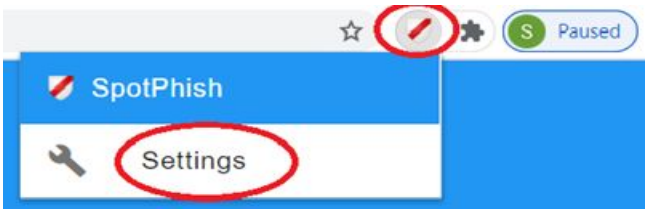
Modular Spotphish

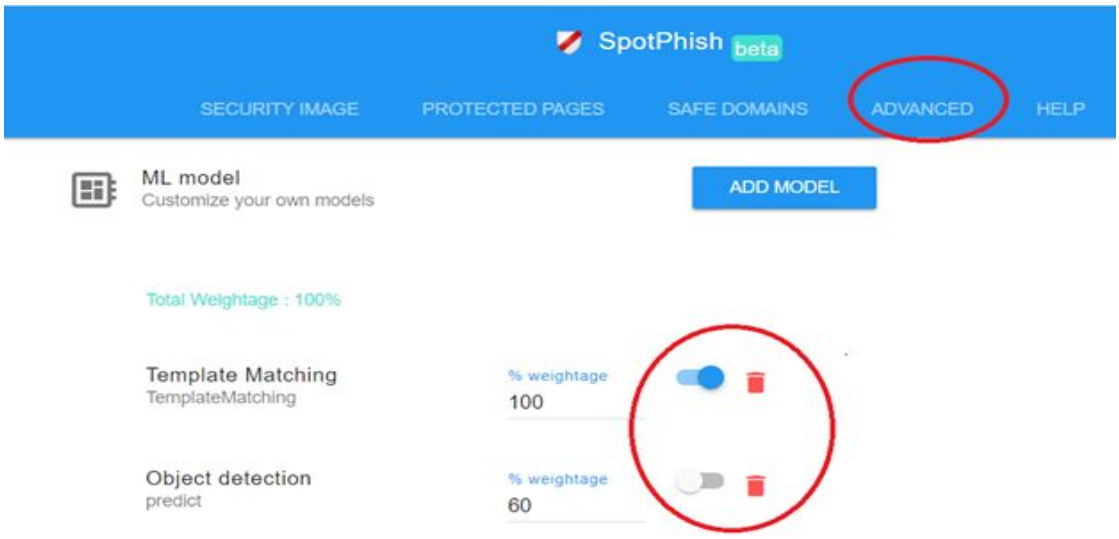
Spotphish has been made modular, with this release ML models can be picked and chosen on the fly. Models can be disabled, enabled and new models can be added online.

