Sugarizer game activity pack

Basic Details

Name	**:**	Sarthak Gupta
Email	**:**	sarthakgupta041@gmail.com
Github	**:**	sarthak-g
IRC	**:**	sarthak041
linkedIn	**:**	https://www.linkedin.com/in/sarthak-gupta-4080b8171/
Degree	**:**	B.tech in Information Technology
Institute	**:**	Bharati Vidyapeeth's College Of Engineering
Language	**:**	English, Hindi(Native)
Location	**:**	New Delhi, India
Timezone	**:**	My timezone is Indian Standard Time (UTC + 5:30).

What is your motivation to take part in GSOC?

GSOC provides an opportunity to interact and become part of a community whose source code is used by a large number of people worldwide. It gives an immense learning experience.

Why did you choose Sugar Labs?

Sugar Labs enhances the education system by teaching concepts in a fun and engaging manner.
It is an awesome feeling that the code I write will be part of the children learning process.
Also, it will improve my perspective of solving a problem since the software is used by children and every little thing has to be taken into consideration so that they do not face any difficulty.

Other open source contributions

- **GCI 2019-20 Mentor of Tensorflow**
  I had interacted with students of age 13-17, solved their doubts and reviewed their pull request.
  certificate

- One of the admin of college open source community ACM BVP. Reviewed and merged pull requests of first time contributors as part of hacktoberfest.
  Link: reviewed and merged pr's

I have been contributing to open source for more than 1.5 years and have created several interesting projects that can be seen on my Github account

Awards

- Won Ultrahack hackathon
  Got an opportunity to represent India in the Galileo Innovation Challenge, Helsinki.
- Matic prize winner of Hack.IT@BVP (24hr hackathon)
- Won Ideahack (30 hr hackathon) organized on HackerEarth.

Skills: HTML,CSS,JS,Python,C++,Vue.js,Node.js,django,Unity 3D, Docker.

Work Samples

- **Arzara**
  Augmented reality app to see how a particular watch model looks on hand. Designed for the use by offline watch showrooms to prevent degrading of polish of watches due to repeated wear by different customers. It has 3 watch models to give an idea of how it works. Each showroom has to integrate 3d models of watch present in their stock to use this app.
  Link: https://github.com/sarthak-g/Arzara

- **FlexERP**
  Developed ERP software from scratch for Flexmotiv Technologies which is further extended to use by all startups(around 20) in India's premier institute IIT Delhi.
  Link: https://github.com/sarthak-g/flexmotiv

- **Infroid**
  Designed intern management and task assigner software for company name Infroid.
  Link: https://github.com/sarthak-g/infroid

Sugarizer Contributions

I am contributing to sugarizer from mid-November 2019. I've contributed to various activities therefore I'm well-versed with coding pattern and how focus should be on every little thing such that children do not face any difficulty.

Open pull request:

- New chess activity

Merged pull requests:

- Corrected wrong title of Network button
- Shared Notes tutorial error
- Overflow of RGB bar in LabyrinthJS and Write activity
- Overflow of color bar in ColorMyWorld activity
- Font size automatically reduces in Fototoon Activity
- Improvement of text palette of Fototoon
- Functionality of "clean all" button of Fototoon
- Error in resize function of TankOp Activity
- New resizing algo for TankOp
- Fullscreen and Unfullscreen functionality of XO Editor
- Fullscreen and Unfullscreen functionality of Gears Activity
- Clear icon of GameOfLife activity

Issues raised:

- Wrong arrow icon in Exerciser activity
- Wrong title of network button in Memorize activity

- **Error in Shared Notes activity when open through sugarizer intro tutorial**
- **RGB bars overflow in LabyrinthJS and Write activity**
- **RGB bars overflow in ColorMyWorld activity**
- **Reduction in font size of Fototoon activity on changing slides**
- **Activities work only in landscape mode**

Interaction with sugar labs GCI student/fellow contributors:

- **https://github.com/llaske/sugarizer/issues/688**
- **https://github.com/llaske/sugarizer/issues/549**
- **https://github.com/llaske/sugarizer/pull/552**

You can check my all contributions here: pull request, issues

Project Details

Name of project: Sugarizer game activity pack.

Aim of the project :

The aim of this project is to develop two new Sugarizer activities needed by teachers from Sugarizer deployment in Saint-Ouen.

- Mind Math Activity
- Tangram Activity

Techstack:

I will implement both activities using Vue.js.
I've implemented a template for sugarizer activities using Vue.js with tutorial, localization and fullscreen mode.You can check it here.

Mind Math Activity :

