SIMS 2.0 Documentation

**Backend:**

The backend is based on REST/json. It is designed to be scalable and high performance.

Currently we have 9 sets of endpoints:

1. Company (Data Stored in Postgres)
2. User (Data Stored in Postgres + Redis)
3. Equipment (Data Stored in Postgres)
4. Sensor (Data Stored in Postgres)
5. Setting (Data Stored in Postgres)
6. Sample (Data Stored in Elasticsearch)
7. Weather (Data Stored in Elasticsearch)
8. Log (Data Stored in Elasticsearch)
9. Migrate (One-time use api to migrate the data from old system)

First five endpoints are working exactly the same. They designed to be consistent. Please note that before using these endpoints you should login using /user/login

List of HTTP methods to use with endpoints

1. GET: to get a single item or a list of them- Always returns json list
2. POST: to store a new item
3. PATCH: to update an item based in id. id should be in the body of form data
4. DELETE: to delete a single/group of item(s)

Format for GET/DELETE requests:

* Single filter: GET /company?filter=id;eq;1
* Multiple filters: GET /company?filter[]=name;eq;scentroid&filter[]=city;eq;toronto

List of HTTP response codes:

1. 200 : Request handled successfully, all methods (GET, POST, PATCH, DELETE)
2. 201: Request to store handled, but in the queue to process
3. 300: Request redirected
4. 400: Request is malformatted.
5. 401: Login required
6. 402: Your account role is not permitted to perform the operation
7. 403: You have reached the API call rate limit
8. 404: GET/PATCH didn’t find the item
9. 500: Server has got some error.

SIMS 2.0 Proposal

In this document we will cover the design and costs of SIMS 2.0.

### Features:

1. Equipment:
   1. It has sensors
   2. It belongs to company departments
   3. It belongs equipment groups
   4. It has certain custom settings (new setting: time zone)
   5. Calibration date+ Serial number + Active/Deactive
2. Sensor
   1. It has Type (Category like wind, temperature, humidity, noise, etc)
   2. It has Unit
   3. It has Port
   4. Its sample are shown in different colors (Green/Yellow/Red)
   5. It belongs to equipment
   6. There are factor to be multiplied for PID sensors gas
3. Sample
   1. It has Value + Timestamp (in UTC timezone)
   2. It has its location data (lat/lon/alt)
   3. It belongs to a sensor
4. Company
   1. It has name, address, telephone, logo
   2. New: It has a admin
   3. New: It has a distributor
5. Department
   1. It has name
   2. It belongs to a company
   3. New: It has a manager
6. Alarm
   1. It has one or more sensor involved
   2. It has threshold criteria for each sensor (more, less, between)
7. Log (will be stored in elasticsearch)
   1. It has time, severity and message
   2. It belongs to either a company or a server
8. Setting
   1. It has category, name, value, help message
   2. It belongs to a company
   3. It supports templating
9. Chart Annotation
   1. It has note and user id
   2. It belongs to a sensor and time
10. User
    1. It has user name, family
    2. It has role, password, password expiry date, last login date
    3. New: It belongs to a department
11. Role
    1. It has name
    2. It has access scope (URLs)
12. OAuth Client
    1. It has name
    2. It has access scope (URLS)
    3. It belongs to an equipment
13. Token (will be in redis)
    1. It has value, expiry time
    2. It belongs to a user or client

### Software Costs Analysis

Software Developments:

* Backend:

Backend with be implemented with Tornado Python Framework

* + CRUD for 13 Entities: Equipment, Sensor, Sample, Company, Department, Alarm, Log, Setting, Chart Annotations, User, Role, Client, Token
  + Swagger UI for the API:
  + OAuth 2.0 for Scentinels and third Party Software access (5 hours)
  + Wind Area Calculator API
  + API Rate Limits (5 hours)
  + Scentinel API receivers (2 hour)
  + Scentiner API Redis Queue to ElasticSeach
  + Software/Scentinel Health Checks
  + New Alarm Conditions and Mail Trigger
  + New Weather status lookup and integration
  + New Air Quality Index (AQI) **(from where?)**
  + Reports **(what reports?)**
* Frontend

