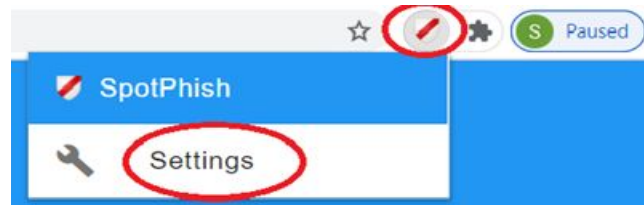


# Modular Spotphish

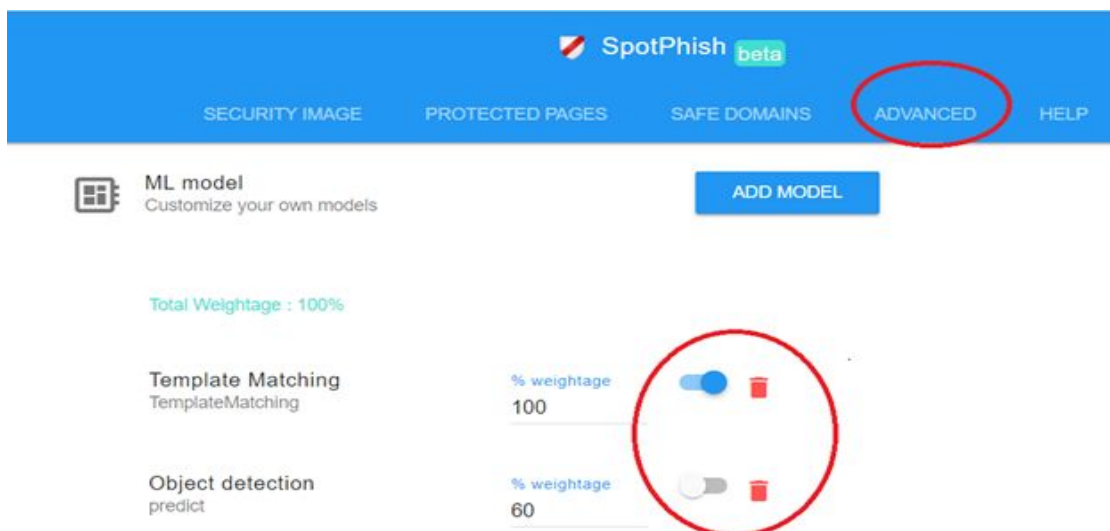
Spotphish has been made modular, with this release ML models can be picked and chosen on the fly. Models can be disabled, enabled and new models can be added online.

