GSoC 2020 Proposal

Sugarizer Game Activity Pack

Personal Information

My name is Dhruv Misra and I'm currently enrolled as an undergraduate student at Guru Gobind Singh Indraprastha University, New Delhi. I'm in my third year pursuing a degree in Computer Science.

Location: New Delhi, India (UTC+05:30)
Languages: English, Hindi (Native)
Preferred working hours: 10:00 AM - 10:00 PM (can be shifted as per the requirement)

Contact Information

E-mail ID: dhruvmisra@live.com
Github: dhruvmisra
LinkedIn: https://www.linkedin.com/in/dhruv-misra-35a96a170/
Resume: Resume
IRC Nick: dhruvmisra or dhruv_misra

Programming Experience

I learned the basics of programming in the final years of high school and spent my freshman year focusing on Competitive Programming. I developed a keen interest in Web Development during the second year of university and learnt the javascript framework Vue.js. I have worked on multiple projects, some personal and some as interns for different organisations using Vue.js and other web technologies.

As my love for the framework grew, I created my own open-source NPM package for Vue.js called Vue-event-card. Apart from this, I have worked on various open source projects, the details of which can be found later in the proposal.

I started contributing to Sugar Labs by fixing bugs and as I became familiar with the codebase, I started implementing functionalities for different activities. I developed

fully-functional activities for Sugarizer and helped in porting others from Sugar. The details of these contributions are listed below.

Contributions to Sugar Labs

I have been contributing to the Sugar Labs' repositories since February and have solved a lot of bugs and added enhancements. Here is a complete list of Pull Requests:

GitHub Link	Title	Status
#652	Added Contents button in toolbar for Ebooks	Merged
#656	Fixed erase button overflow	Merged
#659	Chess Activity for GSoC 2020	Challenge
#662	Fixed text not being visible in Search bar	Merged
#667	Fixed text overflow in Shared Notes	Merged
#672	Fixed transparent background of dialog box	Merged
#674	Fixed opening links in Markdown	Merged
#683	Fraction Bounce Activity Port	Under Review
#690	Fixed the element selection in password	Merged
#234	Added and fixed some Hindi translations	Merged
#236	Fixed scroll on reordering activities	Merged
#239	Fixed dropbox overflow	Merged
#241	Fixed selected text localization	Under Review
#242	Fixed query retention after language switch	Under Review

Here are the issues I raised:

GitHub Link	Title	Status
#651	Add an option to navigate back to Contents in	Fixed

	Ebook Reader	
#655	Video erase button hidden in Record Activity	Fixed
#661	Text not visible in search bar of Calligra Activity	Fixed
#671	Dialog box has no background in Markdown	Fixed
#673	Markdown link opens in frame which crashes the output	Fixed
#689	Password Tutorial targets the wrong element	Fixed
#235	Activity list doesn't scroll while reordering, causes multiple server calls	Open
#238	Overflow on opening any dropbox	Fixed
#240	Selected option's text isn't localized	Open

I will keep updating this list as I make more contributions.

Other open-source contributions

Vue-event-card

A unique and interactive way to showcase your events with smooth animations and flexible designs. This is an NPM package based on Vue.js.

GitHub: https://github.com/dhruvmisra/vue-event-card

InfoXpression 2019

Created the official website for InfoXpresison which is the annual techno-cultural fest of University School of Information and Communication Technology. Created on Vue.js using multiple other javascript libraries.

Website Link: https://infoxpression.in/

GitHub contributions: https://github.com/techspaceusict/infox19/graphs/contributors

Work Samples

Enfeed

Enfeed is a real-time audience engagement platform for events. The web application features multiple functionalities like Live Q&As, polls, feedback, chat with fellow attendees, etc. I was responsible for creating this product as an intern under Xploryo. I learned to work against deadlines and production level workflows handling multiple users at this firm.

Built on Nuxt.js, a framework based on Vue.js and Google's Firebase for backend.
Website Link: https://www.enfeed.in/

Webapp Link: https://app.enfeed.in/

German Leprosy Relief Association India

GLRA-India in a Non-Profit Organisation which has been serving the cause of leprosy for more than 50 years. This was a freelancing project I took in winters where we had to create the entire website for the NGO displaying the various areas of intervention and porting the existing payment portal.

