**OVERLAY NETWORK MONITORING SYSTEM**

**DESIGN DOCUMENTATION**

VERSION 1.2

**Team Name:** Smart Developers

**Team Members:**

* Lohit Srihas Korlepara
* Ramakrishna Bonam
* Phani Varanasi
* Sai Lakshmi Pulagam
* Shravya Nuka
* Vineesha Sana
* Akhila Chatlapalle
* Surya Theja Kovur
* Manikanta Srinivas B V G
* Sai Jagannadh Koya
* Bandhavi Vishnumolakala

1. **PREFACE:**

The main concern of the project is to provide the customer a simple and unified way of maintaining and updating its applications which interact with the monitoring system through a common RESTful API. This is the initial version of the document.

In the remainder of the document, Section II describes briefly about the basic abbreviations used in this document. Section III describes Module 1: Frontend which is divided into Detail design and Unit Test plan. Section IV describes Module 2: database Management and Backend which is divided into Detail design and Unit Test plan. Section V describes Module 2: Backend which is divided into Detail design and Unit Test plan. Lastly Section VI shows list of References used in documentation

**Release v1.2 on 2017-06-16**

* Made changes to Frontend, Database Management and Backend tests
* Made few layout modifications

**Release v1.1 on 2016-12-18**

* Made changes accordingly in the section glossary and abbreviations
* Changes have been made in the module Database and Backend Modules

**Release v1.0 on 2016-05-01**

* Initial release of the document

1. **GLOSSARY AND ABBREVATIONS:**

* **API:** Application Programming Interface

An API is a set of routines, protocols, and tools for building software applications.

* **InfluxDB:** Influx Database

InfluxDB is an open source time series database. InfluxDB has no external dependencies and provides an SQL-like language with built in time-centric functions for querying a data structure composed of measurements, series, and points.

**Grafana:**

Grafana is most commonly used for visualizing time series data for Internet infrastructure and application analytics but many use it in other domains including industrial sensors, home automation, weather, and process control. Grafana features pluggable panels and data sources allowing easy extensibility and a variety of panels, including fully featured graph panels with rich visualization options. There is built in support for many of the most popular time series data sources.

* **SSL:** Secure Sockets Layer

SSL is a standard security technology for establishing an encrypted link between a web server and a browser.

* **RTT:** Round Trip Time
* **HTTPS:** Hyper Text Transport Protocol Secure
* **RESTful:** Representative State Transfer

1. **MODULES**

**MODULE 1: FRONTEND**

This module contains the frontend portion of the project. The dashboard contains network data, represented in the form of graphs. The dashboard connects to admin server via HTTP connection. The user credentials are stored in MySQL database. Graphs are being plotted dynamically with data from server database. When a node is down the graph is plotted by inserting zero values for that specific node.

A RESTful interface of web service is designed with Python flask script. It facilitates communication of data between server node and user nodes through HTTP GET, POST, PUT & Delete requests. A web page is created such that a 3rd party can send a GET request for data related to a specific service, to view the statistical data in a graphical form.

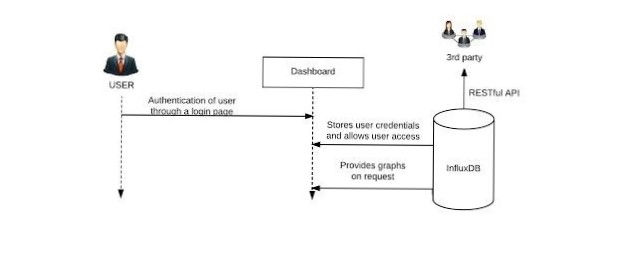


Fig 1.0: Frontend architecture

**INTERFACING:**

1. Creation of new user through *Sign Up Page* (http://<server\_ip:8080>/signup.html)

HTTP Methods: GET, POST

Language: HTML, CSS

Framework: Flask

Database: MySQL

1. User Login through the *Login Page* (http://<server\_ip:8080>/login.html)

HTTP Methods: GET

Language: HTML, CSS

Framework: Flask

Database: MySQL

1. Graphical representation of user statistics through *Grafana* (http://<server\_ip:3000>)

HTTP Methods: GET

Language: HTML, CSS

Framework: Flask

Database: Influx

1. User Account Settings update through *Settings Page* (http://<server\_ip:8080>/account.html)

HTTP Methods: GET, PUT

1. Extraction of parameters from each node and storage in server node’s database through influx.

Language: Python (extraction of parameters using psutil module)