It will be implemented Using React.js

* + New Theme (Using new UI materials and better font, colors):
  + New Dashboard with all-in-one date-time slider: 10 hours
  + New Map (leaflet.js):
  + Form with interactive input validation rules (http://formvalidation.io/)
  + OAuth Allow/Deny Access UI:
  + New UI for alarms:
  + Colors for different sample levels (Green/Yellow/Red):
  + New Chart Libraries + Same Export options + annotations (charts.am or d3.js):
  + Multi-language (i18n):
  + Walk-through help (intro.js):
  + Integrated Wikipages and Support:
  + Health Check Dashboard:
  + Forget password:
  + Integrating with Backend:

Configurations and Operations:

* Deploying API/Frontend servers:
* Migrating current data to the databases/storages:
* Wind Area Calculator software:
* Load Balancer:
* Automated Backups:
* Database Server + Data storage:
* Automation Server:
* Writing this document:
* Writing Technical Documentations for IT/Operation of IDES

### Hardware Costs Analysis

**Servers:**

1. Load Balancer Gateway Server(1\* t2.micro) total monthly cost:
2. API Server for Sentinels (2 \* t2.nano)
3. API Server for Clients (1 \* t2.nano)
4. Automation Server (Jenkins)(1\* t2.nano)
5. Frontend Server (1\* t2.micro)
6. Backend Server (1\* t2.micro)
7. Database Server (1\* t2.small) Postgres Server: Users/ Equipements/ Anything table   
   Redis: Database For Queuing  
   Redis: Temporary API security tokens
8. Data storage cluster (2\* t2.small)

**Disk Usage:**

10 GB cold HDD for server + 10GB Magnetic for elasticsearch 10 GB for database server = 1.92

**S3 backups**

**1TB**

We can scale up as we grow. There will be more costs as we grow. Also please consider %10 extra misc costs.

There will be one year discount on most of the products. You will be paying about $100 dollars per month for the first year: <https://aws.amazon.com/free/>

For number 8: <https://aws.amazon.com/elasticsearch-service/pricing/>

For the rest of servers: <https://aws.amazon.com/ec2/pricing/on-demand/>

If you pay one year upfront, the server cost could decrease from 40-75%:

<https://aws.amazon.com/ec2/pricing/reserved-instances/>

For disk: <https://aws.amazon.com/ebs/pricing/>

# 

# Design

This document is the baseline for designing the features and improvements in SIMS.

### System level design

Current system has many weak points. The first weak point is its availability and data loss. To improve it, we need to add more components to it.

#### New components

1. **Queue server**

We should prevent direct access of any web request to our web application and data storage. This is more for stability and performance

I explain the reason with one example: Imagine due to a bug or some hackers attempt, we receive a lot of data api calls. Right now, we log all of them and save them to DB. No matter how much huge is the number of concurrent connections and how long it does take, we will do it do the best performance of our server. At the same time, users of the dashboard will experience slow UI. The reason is that DB is busy with saving that huge amount of data and the same server is handling http api calls. This brings up the necessity of separating the layer of receiving data.

Nowadays, people use Queue systems like Kafka and Redis. Both of the are simple queues, that data is inserted from one side and you can read from other side. The other good reason to queue servers is we can have many readers, reading from the queue at the same time. Like the code that writes to the database, the code that writes to archive, the code that does machine learning and so on.

1. **Distributed Data Storage**

Mysql and RDBMS databases are good when being used for single server setup and when we have 1-1 ratio of reads and writes. But for our case, we need a solution that can tolerate losing the server and also can help us to read fast and write in a moderate speed. Elasticsearch is the state-of-the-art system to store such data.

1. **Automation server**

In current system, we use cron to have automation. This is the most basic automation in Linux. Cron is the system application that runs the command on the time that you specify in the configuration. However, cron is not suitable for tracking the status of process or keeping the history. Imagine we want to make sure Alarms are sent to our customers when there was an event in the system. Or we want to run a code periodically to pull out the events in the sensor data. In such cases we should use Automation Servers like free open-source servers Jenkins. Jenkins has a dashboard and you can define any code to be automated.

1. **Monitoring server**

Currently, we don’t have any monitoring on our servers. If SIMS goes under pressure or its database, we should recieve notifications to resolve the issue as soon as possible. There are free open-source tool for this purpose. The popular one is called Zabbix. Zabbix has a central server that you can define everything in its UI and then you can install its small software agent on other servers to send the server CPU/Disk/Ram/Network/etc data to the central server.

