GSoC'24 Project Proposal
Sugar Labs: Sugarizer VueJS App

Korada Vishal

Basic Details

Full Name:

Korada Vishal

Email and GitHub Username:

Email: korada.vishal.phe22@itbhu.ac.in
GitHub Username: @Vishalk91-4

Your First Language:

My first language is Hindi but I am proficient in English and I am learning French.

Location and Timezone:

Location: Varanasi, Uttar Pradesh, India
Timezone: Indian Standard Time (UTC + 5:30)

Education Details

I cleared the toughest undergraduate examination in India, and the second toughest exam in the world IITJEE and got admission in Indian Institute of Technology (BHU) Varanasi. I started programming in my Freshman year of college. I have got knowledge over a lot of Computer Science topics through the Institute courses of my department like Computer Architecture,

Data Structures, Operating System and Web Development. I have also been contributing to Open Source projects since the latter half of my Freshman year. I am recently learning Gen AI and trying to build some projects in it.

Share links, if any, of previous work on Open Source Projects

I got into GSSoC'23, an Open Source Program and started my journey there. Since then I have evolved as a developer by having a **MEVN** stack and knowing about Cloud Technologies.

I have been contributing to **Sugar Labs** for the past **7 months.**
In this period, I have contributed to various repositories in Sugarizer main project from sugarizer, to sugarizer-doc and sugarizer-server
Contributing from UI Changes, bug fixes, typo fixes, documentation, enhancements and Migrating activities from webL10n to i18next
These are my **contributions** to **Sugar Labs** in Sugarizer

In **Sugarizer Repository**

Pull Request Link	Description	Status
#1360	Fixed the UI of Next and Done Button	Merged
#1522	Fixed Resize issue in description of activities	Merged
#1383	Migrated Abacaderium from webL10n to i18next	Merged
#1385	Migrated Blockrain from webL10n to i18next	Merged
#1386	Made Fototoon Activity Buttons interactive	Merged

#1391	Migrated Chat Activity to i18next	Merged
#1394	Changing the position and color of spinner in QR Activity	Merged
#1400	Replaced webL10n to i18next in Constellation	Merged
#1401	Replaced webL10n to i18next in Fototoon	Merged
#1405	Replaced webL10n to i18next in Clock	Merged
#1408	Replaced webL10n to i18next in Falabracam	Merged
#1423	Migrated Speak Activity to i18next	Merged
#1450	Migrated Sprint Math Activity to i18next	Merged
#1456	Made Grammatical Changes to README file	Merged
#1465	Migrated Speak Activity to i18next	Merged
#1467	Fixed Typos in CODE_OF_CONDUCT .md and credits.md file	Merged
#1473	Migrated GridPaint Activity to i18next	Merged
#1477	Migrated TankOp Activity to i18next	Merged
#1495	Migrated Record	Merged

	Activity to i18next	
#1502	Migrated GetThingsDone to i18next	Merged
#1519	Ctrl + a/A, ctrl + backspace functionalities in v2	Merged
#1522	Using Same button for Next and Done in v2	Merged
#1548	Fixed selection of popup-title in language settings	Merged

In **Sugarizer-doc Repository**

Pull Request Link	Description	Status
#37	Made Language Dropdown visible in Sugarizer website	Merged
#39	In Explore activities, no. of page selection possible now	Merged
#44	Sort By Alphabetical order, in Mobile version present now	Merged
#53	Copyright automation every year	Merged
#55	Updated All Logos in	Merged

	Sugarizer webpage	
#45	Added Carousel for displaying blogs	Open

I have made a total of 25 issues in Sugarizer. All of them can be found here link

Out of the above contributions, I have made significant contributions in porting 14 activities to i18next and now I am completing the TODO for v2 of sugarizer

I have worked on my personal projects for learning purposes, creating a Linux Foundation Mentorship website for easy selection of Organizations for contributors Backend Repo Frontend Repo website link
Also, I have participated in Hacktoberfest '23 and Advent of Code '23, completing both successfully. Following are some of my open-source contributions to different organizations.

- https://gitlab.com/gitlab-org/gitlab/-/merge_requests/139437
- https://github.com/illacloud/illa-builder/pull/2228
- https://github.com/ucsc-ospo/ucsc-ospo.github.io
- https://github.com/COPS-IITBHU/cops-website

All My other contributions can be found here on my GitHub page link

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 of **Sugar Labs** since **Aug 2023**, with 25+ PRs over various repositories. With this much time spent over the codebase, I have got a nice understanding of it.

- Pull Request: (28 closed, 1 Open)
- Issues: (18 closed, 7 Open: 2 to be released)

I am willing to be a part of Sugar Labs for Google Summer of Code 2024, because I feel this matches very much to my Technical Stack with it's framework VueJS, and the motive of the organization, the connections which I believe, I can make with such a large Organization with many Apps and Activities being run everyday by thousands, the effect a change in my code could bring is what drives me to work in this organization, sticking to one only.

Prerequisites for Sugarizer VueJS Application

I possess experience in Javascript, HTML5 and VueJS framework. And, I am familiar with the codebase of Sugarizer and have fixed issues for the following in the related repositories.

