# Modules

* Normally allow for multiple instances which by default are **inactive**.
  + Add a Singleton module subtype that can only have a single instance.
    - Add a subtype to that for background modules that the user can’t see which can be utilized by other modules if necessary
* Users can add a module from the list of available modules. This creates an instance.
* Each module has its own settings.
  + Consider support for “static settings” which are shared across all instances of the same module.
  + Need to have a way in the configuration GUI of setting values for the same fields of multiple instances at the same time, so as to save time for the user from setting the same thing over and over again.
* Instances should be possible to duplicate (this might just be a GUI thing though).
* Instances should have settable names, to differentiate between them.
* Internal filters

# Events

Many of the things modules can do are implemented as events in the system. One could avoid using the module configurations and make them as custom events instead, but in order to keep the configuration UI simple for simple tasks, modules will also provide configuration options that create events behind the scenes. These will be kept separate from custom events in order to easily maintain a link between the module configuration GUI and the events (otherwise, for example, if the event “on first message by X say Y” – created by adding Y as a welcome message for X in the Welcome module – is treated the same as custom events, then if the user switches to the event editor and changes that event to be “on X using the command “!hello” instead, what would the entry for X in the Welcome module configuration page show?).

* **If**:
  + General filter category
    - Same as before
  + Module category
    - Filters exported by each module instance
* **On**:

Triggers for events. Can have parameters such as triggering for only specific users.

Create parameter filters that can be applied to multiple triggers – e.g. number filters that let you choose if a number needs to be = / ≠ / < / > from a given value, and a user filter which lets you choose if it is/isn’t a specific user or is/isn’t one of several.

Include string length limit for Twitch chat messages.

* + General trigger category
    - All the ones from before
  + Connections category
    - Twitch
      * All the ones from before
      * Cheer [user] [amount]
    - Discord
      * Message by user to server channel, etc.
  + Module category
    - Triggers exported by each module instance.
* **Where**:
  + Trigger-specific filters (e.g. username / mod status for command triggers).
* **Do**:
  + System category
    - Enable/disable modules.
    - Exit bot.
    - Reload configuration from disk.
    - Set global volume level?
  + Connections category
    - Consider making the contents of connection categories top-level, so that, for example, it looks like “Do: Twitch 🡪 Send Message” rather than “Do: Connections 🡪 Twitch 🡪 Send Message.”
  + Module category
  + Overlay category
    - Send commands directly to overlays (e.g. “Show Media”).

# Assets

Assets are saved system-wise rather than for specific modules. Creating a module that uses assets also creates a folder for that module in the assets directory, but that’s as far as it goes with automatic changes by modules. Users can manage the file system as they would any regular file system.

Everything that uses assets should be capable of handling not having them. Even if an asset is assigned, it could be deleted directly from the assets directory, in which case that assignment would be unmade.

Consider adding an enable/disable checkbox per file for fields that use multiple file (e.g. letting you disable existing Welcome images temporarily to show a new one and then you can enable them again after you’ve shown it for the first time on stream).

# Configuration GUI

No apply/revert option this time.

## Sections

### Main Settings

Same as the old bot, minus the channel rewards part. Instead, when choosing a trigger for an event, one of the options will be a channel reward, and there it’ll show the list of rewards available on the channel, with a “refresh” option for reacquiring that list (through the Twitch API).

### Modules

* Module selection: add/remove modules.
* List of present modules, configuration page for each one.

#### Module Configuration

* Fields:
  + Name
  + Enabled
* Messages (strings used by the module).
* Actions and events.

### Event Editor

* For now, just an editable list with all the custom events (should not show built-in module events generated by module configuration pages).

### Assets

* File explorer that shows all the assets in the system with previews and allows for manipulating them (upload, delete, move).
* Some module configuration pages require selecting assets – these will show the asset selector, which is basically just the file explorer with an added option of “OK/Cancel” buttons to perform the selection.
  + Need to support both single selection and multiple selection.
* When selecting an asset for a module, by default its folder will be opened in the asset selector.

## Features

* “Duplicate module” option.
* “Copy settings” option that can let you copy the settings of one module to another.
* Multi-select that lets you set the value of a setting on multiple things at once

# Overlays

* Overlay editor similar to the SE one.
* Widgets taken from modules: add widget 🡪 module selection 🡪 widget selection.
  + Also possibly some general things, such as static images or whatever (not really what this bot is meant for, but some things might fit here).
* Expose widgets which can accept contents (e.g. image or text display) that can be used in events.
  + Keep references and update the information in the events when the widgets or overlays change (when their names change, for example).

## Widgets

These are mini-overlays to be used directly in OBS rather than in the overlay editor. They are provided by modules and each module that has them should provide a link that can be used in OBS (it should also allow opening it in the browser for testing things out).

Consider having a widget editor which is like the overlay editor except the widgets it has available include the internal elements of the widget being edited.

# Connections

Put all connections to the world outside the bot here.  
This will initially include:

* Twitch
  + Using the full Twitch API Node.js library.
* StreamElements

Consider adding:

* Discord
* Ko-fi
* General external API with URL

It should be possible to add a connection plug-in to support additional external connections.  
Each connection plug-in should provide a means of instantiating a connection that takes the necessary parameters (e.g. a channel name for Twitch for a single connection, or multiple names to monitor multiple streams at once, along with the credentials needed), and provide an “on” function for registering for events for an active connection.

# Strings

A generalization of the previous “responses” feature, to be used anywhere in the configuration where text can be entered.

Instead of variables like before, there should instead be contexts that can be attached to a situation which defines which variables are usable there.

Each string should specify which contexts it uses, and some contexts should be attached automatically (e.g. trigger-specific contexts, such as a Twitch command context which provides access to e.g. the command cost – this context should include the Twitch message context which provides access to e.g. the sending user).