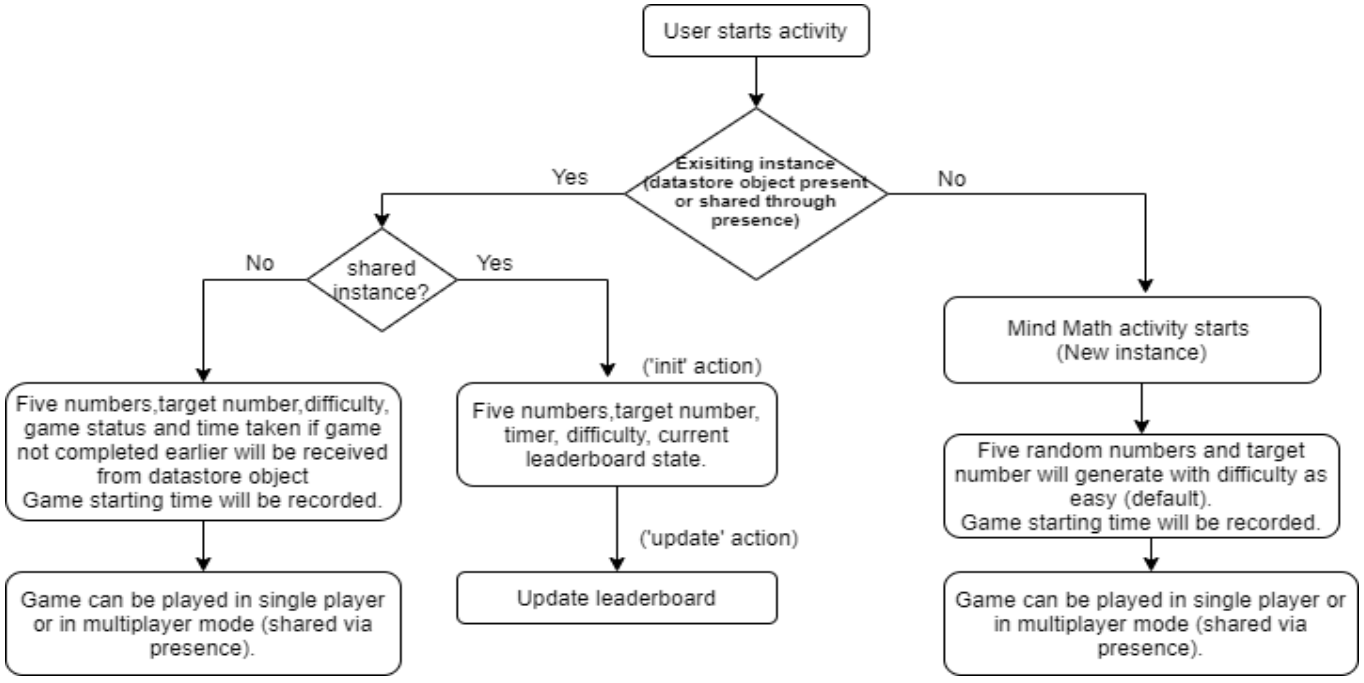
The Mind Math activity will be a game to practice mathematics in a different way. The student will be given five random numbers and they have to form a target number using four basic arithmetic operators (+, -, ×, ÷).
The activity will have following features :

- There will be three levels of difficulty: easy, medium and hard.
    - An easy level will have a target number between 10 and 69.
    - A medium level will have a target number between 0 and 99.
    - A hard level will have a target number between 0 and 99 with some mandatory operations.
- The user will have five random numbers (one between 1-4, one between 1-6, one between 1-8, one between 1-12 and one between 1-20) and 4 slots to find the target number.
- The score will depend of:
    - Number of slots used. Higher the slots more will be the score
    - Bonus when all four arithmetic operations are used.
    - Bonus depending on the time spent to solve the challenge.
- There will be two modes: single player and multiplayer.
    - In single player mode, a solver will be integrated to help the user and show the best result at the end.
    - In multiplayer mode, a leaderboard will be displayed.

- Additional buttons in single player mode:
    - Undo button to undo an operation
    - Restart button to restart the game. On pressing the restart button, an alert will popup which will be designed using picoModal(js library used by sugarizer to create popups).
    - Difficulty button to change the difficulty of the game.
- There will be use of Sugar toolbar and palette.
- All the icons used will be standard sugarizer icons(used across different sugarizer activities). The UI of the activity will be simple, self-explanatory and respect Sugar standards.
- The activity will be integrated with Journal (save its context in the Journal).
- There will be integration of Sugarizer presence to share the activity on the network so that multiple users could play together.
- The activity will be fully localized like other sugarizer activities using webl10n library.
- User will be able to enter full screen mode on clicking the full screen button.
- The activity will include a tutorial to explain the UI which will be executed by pressing the help button.
- The activity will be responsive (will work on any screen size).
- Activity will work on any browser (Chrome, Firefox, Safari) and any platform (Android, iOS, Windows, Linux, MacOS) supported by Sugarizer
- The activity will not depend on Internet connection (will use local content only).

Implementation:

Flow of activity is given below:



Number Generation:

- Five random number will be generated as shown below:

```
this.num_1 = Math.floor((Math.random() * 4) + 1); //Number will be between 1-4
this.num_2 = Math.floor((Math.random() * 6) + 1); //Number will be between 1-6
this.num_3 = Math.floor((Math.random() * 8) + 1); //Number will be between 1-8
this.num_4 = Math.floor((Math.random() * 12) + 1); //Number will be between 1-12
this.num_5 = Math.floor((Math.random() * 20) + 1); //Number will be between 1-20
```

- Target number:
    - I will generate a target number based on above 5 random numbers so that there will not be a case when the target number can't be reached using given numbers

Slots:

- Initial slots array will look like below:

```
this.slots = [
    [-1,'',-1,-1], //slot1
    [-1,'',-1,-1], //slot2
    [-1,'',-1,-1], //slot3
    [-1,'',-1,-1], //slot4
]
```

- Slot no is equal to index of array (eg: slot 1 is equal to slots[0])
- For each slot:
    - index 0 (eg: slots[0][0]) is number before operator
    - index 1 (eg: slots[0][1]) is operator used
    - index 2 (eg: slots[0][2]) is number after operator
    - index 3 (eg: slots[0][3]) is the result of operation. It can't be negative or in fraction.

State variable:

gameComplete	Boolean variable which keeps track whether the game is completed or not.
startTime	It will store time whenever the game starts.
endTime	It will store time whenever the game ends.Initially it will be 0.

- startTime will reinitialize(change its value) and endTime will become 0 whenever gameComplete changes from true to false i.e. the game is continued in the same instance after winning/losing.
- For host, startTime and endTime will reinitialize when it shares the activity.
- For connected users, startTime and endTime will reinitialize when it joins the activity.

