GSoC'21 Proposal

Mobile Touch Devices

CircuitVerse

Please view my proposal via the doc link provided as pdf doesn't support .gif files
https://docs.google.com/document/d/1VSYdepCDBGLrcrpKvAWOXz0hN6i_-2ddRUhh8VI60g8/edit?usp=sharing

PERSONAL DETAILS

Name -	Devjit Choudhury
Course -	Mathematics and Computing
University -	Indian Institute of Technology Kharagpur
Email -	devjitc1@gmail.com
Github -	https://github.com/devartstar
LinkedIn -	https://www.linkedin.com/in/devjit-choudhury-717915151/
Phone -	+91-9065887737
Current Country -	India
Resume -	https://react-portfolio-devjit.herokuapp.com/resume

ABOUT YOURSELF

I am a second-year undergraduate student pursuing integrated M.Sc in Maths and Computing at Indian Institute of Technology, Kharagpur. I am an open-source enthusiast, curious, and a fast learner. Apart from programming, art is my passion.
I am also a President awardee and have won prizes from many prominent figures like Mr. Ratan Tata and Shree. Pratibha Patil.
The thing that distinguishes me is that I focus a lot on perfection in my code or whatever I am creating.

WHY CircuitVerse ?

The thing that I loved most about the CircuitVerse project is that I found a lot of areas in the domain of my skills where I can contribute and enhance the features especially the UI and compatibility.

I could understand the codebase and believe that I can make my little contribution towards the enhancement of some features.

I took a course in Basic Electronics this semester and I found out that there are very few sites that focus on teaching (Interactive Book) as well as giving a platform to design and implement them.

I also found the lack of such a single complete platform on which colleges or universities can completely rely upon. This was a motivation for me to choose CircuitVerse.

So all in all, I would love to contribute to CircuitVerse.

Previous Contributions TO Open Source

1. I participated in KWOC (Kharagpur Winter of Code).
    a. There I spent my winters contributing to Graph Visualization Project.
    b. I have attached below the link of my PRs made. **https://github.com/Graph-Visualization/graph-api/pulls?q=is%3Apr+author%3Adevartstar**

2. I was also a lead frontend team member of iC2C (**https://www.ic2c.in/**) which is a startup focusing on helping students notifying internships and job openings.

COMMITMENTS

1. I am not planning any vacation during this summer.
2. I am not planning on taking any classes this summer.
3. No, I don't have any other employment during the GSoC period.
4. My usual work hours would be(all in IST):
    - 8:00 - 13:00 Hrs
    - 17:00 - 22:00 Hrs
    - 23:00 - 2:00 Hrs

    I plan to put in 40 to 45 hours each week, though I will certainly, make up for the lost time if any, and would be happy to extend my working hours if needed.

CONTRIBUTIONS SO FAR

- **PRs made by me -**

Fixed the Navbar hidden behind the widgets and
Fixed the position of the overlapping Timeline
https://github.com/CircuitVerse/CircuitVerse/pull/2179

- **Issues solved through this PR -**

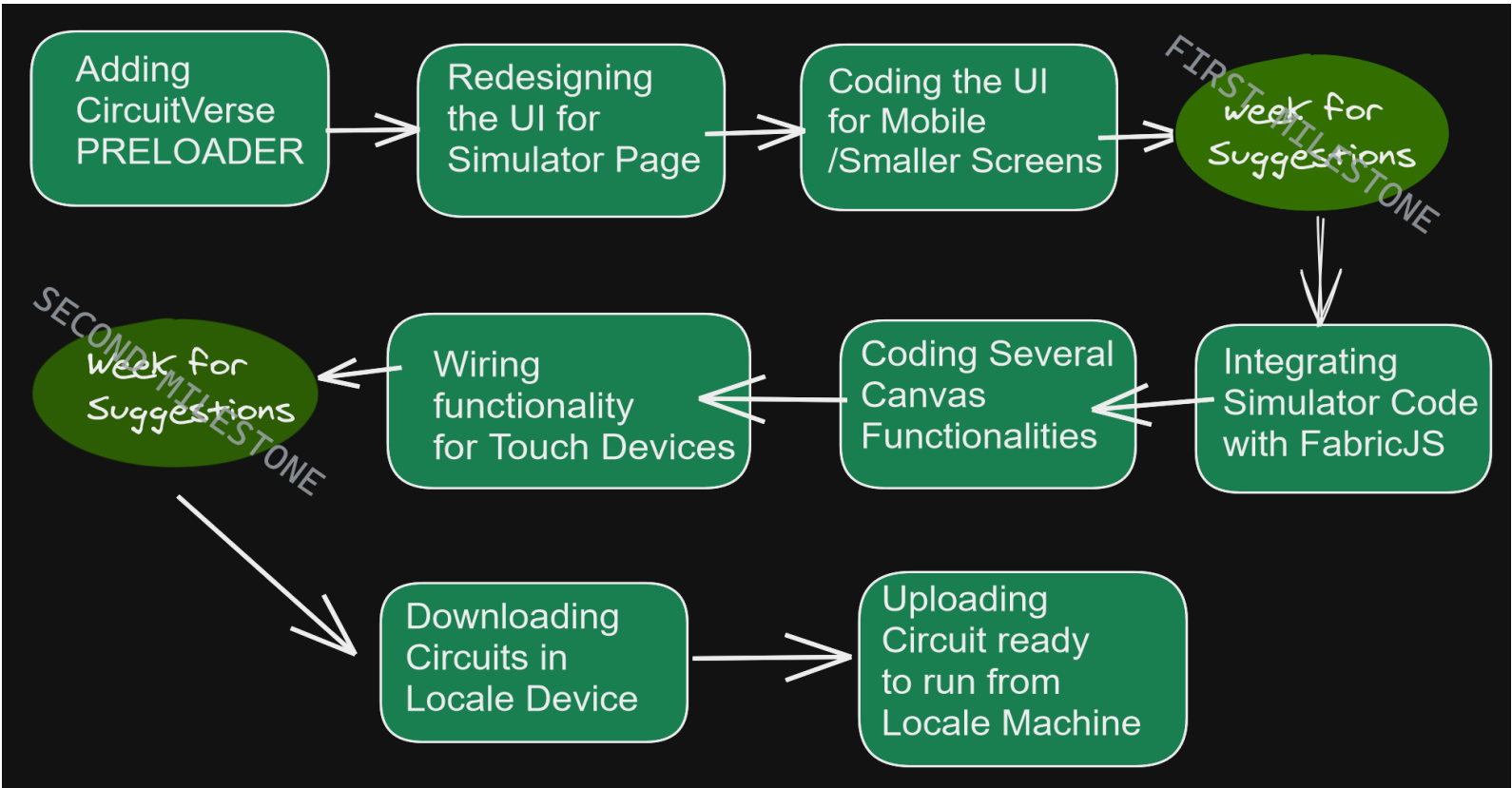
https://github.com/CircuitVerse/CircuitVerse/issues/2108
https://github.com/CircuitVerse/CircuitVerse/issues/1945

- **Issues and Bugs found -**

Improper positioning of Context Menu -
https://github.com/CircuitVerse/CircuitVerse/issues/2145

PROPOSAL OVERVIEW

- The project I choose to work upon is the Touch screen compatibility and Redesigning the complete Simulator Page UI for small screen devices.
- By the end of this project, you can expect to have a fully functional Simulation area for Mobile and Touch Screen Devices with improved UI for better and easier engagements.
- I want to spend the
    1. **first 4 weeks Redesigning the UI completely** for mobile screens
    2. **Next 4 weeks for Wiring and Touch Gesture Functionality** for circuit element
    3. **Next 2 weeks for Saving Circuits Locally** so that it can be imported and used in future.
    4. **Last two weeks for documentation and Presentation.**
- The languages I wish to use in my project are
    1. Ruby on Rails
    2. Javascript & libraries -JQuery, HammerJS, FabricJS
    3. HTML and CSS (SCSS)



PROJECT PLAN : PRELIMINARY

Week Number	Start Date	End Date	Tasks To Be Completed
-	*4th May*	*30th May*	Community Bonding Period
1	*7th June*	*13th June*	Add a custom-made page preloader for CircuitVerse using SVG Animation.
2	*14th June*	*20th June*	Redesign the UI of the Home Page and Simulator Page for small screen devices (and some also some enhancement in the design for desktop view)
3	*21st June*	*27th June*	Code the Newly designed UI for both the Homepage and Simulation Page.
4	*28th June*	*4th July*	Week specifically for taking inputs and suggestions about the new UI from mentors before moving to touch screen compatibility.
			FIRST MILESTONE
5	*5th July*	*11th July*	Integrating the codebase with Hammer JS and Fabric JS
6	*12th July*	*18th July*	Coding several functionalities of canvas and its elements for mobile devices.
7 & 8	*19th July*	*25th July*	Coding for the wiring functionality of the circuit elements.
			SECOND MILESTONE
9 & 10	*26th July*	*8th August*	Downloading the circuits into Local Machine in such a way that it does not lose its circuit properties.
11	*9th August*	*15th August*	Writing and Updating the docs
12	*16th August*	*22nd August*	Presentation for Mentors
			SUBMISSION

PROPOSAL DETAILS

(TASK 3)
UNDERSTANDING THE CODEBASE

Link for the documentation site -
https://circuitverse-codebase.netlify.app/

The Github link for the code -
https://github.com/devartstar/CircuitVerse_Documentation

The HTML files are contained in :

```
CircuitVerse\app\views
```

And the file for the simulator HTML is :

```
CircuitVerse\app\views\simulator\edit.html
```

And all the other Simulator related code can be found in :

```
CircuitVerse\simulator\src
```

WEEK 1 - Adding a custom made SVG animated page Preloader.