## Sims for internet-less customers

We can use multiple virtual machines on the PC that we give it to them. Each virtual machine is one of our servers.

# Ideas

## Front End

1st Phase

· Login page should be improve in terms of design

o Similar to webpage design but just one page and says login and password

o Add forgot password?

· I agree with this bullet point

· After login there should be a dashboard where the communication status of the SL50 can be shown on a map. A 3D google map should be shown which cover all SL50s of the company owns them

- Is having 3D map necessary? 3D maps are difficult to navigate and render. We can have a 2D map with satellite images. Likewise what you have in the next page.

- The 3D map is misleading. It is only an image and you can’t navigate it.

I think we are better off using a normal map (google or otherwise).

So the final decision is to use 2D map.



<<Can be straight google MAPS>>

· Each SL50 should have a name (description) in addition to Serial # which make more easy for end user to refer

- How about adding a nickname of user-choice to each sentinel? Then they can refer them however they want.

Yes this is what we were thinking too

I consider this as approved

· Most of the buttons should have a help/info description for end user

- We should create a help section in the software. Like a visual guide to let them know how to stuff, before doing them. Also, there is a complimentary way. Using walk-through help. Click on “demo” on this page and see it: <http://introjs.com/>

OK I like the introjs demo. It is easy to put lots of hints

I consider this as approved

· By hovering over the SL50s on the 3D google map in the beginning after login in the status or graph of the SL50s should be shown

- Hovering needs mouse. Touch screens cannot do hovering. Also, hovering makes UI a bit complicated since the user should hover everything to make sure he is not missings anything.

OK you are right. Maybe having a button or clicking on them to make baloons apear giving condensed info with option “more” to go into the details of the specific Scentinal itself.

Sure. So the main element of the dashboard will be a big map.

· In the map user should be able to define the data (sensor) reading to display (on the left side can be showing display “sensor, status, alarms, etc) and hovering over would give more details such as a graph and/or exceedances



(no only for the above point).

This system below is has devices with one sensor per device. Let’s say there are different Scentinels with one sensors. That is why the map and the chart on it looks neat. We can design the same thing with the same layout. However, as a web-designer, this UI seems photoshop to me.

Well that is why we wanted to have a feature to pick what you need to see. For example you pick H2S and all H2S readings from all sensors are shown on the map like the picture above. Of course not hovering as you mentioned but clicking on each would produce a graph of that device for the last 24 hours (fixed or variable span of time for the graph)

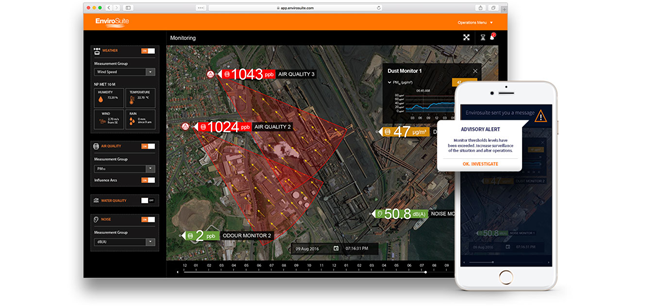
Sure. Understood it. We be in the design

· In one graph end user should be able to choose different SL50 or different sensor of the SL50s be represented in same graph

This is not for the picture but just in the graph page like the one we already have we want to be able to graph sensors from different scentinals in the same graph.

· Each sensor should have category (Air quality, Noise, Health,

· Affected area of the wind should be presents on the 2D map for air quality sensors



Wind direction sensor: If we don’t use the Internet, There is no wind direction. Also, even we have the wind direction, it is only for the weather station that is usually in the airport, not in the factory. Therefore, the wind direction needs some real sensor.

The wind sensor is a real sensor in many Scentinals or we use the nearest public weather station. If there is no internet and no sensor then obviously this feature would be deactivated. Most of our clients are using our cloud server so there won't be an issue with internet.

The idea is only to draw a simple triangle based on a angle that is calculated based on wind speed and a direction that is calculated based on wind direction.

Sure. Understood it.

- Are these four bullet points above, describing the software below? If yes, I will answer them below:

(no only for the above point).