Built on Vue.js and hosted on cpanel.
Website Link: https://www.glraindia.org/

Grynow

Grynow is an influencer marketing company working with influencers on all major platforms including Youtube, Instagram, TikTok, etc. I worked here as a summer intern and was responsible for creating the main website for the company which showcases their exclusive influencers and the various promotions and case studies. I also worked on some other projects for the founders. I learned about SEO and project management here.

Built on pure HTML, CSS and JavaScript using minimal external libraries.
Website Link: https://www.grynow.in/

Motivation for GSoC

I believe the only constant in my life is the process of learning. I have been fortunate to have had good teachers and mentors in my life and have worked on multiple projects and internships but the knowledge and experience that I would gain from the Google Summer of Code will be unparalleled. It will be one of the most unique learning opportunities with a chance to interact with the most experienced mentors.

Why choose Sugar Labs?

It has been my desire to do something for the society and reform the standard educational ideologies to make for a better future. Sugar Labs, from its core, helps children learn better and faster through their platform. Therefore contributing to and working with Sugar Labs would help me fulfil this goal.

About the Idea

I would be working on the Game Pack which includes the following two activities:

Mind Math

The Mind Math activity will have the following features:

- The student is given five random numbers and should use the four basic arithmetic operations (+, -, ×, ÷) to build an operation that will result in the given output.
- Two levels of difficulty will be proposed:

    - Easy level: Chosen number between 10 and 69
    - Medium level: Chosen number between 0 and 99
- The difficulty level could also be increased by making some operations mandatory.
- The user will be able to undo an operation.
- The score will depend of:
    - Number of slots used (more is better)
    - Bonus when all four arithmetic operations are used
    - Bonus depending on the time spent to solve the challenge
- The game could be played alone or against other users on the network.
    - Single-player mode: Integrate a solver to help the user and show the best result at the end
    - Multiple-players mode: Display a leader board

Tangram

The Tangram activity will have the following features:

- The Tangram activity will present a set of Tangram pieces to the right of the screen and a canvas where the user should form a specific shape using these pieces to the left of the screen.
- Two levels of difficulty could be proposed:
    - Easy level: User knows where each piece should be set on the shape and just have to move/rotate the right piece to the right place
    - Medium level: User should guess where each piece should be set and move/rotate it to the right place
- The difficulty level could also depend on the complexity of the shape.
- The user will be able to undo moves

Implementation

Mind Math

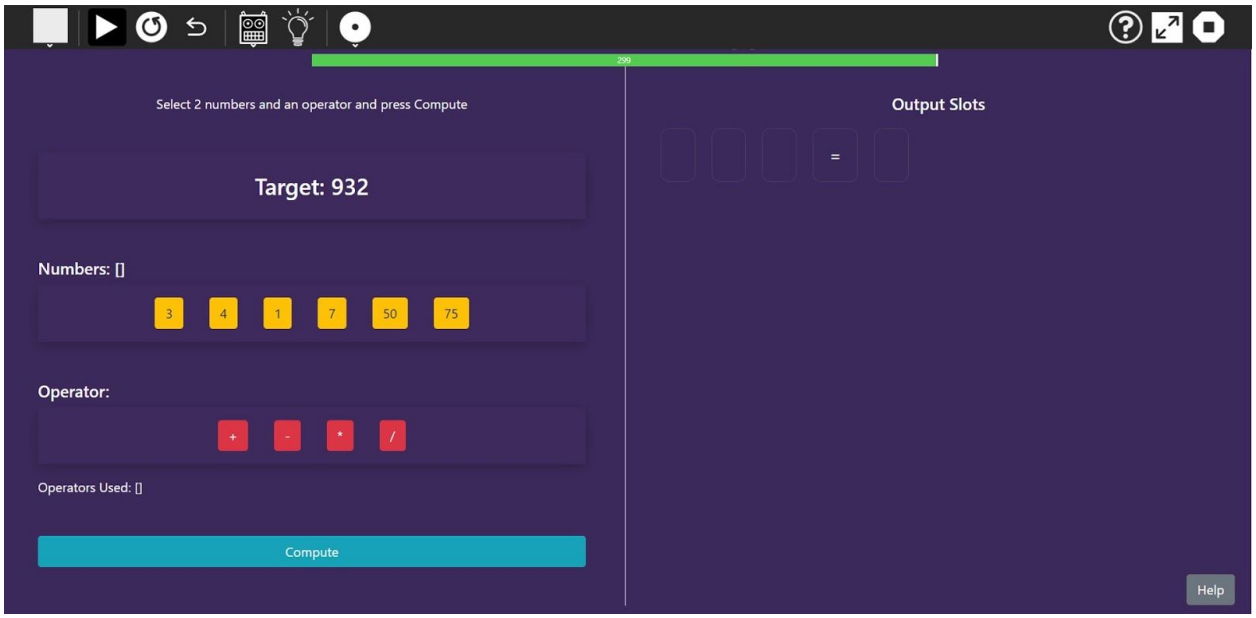
❖ I created a working prototype on Vue.js which can be found here. (GitHub Repository)