Preloaders or loaders are what you first see on the page when the rest of the site's content hasn't loaded yet. Preloaders are mostly animated simple illustrations, they are created to entertain visitors while the server keeps processing.

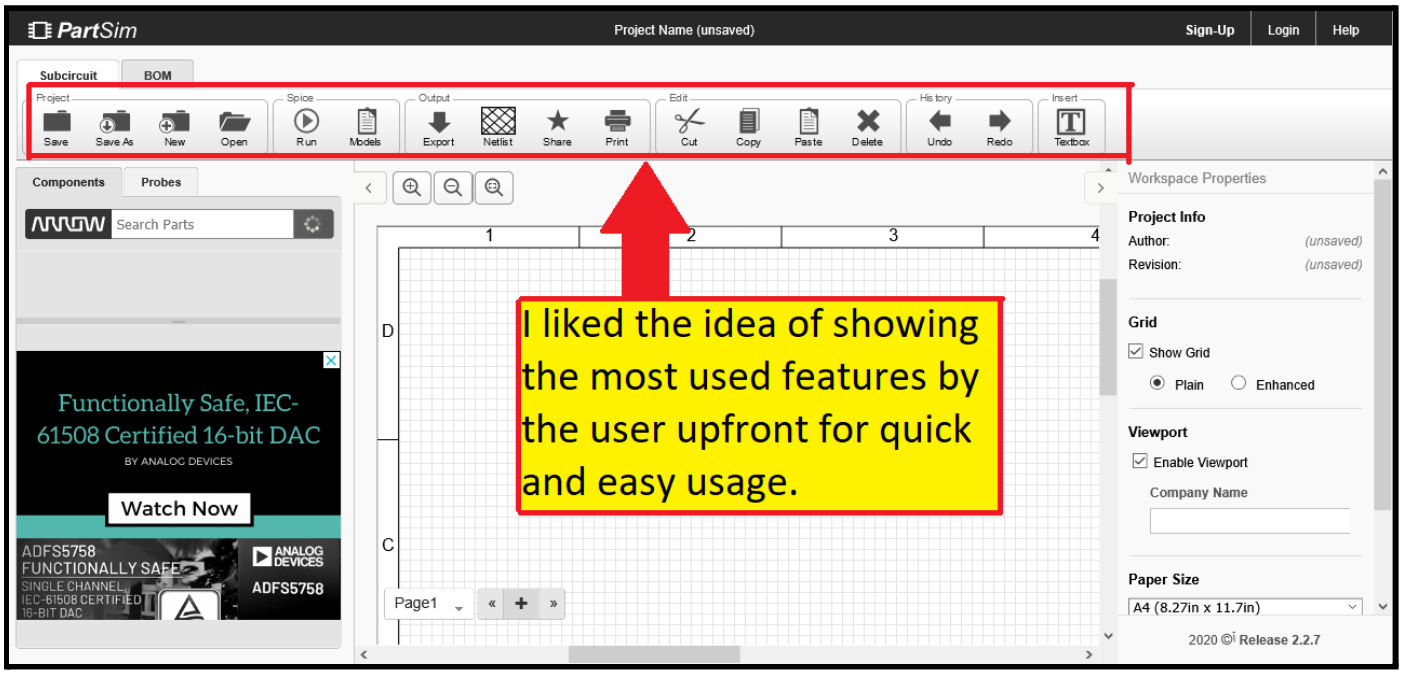
Why Should We Use a Preloader?
Loading time is very crucial for your website's success. We need to keep your users engaged for those milliseconds, if you want them to retain, we have to consider preloaders more seriously.

I am attaching below the code of the preloader I designed -
https://codepen.io/Devartstar/pen/GRrqZVK?editors=1100

WEEK 2 & 3 - Redesigning and Coding the UI for the Simulator Page for mobile devices.

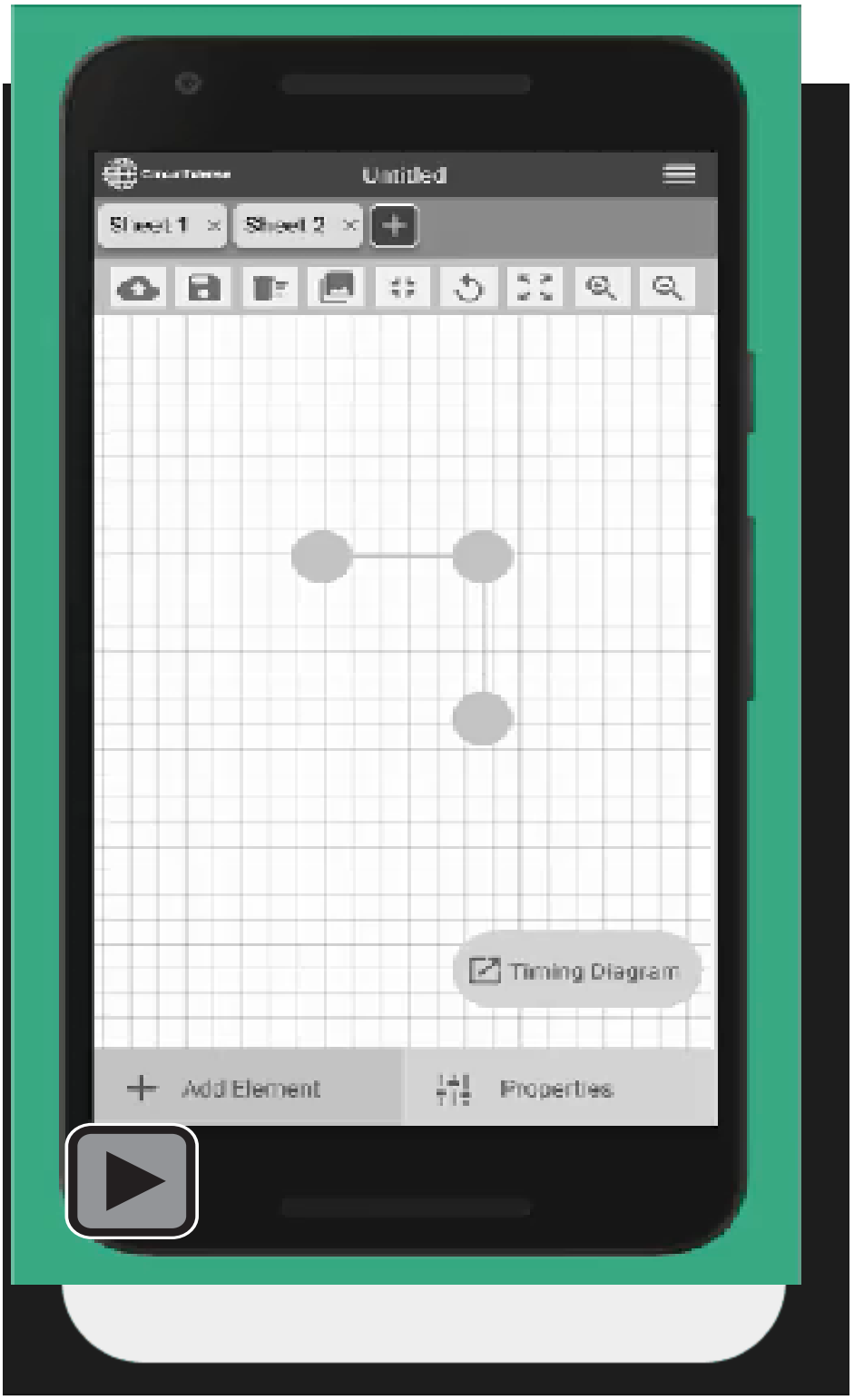
Some of the problems in the current UI of the Simulator Page in Mobile View -

1. On loading , everything (the canvas, widgets, navbar) on the Simulator Page seems to be too small (Zoomed out) which causes a lot of problems for the user to select circuit elements OR work properly in general.
2. On zooming in and out the entire view, the page zooms in and out. I think it would be better if only the canvas gets zoomed in and out.
3. My suggestions - I think CircuitVerse provides a lot of features (like preview circuit, download options ... etc). I feel it would be better if we can provide some of the most used features on the top and not search for them every time.



The rough idea to implement this is to have a navbar at the top only containing the CircuitVerse Logo, ProjectName & SignIn-SignOut options. And immediately below that be all the important and frequently used functionality which is infinitely slidable.