1. Enable/disable model

- Go to Settings

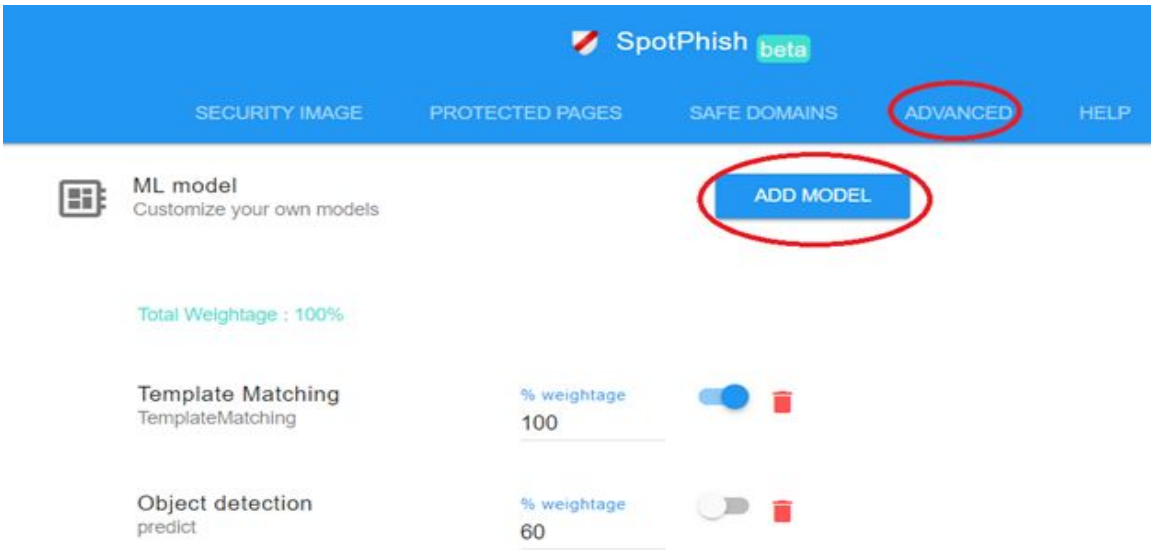


- Go to ADVANCED

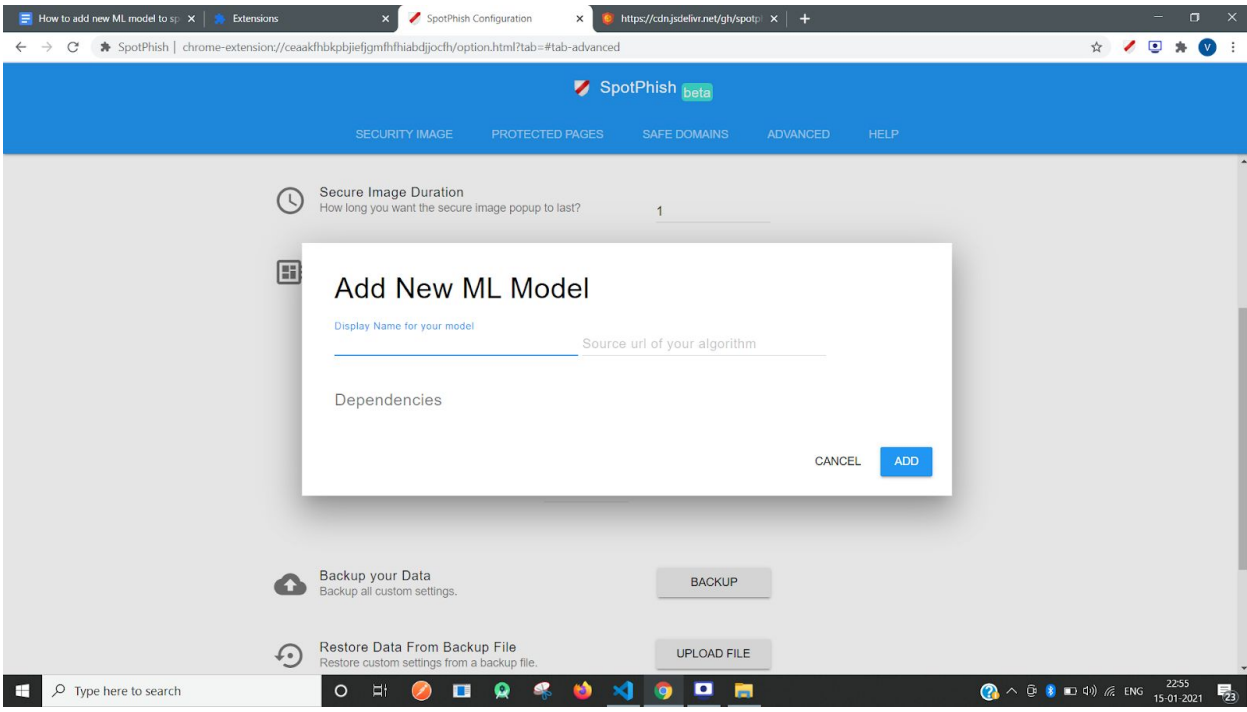


2. Add new model

- Click ADD MODEL button



- Popup for add model



Fields:

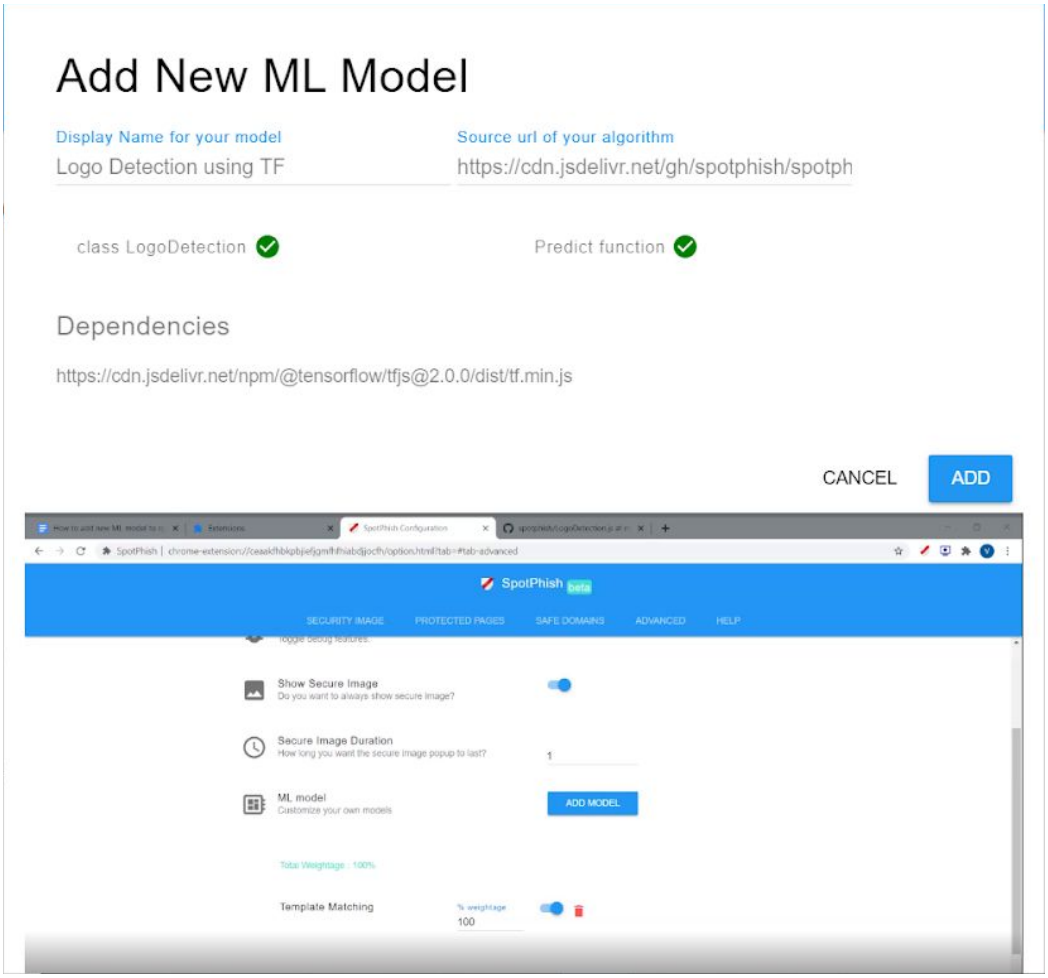
- ➢ Display name for your model.**(Required)**
- ➢ Source Url of the algorithm where it is hosted. **(Required)**
    - **Note:** For now, only **cdn.jsdelivr.net** url are allowed. If your algorithm is hosted on github, then follow these steps to get CDN url.
    - Structure of your algorithm:

```
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)

- 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.



- 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 references to any file into CDN urls.**

5. Advance users

- Go to **examples/LogoDetection** folder of https://github.com/spotphish/spotphish
  - LogoDetection.js is the main file which contains the algorithm of detecting a logo.
  - It depends on tensorflow library

- Converting github url to CDN url

  https://github.com/spotphish/spotphish/blob/master/examples/LogoDetection/LogoDetection.js

  Converted to

https://cdn.jsdelivr.net/gh/spotphish/spotphish/examples/LogoDetection/LogoDetection.js

- Inside LogoDetection.js

```
export default class LogoDetection {
        async predict(src) {
                img.src=src
                loadMyModel(https://cdn.jsdelivr.net/gh/spotphish/spotphish/examples/LogoDetection/model/model.json)
                runMyModel(img)
                return {
                        site: 'XYZ',
                        confidence: 0,
                        time_taken: 0,
                        image: base64 url of predicted image
                }
        }
}
LogoDetection.dependencies =[
   "https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js",
]
```

- If your dependent library is not delivered through cdn then download it,push it to your own github repo and get cdn url following the below steps.

6. Helper tips

- Getting CDN url from github raw file
    - Get the link on github and click on “Raw” version.
    - Change raw.githubusercontent.com to cdn.jsdelivr.net
    - Insert /gh/ before <username>
    - Remove the branch name.
    - (Optional) Insert the version you want to link to, as @version (if you do not do this, you will get the latest - which may cause long-term caching)

Examples:

- http://raw.githubusercontent.com/<username>/<repo>/<branch>/path/to/file.js
    - Latest version http://cdn.jsdelivr.net/gh/<username>/<repo>/path/to/file.js

- Specific version or commit hash
  http://cdn.jsdelivr.net/gh/<username>/<repo>@<version or hash>/path/to/file.js