Senslify User Manual

Version 1.0

By: Christen Ford

V 1.0.0

Table of Contents

[1: Introduction 3](#_Toc75265567)

[2: Terminology 3](#_Toc75265568)

[3: Pages 3](#_Toc75265569)

[3.1: Index/Group/Home Page 3](#_Toc75265570)

[3.2: Sensor Listings Page 3](#_Toc75265571)

[3.3: Sensor Information Page 3](#_Toc75265572)

[3.3.1: Request Statistics Modal 3](#_Toc75265573)

[3.3.2: Request Download Modal 3](#_Toc75265574)

[4: Conclusion 3](#_Toc75265575)

# 1: Introduction

Welcome to Senslify! Senslify is packed with features meant to make visualizing sensor data as simple as possible. With this software, you can view sensor data in real-time, download sensor data for a given period of time, acquire basic statistics on sensor readings for a given period of time, be notified when sensor readings are above or below an administrator configured threshold, provision sensors into the system all by using a simple REST/HTTP and WebSockets based API. Senslify is built to be as lightweight as possible with a minimalist design scheme and low learning curve.

# 2: Terminology

This section explains the various terminology you may run into when working with the Senslify system. If you encounter any terminology that you do not understand or require clarification on, please reference this section of the user manual.

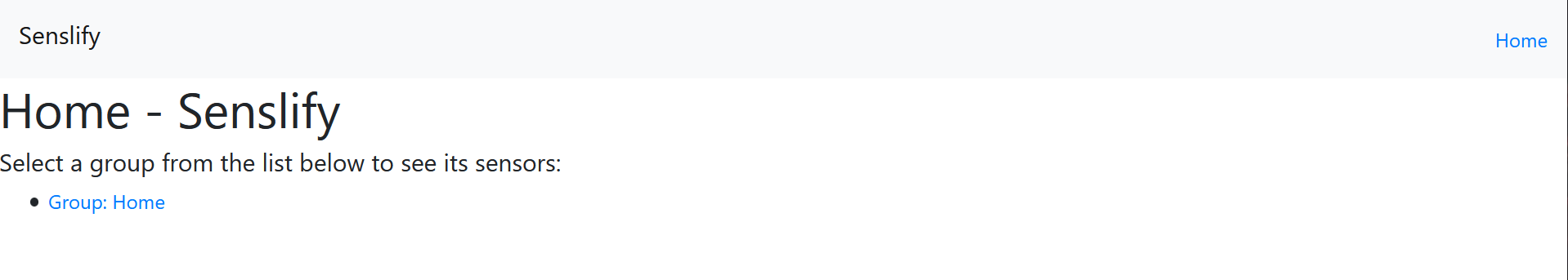
* Application Programming Interface (API): A set of functions exposed by a software developer that allows external software to interface with supported software.
* Cascading Style Sheets (CSS): HTTP-compatible API that allows web developers to style web pages. Determines what web pages look like.
* Gateway: An external device that aggregates sensor readings from sensors and uploads them to the system.
* Group Alias: A plain-English name that stands in for a group identifier in the user interface.
* Group Identifier: A positive integer that represents a group of sensors.
* Hyper-Text Transfer Protocol (HTTP): Application-level protocol that drives the Internet. Determines the content and structure of web pages.
* HTTP Endpoint: A publicly exposed route on a specified domain that lets clients interact with a web server hosted at that domain over the HTTP protocol.
* The Internet: A global system that interconnects disparate, non-homogenous devices and allows them to communicate with each other.
* Internet of Things (IOT): An area of computer research that focuses on the protocols, hardware, and software necessary to bring the Internet to non-traditional computational and electronic devices such as automobiles, ovens, refrigerators, wireless sensors, etc…
* Javascript (JS): Scripting language that allows developers to embed behaviors into web pages. Determines how web pages behave.
* Javascript Object Notation (JSON): Data storage format based on data structures from Javascript. Commonly used for semi-structured data and data exchange in the Internet.
* Provisioning: The act of preparing a system or network to provide new services to its users. Provisioning is used by Senslify to allow users to add groups and sensors to the system.
* Reading Type Identifier: A positive integer that represents a unique type of reading that can be generated by a sensor.
* Representational State Transfer API (REST API): A fancy name for an HTTP-based API that is used to transfer data from one machine to another.
* Route: An alternative name for an HTTP endpoint.
* Sensor Alias: A plain-English name that stands in for a sensor identifier in the user interface. These are generated automatically by the provisioning system.
* Sensor Identifier: A positive integer that represents a sensor within a group of sensors.
* Web Socket: Communication protocol defined in IETF RFC 6455 that allows for creating persistent connections over the HTTP protocol. While the HTTP protocol itself creates persistent connections through the TCP protocol, these connections serve a single purpose. Web Sockets have the capability to create long-lasting connections between any two machines that can be used for pretty much anything.
* Wireless Sensor: Any embedded device that can generate sensor readings drawn from its environment and report them to some other system.