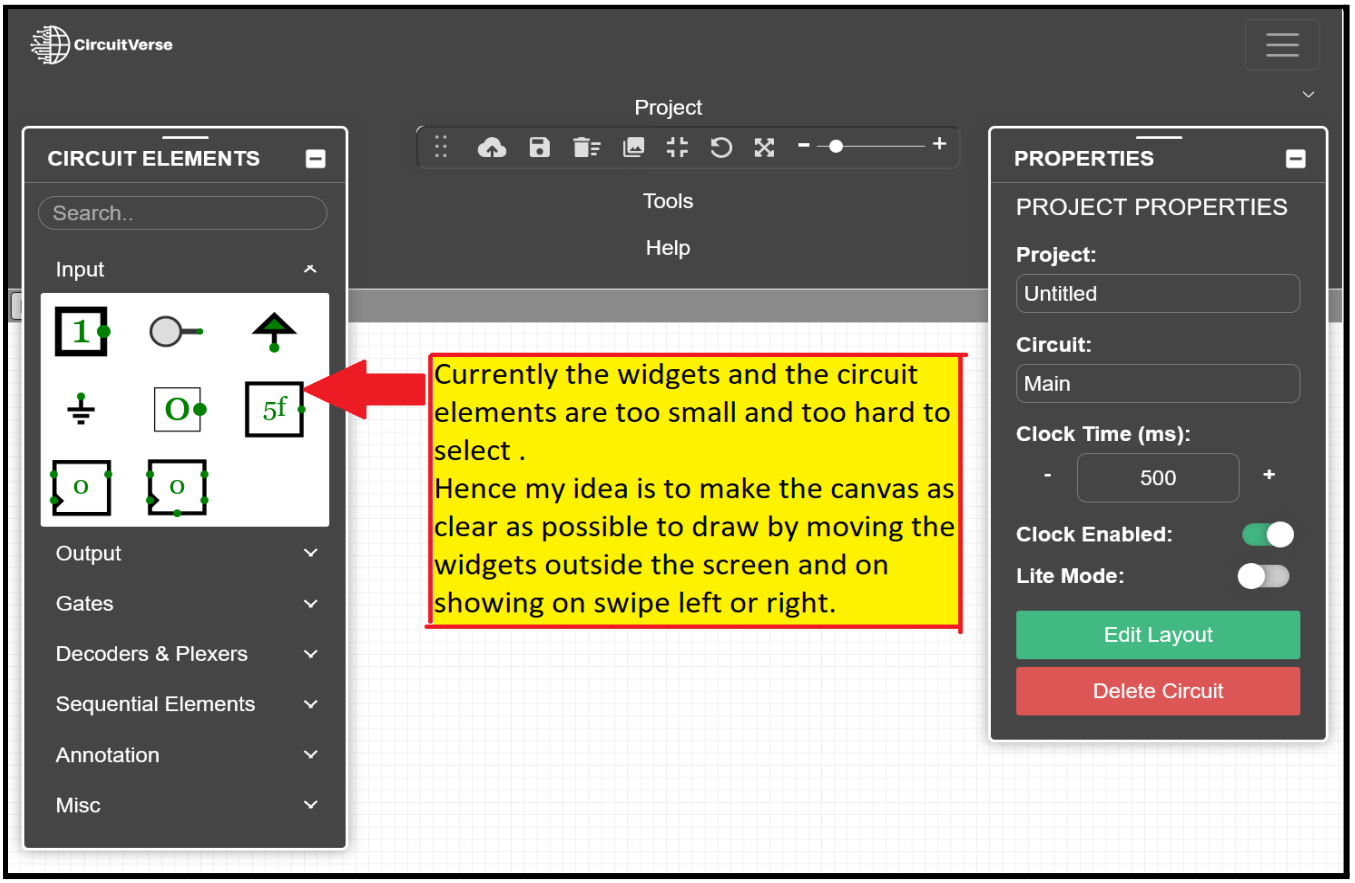
I am attaching below the .gif of Navbars view in Mobile Screen.

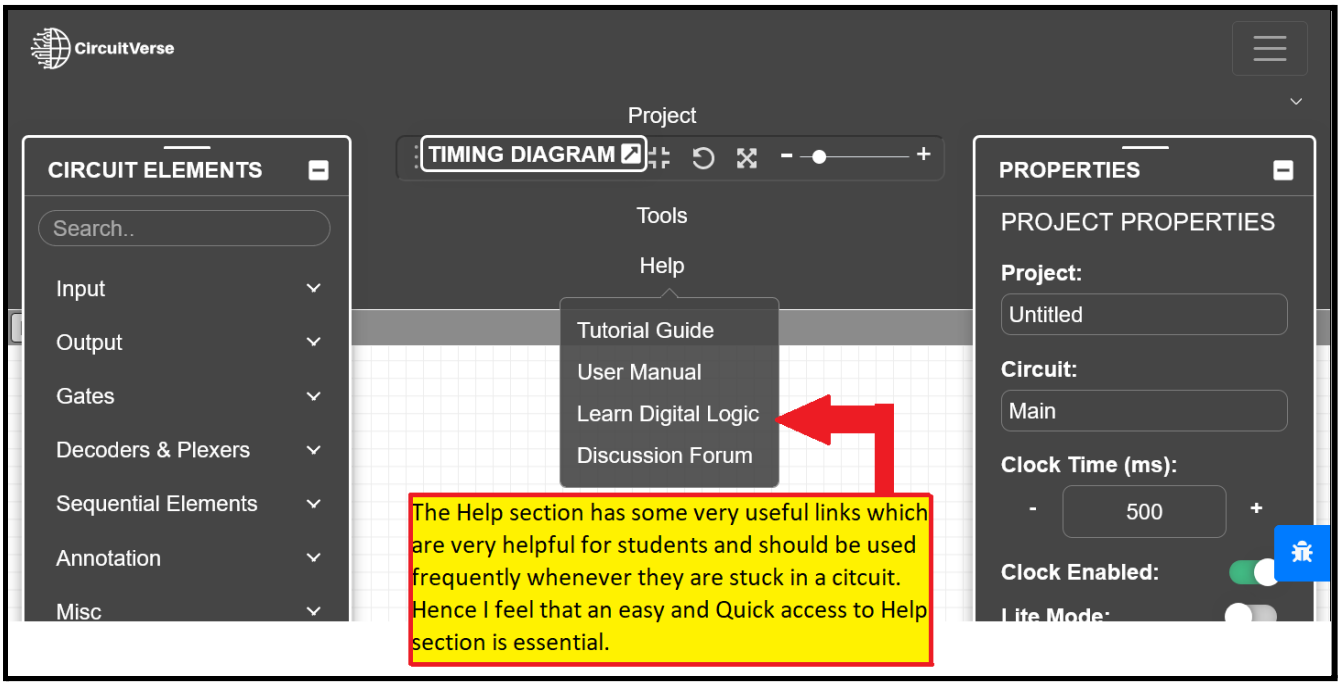


https://codepen.io/Devartstar/pen/oNBYayw

Now, let's focus on the Widgets & Easy accessibility of the Help section in mobile view -

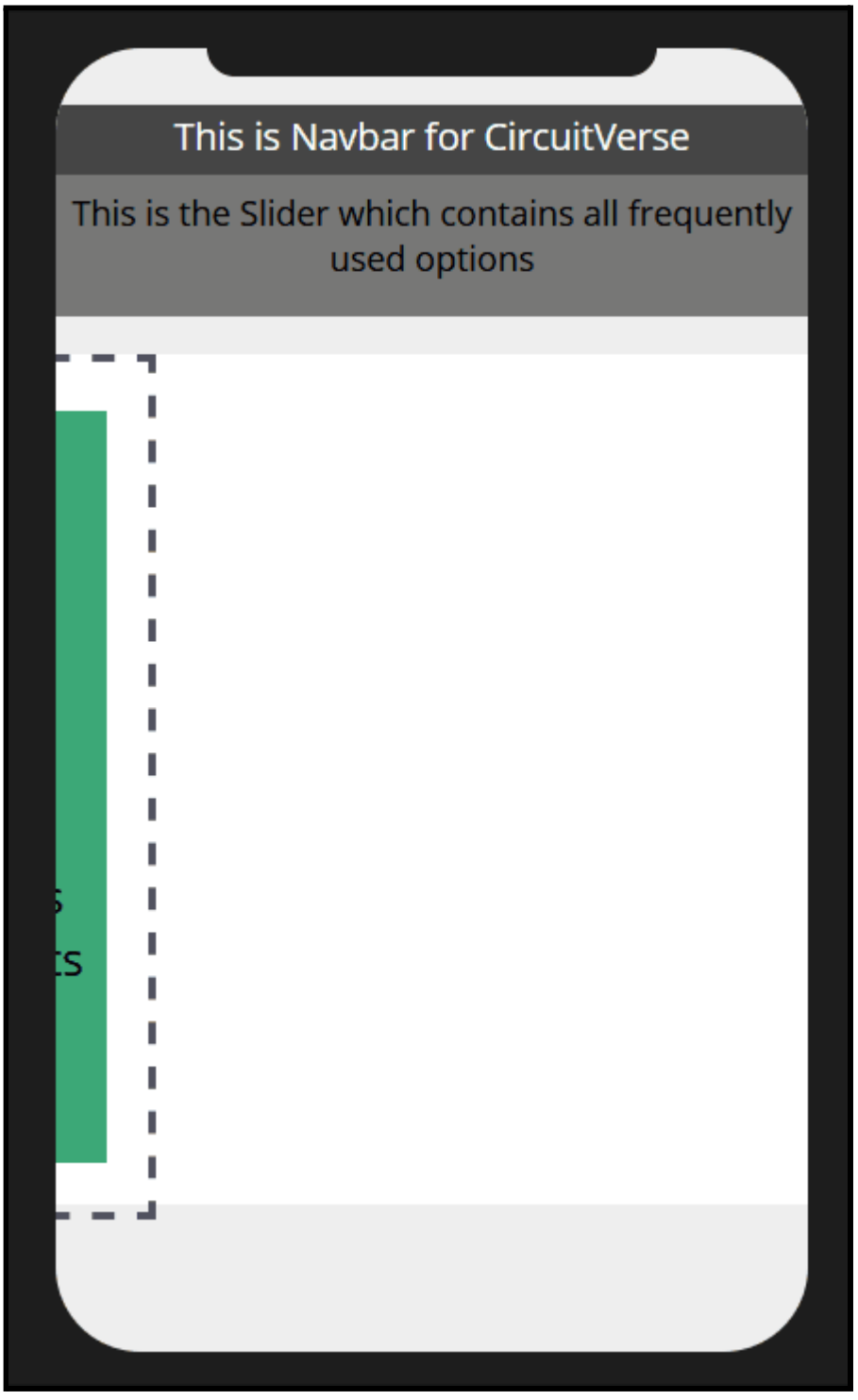
The size of the widgets is too small and hard to use and increasing the size it covers up most of the space in the canvas so drawing circuits becomes hard.





