

# ISAC-SIMO-LITE

## Getting Started Guide

### INTRODUCTION

Intelligent Supervision Assistant for Construction - Sistema Inteligente de Monitoreo de Obra

ISAC-SIMO is a system to validate that the intervention work done for homeowners has been done correctly and safely. It is a Build Change project supported by a grant from IBM.

**ISAC-SIMO-Lite** is a Developer Test Tool that provides a convenient developer test environment to work with pipeline models. Using this Web Application, users can add different models, watson classifiers, processors in a single page dashboard and quickly test images. It shows all the results of each model of the pipeline after the test is successful and any possible errors.

### BEFORE YOU START

Before starting the project, you need to set these requirements.

- Python > 3.7.x

### INSTALLATION

View the Open-Source GitHub repository for [ISAC-SIMO Dashboard Lite](https://github.com/Call-for-Code/ISAC-SIMO-Dashboard-Lite).

First Clone this Project in a suitable directory & Change to project directory.

```
git clone https://github.com/Call-for-Code/ISAC-SIMO-Dashboard-Lite.git
cd ISAC-SIMO-Dashboard-Lite
```

Then, you will need to set up a virtual environment. The easiest way to setup is using **pipenv**. First, install pipenv using the following command.

```
pip3 install --upgrade setuptools
pip3 install pipenv
```

To activate the virtual environment shell you need to run these commands, followed by installing required packages. (Note that every time you run the application server, you need to be inside virtual env shell)

```
pipenv shell
pipenv install --skip-lock
```

Now, start the Application with:

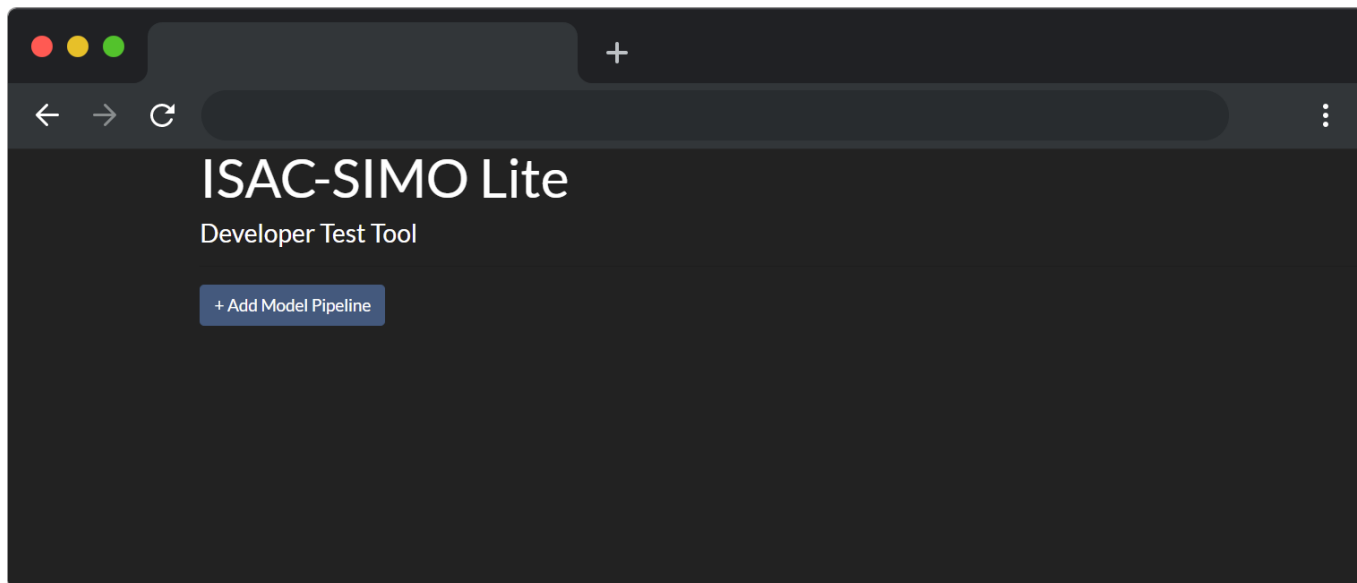
```
python manage.py runserver --noreload
```

And visit <http://127.0.0.1:8000/> in any modern browsers to open the application.

- `/static/` folder should have proper read and write access to current users or groups.
- During setup if package installation caused any error, please install them outside of pipenv and try running the server again.
- And, also note that most of the app controller are kept in `/main/views.py`

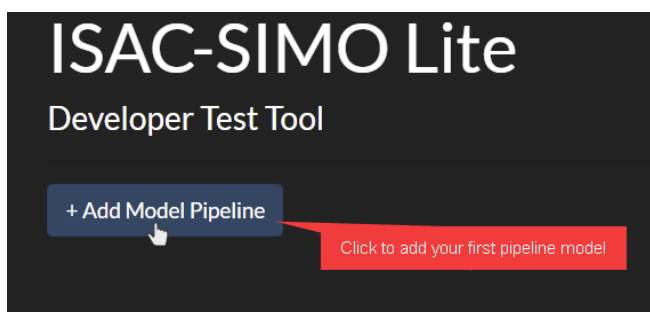
## GETTING STARTED

The application when loaded should look like this.

