spool/logs/nginx-access.log compression;

...

}

}

Another example of the log format enables tracking different time values between NGINX and an upstream server that may help to diagnose a problem if your website experience slowdowns. You can use the following variables to log the indicated time values:

* [$upstream\_connect\_time](https://nginx.org/en/docs/http/ngx_http_upstream_module.html#var_upstream_connect_time) – The time spent on establishing a connection with an upstream server
* [$upstream\_header\_time](https://nginx.org/en/docs/http/ngx_http_upstream_module.html#var_upstream_header_time) – The time between establishing a connection and receiving the first byte of the response header from the upstream server
* [$upstream\_response\_time](https://nginx.org/en/docs/http/ngx_http_upstream_module.html#var_upstream_response_time) – The time between establishing a connection and receiving the last byte of the response body from the upstream server
* [$request\_time](https://nginx.org/en/docs/http/ngx_http_log_module.html#var_request_time) – The total time spent processing a request

All time values are measured in seconds with millisecond resolution.

http {

log\_format upstream\_time '$remote\_addr - $remote\_user [$time\_local] '

'"$request" $status $body\_bytes\_sent '

'"$http\_referer" "$http\_user\_agent"'

'rt=$request\_time uct="$upstream\_connect\_time" uht="$upstream\_header\_time" urt="$upstream\_response\_time"';

server {

access\_log /spool/logs/nginx-access.log upstream\_time;

...

}

}

When reading the resulting time values, keep the following in mind:

* When a request is processed through several servers, the variable contains several values separated by commas
* When there is an internal redirect from one upstream group to another, the values are separated by semicolons
* When a request is unable to reach an upstream server or a full header cannot be received, the variable contains 0 (zero)
* In case of internal error while connecting to an upstream or when a reply is taken from the cache, the variable contains - (hyphen)

Logging can be optimized by enabling the buffer for log messages and the cache of descriptors of frequently used log files whose names contain variables. To enable buffering use the buffer parameter of the [access\_log](https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log) directive to specify the size of the buffer. The buffered messages are then written to the log file when the next log message does not fit into the buffer as well as in some other [cases](https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log).

To enable caching of log file descriptors, use the [open\_log\_file\_cache](https://nginx.org/en/docs/http/ngx_http_log_module.html#open_log_file_cache) directive.

Similar to the error\_log directive, the [access\_log](https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log) directive defined on a particular configuration level overrides the settings from the previous levels. When processing of a request is completed, the message is written to the log that is configured on the current level, or inherited from the previous levels. If one level defines multiple access logs, the message is written to all of them.

### 3.5.3 Nginx Socket Logging

The syslog utility is a standard for computer message logging and allows collecting log messages from different devices on a single syslog server. In NGINX, logging to syslog is configured with the syslog: prefix in [error\_log](https://nginx.org/en/docs/ngx_core_module.html#error_log) and [access\_log](https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log) directives.

Syslog messages can be sent to a server= which can be a domain name, an IP address, or a UNIX-domain socket path. A domain name or IP address can be specified with a port to override the default port, 514. A UNIX-domain socket path can be specified after the unix: prefix:

error\_log syslog:server=unix:/var/log/nginx.sock debug;

access\_log syslog:server=[2001:db8::1]:1234,facility=local7,tag=nginx,severity=info;

In the example, NGINX error log messages are written to a UNIX domain socket at the debug logging level, and the access log is written to a syslog server with an IPv6 address and port 1234.

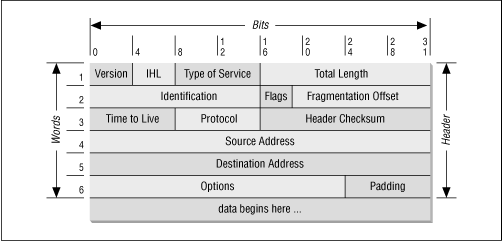
The facility= parameter specifies the type of program that is logging the message. The default value is local7. Other possible values are: auth, authpriv, daemon, cron, ftp, lpr, kern, mail, news, syslog, user, uucp, local0 ... local7.

The tag= parameter applies a custom tag to syslog messages (nginx in our example).

