SUGARIZER STORY ACTIVITY

Basic Details:

Full Name: Saurabh Gupta

Email: saurabh_g@me.iitr.ac.in (Institute Email),
saurabhguptajpr@yahoo.in (Personal Email)

GitHub Username: saurabhhere

IRC Nickname: saurabhhere

Languages: Hindi (Native), English

Location and Timezone: Jaipur, Rajasthan, India (UTC +5:30)

Institute: Indian Institute of Technology, Roorkee

Degree: Bachelor of Technology (B. Tech)

Major: Production and Industrial Engineering

Expected Graduation Year: 2023

Resume: Resume

Skills: Javascript, ReactJs, Redux, NodeJs, Express, C++, Vue.js, HTML, CSS, Cordova, Socket.io, Bootstrap, Postman, Firebase, Linux, Git, Docker, MongoDB

Current Roles:

- **Developer at Information Management Group (IMG) IIT Roorkee**

Responsible for developing, enhancing and maintaining all main portals and websites of IIT Roorkee along with a team of 60 members.

- **Developer at IIT Roorkee Motorsports**

Responsible for developing, enhancing and maintaining the main website of IIT Roorkee Motorsports along with the team.

Working Hours and Contact Details:

Most active on email, github and slack. Less Active on IRC. Any other common platform can also be decided which is convenient to all.

I am flexible with my working hours. I can work anytime between:

- 10:00 (UTC+5:30) - 15:00 (UTC+5:30)
- 17:00 (UTC + 5:30) - 1:00 (UTC + 5:30)

Projects:

1. **Fan Club Portal:**

   At many times, some character (be it real or fictitious) catches our eye, and we cannot help but become their die-hard fans. Fan Club Portal is for every loyal fan out there. In this App, Users can create a chat room for any movie/series they like which can be followed by other fans logged into the portal and admin can modify chat room details and make other members admin.

   Code: https://github.com/saurabhhere/Fan-club-portal

   Live At: https://fan-club-portal.netlify.app

2. **BookMan:**

   It is an online library designed for Bibliophiles or any person who loves to read books, where people can register their books/novels and at the same time have access to all the amazing books/novels belonging to the students on the IITR campus.

   Code: https://github.com/saurabhhere/Bookman-bookborrowsystem

   Live At: https://book-man.netlify.app

3. **IIT Roorkee Motorsports Website:**

   IIT Roorkee Motorsports is the official Formula Student Team of IIT Roorkee. I along with my team members are responsible for developing and maintaining this website.

   Code: https://github.com/IITR-Motorsport/iitrms-new

   Live At: http://motorsports.iitr.ac.in

4. **VidVan:**

   This website is designed for Sanskrit learners who want to learn Sanskrit language from basics, here they can get all the free resources required to learn Sanskrit with an amazing user experience. This website **won third place** in Productathon 2021 organized by ECell IIT Roorkee.

   Code: https://github.com/saurabhhere/VidVan

   Live At: https://vidvan.netlify.app

Open Source Contributions:

I am contributing to different open source projects related to my Technical skills, not specific to any particular organization.

Contributing to Sugarlabs from Oct 2020.

Contributions to SugarLabs:

My contribution in **Sugarizer** includes:

Pull Request:

#934 (Merged): Made video responsive in Video Viewer activity

#930 (Merged): Added responsiveness in palettes in Dollar Street Activity

#928 (Merged): Added popup when image is exported in Planets activity

#923 (Merged): Implement a full 24h mode in Clock activity

#922 (Merged): Added country flag in Color My World activity

#919 (Closed): Added Tutorial in Get Things Done Activity

#916 (Merged): Removed typo from tutorial in Fototoon activity

#912 (Merged): Added export image option in Paint activity

#909 (Open): Handle canvas responsiveness in FotoToon Activity

#906 (Merged): Improved Responsiveness in Game of life Activity

#905 (Merged): Improved Responsiveness of TankOp activity

#891 (Merged): Removed Typo from Step 2 of tutorial

Issues created:

#940 (Fixed): Dollar Street Activity no longer work

#933 (Fixed): Video not responsive in Video Viewer Activity

#931 (Discussion, Closed): Add background to Sugarizer home screen

#929 (Fixed): Palettes not responsive in Dollar Street Activity

#927 (Fixed): Add a popup when image is exported in Planets Activity

#924 (Closed): No Tutorial in Get Things Done Activity

#895 (Fixed): Wrong values for inverse trigonometric function in Calculate Activity

#892 (Closed): Actual File Structure of ActivityTemplate is Different than what it is shown in Tutorial

#890 (Fixed): Typo in Step 2 of Tutorial

My contribution in **Sugarizer-server** includes:

Pull Request:

#277 (Merged): Fixed Permission error in install.md

Issues created:

#276 (Fixed): Permission denied in installing Docker Compose

My contribution in **Sugarizer-School-Portal** includes:

Issues created:

#2 (Fixed): Error 404 on README.md Links

My contribution in **Sugarizer-APK-Builder** includes:

Pull Request:

#9 (Merged): Fixed Typo in README.md

Project Details:

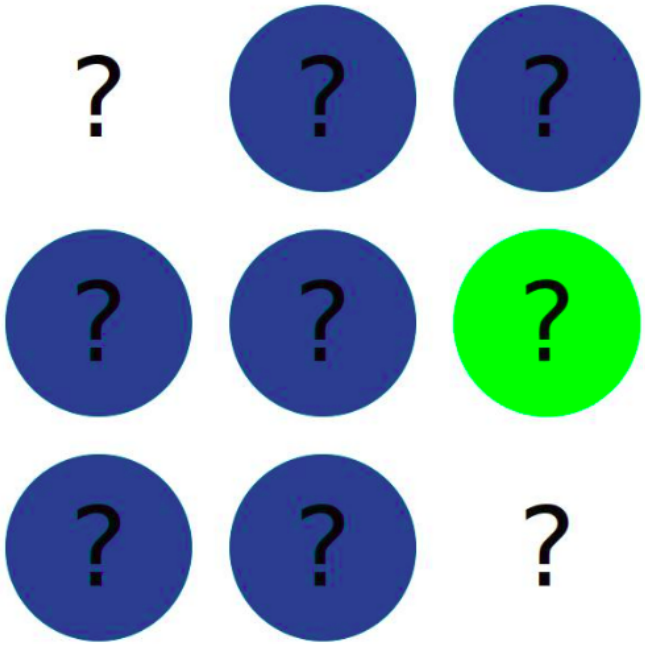
Title: Sugarizer Story Activity
Mentors: Ashish Aggarwal, Lionel Laské

What are you making?

We will make Story Activity. Story is a fun way to get children to engage in narration. We simply generate some images and the learner should try to tell a story that ties the images together into comprehensive narrative.

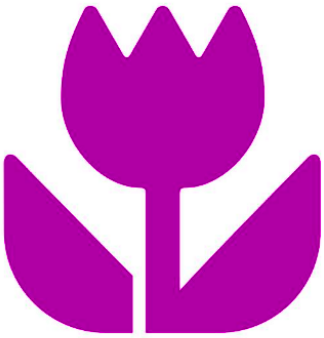
Below are the features of Story:

- Activity will generate some random images from the existing images database or newly created one.



Random colors will be generated just like in Story before image loads and we will take those colors from the environment variable using the getEnvironment method of Sugar Web library.

- Learners can either view all the images at once or view a single image at a time.



Previous Image Next Image

- Learners can restart the activity by clicking on the 'Restart button'. It will reload new images from the database.

- Learners can save the randomly generated images to journal by clicking on the 'Save as Image' button.