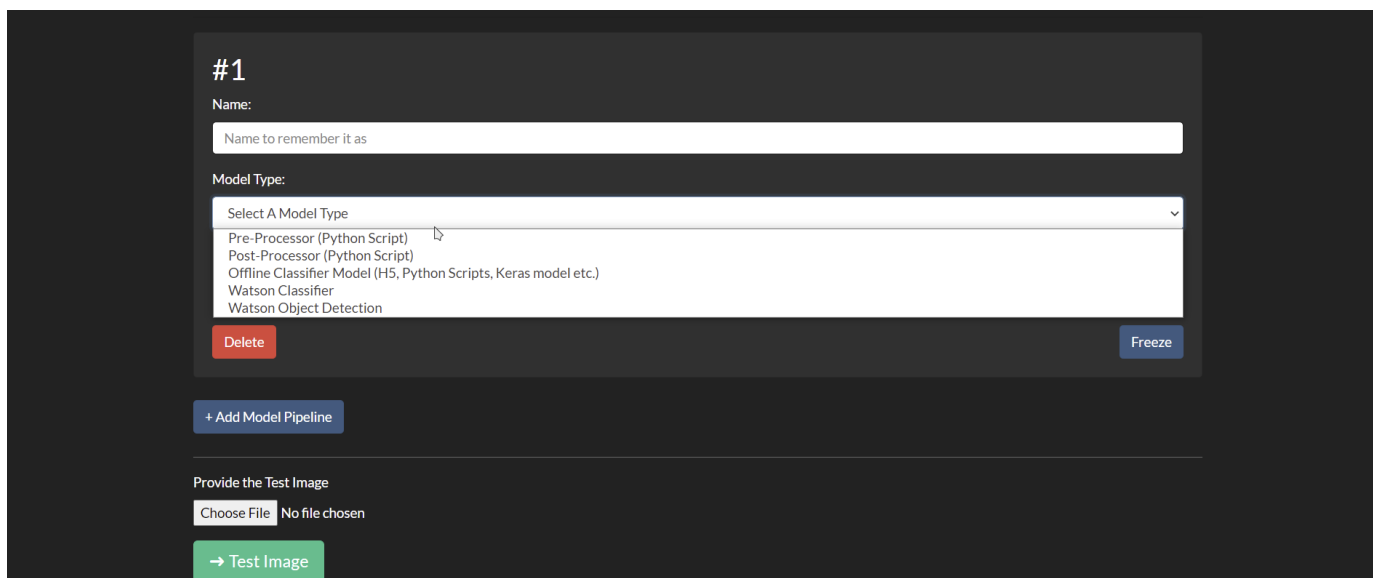


## ADDING MODEL PIPELINE

You can add new models to the pipeline by clicking on the “+ **Add Model Pipeline**” button. After clicking on the button, it will add a new card where you can choose the type of model, provide it a name, and fill other fields as required.



Provide an appropriate name to the model and choose the type of model as shown below.



When you choose the model type, it will add new fields as required by that specific model. For example, a pre-processor needs a python script file, Watson classifier needs IBM API Keys, Collection ID etc. Fill the form as required.

The screenshot shows a dark-themed form for configuring a model in a pipeline. At the top, it is labeled "#1". Below this, there are several input fields and buttons:

- Name:** A text input field containing "rebar-classifier".
- Model Type:** A dropdown menu showing "Offline Classifier Model (H5, Python Scripts, Keras model etc.)".
- Model File:** A section containing a "Choose File" button and the text "classifier1.py".
- Comma Separated Label:** A text input field containing "go,nogo".
- At the bottom, there are two buttons: a red "Delete" button on the left and a blue "Freeze" button on the right.

You can remove any model from the pipeline by clicking on the **“Delete”** button. The **“Freeze”** button is useful to disable the card to prevent accidental clicks and changes.

More models can be added to the pipeline by clicking **“+ Add Model Pipeline”** as before and filling the forms.

## PROVIDING THE TEST IMAGE

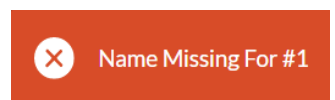
After you have added all the pipeline models properly, you can provide a test image which is used by the models and classifiers. You can choose the file using the field called **“Provide the Test Image”**.

The screenshot shows a dark-themed section titled "Provide the Test Image". Below the title, there is a "Choose File" button and the text "No file chosen".

Choose an appropriate image file for testing.

## TESTING THE IMAGE

Now, the **“→ Test Image”** button can be clicked to test the image using the provided pipeline of models. Any validation errors will be displayed properly with error messages and which model caused it.



**Note:** When you have already chosen a file or image in the web application and later modify it (locally) and try to run the test, it might not pass in chromium browsers (due to permissions issue). The application in that case will show **“File has been modified”** error and you need to choose the file again using the picker.

If all validation passes, it will start the testing process and on success show the results for each model.

Click on the “**View Test Result** ▾” button to toggle its result. It might contain an error message in red font color if the model did not run successfully. Else, it will show the result properly as required.

The pre-processor will show the processed image. The post-processor and classifier will show its JSON response. Similarly, Watson classifier and object detection will show its own response along with best result and score.

#1

Name:

Model Type:  


Pre-Processor (Python Script) ▾

Model File  

Choose File pre1.py

Delete Freeze

View Test Result ▸



Pre-Processed Image

(This Image will be used by all models below.)

*This image shows the test result of the Pre-Processor*

View Test Result ▸

```
[
  {
    "class": "ivory color",
    "score": 0.957
  },
  {
    "class": "triangle",
    "score": 0.896,
    "type_hierarchy": "/musical instrument/percussion instrument/triangle"
  }
]
```

Result: ivory color

Score: 0.957

Watson Classifier Result

*This image shows the test result of the Watson Classifier*

Results of the whole pipeline can be seen on the bottom of the page under the title of “**Pipeline Result**”. You can always modify the model, change the file or image and run the test again without reloading the page. *Any unhandled errors might be found in Django Logs.*

## Other Documentations and Guides:

- 1) [ISAC-SIMO Documentation \(Latest\)](#)
- 2) [ISAC-SIMO Backend Developer Guide](#)
- 3) [ISAC-SIMO Documentation & Guide](#)
- 4) [Github Repository](#)