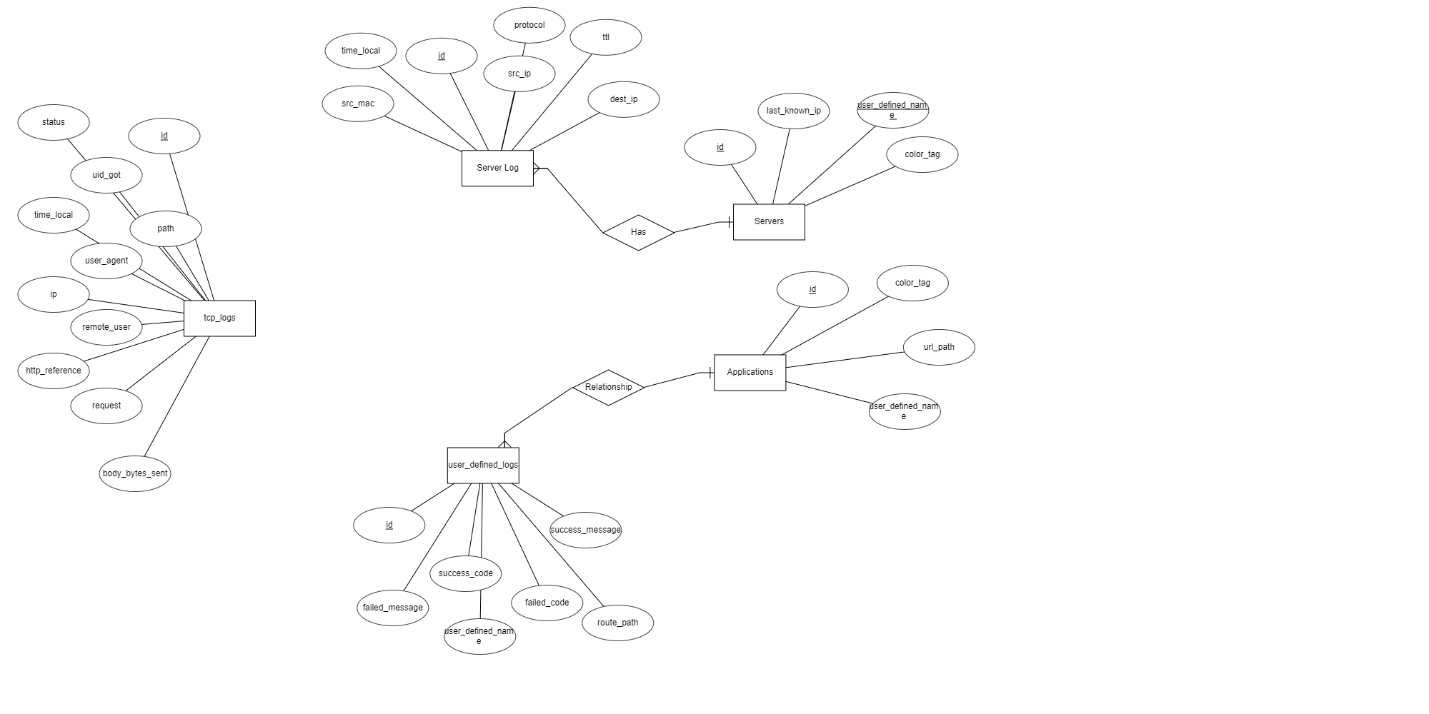
The severity= parameter sets the severity level of syslog messages for access log. Possible values in order of increasing severity are: debug, info, notice, warn, error (default), crit, alert, and emerg. Messages are logged at the specified level and all more severe levels. In our example, the severity level error also enables crit, alert, and emerg levels to be logged.

3.5.4 Python Socket Sniffing

The structure of a packet can be seen below. Its all just a matter of intercepting a packet and decrypting it. A tutorial of this can be followed at [address](https://www.youtube.com/watch?v=dM9grWOdTBI).



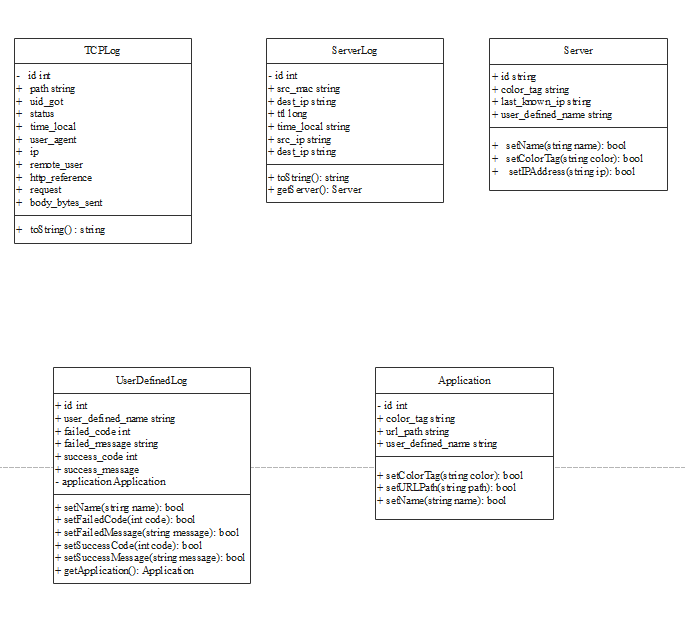
## 3.6 Database Design



Above is the ERD diagram for the prototype database.