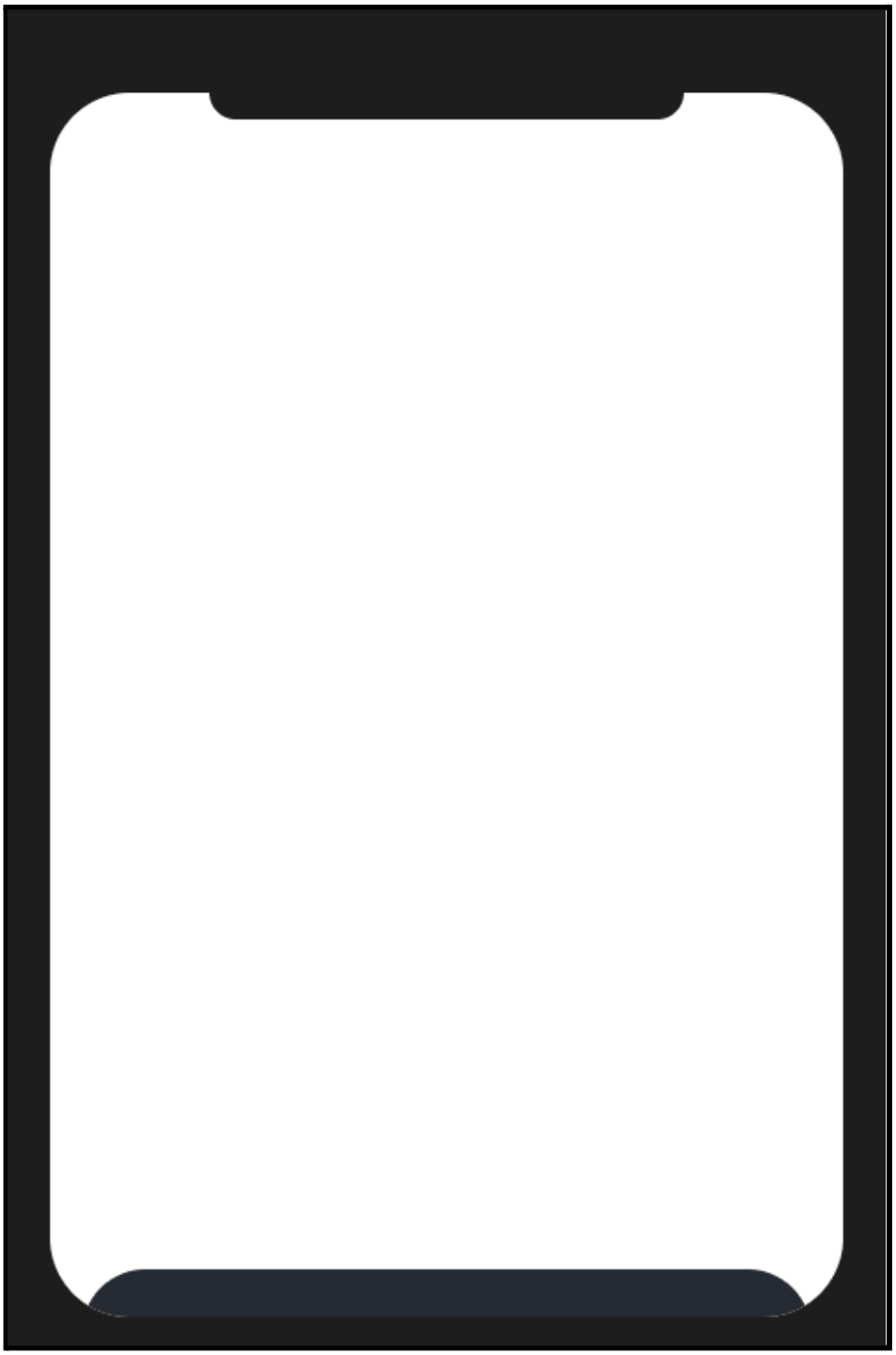
My idea is to make the canvas as clear and empty as possible to make the circuit easily. So making a Slidable Widgets - Once we select an element from the widget it will automatically slide back out of the view.

I am attaching below the GIF of the idea of how it will look in the Mobile Screen - (WireFrame for Action)



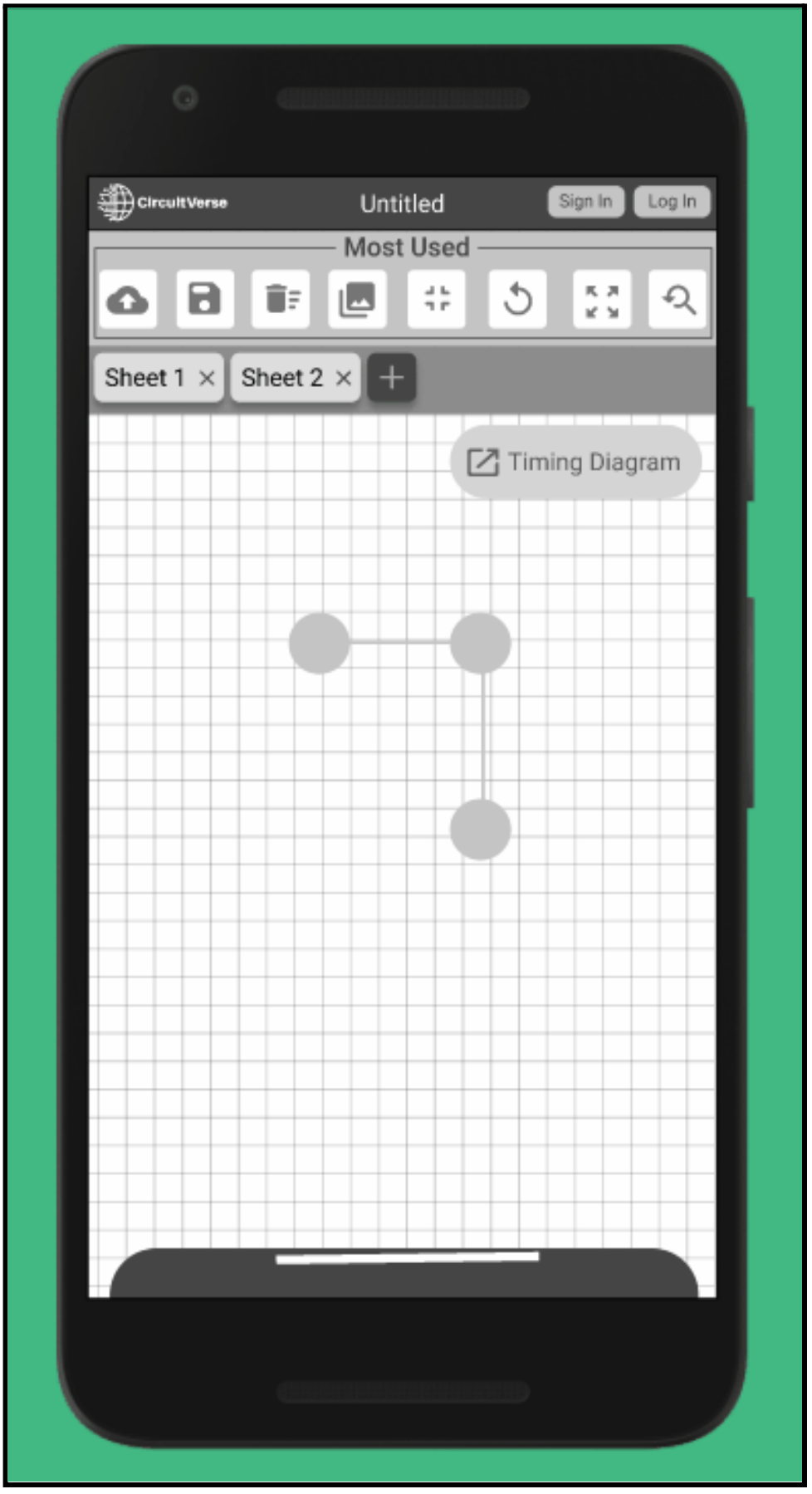
I am attaching below the code for the following below -
https://codepen.io/Devartstar/pen/PoWGXVE

I am attaching below the GIF of the idea of how it will look on, the Mobile Screen -