Database: Influx (for storage)

**Programming Language/Frameworks Used:**

1. Python 2.7 – The backend code is developed using python script
2. Flask – The Rest API framework is developed using flask module

**Unit Test Plan:**

**F-T1:** Login

**Test**: Login page test

**Purpose**: To prevent unauthorized users from having access to the dashboard

**Requirements:** Req\_SYSF1, Req\_SYSF2, Req\_NFSYS2

**Environment:** Browser for loading the webpages

**Operation**:

1. Open web browser
2. Go to “server\_ip\_address:port\_number” from localhost, sign up as a new user and enter user credentials
3. If correct credentials are entered, it will redirect to Dashboard
4. If incorrect credentials are entered, alert will be shown and access will be denied

**Expected Result**: Displays Tool Dashboard when correct username and password are entered

**Comment**: Before rendering the web page the Webserver should be started

**F-T2**: Monitoring data storage in Database

**Test**: RESTful API test

**Purpose**: To transfer data to a 3rd party using RESTful API

**Requirements:** Req\_SYSF1, Req\_SYSF2, Req\_NFSYS2

**Environment**: Terminal, SQLAlchemy

* Open terminal
* Enter “mysql –u root -p”
* Enter the database name using the command “select \* from users”

1. Check the user credentials of a particular user through the table displayed

**Expected Result**: Data is stored in the MySQL database

**Comment**: Care must be taken to reduce the overloading of the network which wouldlead to slow down

**F-T3**: Plot statistics of the node

**Test**: Testing of User metrics through graphical representation

**Purpose**: To check the generated graphs by python script using data in Influx database.

**Requirements:** Req\_SYSF2

**Environment**: Web browser and Grafana

* Open web browser and login to the web page using user credentials
* Click on “Statistics” on the Dashboard to redirect towards Grafana

**Expected Result:** Redirected towards Grafana with the graphs being plotted

**Comment:** It is used to view the graphical representation of a user’s system metrics

**F-T4:** Add/Remove nodes for Admin

**Test:** Admin can add or remove the nodes connected to it

**Purpose:** To give the admin the ability to manage nodes

**Requirements:** Req\_SYSF1, Req\_SYSF2, Req\_NFSYS2

**Environment:** Web browser

* Open the web browser and login (admin) to the web page
* Click on “Settings” and select “Add/Remove Nodes”