## 3.7 Class Diagram

Below is the class diagram of the IAAS.



## 3.8 API Calls

Below is summary of the API calls.

* /api/getServers
* /api/getServerLogs
* /api/getTCPLogs
* /api/getUserDefinedLogs
* /api/getUDLogs
* /api/getApplications
* /api/createUserDefinedLog
* /api/configureService
* /api/getServiceStatus

### 3.8.1 Get Servers

This request is meant to provide a client application with a list of servers, and carries 1 optional parameter **server\_id**. If the server\_id is defined, only the server with that id will be returned instead of the entire list.

Route: /api/getServers

Route (With Parameter): /api/getServers?server\_id=<your\_server\_id>

A response code of 500 would mean an internal server error, and a 404 will mean the server of specified server\_id was not found.

### 3.8.2 Get Server Logs

This request is meant to provide a client application with a list of server logs, and carries6 optional parameters, **server\_id**, **last\_log\_id, pending\_logs**, **start\_time, src\_ip, dest\_ip, src\_mac, log\_count**. When called without any parameter the server will return the last 1000 logs from all servers combined.

The optional parameters are filters that are execute in the order they appeared in the above text. Adding the filter **server\_id** will return the last 1000 logs of the server with that id. Adding **last\_log** will return 1000 logs before the id of the specified log. Adding **log\_count** will return n logs. Adding **pending\_logs** will return the new logs from the specified log id.sss

Route: /api/getServerLogs

Route (With Parameter): /api/getServerLogs?server\_id=<your\_server\_id>&log\_count=<your\_log\_count>

Example:

/api/getServerLogs?server\_id=ZWLHRE0001&dest\_ip=127.43.21.215&log\_count=200

Will return the last 200 logs from the server ZWLHRE0001 that have a destination address of 127.43.21.215

A response code of 500 would mean an internal server error, and a 404 will mean the server of specified server\_id was not found.

### 3.8.3 Get TCP Logs

This request is meant to provide a client application with a list of TCP logs, and carriesn optional parameters, **path**, **last\_log\_id, start\_time, user\_agent, source\_ip, dest\_ip, http\_reference, body\_bytes, log\_count.** When called without any parameter the server will return the last 1000 TCP logs from all paths combined.

The optional parameters are filters that are execute in the order they appeared in the above text. Adding the filter **path** will return the last 1000 logs of the specified path. Adding **last\_log** will return 1000 logs before the id of the specified log. Adding **log\_count** will return n logs.

Route: /api/getTCPLogs

Route (With Parameter): /api/getTCPLogs?path=<your\_path>&log\_count=<your\_log\_count>

Example:

/api/getTCPLogs?dest\_ip=135.43.21.215&log\_count=200

Will return the last 200 logs from the destination address 135.43.21.215.

A response code of 500 would mean an internal server error, and a 404 will mean the server of specified server\_id was not found.

### 3.8.4 Get User Defined Logs

This request is meant to provide a client application with a list of user defined logs, and carries1 parameter **Application,** and will return all user defined logs of that application.

Route: /api/ getUserDefinedLogs?Application=<your\_application\_id>

A response code of 500 would mean an internal server error, and a 404 will mean the server of specified server\_id was not found.

### 3.8.5 Get Applications

This request is meant to provide a client application with a list of all defined applications.

Route: /api/ getApplications

### 3.8.6 Get UDL Logs

This request is meant to provide a client application with a list of UDL logs. It carries 1 compulsory parameter **UDL** which is the id of the User Defined Log.

Optional parameters are **Response**, **last\_log, start\_time, user\_agent, source\_ip, dest\_ip, http\_reference, body\_bytes** and **log\_count**. If no optional parameters are defined the server will return the last 1000 of the specified UDL.

Route: /api/getUDLogs?UDL=<your\_udl\_id>

Example:

/api/getUDDLogs?UDL=EIDJA21K&last\_log=1321&dest\_ip=54.163.21.23&log\_count=200

Will return the last 200 logs before the log 1321, from the destination address 54.163.21.23 of the user defined log EIDJA21K.

### 3.8.6 Create UDL Log

This request creates a user defined UDL. It takes in a JSON object of class UserDefinedLog.

Route: /api/createUserDefinedLog

A response code of 500 means the process failed, and within the body is the reason why the action failed.

### 3.8.7 Configure Service

This request creates configures the service engine. A json containing the **gateway\_application\_ip**, **log\_retrival\_count and server\_log\_buffer** should be posted along with the request.

Route: /api/ configureService

A response code of 500 means the process failed.

### 3.8.8 Configure Service

This request is used to check if the IAAS is configured.

Route: /api/getServiceStatus

A response code of 500 means the server is not configured.