



Google Summer of Code 2023

Sugarizer Word Puzzle and Chart activities

Dhruv Mishra

General Information

About Me

I am **Dhruv Mishra**, a 3rd-year undergraduate student pursuing **Computer Science and Engineering** from the **Indian Institute of Technology BHU Varanasi, India**. I am a curious student and am still exploring as much as possible. The field of web development has intrigued me and I always try to implement my skills in solving real life issues and always aim to work for the betterment of society. I have worked on various projects ranging from personal projects to the development of [E-CELL IIT BHU](#) website.

I have a strong foundation in software engineering principles, and I enjoy exploring new technologies and approaches to improve my skills. Apart from this, I enjoy reading books (currently migrating from fiction to self-help) and I write poems in my native language. As expressed above, I like to work for the betterment of society and what can be more joyful than contributing to such a great organization, which strived for the development of small children. Sugar Labs coordinates volunteers around the world who are passionate about providing educational opportunities to children through the Sugar Learning Platform. I believe that my skills and my passion for innovative

research make me a good fit for this project, and I am excited about the opportunity to contribute to its success.

Contact Information

Full name - Dhruv Mishra
Email address - mishra.x.dhruv18@gmail.com
Phone number - +91 7217213319

Education

Institute - Indian Institute of Technology BHU Varanasi
Degree - B. Tech
Major - Computer Science and Engineering
Graduation year - 2024
Courses taken - Intelligent Computing, AI, Computer Graphics, IT, Computer Vision, Computer Architecture, DBMS, OS, Software engineering etc.
Current CGPA - 9.42/10

Other details

1. I had done my internship at [CISCO](#) Systems.
2. I am the tech team manager at E-CELL IIT BHU.
3. Worked with other top firms like BYjus as a freelancer.

Tech-stack and programming languages

[C++](#), [C](#), [Python](#), [JavaScript](#), [MERN](#), [VueJs](#), [ChartJs](#), [HTML](#), [CSS](#), [Django](#)

Resume link : [Resume@Dhruv_Mishra](#)

Github link : [DhruvMishra1826](#)

LinkedIn : [Dhruv Mishra](#)

Languages i speak : English (Full proficiency) and Hindi (Mother tongue)

Location & time zone : Varanasi, India (UTC +5:30)

Project Details

Description :

The goal of this project is to develop new Sugarizer activities asked by teachers from Sugarizer deployments.

Specifically, the goal of this project is to:

- Develop a new Chart activity
- Add a new template Word Puzzle for Exerciser activity

Task and details :

1. Word Puzzle template

The new template Word Puzzle in Exerciser activity will enable a teacher to create word puzzles exercise by inputting custom words on the go during a lesson and have the learners practice.

The new template will allow you to type words by text, images, sounds, speech or videos. It should work with the mouse and on touch devices.

1.1 Scope :

The project will cover the following topics:

- 1.1.1** Implementation of word puzzle template.
- 1.1.2** Templates created will be visually appealing.

- 1.1.3 The question format given by the teacher can be text, image, sound or videos.
- 1.1.4 Teacher would be able to test the assignment before assigning it to the student.
- 1.1.5 The student will be able to drag over the puzzle and can select a word.
- 1.1.6 Other necessary amendments can also be done on recommendation of a mentor.

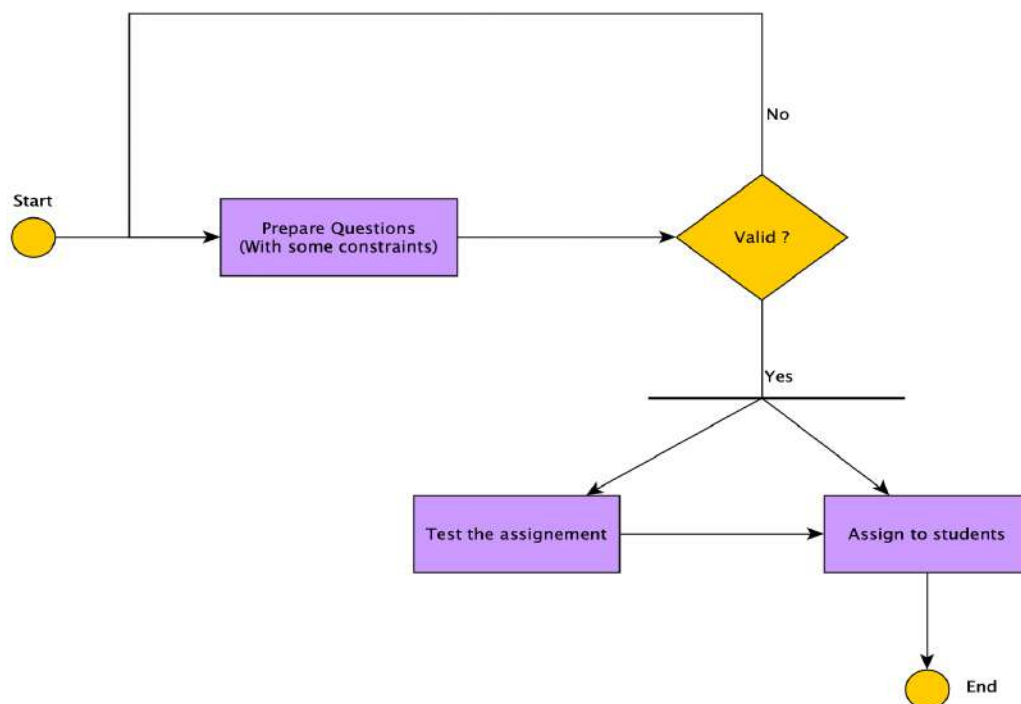
1.2 What am I making ?

I will be implementing this puzzle activity with help of javascript, ReactJs, VueJs, HTML, CSS. As the target of this project is to develop puzzle activities for kids, my target will be to implement a user-friendly interface, which will have all necessary features as stated above. This puzzle activity will be flexible to changes that may be suggested by the project mentors.

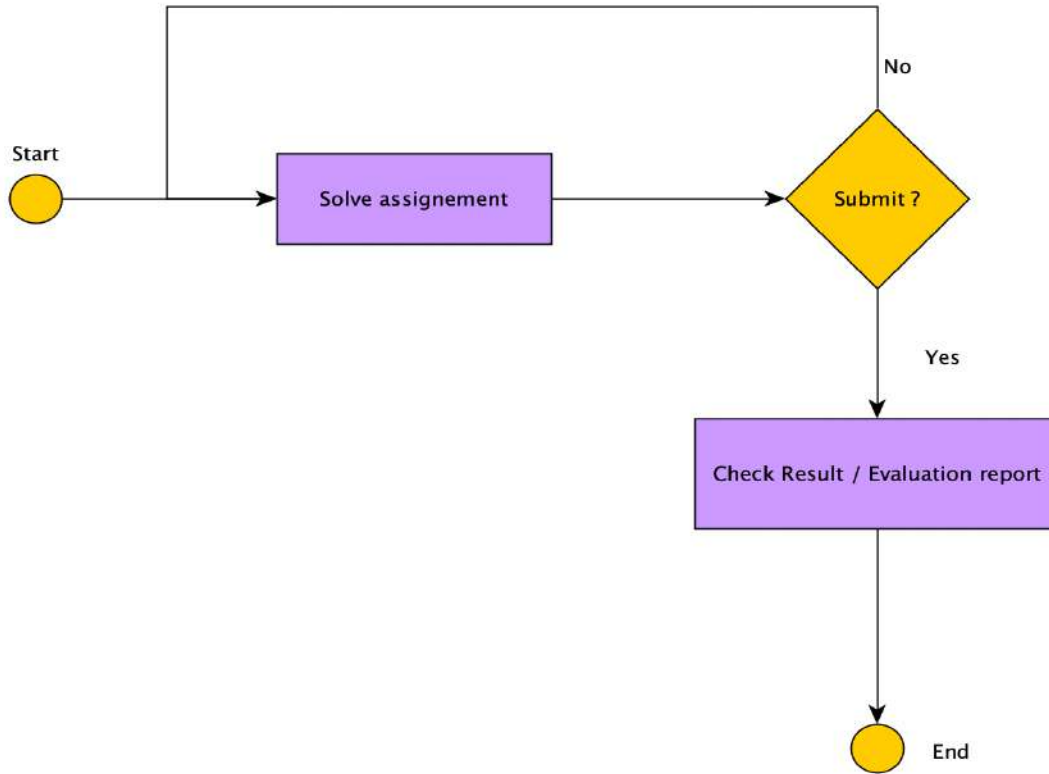
Let me give, a brief description of the steps/checkpoints, which i will be following in order to make this project a success :

Note : (All figures given below are implemented by me for providing mentors a rough idea about work flow)

Activity Diagrams



(Teacher / Instructor view)



(Student view)

Step 1: Implementation of dashboard for the teacher to form questions

It is one of the most important components of this project, as the description and the questions of the activity will be mentioned by the teacher in this section only. This section basically consists of the question, description of activity, the image of activity, questions to be asked and the correct answers to questions.