Score calculation:
By considering points in GSOC idea description and taking inspiration from Mathador score methodology, I will calculate score as below:

- If a target number is found, all 4 slots are used and each slot has a different operator then the user will be awarded 18 points.
- If all slots are **not** used but the target number is found then:
    - 5 points for finding the target number
    - 1 point for each slot
    - 1 point for 'x' or '+' operator
    - 2 points for '-' operator
    - 3 points for '/' operator
- No score will be allotted if the target number is not found.
- There will be **no** timer feature. By timer feature, I mean automatically losing the game after a specific time period ends. However, time taken to solve will be calculated.
- Time taken to solve will be calculated using state variables startTime and endTime.
  time taken to solve = (endTime - startTime)
- Score will also depend on time taken to solve:
    - If the time taken is less than 5 minutes then 2 points will be added to the initial score.
      i.e. Final score = earlier score + 2
    - If the time taken is greater than 5 minutes and less than 10 minutes then 1 point will be added to the initial score.
      i.e. Final score = earlier score + 1
    - Otherwise , **Final score = earlier score**

Solver for one-player mode:

- From many of the existing solutions, I've found below two pretty useful:
    - Mathador cheater (licensed python code for generating all possible solutions with points according to mathador scores)
    - Cntdn (free open source js library)

- Mathador cheater give accurate results but it takes a lot of time to calculate.
- Cntdn consist of two types of number solvers:
    - Normal solver
    - Trickshot
- I will use the Trickshot version as the normal solver fails to solve when the same number as target number is present in the input.
- Things I will improve in trickshot version:
    - The solution from it occupies 3 slots in answer. I will improve it's algo to show 4 slots by making improvements in function "_recurse_solve_numbers" and "tidyup_result"
    - Currently, it gives only operator based preference. I will update code to give both operator and slot based preference.
    - From all possible solutions, preference will be given according to the above score calculation methodology.

Datastore object:

- For existing instance, datastore object will look like below:

```
{
  "gameContinue": ... ,   // value of gameComplete(true/false)
  "timeTaken":     ...,   // time taken (endTime - startTime)
  "difficulty":    ...,   // 0 for easy, 1 for medium, 2 for hard
  "number1":       ...,
  "number2":       ...,
  "number3":       ...,
  "number4":       ...,
  "number5":       ...,
  "target":        ...,
  "slots":         ...,   // slots array
  "score":         ...,
}
```

- For existing instance :

Time taken to solve = (time taken from data store object) + (time taken of current instance)

Shared Activity:

- Network functionality will be implemented using presence.

- ‘init’ method in presence will initialise 6 numbers(1 target and 5 others), difficulty, timer and current leaderboard state which then also call ‘update’ action to update leaderboard of all users (as new user joined).
- ‘Update’ method will update the leaderboard of every connected user. This action will be called whenever there is change in no. of users and score.
- Leaderboard will display rank, xo icon,name and score.
- For the leaderboard, array will be maintained. This array will contain objects equal to no. of users.
    - Each object will look like below:

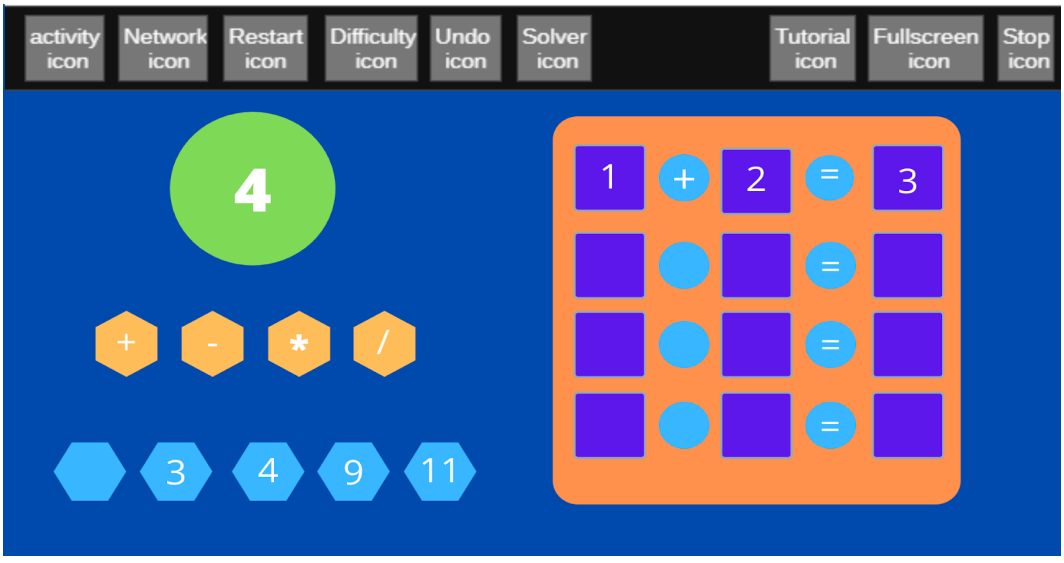
```
{
   "userInfo": ... ,  // user object obtained from looping inside
                      // “listSharedActivityUsers” of presence
   "score":    ... ,
}
```