I have been a MEVN developer, and have already made projects with JavaScript, HTML5 and VueJS framework. Here are some of my work

- **COPS SDG Site**: link
  I am a core team member in Club Of ProgrammerS (COPS) IIT BHU, and have worked in their SDG Site, for software development group I have enhanced the UI of the site, made it more accessible, and made the frontend components in it.
  Technologies Used: VueJS framework, JavaScript, SCSS

- **E-Summit '24**: link
  I am the Tech Team Member in the Entrepreneurship Cell of my institute and during our Flagship summit, I developed the speakers section, added UI Components like navbar and carousel, integrated UI libraries, and have also done Unit testing of components in the E-Cell Main website using Cypress
  Technologies Used: NextJS, JavaScript, TypeScript, CSS, Shadcn, SCSS

These are some of the projects, which I have listed, I have some more and over various TechStacks like Golang, did my Advent of Code '24 in Python, Golang, you can find more on my GitHub page link

I have previously contributed to a lot of Sugarizer Activities, and worked around the sugarizer website and it's server

Links to my contributions are listed here, link and in the above section.

To mention some of them :-
Fototoon
https://github.com/llaske/sugarizer/pull/1386
QR
https://github.com/llaske/sugarizer/pull/1394
FoodChain
https://github.com/llaske/sugarizer/pull/1406

I completed both the Sugarizer Vanilla Javascript activity development tutorial and the Sugarizer Vue.js activity development tutorial.

Here is the link to my repository containing those tutorials.

- https://github.com/Vishalk91-4/sugarizer/tree/pawn-issue/activities/Pawn.activity
- https://github.com/Vishalk91-4/sugarizer/tree/pawn-vanilla/activities/Pawn.activity

Video of those tutorials ➖

`Vue JS` ➖
https://discord.com/channels/1078051575580336249/1078054265517506681/1201947911022460928
`Vanilla JS` ➖

https://discord.com/channels/1078051575580336249/1078054265517506681/1217162361069244426

The Making of above tutorials have helped me a lot in understanding the making of new activities, their procedure the components to add, what additional functionality could be implemented and how would such things affect the rest of the codebase

I have also `tested sugarizer`, for `v2`, both using `file://` and also using `docker` to host it locally at `localhost:8080`

Video Link to file:// ➖
https://github.com/Vishalk91-4/sugarizer-links/blob/main/sugarizer-server.webm
Video Link to localhost:8080 ➖
https://github.com/Vishalk91-4/sugarizer-links/blob/main/file_sugarizerserver.webm

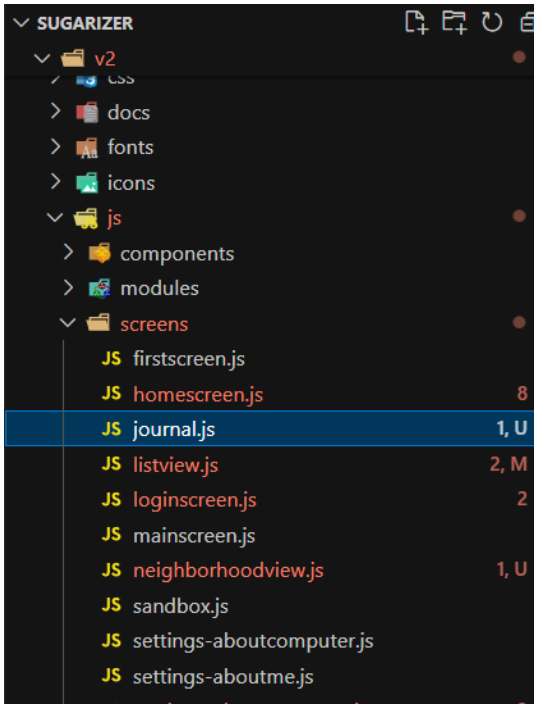
Project Details

What are you making ?

The aim of this project is to complete the implementation of Sugarizer VueJS Application. So, I will be

- Implementing the remaining screens of Neighbourhood and Journal View
- Implementing the tutorial of for all activities in latest version
- Adding Electron Components and ensuring its compatibility with the new Sugarizer VueJS interface
- Developing Sugarizer on Android, then thoroughly test to confirm functionality

I will be adding the following files in 'v2/js/screens/' directory, the **screens** remaining for migration to VueJS



- journal.js
- neighborhood.js
- tutorial.js in 'lib/' directory
- Porting electron, android in util.js and main.js