There will be few planned **constraints**, which are as follows:

- a. Title of the activity should not be an empty / null string.

- b. Exercise description cannot be empty.
- c. Question string should not be empty.
- d. Number of questions in the puzzle should be equal to the number of answers provided.
- e. There should be no empty answer block.

Key Objective : My key objective is not just the implementation of very fancy things, rather it is the development, which follows the trends and the template / UI of sugar labs. So, all my implementations are made keeping in mind the way templates are fixed for other activities.

Dashboard implementation snippets / rough imagination:

File location : `src/containers/builders/PuzzleForm`

```
Initial header files :
import React, { Component } from "react";
import { connect } from "react-redux";
import { incrementExerciseCounter } from
"../../store/actions/increment_counter";
import { addNewExercise, editExercise } from
"../../store/actions/exercises";
import { FormattedMessage } from "react-intl";
import datastore from "lib/sugar-web/datastore";
import chooser from "lib/sugar-web/graphics/journalchooser";
import env from "lib/sugar-web/env";
import meSpeak from "mespeak";
import withMultimedia from "../../components/WithMultimedia";
import {
    QuestionOptionsJSX,
    QuestionJSX,
} from "../../components/MultimediaJSX";
import {
    FINISH_EXERCISE,
    QUESTION,
    TITLE_OF_EXERCISE,
    TEST_EXERCISE,
    PUZZLE,
```

```

    ADD_ANSWER,
    DELETE_ANSWER,
    DESCRIPTION_OF_EXERCISE
  } from "../translation";
import { withRouter } from "react-router-dom";
import "../css/PuzzleForm.css";
import { MULTIMEDIA, setDefaultMedia } from "../utils";

```

a. Implementation of PuzzleForm class.

```


class PuzzleForm extends Component {
  constructor(props) {
    //Initial settings
    super(props);
    this.state = {
      edit: false,
      id: -1,
      title: '',
      description: '',
      question: {
        type: '',
        data: ''
      },
      puzzleText: '',
      answers: [''],
      scores: [],
      times: [],
      isFormValid: false,
      errors: {
        question: false,
        list: false,
        title: false,
      }
    }
  }
}

```

This part will be doing all initial initializations required for the form to have in order to function.

b. Setting the title of the puzzle activity

CrossWord



The screenshot shows a crossword puzzle interface. At the top, the title "CrossWord" is displayed. Below it, a crossword grid is shown with a yellow highlight on the top row. A double-headed arrow indicates the width of the grid. Below the grid, there are two input fields: "Title of Exercise" and "Exercise Description". A download icon is visible in the bottom right corner of the form area.

```
handleChangeTitle = e => {
  //Handling the event of change of title
  let error = false;
  if (e.target.value === '') {
    error = true;
  }
  this.setState({
    ...this.state,
    title: e.target.value,
    errors: {
      ...this.state.errors,
      title: error
    }
  }, () => {
    this.validityCheck();
  });
};
```


c. Setting the description of activity

```
handleChangeDescription = e => {
  //Handling the event of change in description
  let error = false;
  if (e.target.value === '') {
    error = true;
  }

  this.setState({
    ...this.state,
    description: e.target.value,
    errors: {
      ...this.state.errors,
      description: error
    }
  }, () => {
    this.validityCheck();
  });
};
```

d. Setting up the medium of question

Question:



```
QuestionType = (TypeOfMedia) => {
  //Selecting the media of questions
  //Function is bit long, so i resisted to include it here
};
```

e. Assigning the question to the activity

Question:



```
handleChangeQues = e => {  
  //Handling questions  
  let error = false;  
  if (e.target.value === '') {  
    error = true;  
  }  
  this.setState({  
    ...this.state,  
    errors: {  
      ...this.state.errors,  
      question: error  
    },  
    question: {  
      ...this.state.question,  
      data: e.target.value  
    }  
  }, () => {  
    this.validityCheck();  
  });  
};
```

f. Setting up the questions asked in puzzle

CrossWord:

1. Where capital of France located ?
2. Where capital of India located ?
3. It is the best 4.5 gen fighter aircraft. Name it.



```

handlePuzzleQuestions = e => {
  //Adding the questions of puzzle(What to find in puzzle)
  let error = false;
  if (e.target.value === '') {
    error = true;
  }
  this.setState({
    ...this.state,
    errors: {
      ...this.state.errors,
      cloze: error,
    },
    puzzleText: e.target.value,
  }, () => {
    this.validityCheck();
  });
};

```

g. Add answer to the question in the puzzle

1

2

3

Add Answer

Delete Answer

```

handleNewAns = () => {
  //Adding new answer
  const { answers } = this.state;
  this.setState(
    { answers: [...answers, ''] },
    () => {
      this.validityCheck();
    }
  );
};

```

h. Remove answers

In the above figure, you can see a “Delete Answer” button. What it will do is, it will delete the answer to the question you want.

```
handleRemoveAns = () => {
  //Removing the answer
  if (answers.length > 1) {
    answers.pop();
    this.setState(
      { answers: answers },
      () => {
        this.validityCheck();
      }
    )
  }
};
```

i. Change answer of a question in the puzzle

1 Paris

2 Ne Delhi

3 Rafale

Add Answer Delete Answer

```
handleChangeAns = e => {
  //Handling change in answer of any puzzle problem

  const index = Number(e.target.name.split('-')[1]);
  const ans = this.state.answers.map((ans, i) => (
    i === index ? e.target.value : ans
  ));
  let error = false;
  if (e.target.value === '') {
```

```

        error = true;
    }
    this.setState({
        ...this.state,
        answers: ans,
        errors: {
            ...this.state.errors,
            answers: error
        }
    }, () => {
        this.validityCheck();
    });
};

```

j. Checking whether the puzzle is valid and is not violating the constraints.

```

validityCheck = () =>{
    //Function to check validity of form filled by the teacher
    //Again it contains a lot of checks and it's a bit long so,I
    will not be including its snippets.
};

```

k. Testing of puzzle by teacher before assigning

Routing the button to **File location : src/containers/Players/PuzzlePlayer**



```

TestPuzzle = (e) => {
    //To test the working of puzzle
    //It will go to original location from where a student will
    access it
};