Existing solutions for computing the best possible result for the game:

- Cntdn - Uses Depth-First-Search to find the solution to the countdown numbers game. Portable javascript library which can be manipulated to work with any UI.
- Numbers/Numbers-js: An approach using a calculation strategy called Reverse Polish Notation (RPN).

I would suggest taking inspiration from these existing solutions and writing a custom implementation in Vue.js to compute the best possible result.

The main UI of the activity:



Toolbar buttons: Play/Pause, Restart, Undo, Difficulty, Hint, Network

The **hint button** in the toolbar will call the function to compute the best possible result using cntdn.js or numbers-js as previously mentioned. The function will accept the target value and the selectedNumbers and would return an array of strings representing the best possible solution.

The basic workflow of the game can be implemented using multiple arrays:

- Numbers[]: Stores all the randomly generated numbers
- Operators[]: Stores all the allowed operators
- Slots[]: Array of objects storing the information about the current slot

These arrays are displayed dynamically using the v-for attribute provided by Vue.js.

Some state variables:

- target: Stores the randomly generated target
- timer: Stores the time left to play
- status: Saves the current state of the game (running/won/lost)
- currentSlot: Stores the index of the slot to be filled

User **inputs** are stored in the following state members:

- seletedNumbers[]: Store the 2 numbers selected by the user for the current Slot
- selectedOp: Stores the operator selected the user for the current Slot

The mounted() lifecycle hook is used to initialize the timer and other data members as it is fired once the DOM has been mounted to the browser.

The watch property is used to watch changes on seletedNumbers and selectedOp to update the current slot to show the same values.

A stack will be maintained which will store the states of slots for the **undo** functionality.

The score can be computed using the following data members:

- currentSlot+1: Number of slots used to find the answer
- timer: The time left to play
- operatorsUsed[].length: Number of unique operators used to find the answer

Score = (**currentSlot**+1) + (**timer**) + (**operatorsUsed**.length)

This scoring formula takes all the three required parameters into consideration. It can be modified to give other types of bonus. The highscore can be computed after each run of the game.

❖ Exporting the highscore to the **journal** can be accomplished using the default approach by stringifying the context with highscore. The context can be restored on launch of the activity.

Methods to handle the activity:

- select/unselectNumber(): Used to handle click events on numbers

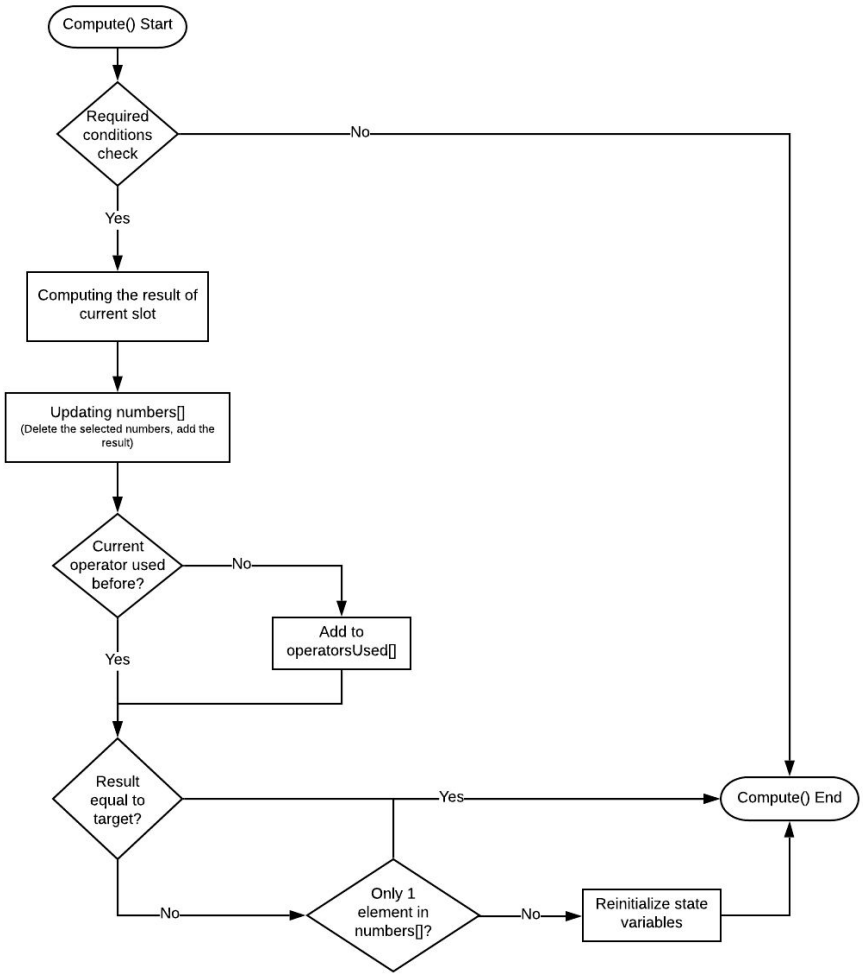
- select/unselectOp(): Used to handle click events on operators
- compute(): Used to compute the result of the current slot and check answer
    - Can be fired using a button or when seletedNumbers contains 2 values, as in Mathador

