



sugarlabs

GSoC 2023 Proposal

Sugarizer Vuejs

Core

Nischay Goyal

Basic Details

Full Name:

Nischay Goyal

Emails and Contacts:

Primary email: goyalivuq1@gmail.com,

Secondary Email: Nischay.20scse1010893@galgotiasuniversity.edu.in

Github: [NischayGoyal1](#)

Matrix Username: [NischayGoyal1](#)

LinkedIn Profile: [Nischay Goyal](#)

Phone: [+91 7696073734](tel:+917696073734)

Resume: [Resume link](#)

Your First Language:

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

Location and Timezone:

Location: Delhi, India

Timezone: Indian Standard Time (UTC+5:30)

Communication:

The time I will be comfortable working with:

- UTC 12:30 - UTC 16:00 (IST 18:30 - IST 21:30)
- UTC 17:30 - UTC 21:30 (IST 23:00 - IST 03:00)

To ensure that we can work together seamlessly, I am willing to be flexible with my schedule and can be reached anytime through my mobile number and email.

Education Details

I am a Computer Science and Engineering undergraduate student at the **Galgotias University** pursuing a Bachelor of Technology in my third year. I was introduced to the world of programming and software development in my first year. Since then, I have delved into subjects such as Data Structures and Algorithms, Web Development, Machine Learning, and Operating Systems, gaining valuable knowledge and skills along the way. I have also been an active contributor to open source programming since my second year, participating in various hackathons and events to further hone my abilities. My primary focus has been on web-based technologies, as well as Data Structures and Algorithms, utilizing Python Programming in my projects. Additionally, I have been learning about the exciting field of Blockchain and its potential applications.

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

I was introduced to open-source projects in my second year of undergraduate studies and have since been actively involved in developing and learning about various software and technologies.

I have been contributing to **Sugar Labs** for the past **4 months**. During this time, I have contributed to many repositories and fixed documentation, bugs, UI changes, enhancements, and updated versions and packages. These past four months have been a great learning experience. These are my contributions to **Sugar Labs**:

Pull request link	Description	Status
#1190	Replaced Bootstrap Tour by IntroJs in Vote Activity (Link)	Merged
#1190	Replaced Bootstrap Tour by IntroJs in Write Activity (Link)	Merged
#1196	Solved Step Numbers are not visible Chess Activity(Link)	Merged
#1290	Fixed Navbar Still Active in Full-Screen Mode in Physics JS Activity (Link)	Merged
#1273	Added key listening feature to pagination of Ebook Reader Activity (Link)	Merged
#1321	Fix the bug related to language change button functionality in Abecedarium Activity (Link)	Merged
#1249	Enhanced Inro tour in (Abacus and video viewer Activity) (Link)	Merged
	Refactored the source code and removed unwanted files in 11 sugarizer activities (Link)	Merged

I have made over **23 commits** , **11 issues** and **11 (closed) PR till now**

Out of these contributions in Sugar Labs, the most significant ones are Migrating the Bootstrap tour to Intro js in (Vote Activity and Write Activity) , In this PR I have changed more than 20+ files by adding 8167 and removing 2371 line of code

I worked on some of my personal projects for learning purposes and I have also participated in HactoberFest'22. Other than this I also contributed to many open-source projects. Following are some of my open-source contributions to different organizations.

- <https://github.com/Ank221199/E-CommerceVueAndFlask/pull/17/files>
- <https://github.com/KODEL/HacktoberFest-2022/pull/8>
- <https://github.com/Yashika1410/Code-base-2022/pull/61>

Details of my other contributions can be found on my Github profile here.

Project Details:

What is the name of your project?