```

l. Submission of puzzle

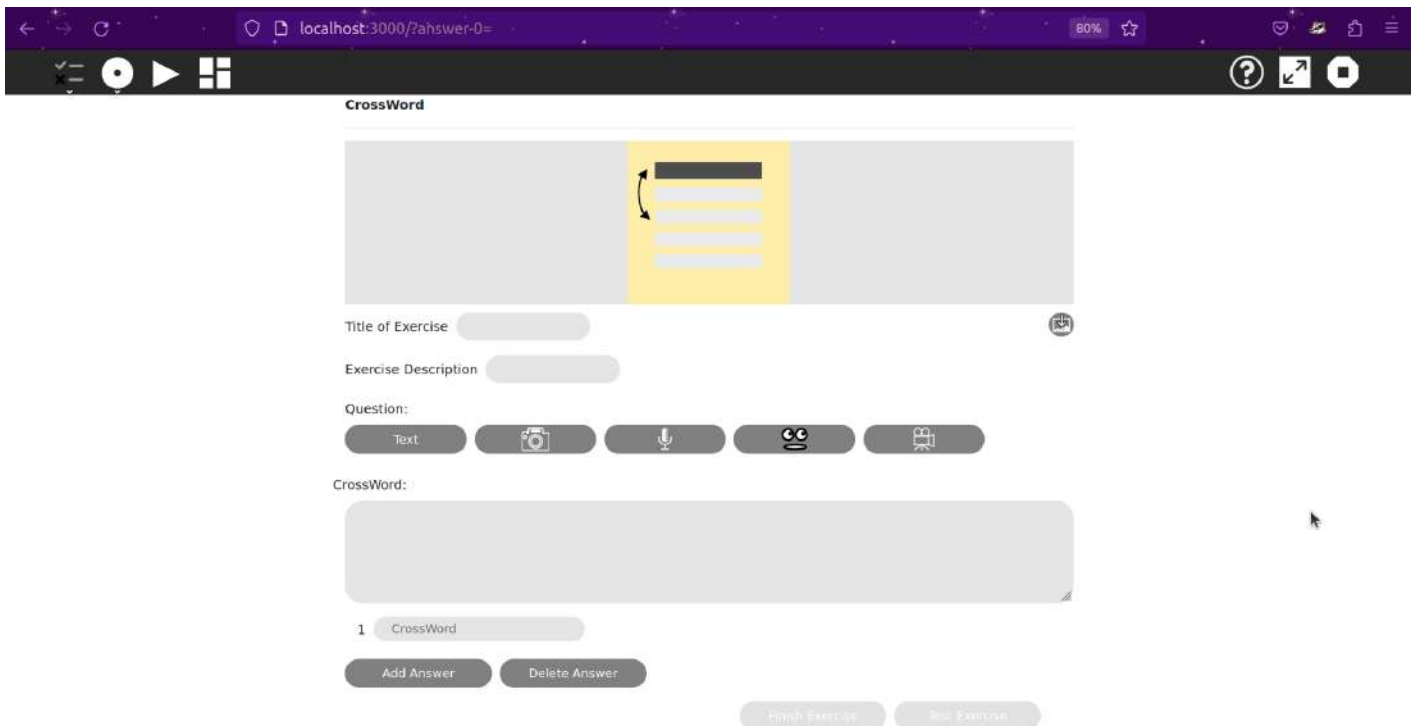


By clicking on the “Finish” button, this puzzle form will be submitted and will be available on the main portal for students to solve.

```
submitPuzzle = (e) => {  
  //Handling final submission of puzzle  
  //Re-directing teacher to home page  
};
```

m. Any other function can also be implemented as per the wish of the mentor.

Dashboard resembling figure 1



Dashboard resembling figure 2 / Example

The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. In the top right corner, there is a progress indicator showing '67%' and a star icon. The main content area is titled 'CrossWord' and features a large grey rectangular area with a yellow vertical bar in the center. Inside the yellow bar, there are several horizontal lines representing a crossword grid, with a double-headed arrow indicating a selection or movement action.

Below the grid, the following information is displayed:

- Title of Exercise: Answer the followir
- Exercise Description: All the best !!
- Question: Drag cursor over the desired word.

Under the heading 'CrossWord:', there is a list of 14 questions:

1. Name the red fruit.
2. King of fruits
3. Known as Santra in hindi
4. It's red coloured vegetable
5. Used in making of french fries
6. Best maneuvering aircrafts
7. Kargin hero aircraft
8. Best interceptor
9. F-16 belongs to which company
10. Make largest aircrafts
11. F-22 belongs to which company
12. French beast !!
13. Aircraft named after storm
14. Indias most reliable bomber

At the bottom, there are five input fields with the following answers:

1. apple
2. mango
3. orange
4. tomato
5. potato

Dashboard resembling figure 3 / Example

The screenshot shows a web application running on localhost:3000. The interface is dark-themed with a purple header. The main content area is white. At the top, there's a progress indicator showing 50% completion and a star icon. Below the header, the exercise title is "Answer the followir" and the description is "All the best !!". The question is "Drag cursor over the desired word." Below this, there's a "CrossWord:" section with a list of 14 questions. The first question is "1. Name the red fruit." and the answer "apple" is entered in the input field. The other questions are: 2. King of fruits, 3. Known as Santro in hindi, 4. It's red coloured vegetable, 5. Used in making of french fries, 6. Best maneuvering aircrafts, 7. Kargin hero aircraft, 8. Best interceptor, 9. F-16 belongs to which company, 10. Make largest aircrafts, 11. F-22 belongs to which company, 12. French beast !!, 13. Aircraft named after storm, 14. Indias most reliable bomber. At the bottom, there are buttons for "Add Answer", "Delete Answer", "Finish Exercise", and "Test Exercise".

localhost:3000 50% ☆

Title of Exercise Answer the followir

Exercise Description All the best !!

Question: Drag cursor over the desired word.

CrossWord:

1. Name the red fruit.
2. King of fruits
3. Known as Santro in hindi
4. It's red coloured vegetable
5. Used in making of french fries
6. Best maneuvering aircrafts
7. Kargin hero aircraft
8. Best interceptor
9. F-16 belongs to which company
10. Make largest aircrafts
11. F-22 belongs to which company
12. French beast !!
13. Aircraft named after storm
14. Indias most reliable bomber

1 apple

2 mango

3 orange

4 tomato

5 potato

6 sukhot

7 mirrage

8 mikoyan

9 turbofan

10 boeing

11 lockheed

12 rafale

13 typhoon

14 jeguar

Add Answer Delete Answer

Finish Exercise Test Exercise

Dashboard resembling figure 4 / (How it looks on builder menu)

The screenshot shows a builder menu for a crossword puzzle exercise. The menu is titled "CrossWord" and has a description: "Build an exercise to fill puzzle corresponding to a question." Below the description is a "Choose" button. The background of the menu is yellow and features icons representing different exercise types.

CrossWord

Build an exercise to fill puzzle corresponding to a question.

Choose

Step 2: Implementation of student side view of activity

This is what will be visible to the students. This section basically consists of the puzzle grid, Instruction pop-up, the questions list, and the answered questions will be replaced by the correct answers if you have found it

There will be few planned **constraints**, which are as follows:

- a. Students are only allowed to drag the cursor either horizontally or vertically, not diagonally.
- b. If the student finds the word, then he/she has to drag over the complete word in order to get that correct.
- c. Selection of words will be only considered if the cursor is within the boundaries of the puzzle.

Puzzle activity implementation snippets/rough imagination:

File location : `src/containers/Players/PuzzlePlayer`

- a. Initializing a few containers that will be storing certain values in it.

```
var questionList = new Array(); //Contains questions
var AnsList = new Array(); //Contains answers
const [currScore,setScore] = useState(0); //Score tracker
const [grid,setGrid] = useState([]); //Initial grid
const [currWord,setCurrWord] = useState(''); //selected word
```

- b. Implementation of grid

This function is intended to store the words used in the word search puzzle. To determine the appropriate size of the array, the function computes the longest word present in the input array and uses its length as the width of the array, while the number of words in the input array determines the height. The resulting empty grid array is then returned by the function.

```

makeGrid = (AnsList) => {
  //Implementation of the grid from the given answers list
  // Pseudocode :
    // Width of grid = longestWord + 2;
    // Length of grid = longestWord + 2;
    // Filling the answers list in grid;
    // For remaining places, i will be randomly
    // filled with english characters.
};

```

```

▼ grid: Array(180) [ (180) [...], (180) [...], (180) [...], ... ]
  ▼ [0..99]
    ▼ 0: Array(180) [ {...}, {...}, {...}, ... ]
      ▼ [0..99]
        ▶ 0: Object { character: "B", items: [] }
        ▶ 1: Object { character: "H", items: [] }
        ▶ 2: Object { character: "V", items: [] }
        ▶ 3: Object { character: "K", items: [] }
        ▶ 4: Object { character: "P", items: [] }
        ▶ 5: Object { character: "V", items: [] }
        ▶ 6: Object { character: "J", items: [] }
        ▶ 7: Object { character: "N", items: [] }
        ▶ 8: Object { character: "J", items: [] }
        ▶ 9: Object { character: "Y", items: [] }
        ▶ 10: Object { character: "Y", items: [] }
        ▶ 11: Object { character: "X", items: [] }
        ▶ 12: Object { character: "Z", items: [] }
        ▶ 13: Object { character: "V", items: [] }
        ▶ 14: Object { character: "G", items: [] }
        ▶ 15: Object { character: "D", items: [] }
        ▶ 16: Object { character: "I", items: [] }
        ▶ 17: Object { character: "J", items: [] }
        ▶ 18: Object { character: "D", items: [] }
        ▶ 19: Object { character: "T", items: [] }
        ▶ 20: Object { character: "O", items: [] }
        ▶ 21: Object { character: "I", items: [] }
        ▶ 22: Object { character: "X", items: [] }

```

c. Showing initial setup of puzzle

This function manages the initial setup, a student will see as this puzzle activity loads. This function basically keeps track of the initial puzzle

structure, which we will be getting from the “makeGrid()” function. It will also ensure the necessary elements like title, hint box also pops up as page load.

```
showInitial = () =>{  
  //Will be showing initial setup of puzzle  
  // Pseudocode:  
    // makeGrid(AnsList);  
    // Initial question list on right  
  
  //Initial designing components will also be called  
};
```

d. Display grid

```
displayGrid = (grid) =>{  
  //displaying the grid formed on screen  
};
```