Here is an implementation of the compute() method:

```
compute() {
      // Checking the required conditions
      if(this.selectedNumbers.length < 2 || this.selectedOp == '') {
        return;
      }

      let num1 = this.selectedNumbers[0],
          num2 = this.selectedNumbers[1]

      // Computing the result
      switch(this.slots[this.currentSlot].op) {
        case '+':
          this.slots[this.currentSlot].res = num1 + num2;
          break;
        case '-':
          this.slots[this.currentSlot].res = num1 - num2;
          break;
        case '*':
          this.slots[this.currentSlot].res = num1 * num2;
          break;
        case '/':
          this.slots[this.currentSlot].res = Math.round((num1 / num2)*100)/100;
          break;
      }

      // Update numbers array
      this.numbers.splice(this.numbers.indexOf(num1), 1);
      this.numbers.splice(this.numbers.indexOf(num2), 1);
      this.numbers.push(this.slots[this.currentSlot].res);

      // Update operatorsUsed
      if(this.operatorsUsed.indexOf(this.slots[this.currentSlot].op) == -1) {
        this.operatorsUsed.push(this.slots[this.currentSlot].op);
      }

      // Checking answer and end of game
      if(this.target == this.slots[this.currentSlot].res) {
        this.status = 'won';
        clearInterval(this.timerInterval);
        return;
      } else if(this.numbers.length == 1) {
```

```
      this.status = 'lost';
      clearInterval(this.timerInterval);
      return;
    }

    // Reinitializing state variables
    this.selectedNumbers = [];
    this.selectedOp = '';
    this.currentSlot++;
    this.slots.push({
      num1: null,
      num2: null,
      op: '',
      res: null
    });
  },
```

Here is a flowchart to sum up the implementation:



A series of v-if and v-else attributes are used to conditionally render Success or Failure messages to the DOM

❖ Network functionality using **Presence** will be added which will also add a leaderboard showing the current/high score of all the connected users.

The following presence actions can be defined:

- init: Synchronize the current leaderboard of the new user with the host
- update: Update the value of current user's score in all leaderboards

