

---

**succulent**

*Release 0.2.5*

**Tadej Lahovnik, Iztok Fister Jr.**

**Aug 18, 2023**



# USER DOCUMENTATION

<b>1</b>	<b>General outline of the framework</b>	<b>3</b>
<b>2</b>	<b>Detailed insights</b>	<b>5</b>
<b>3</b>	<b>Documentation</b>	<b>7</b>
3.1	Getting started . . . . .	7
3.2	Installation . . . . .	8
3.3	Testing . . . . .	9
3.4	Documentation . . . . .	9
3.5	API Reference . . . . .	9
3.6	License . . . . .	12
3.7	Contributor Covenant Code of Conduct . . . . .	13
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



succulent – Collect POST requests easily

- **Free software:** MIT license
- **Github repository:** <https://github.com/firefly-cpp/succulent>
- **Python versions:** 3.7.x, 3.8.x, 3.9.x, 3.10.x, 3.11x



## **GENERAL OUTLINE OF THE FRAMEWORK**

Sending sensor measurements, data, or GPS positions from embedded devices, microcontrollers, and [smartwatches](#) to the central server is sometimes complicated and tricky. Setting up the primary data collection scripts can be time-consuming (selecting a protocol, framework, API, testing it, etc.). Usually, scripts are written for a specific task; thus, they are not easily adaptive to other tasks. [succulent](#) is a pure Python framework that simplifies the configuration, management, collection, and preprocessing of data collected via POST requests. The inspiration for the framework comes from the practical data collection challenges in [smart agriculture](#). The main idea of the framework was to speed up the process of configuring different collected parameters and providing several useful functions for data transformations. The framework allows users to configure the whole endpoint for data collection in several minutes and thus not spend time on server-side scripts.



## **DETAILED INSIGHTS**

The current version includes (but is not limited to) the following functions:

- Request URL generation for data collection
- Data collection from POST requests
- Storing data in different formats (CSV, JSON, SQLite, images)
- Defining boundaries for collected data (min, max)



## DOCUMENTATION

The documentation is organized into the following sections:

- *User documentation*
- *Developer documentation*
- *API Reference*
- *About*

### 3.1 Getting started

This section demonstrates the usage of the succulent framework.

#### 3.1.1 Installation

To install the succulent package, run the following command:

```
pip install succulent
```

#### 3.1.2 Usage

##### Configuration

In the root directory, create a file named `configuration.yml` and define the following:

```
data:
  - name: # Measure name
    min: # Minimum value (optional)
    max: # Maximum value (optional)
```

To collect images, create a file named `configuration.yml` in the root directory and define the following:

```
data:
  - key: # Key in POST request
```

## Example

```
from succulent.api import SucculentAPI
api = SucculentAPI(host='0.0.0.0', port=8080, config='configuration.yml', format='csv')
api.start()
```

## 3.2 Installation

### 3.2.1 Development environment

#### Requirements

- Python: <https://www.python.org>
- Poetry: <https://python-poetry.org/docs>

After installing Poetry and cloning the project from GitHub, execute the following command in the root directory of the cloned project:

```
$ poetry install
```

All of the project's dependencies should be installed and the project should be ready for further development. Note that Poetry creates a separate virtual environment for the project.

#### Development dependencies

List of succulent's dependencies:

Package	Version
pyyaml	^6.0
pandas	^2.0.0
flask	^2.2.3
sphinx	^4.4.0
sphinx-rtd-theme	^1.0.0
sphinxcontrib-bibtex	^2.4.1

List of succulent's development dependencies:

Package	Version
pytest	^6.2

## 3.3 Testing

Before making a pull request, provide tests for added features or bug fixes.

Any failed test cases should be resolved before the changes are merged into the main branch.

To check if all tests pass locally, run the following command:

```
$ poetry run pytest
```

## 3.4 Documentation

To locally generate and preview documentation, run the following commands in the root directory of the project:

```
poetry install --extras docs
poetry run sphinx-build ./docs ./docs/_build
```

If the documentation builds successfully, it can be previewed in the docs/\_build folder by opening the index.html file.

## 3.5 API Reference

### 3.5.1 API

**class** succulent.api.SucculentAPI(*host, port, config, format='csv'*)

Bases: object

Succulent API server.

**Args:**

host (str): The host address to run the API server. port (int): The port number to run the API server. config (str): Path to the configuration file. format (str, optional): The format of the data ('csv', 'json', 'sqlite', or 'image').

**host**

The host address to run the API server.

**Type**

str

**port**

The port number to run the API server.

**Type**

int

**config**

Configuration file.

**Type**

dict

**format**

The format of the data ('csv', 'json', 'sqlite', or 'image').

**Type**

str

**app**

Flask application.

**Type**

Flask

**processing**

Instance of the Processing class.

**Type**

*Processing*

**measure()**

Process the incoming request and store the timestamp.

**Returns**

JSON response with a success message and timestamp.