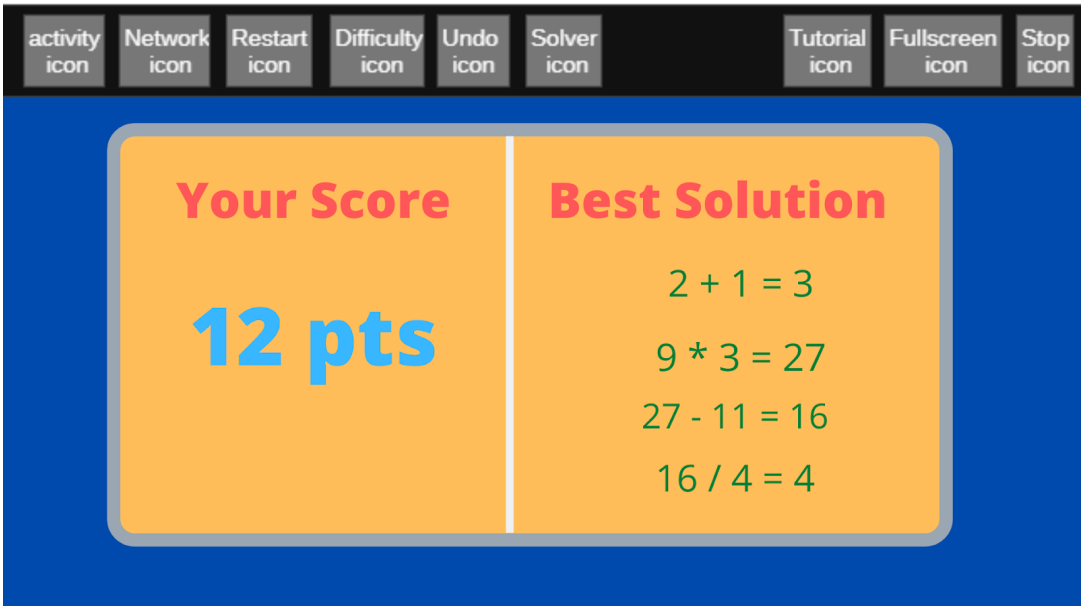
- Array will be sorted according to “score” in object
    - If score is different then:
        - Rank of user = index of object in array + 1
    - If score of users is same then their rank will also be same:
        - Rank of user = index of first occurence of that score in array + 1

Proposed UI:

- Background color will be the same as buddy color.
- UI will be colourful and as simple as possible.
- I’m planning to give the activity UI like below.
- Note: This is not a fixed UI. I will modify it as per mentor’s suggestion.

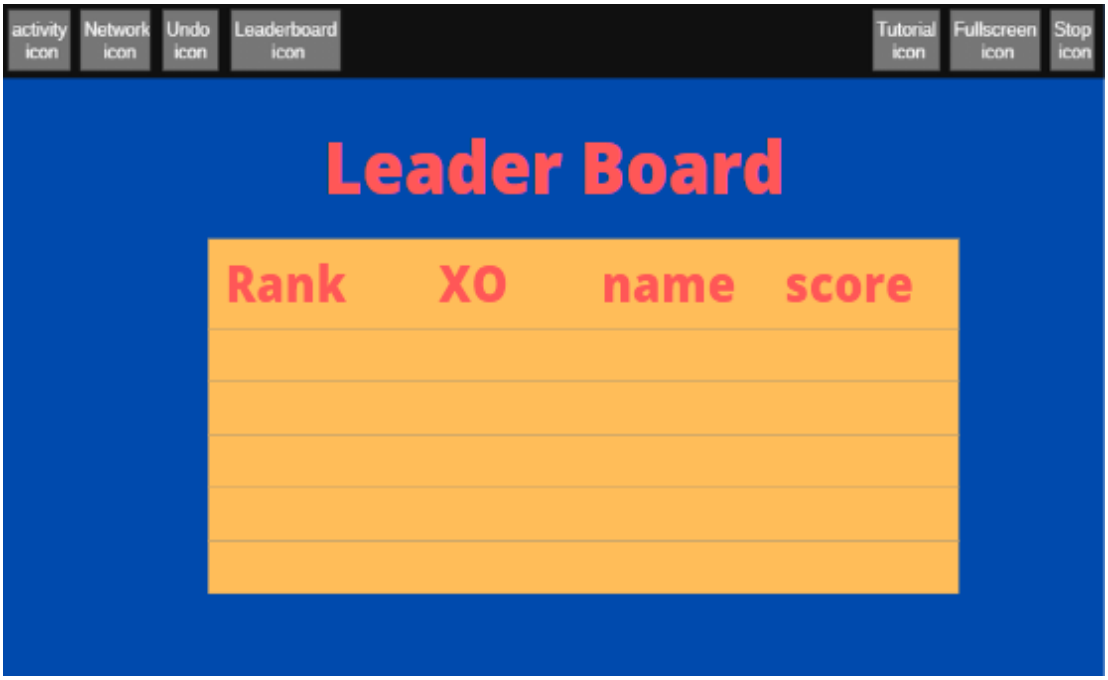
One-player mode:





Multiplayer mode:

- Screen in multiplayer will be the same as the first image except restart, difficulty and solver button.
- There will be a button to check leaderboard.