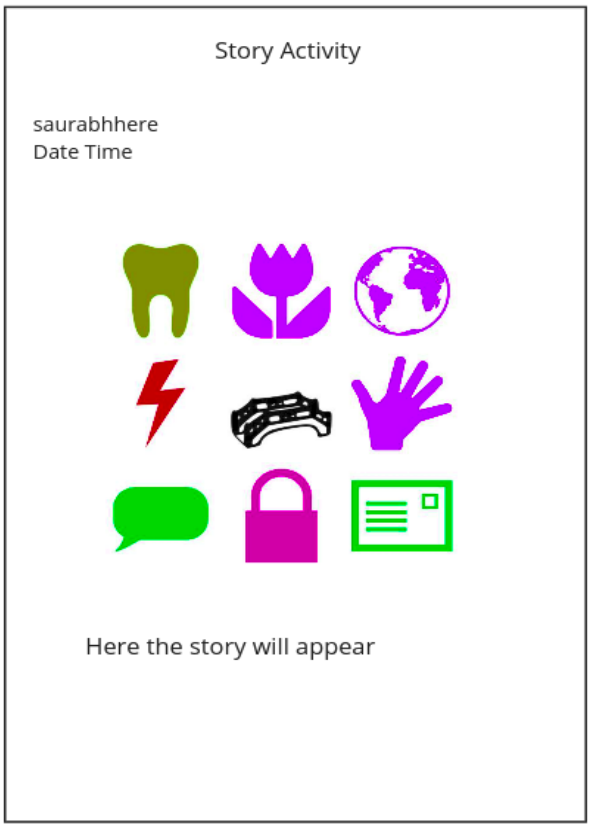
Preview of Image saved in journal:

For Multi-image mode

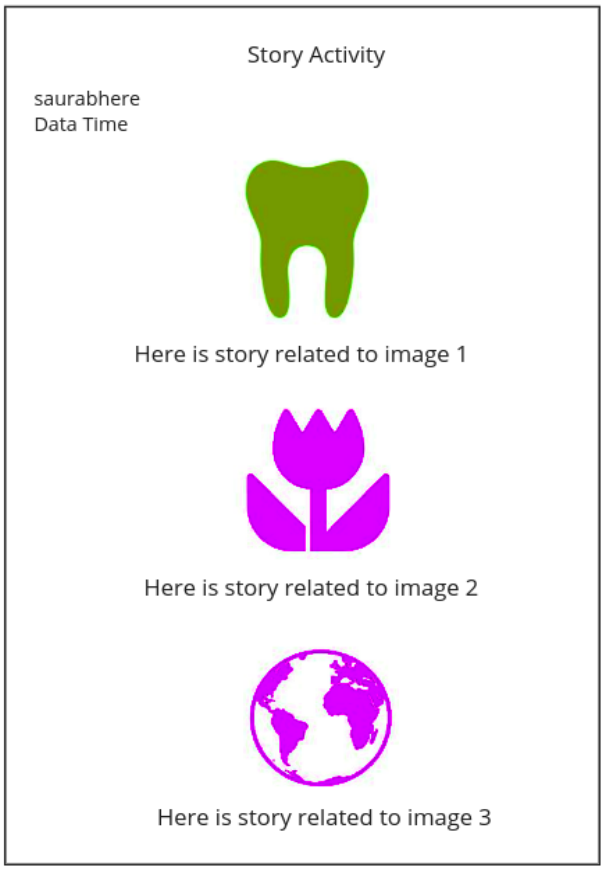
For Single-image mode

- Learners can save their written story along with their images by clicking on the ‘Save as pdf’ button.

Preview of pdf:



For Multi-image mode



For Single-image mode

- Learners can write their story on a text editor given below the images.
    - For Multi-image (Grid) mode:
        - One text editor will be provided for all images.
    - For Single-image mode:
        - Different text editor for all different images.
        - Users have to write the part of the story related to the particular image shown in their corresponding text editor.
    - All text editors will have different states, and will be stored independently.

    - We will bind the content of the editor with data in VueJs directly.
- Learners can change the number of images using the Image count slider. We will add that number of images in the images Array which will be displayed.
    - Limits of the slider will be finalized after discussing with the mentor.
- Learners can autoplay their story in Single-image mode using the “Play” button.
    - It will play a slideshow moving to the next image after a fixed amount of time.
    - Maybe we can add text-to-speech here also and it will move to the next image once the narration of a particular image finishes. (I have to discuss with the mentors regarding text-to-speech in autoplay mode)
    - Autoplay mode will be disabled in Multi-Image (Grid) view.
- Learners can style their story in the editor.
    - Change font color and background color (to highlight).
    - Update Font Style to Bold, Italic, Underline, Strikethrough using “Text Style” button
    - Increase and decrease the font size.
    - Change editor font using “Font” button.
- Learners can record their narration of the story using the record button and their narration will be stored in the journal.
- Learners can play their recording using the Play audio button.
- Activity can also narrate the story written by the users using Speak the Story button. It will work both in Multi-image and Single-image mode.
- Activity can be shared with multiple users using the presence palette.

