GSOC’ 23 PROJECT PROPOSAL

Sugarizer Vue.JS Core

Basic Details

Fullname: Vinayak Nayar

Email and Github Username: nayarvinayak03@gmail.com and vinayaknayar

First Language: Hindi

Location and Timezone: Uttar Pradesh, India(UTC +5:30)

Share links, if any, of your work on previous open-source projects.

Here are some links to my previous contributions.
Improving documentation for a tool named Ugit(undo git commands).
Fixed a minor bug in amibot(a WhatsApp bot for amity university students)

Besides that, I have been working on the development of an Open source project - Mesazh (A chat application)

Convince us that you'll be a good fit for this project, by sharing links to your contribution to Sugar Labs.

During my internship at Essentia Softserv, I gained valuable experience in Vue.js and developed UI components for various sections of a website. I also conducted unit testing of components using Jest, which helped me refine my skills in front-end development and testing.
In addition to my technical expertise, I pride myself on being reliable, responsible, and communicative. These qualities enable me to work effectively as part of a team and to take ownership of my responsibilities. As a result, I believe I would be a great fit for any open-source project.
Recently, I have been exploring the Sugarizer app and have worked on integrating the Sugarizer Server API as a demo. This experience has given me a deeper understanding of the project's objectives and technical requirements. As someone passionate about open-source software and its potential to benefit society, I am eager to make meaningful contributions to the Sugar Labs community.
I am particularly excited about the opportunity to contribute to the Sugarizer app and collaborate with other members of the community. With my experience in Vue.js, UI development, and unit testing, as well as my personal qualities of reliability, responsibility, and communication, I am confident that I can make valuable contributions to the project. I look forward to exploring potential opportunities to contribute and learn from other community members.

Proof of work:
Vanilla Javascript activity tutorial -> Link
VueJS activity tutorial -> Link
PRs Sugarizer App -> Link
PRs Exerciser Activity -> Link
Demonstration of integration of Sugarizer Server API -> Video Link Domain Link