## 1. Enable/disable model

- Go to Settings

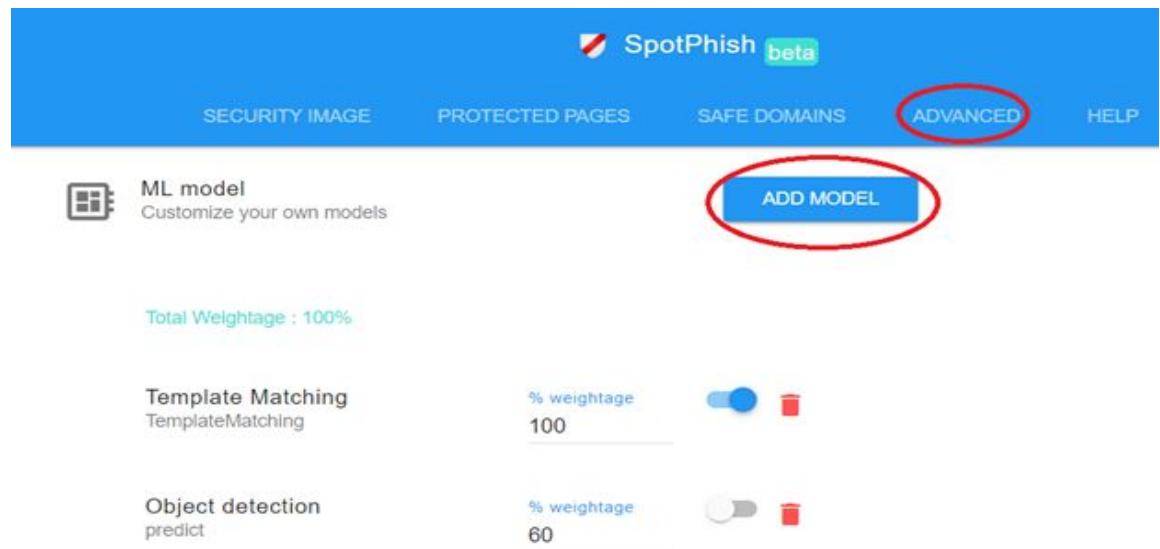


- Go to ADVANCED

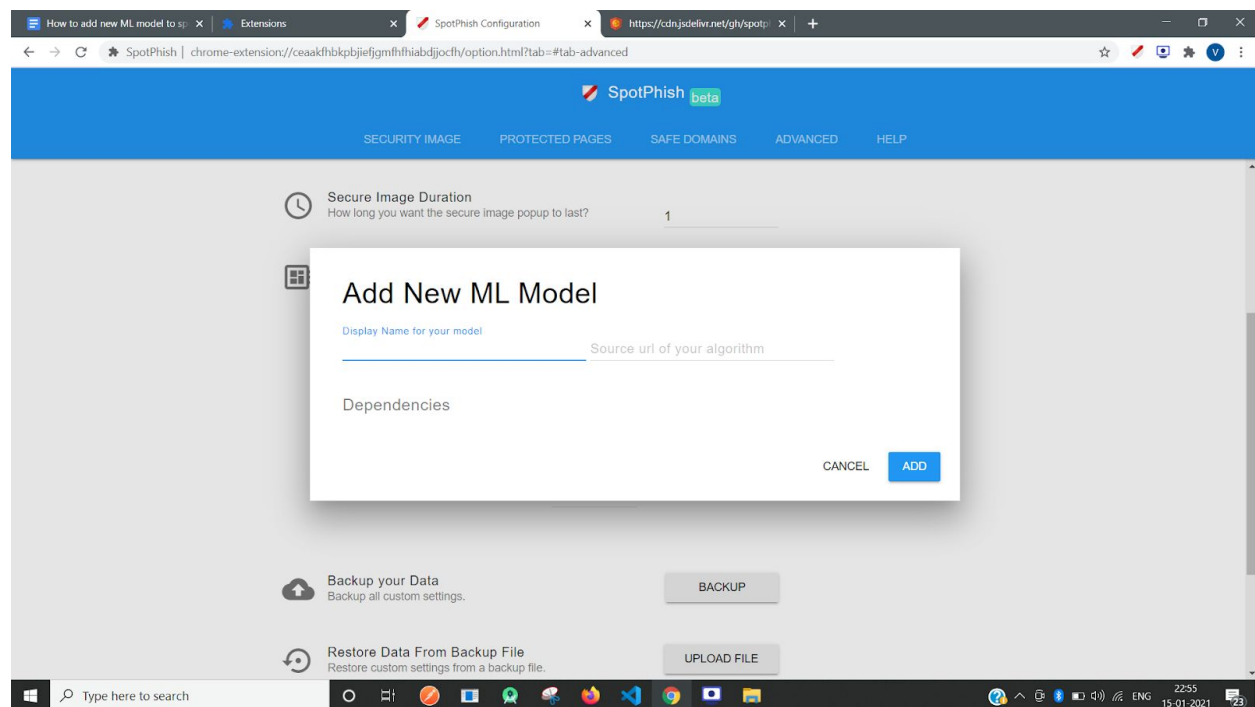


## 2. Add new model

- Click ADD MODEL button



- Popup for add model



**Fields:**

- Display name for your model. **(Required)**
- Source Url of the algorithm. **(Required)**
  - It is the github url of the directory in which your algorithm is kept.
  - Name of the algorithm file must be **Model.js**. Your directory must contain the file with the same name as mentioned above.
  - If you maintain versioning, the latest version of the files will be picked. Otherwise, files from the default branch will be picked.
  - Structure of your Model.js file:

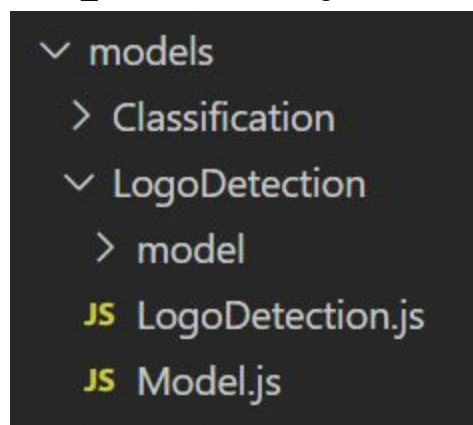
Import ...

```
export default class class_name{
  predict(img){
    return{
      site: <name of predicted class> (string),
      confidence: <confidence %> (int),
      time_taken: <time taken to predict> (int),
      image: <base64 url of predicted image>
    }
  }
}
```

class\_name.dependencies=[ ]