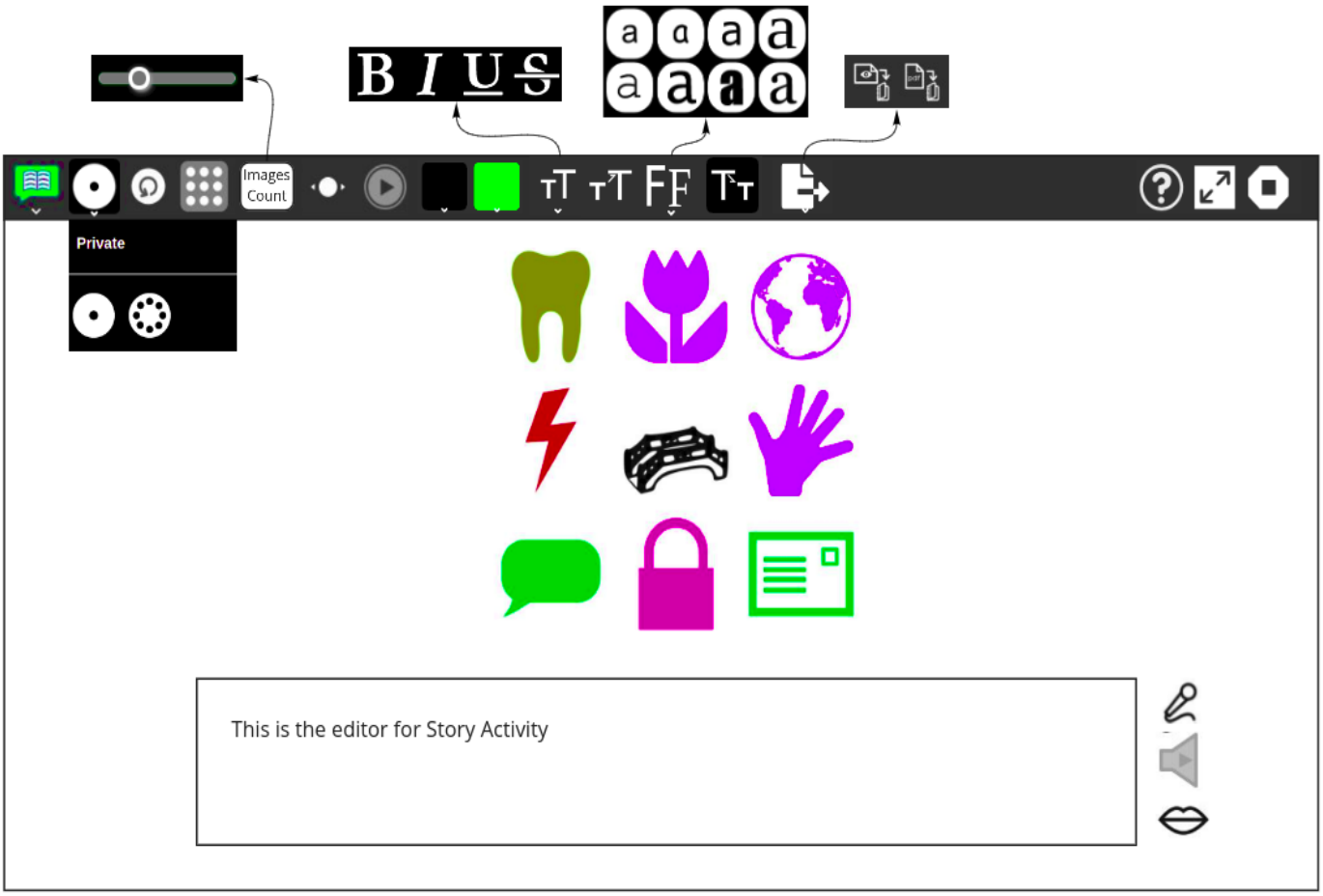
➔ In Shared Mode:

- Restart Button will be disabled.
- Also no one (including the host) can change the image count once the activity is shared.