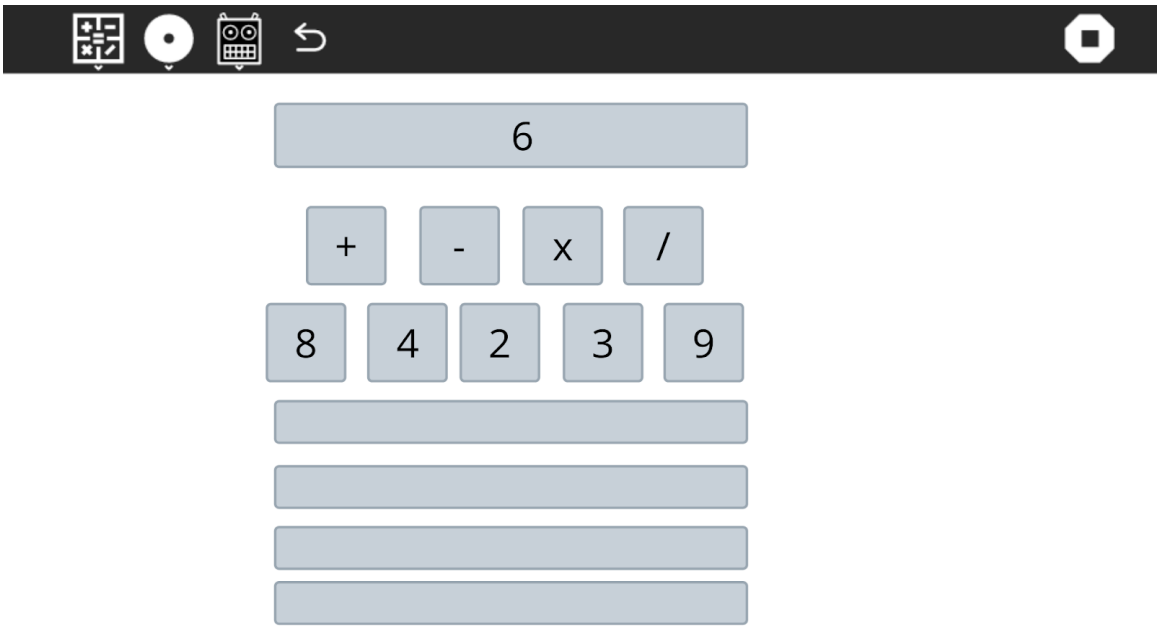


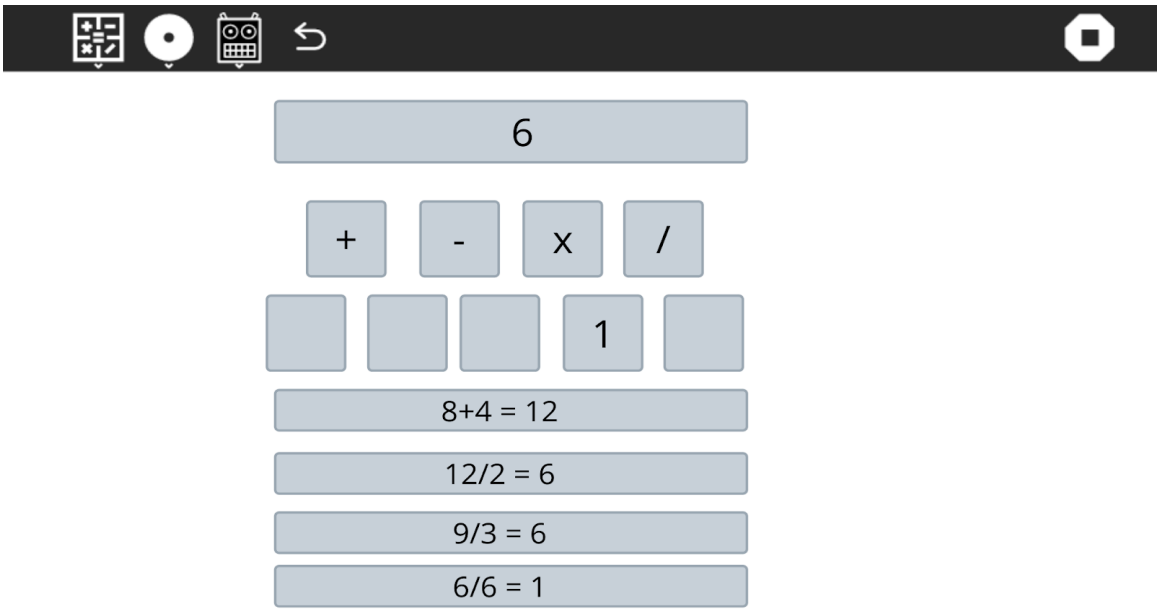
Responsive:

- I will use bootstrap to make activity responsive.

Basic prototype in sugarizer:

- I’ve implemented a basic prototype for mind math activity in sugarizer. All buttons are not present in this prototype.





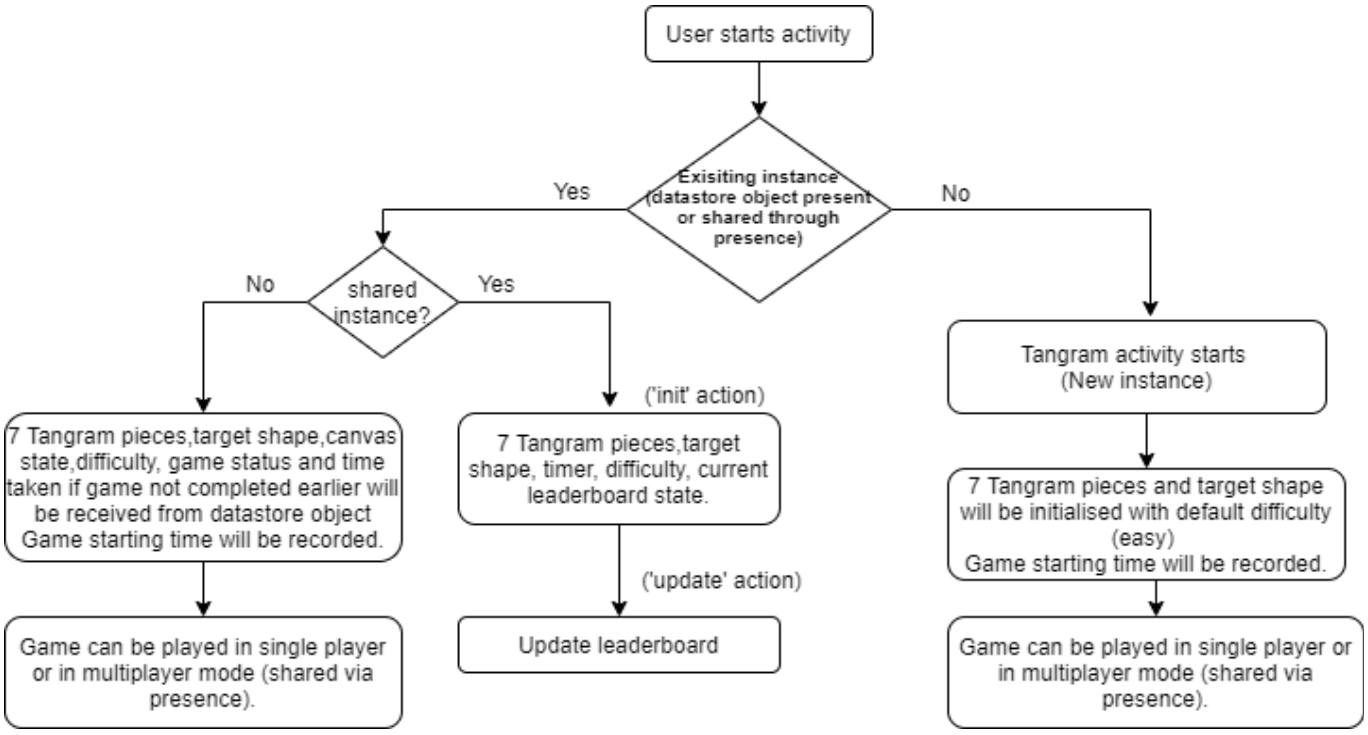
Tangram Activity:

The Tangram activity will be an activity to play the traditional Tangram game.
Activity will have following features:

- There will be a set of 7 tangram pieces, a target shape to be formed and a canvas to make the puzzle.
- There will be two levels of difficulty, easy and medium.
- In an easy level, the user will know where each piece should be set on the shape and just have to move/rotate the right piece to the right place.
- In a medium level, the user will guess where each piece should be set and move/rotate it to the right place.
- Difficulty level will also depend on the complexity of the shape.
- There will be two modes: single player and multiplayer.
    - In single player mode, there will be no timer but the time taken to solve the puzzle will be calculated.
    - In multiplayer mode, there will be a common timer visible to all the connected users. The user who solves the maximum number of puzzles in that time period will win the game. A leaderboard will be visible throughout the game.
- Additional buttons in single player mode:
    - Restart button to restart the game. On pressing the restart button, an alert will popup.
    - Difficulty button to change the difficulty of the game.
    - Arrows button to change the puzzle within the same difficulty level.
- There will be use of Sugar toolbar and palette.
- All the icons used will be standard sugarizer icons(used across different sugarizer activities). The UI of the activity will be simple, self-explanatory and respect Sugar standards.
- The activity will be integrated with Journal (save its context in the Journal).
- There will be integration of Sugarizer presence to share the activity on the network so that multiple users could play together.
- The activity will be fully localized like other sugarizer activities using webl10n library.
- User will be able to enter full screen mode on clicking the full screen button.
- The activity will include a tutorial to explain the UI which will be executed by pressing the help button.
- The activity will be responsive (will work on any screen size).
- Activity will work on any browser (Chrome, Firefox, Safari) and any platform (Android, iOS, Windows, Linux, MacOS) supported by Sugarizer
- The activity will not depend on Internet connection (will use local content only).

Implementation:

Flow of activity is given below:



We can implement this activity using three ways:

- Using jQuery and CSS.
- Using EaselJS:
  EaselJS is part of the CreateJS (used in Fototoon activity) suite, a JavaScript library for building rich and interactive experiences, such as web applications and web-based games that run on desktop and mobile web browsers.
- Using Konva.js:
  Konva.js is an HTML5 Canvas JavaScript framework that extends the 2d context by enabling canvas interactivity for desktop and mobile applications.Konva enables high performance animations, transitions, node nesting, layering, filtering,caching, event handling for desktop and mobile applications, and much more.

I will implement a game using Konva.js as it gives much more flexibility and will help in handling pieces and canvas in multiplayer mode. It also works pretty well with Vue.

Explanation about implementation using Konva:

- Konva stage will be initialised with width and height equal to activity canvas. New layer will be initialised.

```
this.stage = new Konva.Stage({
  container: 'container',
  width: width,
  height: height
});
this.layer = new Konva.Layer();
```

- Pieces or tangs will be made using Konva shapes or polygons..
- Sample of initialisation of shape(square) is shown below:

```
this.square = new Konva.RegularPolygon({
  x: 60,
  y: 60,
  sides: 4,
  radius: 50,
  fill: "rgba(200,0,0, 0.5)",
  stroke: "black",
  rotation: 0,
});
```

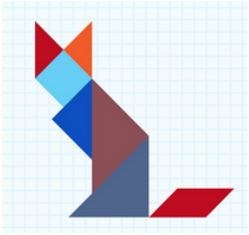
- All pieces will be added to the Konva layer.
- Each piece will contain following mouse events:
    - Click(to rotate the shape on mouse click)
    - dragstart
    - dragmove
    - dragend

Inspiration:

- I will take inspiration from below resources:
    - aniamika/Tangram
    - serkankayaa/konvajs-puzzle
    - shgalus/tangram

Easy level:

- In an easy level, there will be simple target shapes.
- Tangram pieces will be of a different color(as in below image) so that children can easily identify the correct position of each piece from target shape.



Medium Level:

- In a medium level, there will be shapes more complex than easy level.
- Tangram pieces will be black or of the same color and there will be no outline of pieces in target shape (as shown in below image).

State variable:

gameComplete	Boolean variable which keeps track whether the game is completed or not.
startTime	It will store time whenever the game starts.
endTime	It will store time whenever the game ends.Initially it will be 0.

- startTime will reinitialize(change its value) and endTime will become 0 whenever gameComplete changes from true to false i.e. the game is continued in the same instance after winning/losing.
- For host, startTime and endTime will reinitialize when it shares the activity.
- For connected users, startTime and endTime will reinitialize when it joins the activity.

