System/Subsystem Design Description Document for Logic Gate Simulator

November 20th, 2018

Aby Prasad, Jenna Westfall, Bright Sun, Robert Rose, Yael Weiss

Table of Contents:

1. Scope

1.1 Identification

This document describes the purpose and design of Logic Gate Editor, a browser-based circuit simulation application, Version Number 1.01.

1.2 System overview

Logic Gate Simulator will be a web version of the logic gate learning tool Logisim. Like Logisim, the app will simulate basic circuitry and logic gates. The end goal is to make a simulation environment that will be easily accessible and user-friendly to students. A top priority will be to improve upon Logisim's design by making a logic gate simulator that will be more user-friendly and more accessible (easy to access without download/setup, and be able to share work with a web link).

1.3 Document overview

This document is used to establish concrete design decision for the product.There are no known security or privacy risks known from viewing this document. Section 2 provides all referenced documents used in the following sections. Section 3 discuss design decisions that impact the entire systems. Section 4 goes into further detail on individual components and classes of the system. Section 5 describes the traceability of each requirement corresponding to their associated CSCI.

1.4 Definitions

Throughout this documents the following terms are used:

- **Logic Gates:** Refers to an elementary building block of a digital circuit. Logic gates have inputs and one output. The inputs and outputs are binary values that indicated by 1 and 0. Different types of logic gates output different results.
  Ex: AND gate, OR gate, NOT gate, XOR gate, NAND gate, NOR gate, XNOR

- **Circuits:** A collective of logic gates that use one and others as inputs. They produce a single binary output.

- **Diagram:** The entire display consisting all a logic gates and circuits constructed.

- **GUI:** Stands Graphical User Interface
- **Logisim:** A downloadable program used to simulate logic gates and circuits.
- **Wires:** Used to connect the output of one logic gate to the input of another gate thereby creating a circuit.
- **Simulate:** To have the circuits constructed display its output.
- **Save:** Stores the constructed circuit as a file
- **Load:** Retrieves a constructed circuit as a file
- **Share:** Saves the constructed circuit to a public database that allows it to accessed by URL.

2. Referenced documents.

The following documents were used in the development of this SSDD and are referenced in the proceeding sections.

2.1 Government Documents

MILSTD 498 SSDD Template: Provided template for which this document was based on. Can be accessed via the following URL: https://github.com/RobRoseKnows/logisim-browser .

2.2 Non-Government Documents

Software Requirements Specification (SRS): Document listing and explaining the agreed upon requirements by the client and the developers. Can be accessed via the following URL: https://github.com/RobRoseKnows/logisim-browser .

3. System-wide design decisions

3.1 Accepted Inputs and Outputs

The main forms of external input the program will receive will be through dragging and dropping gates and through buttons (share, save, load, clear, simulate, help, and wires). The share button will correspond to a pop-up GUI (GUI2 in section 4.3) that allows the user to enter an email address; this will be the only text-based user input accepted. The program will also output a URL with the share GUI.

3.2 Exception Handling

Users who place gates outside the accepted edit area will have the gates return to drawer panel and there will be no effect on the edit area. Exceptions errors resulting from undefined urls will be designed out of the system. Any faulty urls entered will be handled by the the web browser and a 404 page not found exception will be thrown by the browser. An error involving a JSON file not being able to load will will be handled by the system and an exception will be thrown by the program.

3.3 File/Database visibility

The user will not have direct access to the database. The application is web-based, so the user will only have access files upon saving or exporting their diagram. The user will be able to save in JSON file format, and load in JSON file format. The application will load from the database upon using a previously generated URL.

3.4 Safety, Security, and Privacy

There are no safety concerns with using the product, beyond those associated with normal use of a computer. In order to satisfy privacy constraints, we will not log any additional data beyond what is necessary to operate the service. Security requirements will be met by keeping the server and all libraries up-to-date on the latest security patches.