- Activity will be localized, responsive, have a tutorial, have a fullscreen button as all other Sugarizer activities.
- Activity will support on any browser (Chrome, Firefox, Safari) and any platform (Android, iOS, Windows, Linux, MacOS) supported by Sugarizer
- On clicking the ‘Stop button’ the context will be stored in the journal and retrieve later.

Note: Context for this activity will be the state of editors and an array of images.

UI Proposed:



Buttons from left to right:

1. Activity Button
2. Presence Palette (Private and Shared Mode)
3. Restart Button
4. Multi-image/ Grid mode button
5. Image Count Slider (Limits of the slider will be decided by discussing with mentor)
6. Single-image mode
7. Play button (Autoplay the story in view images one by one mode)
8. Text Color
9. Background Color (To highlight any line or word)
10. Text Style (Bold, Italic, Underline, Strikethrough)
11. Increase Font Size

12. Font
13. Decrease Font Size
14. Export Palette
    a. Save as Image
    b. Save as pdf
15. Tutorial
16. Fullscreen
17. Stop

Buttons on bottom right from top to bottom:

1. Record
2. Play audio
3. Speak the story (Text-to-Speech)

UI for mobile:

Implementation:

- **Get Images From Database**
  - We will create image.json files in Story.activity/database which contains the names of all images present in Abecedarium database.
  - Make new XMLHttpRequest (GET) to image.json from Story.activity/database inside the component:

```
var client = new XMLHttpRequest();
var source = document.location.href.substr(0, document.location.href.indexOf
("/activities/"))+"/activities/Story.activity/database/image.json"
client.onload = function() {
    // fired when an XMLHttpRequest transaction completes successfully.
};
client.onloadstart =  function(){
    // fired when a request has started to load data
}
client.onloadend = function(){
    // fired when a request has completed, whether successfully or
    unsuccessfully
}
client.onerror = function() {
    // fired when the request encountered an error
}
client.open("GET", source);
client.send();
```

  - Pick n random names of images from the data received and add them in the images array with prefix as *"../Abecedarium.activity/images/database/"* (location where images in Abecedarium activity are stored)

    Note: Here n is the Image count that we are updating using the Image count slider.
  - Then after creating images array of required no of images from the response received we will simply use

```
<img v-for="(img,index) in images" v-bind:key="index" v-bind:src="img"/>
```

to loop through all images at once if we have to show all images in grid.

- Or we can display just one image at a time and update the img variable from the images array on previous and next click directly.
  - For this we will define two methods onNextClick and onPreviousClick inside component and update the currentActiveImage variable present in data.
- Other than using images of Abecedarium activity, we can create Story activity own database for images.
  - For this we will follow the same procedure, create another json file having the names of all images.
  - Send XMLHTTPRequest to get the names of images.
  - And then add the prefix while adding names in the images array.
  - Prefix will be the location of images in Story activity.
  - All images we use in Story Activity will be downloaded to make the activity available in offline mode also.
- Other than images we can also use SVG icons from "The Noun Project" and color them according to the user.

- **Record and Play Audio**
  - On the web:
    - In record activity, for recording and converting data to the url RecordRTC library is used, code for recording can be found here.
    - For playing the recording, HTML DOM Audio Object is used, code for playing recording can be found here.
    - HTML Audio Object has its own properties and method to play, pause and control the audio object.
  - On Android/ios:
    - For recording and playing audio in android/ios we will use cordova-plugin-media-capture which is again what we have used in Record Activity.
    - Record and play audio code in Record Activity for android/ios using cordova can be found here and here.

I had also worked on issue #932 Sound and Video recording in Record activity don't work on Android/iOS and tested the changes made in #939 through Sugarizer app built using Sugarizer APK Builder and Docker. This issue is now fixed by Lionel while updating to Cordova 10 and now record activity is working fine in android/ios also.

- **Text to Speech**
  - We can use a text-to-speech library just like we have used "mespeak" in Speak Activity.
  - In VueJs, we have SugarSpeak which is using "mespeak" directly from Speak Activity.
  - Implementation for Text to speech in Speak Activity is given here and here.

