

# Sugarizer Word Puzzle and Chart activity

## **Basic details:**

### **Name:**

Disha Talreja

### **Contact details:**

E-mail: [dishatalreja1202@gmail.com](mailto:dishatalreja1202@gmail.com)

Github profile: [disha1202](https://github.com/disha1202)

LinkedIn profile: [dishatalreja](https://www.linkedin.com/in/dishatalreja)

Phone: (+91) 9752311007

### **Your first language:**

My first language is Hindi but I'm proficient in reading, writing and speaking in English.

### **Location:**

Indore, Madhya Pradesh, India

### **Timezone:**

Indian Standard Time (UTC+5:30)

## **Educational background:**

Shivajirao Kadam Institute of Technology and Management

Bachelor of Technology - BTech, Computer Science

2020 - 2024

## **Experience:**

- **Software Development Intern**

HotWax Commerce · Internship

Jun 2021 - Feb 2023 · 1 yr 9 mos

Indore · On-site

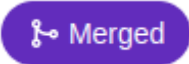
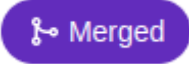
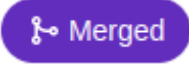
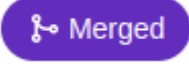
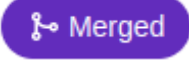
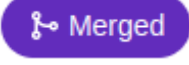
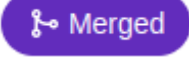

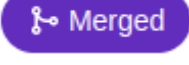
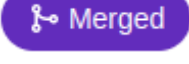
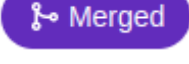
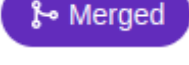
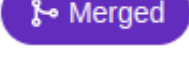
As an intern at Hotwax Commerce, I was a part of the product team that is building a Digital Experience Platform for Omnichannel Order Management Solution for Shopify Plus customers. I have more than 1 year of hands-on experience developing enterprise applications using HTML/CSS, JavaScript, Vuejs and Ionic.

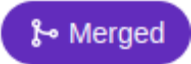
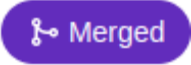
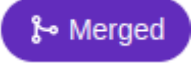
**Skills:** HTML/CSS, JavaScript, TypeScript, Vuejs, Vuex, Ionic, JQuery, Shopify, Git/GitHub

## **Previous work on open source projects:**

I am an Open Source enthusiast, I love the idea of open source software. I have worked on so many applications using open source frameworks and libraries such as [Ionic](#), [Vue.js](#), [vue-barcode-reader](#) and [PapaParse](#). Such open source frameworks and libraries have made a developers life so easy and the whole development process really smooth, without these the development process would have been much more complex. I really wish to give back to the open source community by contributing any way possible, whether it be contributing to the code base, maintaining the documentation or managing

the community. I have been an active contributor in SugarLabs since August 2022. Here are some of my contributions:

Repository	Pull Request	Status
Sugarizer	<a href="#">Replaced Bootstrap tour by IntroJs in Moon activity</a>	
Sugarizer	<a href="#">Replaced Bootstrap-tour by IntroJs in Paint activity</a>	
Sugarizer	<a href="#">Refactor: Removed unwanted code from paint activity</a>	
Sugarizer	<a href="#">Downgraded the version of IntroJs in Moon activity</a>	
Sugarizer	<a href="#">Replaced bootstrap-tour by introJs in calligra activity</a>	
Sugarizer	<a href="#">Replaced bootstrap-tour by introJs in Blockrain activity</a>	
Sugarizer	<a href="#">Replaced bootstrap-tour by IntroJs in ChatPrototype activity</a>	
Sugarizer	<a href="#">Replaced bootstrap-tour by IntroJs in ColorMyWorld activity</a>	
Sugarizer	<a href="#">Replaced bootstrap-tour by Introjs in constellation activity</a>	
ExerciserReact	<a href="#">fix: MCQ answers should be in rows</a>	
ExerciserReact	<a href="#">fix: two clicks to validate mcq answers in evaluation mode</a>	
ExerciserReact	<a href="#">replaced reactour by introJs-react</a>	
ExerciserReact	<a href="#">UI improvements</a>	

musicblocks-v4	<a href="#">feat: reusable image component</a>	
musicblocks-v4	<a href="#">feat(common): [components] add WTextButton</a>	
musicblocks-v4	<a href="#">feat(common):[components] add WCheckbox</a>	