Tangram

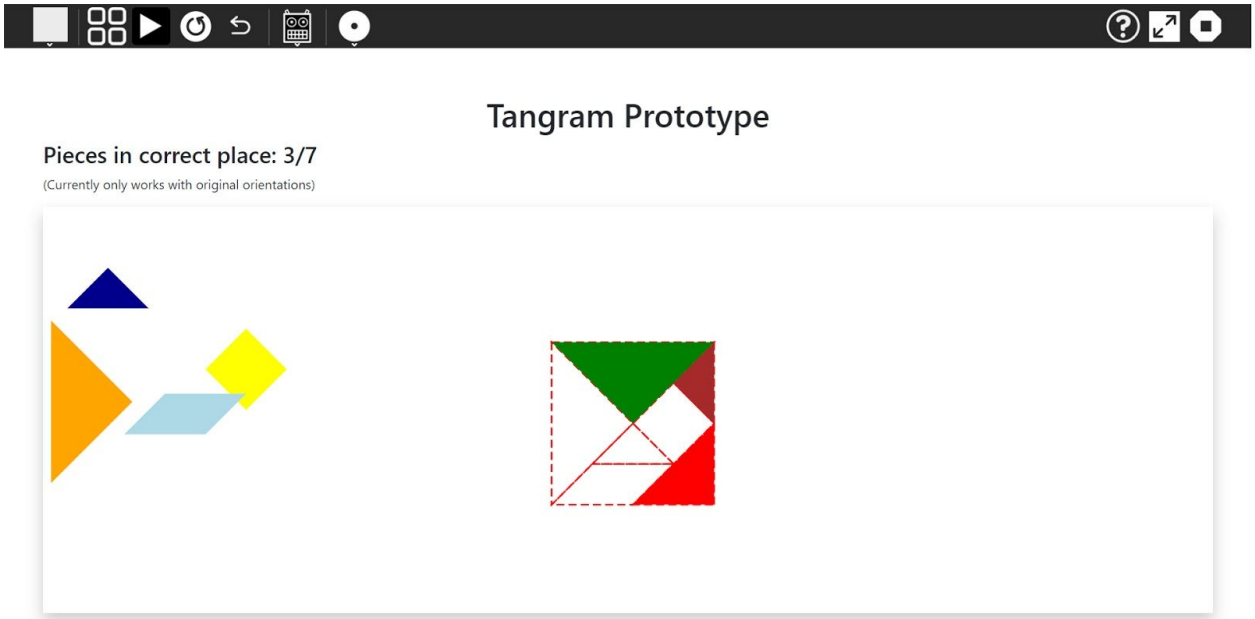
❖ I created a working prototype on Vue.js which can be found here. (GitHub Repository)

Libraries which will potentially be utilised to create this activity:

- Konva.js: HTML5 2D canvas js library for desktop and mobile applications. This is a very powerful library which will be used to create the tangram pieces and shapes.
- Js-clipper: A library which performs clipping and offsetting for both lines and polygons. Will be used to get only the outline of the shape in Medium mode.
- Tangrams: A simple version of the game created using HTML, CSS and JavaScript which can be used for inspiration.

I intend to use Konva.js extensively and take inspirations from other implementations of Tangram available on the internet.

The main UI of the activity:



Toolbar buttons: Levels Menu, Play/Pause, Restart, Undo, Difficulty, Network

The activity will mainly work with the 7 pieces of Tangram. These draggable pieces will be created using Konva lines, specifying the points array for each shape. These will then be added to a Layer and then finally to the Konva Stage.

Sample object in pieces array:

```
{
        name: "LargeRightTriangle1",
        x: 60,
        y: 240,
        offsetX: 50,
        offsetY: 100,
        rotation: 0,
        points: [0, 0, 0, 200, 100, 100],
        fill: "orange",
        closed: true,
        draggable: true
},
```

The shape to be formed can be created using similar piece-sized outlines which again will be constructed using Konva lines with only ‘stroke’ and no ‘fill’ property specified. The ‘x’ and ‘y’ property will determine the position of all these outlines. A **shape** would

therefore be represented by an array of objects containing the relative positions of all 7 piece outlines.

Sample object in the outlines array:

```
{
        name: "LargeRightTriangleOutline1",
        x: 50,
        y: 100,
        offsetX: 50,
        offsetY: 100,
        rotation: 0,
        points: [0, 0, 0, 200, 100, 100],
        stroke: "red",
        strokeWidth: 2,
        closed: true,
        draggable: false,
        dash: [10, 5]
},
```

- A **Director mode** can be added which will allow the creation of custom shapes which will be saved in the journal and can be shared with other users on the network. This can be implemented by allowing ‘draggable’ property on outlines. Then after a series of placement and rotation operations, the final result will be saved as the array of objects having all relative positions and rotations.

  The mounted() lifecycle hook will initialize the timer and Konva outline placements.

  The users will drag the pieces onto the shape outline. When a piece is dragged to near its matching outline, it will snap to the outline, signifying the correct position of the piece. For the Medium difficulty, we can use js-clipper to clip and get only the outer lines of the shapes. This would make it relatively tougher to complete the shape.

  A stack will be maintained which will store the states of ‘x’, ‘y’ and ‘rotation’ of each piece in the pieces array for the **undo** functionality.

  A **Score** parameter should be created which can be calculated using the timer, i.e., the amount of time remaining to complete a level. The score will be saved per level and will be used while playing against others on the network.