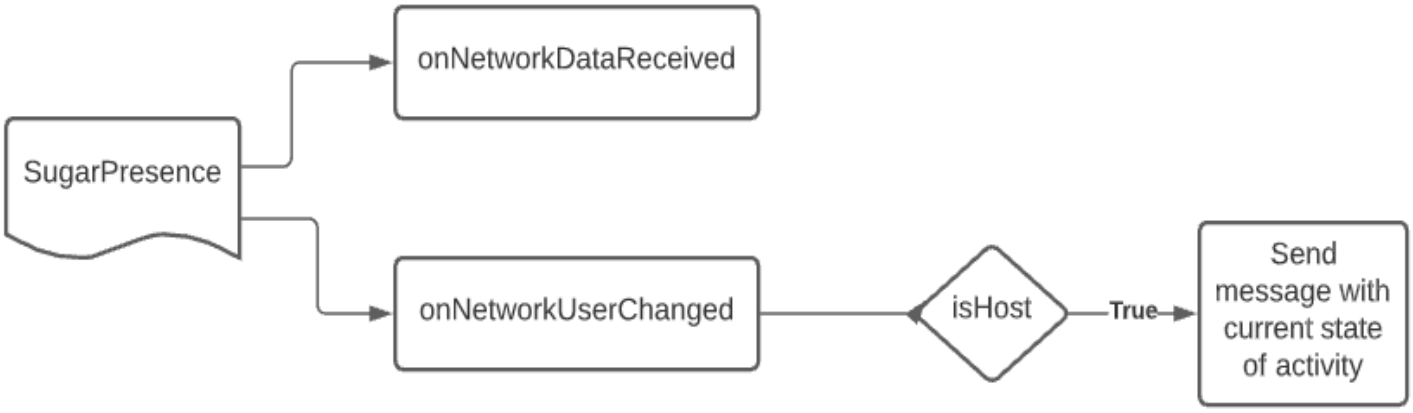
- **Save PDF and Save Image**
  - For Save as PDF, we will use html2canvas and jspdf library just like we did in Write Activity.
  - Code to save as pdf in write activity is given here.
  - If a user wants to update the story he/she can reopen the story from the journal, update it and then export it again.
  - For Save as Image, we will directly convert it into base64 string by using "toDataURL" and then store it using SugarJournal Library.

  I already have implement Save Image feature in Paint activity in PR #912 Added export image option in Paint activity.

  When export is done a popup will appear using SugarPopup Library.

- **Multiplayer mode**

  Will we use SugarPresence library to handle multi-users.



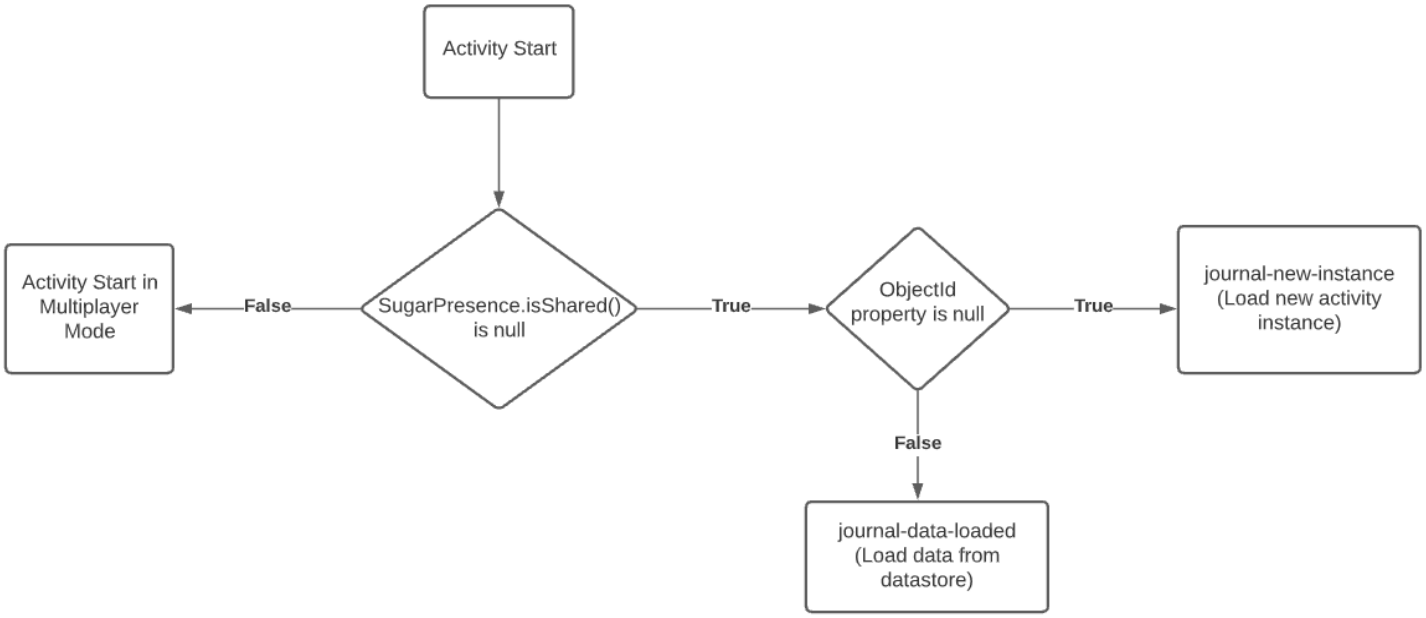
For **onNetworkDataChange** (i.e. when data received from server) we will create two actions:

'Init':
All Editors State and the Image array from the host will be shared to New User.
'updateEditor':
Update the editor State in realtime for every user whenever any user makes any change.

- **Editor tools**
  - We will directly use Quill.js methods for editor tools from Write Activity.

- **Responsiveness and Full Screen**
  - Will use CSS to handle Responsiveness and Full Screen.

- **Load Context from Datastore**
  To handle datastore we will use SugarJournal library. Events provided by Sugar Journal which we will use here are:
  - journal-data-loaded: Fired when objectId is present and data from previous context is loaded.
  - journal-new-instance: Fired when objectId is null and a new instance is created.
  - journal-load-error: Fired when any error occurred while handling journal



- **Localization**
  We will use [webL10n](#) Javascript library to handle localization. To handle localization inside components we use the 'SugarL10n' library directly.

- **Tutorial Integration**
  We use [Bootstrap tour library](#) to produce UI for the tutorial. In VueJs, we will directly use "SugarTutorial" .

We can also build custom methods other than given in pre build Sugar web libraries wherever required.
All dependent libraries will be downloaded and added in the 'lib' folder so that they can be used offline like most of the other sugarizer activities.
I have completed tutorials in both [VanillaJs](#) and [VueJs](#) and have a good idea of how Sugar Web Library works in VueJs.

We will try to keep the interface simple as it will be used by children. Final Implementations will be decided which are best to use here after discussing with the mentors.

How will it impact Sugar Labs?

Story is a fun way to get children to engage in narration. It will help children to engage themselves in learning new things. It will help to enhance the imagination and creativity of the children.

To engage a large number of children with Sugarizer we should add new activities which help them in their overall development. Addition of Story to Sugarizer will promote the reach of Sugarizer and hence more children will get involved and learn from it.

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

I will use HTML, CSS, Javascript (to write the core concept of the activity), VueJS (as a framework), some third party libraries mentioned above (libraries which have good support will be preferred)

Working Environment:

- I use Ubuntu 18.04, 20.04 and 20.10 on my different machines accordingly.
- Android Version 10 (Oneplus 7T), Version 7 (Redmi Note 4, Moto G4 Plus) for Cordova
- Visual Studio as a code editor
- Sugarizer APK Builder for building Sugarizer apk for Android.