3.5 Main Form Design

The main interface is focused on having a minimalist design in order to meet the program's goal of being user-friendly. The interface will have large icons representing each gate, as well as a wire mode button. Clear text will be used to label each button on the toolbar. A help button will be easily accessible from the toolbar for a more detailed explanation of the program's features.

4. System architectural design.

4.1 System components.

The following are the main system functionalities that will make up the application:

CSCI1: Edit pane

A grid panel that allows drag and drop of components like gates and wires. Upon hitting the "simulate" button, the edit pane will simulate the connections. The foundation of this component will be supplemented by a framework. The functionality will be built from scratch.

CSCI2: Toolbar

The user will be able to access the certain functionalities such as save, load, and run simulation. This component will use a generic container type from a third-party framework.

CSCI3: Drawer panel

A section of the interface used to display simple gates (CSCI4) available to the user. The user will also be able to drag gate from this section to add them to the edit pane(CSCI1).

CSCI4: Gates

This CSCI will be in near-constant use, with the exception of when a user needs to share/save/export. The gates will be designed to be drag and drop. A user may select a gate from the left side of the screen, drag it over to the edit pane (CSCI1), and drop it where they need it. The gate will be able to be deleted and removed from the edit pane as well. This component will be a unique class built from scratch.

CSCI5: Wires

A user can select the wire button whenever they wish to add wires. When in wire draw mode, they will not be able to pick up gates. A user will be able to create a wire on the edit pane along a grid-system (CSCI1**)**. Where the user clicks and drags the cursor, a wire will appear. A wire will be able to attach to a gate component. A wire will change color according to its input when under simulation. This component will be built from scratch.

CSCI6: Share diagram

The share diagram function shall be activated by clicking a button labeled “share” near the top of the screen. A pop-up GUI will allow the user to enter an email address to send the diagram to. The program will store the diagram data into the database and generate a unique URL for another user to access the diagram. This component’s foundation and functionality will be based off of existing generic components, such as a window form and entry boxes.

CSCI7: Save diagram

The save diagram function will be activated when the user clicks a button labeled “save.” The program will open a pop-up GUI (file browser for their desktop) with which they can use to save a copy of their work in a JSON file.

CSCI8: Load File

The user will be able to open a file explorer GUI that will allow the user to navigate to a locally stored JSON file and upload it to the application. This functionality will be based on the existing windows file browser feature.

CSCI9: Help Manual

There will be a button on the toolbar labeled “Help Manual” that upon clicking it, will open a small window (pop-up GUI) explaining how to use the application. This CSCI utilize a generic window form from a third-party framework.

CSCI10: Main Form

The main driver of the system responsible for rendering all other CSCI components. The document will also handle input from the user and deal with exceptions.

Component Diagram:

Below is a component diagram illustrating the relationships amongst the stated components. Every component is attached to the main interface “MainForm.” Most of the program functionalities will take place within this form, the other interfaces (represented on the diagram as “PopUpGUI”) are supplementary.