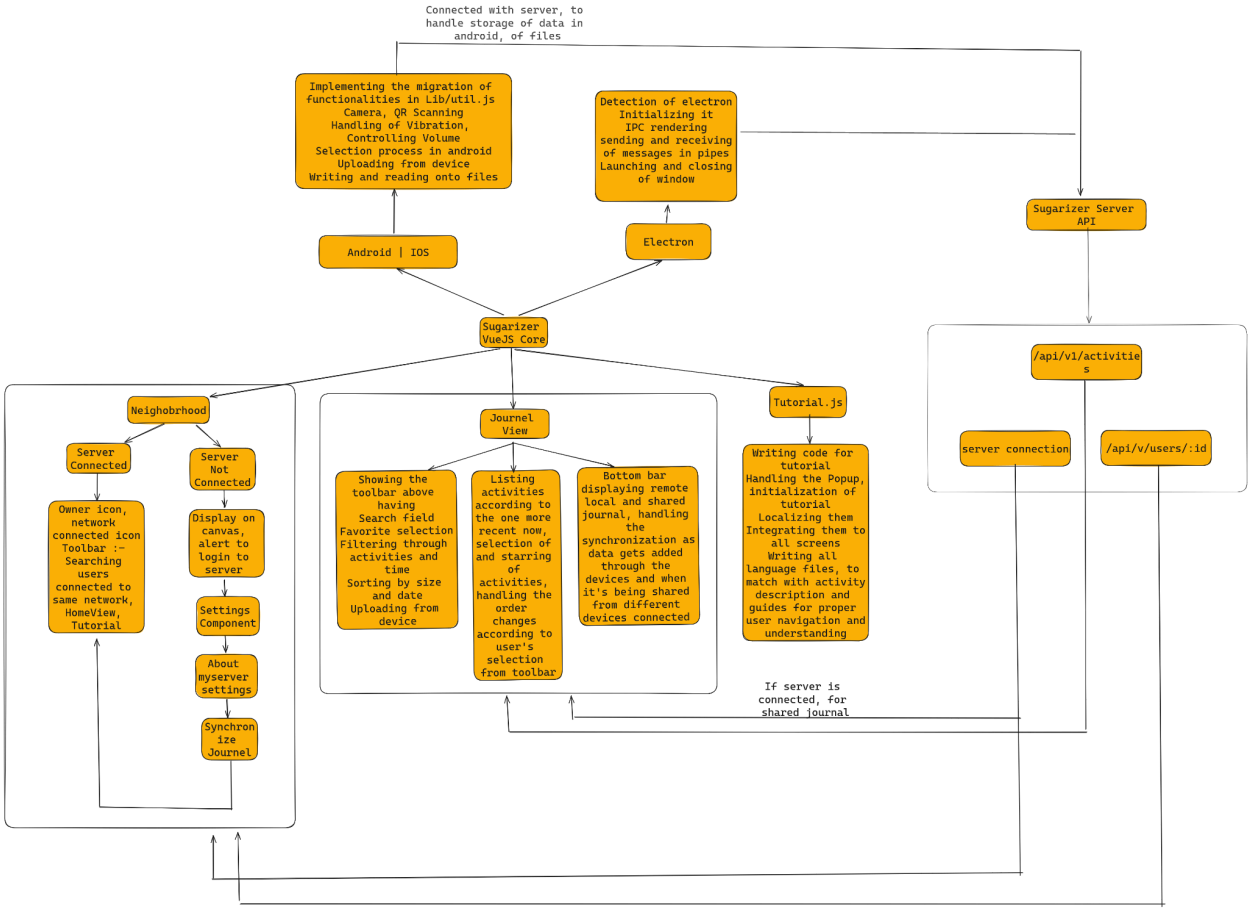
Other than this I will also be migrating these files etc and other small files which are currently based on EnyoJs to the VueJs framework.

I will also be adding these files for complete migration to VueJS

- loader.js
- audio.js (if time permits otherwise outside of GSoC Timeline), just to make the versions more streamlined and more smoother interface

The interconnection can be summarized in the image given below

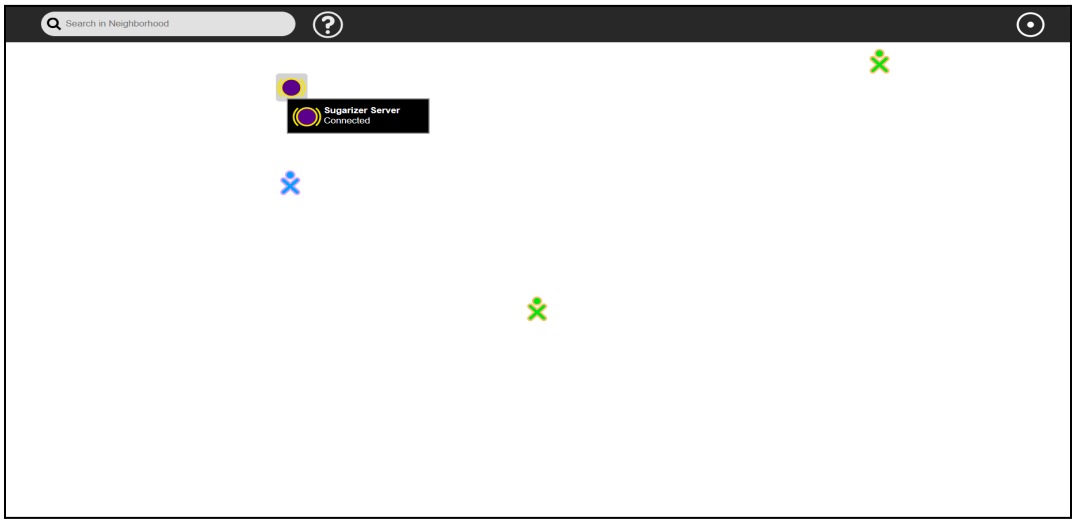
Sugarizer Project Architecture Link



Details of these files and how the changes will be written are described below:

❖ Neighborhood.js

The Neighborhood screen is going to look like this



This file will contain two classes `Sugar.NeighborhoodView` for displaying users connected to the same server and `Sugar.NeighborhoodToolbar` for filtering of networks and tutorial display on the toolbar