The name of my project is **Sugarizer VueJS Core**

Aim of Project

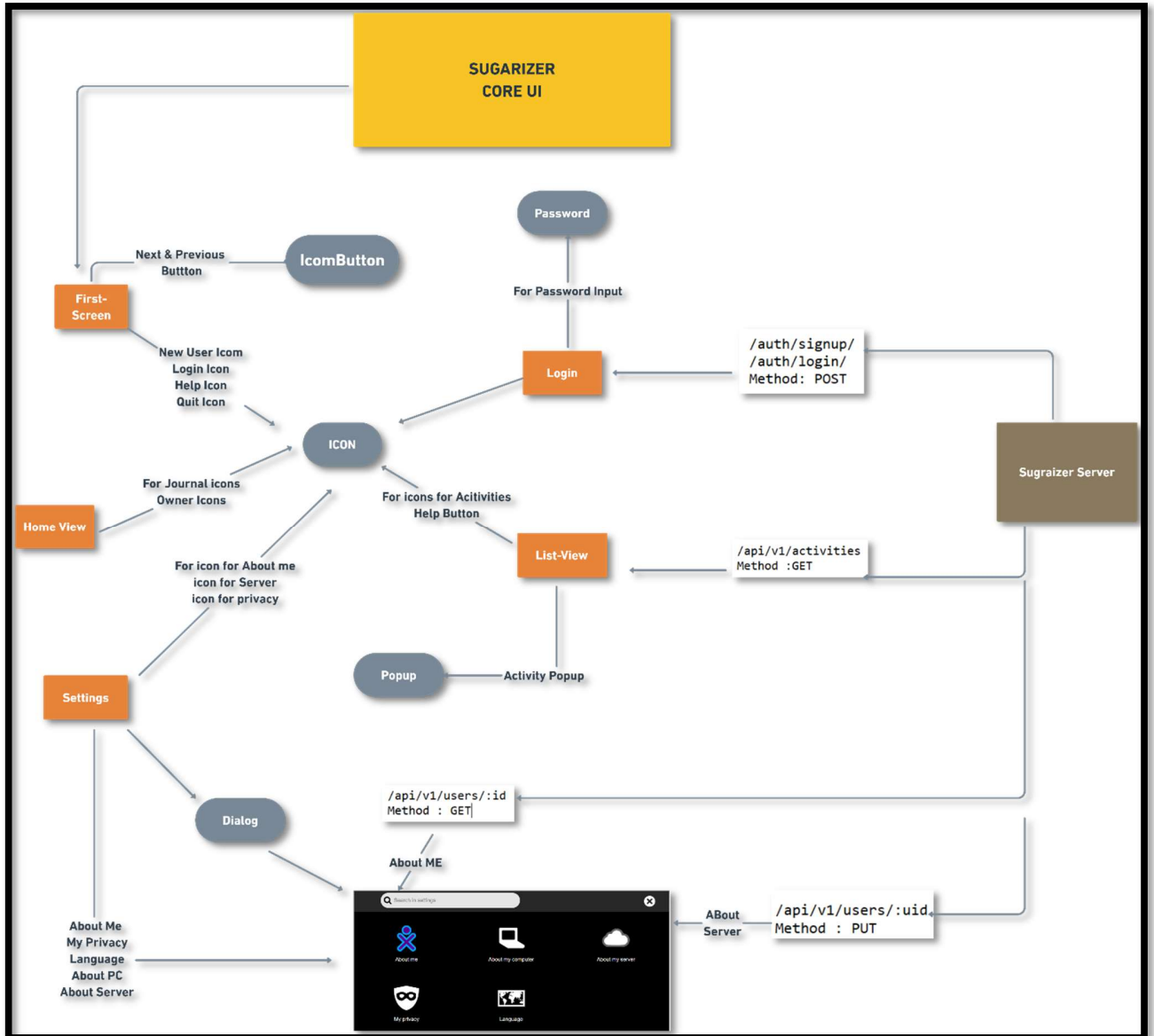
We aim to create Vue.js UI components that match the Sugarizer UI by replacing the outdated EnyoJS framework designed for WebOS. We'll use Sugarizer VueJS components and the Sugarizer Server API to create screens for the First screen, Login, Home view, List view, and Settings.

For this, I will be rewriting the following files of 'js/' directory:

- 1) /firstscreen.js
- 2) /homeview.js
- 3) /listview.js
- 4) /dailog.js

The relationship between these files can be summarized in the image provided below:

LINK : [FlowChart \(whimsical.com\)](https://whimsical.com)



First Screen

It's a class for Sugar First Screen based on the existing kind, enyo.control in EnyoJs used to render first screen in sugarizer UI

The first screen has :

Toolbar (which is hidden and its background color is set to white)

```
const toolbar = document.getElementById("toolbar");
vm.backgroundColor = toolbar.style.backgroundColor;
toolbar.style.backgroundColor = "white";
document.getElementById("canvas").style.overflowY = "hidden";
util.hideNativeToolbar();
```

HelpButton (which will give a starting tutorial):

HelpButton will be using icon component which already written in gsoc22 to render **SVG** icon for the button, Clicking the button will trigger the tutorial (that will be using **intro js** for the tour) that will be using the function **startTutorial()**

If the user has no browsing history, the tutorial will automatically start after a brief delay .

```
window.setTimeout(function() {
  if (vm.history.length = 0) {
    vm.startTutorial();
  }
}, constant.timerBeforeTutorial);
```

NewUser and Login

There are Newuser and login button that are using **icon js** component for rendering SVG for there icons , Clicking Newuser and Login will trigger **newUser()** and **login()**

functions respectively.

Previous Logins List

These container are using **lib/history.js** which is sugar web library , it is used to access the Sugar data store, It creates an object named "historic" with a method named "get" that retrieves the history content from the Sugar data store

Our firstscreen will be using this get method to Gets the user's history and sets it to an empty array if it's not available, Sets the count of the historybox component to the length of the history array.

```
vm.history = historic.get() || [];
vm.$refs.historybox.set("count", vm.history.length, true);
```

These containers are using **js/lconsbutton.js** vue component and have two methods **setupHistory()** and **historyClicked()**

historyClicked() will have a event listner and it will be triggers when user clicks on an item in the history list

```
historyClicked(inSender, inEvent) {
  const user = this.history[this.history.length - inEvent.index - 1];
  this.$refs.name.setValue(user.name);
  this.$refs.server.setValue(user.server ? user.server.url : "");
  const that = this;
  if (user.server && user.server.url) {
    // Retrieve the server in history and go to login
    myserver.getServerInformation(this.$refs.server.getValue(), function(inSender, inResponse)
  {
    inResponse.url = that.$refs.server.getValue();
    preferences.setServer(inResponse);
  });
  }
  ....
}
```