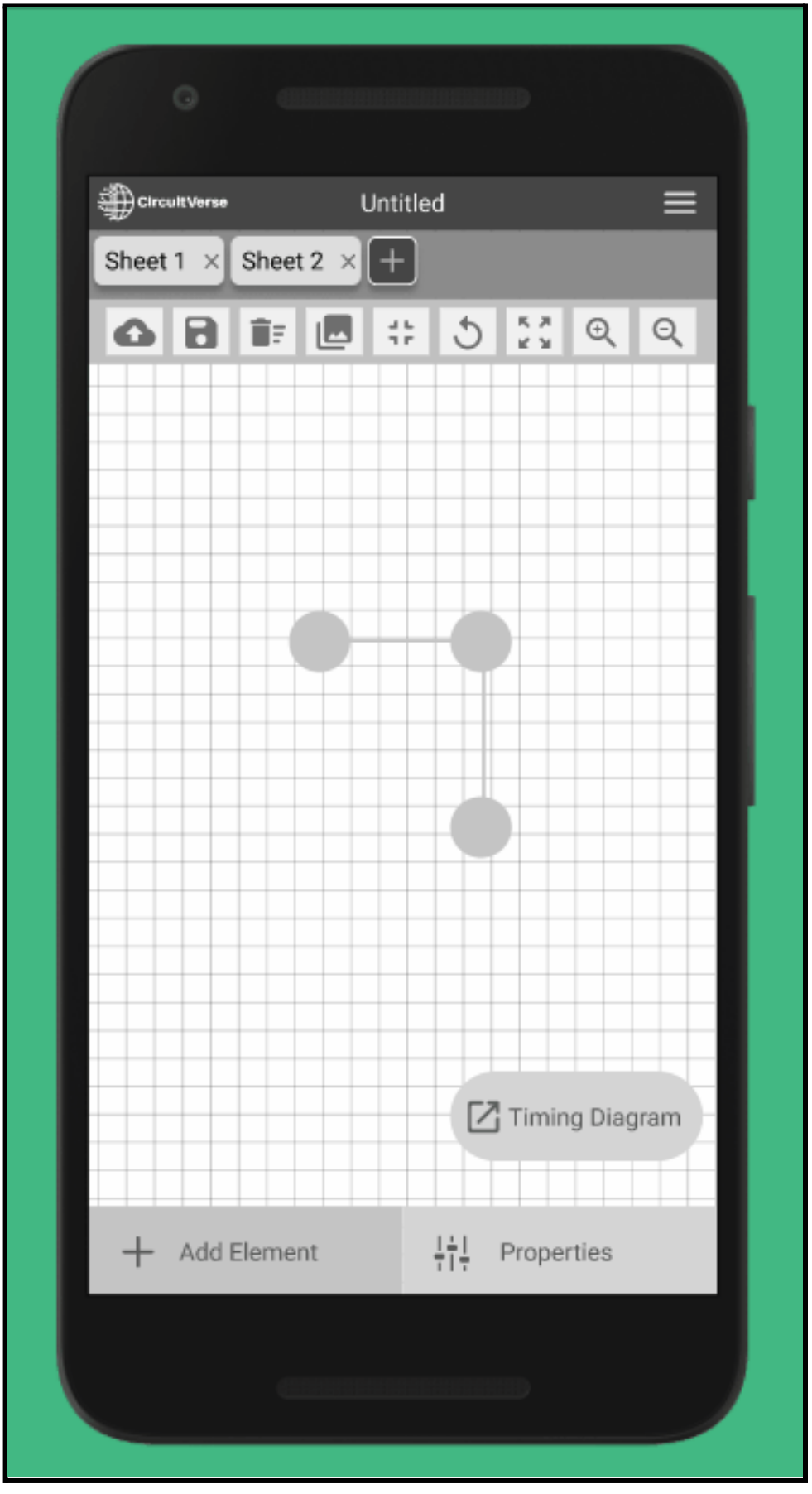


I am attaching below the code for the following below -
https://codepen.io/Devartstar/pen/WNRRNoN

GIF of the complete design in Figma



Check out GIF of an Another Design in Figma



(TASK 4)

I am attaching below the UI design in Figma -

https://www.figma.com/file/IlsAftaQth4q54U4SOJITS/CircuitVerse?node-id=0%3A1

There are two pages in the file each pertaining to a different design
(For the final design I will make the changes as suggested by mentors)

WEEK 4 - Buffer Week to take Input from Mentors about the new UI and to complete pending tasks.

Assigned this week to take suggestions from mentors about the newly made UI and to implements the input suggestions or to complete any pending work.

(TASK 6)

In the following weeks I will discuss the integration of FabricJS and the implementation of wiring and other functionalities

WEEK 5 & 6 - Integrating Fabric JS in the codebase to work along with Canvas API.

I wanted to reconstruct the entire Simulator code using FabricJS. But since the majority of the Simulator Code has already been written using the simple Canvas Web API we can integrate both in such a way to use both the features.

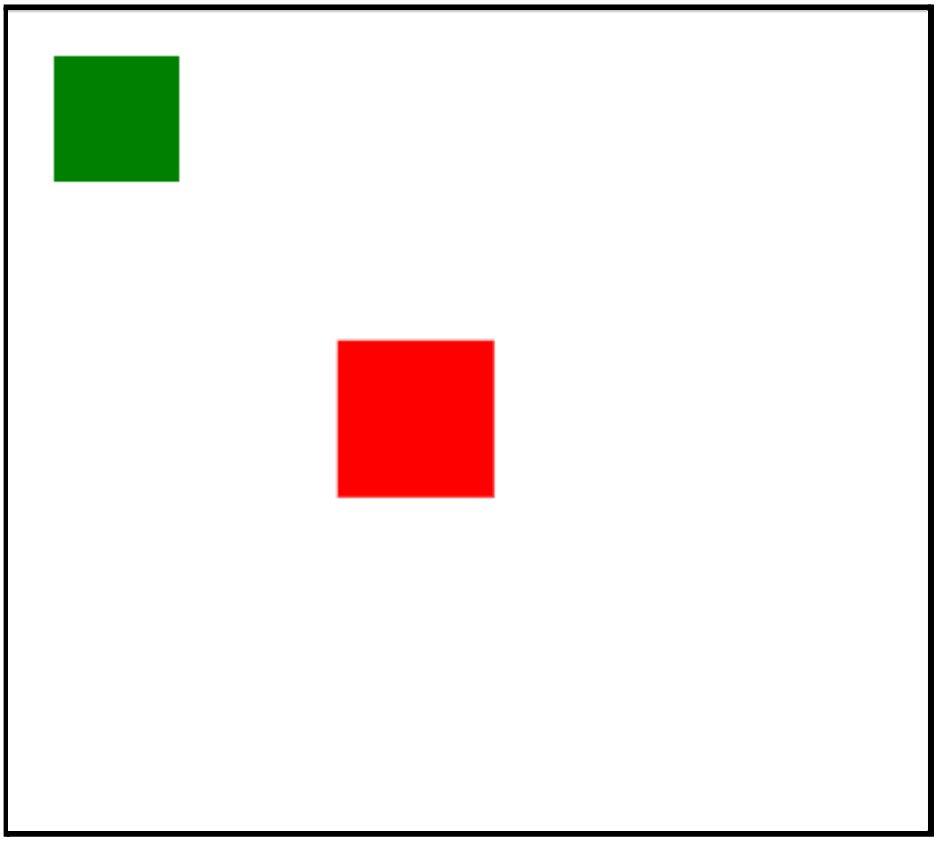
To do so let us add canvas to our HTML.

```
<canvas id='c' class="can" width="600"
height="600"></canvas>
```

```
var canvas = new fabric.Canvas('canvas');
const c = document.querySelector('.upper-canvas');
```

```
const ctx = c.getContext('2d');

var rect = new fabric.Rect({
  left: 100,
  top: 100,
  fill: 'red',
  width: 50,
  height: 50
});



//  Red Rectangle via FabricJS
canvas.add(rect);

//  Green Rectangle via Canvas API
ctx.fillStyle = 'green';
ctx.fillRect(10, 10, 40, 40);
```

From the following code snippet - Red rectangle formed using FabricJS and Green rectangle is formed using simple canvas API.



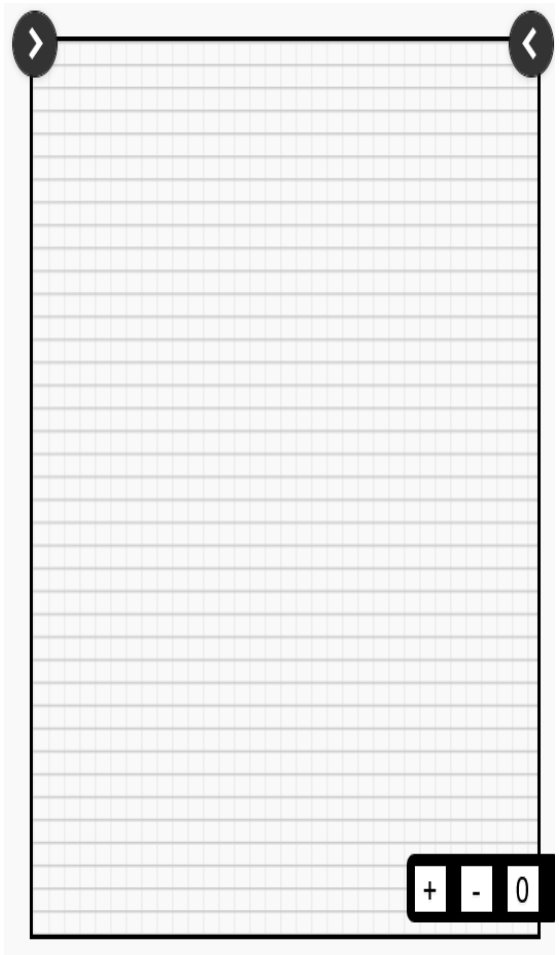
What FabricJS does is that it divides the canvas into two parts, The Upper Canvas, and the Lower Canvas. The Lower Canvas serves as a reference to the upper canvas (where all the circuit elements will be drawn)

So, the Upper Canvas is basically the canvas we need to refer to while using a simple canvas API.