F	Z	J	B	R	V	O	E	O	T	U	T	T	I	S
X	P	Z	F	R	U	M	Y	T	K	I	Y	O	U	U
D	K	E	F	F	M	I	U	W	V	A	C	Q	T	K
K	F	T	I	U	I	U	B	Y	X	Y	R	K	Y	H
W	Q	I	L	Q	R	I	R	S	Z	M	A	N	G	O
G	Z	O	Q	B	R	U	V	W	F	W	P	K	A	I
L	Y	T	O	M	A	T	O	K	N	S	P	J	H	N
M	W	Y	J	A	G	U	A	R	Q	S	L	T	V	S
N	M	W	O	B	E	R	A	F	A	L	E	L	X	Y
Q	U	T	U	R	C	B	O	E	I	N	G	R	Z	N

e. Selection of a particular word in puzzle

Constraint : Selection word should either horizontal or vertical (not diagonally) and should be in continuous manner and those be within the boundaries of the grid.

```
selectWord = (grid) =>{  
    //defines selecting of word by dragging over the puzzle  
    // Implementing actions to take onMouseDown() a  
    particular letter  
};
```

f. Checking the correctness of marked word

Now the region whatever you dragged the cursor will be selected to form a word. If that word is the answer to any question then that particular region will be marked with a green color to state that the selected word is the correct answer.

```
checkSelectedWord = (word) =>{  
    //Checking whether the selected word area is a valid  
    answer of any question or not  
};
```

F	Z	J	B	R	V	O	E	O	T	U	T	T	I	S
X	P	Z	F	R	U	M	Y	T	K	I	Y	O	U	U
D	K	E	F	F	M	I	U	W	V	A	C	Q	T	K
K	F	T	I	U	I	U	B	Y	X	Y	R	K	Y	H
W	Q	I	L	Q	R	I	R	S	Z	M	A	N	G	O
G	Z	O	Q	B	R	U	V	W	F	W	P	K	A	I
L	Y	T	O	M	A	T	O	K	N	S	P	J	H	N
M	W	Y	J	A	G	U	A	R	Q	S	L	T	V	S

g. Correct answer changes in UI

This function basically defines changes that should take place once it is noticed that you have selected the correct answer.

These changes in the UI can be very easily molded as per the convenience and liking of the mentor.

```
correctAnswer = () =>{  
    //If selected word is an answer, do relevant changes  
    on UI of puzzle  
};
```

In my case I think that, if a student finds the correct answer then the question corresponding to that answer should be replaced with the answer in the questions list. It looks better.

T	I	S
O	U	U
Q	T	K
K	Y	H
N	G	O
K	A	I
J	H	N
T	V	S

1. Name the red fruit.
2. King of fruits
3. Known as Santra in hindi
4. It's red coloured vegetable
5. Used in making of french fries
6. **SUKHOI**
7. Kargin hero aircraft
8. Best interceptor
9. F-16 belongs to which company

Note : These changes are temporary and can be changed as per the wish of the mentor.

h. Calculating the score

```
scoreCalculator = () =>{  
  //Storing the current score  
  //Pseudocode :  
      // if word is correct then :  
          // currScore = currScore + 1  
  // setScore(currScore)  
};
```

i. Remove the selected word

Now, it's highly possible word is not the answer to any of the questions. So, we have to unmark the complete dragged part back to normal state, and this should not affect other correctly marked answers.

```
removeSelectedWord = () =>{  
  //If selected word is not a valid answer, unmark that  
  region back to original  
};
```

j. Updating question list present right of puzzle

In my case if a student finds the correct answer then the question corresponding to that answer should be replaced with the answer in the questions list. It looks better. (Figure above).

```
updateQuestionList = () =>{  
  //As shown in UI, there will be a question list on  
  right side of puzzle hence we will be updating it, if  
  correct answer is found  
};
```

k. Submission of assignment

After completing the assignment, student can submit their test and through “**submitActivity()**” function, i will be doing two things:

1. **Checking the validity** of form, means whether students validly solved questions or not as directed in the instruction part.
2. Accept the form and route students to the **result and evaluation part**.

```
submitActivity = () =>{  
  //Submit the assignment  
  // route to evaluation and result part.  
};
```

- l. Any other function can also be implemented as per the wish of the mentor.

Enhancement in puzzle activity :

(Case study)

There lived a boy Alice in Bordeaux, he belonged to an area where power cuts were very common. Now, his online assignment is going on and he has to solve the assignment by 4 PM. It's 3:50 PM, and unfortunately, the power went off. Now, surprisingly power came just 3 minutes before the deadline. As he opened the portal for the test on his laptop, he saw that all the answers he marked were now vanished. Now, he started crying. And this is the end.

Problems : Assignment was not stored at a particular place. Other sugar assignments save the progress of the student. So, I also want to implement that feature in this activity,

1. Recording the Chart Activity in a journal :

The purpose of using a journal is to preserve the state of an activity, ensuring that when the user reopens it, it resumes from the precise point at which the user left off.

Sugar-web facilitates the use of the journal by providing the `sugar-journal` component, which must be included within the `app` element of the HTML file.

2. Save the context in the datastore :

The concept involves storing the current activity's context in the datastore whenever we exit the activity. Sugar utilizes the Journal as its datastore, which Sugar-Web initializes automatically during activity setup.

Here's how we can save the state of the activity: First, we create a context object and include the required data, such as the chart data. Next, we provide this object as an argument to the `saveData()` method of Sugar-Journal.

First, we will define the context object and add the charts to it, then pass this to the `saveData()` method.

```
var context = {
  currentGrid: this.currentGrid
};
this.$refs.SugarJournal.saveData(context);
```

```
methods: {
  ...
  onStop: function () {
    // Save current pawns in Journal on Stop
    var context = {
      currentGrid: this.currentGrid
    };
    this.$refs.SugarJournal.saveData(context);
```



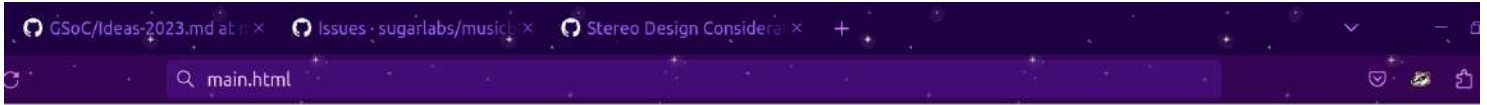
```
}  
}
```

This whole code should be called at the end of the activity. To do that we have to catch the click on the Stop button of the activity. So let's create another method in `js/activity.js` to call when stopping the activity.

Then add the event listener to the stop-button in `index.html`.

```
<sugar-toolitem id="stop-button" title="Stop" class="pull-right"  
v-on:click="onStop"></sugar-toolitem>
```

Puzzle resembling figure 1



sugarlabs

Puzzle

F	Z	J	B	R	V	O	E	O	T	U	T	T	I	S
X	P	Z	F	R	U	M	Y	T	K	I	Y	O	U	U
D	K	E	F	F	M	I	U	W	V	A	C	Q	T	K
K	F	T	I	U	I	U	B	Y	X	Y	R	K	Y	H
W	Q	I	L	Q	R	I	R	S	Z	M	A	N	G	O
G	Z	O	Q	B	R	U	V	W	F	W	P	K	A	I
L	Y	T	O	M	A	T	O	K	N	S	P	J	H	N
M	W	Y	J	A	G	U	A	R	Q	S	L	T	V	S
N	M	W	O	B	E	R	A	F	A	L	E	L	X	Y
Q	U	T	U	R	C	B	O	E	I	N	G	R	Z	N
A	S	U	J	V	L	O	C	K	H	E	E	D	C	F
K	G	M	H	Y	Z	F	Z	F	C	L	Q	L	D	B
M	I	K	O	Y	I	A	N	Y	R	K	I	H	I	T

1. Name the red fruit.
2. King of fruits
3. Known as Santra in hindi
4. It's red coloured vegetable
5. Used in making of french fries
6. Best maneuvering aircrafts
7. Kargin hero aircraft
8. Best interceptor
9. F-16 belongs to which company
10. Make largest aircrafts
11. F-22 belongs to which company
12. French beast !!
13. Aircraft named after storm
14. indias most reliable bomber

(Initial setup)

Puzzle resembling figure 2

sugarlabs Puzzle

F	Z	J	B	R	V	O	E	O	T	U	T	T	I	S
X	P	Z	F	R	U	M	Y	T	K	I	Y	O	U	U
D	K	E	F	F	M	I	U	W	V	A	C	Q	T	K
K	F	T	I	U	I	U	B	Y	X	Y	R	K	Y	H
W	Q	I	L	Q	R	I	R	S	Z	M	A	N	G	O
G	Z	O	Q	B	R	U	V	W	F	W	P	K	A	I
L	Y	T	O	M	A	T	O	K	N	S	P	J	H	N
M	W	Y	J	A	G	U	A	R	Q	S	L	T	V	S
N	M	W	O	B	E	R	A	F	A	L	E	L	X	Y
Q	U	T	U	R	C	B	O	E	I	N	G	R	Z	N
A	S	U	J	V	L	O	C	K	H	E	E	D	C	F
K	G	M	H	Y	Z	F	Z	F	C	L	Q	L	D	B
M	I	K	O	Y	I	A	N	Y	R	K	I	H	I	T