To have the same layout we need these features:

- - For red/green/yellow we need some threshold numbers. Per sensor type.

Yes this is from the AQI thresholds that we talk about below. They are user defined in sensor setup.

· Login in level:

o Root admin (Scentroid)

Can create all user levels

Can create new Scentinals, companies, and define sensors for scentinals

Can setup alarm levels

Can see warning and health of scentinals

Can see log of raw data transmissions

Can change calibration setting

o Distributer

Can see data from multiple companies under his account.

Can create Manager level and below user accounts

Can’t create company scentinal or define sensors

Can change calibration setting

o Manager

o Users

o Public What does this do? Currently, we don’t have anything for public rather than login page.

Here is a table for clarification

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User level | Create Users and change passwords | Create New Scentinal/Company, and define sensors | Setup alarm Levels | See warnings and alarms | See log of raw data | Change Calibration Setting | See collected data | Generate Reports | Delete Data |
| Root | All | X | X | X | X | X | X | X | X |
| Distributor | Manager/User/Public |  | X | X | X | X | X | X |  |
| Manager | User and public |  | X | X |  | X | X | X |  |
| User |  |  |  | X |  |  | X | X |  |
| Public |  |  |  |  |  |  | X (selectable sensors) |  |  |

· 3 alarm condition should be available to be setup for each sensor (and Total Odour). Exceedance should be based on individual reading or averaged reading (1 hour, 8 hours, 24 hours). Odour is 10 min average which is based on sampling period of the Scentinal.

· After changing setting, deleting data, etc, confirmation page should ask for yes/no

- Agree

· In login page we can have slider instead of one picture to show different industries/cities

- What is the benefit in login page? If it is marketing purpose, it is useless. If it is the cities, it might be confidential.

The login page is for all scentinals (sims.scentroid.com) it is for advertising as the first image the client sees is the login page

Sure. Understood it.

· General report form need to be design so that end user can download reports in pdf format

- Reports needs to be defined before being designed. Please define them as soon as possible. We should know about the desired reports before designing any software or data structure.

OK we will make an example report and send you.

· Adding Air Quality Index (AQI) to : dashboard and data management page

o 6 levels of user defined thresholds. Colors fixed (green to burgundy)

- Does it have a sensor or should we get it from internet? The internet data is not valuable because it is the sensor data from airport telemetry station.

This is just colour coding based on sensor reading. For example for H2S 0-10 ppb is green, 10-30 ppb is yellow, etc. The colours will be set to 6 colours we will provide as RGB values. The thresholds (10,30, etc in example above) would be user defined in Sims.

· User to be able to add note on graph for describing any peak or unusual status (should be recorded in the database)

- Annotation notes can be added, but should be personal or company-wide visible?

Company wide visible

· Exceeds\* chart (show how often the data exceeded thresholds)

o Number of 1 hour and 8 hour exceedances per day

o Number of 24 hour exceedances per month

2nd phase

· Predictive model (TOMS)

- Needs more description

This is much down the line. If you want we can explain now but it is extensive and we dont want to implement now.

· Should we go for Application or mobile user friendly website is sufficient

- App is always better. If we use react, we can use the same UI components inside android and iOS. It is not HTML5 app. It will convert to native UI for those operating systems. (please look up: react native)

The push notification is a great idea and having an app like you said should be faster. We just need to know what is the downside? Cost, reliability, etc.

Push notification to some extend is free and provided from Google. There are services out there and we can have our own server too based on opensource software.

App is dependent and can make our web-application designed simpler. However, having an app and maintaining it is costly a bit. Apps can use the APIs. Therefore, they would be very expensive but still, having an app developer working for the project is expensive.

· Translate?

- I think this should be in the design of first phase. If we don’t foresee i18n (internationalization) from now, it will be omitted.

The clients do ask for this so lets see what is the process? Do we just need a file with translations or is there more involved programming

A file is enough per language.

## Backbone:

· Having mirror sites on different location

- Replication is important. For speed and latency.

OK but anything in particular we need to do for this on design stage? How do the servers sink and how do they assign workload?

I will write down the details

· Improving alarm API

- We should improve it in different aspects:

1) It should be able to handle more complicated criteria. Like when two sensors are reporting bad numbers at the same time.

2) We should start to use “push notifications” beside emails. This way, users can see the notifications on the their phones using our app or third party apps.