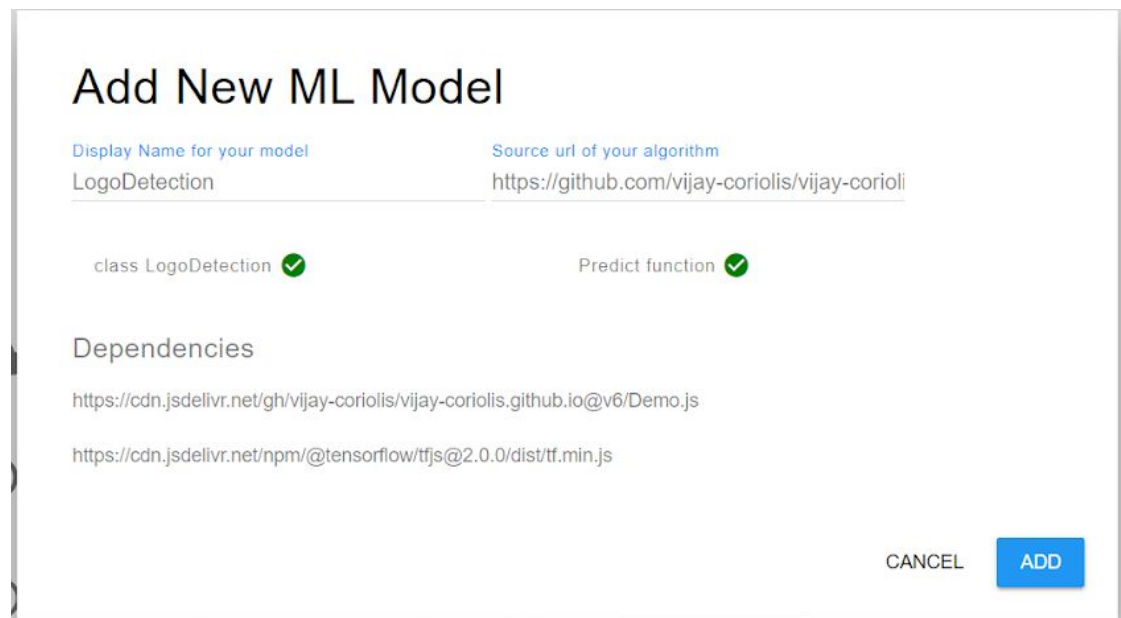
**Note: words highlighted in red should be kept as it is. (case sensitive)**

- All local paths in your algorithm must be relative to the root directory. Use the constant **ROOT\_DIR** to point to the root directory.
- Eg. If your project has a models folder in the root directory. Then to access LogoDetection.js in Model.js , Use the path *ROOT\_DIR+"/models/LogoDetection/LogoDetection.js"*



- You will see the class name which is exported from your algorithm
- Dependencies will be listed down(if any)

**Note : A video is attached below the popup for your reference.**



The image shows a 'Add New ML Model' popup form. It has two input fields at the top: 'Display Name for your model' with the value 'LogoDetection' and 'Source url of your algorithm' with the value 'https://github.com/vijay-coriolis/vijay-corioli'. Below these are two status indicators: 'class LogoDetection' with a green checkmark and 'Predict function' with a green checkmark. Under the heading 'Dependencies', there are two URLs: 'https://cdn.jsdelivr.net/gh/vijay-coriolis/vijay-coriolis.github.io@v6/Demo.js' and 'https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js'. At the bottom right, there are two buttons: 'CANCEL' and 'ADD'.

- Click Add, ML model is added in the list

### 3. Assign weightage to models

- Total model weights should be 100%
- Distribute total weights (100%) to the enabled models

### 4. Train models

- Train models - it can be template matching, classification, object detection or any other model. Models can be trained using any algorithm and any tool.
- Trained models should accompany a javascript file. Here is the structure of the JS file.
  - a. A Class should be exported
  - b. Class must have a predict function.  
**Input** : src of img for which model will predict  
**Output**: json format output

```

{
    site: <name of predicted class> (string),
    confidence: <confidence %> (int),
    time_taken: <time taken to predict> (int),
    image: <base64 url of predicted image>
};

```

- c. Import your dependent js files, libraries, modules in this file.
- d. If those dependent files, libraries, modules can not be imported and need to be injected then, specify them into a **Class level variable** named as **dependencies**
- e. Structure of your file:

```

Import ...
export default class class_name{
    predict(img){
        return{
            site: <name of predicted class> (string),
            confidence: <confidence %> (int),
            time_taken: <time taken to predict> (int),
            image: <base64 url of predicted image>
        }
    }
}
class_name.dependencies=[ ]

```

**Note: words highlighted in red should be kept as it is. (case sensitive)**

- f. **Don't forget to convert all your local paths relative to the root directory using ROOT\_DIR constant.**
- g. **If your dependent library is not delivered through cdn then download it, push it to your own github repo and specify in the dependencies array as ROOT\_DIR+<path to library> .**