1. Name the red fruit.
2. King of fruits
3. Known as Santra in hindi
4. It's red coloured vegetable
5. Used in making of french fries
6. **SUKHOI**
7. Kargin hero aircraft
8. Best interceptor
9. F-16 belongs to which company
10. Make largest aircrafts
11. F-22 belongs to which company
12. French beast !!
13. Aircraft named after storm
14. **JAGUAR**

(Selection of a correct word)

[Question will be replaced by correct answer]

Puzzle resembling figure 3

F	Z	J	B	R	V	O	E	O	T	U	T	T	I	S
X	P	Z	F	R	U	M	Y	T	K	I	Y	O	U	U
D	K	E	F	F	M	I	U	W	V	A	C	Q	T	K
K	F	T	I	U	I	U	B	Y	X	Y	R	K	Y	H
W	Q	I	L	Q	R	I	R	S	Z	M	A	N	G	O
G	Z	O	Q	B	R	U	V	W	F	W	P	K	A	I
L	Y	T	O	M	A	T	O	K	N	S	P	J	H	N
M	W	Y	J	A	G	U	A	R	Q	S	L	T	V	S
N	M	W	O	B	E	R	A	F	A	L	E	L	X	Y
Q	U	T	U	R	C	B	O	E	I	N	G	R	Z	N
A	S	U	J	V	L	O	C	K	H	E	E	D	C	F
K	G	M	H	Y	Z	F	Z	F	C	L	Q	L	D	B
M	I	K	O	Y	I	A	N	Y	R	K	I	H	I	T
T	Y	P	H	O	O	N	G	H	E	J	N	Q	C	A
I	O	O	W	R	K	H	J	F	Q	Y	L	V	R	P
T	Q	T	I	A	I	J	K	J	T	L	B	V	P	I
G	F	A	U	N	T	G	B	D	A	N	R	N	X	G
L	H	T	Z	G	R	O	X	B	J	C	W	J	H	F
T	B	O	E	E	E	L	M	U	P	W	C	X	N	C

1. **APPLE**
2. King of fruits
3. Known as Santra in hindi
4. It's red coloured vegetable
5. Used in making of french fries
6. **SUKHOI**
7. Kargin hero aircraft
8. Best interceptor
9. F-16 belongs to which company
10. Make largest aircrafts
11. F-22 belongs to which company
12. **RAFALE**
13. **TYPHOON**
14. **JAGUAR**

(Proposed UI of puzzle)

Evaluation Mode / rough imagination:

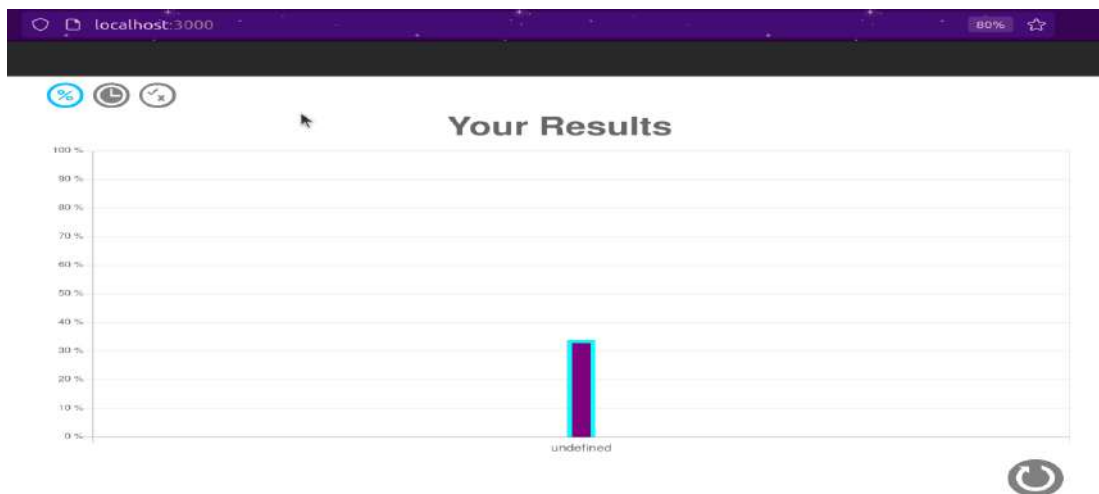
File location : src/store/actions and src/store/reducers

I will be using the standard way of evaluation of sugarLabs as it feels it's the best in current setUp. Below are some of the instances of evaluation that I will be implementing.

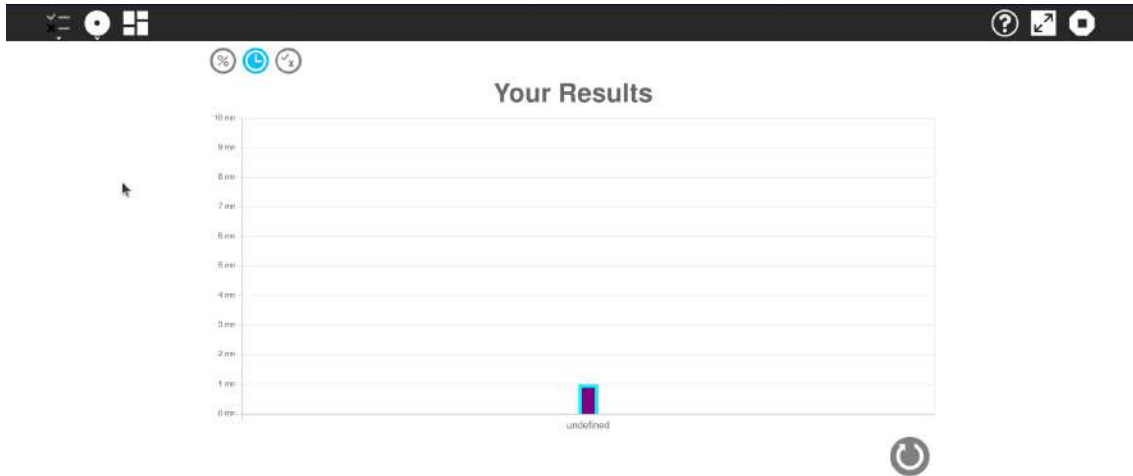
1. Question wise correctness check :

Question	Correct Answer	Correct / Wrong
Name the red fruit.	APPLE	✗
King of fruits	MANGO	✓
Known as Santra in hindi	ORANGE	✓
It's red coloured vegetable	TOMATO	✗
Used in making of french fries	POTATO	✗
French beast !!	RAFALE	✗

2. Total score analysis :



3. Time taken to solve :



Step 3: Connection of student side view with teacher's dashboard.

1.3 What's after this ?

I am open to new ideas and manipulations from your side and I assure you, I will be trying my best to incorporate that.

2. Chart Activity

2.1 Scope :

The project will cover the following topics:

- 2.1.1 Implementation of chart activity.
- 2.1.2 Addition of few new features like sharing of activity and exporting charts as images.
- 2.1.3 All necessary amendments asked by the mentor and sugar Labs organization.

2.2 What am I making ?

I will be implementing this Charts activity with help of chartJs, javascript, ReactJs, VueJs, HTML, CSS and any other existing library for chart activity can also be used, i am flexible to use such libraries. As the target of this project is to develop and modify existing chart activities for kids, my target will be to implement a user-friendly interface, which will have all necessary features as stated above. This chart activity will be flexible to changes that may be suggested by the project mentors.