- ➢ In the template, there would be
  - There would be a parent `div` with class named `neighborhoodview`, and inside it there would be two `sugar-icon` component to show the owner's icon, `buddy icon` and the other one to show server's icon, showing it's connected and displaying of shared activities, for users in same network to join
  - There would be `div` inside the parent div, containing the network components like displaying the user's which are connected to the same network and removing them as soon as they are disconnected from the network in that process it will also, refresh the `journal` and `synchronize` it, so as to remove the files inside the shared network column

```
<div v-show="showNetwork">
   <div v-for="(item, index) in networkItems" :key="index" class="network-icon-
container">
      <sugar-icon
        :icon="item.icon"
        :size="constant.sizeNeighbor"
        :colorized="true"
        :classes="item.classes"
        @popupShow="showPopup(item)"
        @popupHide="hidePopup(item)"
      />
   </div>
</div>
```

- ➢ **Components**, needed, some of which are popup, connected, homescreen
- ➢ We will also be adding different `methods` like

- When the app is connected to the server, we have, `showPopup`, `showPopupTimer`, `removePopup`, `removePopupTimer` , to show the owner's buddy icon and to also show the buddy icon of all those user's connected within the same network and the removePopup one to remove them as soon as they get disconnected or logoff from the network

```
    showPopup(event) {
      this.showPopupContent = true;
      this.popupPosition = { x: event.clientX, y: event.clientY
};    document.addEventListener('click', this.removePopup);
    },
    removePopup(event) {
      if (!this.$el.contains(event.target)) {
        this.showPopupContent = false;
        document.removeEventListener('click', this.removePopup);
      }
    },
```

- When it's not connected to the server, we have `showServerPopup`, `hideServerPopup`, this methods, redirect the user to go the the settings, and in about my server, where they enter the server details, then the journal is synchronized again, so that the shared files are visible and then the buddy icon of owner's and the connected networks is visible

- `filterNetwork`, for searching the users connected to the network, once found, their buddy icon's color is highlighted, if not found all icons are turned gray

- `networkState`, this method is used to check if the owner is connected to a server or not, if connected, then shows the owner and neighborhood users along with a icon to indicate that server is connected, if it's not connected then, displaying unable to connect to server

and, showing the settings button
There would also be other functions in it like refresh, logoff, quit, when one logs off it's network then in the view, the buddy icon corresponding to it is removed and the journal is synchronized i.e. the shared files, which are removed, added when another user joins or quits the network

- `Shared Activities`, this method will show all the activities that are shared by other users within the network, which the owner can join and then shared activity is performed on those activities.

- `Cache Handling`, pseudocode for it, used
This is used to handle the data, of people who get disconnected and reconnect to the same network, and so that the data remains clean also

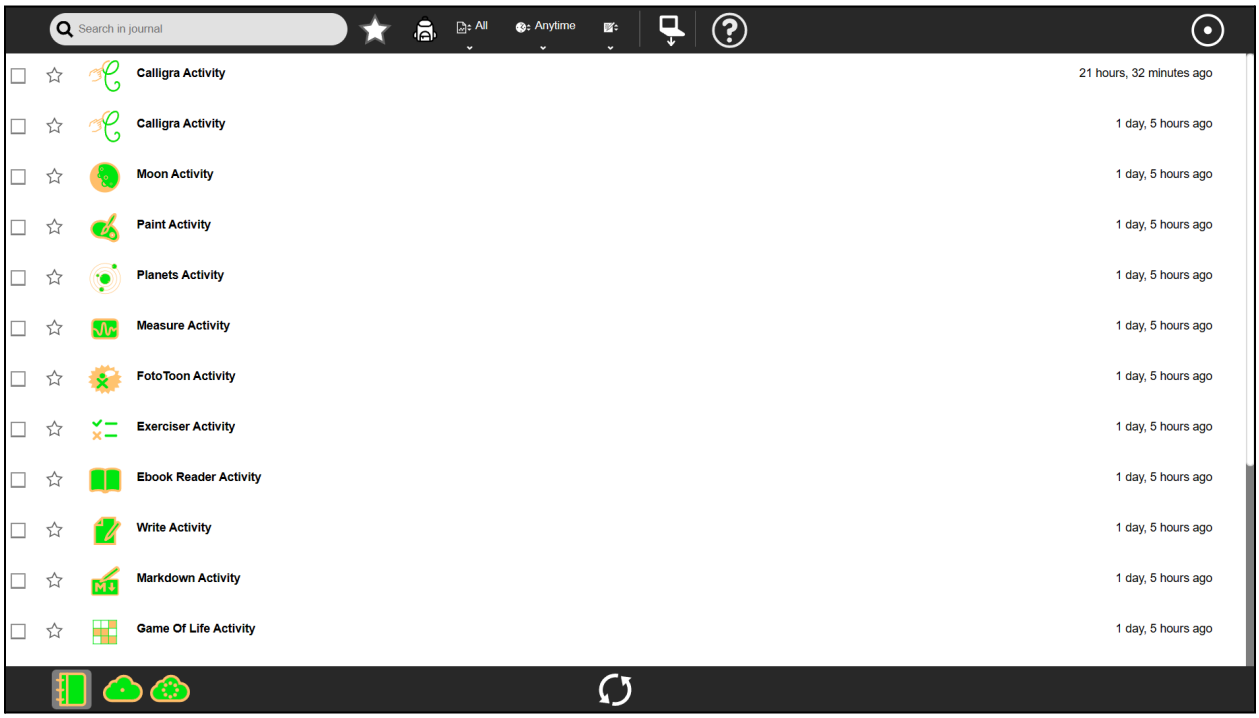
```
const clearCacheAndReload = () => {
  console.log('Clearing cache and hard reloading...');
  if (caches) {
    caches.keys().then((names) => {
        for (let name of names) caches.delete(name);
    });
  }
window.location.reload(true);
};

const checkVersionMismatch = (latestVersion, currentVersion) => {
    const latestVersionArray = latestVersion.split(/\./g);
    const currentVersionArray = currentVersion.split(/\./g);
    while (latestVersionArray.length || currentVersionArray.length) {
      const a = Number(latestVersionArray.shift());
      const b = Number(currentVersionArray.shift());
      if (a === b) continue;
      return a > b || isNaN(b);
    }
    return false;
};
```

```
onMounted(() => {
 fetch(`/meta.json?${new Date().getTime()}`, { cache: 'no-cache' })
    .then((response) => response.json())
    .then((meta) => {
      const latestVersion = meta.version;
      const currentVersion = appVersion.value;
      const shouldForceRefresh = checkVersionMismatch(latestVersion, currentVersion);
      if (shouldForceRefresh) {
        console.log(`New version - ${latestVersion}. Available, need to force refresh`);
        isLoading.value = false;
        isLatestVersionAvailable.value = false;
      } else {
        console.log(`Already latest version - ${latestVersion}. No refresh required.`);
        isLoading.value = false;
        isLatestVersionAvailable.value = true;
      }
    });
});
```