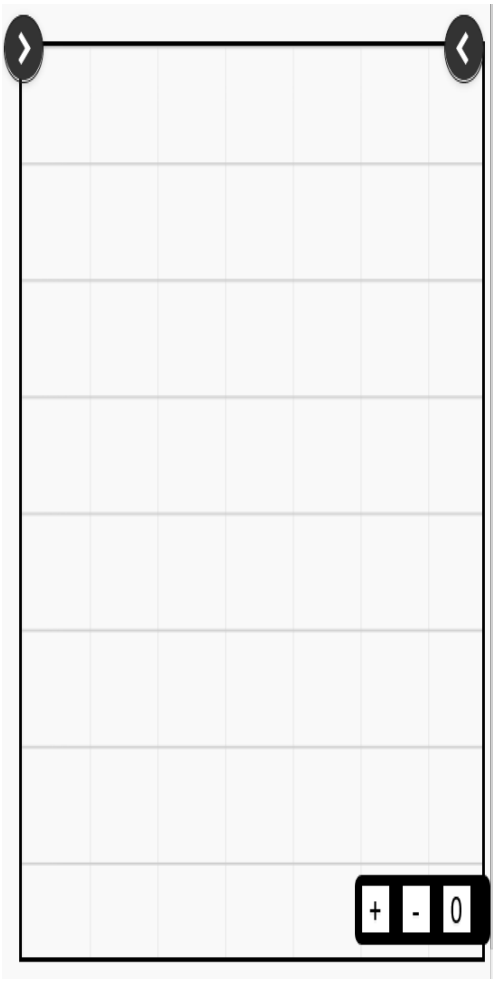
WEEK 6 & 7 - Integrating the following Functionality for the Touch Devices.

Dividing Canvas into Grids Lines using FabricJS.

These grids are the references/constraints for the objects on the canvas to move.



Grid - Size: 10



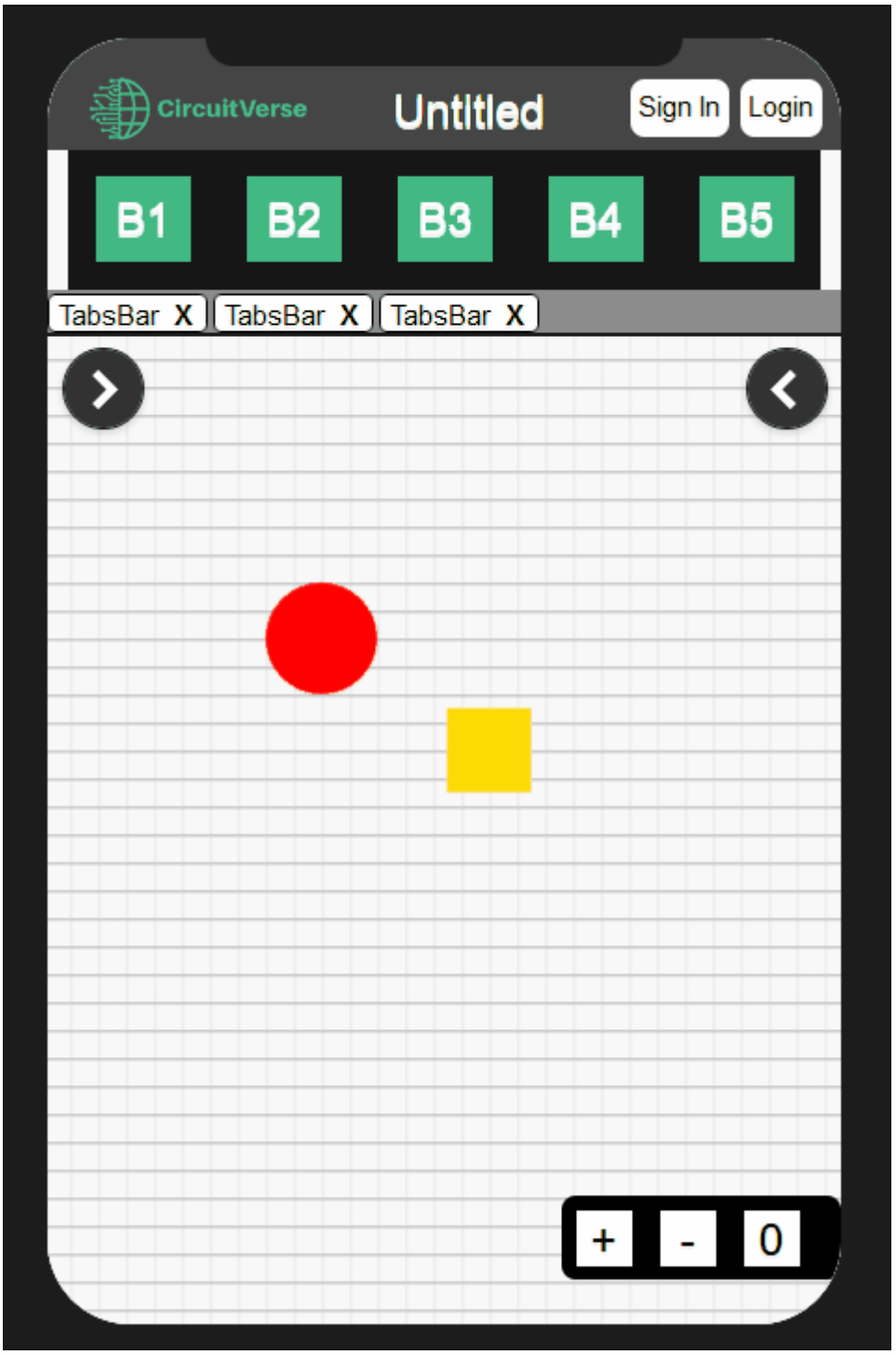
Grid - Size: 50

```
//  CREATING GRID FOR A CANVAS

const grid = 10;
for (var i = 0; i < 1000; i++) {
     const vlines = new fabric.Line([i * grid, 0, i *
grid, 100000],

{
     stroke: '#eee',
     selectable: false
     })
     canvas.add(vlines);
}
for (var i = 0; i < 1000; i++) {
     const hlines = new fabric.Line([0, i * grid,
100000, i * grid], {
     stroke: '#ccc',
     selectable: false
     })
     canvas.add(hlines);
}
```

1. **Adding the Zoom In and Zoom Out functionality** For mobile screens there would be two options for Zooming In and Out.
    - Firstly, By clicking on the + - 0 at the bottom, or by using the zooming slider

    - Secondly, By using the pinch functionality. Moreover, by using HammerJS for this touch gesture we can control the extent of the pinch and the speed of Zooming in and out of the canvas.

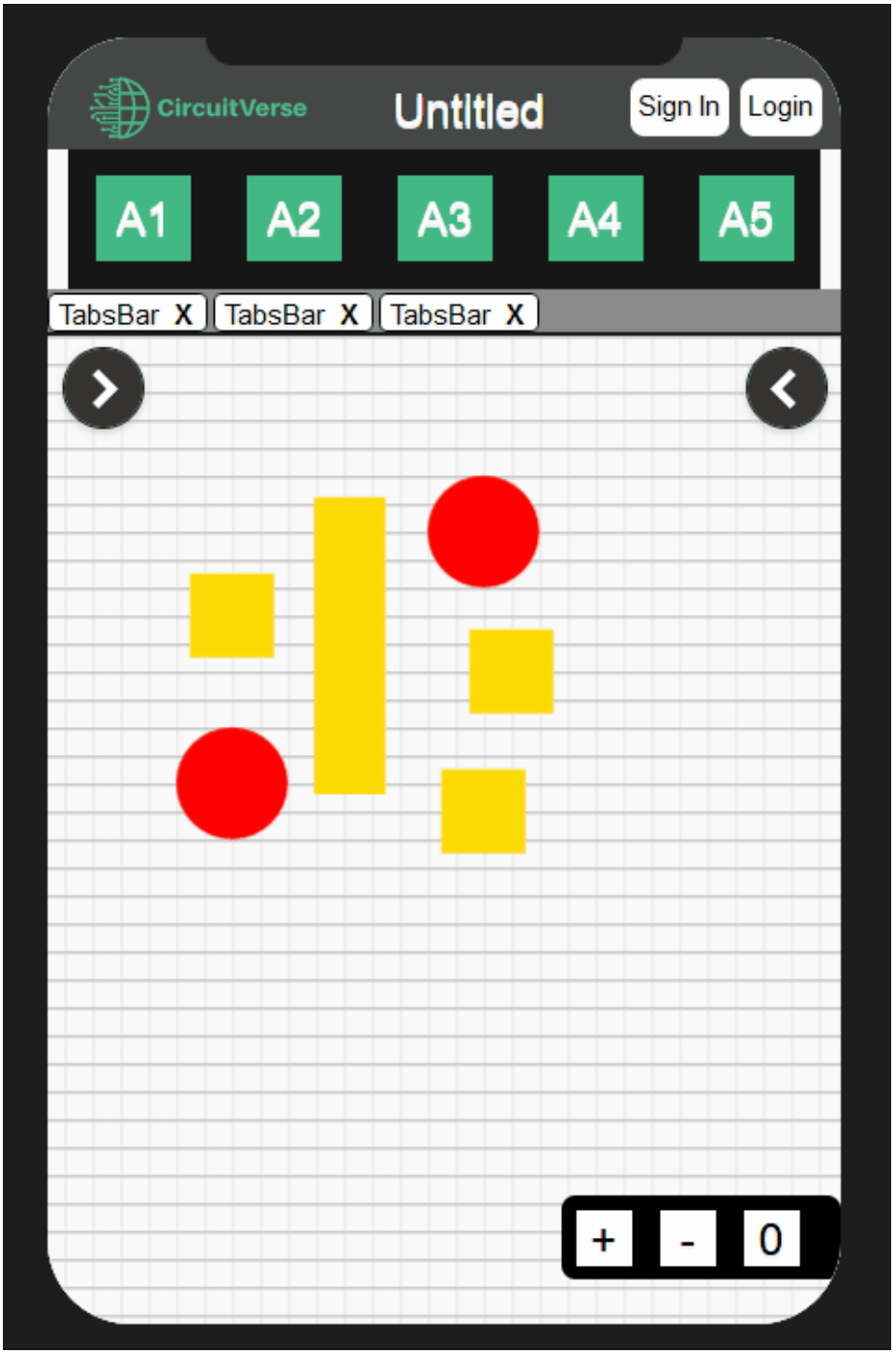
Clicking '+'	Zooms IN
Clicking '-'	Zooms OUT
Clicking '0'	Zooms to Default and center position.