- Progress will be exported to the **journal** which will save the levels completed along with their scores as an array of objects. Along with the level information, the custom shapes

created in Director mode will also be stored in the journal. This will be stringified and stored as JSON. This context will be restored at the launch of activity.

- Network functionality using **Presence** will be added which will also add a leaderboard showing the score of all the connected users in the selected level.

  The following presence actions can be defined:

  - init: Synchronize the current leaderboard of the new user with the host
  - selectLevel: Synchronize a level across all connected users
  - update: Update the value of current user's score in all leaderboards

How will it impact Sugar Labs?

Game-based learning plays an important role in teaching as it allows students to interact, collaborate and develop critical thinking skills while at the same time be entertained. Addition of these two activities would prove to be an enhancement to the learning and practicing experience and would also help in increasing student engagement with the platform.

The Mind Math activity would help students improve the speed and accuracy of calculations of all basic mathematical operations. This shall prove to be helpful in all STEM fields and non-technical fields as well such as Accounting, hence making this platform essential for all fields of study.

The Tangram activity invokes pattern finding and puzzle solving abilities in students which helps raise general aptitude. All these qualities would make students better problem solvers and allow them to find out-of-the-box solutions.

Availability during GSoC Project

My semester will end on **May 27** therefore I will need a break for 9 days which I will cover up by putting in extra work hours everyday after my exams.

I will devote at least 40-50 hours a week on my project and can extend my involvement if the need persists. My aim is to stick to the timeline and I'm confident that the project will be completed in a timely manner.

I will be committed to this project throughout my GSoC tenure as I have no other internships or projects to be distracted by.

Timeline

Time Period	Task
28 April - 18 May	Exploring various technologies and libraries. Discussing the implementation approaches with mentors and the community. Development environment setup and structuring the project.
18 May - 27 May	Break for semester examinations. Staying in touch with mentors to finalize the feature list.
27 May - 2 June	Start creating the Mind Math activity. Finalize UI design, icons and colours. Implement basic workflow. Create a simple functional single-player game. Store and retrieve context from the Journal.
2 June - 8 June	Study various implementations for finding solutions. Write a custom function library to find the best solution for Mind Math. Integrate the library with the activity through the toolbar icon.
8 June - 14 June	Implement the undo stack and add restart functionalities. Add the different modes of difficulty after discussing with mentors. Add basic leaderboard and Presence support to

	share activity with others.
15 June - 19 June	**Phase 1 Evaluations** (Progress: Completely functional single-player mode, hint button implementation, difficulty modes, basic leaderboard and network functionality)
20 June - 26 June	Finalize the leaderboard and add a tutorial for the activity. Make the required changes after evaluation. Test activity on different platforms. Start working on the Tangram activity. Study more about Konva.js.
27 June - 3 July	Create the basic drag and drop functionality on a shape. Find shapes created using Tangram and create different levels. Add Journal support to store levels completed and scores.
4 July - 12 July	Add undo stack and other toolbar functionalities. Add Director Mode to create new shapes. Add Journal support to store and retrieve custom shapes.
13 July - 17 July	**Phase 2 Evaluations** (Progress: Mind Math fully functional, basic implementation of Tangram, levels' grid functional, Director mode implementation, journal support)
18 July - 24 July	Add js-clipper to show only outer lines for Medium level difficulty. Add network sharing of the activity using Presence and creating the leaderboard.

25 July - 31 July	Finalize the leaderboard implementations. Discuss with mentors and add functionality to change levels and share custom shapes with the network.
1 Aug - 9 Aug	Add a tutorial to the activity. Fix any bugs in both activities and perform various types of unit and integration testing.
10 Aug - 17 Aug	Buffer week to enhance the look and feel of the activities along with beta testing.
17 Aug - 24 Aug	**Final Evaluations** (Progress: Medium difficulty for Tangram, Network sharing and leaderboard, shape sharing over the network, both activities completed)

Plans post GSoC period

After my tenure at GSoC, I will continue to maintain the activities and fix any bugs that show up. I would also continue to work as an active community member and help new developers and contributors get familiar with the platform. I hope to keep working towards the growth of this organisation.

At last, I should mention that it has been a great learning experience for me while contributing to Sugar Labs and the healthy community really helped me work my way through the platform to utilize my skills. I am very enthusiastic towards working with Sugar Labs in the Google Summer of Code 2020 and make contributions to help the community even further. I am determined to make this project successful.

Thank You,
Dhruv Misra
Undergraduate, Third Year
Guru Gobind Singh Indraprastha University, New Delhi