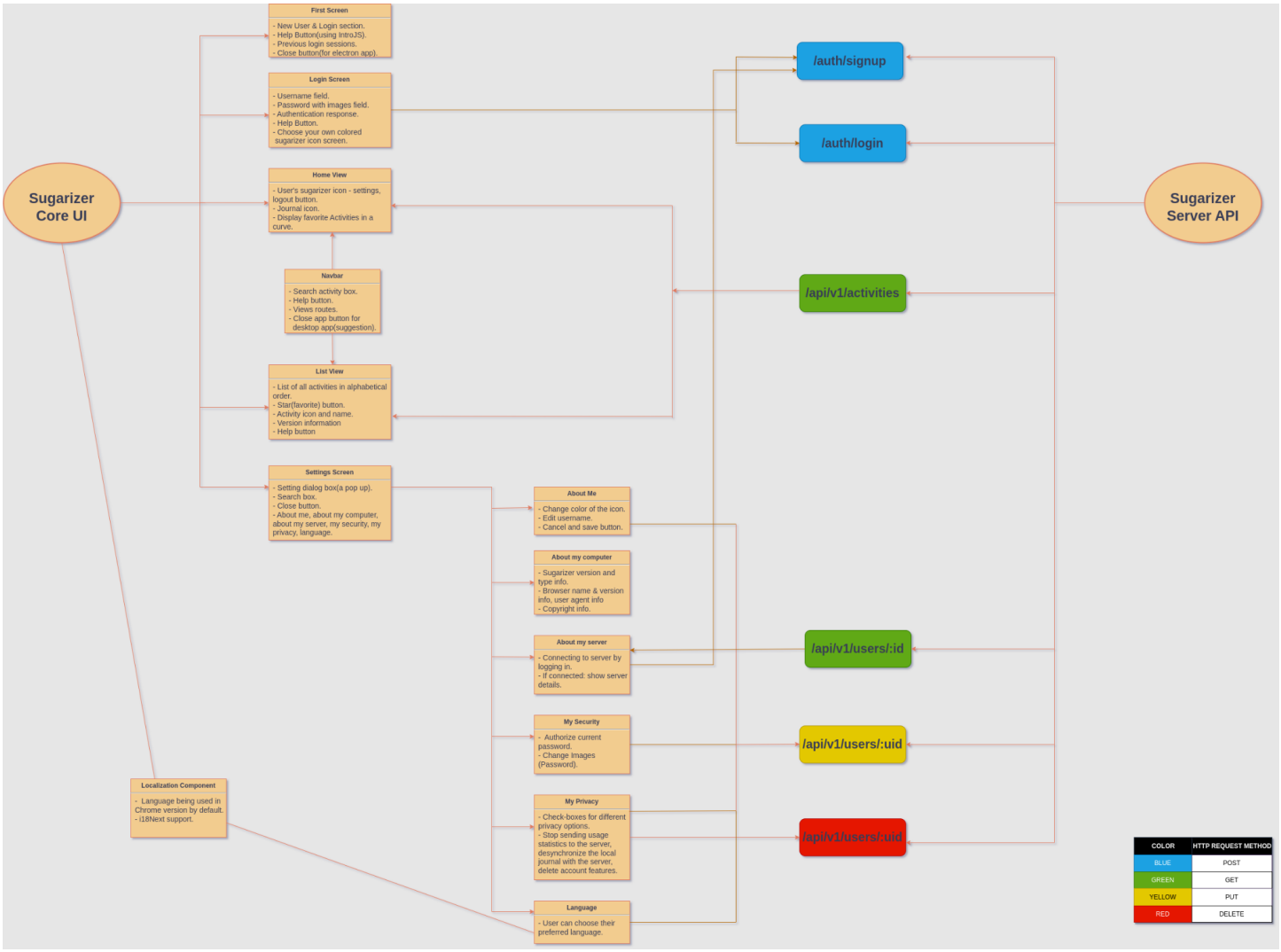
Project Details

What are you making?

This project aims to reimplement the Sugarizer Core UI by utilizing Sugarizer Vue.JS components. The current implementation of the Sugarizer Core UI relies on Enyo.JS, a deprecated web framework. Therefore, this project is necessary to update and improve the Sugarizer Core UI to ensure that it remains a modern and efficient tool.

The link below is the flow chart that describes the project's architecture.

- sugarizer-vuejs-core-ui.drawio.png (for better view)



A detailed explanation of the components:

Localization Component

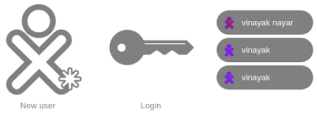
We will use i18Next to localize the sugarizer app, which will need us to create distinct JSON files for each language instead of the existing reliance on.ini files. In settings, the user should choose their favorite language (by default language being used in chrome should be used). Assuming we have the user's information saved on our server, we would retrieve their chosen language by sending an HTTP GET request to /api/v1/users/:uid.

First Screen

In order to move to the login and new user screens from the first screen, we don't need to perform any API calls, all we need is a straightforward user interface.

The main feature, aside from that, is to display the navigation buttons based on the history of past logins on the screen after retrieving it from local storage.

We also have a help button that the user may click to get more information on the first screen's components, I only want to use IntroJS for this (because we already are).



Login Screen

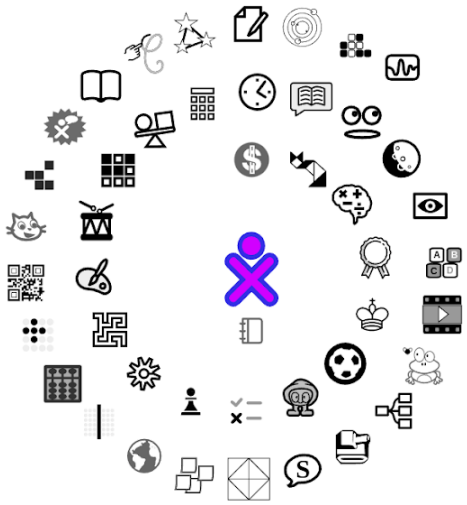
- Login: The user will be directed to the login screen if they click the login icon on the first screen. Here, we will display the components that will ask for the server's information, the user’s name, and the password, respectively. Once we have this data, we will send an HTTP POST request to /api/auth/login to validate the user information and, if it is accurate, the user will be directed to the home view.