# 3: Pages

This section details the pages the user encounters when using the Senslify web-browser based client. Each section shows a screenshot of the page as well as a description of what the user is seeing on the page.

## 3.1: Index/Group/Home Page

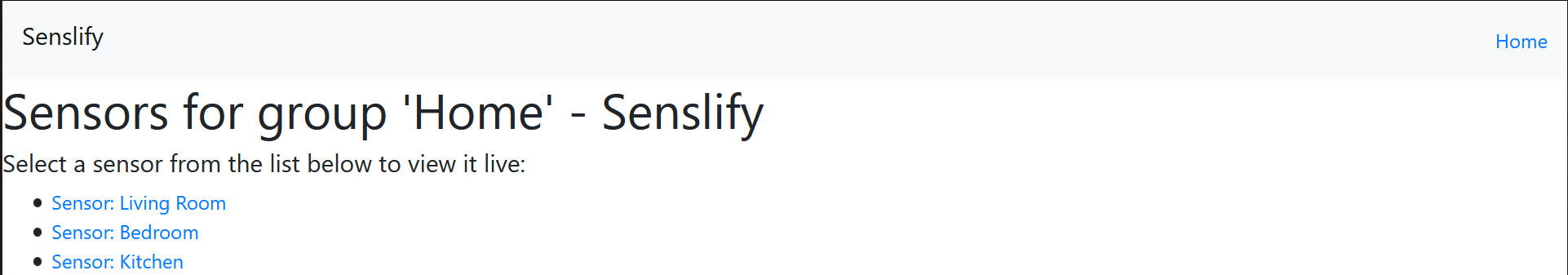
This is the first page users see when they visit the Senslify web client. It looks like this:



This page displays links to each of the groups currently provisioned in the Senslify database. Clicking one of these links will redirect the user to the Sensor Listings page for that group which contains links to each of the sensors that are provisioned within that group.