3) We should use template texts like what we have for diagnosis API.

· Having health page for Scentroid

- First we should define the health criteria. I suggest these criteria:

1) Networks, Memory, Disk, CPU of the Raspberry Pi machine to help us find faulty ones.

Yes Ameer please define some health: I put some examples below

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Description** | **Action to be taken by admin** | **Scentinal action** | **Message type** |
| Pump Wear | Current exceed x mA | Schedule pump replacement in 30 days or less | Warning with mA reading | Warning |
| Pump Failure | Current Exceed x mA | Replace pump immediately and restart system | Scentinal shuts off pump until restart | Critical Alarm |
|  |  |  |  |  |

2) Raspberry Pi and Arduino Firmware versions.

3) Latest software error messages shown in each machine. This will help us solve the issues before the customers start to complain. All mobile apps and web application nowadays collect and analyze the internal error message.

4) Last connection time and speed of data transmitting

Please add any more criteria you think it is vital.

· Encryption of data transfer

Right now, we use HTTPS. HTTPS uses AES and RSA. There is not point of making the encryption more complicated. You can buy stronger certificates to have the data more secure.

But we should have better handshake. Right now, anybody with the serial number can send data or receive settings. We should use one-time/expiring passwords and force the Scentinel to use it for every connection. If you have better suggestion please advise.

I think on-time or expiring passwords is enough.

I consider this as the decision.

· Admin (meaning Root) dashboard should be designed to be simple, searchable and also adding filter:

- Filter?

To put filter for example to show all scentinals in this company or all the ones in South Africa or all the ones online. We need to define the filter criteria.

- We can add a searchbox for features/setting-name/help-topics. The user types there and receives suggestions. Clicking on any item in the list sends use to correct page.

Yes that is also good idea but this is only for root.

We can have it for others too. Any functionality can have hel/walk-through.

· Should we send mv or ppm?

- This is vague. But if you mean the data, I would rather data conversions happens once and in one central location. The devices do not need to do extra work.

OK lets leave it as ppm but we should record mv on the SD card.

That is the electronic board. You know how to do it.

· Having a template SL50 with default specs, / having dropdown list of sensors for increasing speed of setup of each SL50

- Default setting for a Scentinel or “replicating one Scentinel”. Also, we can implement a json file for a Scentinel to save/load it as a file. Great

I consider all as approved.

**More Technical Thoughts:**

- Google Maps:

Google Maps is NOT free. You will be charged as soon as requests are increasing. You can check your API free capacity in Google developer console. We should better and customizable solutions that we could give our own offline maps to it. I would suggest this map:

<http://leafletjs.com/>

This looks good. Its also much smaller and maybe can run on RPI without internet access

I consider this as approved.

- Integrated support:

When there is an issue, the users should be able to open a support ticket within the software. Emailing you for each issue is not a desirable user experience.

This is a great idea

I consider this as approved.

- Integrated help:

We should add help topics inside the frontend app.

OK

I consider this as approved.

- Integrating SIMS-cloud and LIMS:

Having two separate software is not ideal. Maintaining them is a headache. Also, this will help you for marketing lab technologies of LIMS in SIMS. It can be shown as a disabled menu item inside the app.

No we dont need LIMS to be same as SIMS. It is for seperate thing and LIMS is not that important as SIMS.

I consider this NOT approved.

- User groups:

We might need to categorize users into different departments. Like branches of a factory. The should not access the same Scentinels.

OK

I consider this as approved.

- API Capacity Quotas:

These days, we have limited numbers of users accessing our API for Data API. However, Querying too much from our database can degrade our system performance. We should start to implement Hourly limits on the URLs. Not even API but the Front end too. Because the users can use automated tools to get data from our front end chart APIs.

OK

I consider this as approved.

- OAuth 2.0 for third-party access:

Right now, when we give Data API access, the customers have customer access key and secret code to access each Scentinel. Although, this is safe, but there are more secure and flexible standards implemented.

Users can give access to Third-party apps as they want. Also, we can define different levels like read/write/action. This can be not limited only to Data API. They can manage their users and Scentinels.

Having OAuth 2.0 is useful for us too. When we have SIMS mobile app, it can authenticate using OAuth 2.0.

SOunds good

I consider this as approved.