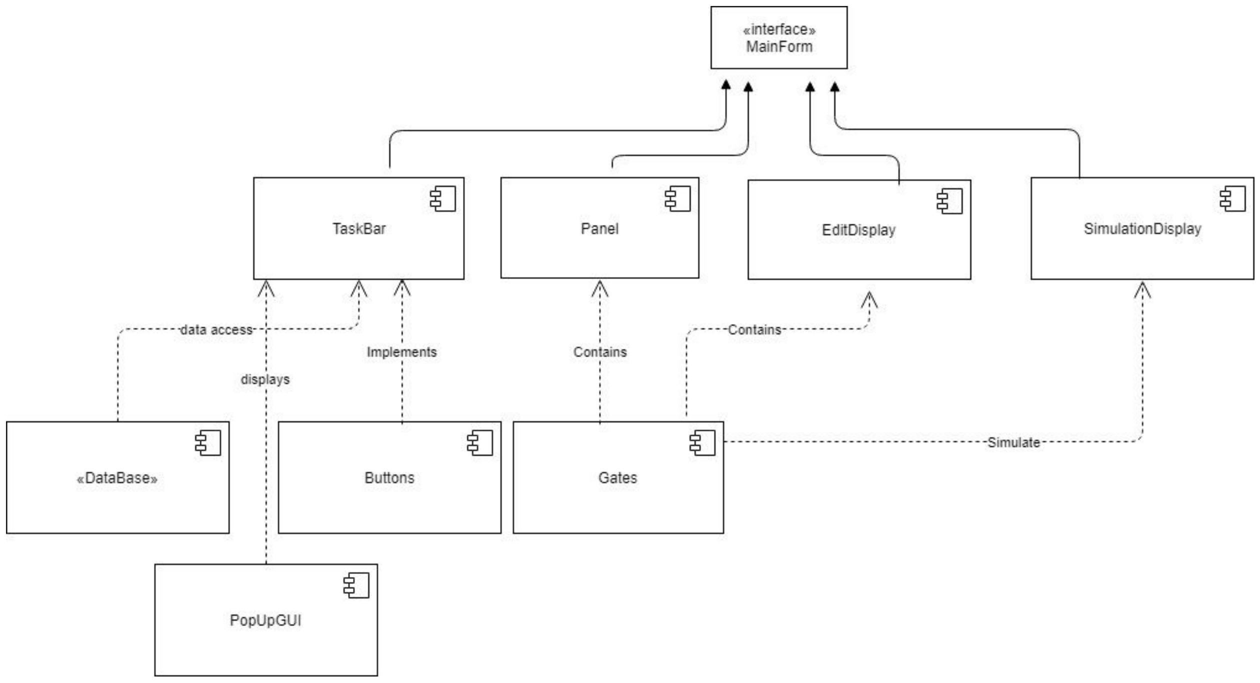

Diagram 1: Class Structure

UML Class Diagram:

Below is a diagram displaying information on the classes for this system. The driver class, MainForm, will hold and run instances of the other classes. The MainForm will display an instance of the TaskBar, Panel, EditDisplay, and SimulationDisplay classes. MainForm will have function LoadComponents() to load each class into the display. The TaskBar class will contain multiple instances of the button class in order to execute certain functionality. The Panel, EditDisplay and Simulation classes will implement the gate class each in a unique way. The Panel class will hold each of the gate classes as an option for the user to select and drag into the EditDisplay class. The SimulationDisplay class will then display the result of the circuit diagram.