## 3.2: Sensor Listings Page

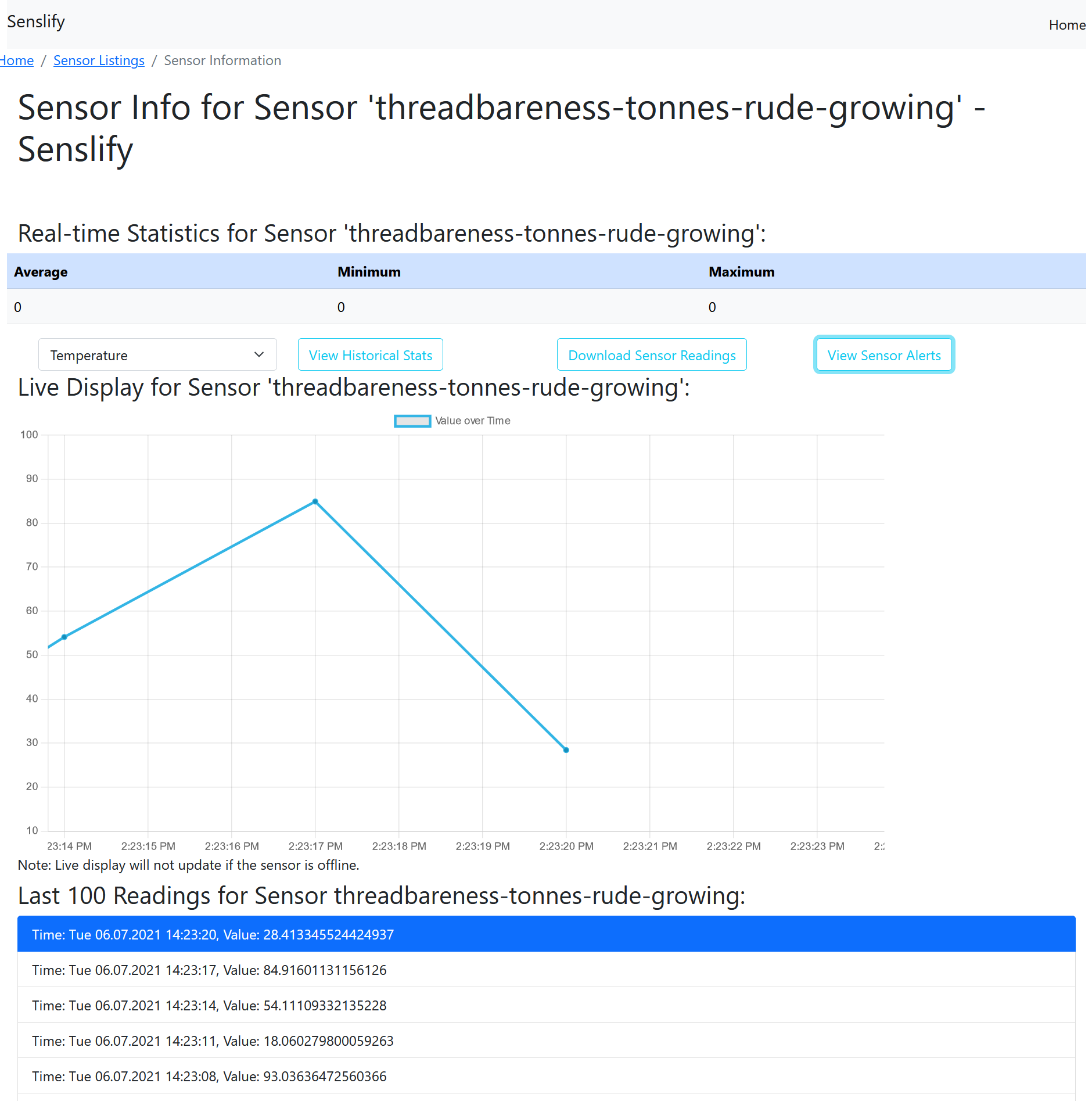
This is usually the second page users see when they are using the web client. It looks like this:



Like the home page, this page lists clickable links to each of the sensor in group the user clicked on in the prior page. Clicking one of these links will redirect the user to the Sensor Information page for that sensor.

## 3.3: Sensor Information Page

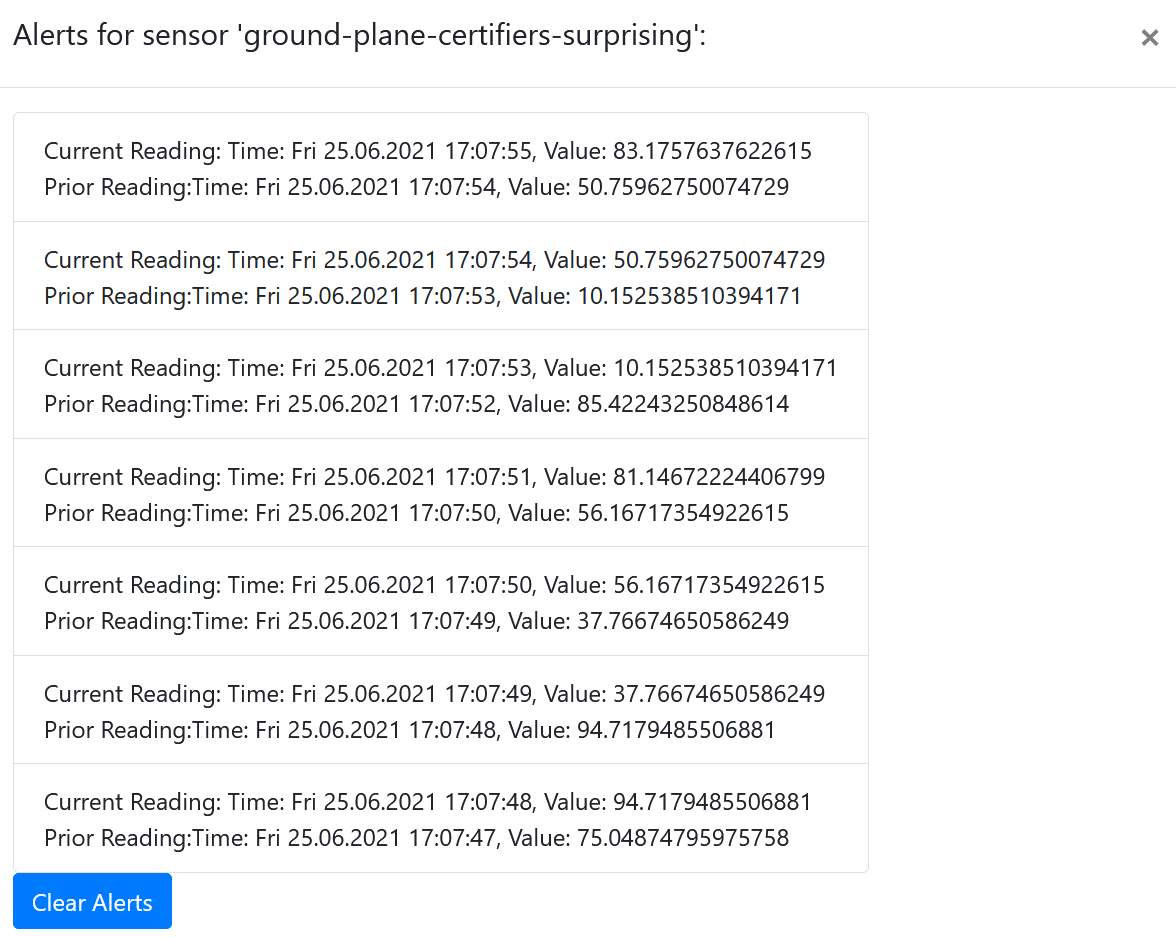
This is usually the third page the user sees when they are using the web client. It looks like this:



This page has a lot more going on then the prior two pages, so I will take some time to explain it. At the top of the page, you can see the average, minimum, and minimum sensor readings for the lifetime the user has been viewing this page with the current sensor reading type. These values are all initially zero, but are updated each time a sensor reading comes into the system corresponding to the selected reading type or when the user changes reading types. Below this are three buttons that the user can click on that open various modal windows offering additional functionality including viewing historical stats, downloading sensor readings, and viewing sensor alerts. Below this is a live chart that displays sensor readings on a line graph as they are uploaded to the system. Below this is a drop-down selector that allows the user to specify the type of readings they want to receive. Note that not every sensor will report readings for each type of reading. Lastly, below this is a real-time listing of the top-N (web server configurable) sensor readings where the most recent reading is the first reading highlighted in blue with readings sorted in descending order from most recent to least recent.

### 3.3.1: Alerts Modal

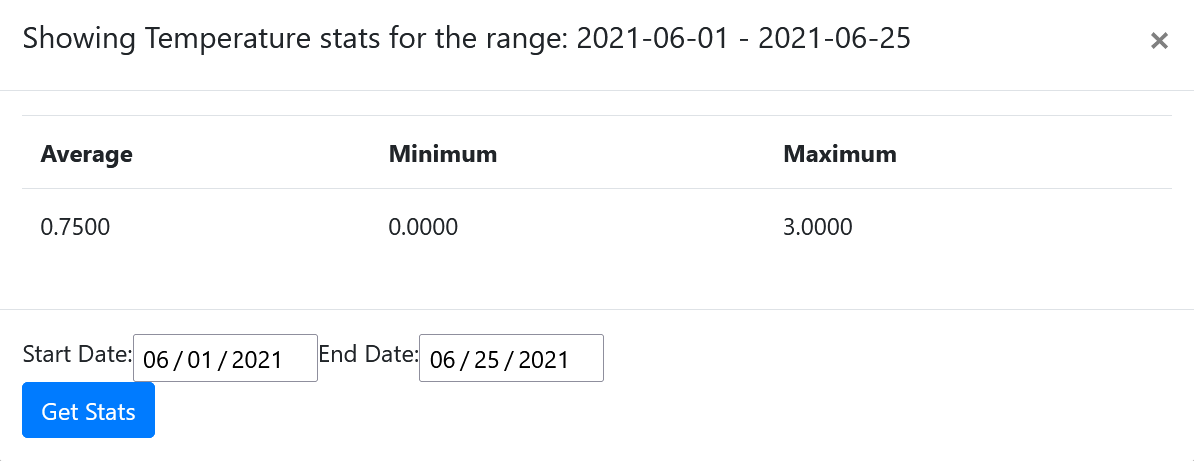
The user can open this modal by clicking on the `View Alerts` button on the Sensor Information page. It looks like this:



This modal window lists all of the alerts generated by the system for the currently selected sensor and sensor reading type. It lists the reading that caused the alert (as the Current Reading) and the prior reading. These alerts are always sorted in descending order. The user may clear the alerts by clicking the `Clear Alerts` button. They may also close the modal by clicking the ‘X’ button in the upper right of the modal.