2. Making the Canvas draggable -

Sometimes when drawing large circuits instead of zooming out we can use the **Toggle Canvas** functionality which helps in moving the entire canvas along with all the circuit elements. **Also, we have to disable canvas drag before trying to adjust elements' positions.**

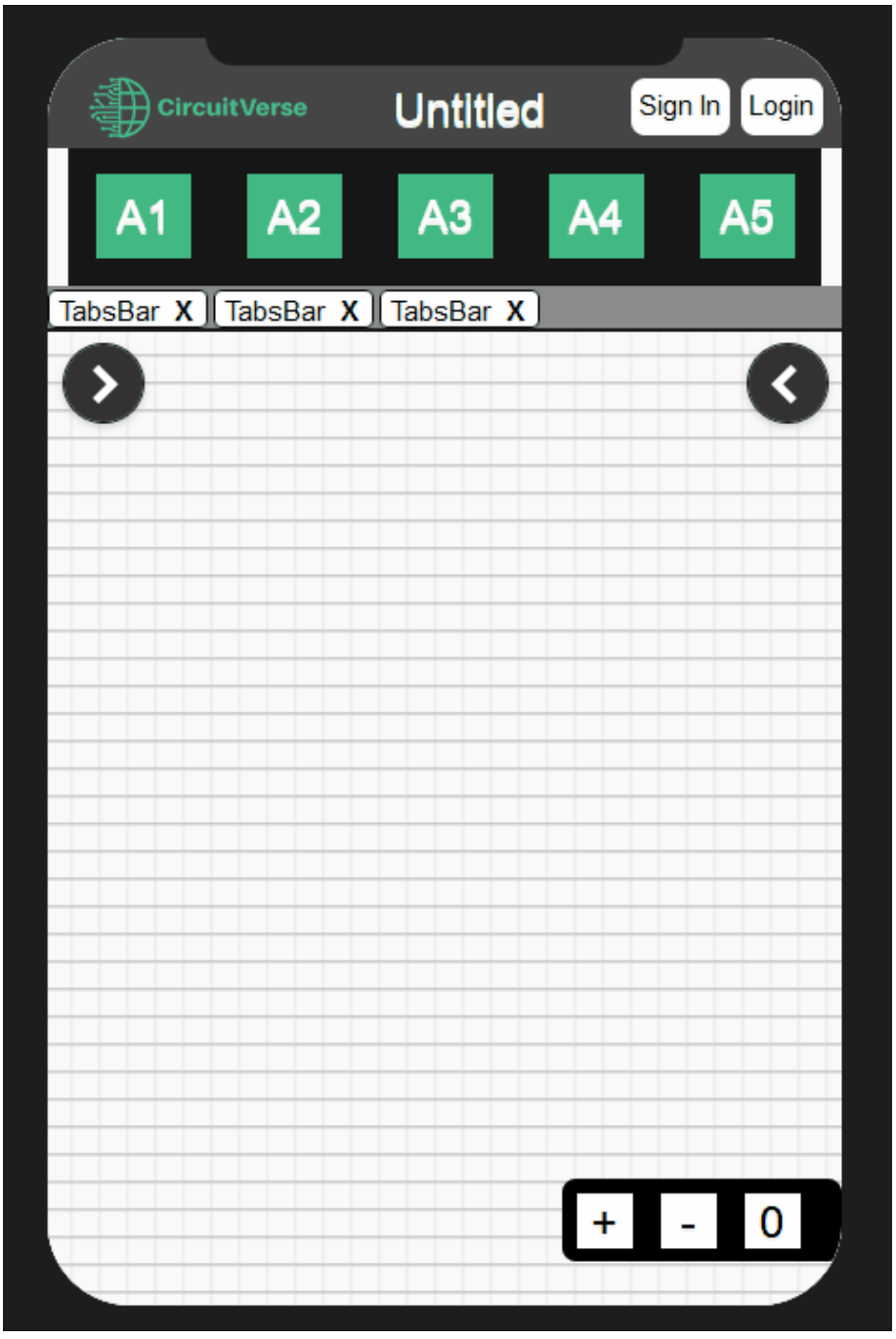
We have to enable this functionality first by clicking the adjust button else enable this functionality freely may cause a lot of problems while drawing circuits.



3. Scaling and Rotation of Elements -

Since Zooming IN or OUT of the canvas is not always the best practice. Hence being able to zoom in or out the circuit elements will make building the circuit easier.

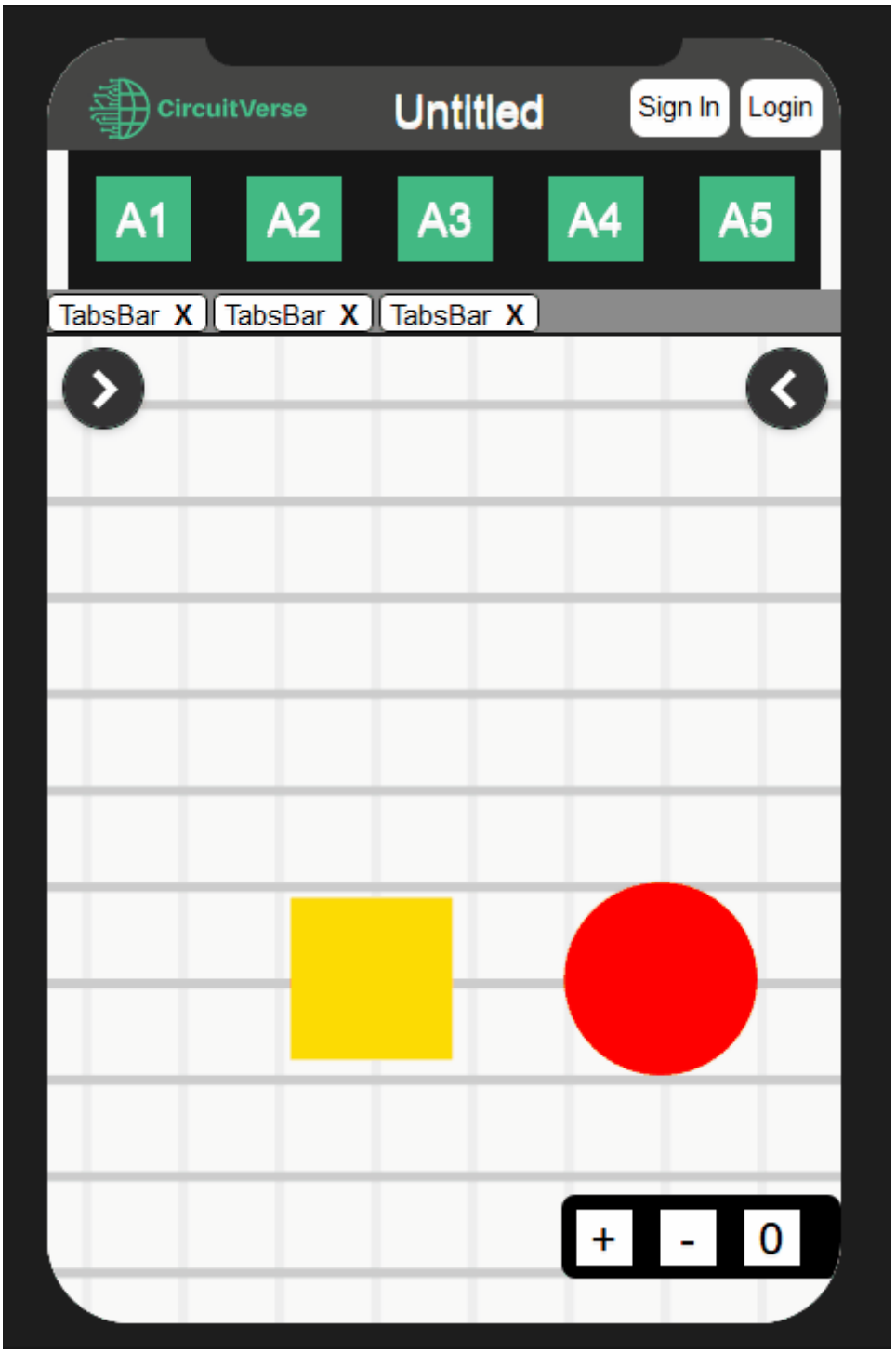
Also presently there is no option for rotation for the circuit elements which can also be easily enabled using FabricJS. Enabling rotation will help in making wire connections cleaner.



4. Dragging and Placements of Circuit Elements

Currently, CircuitVerse does not allow the dragging of circuit elements in touch devices. Allowing free movement of circuit elements is not an ideal idea as it may cause some problems while connecting wires. Hence I want to apply restricted movements according to the grids positioning.

So let's first zoom in on the canvas so that the restricted movement is prominently visible.



Furthermore, we can apply restricted rotation and restricted rotation-

- In the following project i have restricted rotation to the following angles - [0, 45, 90, 135, 180, 225, 270, 315, 360] degrees.
- We can only allow scaling in multiples of grid size

ALL THESE ABOVE FUNCTIONALITIES WILL ALSO BE ENABLED FOR DEVICES WITH LARGER SCREEN or DEVICES USING MOUSE.

The above functionalities can be also coded using Vanilla JS.