I have also participated in Google code-in 2019 and Hacktoberfest 2023. Other than this I have also contributed to many other open source projects. Following are some of my open source contributions to different organizations:

- <https://github.com/vuestorefront/vsf-capybara/pull/682>
- <https://github.com/moja-global/community-website/pull/335>
- <https://github.com/CircuitVerse/CircuitVerse/pull/3499>
- <https://github.com/hotwax/import/pull/138>

Further details about my open source contributions can be found on my [github profile](#).

Apart from the coding contributions I have also worked as a volunteer for hands-on workshops and sessions on **Open Source, Git, and GitHub** held at various colleges in Indore, with the idea of spreading awareness about Open Source Software and Communities among students.

Also I delivered a session on **Opportunities for Students in Open Source**, where I briefed about various Open Source programs such as GSOC, GSOD, MLH Fellowship, LFX Mentorship, GSSOC (GirlScript Summer of Code), Hacktoberfest etc.

## **Convince us that you will be a good fit for this project, by sharing links to your contribution to Sugar Labs**

I have been an active contributor in SugarLabs since September 2022. I have contributed to various projects of SugarLabs with over [23 merged commits](#). Contributing to SugarLabs so far has helped me gain thorough understanding of the code base and design patterns used in various projects of SugarLabs.

- Issues: 5 ( 3 closed, 2 open)
- Pull Request: 16 ( 16 merged, 0 open)

I have completed both Sugarizer Vanilla Js activity development tutorial and Sugarizer Vue.js activity development tutorial.

- [Vanilla Js activity development tutorial](#)
- [Vue.js activity development tutorial](#)

Short video of the sample activity that I build with the help of activity development tutorial:

<https://discord.com/channels/1078051575580336249/1078054511156932658/1090655303382614077>

This has helped me understand how activities work internally and how different features in an activity are implemented in Sugarizer.

I also Explored the code base of [ExerciserReact](#) activity and fixed some of the issues:

- fix: MCQ answers should be in rows  
<https://github.com/llaske/ExerciserReact/pull/140>

- fix: two clicks to validate mcq answers in evaluation mode  
<https://github.com/llaske/ExerciserReact/pull/141>
- replaced reactour by introJs-react  
<https://github.com/llaske/ExerciserReact/pull/143>
- UI improvements  
<https://github.com/llaske/ExerciserReact/pull/147>

I have also understood the workflow of the word grid template for creating apps in [learningApps.org](https://learningapps.org) by creating multiple apps with different configurations.

## **Project Details:**

**What are you making?**

### **Chart Activity**

The aim of the project is to develop a new **Chart Activity** inspired by the [Sugar Chart activity](#) which will reproduce the current feature of the existing chart activity along with some new features such as share the activity and export chart as image.

The new chart activity will be written in Vue.js and will be using [chartjs](#) for generating different types of charts.

Chart.js is a free JavaScript library for making HTML-based charts. It is one of the simplest visualization libraries for JavaScript, and comes with the various built-in chart types.

Chartjs requires minimal markup: a `canvas` tag with an `id` by which we will reference the chart later.

We just need to provide a chart `type` (`bar`) and provide data which consists of labels (often, numeric or textual descriptions of data points) and an array of datasets (Chart.js supports multiple datasets for most chart types). Each dataset is designated with a label and contains an array of data points.