❖ journal.js

The Journal Screen is going to looks like this



This file will contain two classes `Sugar.JournalView` for displaying the various activities, and their filtering through various parameters like

assignment, favorites etc, another class `Sugar.JournalToolbar`, which will contain the tooling items like local journal, shared journal, remote journal and handle their activities

➢ In this template there would be

- There would be a parent `div` with class named `JournalView`, and inside it there would be `icon box`, `dialog box`, `Journal list`, `popup` for activities and instructions
- There would be another div with class named `JournalToolbar`, and inside it there would be `Journal` button, `remote journal` button, `local journal` and `syncing`

```
<template>
 <div id="app" class="JournalView">
    <journal-toolbar></journal-toolbar>
    <dialog-box></dialog-box>
    <journal-list :entries="journal"></journal-list>
    <popup-activity></popup-activity>
    <popup-instructions></popup-instructions>
 </div>
</template>

});
```

➢ We will also be adding different `methods` like

- `Colorized` method, all activities displayed have Colorize icons, the icons listed in journal must be colored according to the owner's icon color selected
- `Date and time` method to exactly display the time from which the activity was last opened
- `Assignment` method handling many functions related to assignments,
  showing assignments, corresponding to particular

`assignment id`,
showing assignments which are `turned in late`,
Calculating the `due date` of assignments

```
calculateDueDate() {
    const currentDate = new Date();
    const dueDate = new Date(this.entry.metadata.dueDate);
    const difference = dueDate - currentDate;
    if (difference > 0) {
      this.formattedDueDate = `Expected: ${this.formatDate(dueDate)}`;
    } else {
      this.formattedDueDate = 'Due Date Passed';
      this.showSubmitButton = this.entry.metadata.lateTurnIn;
    }
},
```

- Another method `Selection` which has functions for handling all activities like implementation of `selectall`, `update`, `remove`, `copy`, `display`, `duplication` , `unselectall` activities and then performing the functions as required by the owner which could be porting to any of other journal like `local, remote or shared` of removal of them
- `Filtering entries` method based on `activities`, like which activities, `date time` like which date and time the activities are being last opened
- Adding Popup for activities and for instructions, showing all activities and restarting them when we sync it
- Methods to copy activity content to local and into device file, into remote journal and the entries onto the device
- Inside each `journal`, handling them like updating these activities, whenever each are added and refreshing to sync them and updating them, same goes for removing them

- Journal toolbar, which has the start of tutorial, filtering and sorting of activities and their assignments
  Filtering is done on basis of assignment and favorites
  Sorting is done on the basis of time at which they were added

```
methods: {
  gotoDesktop() {
    util.vibrate();
    app.showView(constant.radialView);
  },
  filterFavorite() {
    this.favoritebutton.colorized = !this.favoritebutton.colorized;
    this.filterEntries();
  },
  filterAssignment() {
    this.assignmentbutton.colorized = !this.assignmentbutton.colorized;
    this.filterEntries();
  },
  showTypePalette() {
    this.typepalette.switchPalette(app.otherview);
  }
```

- `JournalChanged` method for checking and changing of journal from local to remote to shared, and displaying corresponding activities after fetching from datastore
  It will have different functions to display whether the journal is empty or not, displaying the activities, clearing the search
- Journal Footbar is the place from where we change the display of Journal View from user to remote to shared
- Here we first create a `method` to check from the `server` which Journal the user is in, then `display the stats` of that particular Journal
- With `Methods` for `synchronizing` them with the respective journal, and getting the session back through a small token
- We are storing the `context` in `localStorage`, the data every time of Journal