- Sign Up: The user will be directed to the new user sign-up screen if they click the new user icon on the first screen. Here, we will display the components that will ask for the server's information, the user's name, and the new password, respectively. Here, after receiving the user’s name we will make an HTTP POST request to /auth/signup with the parameter “beforeSIgnup: true” to check whether the name already exists in the server or not, if it does not exist we will move forward to the password field. Once we have this data, we will send an HTTP POST request to /auth/signup. This will create a new user in the server, and we'll use this information to make a POST request to /auth/login so that the user can log in and be taken to the app's home screen.

- UI aspect: The UI of the password field is unique and I would like to take reference from the existing codebase for that. The other components don’t have much complex UI and should be easier to implement.

Home View

For the home view, we need to render all the favorite activities of the user in a spiral manner, and for that, we need to make an HTTP GET request to the /api/v1/activities?favorite=true endpoint. We also have a settings and logout option there. Log out is simple to implement, by removing the user's data from the local storage and navigating the user to the first screen should work for that. Besides that, when the user clicks on the settings option, the settings dialog box should be displayed.

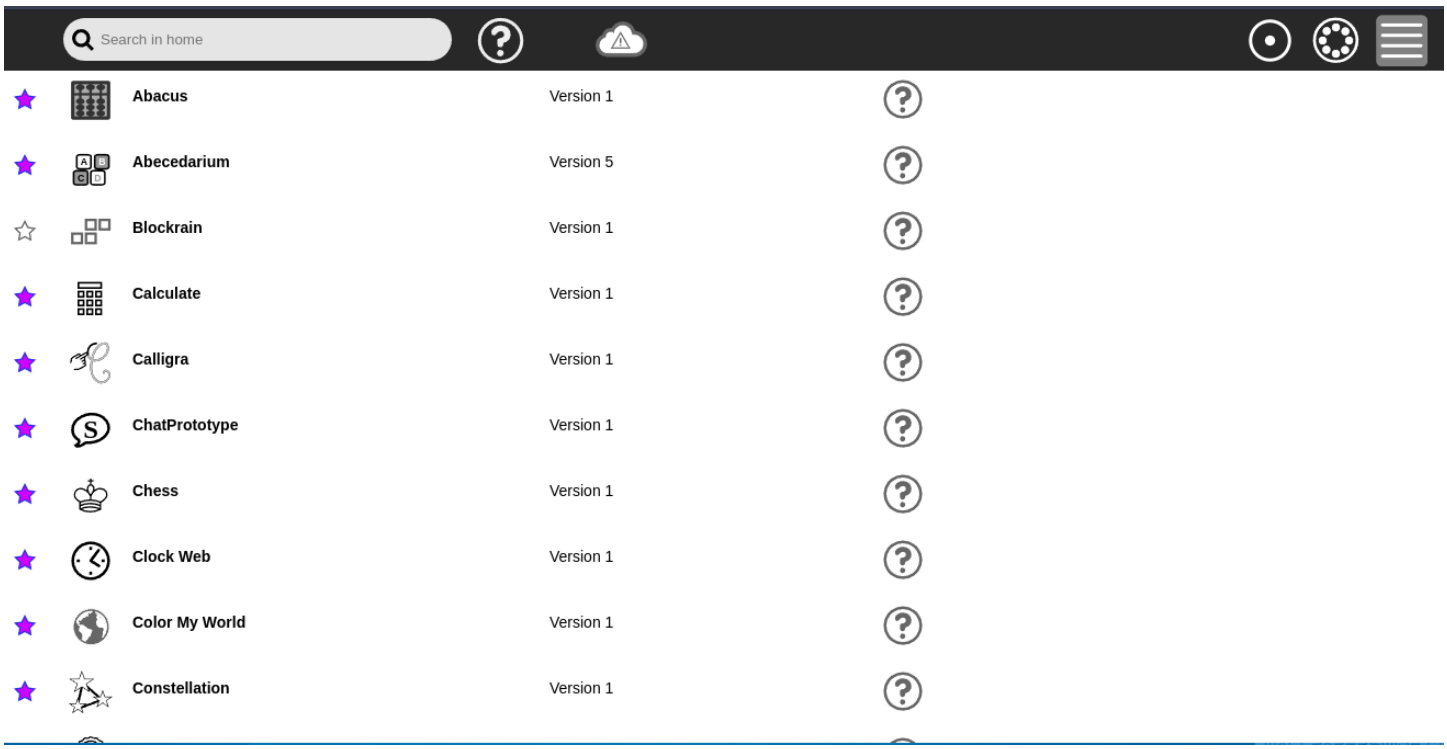


Navbar

Navbar is the common part in the Home and List View. It has a few things to keep in mind. It has a search bar; buttons for home view, list view, and neighborhood view; a help button, and a cloud-like element to display if the user is connected to the server or not.
However, for the scope of this project, the most crucial elements of the Navbar are the buttons for the home view and list view. These buttons act as triggers that allow users to switch between the different views of the app.

List View

When the user clicks on the "List View" button in the Navbar, we can retrieve all available activities in alphabetical order by sending an HTTP GET request to the "/api/v1/activities" endpoint. This will allow us to display the activities in an organized manner.
In addition, we want to provide the option for the user to mark/unmark an activity as a favorite. To accomplish this, we need to send an HTTP PUT request to the "/api/v1/users/:uid" endpoint. This will update the user's favorite activities list accordingly.
Along with the favorite option, we can also retrieve other useful information for each activity, such as its name, icon path, and version information. We can get this information from the same GET request to the "/api/v1/activities" endpoint.