**Expected Result**: The desired nodes are added/removed from its network

**Comment:** Inactive nodes are removed for efficient functioning of the server

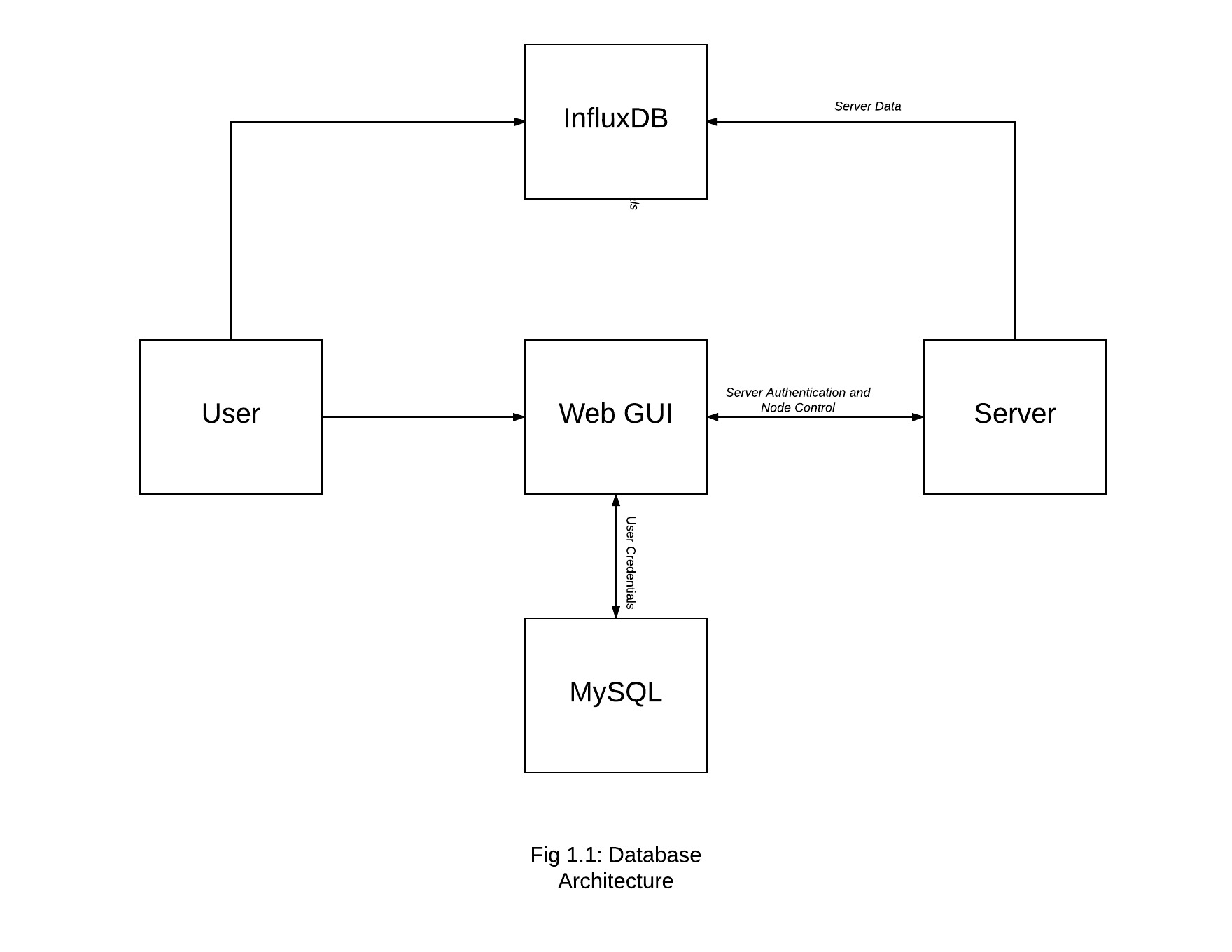
**MODULE 2: DATABASE MANAGEMENT**

This module consists of Database Management portion of the product. Database Management interacts with the user and the database itself to analyze and obtain the data.

**Influx** database is used in the project. This database acts as an intermediate between theother two modules that is the Frontend and Backend.

**Detailed Design:**

This module gives a detailed explanation about the database management in the product. In order to access, add and process the data stored in the database it requires Influx database management system. If the peer goes down the Syslog file is updated in the Influx database. Tables are created and loaded with data and the information is retrieved by selecting the data. Service time, uptime and the graphs related to uptime are displayed in the Web GUI while uptime graphs are drawn using Python script

****

**Unit Test Plan:**

**DM-1:** Influx database contains network measurements such as CPU utilization, system load, free and used memory as well as Disk I/O