Datastore object:

For existing instance, datastore object will look like below:

```
{
   "gameContinue":          ... ,   // value of gameComplete(true/false)
   "timeTaken":             ...,    // time taken (endTime - startTime)
   "difficulty":            ...,    // 0 for easy, 1 for medium
   "puzzles_solved":        ...,
   "pieces_left": [                  //array containing objects of pieces which are not
                                     // placed to their correct place
                    {
                      ...            //object containing properties of particular piece
                    },
                    ...
            ],
   "pieces_correctly_placed": [      //array containing objects of pieces which are
                                     // placed to their correct  place
                    {
                      ...            //object containing properties of particular piece
                    },
                    ...
            ],
   "target_shape":{
                    ...             // object containing properties of target shape
                  },
}
```

- For existing instance :

  Time taken to solve = (time taken from data store object) + (time taken of current instance)

Score calculation in multiplayer mode:

- Score will be equal to the number of puzzles solved in a given time interval.

Shared Activity:

- Network functionality will be implemented using presence.
- ‘init’ method in presence will initialise 7 tangs,target shape,timer,difficulty and current leaderboard state which then also call ‘update’ action to update leaderboard of all users (as new user joined).
- ‘Update’ method will update the leaderboard of every connected user. This action will be called whenever there is change in no. of users and score(= no. of puzzles solved).

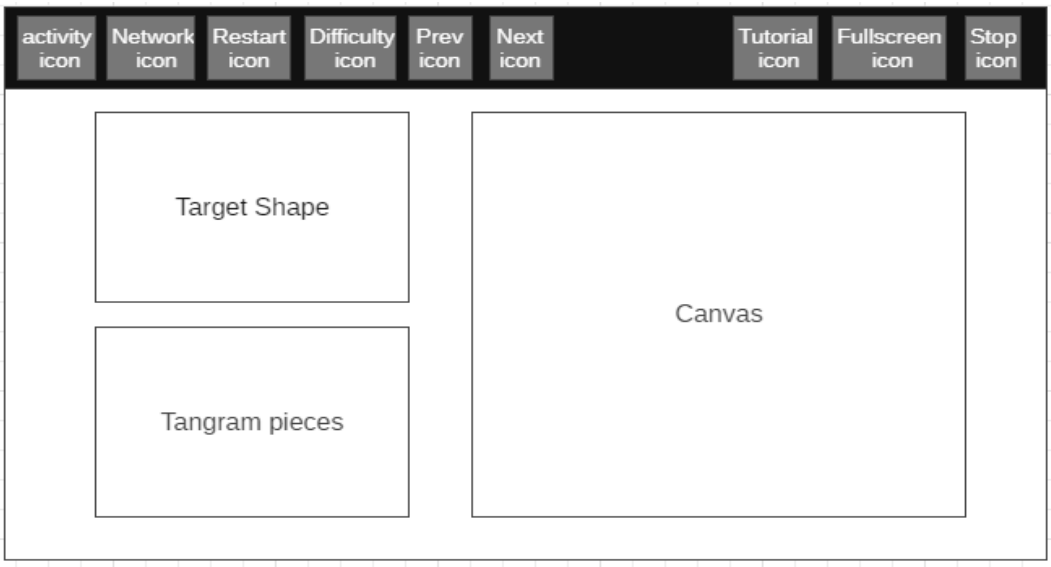
- Leaderboard will display rank, xo icon,name and score.
- For the leaderboard, array will be maintained. This array will contain objects equal to no. of users.
    - Each object will look like below:

```
{
   "userInfo": ... ,  // user object obtained from looping inside
                      // "listSharedActivityUsers" of presence
   "score":    ... ,
}
```

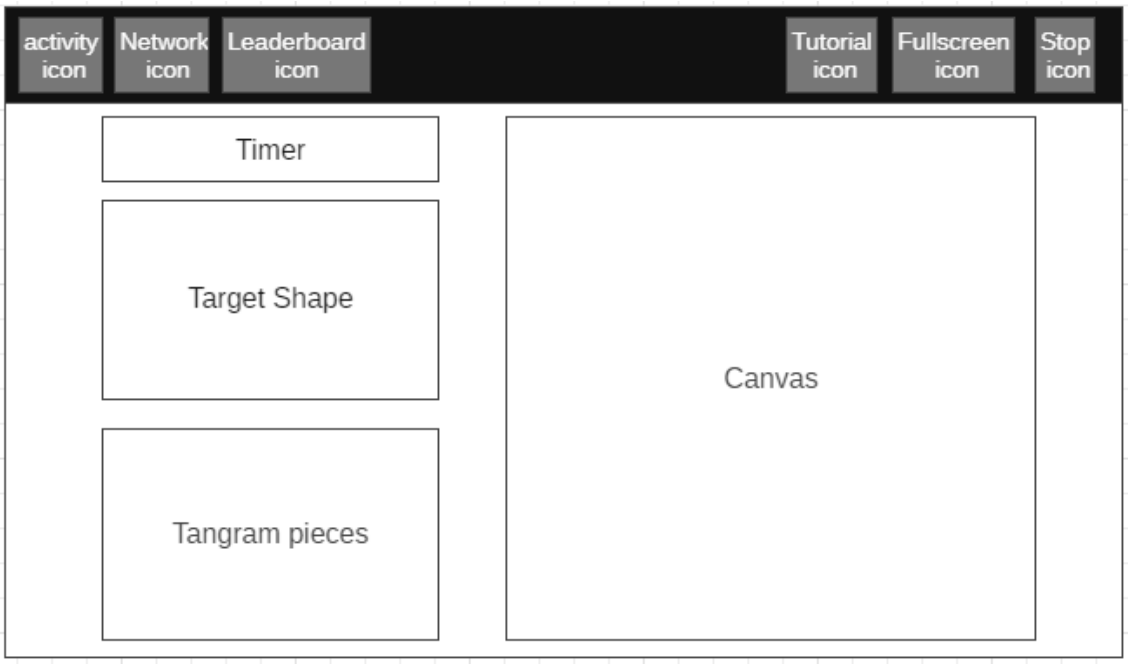
- Array will be sorted according to "score" in object
    - If score is different then:
        - Rank of user = index of object in array + 1
    - If score of users is same then their rank will also be same:
        - Rank of user = index of first occurence of that score in array + 1

UI wireframes

One-player mode:



Multiplayer mode:



Leaderboard:

- leaderboard will be similar to Mind Math activity.