## Vertical Bar Chart

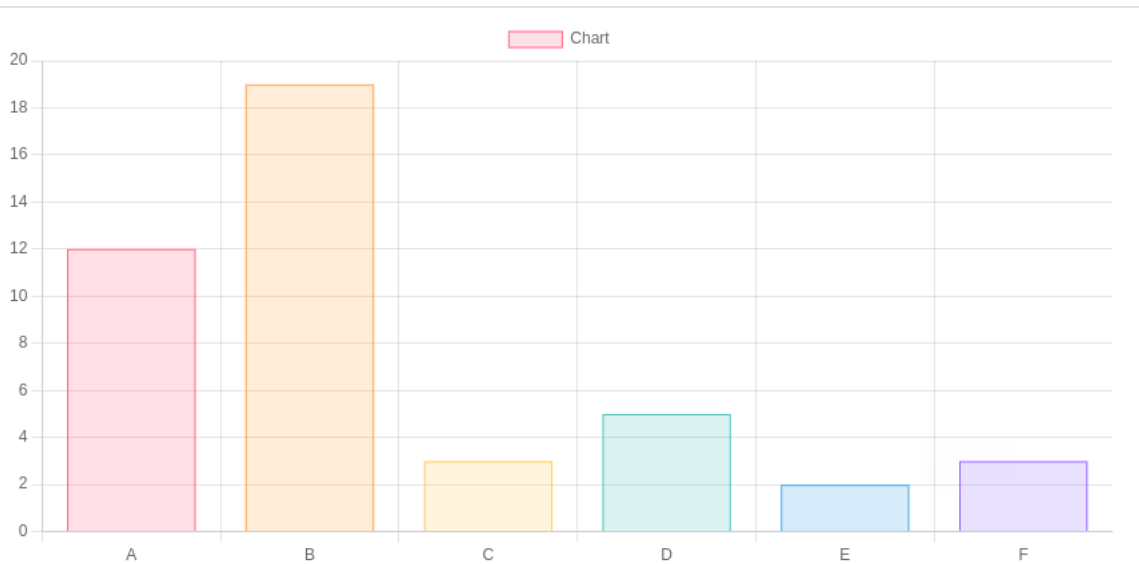
Below is the sample configuration for generating a vertical bar chart.

```
{
  type: 'bar',
  data: {
    labels: ['A', 'B', 'C', 'D', 'E', 'F'],
    datasets: [{
      label: 'Chart',
      data: [12, 19, 3, 5, 2, 3],
      backgroundColor: [
        'rgba(255, 99, 132, 0.2)',
        'rgba(255, 159, 64, 0.2)',
        'rgba(255, 205, 86, 0.2)',
        'rgba(75, 192, 192, 0.2)',
        'rgba(54, 162, 235, 0.2)',
        'rgba(153, 102, 255, 0.2)'
      ],
      borderColor: [
        'rgb(255, 99, 132)',
        'rgb(255, 159, 64)',
        'rgb(255, 205, 86)',
        'rgb(75, 192, 192)',

```

```
    'rgb(54, 162, 235)',  
    'rgb(153, 102, 255)'  
  ],  
  borderWidth: 1  
}]  
,  
}
```

Here is the chart generated from the above data.



The source code for generating the above bar chart can be found [here](#)

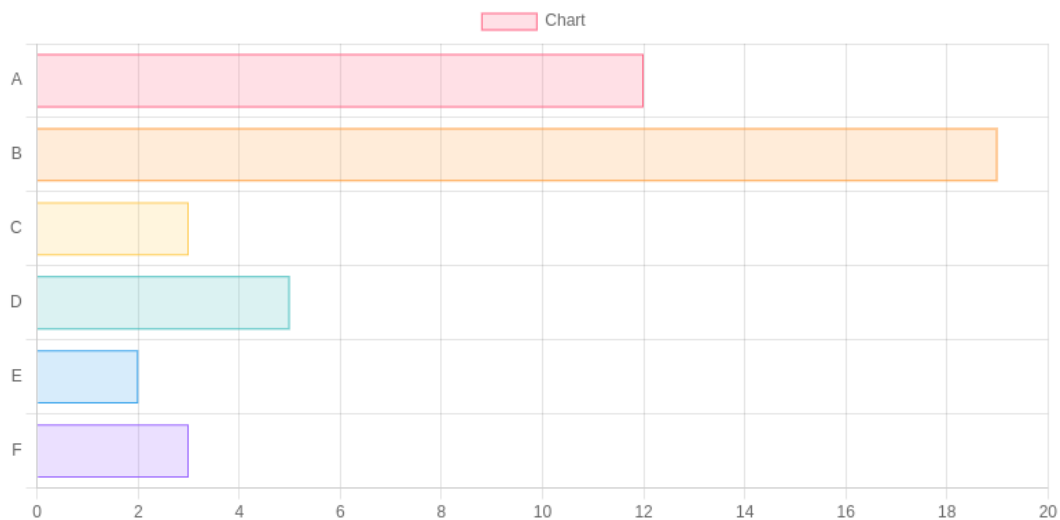
We can generate other charts such as **horizontal bar chart**, **line chart** and **pie chart** as well in a similar fashion by making a few changes in the data set.