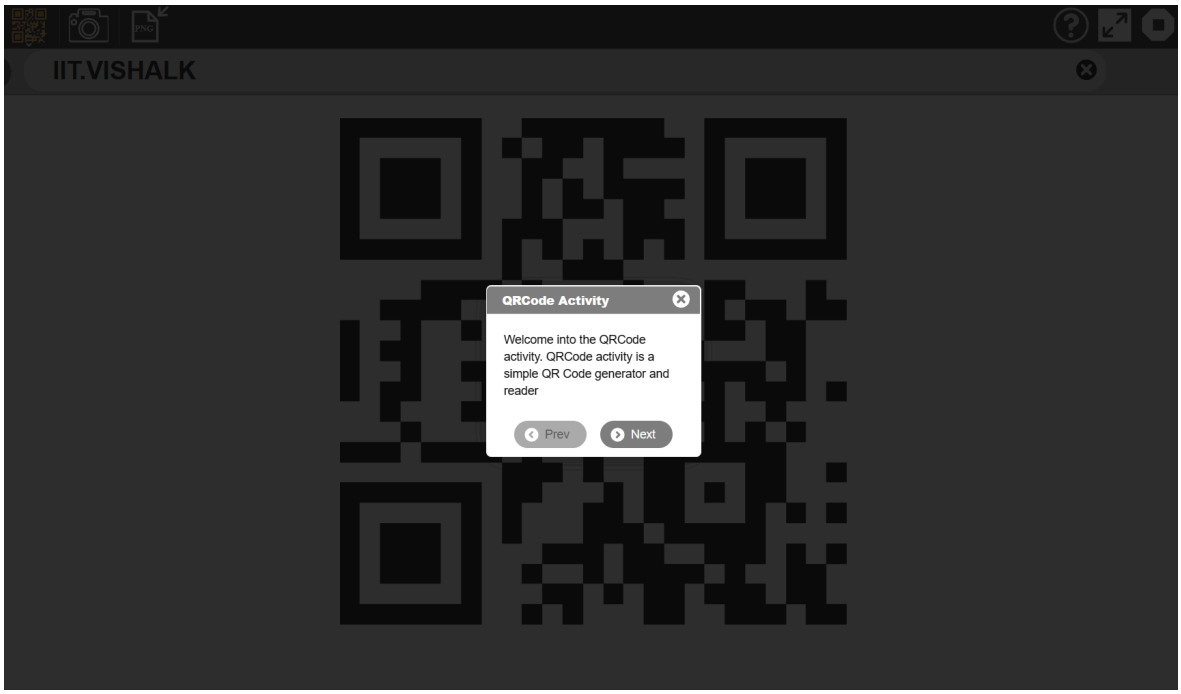
- Code for each part will be different like `updating` or `removing` data from `certain journal` will be `different` as compared to the other
- `Assignments` :- There will be a button through which the user could view the remaining Assignments
- Writing a method in Journal.js file to check whether there is an `assignment present or not`, for `displaying on the other side to the user`
- If Assignment is there then on the interface, the `date of due` and particular to the `assignment id` it has to be located from the server side
- The `main icons` would be there, one to `view` which assignment is being given, other for `instructions` regarding that assignment, the last one to `submit or hand in` the assignment
- With displaying messages when connected, so that an assignment cannot be duplicated from other journal

```
methods: {
   submitAssignment() {
     const assignmentId = '{AssignmentId}';
     const objectId = '{ObjectId}';
     const serverUrl = '{ServerUrl}';
     const submitAssignmentEndpoint = '{submitAssignment}';

     axios.put(`${serverUrl}${submitAssignmentEndpoint}/${assignmentId}`, null, {
       params: {
         oid: objectId,
       },
       headers: headers,
       responseType: 'json',
     })
     .then(response => {
       this.submissionStatus = true;
       this.submissionSuccess = true;
       console.log("Assignment submitted successfully:", response.data);
     })
     .catch(error => {
       this.submissionStatus = true;
       this.submissionSuccess = false;
       this.submissionError = error.message;
       console.error("Error submitting assignment:", error);
     });
   },
},
```

❖ tutorial.js

The Tutorials are going to be a popup, looking like this



- Adding `tutorial.js` file inside the lib directory, so that tutorial could be displayed for all buttons, and also inside the lib directory of those activities which are do not follow IntroJS for tutorial.

- The tutorial file will contain methods like `initTutorial` to initialize the Tutorial, `getElement` to get the tutorial value corresponding to an element and displaying the same, `startTutorial` to launch the Tutorial and similar for stopTutorial.

- Adding the code to start tutorial, inside the files for all screens like listview, homescreen, neighborhood, journal, settings

```
Vue.component('sugar-tutorial', {
    data: function () {
        return {
            tutorial: {
              elements: [],
              icons: null,
              activityId: null,
            }
    },
    methods() {
        getElementAsObject(name) {
      return this.tutorial.elements[name];
    },
    startTutorial() {
      if (this.app.getView() === undefined)
{       return;
      }
      if (this.launched) {
        return;
      }
      this.initTutorial();
      th
    },
```

❖ Porting Electron to adapt to new implementations

➢ lib/util.js

- Writing in terms of vue.js, we could include the utility component inside export and using a setup() function and adding the utilities for detection of electron
- Writing to a file, in electron using ipc communication, for calling the API and making changes to web content

- IPC Main, IPC Renderer to emit and receive data, using `ipcRenderer.send` to send messages to main

- Pseudo code to show IPC usage using electron in Vue

```
export default {
  methods:{
    send(){
      ipcRenderer.invoke('perform-action', {a:true})
    }
  }
}
```