**Return type**

Response

**start()**

Start the Flask application on the specified host and port.

**Returns**

None

**url()**

Generate URL with parameters for measurements.

**Returns**

JSON response with the generated URL.

**Return type**

Response

### 3.5.2 Configuration

**class** succulent.configuration.**Configuration**(*path*)

Bases: object

Class for loading configuration settings from a YAML file.

**Parameters**

**path** (*str*) – The path to the YAML configuration file.

**path**

The path to the YAML configuration file.

**Type**

str

**load\_config()**

Load the configuration settings from the YAML file.

**Returns**

The loaded configuration settings as a dictionary.

**Return type**

dict

### 3.5.3 Processing

**class** succulent.processing.**Processing**(*format, config, unittest=False*)

Bases: object

Class for processing and storing data.

**Parameters**

- **format** (*str*) – The format of the data ('csv', 'json', 'sqlite', or 'image').
- **config** (*dict*) – The configuration settings for data processing.
- **unittest** (*bool, optional*) – If True, the class is being used for unit testing. Default is False.

**directory**

The path to storage location.

**Type**

str

**format**

The format of the data ('csv', 'json', 'sqlite', or 'image').

**Type**

str

**columns**

List of feature names in the data.

**Type**

list

**boundaries**

List of minimum and maximum boundaries for each feature in the data.

**Type**

list

**df**

The DataFrame to hold the data (for 'csv', 'json', and 'sqlite' formats).

**Type**

pandas.DataFrame

**key**

The key to retrieve the image file from the request (for 'image' format).

**Type**

str

**Raises**

**ValueError** – Invalid format is provided.

**boundary**(*value*, *boundary*, *column*)

Check if the value is within the specified boundary for a column.

**Parameters**

- **value** (*float*) – The value to check.
- **boundary** (*dict*) – A dictionary containing ‘min’ and/or ‘max’ boundaries for the column.
- **column** (*str*) – The name of the column being checked.

**Raises**

**ValueError** – Value is outside the specified boundary.

**Returns**

The original value (within the boundary).

**Return type**

float

**parameters**()

Generate URL parameters based on data columns.

**Returns**

URL parameters as a string.

**Return type**

str

**process**(*req*)

Process the incoming request and store the data accordingly.

**Parameters**

**req** (*Request*) – The incoming request object.

**Returns**

None

## 3.6 License

MIT License

Copyright (c) 2023 Iztok Fister Jr.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

(continues on next page)

(continued from previous page)

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 3.7 Contributor Covenant Code of Conduct

### 3.7.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

### 3.7.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

### 3.7.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

### **3.7.4 Scope**

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

### **3.7.5 Enforcement**

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

### **3.7.6 Attribution**

This Code of Conduct is adapted from the [Contributor Covenant](http://contributor-covenant.org/version/1/4), version 1.4, available at <http://contributor-covenant.org/version/1/4>.

Homepage: <http://contributor-covenant.org>

Version: <http://contributor-covenant.org/version/1/4>

## PYTHON MODULE INDEX

### S

`succulent`, ??  
`succulent.api`, 9  
`succulent.configuration`, 10  
`succulent.processing`, 11



## A

`app` (*succulent.api.SucculentAPI attribute*), 10

## B

`boundaries` (*succulent.processing.Processing attribute*), 11

`boundary()` (*succulent.processing.Processing method*), 12

## C

`columns` (*succulent.processing.Processing attribute*), 11

`config` (*succulent.api.SucculentAPI attribute*), 9

`Configuration` (*class in succulent.configuration*), 10

## D

`df` (*succulent.processing.Processing attribute*), 11

`directory` (*succulent.processing.Processing attribute*), 11

## F

`format` (*succulent.api.SucculentAPI attribute*), 9

`format` (*succulent.processing.Processing attribute*), 11

## H

`host` (*succulent.api.SucculentAPI attribute*), 9

## K

`key` (*succulent.processing.Processing attribute*), 11

## L

`load_config()` (*succulent.configuration.Configuration method*), 10

## M

`measure()` (*succulent.api.SucculentAPI method*), 10

`module`

`succulent`, 1

`succulent.api`, 9

`succulent.configuration`, 10

`succulent.processing`, 11

## P

`parameters()` (*succulent.processing.Processing method*), 12

`path` (*succulent.configuration.Configuration attribute*), 10

`port` (*succulent.api.SucculentAPI attribute*), 9

`process()` (*succulent.processing.Processing method*), 12

`Processing` (*class in succulent.processing*), 11

`processing` (*succulent.api.SucculentAPI attribute*), 10

## S

`start()` (*succulent.api.SucculentAPI method*), 10

`succulent`

`module`, 1

`succulent.api`

`module`, 9

`succulent.configuration`

`module`, 10

`succulent.processing`

`module`, 11

`SucculentAPI` (*class in succulent.api*), 9

## U

`url()` (*succulent.api.SucculentAPI method*), 10