Responsive:

- I will use bootstrap to make activity responsive.

What technologies and dependencies will you be using ?

- For Mind Math Activity, I will use:
  - Vue.js
  - Cntdn
  - Bootstrap
- For Tangram Activity, I will use:
  - Vue.js
  - Konva
  - Bootstrap

How will it impact Sugar Labs ?

Both the activities in this project will help children to develop new skills.

Mind Math activity will improve children's algebra and speed of mental math calculation. Children have to work on the calculation, develop strategies and make links between the different operations. This game will add a social dimension to the practice of mathematics.

Tangram activity will improve logical thinking and problem-solving skills. It is also believed that tangram improves many mathematical concepts such as congruency, symmetry, area, perimeter, and geometry. This game will help children to build a strong foundation.

Teaching important concepts using games will not bore the children and will engage them.

Timeline:

Break down the entire projects into chunks and tell us what will you work on each week.As the summer goes on, you and your mentor will adjust your schedule, but it's good to have a plan at the beginning so you have an idea of where you're headed.

DURATION	TASK
4 May - 1 June	**Community Bonding Period** ● Communicate with my mentors (Lionel and Ashish) and other org members to finalize the features and UI of both the activities. ● Discuss implementation approaches. ● End term exams in the last week of may.
1 June - 8 June	● Extending Vue.js template for Mind Math activity. ● Start working on single player mode for new instance ● Implement difficulty palette, number generation,slots filling and score calculation logic
8 June - 15 June	● Restart and undo functionality. ● Store and retrieve context from the journal. ● Start working on integration of solver
15 June - 22 June	● Complete solver ● Start implementing multiplayer mode using presence.

	○ Sharing activity with user join/leave notification ○ Init and update actions ● Displaying and maintaining Leaderboard for multiplayer.
22 June - 29 June	● Complete multiplayer mode. ● fullscreen/ unfullscreen mode ● Integrate tutorial ● UI(and responsiveness) improvement ● Implement text localization. ● Test across different browsers and devices.
29 June - 3 July	**Phase 1 evaluation** Progress**:** ● Completion of single player and multiplayer mode of Mind Math activity in sugarizer. ● Testing of activity on different devices and browsers ● Tutorial and fullscreen mode integration.
3 July - 10 July	● Implement changes suggested during phase 1 evaluation ● Test the changes made ● Complete documentation of Mind Math Activity. ● Extending Vue.js template for Tangram activity. ● Start working on single player mode for new instance
10 July - 17 July	● Start making tangs/piece and initialise piece properties in object ● Target shape set according to difficulty. ● Implement click,dragstart,dragmove and dragend mouse events on each piece.
17 July - 22 July	● Complete both easy and medium level for single player ● Game Restart functionality ● Store and retrieve data from journal.
22 July - 27 July	● Test functionality of single player mode. ● Start implementing multiplayer mode using presence. ● Displaying and maintaining Leaderboard for multiplayer.
27 July - 31 July	**Phase 2 evaluation** Progress**:**

	● Completion of Mind Math activity. ● Completion of single player mode of Tangram activity. ● 90-95% completion of multiplayer mode of Tangram activity.
1 August - 7 August	● Make improvements suggested during phase 2 evaluation. ● Complete multiplayer mode. ● Test functionality of multiplayer mode. ● Implement fullscreen mode. ● Integrate tutorial.
7 August - 14 August	● UI(and responsiveness) improvement. ● Implement text localization. ● Test across different browsers and devices. ● Complete documentation of Tangram Activity.
14 August - 21 August	● Take feedback from mentors and community for both the activities. ● Make changes according to their feedback. ● Test both the activities again after making changes.
21 August - 24 August	● Update documentation of both activities for the changes done. ● Prepare for final evaluations.
24 August - 31 August	**Final evaluation** Progress: ● Completion of both the activities with documentation and thorough testing.

I will be needing a break for about 9 days for my End semester examination. Earlier it was scheduled on 20th of may but it will reschedule due to covid-19 pandemic situation. I will cover up for this break by working for some extra hours after exams.

How many hours will you spend each week on your project ?
I am planning to work from 5:00 to 16:00 (UTC) but my timings are flexible. I will easily give 48-50 hours a week as I do not have any other commitments. My main aim would be to stick by the timeline and perform all the mentioned tasks efficiently.

How will you report progress between evaluations ?
I will make regular pull requests to sugarizer, so anyone in the org can view my progress. I will also maintain a medium blog to share the challenges I came across, how

I found a solution for them and about the overall update of my project. For problems or questions, I will be using the mailing list as the community is very active there and will keep in touch with my mentors through email also.

Discuss your post GSoC plans. Will you continue contributing to Sugar Labs after GSOC ends ?

I will continue to work on this project after GSOC as an active contributor. I will entertain all the issues of new/old contributors regarding this project and update this project with time.

I always want to work towards the growth of this organization and wish to apply as a mentor for further programs(GCI and GSOC).

Screenshot of chess activity