Guide for extending the block-based system in “Plugging It In”

The research paper for the implementation of the block-based system in “Plugging It In” is in:

- **project-reports/seng402_final_lugtu-rchi.pdf**

This paper details all the major design decisions and work completed throughout the project, such as how each feature was implemented and the process of how the system was evaluated to get to where is it now.

Extending/Modifying the custom blocks

To extend/modify the custom blocks used in the system, open the Blockly Developer Tools and import the custom blocks library XML file I added in the CS Unplugged repository which is located in:

- **csunplugged/plugging_it_in/custom-block-based-blocks.xml**

Once you have imported the XML file into the Blockly Developer Tools, there you will see how each custom blocks were created as shown in Figure 1. From there, you can modify/customize the blocks to however you like.

The block definitions and code generators for the custom blocks are defined in:

- **csunplugged/static/js/custom-blockly-blocks.js**

Once you have made a modification to the custom blocks, you can copy and paste the block definition and/or code generator for that block to the *custom-blockly-blocks.js* file.

Note:

- You will see the colour of the blocks are black and the shapes of the blocks do not look like Scratch blocks. This is because the Zelos renderer and a style is applied to the blocks.
- There is a style attribute in each block definition which is responsible for adding colours to the custom blocks. Ensure that when adding or modifying an existing custom block, that the style attribute is there, otherwise the custom blocks will not have any colour applied to it.
- Figure 2 shows one of the styles added to the Say block. The style, which defines the colours of the blocks, are in the *jobe-editor.js* file located in:
    - **csunplugged/static/js/jobe-editor.js**

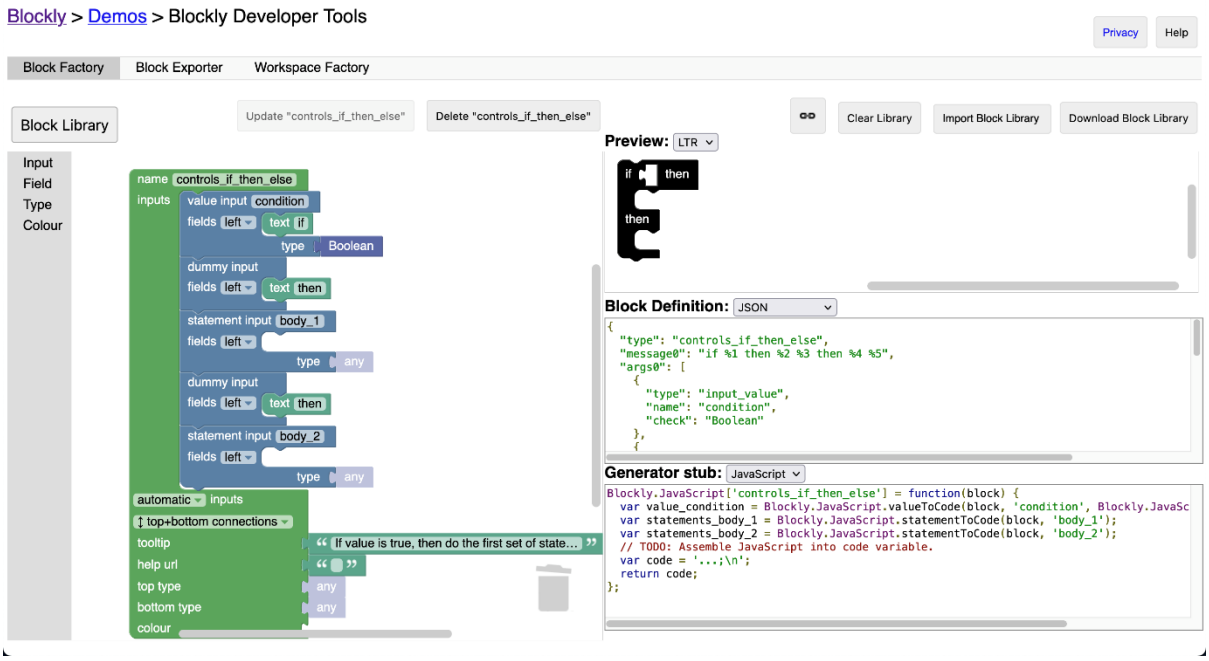

Figure 1: Blockly Developer Tools with the library XML file imported.

```
30      // Looks say block
31      {
32      "type": "looks_say",
33      "message0": "say %1 %2",
34      "args0": [
35          {
36          "type": "input_dummy"
37          },
38          {
39          "type": "input_value",
40          "name": "value"
41          }
42      ],
43      "inputsInline": true,
44      "previousStatement": null,
45      "nextStatement": null,
46      "style":  "looks_blocks",
47      "tooltip": "Say the specified text, number or other value.",
48      "helpUrl": ""
49      },
```

Figure 2: Shows the style applied to the block definition of the Looks Say block.