To achieve the **Horizontal bar chart** you will have to set the `indexAxis` property in the options object to `'y'`. The default for this property is `'x'` and thus will show vertical bars.

```
{  
  type: 'bar',  
  data,  
  options: {  
    indexAxis: 'y',  
  }  
}
```

The data object will be as is for the vertical bar chart.

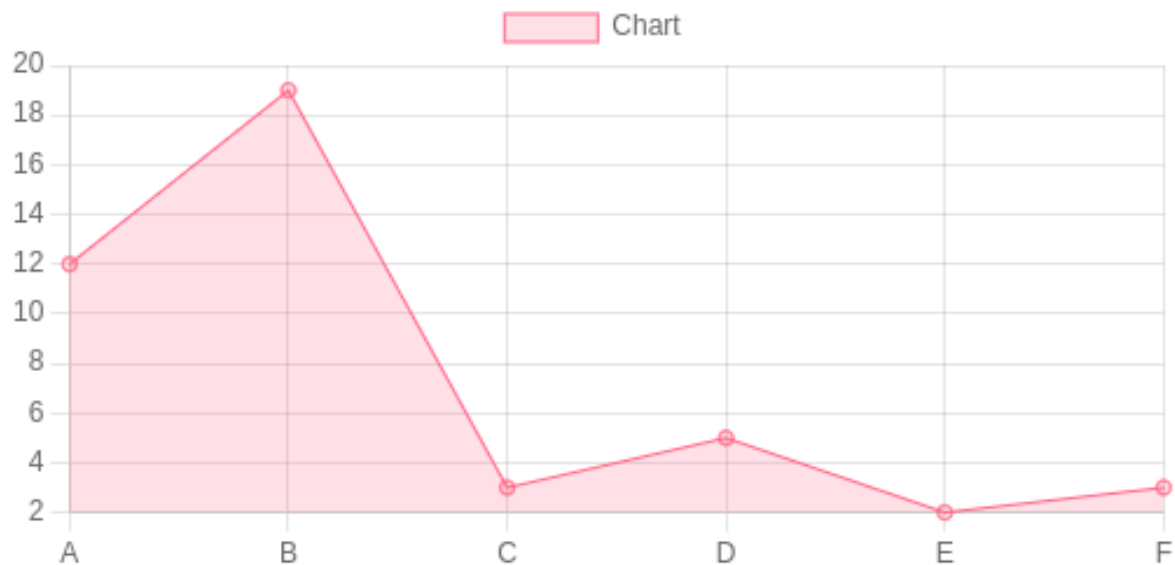


The **line chart** also accepts a similar kind of data set with a few differences, The `backgroundColor` and `borderColor` property accepts a single value instead of an array of values and there is a property `fill` when set to `true`, the background color fills the area underneath the line chart.

Sample configuration:

```
{
  labels: ['A', 'B', 'C', 'D', 'E', 'F'],
  datasets: [{
    label: 'Chart',
    data: [12, 19, 3, 5, 2, 3],
    backgroundColor: 'rgba(255, 99, 132, 0.2)',
    borderColor: 'rgb(255, 99, 132)',
    borderWidth: 1,
    fill: true
  }]
}
```

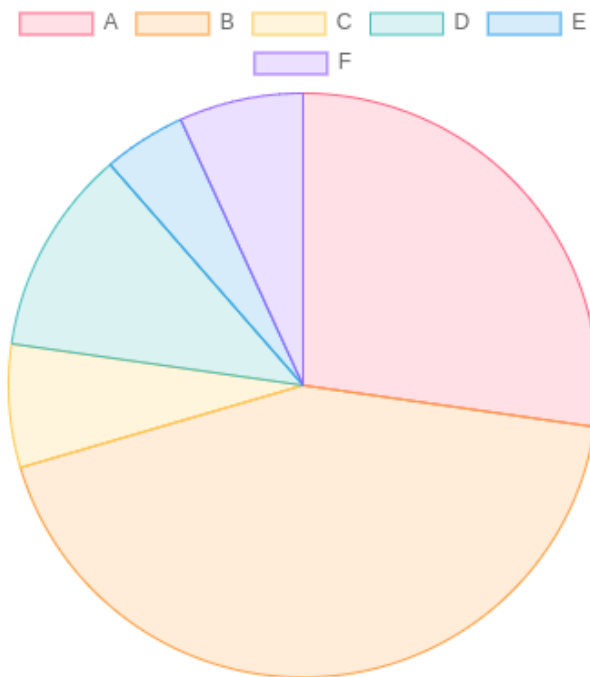
Line chart generated by the above configuration