- Pseudo Code to show how main process handles messages

```
export default {
  methods:{
    send(){
      ipcRenderer.invoke('perform-action', {a:true})
    }
  }
}
```

- Pseudo code to send a command from render process to main process

```
export default {
  name: 'App',
  mounted() {
    // handle reply from the backend
    window.ipc.on('READ_FILE', (payload) => {
      console.log(payload.content);
    });
  },
  methods: {
    readFile(path) {
      // ask backend to read file
      const payload = { path };
      window.ipc.send('READ_FILE', payload);
    },
  },
};
```

- Pseudo Code for Loading of files

```
methods: {
  askDirectory() {
     const { dialog } = require('electron').remote;
     dialog.showOpenDialog({
       properties: ['openDirectory']
     }).then(result => {
       if (!result.canceled) {
         console.log(result.filePaths);
       }
     }).catch(err => {
       console.error(err);
     });
  },
}
```

➢ **main.js**

- This file is used only for electron, we are porting this to Vue.js, we are also to look to make it compatible with the current changes, so as to make sure their main applications which include transferring of data through pipe implementation is successful for continuous upload or download from or to the device

- Creating a vue instance for main window, adding various methods for file operations like saving and loading of file, function for window creation

```
  mounted() {
    let mainWindow = new BrowserWindow(this.mainWindowOptions);
    mainWindow.loadURL('file://' + app.getAppPath() +
'/index.html');
    mainWindow.webContents.once('did-finish-load', () => {
      mainWindow.show();
    });
    mainWindow.on('closed', () => {
      mainWindow = null;
    });
```

- Pseudo code for activation of activities and windows all closed

```
created() {

    ipcRenderer.on('save-file-dialog', this.saveFile);
    ipcRenderer.on('choose-files-dialog', this.loadFile);

    app.on('ready', () => {
      this.createWindow();
    });

    app.on('activate', () => {
      if (this.mainWindow === null) {
        this.createWindow();
      }
    });

    app.on('window-all-closed', () => {
      if (process.platform !== 'darwin') {
        app.quit();
      }
    });
  },
```

❖ **Porting Android || IOS to adapt to new implementations**

lib/util.js

- Writing in terms of vue.js, we could include the utility component inside export and using a setup() function and adding the utilities for detection of electron

- Function to get browser name, android and ios, getting client name to app

- Storing data properties `currentCamera` to 0, `qrCancelCallBack` to null, buddy icons svg

- Then methods like getBrowserName to check for android, IOS, to set `minPasswdSize, restartApp, vibrate, Volume Controller, Scan Code`

- Pseudo code for Handling Volume in Android

```
methods: {
  handleVolumeUp() {

    cordova.plugins.VolumeControl.getVolume(volume => {
      cordova.plugins.VolumeControl.setVolume(volume + 10);
    }, () => {});
  },
  handleVolumeDown() {

    cordova.plugins.VolumeControl.getVolume(volume => {
      cordova.plugins.VolumeControl.setVolume(volume - 10);
    }, () => {});
  };
```

- Pseudo Code for handling QR Code Scan

```
methods: {
  startQRScan() {
    if (this.util.platform.android || this.util.platform.ios) {
      this.qrCancelCallback = this.handleQRScanResult;
      this.qrScannerPrepare();
    } else {
      console.log("QR scanning is not supported on this platform.");
    }
  },
  qrScannerPrepare() {
    QRScanner.prepare((err, status) => {
      if (err) {
        console.error("Error preparing QR scanner:", err);
      } else {
        QRScanner.scan(this.handleQRScanResult);
      }
    });
  },
```

- We would also have a method for handling File Management in Android / IOS, whenever user upload or downloads a file to or from the device

```
methods: {
   loadFile(event) {
     const file = event.target.files[0];
     const reader = new FileReader();

     reader.onload = e => {
       console.log(e.target.result);
     };
     reader.onerror = err => console.error(err);

     if (file.type.startsWith('text/') || file.type === 'application/json') {
       reader.readAsText(file);
     } else {
       reader.readAsDataURL(file);
     }
   },
```

- We will have to update the same inside `SugarDevices.JS`, `neighborhood.js` and `journal.js` for the adaption of same in them

These are the major changes which I am going to implement, along with that there would also be minor changes in other files like `audio.js`, `loader.js` for the new version, as I have found that they haven't been ported yet

Testing Plan

- ❖ Using Vue Test utils with Jest for testing the vue components created across various files
- ❖ I am also thinking of using Selenium Testing for testing methods which I propose to create for various files, which would help in automation of them
- ❖ I am learning on how to perform those tests with Selenium so it might take some time, so that I can research more on that, and work with any circumstances occurring around it, understanding it's behavior, it may take an additional week for doing it, keeping a buffer for it, read this blog to get some info link
- ❖ This is just a rough implementation and not a rigid one. These may change once I will start working on the project to test all the possible occurring scenarios.

How will it impact Sugar Labs ?