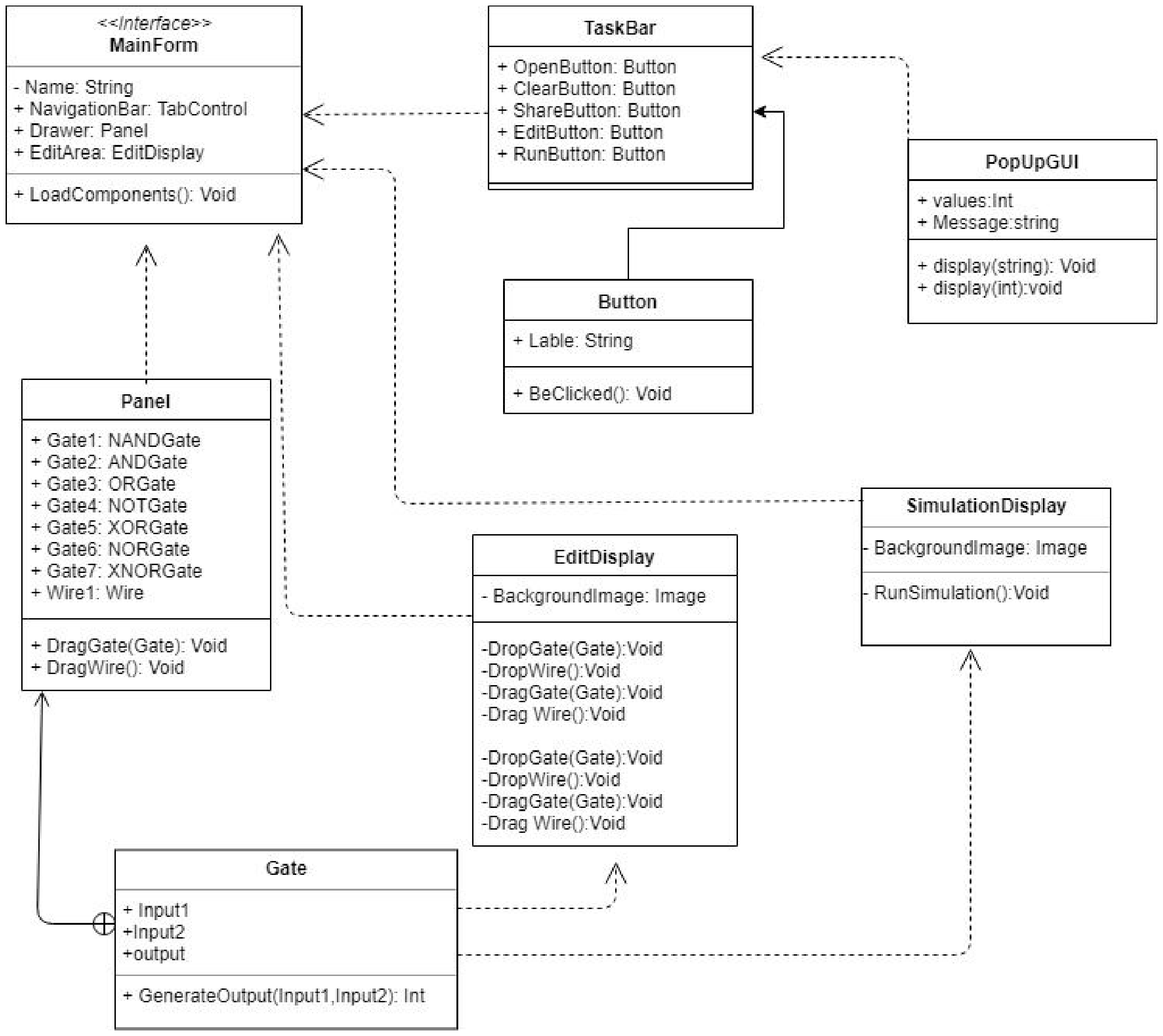

Diagram 2: UML Class Diagram

4.2 Concept of execution.

Below is a diagram that shows simple flow control between the different components of the interface and decisions that must be made regarding the interface.