The **pie chart** has the same configuration as the bar chart except the fill property.


```
{
  type: 'pie',
  data: {
    labels: ['A', 'B', 'C', 'D', 'E', 'F'],
    datasets: [{
```


```
label: 'chart',
data: [12, 19, 3, 5, 2, 3],
backgroundColor: [
  'rgba(255, 99, 132, 0.2)',
  'rgba(255, 159, 64, 0.2)',
  'rgba(255, 205, 86, 0.2)',
  'rgba(75, 192, 192, 0.2)',
  'rgba(54, 162, 235, 0.2)',
  'rgba(153, 102, 255, 0.2)'
],
borderColor: [
  'rgb(255, 99, 132)',
  'rgb(255, 159, 64)',
  'rgb(255, 205, 86)',
  'rgb(75, 192, 192)',
  'rgb(54, 162, 235)',
  'rgb(153, 102, 255)'
],
borderWidth: 1,
}]
}
```







The values of `backgroundColor` and `borderColor` can be replaced with `currentenv.user.colorvalue.fill` and `currentenv.user.colorvalue.stroke`

We can implement a `chartConfig` data property which will store the configuration of the chart. The data structure of the `chartConfig` property will be similar to the configuration used for generating the chart.

Whenever a user clicks on the **add button**  and enters the **label** and **value**, we push the value of the label in `chartConfig.labels` array and the value corresponding to that label in `chartConfig.datasets[0].data` array and also push the `currentenv.user.colorvalue.fill` value in `chartConfig.datasets[0].backgroundColor` array and `currentenv.user.colorvalue.stroke` value in `chartConfig.datasets[0].borderColor` array.

Similarly, we can have another data property `selectedValue` which will store the index of the selected value. When the user clicks on remove button , we can use the `splice()` method of array which takes the index of the element to be removed and the number of elements you want to remove on `chartConfig.labels`, `chartConfig.datasets[0].data`, `chartConfig.datasets[0].backgroundColor` and `chartConfig.datasets[0].borderColor` arrays to remove the selected value.

Next, to change the type of the chart, whenever the user clicks on any of

these buttons , ,  or  We can implement separate methods for each one of them which are called on click events and will update the `chartConfig` data property as per the type of chart.

## Config options

For Changing the Horizontal and Vertical labels in the configs toolbar



In Chartjs for adding labels to the Horizontal and Vertical axes you need to add the below configuration. The below sample will add `xAxis` and `yAxis` as the label of Horizontal and Vertical Axes.

```
options: {
  scales: {
    x: {
      display: true,
      title: {
        display: true,
        text: 'xAxis'
      }
    },
    y: {
      display: true,
      title: {
        display: true,
        text: 'yAxis'
      }
    }
  }
}
```



We can update the `text` property of the x and y axis for changing the horizontal and vertical labels.

For changing the color we can simply update the `backgroundColor` and `borderColor` properties of the chart configuration.

### Share the activity

The activity can be shared with multiple users with the help of `SugarPresence`.

The `presence` framework provides real time communication between a set of clients. To do that the framework is based on the [publish/subscribe](#) pattern. Every client could create one or more topics. Other clients could

subscribe to these topics and everyone could publish messages on a topic. When a message is published on a topic, only clients connected to this topic receive the message.

In the context of Sugarizer, clients are Sugarizer App/WebApp connected to the Server. One topic is a shared activity. The Sugarizer Server is responsible to keep the list of topics and clients and dispatch messages to clients subscribed to topics. So the server is the central point and in fact, clients communicate only with the server.