### 3.3.2: Request Statistics Modal

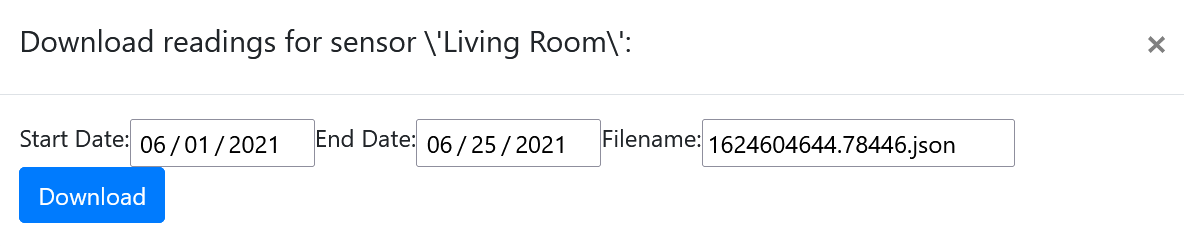
This modal window allows the user to request statistics for the currently selected sensor reading type. The values reported in this window will always be calculated over the time window specified by the user when they submit their request. If the user specifies a window where no readings exist for the currently selected reading type, then the values displayed here will not update and will remain 0. This modal window looks like this:



The user specifies the start date by selecting it from the start date picker. Likewise, the user specifies the end data by selecting it from the end date picker. The user requests stats by clicking on the `Get Stats` button and can close the modal window by clicking on the `X` button in the top right of the window.

### 3.3.3: Request Download Modal

This modal window allows the user to request downloads for sensor readings over a given period. Readings downloaded using this modal window will include readings of any sensor reading type. This modal window looks like this:



The user specifies the start and end date with the start date and end date pickers. A filename is automatically generated for the user prior to them clicking the Download button. They can however change this filename by typing a new filename in the filename textbox. The `Download` button initiates the download while the `X` button closes the modal window. Currently, there is no way to cancel a download request once it has been made and there is no confirmation in the browser that the download was successful other than the file appearing in the users local download directory. These issues will be addressed in a future release. Note that the web server always formats the response to a download request in JSON format. As such, the user should always make sure that the filename has the `.json` filename extension. Any other extension will prevent the users operating system from properly interpreting the file.

# 4: Routes

This section details the HTTP endpoints or routes that are accessible through the Senslify REST API. Clients may access these routes through any HTTP-compatible client, though the most common use case will be through a web browser, gateway, or wireless sensor. The REST API exposes the following endpoints that are all accessible via an HTTP-enabled device. The user only needs to properly format their request and await the servers response. The following table lists the commands available via the REST API.

|  |  |  |
| --- | --- | --- |
| **API Commands** | | |
| API ID | Name | Description |
| REST-000 | find | Retrieves documents for pertaining to the target and filters them based on provided parameters. Always accepts a JSON object and returns a JSON array. |
| REST-001 | stats | Retrieves sensor reading statistics based on provided parameters. Always accepts a JSON object and returns a JSON object. |
| REST-002 | download | Retrieves sensor readings based on provided parameters. Always accepts a JSON object and returns a JSON array. |
| REST-003 | upload | Uploads sensor readings to the system. Always accepts and returns a JSON object. |
| REST-004 | provision | Provisions a group or sensor into the system. Always accepts a JSON object and returns a JSON object. |

The following table lists the parameters the API accepts as input for each command. Note that the API expects these inputs to be sent in a JSON object.