**Test:** Data storage

**Purpose:** To confirm that database contains networkmeasurements and system metrics

**Requirements:** Req\_SYSF2

**Environment:** Influx database contains a table for network measurements and system metrics

* Pre-installed Python InfluxDB client
* Pre-installed InfluxDB

**Operation:**

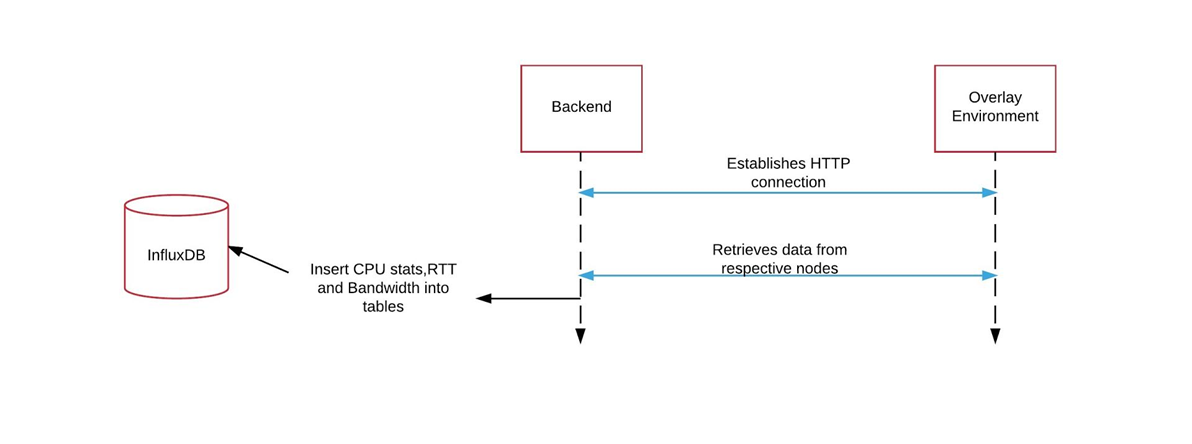
1. Insert the network parameters into the respective fields while we run the backend script
2. Check ‘cpu’ value in the server’s database for the network measurements for each service.

**Expected Result:** The CPU, Used Memory, Free Memory, read Disk I/O, write Disk I/O, RTT and Bandwidth are the metrics which are stored in field values

**MODULE 3- BACKEND:**

The backend interacts with the overlay monitoring tool, carrying out operations requested by the user from the frontend. The python script used as backend script uses http connection with server. It retrieves status data of service from the nodes and passes this data to the frontend to plot the graph.

|  |  |
| --- | --- |
|  |  |

 Fig 1.3: Backend Architecture

**Unit Test Plan:**

**B-1:** Syslog Facility

**Test:** Syslog functionality

**Purpose:** A log file in which the history of the nodes activities is logged

**Requirements:** Req\_SYSF1, Req\_SYSF2, Req\_NFSYS2

**Environment:**

* Web/REST server should be running
* Related python script should be running as well

**Operation:**

* Multiple requests from the users are recorded in the syslog

**Expected Result:** Whenever a node is down it is updated in the syslog facility by the admin server REST script

**Comment:** The admin server contains the syslog file which contains whenever the node is up or down

**B-2:** To ensure E-mail confirmation.

**Test:** E-mail confirmation

**Purpose:** To confirm the user that the account is created.

**Requirements:** Req\_SYSF1, Req\_NFSYS2

**Environment:**

* Web browser should be up and running

**Operation:**

* The group e-mail ID is used to send the confirmation mail to the registered user
* Python code is integrated with REST API code for sending the mail

**Expected Result:** The user must successfully receive the e-mail for confirmation.

**Comment:** User can view the account details in the confirmation mail.

**B-3:** To ensure that the encryption is disabled for debugging purpose.

**Test:** Mod3-Test\_3

**Purpose:** For identifying and removing the errors

**Requirements:** Req\_SYSF1, Req\_NFSYS2

**Environment:**

* Pre-installed Curl library for debugging purposes.

**Operation:** The system automatically restarts by disabling encryption after bug fixes.

**Expected Result:** Successfully restarts when the encryption is disabled after fixing the bugs.

**Comment:** Disabling the encryption before debugging makes the work easier.

**REFERENCES:**

1. <https://en.wikipedia.org/wiki/Front_and_back_ends>
2. [https://opennetworkingusergroup.com/wp-content/uploads/2015/05/ONUG-Overlays-](https://opennetworkingusergroup.com/wp-content/uploads/2015/05/ONUG-Overlays-Whitepaper_Final1.pdf)

* http://www.mi.parisdescartes.fr/~jelias/papers/PoliMi\_Thesis\_JocelyneElias.pdf