Sugarizer is a learning tool for children. It was conceived to work fully offline, so it could work by calling local files and with electron in a PC, Currently, `Sugarizer's core UI is made using VueJS` in the previous year, now we have to implement it's final solution for cross-platform mobile, and desktop, `which is a step towards migration from offline Sugarizer App, as it is a web frontend for Sugarizer-Server`. In this project, I will be rewriting some *.js files written in EnyoJs to VueJs. This will help Sugarizer to replace EnyoJs in future versions. VueJs documentation is easy to follow and it's updated from time to time. Vue is easy to maintain and considering my future prospects it is necessary to have a codebase easier to understand and contribute, to let new contributors comfortably come in.

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

The main part of the project will revolve with coding in Vue.js 3, along with making CSS adjustments to ensure the new component templates align with the current Sugar UI. Additionally, for tutorial.js, the transition will necessitate some frontend CSS modifications to maintain consistency with the current tutorial UI and tour box in the newer version.

Timeline

Break down the entire projects into chunks and tell us what will you work on each week

Days	Tasks
Pre GSoC **Feb 22, 2024 - May 1, 2024**	❖ Learning the working of Vue.JS 3, understanding the files and architecture of Sugarizer ❖ Working on v2 of Sugarizer, completing the TODO.md ❖ Contributing to Sugarizer and staying connected with the community
Community Bonding **May 1, 2024 - May 26, 2024**	❖ Discuss the creation of testing for this project and learn about it ❖ Going through VueJs documentation for code understanding about the integration to android and iOS platform ❖ Discussing about the new version and features to be added

	❖ Working around the tech stack for it
Week 1 **May 27, 2024 - Jun 2, 2024**	❖ Integrating the testing app, for testing the rewritten files ❖ Writing the order in which files to be created
Week 2 **Jun 3, 2024 - Jun 9, 2024**	❖ Unit testing of code files, researching around the testing ❖ Connection with api to handle storage ❖ Keeping sharing token so that it becomes easier to share between v1 and v2
Week 3 **Jun 10, 2024 - Jun 16, 2024**	❖ Testing to be done for leftover code files, so as to start with development of screens ❖ Development of neighborhood ui, writing the methods for the file
Week 4 **Jun 17, 2024 - Jun 23, 2024**	❖ Completion of writing the neighborhood.js file ❖ Implementation of ui of neighborhood
Week 5 **Jun 24, 2024 - Jun 30, 2024**	❖ Starting of journal view ❖ Creation of tool bar, bottom bar for journal
Week 6 **Jul 1, 2024 - Jul 7, 2024**	❖ Creation of methods, functions for journal.js ❖ Syncing Journal both at UI and at server data using API request at each reference ❖ Addition of log, seeing updation of journal

Mid Phase Evaluation **Jul 8, 2024 - Jul 14, 2024**	❖ Testing of methods using Selenium ❖ Completion of Journal and Neighborhood Screens ❖ Including them in MainScreen
Week 8 **Jul 15, 2024 - Jul 21, 2024**	❖ Starting of creation of Tutorial file
Week 9 **Jul 22, 2024 - Jul 28, 2024**	❖ Localization of tutorial ❖ Writing Scripts for the same
Week 10 **Jul 29, 2024 - Aug 4, 2024**	❖ Development of tutorial for all activities, for all screens ❖ Completion of writing and implementation of Tutorial for all screens
Week 11 **Aug 5, 2024 - Aug 11, 2024**	❖ Fixing Electron code in Main.js file to work to vue.js implementation
Week 12 **Aug 12, 2024 - Aug 18, 2024**	❖ Converting the IPC in lib/util.js to work with Vue, ❖ Starting to migrate the Android code to work with Vue
Final Phase Evaluation **Aug 19, 2024 - Aug 25, 2024**	❖ Transferring changes to functionalities like volume, scanning for QR Code, vibrating, controlling of volume, restarting of the application
Final Evaluation Submission **Aug 26, 2024 - Sep 1, 2024**	❖ Writing other small files, loader, audio if needed ❖ Cleaning and wrap up the code

*It may not be a task but Handling PR feedback would also be an important thing to work upon every other week
In Aug 2nd week I would be having my semester examinations at that time I would work for a little less duration

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

I have my Institute summer vacations starting from May 11 to July 17. In this period I can give about 28-40 hours per week and after college starts, I can give about 20-25 hours per week. I have no other commitments for the summer vacation, so I can devote most of my time to GSoC.

Highlight the work you plan to complete before each evaluation.

`Before Mid Evaluation` Initializing the testing app, development of the Journal and Neighborhood Screens, complete functional working of them

`Before End Evaluation` Working towards the tutorial section for it's displaying on each screen. Porting the electron part, so that it fixes with VueJS, Converting the android so that all features like QR Scan, vibrate, downloading and uploading the documents and handling of windows. Performing the unit testing and code coverage for the functionalities.

How will you report progress between evaluations ?

Every week I would have a meeting with the mentors on discord or on matrix, where I would share my progress.
I will also submit the notes of every meeting and next week tasks on sugar-devel channel
I will be active on GitHub and will contribute regularly to the project, so everyone in the Organization will be able to view my progress.
I am also thinking of writing blogs, to share my progress.

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

I am thinking of working continuously on the project after the project ends. After this project, I would like to rewrite all activities to VueJS, update the versions and if approved, propose new UI changes to activities. I aim to develop mentorship skills and the ability to guide others and try to give back to the community by mentoring and guiding others. I hope to mentor future GSoC students, as I improve my skills and become more experienced in the Sugar Labs community.

I am Looking forward to contribute to Sugar Labs, for this summer
Kind Regards