Let me give, a brief description of the steps/checkpoints will be following in order to make this project a success :

Step 1: Implementation of chart activity having existing features

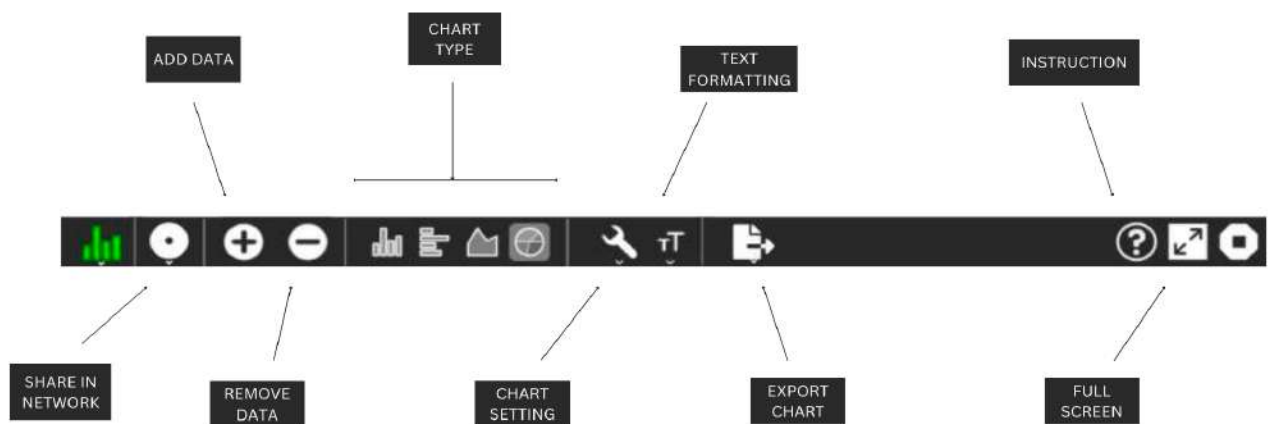
The chart activities will have following features:

- a. One can create various types of graphs like Bar graph, horizontal bar graph, Line graph and Pie chart.
- b. One can add and remove entries from the graphs
- c. One can change the colors and UI of graphs/charts by changing the settings.
- d. One can access the puzzles both in normal mode and full screen mode.
- e. The tutorial is also present.

Step 2: Additional features that we can add :

- Sharing of chart activity can be implemented.
- Change in labels and colors as per convenience of the user.
- New chart activities like doughnut charts, area charts etc can also be implemented.
- Export charts as images.

Implementation / rough imagination:



3. Some Initializations

```
import React, { Component } from 'react';
import './App.css';
import Pie from './Pie'; //Pie chart implementation
import Line from './Linegraph'; //Line graph implementation
import Bar from './Bars'; //Bar graph implementation
import Hbar from './Hbars'; //Horizontal bar graph imp
```



```

import { useState } from 'react';

export default class App extends Component{

  constructor(props) {
    super(props);
    this.state = {
      showPie: false,
      showLine: false,
      showBar: false,
      showHbar: false,
      addNew:{value:'',num:0},
      List: [{
        value : "",
        number : '',
      }
    ]
  };
  this.handleClickPie = this.handleClickPie.bind(this);
  this.handleClickLine = this.handleClickLine.bind(this);
  this.handleClickBar = this.handleClickBar.bind(this);
  this.handleClickHbar = this.handleClickHbar.bind(this);
}
}

```

4. Add data (+)

Users can utilize the "add" button to incorporate fresh labels and their corresponding values, which will then be presented on the charts visible on the screen. An event will activate the method upon the user's selection of the "+" button. This method will obtain user input values, such as the label and data, and subsequently append them to the chart data array. Following the addition of data, the chart will undergo re-rendering to exhibit the updated chart, now including the new data.

Working :

Initial list :

COLOUR	VALUE
Red	58
Yellow	65
Green	80
Purple	82

Input color name Input color value

Intermediate step :

<input type="text" value="Blue"/>	<input type="text" value="18"/>
-----------------------------------	---------------------------------

After addition :

COLOUR	VALUE
Red	58
Yellow	65
Green	80
Purple	82
Blue	18

Input color name Input color value

```

add={()=>{
  const { List } = this.state;

  let val1 = document.getElementById('name').value;
  let val2 = document.getElementById('number').value;
  document.getElementById('name').value='';
  document.getElementById('number').value='';

  let obj = {value: val1, number: val2};

  this.setState(
    { List: [...List, obj] },
    () => {
      });

  let data = document.getElementById("tabledata");

  let html = `- 

```

5. Remove data (-)

Users can make use of the "Remove Data" feature to eliminate data from the chart. When the user clicks on the "-" button, an event will be triggered to activate the method.

```

remove={()=>{
  const { List } = this.state;

  if (List.length > 1) {
    List.pop();
  }
};

```

```

    this.setState(
      { List: List },
      () => {

      }
    )}
  // console.log(List);
  this.handleClickBar()
}

```

Working :

Before deletion and after deletion :

COLOUR	VALUE
Red	58
Yellow	65
Green	80
Purple	82

Input color name Input color value

6. Charts Implementation

4.1 Bar graph

4.1.1 Initializations and state definition

```

import React from 'react';
import { Chart, CategoryScale, LinearScale, BarElement } from

```

```

'chart.js'
import {Bar} from 'react-chartjs-2';
import './Bars.css'
Chart.register(CategoryScale, LinearScale, BarElement)

const state = {

  labels: props.labels,
  datasets: [
    {
      label: props.label;
      backgroundColor: props.bgColor,
      borderColor: props.borderColor,
      borderWidth: props.bWidth,
      data: props.data
    }
  ]
}

```

4.1.2 Class “Bars” implementation

```

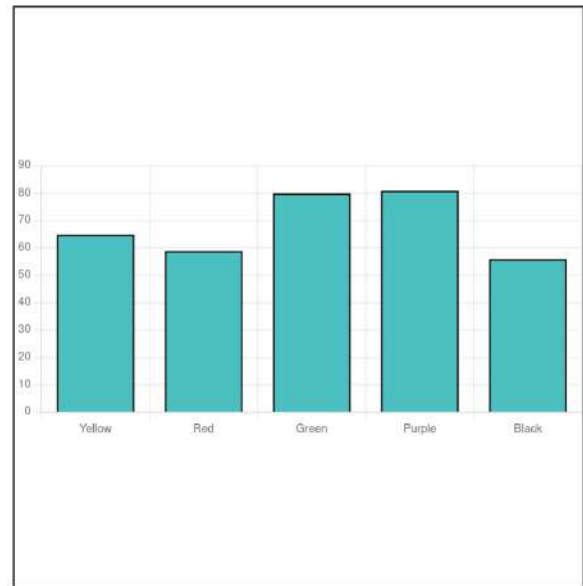
export default class Bars extends React.Component {
  render() {
    return (
      <div className='Bars'>
        <Bar
          data={state}
          options={{
            title:{
              display:true,
            },
            legend:{
              display:true,
              position:'right'
            }
          }}
        />
      </div>
    )
  }
}

```

```
    </div>  
  );  
}  
}
```

COLOUR	VALUE
Red	58
Yellow	65
Green	80
Purple	82
Black	56

Input color name Input color value



4.2 Line chart

4.2.1 Initializations and state definition

```
import React from 'react';  
import {Line} from 'react-chartjs-2';  
import { Chart, LineController, LineElement, PointElement,  
LinearScale, Title, CategoryScale } from 'chart.js';  
import './Linegraph.css';
```

```
Chart.register(LineController, LineElement, PointElement,  
LinearScale, Title, CategoryScale);  
Chart.register(CategoryScale);
```

```

const state = {
  labels: props.labels,
  datasets: [
    {
      label: props.label,
      fill: props.fill,
      lineTension: props.LineTension,
      backgroundColor: props.bgColor,
      borderColor: props.borderColor,
      borderWidth: props.borderWidth,
      data: props.data
    }
  ]
}

```

4.2.2 Class “Linegraph” implementation

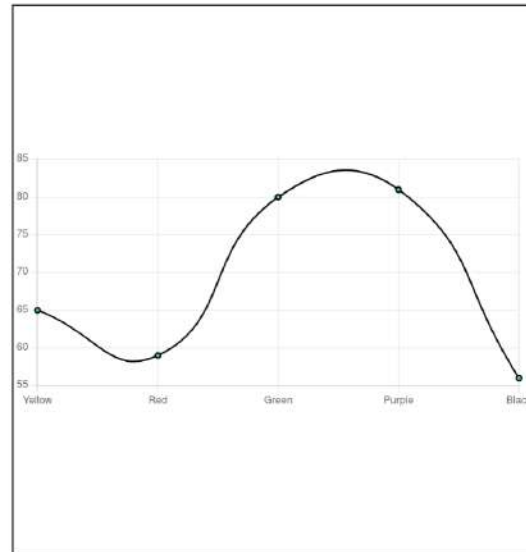
```

export default class Linegraph extends React.Component {
  render() {
    return (
      <div className='line'>
        <Line
          data={state}
          options={{
            title:{
              display:true,
            },
            legend:{
              display:true,
              position:'right'
            }
          }}
        />
      </div>
    );
  }
}

```

```
}  
}
```

COLOUR	VALUE
Red	58
Yellow	65
Green	80
Purple	82
Black	56



4.3 Pie chart

4.3.1 Initializations and state definition

```
import React from 'react';  
import {Pie} from 'react-chartjs-2';  
import {ArcElement} from 'chart.js';  
import { Chart, LineController, LineElement, PointElement,  
LinearScale, Title, CategoryScale } from 'chart.js';  
import './Pie.css';  
Chart.register(LineController, LineElement, PointElement,  
LinearScale, Title, CategoryScale);  
Chart.register(ArcElement);  
  
const state = {  
  labels: props.labels,  
  datasets: [  