If the activity is shared with the network (i.e. `SugarPresence.isShared()` is `true` which tells if presence is initialized and the activity is shared), we call the `sendMessage()` method. As its name implies, `sendMessage()` is the method to send a message to the server. The parameter is the message you want to send. The content of the `message` parameter will vary as per the action performed by the user.

For example, in case of adding a value we can split the message into two parts: information about the user that sent the message and the content, which contains the chart configuration. The user info is obtained from the `SugarPresence` component using the `getUserInfo()` call: it will retrieve an object with `name`, `networkId` and `colorvalue`.

```
var message = {  
  user: this.SugarPresence.getUserInfo(),  
  content: this.chartConfig  
}  
  
this.SugarPresence.sendMessage(message);
```

There are 2 important events that you need to handle from the SugarPresence instance: `data-received` and `user-changed`.

We will call the `onNetworkDataReceived()` method on the `user-changed` event.

The `onNetworkDataReceived` method is called each time data is received from any other connected user on that activity.

```
onNetworkDataReceived(msg) {  
  
    this.chartConfig = msg.content  
  
},
```

We add the message content (i.e. the `chartConfig` of the user sending the message) to our `chartConfig` property.

If a user joins the activity, then the previous state of the activity i.e. the chart should be visible. To handle this, we will use the `user-changed` event. We'll call the `onNetworkUserChanged()` method on the `user-changed` event.

```
onNetworkUserChanged: function(msg) {  
  
    if (this.SugarPresence.isHost) {  
  
        this.SugarPresence.sendMessage({  
  
            user: this.SugarPresence.getUserInfo(),  
  
            content: this.chartConfig  
  
        });  
  
    }  
};
```



```

}
},

```

The `message` parameter here is sent automatically by the server when the subscriber's list for a shared activity has changed. You will receive in the message a `move` field telling if the user has joined (the `move` value is 1) or left (the `move` value is -1). And you will receive in the `user` field the message information (`name`, `networkId` and `colorvalue`) about the user.

This message is useful to display a list of users currently connected, and for example displaying this list.

The idea is to identify the host for the share. When a new subscriber joins the share, the host - and only the host - sends to the new subscriber a message with the current chart state.

### Export chart as an image

To export the chart as an image, the `HTMLCanvasElement` has a special method `toDataURL()` which returns an encoded data URI representing the image in the specified format (defaults to PNG).

#### Syntax:

```
canvas.toDataURL(type, encoderOptions);
```

#### Parameters:

`type` (optional)

- It indicates the type of image format.
- It will have the value of type string and is an optional parameter with default format type value "image/png".

encoderOptions (optional)

- It accepts a number between 0 and 1 indicating the image quality to be used when creating images using file formats that support lossy compression (such as image/jpeg or image/webp).
- It will use the default quality value which is 0.92, if this option is not specified or if the number is outside the allowed range.

### Return value:

It returns a string containing the requested data URL.

### Example:

```
<canvas id="canvas" width="640" height="360"></canvas>
var canvas = document.getElementById("canvas");
var dataURL = canvas.toDataURL();
```

## Exerciser Activity

Add a new template Word Puzzle for Exerciser activity. 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 exercise will have the following data structure:

```
let exercise = {
```

```
searchedWords: [  
  {  
    word: '',  
    hint: {  
      type: "text/image/audio/video/text-to-speech",  
      data: ""  
    }  
  }  
],  
scores: [],  
thumbnail: "",  
times: [],  
title: "",  
type: 'puzzle',  
userLanguage: 'en'  
}
```

The searchedWords array will contain the list of all the words. Each searchedWord will contain a word property of type string, which will store the word to be searched in the puzzle.

It will also contain a hint object, which will have two properties i.e. type and data. The type property will store the type of the hint which can be of type text, image, audio, video, text-to-speech. The data property will store the value of the hint in the form of a string.

The Word Puzzle Template will have the following features:

- Show/hide the searched words.

- Show/hide the hints.
- Case of the letters.