GitHub link for the complete project -

https://github.com/devartstar/GSoC_Proposal_CV_Simulator

Page Link -

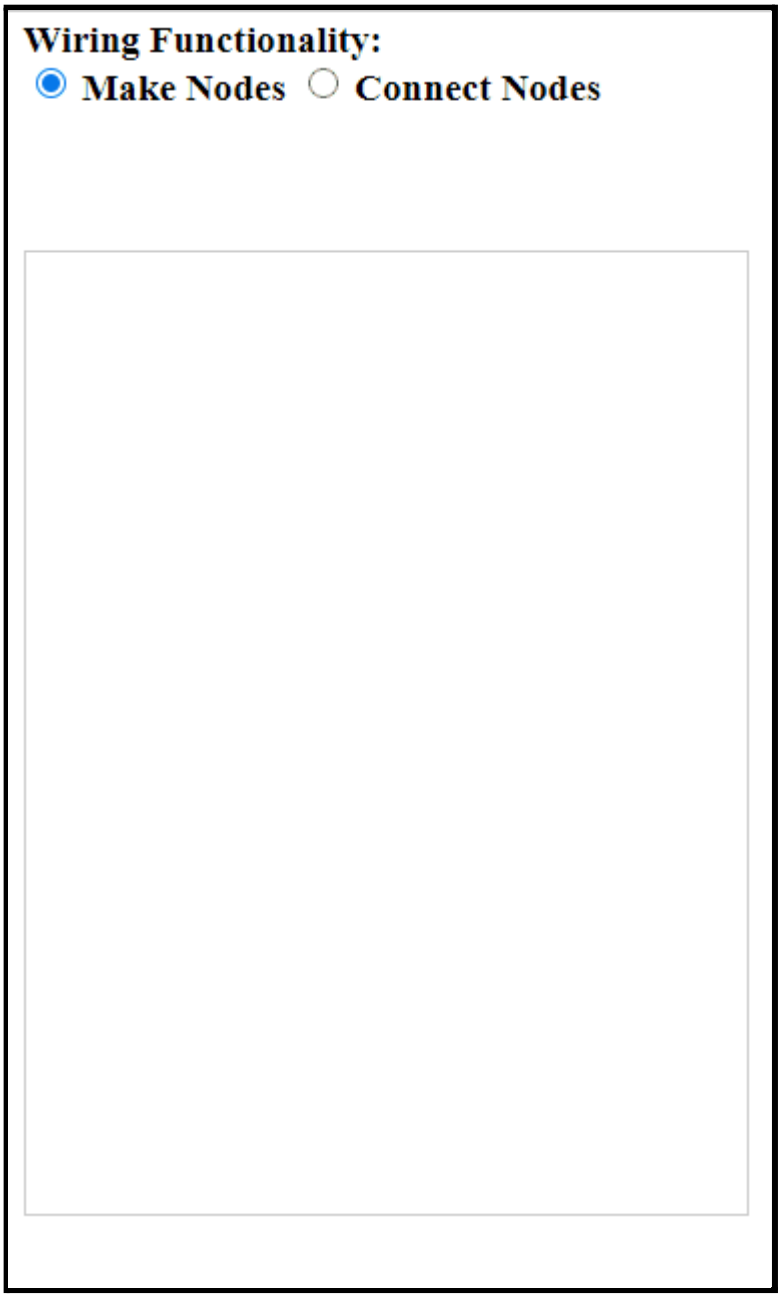
https://devartstar.github.io/GSoC_Proposal_CV_Simulator/.

WEEK 7 & 8 - Coding the Wiring Functionality for Circuit Elements in Touch Devices.

Idea - The circuit elements would be SVG elements on the canvas. We can easily include SVG elements in the canvas by

```
fabric.loadSVGFromURL(url, function(objects, options) {
      objects.forEach(function(svg) {
         svg.set({
            top: 90,           // position on the canvas
            left: 90,
            originX: 'center', // reference for position
            originY: 'center'
         });
         canvas.add(svg).renderAll();
      });
   });
```

Next at the points where we want to connect wires, we can add nodes (we can also add a hover effect on that part of the SVG where nodes should be created) and then connect them with Lines. (Wires) to end the connection tap on open space.



The code for the following can be found below -

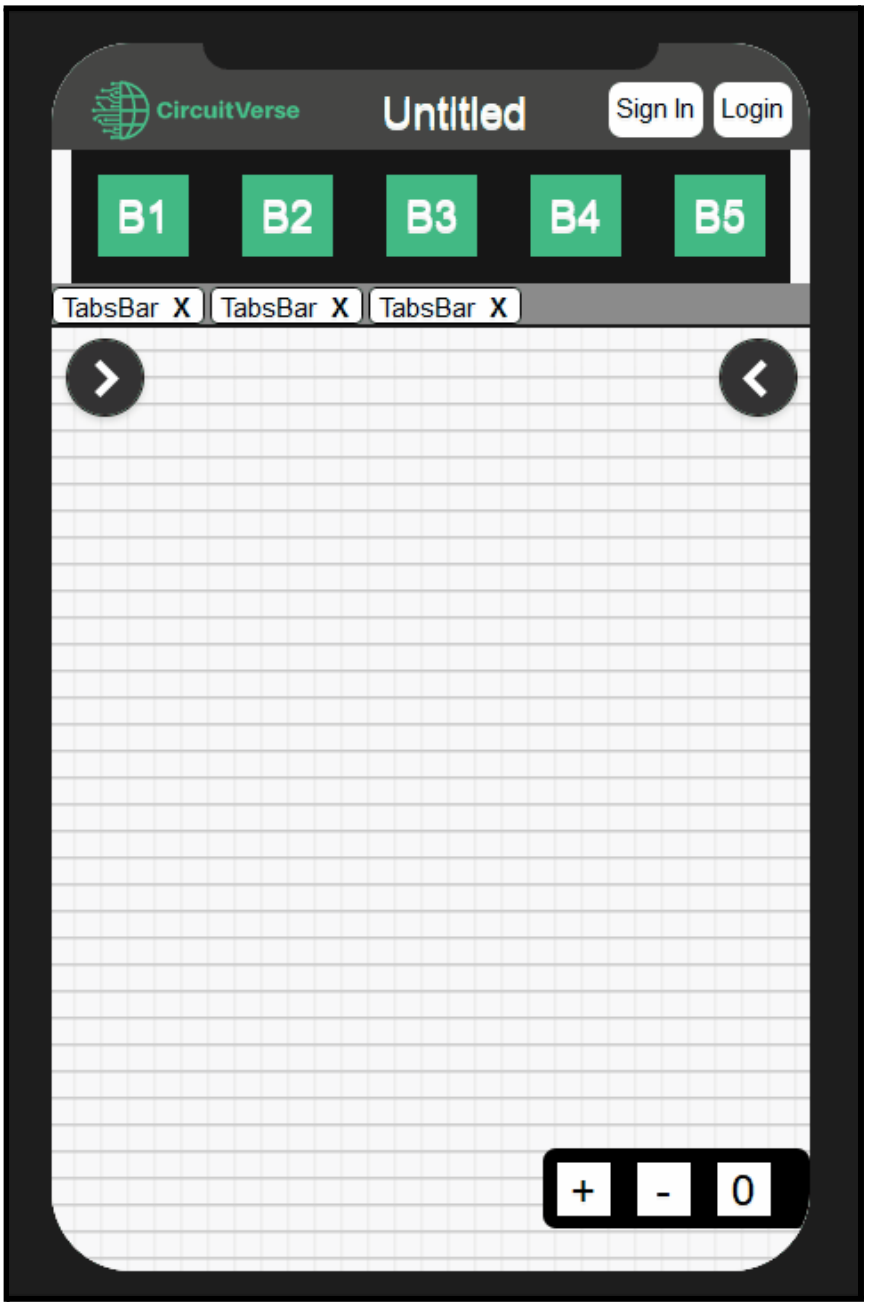
https://jsfiddle.net/v96g78h4/14/

WEEK 9 & 10 - Downloading Circuit Elements as SVG (Maybe in another Necessary format) & Enable importing of these Circuits.

I would like to research different possible ways to save circuits in our locale machine in the form of some file and being able to open it from the locale machine.'

Just like Falstad exports circuits as a text file to the local machine which can be opened again to get back the circuit.

The following example shows importing a dummy circuit made in PartSim.



WEEK 11 - **Preparing Documentation for all the changes Made to make it easier for others to contribute.**

While I will be keeping Sundays for Updating the document for the changes made Each Week in JS Docs but assigning this week to make some final changes or enhancement in the doc.

WEEK 12 - **Presentation for Mentors**

Research on JavaScript Libraries

After researching a lot I found out that the following JS libraries are the best to use based on ease in integration, maintenance of codebase, support across different devices and OS, user experience, performance, etc.

(TASK 1 & 2)

Understanding Touch Events & HammerJS

https://docs.google.com/document/d/1bzt_mYUM9mT80e3-6vLIF6d9oJLheM7ldJDKpDRL8RE/edit?usp=sharing

Most Amazing HTML5 Canvas Library : FabricJS

https://docs.google.com/document/d/1CQPH0QPQOKR_S-E69MMDnYrrm8Zd68T42d1FHXB3mW0/edit?usp=sharing

FUTURE WORKS

Creating Real-Time Shared Canvas with WebSockets

In the future, I want to work on the idea of Live Collaboration in CircuitVerse.
Since I have been learning and exploring Node for the past several months, I would like to work and collaborate on this project.

Github Repository Link for all the Tasks related Codes, images, GIFs, Videos, and Documentation related to the Proposal.
https://github.com/devartstar/GSoC_Proposal_Tasks