```

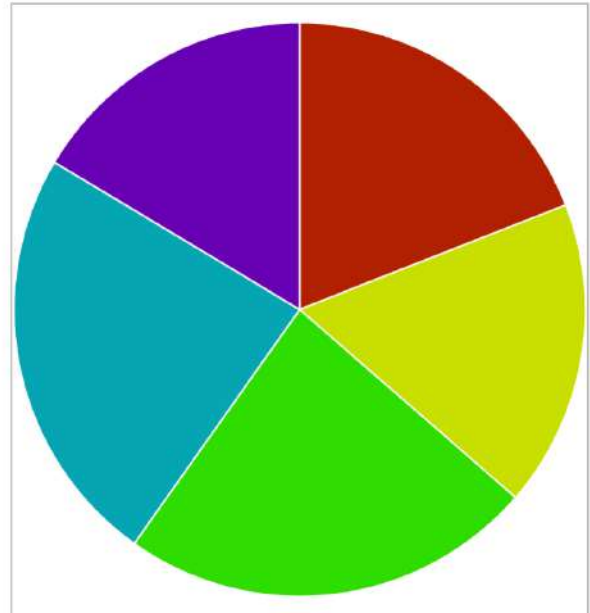


```
{
  label: props.label,
  backgroundColor: props.bgColor,
  hoverBackgroundColor: props.hbgColor,
  data: props.data
}
]
}
```

4.3.2 Class “Pies” implementation

```
export default class Pies extends React.Component {
  render() {
    return (
      <div className='pie'>
        <Pie
          data={state}
          options={{
            title:{
              display:true,
            },
            legend:{
              display:true,
              position:'right'
            }
          }}
        />
      </div>
    );
  }
}
```

COLOUR	% SHARE
Red	20
Yellow	10
Green	30
Sky Blue	30
Purple	10



7. Chart setting

Within the chart activity, the "Chart Settings" function empowers users to personalize their chart by modifying the labels of the horizontal and vertical axes. This function carries significance since it enables users to label the axes correctly, resulting in an accurate representation of their data.

Moreover, users can also alter the **label color** to highlight them or match a specific theme. To execute this feature, Vue.js components will be utilized to ensure that it remains change sensitive. By offering users the ability to customize their chart, they will have greater command over how their data is presented, enabling them to make informed decisions based on the insights they derive from the chart.

```
settings(){  
  //this function will act as a single window to all functions that  
  will be changing the charts
```

```
//Pseudocode :  
    // updateLabel()  
    // changeColors()  
    // changeChartName()  
    // updateChart()  
    // Any other variation wanted by mentor  
};
```

8. Text formatting :

Within the chart activity, there are multiple tools available for users to format the appearance of text.

These formatting options include altering the:

- a. text's color,
- b. size, and
- c. font style

from a selection of predefined options in the application. To apply these formatting changes to the text, users can conveniently select the desired formatting tool from the toolbar provided

These features can be well implemented by using internal libraries of sugarizer like **sizePalette.js**, **ForegroundColourPalette.js**, **fontPalette.js** & **formatTextPalette.js**.

```
textFormatter(){  
    //this function will be just used to enhance the UI and way of  
    representation of content, by enhancing the text fonts, text size  
    ,text colors etc.  
    //Pseudocode :  
        // updateFont()  
        // changeTextsize()  
        // changeTextColor()  
};
```

9. Export Chart :

Basically there are two ways in which we can export the charts. The first is to “**export chart as image**” and “**export chart as pdf**”. Now, this can be achieved by two different libraries, viz “**jspdf**” and “**html2canvas**”. Let’s discuss each part in a bit of detail.

Export charts as pdf:

The act of exporting a chart as PDF is a very useful enhancement / feature. This enables a user to take a current snap of your current chart work and save it as PDF on your local devices. There are quite a few methods to implement this but the best in my opinion is the application of the “jsPDF” library. jsPDF is a popular open-source JavaScript library that allows you to generate PDF documents from your HTML content using pure client-side JavaScript. It provides an easy-to-use API that enables you to create PDF documents with text, images, graphics, and other elements.

Implementation :

1. I will be implementing a “exportPDF()” function and the first step of it will be to create a new instance of jsPDF.

```
var doc = new jsPDF();
```

2. Now, I will be using the “getElementById()” method, in order to get the HTML element to be exported.

```
var region = document.getElementById("region-to-export");
```

3. Now, I will be using “html2canvas”, which will render the selected region as an image.

```
html2canvas(element, {scale}).then(function(canvas) {  
  // Using html2canvas to render the region as an image  
})
```

4. Using “html2canvas”, I will render the region as an image and then add the image to PDF using “doc.addImage()”.
5. Finally, save the PDF using “doc.save()”.

```
html2canvas(element, {scale}).then(function(canvas) {  
  
  var imgData = canvas.toDataURL('img/png');  
  doc.addImage(img, 'PNG', 0, 0);  
  
  doc.save('exported-page.pdf');  
});
```

Export charts as image:

Implementation :

1. First of all, I will be importing the “html2canvas” library.

```
import html2canvas from 'html2canvas';
```

2. Create a canvas element that will hold the chart that I want to export as an image and create a button to fire the “exportChart()” function.

```
<canvas id="myChart"></canvas>  
<button onclick="exportChart()">Export chart as image</button>
```

3. I will be rendering the chart on the canvas element.

```
var canvas = document.getElementById("myChart");  
var ctx = canvas.getContext("2d");
```

4. I will write the “**exportChart()**” function. The chart will be exported as a PNG file, when the user clicks the button.

```
function exportChart() {  
  var ele= html2canvas(document.getElementById("myChart"));  
  
  ele.then(function(canvas) {  
    document.createElement("a").download = "chart.png";  
    document.createElement("a").href =  
    canvas.toDataURL("image/png").replace("image/png",  
    "image/octet-stream");  
  
    document.createElement("a").click();  
  });  
}
```

10. Share activity :

This part can be best implemented by taking into reference the tutorial provided by sugarizer i.e: [Handle multi-user with presence](#) . Each and every small step is mentioned in this blog.

To enable this feature, a pop-up kind of box can be implemented like cards and palettes. To facilitate reuse, a data variable will be employed to reference the component instance multiple times. Additionally, a mounted function will be included in the Vue main application.

To incorporate presence into the network button, we will modify the sugar-toolitem for the same. To begin with, we will handle the click on the share button by adding several attributes such as `palette-event="shared"`, `v-if="SugarPresence"`, and `v-on:shared="SugarPresence.onShared"`.

```
data: {
  SugarPresence: null,
  ...
},
mounted: function () {
  this.SugarPresence = this.$refs.SugarPresence;
  ...
},
```

```
<sugar-toolitem
  id="network-button"
  title="Network"
  palette-file="sugar-web/graphics/presencepalette"
  palette-class="PresencePalette"
  palette-event="shared"
  v-on:shared="SugarPresence.onShared"
  v-if="SugarPresence"
></sugar-toolitem>
```

When the user clicks the share button, it will activate the shared event specified by **"palette-event,"** resulting in the sharing of the activity, which can then be seen by others in the neighborhood view. The component's **"onShared()"** method will take care of this process.

```
onNetworkDataReceived(msg) {
  // Handles the data-received event
},

onNetworkUserChanged(msg) {
  // Handles the user-changed event
},
```

During a shared activity, when a user joins. To handle it, I will be using the events provided by the sugar-presence component.

11. Instruction / activity tour / tutorial:

The working source for me for this part will be the amazing documentation named [Integrate a tutorial](#). I will be following this blog for implementation of this part.

To incorporate the tutorial tour into the activity, we will utilize the intro.js library in conjunction with the sugar-tutorial component that is supplied by sugar-web. The Intro.js is a compact JavaScript library that empowers us to generate compelling, progressive customer onboarding tours.

```
onHelp: function () {
  var steps = [
    {
    },
    {
    },
    {
    }
  ];
  this.$refs.SugarTutorial.show(steps);
},
```

By utilizing the sugar-toolitem component, we will append a button and subsequently allocate the v-on:click directive to it, which will activate an onHelp function.

Any change or suggestion made by a mentor can be easily accommodated.

12. Chart activity localization :

This will enable the activity to run in multiple languages. For this purpose we will be using another sugar-web component named “sugarL110n”. We will update the

locale.ini file with translations of all the strings in multiple languages in order to localize the activity.