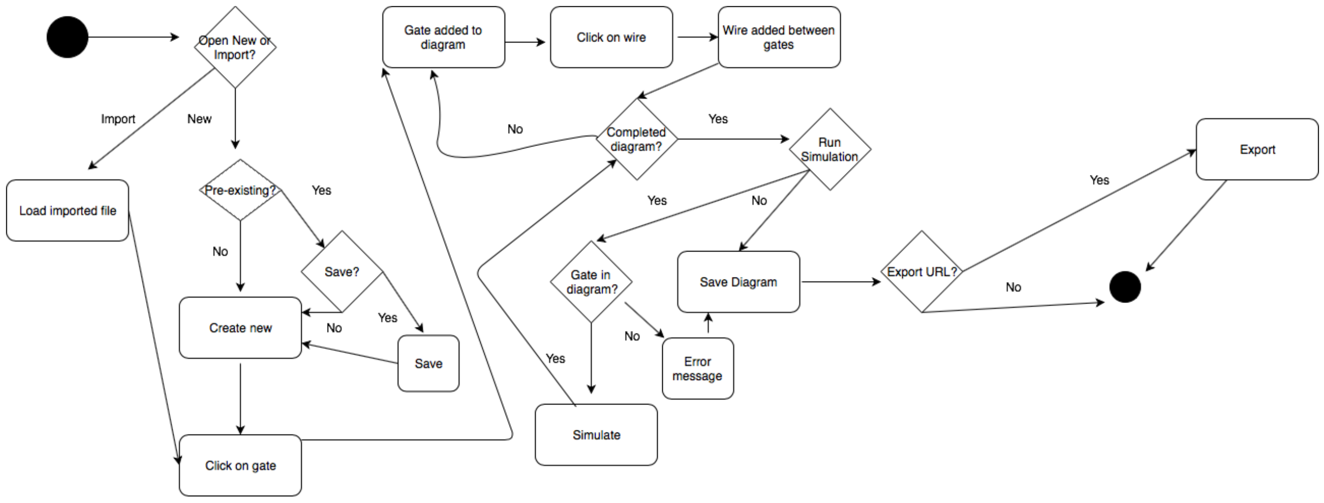

Diagram 3: Interface Design Diagram

4.3 Interface design.

The following sections are unique interfaces the program will utilize, excluding generic types of interfaces such as the file explorer type.

4.3.1 GUI1 - Edit interface

The edit interface will be the main form. It will allow a user to drag and drop gates and wires to create custom circuit diagrams. The user will select from pre-made gate classes that will be handled by the programs logic. The user will also be able to simulate these circuits. This is the main interface which will allow access to other interfaces implemented in the system, such as I2 and I3. The edit interface will also communicate with the database when loaded via custom URL. The user will not have access to the database directly through this interface.

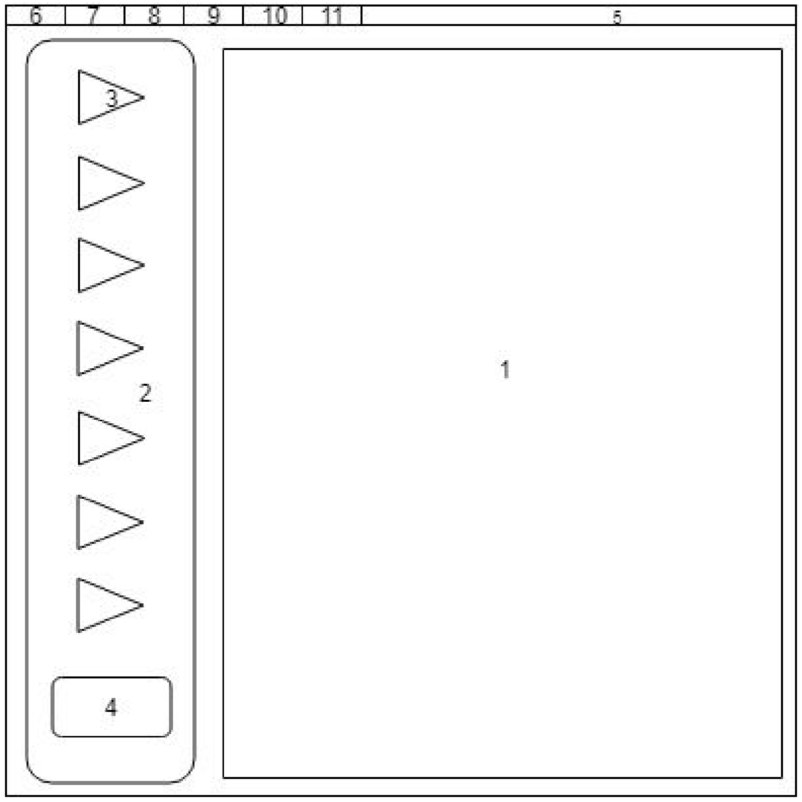

Figure I1: Main Interface

Figure I1 Key:

1. Edit panel: a container that will hold and keep track of gates and wires placed on it. Will also be where circuit simulation occurs.
2. Gate Panel: a container that holds gates.
3. Gate: every sideways triangle represents a gate. There are seven planned gates: AND, OR, NOT, XOR, NAND, NOR, XNOR.
4. Wire button: click this to swap from drag and drop mode to wire mode, which allows you to click anywhere on the edit panel and create a wire.
5. Toolbar: container to hold options, primarily the four buttons (save, share, clear, and help).
6. Save Button: saves work as JSON file.
7. Load Button: loads JSON file from local desktop.
8. Share Button: generates a unique URL to open the current work in another context.
9. Clear Diagram Button: clears the current diagram.
10. Simulation Mode/Edit Mode Button: swaps the edit pane from the simulation mode to the edit mode, or vice versa, depending on which mode the current state is set to.
11. Help Button: opens the program manual in a small window.