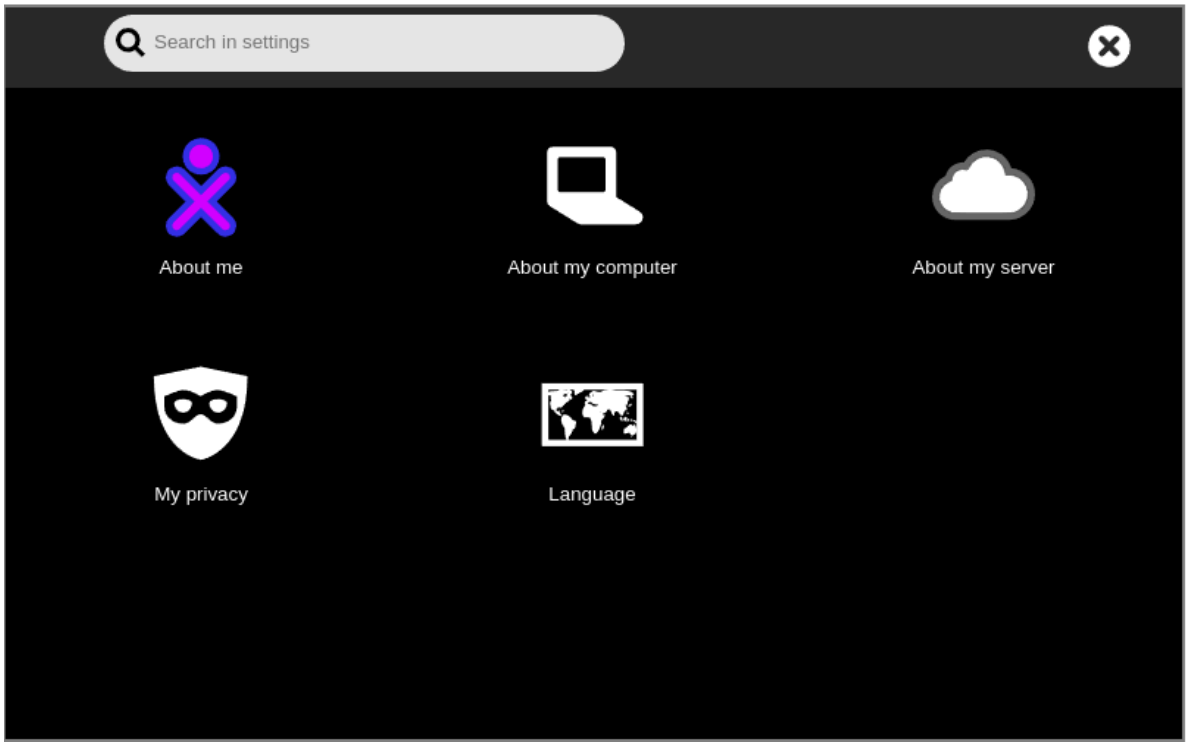


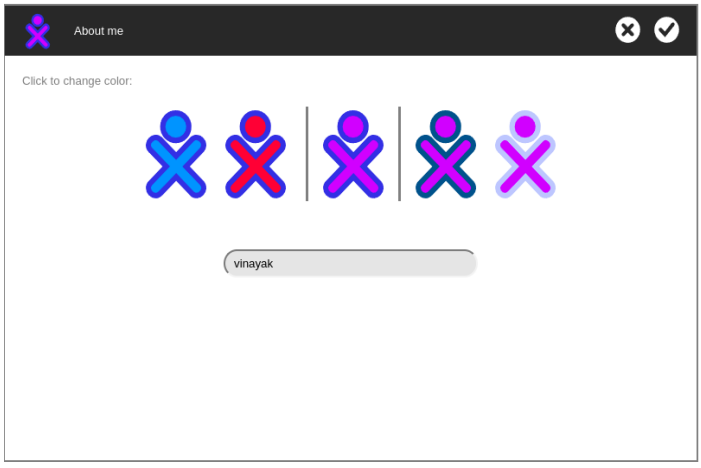
Settings

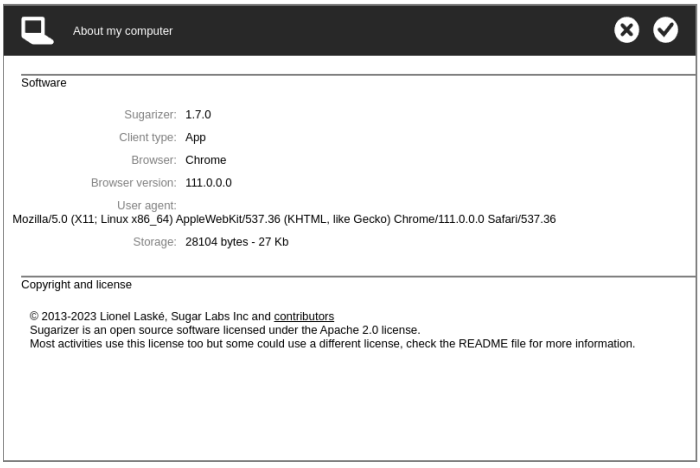
The settings screen consists of the search bar and six other components. About me, about my computer, about my server, my security, my privacy, and language.
HTTP Requests we will need are as follows.

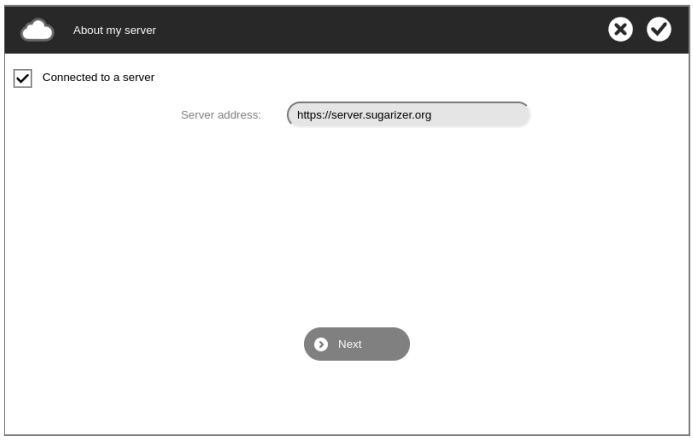
- PUT /api/v1/users/:uid - To edit the user’s name, password, preferred language, and privacy options - server statistics, and synchronization of the journal.