Eg- Like this,

```
[*]
.
.
Hello=Hello {{name}}!
Played={{name}} played
AddPawn=Add pawn

[en]
.
.
Hello=Hello {{name}}!
Played={{name}} played
AddPawn=Add pawn

[fr]
.
.
Hello=Bonjour {{name}} !
Played={{name}} a joué
AddPawn=Ajouter pion

[es]
.
.
Hello=Hola {{name}} !
Played={{name}} jugó
AddPawn=Agrega un peón
```

13. Recording the Chart Activity in a journal :

The purpose of using a journal is to preserve the state of an activity, ensuring that when the user reopens it, it resumes from the precise point at which the user left off.

Sugar-web facilitates the use of the journal by providing the sugar-journal component, which must be included within the app element of the HTML file.

14. Save the context in the datastore :

The concept involves storing the current activity's context in the datastore whenever we exit the activity. Sugar utilizes the Journal as its datastore, which Sugar-Web initializes automatically during activity setup.

Here's how we can save the state of the activity: First, we create a context object and include the required data, such as the chart data. Next, we provide this object as an argument to the saveData() method of Sugar-Journal.

First, we will define the context object and add the charts to it, then pass this to the saveData() method.

```
var context = {
    barChart: this.barChart
};
this.$refs.SugarJournal.saveData(context);
```

```
methods: {
    ...
    onStop: function () {
        // Save current pawns in Journal on Stop
        var context = {
            barChart: this.barChart
        };
        this.$refs.SugarJournal.saveData(context);
    }
}
```

This whole code should be called at the end of the activity. To do that we have to catch the click on the Stop button of the activity. So let's create another method in js/activity.js to call when stopping the activity.

Then add the event listener to the stop-button in index.html.

```
<sugar-toolitem id="stop-button" title="Stop" class="pull-right"
v-on:click="onStop"></sugar-toolitem>
```

15. Stop / Full screen button :

File location : src/components/FullScreenAndTutorial.js

Implementation of class :

```
class FullScreenAndTutorial extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isfullScreen: false,
      showTutorial: false,
    }
  }
}
```

Implementation of Functions :

```
startTutorial = () => { //Tutorial initiated
  this.setState({
    showTutorial: true,
  });
};
//Toggling through various cases
```

```

stopTutorial = () => {                                     //Tutorial terminated
    this.setState({
        showTutorial: false,
    });
};
goFullscreen = () => {                                     //Full screen initialized
    this.setState({
        isfullScreen: true,
    });
};

gounFullScreen = () => {                                   //Full screen terminated
    this.setState({
        isfullScreen: false,
    });
};

```

Implementation of react fragment (subject to variation based on className and id namings) :

```

<React.Fragment>
  <MainToolbar
    {...this.props}
    {...navFunctions}
    showTutorial={this.state.showTutorial}
    shared_exercises={this.props.shared_exercises}
    evaluationMode={this.props.evaluationMode}
  />

  <button
    className={"toolbar-button" + (!this.props.inFullscreenMode ?
" toolbar-hide" : "")}
    id='unfullscreen-button'
    title={unFullScreen}
    onClick={this.props.toggleFullscreen}
  />
</React.Fragment>

```

Open source experience and contributions

I got introduced to open-source projects and community in the second year of my undergraduate studies. Since my first year, I kept myself involved in developing and learning about software and technologies. I have been contributing to Sugar Labs for the past few months. During this time, I have contributed to various repositories of sugar labs like **ExerciserReact** , **sugarizer** and **music blocks** and fixed documentation, bugs, UI changes, enhancements etc.

These past two months have been a great learning experience. I also created a few genuine issues and some of them were considered by mentors and were fixed by me.

These are my contributions to Sugar Labs:

Pull request / Issues links	Description	Status
#3228	Fixes in the sugarizer Palette box's search bar (PR)	Merged
#172	Drag box fixes (PR)	Merged
#1347	Description box overflowing issue (PR)	Open
#161	Title bar lacks favicon (Issue)	closed
#162	Exerciser landing page UI (Issue)	closed
#165	MCQ Quiz question alignment (Issue)	Open

and many other issues

I am actively contributing to other github repositories also for the sake of overall development of the open source community. I am also a competitive programmer active on codeforces, codechef and atcoder and this too enhances my development skills as I not only aim to fix bugs or suggest solutions rather I aim to lay down a more optimized and efficient approach to solve them.

Projects / research works :

a. WaDS: IP Watermarking Using Dated Handwritten Signature

I am the **co-author** of this research paper.

This paper presents a novel dated handwritten signature based IP watermarking technique to secure the IP cores against piracy, counterfeiting and false claim of IP ownership threats. The results reveal that the proposed scheme outperforms the related approaches without incurring considerable design cost overhead.

Status : Finished (To be published by Elsevier)

b. A decentralized approach to criticality and energy aware fog computing for remote health monitoring.

I am the **author** of this paper.

This paper basically deals with the computation of severity of the patient on the basis of data collected by various sensors on a local device like mobile and IOT devices. As the computation power of local devices are limited, hence for fast and latency free computation of severity of patients, we will be doing a medical offloading to fog servers which have higher computation power and allocation will be based on decentralized SAP based approach. Our main objective is to maximize the utility function which increases with decrease in patient cost and increases with increase in revenue to hospital.

Status : Finished (Yet to be published)

c. A machine learning approach to matching graphs of audio and graphs of visuals of a news video, so that it can be checked whether what a reporter is speaking and what visuals the audience are viewing are congruent or not.

Status : Finished

Timeline and Deliverables

	TIMELINE	DELIVERABLES	
MILESTONE 1	Week 1 May 29, 2023 - Jun 4, 2023	Basic planning of files and folder structure	Puzzle Activity
	Week 2 Jun 5, 2023 - Jun 11, 2023	Creation of dashboard for teachers	
	Week 3 Jun 12, 2023 - Jun 18, 2023	Puzzle UI creation and functions implementation	
	Week 4 Jun 19, 2023 - Jun 25, 2023	Puzzle functions implementation	
	Week 5 Jun 26, 2023 - Jul 2, 2023	Linking the puzzle UI with dashboard and necessary amendments	
	Week 6 Jul 3, 2023 - Jul 9, 2023	Necessary amendments and enhancements	
	Week 7 Jul 10, 2023 - Jul 16, 2023	Chart activity implementation - 1	Chart Activity
	Week 8 Jul 17, 2023 - Jul 23, 2023	Chart activity implementation - 2	
Week 9 Jul 24, 2023 - Jul 30, 2023	features addition - 1		
Week 10 Jul 31, 2023 - Aug 6, 2023	features addition - 2		
Week 11 Aug 7, 2023 - Aug 13, 2023	Amendments	MILESTONE 2	
Week 12 Aug 14, 2023 - Aug 20, 2023	Buffer		

	Week 13 Aug 21, 2023 - Aug 27, 2023	Buffer	
--	--	--------	--

General Questions

1. Tutorial links:

[Sugarizer Vanilla Javascript activity development tutorial](#)

[Sugarizer Vue.js activity development tutorial](#)

Video link : [Tutorial Video](#)

2. How many hours will you spend each week on your project?

Starting from May 16th until July 18th, I will be on summer break from college. During this period, I will be available for approximately 48-50 hours per week, and once college resumes, I can devote about 40-45 hours per week. With no other obligations during my summer break, I am able to dedicate the majority of my time to GSoC.

3. How will you report progress between evaluations?

My activity on GitHub will be consistent as I plan to regularly submit pull requests to Sugarizer and engage with my mentors, providing visibility of my progress to anyone in the organization. This ensures that my advancements are well-documented on GitHub. Additionally, I intend to create weekly or bi-weekly blog posts to share updates on my progress, challenges encountered, and their corresponding solutions.

4. How will it impact Sugar Labs?

Both the activities which we are trying to accomplish in this complete project will enhance the learning experience of students. It's well known that the

efficiency of learning increases when we learn in a fun manner and what can be more joyful than solving a puzzle from the topic you learned. The chart activity will make students understand and visualize the statistics and make them visualize data in a better manner. Overall we can conclude that it's highly fruitful to have both the features in Sugarizer.

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

As I discussed in the introduction part only, I like to work for noble causes. The vision with which sugar Labs is working, highly attracts me to work for it even during GSOC and after it. The satisfaction which I get after solving such issues provides me immense satisfaction and motivates me to work better in the next go. As, to conclude my answer is a big "YES".

6. Why should you select me for this project ?

A simple and crisp answer to this question, I am very confident about this project and the work which I have to do in this project. For 1 month I have been scanning the complete codebase of sugarizer and music blocks. I feel I have required necessary skills and will be a highly fit person for this project. I have also tried to implement the things which are to be implemented during GSOC (screenshots provided). I hope you will surely provide me with a chance to prove it.

I am looking forward to contributing to Sugar Labs this summer season.

Kind Regards.