Timeline:

Project Timeline

Start Date - End Date	Tasks
May 17 - May 26 (Week 1) **Community Bonding Period**	- Get familiarised with the codebase - End Term Examination (Break)

May 27 - May 30 (Week 2)	- Explore the technologies and libraries that will be used in the project - Interact with the my mentors and other org members - Discuss about the UI of Story activity.
May 31 - June 6 (Week 3)	- Discuss about the limits of slider and text-to-speech in autoplay mode. - Discuss about images databases we can use other than an abecedarium activity database. - Finalize the feature and UI of the Story activity
June 7 - June 13 (Week 4) **Coding Period**	- Create new Story activity in Sugarizer and structure the project - Implement UI - Build a toolbar - Create palettes for Editor tools and export buttons. - Create Editor to write story - Start implementing single player mode
June 14 - June 20 (Week 5)	- Set up getting images from the database. - Implement Multi-image Button, Single-image button and Image Count Slider with their features. - Save Images after making XMLHTTPRequest in Images array and display them in the activity accordingly.
June 21 - June 27 (Week 6)	- Implement Random color before Image load feature. - Integrate Reset button - Integrate Autoplay button - Test full images implementation
June 28 - July 4 (Week 7)	- Add Record and play audio features in the activity. - Add Speak Story (Text-to-speech) feature in the activity. - Test Record feature on different platforms.

July 5 - July 11 (Week 8)	- Finalize the previous work before the evaluations
July 12 - July 18 (Week 9)	**- Evaluations:** **- Display images in grid mode and single image mode from the image database.** **- Change number of images using Image count slider** **- Able to record and play story narration.** **- Text-to-speech feature using Speak story button.**
July 19 - July 25 (Week 10)	- Integrate editor toolbar (Text Color, Background Color, Text Style, Increase Font Size, Font, Decrease Font Size) - Implement Export as PDF and Export as Image.
July 26 - Aug 1 (Week 11)	- Implement features to share the activity via presence and implement multiplayer mode. - Test the multiplayer feature
Aug 2 - Aug 8 (Week 12)	- Implement localization using SugarL10n - Integrate tutorial - Test complete activity on different platforms - Try to fix any remaining bugs in the activity
Aug 9 - Aug 16 (Week 13)	- Finish any remaining work before final evaluations. - Review the final documentation - Prepare for final evaluation
Aug 16 - Aug 23	**Final Evaluations** **- Fully functional story activity with shared mode.** **- Documentation completed** **- Tested on different platforms**

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

I will have to take a break of 10 days from 18th May to 28th May for my end semester examinations. After that I have my summer vacations and I have no other commitments during my summer vacation (i.e. June 1 to July 30). I can easily target 40-45 hours per week. And can extend if required. From July 31, our college reopens and I will have classes so I can devote 30-35 hours per week and extra hours on weekends.

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

After GSOC ends, I will continue to contribute to Sugarizer and keep fixing issues and implement new ideas to enhance it. I will be an active contributor and will guide new contributors.
Also I will explore the Javascript activities in Sugar which require fixes as discussed with James on Sugar-devel (Vol 149, Issue 22) and try to fix them.
The aim will be the growth of the organization always.

Motivation:

What is your motivation to take part in Google Summer of Code?

I have used a lot of open source technologies but never had a chance to work on software that has a great effect on millions of people around the world. GSOC provides a very good opportunity to work on such projects along with community that leads to learning and growth. In this way I will also be able to give back to the community.

Why did you choose Sugar Labs?

Sugar Labs is a community that focuses on providing an open-source learning platform for children and provides equal opportunities to them to learn, irrespective of their background.

My journey with Sugar Labs from my first Contribution in Oct 2020 has been great. I have learned a lot in every discussion I had with the community in any issue or Pull Request or on mailing list. It was only possible because of the constant feedback from the community.

Sugar Labs provides me the way by which I can learn and help many other children to learn something new in an enjoyable way. When you write code that will help so many children to grow gives me a sense of satisfaction.

Why do you want to work on this particular project?

Creating a new activity which enhances children's imagination is a very interesting project I would like to work upon. Also the technology stack required in this project aligns with my skills and my past work.

I have worked on different Sugarizer activities and am familiar with their codebase which surely will help me here to reduce the complexity.

What are your expectations from us during and after successful completion of the program?

During the program, I expect at least weekly feedback and suggestions from the mentors so that I can discuss new ideas on the project which I can implement and learn which will eventually help me to add value to the community and help others. After the successful completion of the project, I plan to continue my contributions to Sugar Labs as I am fortunate to have met this vibrant community of motivated developers and creators. Even before GSoC has begun, I have learned so much, which only makes me look forward to all the more things I can learn Post GSoC.