## What technologies (programming languages, etc) will you be using?

### Chart Activity

I will be using the following tech stack for the development of chart activity:  
HTML/CSS, JavaScript, Vue.js, chartjs, intro.js

### Word Puzzle Exercise

I will be using the following tech stack for the development of word puzzle exercise: HTML/CSS, JavaScript, react, intro.js-react

### Timeline

<u>Days</u>	<u>Plan</u>
<b>Pre Gsoc period</b>	<ul style="list-style-type: none"> <li>• Understand the sugarizer core architecture.</li> <li>• Familiarize myself with the current implementation of ExerciserReact activity.</li> <li>• Stay connected with the community and contribute to SugarLabs.</li> </ul>
<b>Community Bonding period</b>	<ul style="list-style-type: none"> <li>• Discuss about the features of the chart activity and word puzzle exercise.</li> <li>• Go through the documentation of chart.js</li> <li>• Explore the tech stack required for the project</li> <li>• Study the design and development patterns of sugarizer and ExerciserReact</li> </ul>
<b>Week 1</b>	<ul style="list-style-type: none"> <li>• Start working on the chart activity.</li> <li>• Setup the environment for the development of chart activity.</li> <li>• Implement the basic UI with add and remove</li> </ul>

	<p>buttons and logic to create, update, delete data.</p>
<b>Week 2</b>	<ul style="list-style-type: none"> <li>● Implement the basic logic to generate charts using the data entered by the user.</li> <li>● Implement functionality to generate different types of charts from the toolbar.</li> </ul>
<b>Week 3</b>	<ul style="list-style-type: none"> <li>● Continue working on functionality to generate different types of charts from the toolbar.</li> <li>● Implement features to change chart color, line color, horizontal and vertical labels.</li> </ul>
<b>Week 4</b>	<ul style="list-style-type: none"> <li>● Implement feature to format tick font and label font</li> <li>● Implement feature to export chart as an image</li> </ul>
<b>Week 5</b>	<ul style="list-style-type: none"> <li>● Add functionality to read data from existing activities such as measure and stopwatch</li> <li>● Add support for multiple users with the help of sugar presence</li> </ul>
<b>Week 6</b>	<ul style="list-style-type: none"> <li>● Implement functionality to retain the data when the user stops the activity using sugar journal.</li> <li>● Clean code, improve logic, test the workflow of the whole activity and fix bugs in case any.</li> </ul>
<b>Week 7</b>	<ul style="list-style-type: none"> <li>● Start working on word puzzle exercise, start with building the static UI for puzzle template.</li> <li>● Implement functionality to take input from the user in different formats, discuss and design the data structure for storing the values.</li> </ul>
<b>Week 8</b>	<ul style="list-style-type: none"> <li>● Continue working on functionality to take input from the user in different formats.</li> <li>● Implement some basic features such as choose a different thumbnail from journal, add and remove words, add the title of the activity, disable the finish and test exercise buttons based on certain conditions.</li> </ul>

<b>Week 9</b>	<ul style="list-style-type: none"> <li>● Implement logic to generate the puzzle grid from the input words.</li> <li>● Implement logic to show/hide searchWords in the exercise.</li> <li>● Implement code to handle the user inputs in the word puzzle play mode and highlight the searchWord in puzzle.</li> </ul>
<b>Week 10</b>	<ul style="list-style-type: none"> <li>● Continue working on logic to handle the user inputs in the word puzzle play mode and highlight the searchWord in puzzle.</li> <li>● Make the finish exercise and test exercise buttons functional.</li> </ul>
<b>Week 11</b>	<ul style="list-style-type: none"> <li>● Work on the sample word puzzle exercise.</li> <li>● Add support to edit and delete the puzzle activities.</li> </ul>
<b>Week 12</b>	<ul style="list-style-type: none"> <li>● Clean code, improve logic and test the workflow of the new features implemented and resolve bugs in case any.</li> </ul>

### **How many hours will you spend each week on your project ?**

I will be working 15 - 20 hours per week, the working hours and the number of hours are flexible, and can be adjusted as per the project requirement.

### **How will you report progress between evaluations ?**

I plan on making weekly reports which will contain the initial tasks planned for the week, the tasks completed throughout the week and blockers in case any.

I also plan to share my weekly progress in the form of blogs on platforms like medium.

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

I am planning to continue contributing to SugarLabs after the program ends. I really like the idea of providing a good learning experience in the form of interactive and fun activities and would love to contribute to the cause. It will also help me explore and work on different technologies that are used in SugarLabs for development.