4.3.2 GUI2 - Share Diagram

Collects the data from the edit panel component and stores into the database. The program will then generate a unique URL with which to restore the data, providing it in the first entry box. If a user wishes to send the link via email, the interface will collect this information and send it to the specified recipient. This GUI is supplementary to the main form, GUI1. The email address added will have to be of standard email format (i.e: provide a domain and '@' symbol).

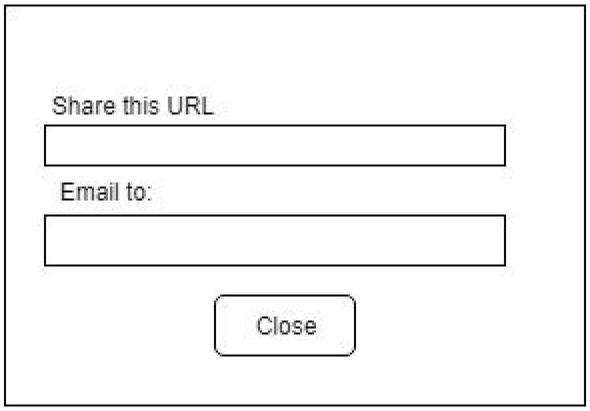

Figure I2: Share GUI

4.3.3 GUI3 - Clear Diagram

The clear diagram interface will be accessed through the edit interface. It will be mapped to a button on the tool and will display a pop up window that has priority over all other interfaces. It is designed to allow the user to clear the current edit panel. The user will be asked to confirm their decision and will be given two options, yes and no. If yes, interfaces with the edit panel component and wipes the data. If no then nothing is done. Both options return control back to the edit panel.

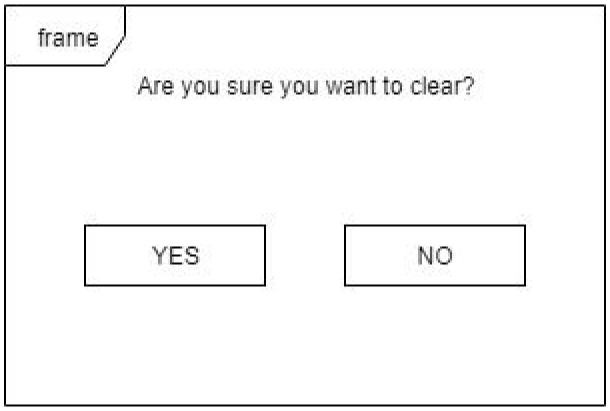

Figure I3: GUI Pop-Up Window

5. Requirements traceability.

5.1 CSCI Compatibility

Below is a table listing the internal interface requirements paired with a CSCI that handles them. Features are represented by their corresponding ID (i.e: CSCI1: Edit Pane) listed in section 4.1.

Requirement:	**Handled By:**
R1	CSCI4
R2	CSCI10
R3	CSCI6
R4	CSCI7
R5	CSCI8
R6	CSCI4
R7	CSCI7
R8	CSCI4
R9	CSCI10
R10	CSCI10
R11	CSCI6
R12	CSCI10
R13	CSCI10
R14	CSCI10
R15	CSCI10
R16	CSCI6, CSCI10
R17	CSCI10
R18	N/A
R19	CSCI10

6. Notes

6.1 Required States

State:	**UI:**	**Logic Component:**	**Data Component:**
Edit	Edit Interface	Java/Python UI Lib	Java/Python Object
View	Edit Interface	Java/Python UI Lib	Java/Python Object
Database (DB)	None	MySQL/Mongol Connector	JSON file structure
File IO	Pop-up GUI Save/Share	Java/Python IO Lib	JSON file structure
Simulator	Simulation Interface	Java/Python Code	Java/Python Object