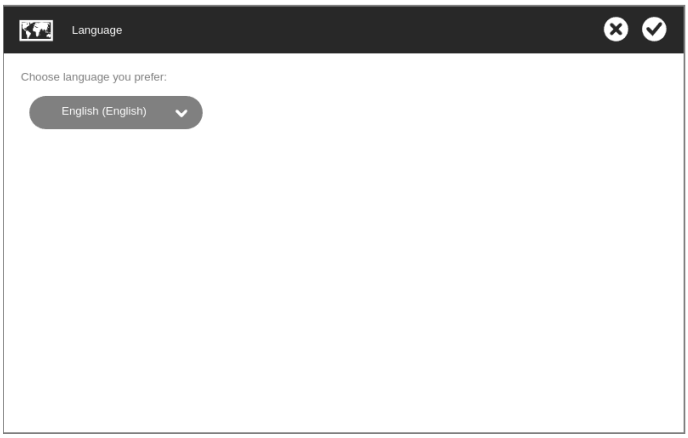
- POST /auth/signup - If the user isn't connected to the server we will need to create a new account for the user.
- GET /api/v1/users/:id - To retrieve the information of the user if the user is connected to the server.
- DELETE /api/v1/users/:id - If the user chooses to delete the account from the server(option available in my privacy tab).











How will it impact Sugar Labs?

In this project, we will be enhancing the Sugarizer Core UI by implementing it using the Vue.js framework, known for its high speed, efficiency, and superior performance. By leveraging Vue.js, Sugarizer can become a more efficient and maintainable application.

What technologies will you be using?

Sugarizer Vue.JS Components

To achieve the aim of this project, we will need to use Sugarizer Vue.JS components, which are already being used in some of the activities.

Require.JS

To import/load javascript files and modules.

Vue test utils with Jest

For unit testing of our Vue components.

Axios

For making HTTP requests to the sugarizer server.

i18Next

An internationalization framework that we will use for the translation of the content in different languages.

IntroJS

For making progressive help/ tutorial components.

Timeline

May 4 - 28 (Community bonding period)

- Completing the required setup for the project
- Discussing the overall workflow of the project with my mentor
- Understand the current architecture of sugarizer in-depth and I will contribute to the existing sugarizer repo achieving the same.

Note: Might be off-grid during this period due to my end-semester exams.

May 29 - June 4 (Week 1)

- Adding localization compatibility using i18Next.
- Reimplementing the required features of the current lib/sugar-web directory.
- Unit testing(I want to follow test driven development process)

June 5 - June 11 (Week 2)

- Development of UI of the first screen. (New user, Login, Help, Previous Logins)
- Unit testing

June 12 - June 25 (Week 3 & Week 4)

- Development of the Login Screen (the password screen, new user screen, and login screen).
- Unit testing.

June 26 - July 2 (Week 5)

- Development of Home View.
- Unit Testing

July 3 - July 9 (Week 6)

- Fixing the patches (if they exist).
- Sugarizing the UI (if needed).

Note: By Sugarizing I mean to make the UI more approachable for children (for eg, using less text) and making it up to the mark of the current sugarizer UI.

July 10 - July 14 (Mid-term evaluation)

Goals for mid-term evaluation

Majorly, Localization of the app should be implemented. The first Screen, Login screen and Home View should work perfectly with most of the functionalities.

July 14 - July 23 (~ Week 8)

- Navbar UI.
- Development of List View.
- Unit Testing.

July 24 - August 6 (Week 9 & Week 10)

- Implementation of the settings screen.
- Unit Testing.

August 7 - August 13 (Week 11 & Week 13)

- Unit testing of the components left.
- Buffer Period.

August 14 - August 20 (Week 14)

- Fixing the patches (if they exist).
- Sugarizing the UI (if needed).

August 21 - August 27 (Final Week)

- Finishing/ Final touch to the project.

August 28 - September 4 (Final Evaluation)

Goals for final evaluation

Completion of all five screens (first screen, login, home view, list view, and settings screen), implementation of other important features of sugarizer like localization and unit testing coverage of the components.

How many hours do you plan to spend each week on your project?

I plan to spend 24-30 hours a week. I don't have any other commitments during the summer besides my university class may resume after July 15.

How will you report progress between evaluations?

I would like to provide weekly progress reports and would appreciate it if you could suggest a suitable platform or method for me to share them with you regularly.

Post GSoC plans?

As someone new to Sugar Labs, I find the fact that it is a free/libre open-source organization whose main aim is to make the learning process for students easy and enjoyable to be a bit intimidating. However, I am eager to become an active contributor and to learn as much as possible during my journey with the organization. I see myself eventually becoming a mentor for programs like GSoC, as I become more experienced and skilled in the Sugar Labs community. Even if I don't get selected for GSoC, I would like to be a contributor.