|  |  |  |  |
| --- | --- | --- | --- |
| **API Input Parameters** | | | |
| API ID | Name | Type | Description |
| ALL | cmd | String | The API command to run. This must match one of the commands specified in the previous table. |
| ALL | params | JSON object | The parameters for `cmd`. These are the parameters that will be passed to the API command when `cmd` is routed internally by the API handler. The API will return an error if the `cmd` is anything else. |
| REST-000 | target | String | The target for the find command. This must be one of ‘groups’, ‘rtypes’, ‘sensors’, or ‘readings’. The API will return an error if the `target` is anything else. |
| REST-000 | groupid | Integer | The group identifier to use in `sensors` or `readings` targets. Only applies when the `sensors` or `readings` target is specified. Is ignored by all other targets. This must be a valid group identifier otherwise; the API will return an error. |
| REST-000 | sensorid | Integer | The sensor identifier to use in the `readings` target. Only applies to the `readings` target. Is ignored by all other targets. This must be a valid sensor identifier otherwise; the API will return an error. |
| REST-001 | target | String | The target for the stats command. This must be one of ‘groups’ or ‘sensors’. The API will return an error if the `target` is anything else. |
| REST-001 | groupid | Integer | A group identifier. This must be a valid group identifier otherwise; the API will return an error. |
| REST-001 | rtypeid | Integer | The reading type identifier. This must be a valid reading type identifier otherwise; the API will return an error. |
| REST-001 | start\_ts | Integer | The start timestamp for the window. This must be less than the end timestamp otherwise; the API will return an error. |
| REST-001 | end\_ts | Integer | The ending timestamp for the window. This must be greater than the start timestamp otherwise; the API will return an error. |
| REST-002 | sensorid | Integer | A sensor identifier. This must be a valid sensor identifier otherwise; the API will return an error. |
| REST-002 | groupid | Integer | A group identifier. This must be a valid group identifier otherwise; the API will return an error. |
| REST-002 | start\_ts | Integer | The start timestamp for the window. This must be less than the end timestamp otherwise; the API will return an error. |
| REST-002 | end\_ts | Integer | The ending timestamp for the window. This must be greater than the start timestamp otherwise; the API will return an error. |
| REST-003 | readings | Array | A JSON array containing properly formatted JSON objects. These JSON objects should always have a `groupid`, `sensorid`, `rtypeid`,`ts`, `val` of types integer, integer, integer, integer, and float respectively. The `groupid`, `sensorid`, and `rtypeid` fields must be positive and must exist in the system otherwise; the API will return an error. The API will also return an error if the  `ts` field is not positive. |
| REST-004 | target | String | The target to run the `provision` command against. Must be one of `sensor` or `group` otherwise; the API will return an error. |
| REST-004 | groupid | Integer | The group identifier to use with the `sensor` target. Only used by the `sensor` target. Is ignored by all other targets. Must be an existing group in the system otherwise; the API will return an error. |
| REST-004 | alias | String | The plain-English alias to use with the target being provisioned. Optional. If not supplied as a parameter, the system will generate an alias for the target. |

Note that in the above table, any API ID of ALL indicates that the parameters must be included in all requests made to the API or is returned by all responses generated by the API. The REST API exists at the `/rest` endpoint. Say the web server is running at 192.168.0.5. To make a REST request, the user would create a JSON object conforming to input fields in the second table above for the command they want to run. They would then send this object to the `http:// 192.168.0.5/rest` endpoint using their web client. Assuming the request is formatted correctly, the server will execute the command and send back the results in the proper format specified in the first table above as well as an HTTP 200 status code. If the command fails to execute, the server will send back a JSON object containing an error message keyed to the `error` parameter as well as an HTTP 400 or 403 status code depending on if there was an error with the parameters or if the server ran into an issue.

# 5: Conclusion

This user manual is entirely exhaustive as to the Senslify system. If you encounter any bugs or outstanding issues please check the Github repository first to make sure no one else has already filed a bug report for your issue. If your bug report is unique and you can reproduce bugs you are experiencing, please create a bug report including the following information:

* Web Browser/Client Name and Version
* Steps needed to reproduce the problem
* Solutions you attempted to fix the problem
* The severity and impact of the problem

After you complete your bug report, open an issue in the Github repository with your bug report as the description at: <https://github.com/gollum18/senslify-web>. Note that the more detail you include in your report, the easier it is for the maintainers to track down the problem.

If someone else has already experienced the bugs you are reporting, please include the above information in a comment on the existing Github issue. This helps the maintainers prioritize which issues need addressed first and which are not as important.