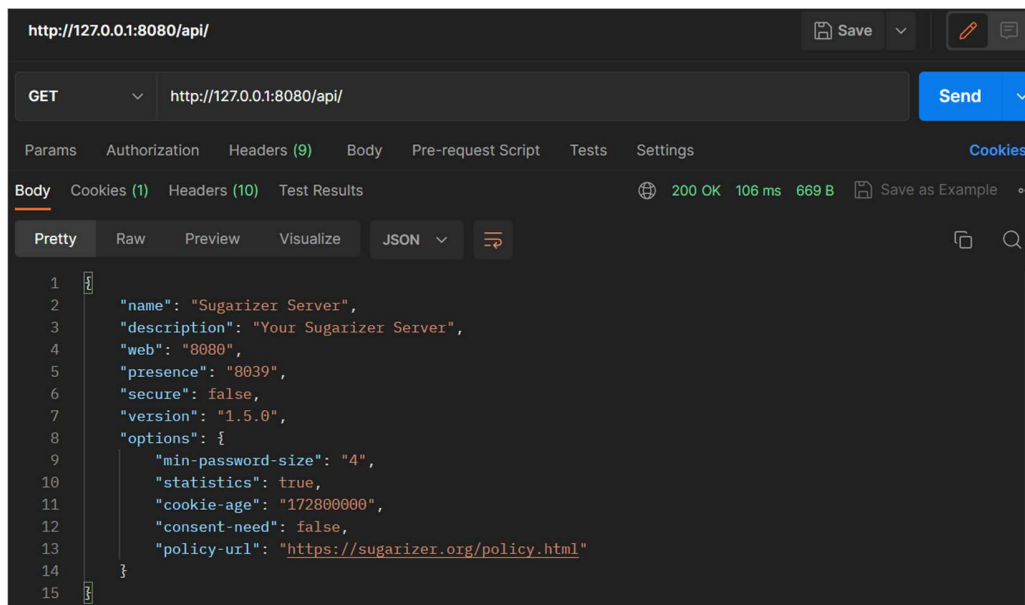
This **historyClicked()** method is using -

myserver.getServerInformation(this.\$server.getValue(), function(inSender, inResponse
Which is function form lib/server.js (sugar library) containing all the functions related to API calls.


```
server.getServerInformation = function(serverurl, response, error) {
  var serverUrl = serverurl;
  if (serverUrl.length && serverUrl[serverurl.length-1] == '/') {
    serverUrl = serverUrl.substr(0, serverUrl.length-1);
  }
  var ajax = axios.create({
    responseType: "json"
  });
  ajax.get(serverUrl + constant.serverInformationURL).then(function(inResponse) {
    if (response) response(null, inResponse.data);
  }).catch(function(inResponse) {
    if (error) error(inResponse, getErrorCode(inResponse));
  });
}
```

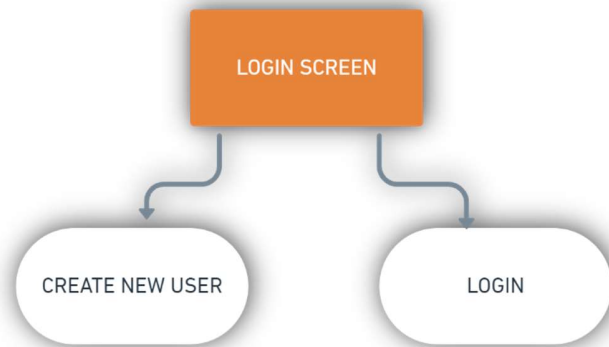
This function is using **Sugarizer-Server API** , Using **Axios** to making a **Get request to the route `/api`** (`constant.serverInformationURL`)

Below Is the response generated by hitting the `/api` route-



```
http://127.0.0.1:8080/api/
GET http://127.0.0.1:8080/api/
200 OK 106 ms 669 B
Body
Pretty
1  "name": "Sugarizer Server",
2  "description": "Your Sugarizer Server",
3  "web": "8080",
4  "presence": "8039",
5  "secure": false,
6  "version": "1.5.0",
7  "options": {
8    "min-password-size": "4",
9    "statistics": true,
10   "cookie-age": "172800000",
11   "consent-need": false,
12   "policy-url": "https://sugarizer.org/policy.html"
13 }
14
15
```

Login Screen:



CREATE NEW USER

Clicking the New User on the First screen will redirect the user to the Create New user screen. Which will have Choose name option and Next and Previous button (These buttons will be using **js/iconbutton.js** component) These next and previous buttons are bind to **next()** and **previous()** functions respectively, along with these we also have a

Help-Button which will trigger tutorial tour .

After the choosing the name when we click to **next** button user will be directed to Password section , which will be using **js/password.js** a prewritten Vue component in gsoc22 , The user will be asked for password in the form of emoji (Atleast 4) . After choosing the password user will be directed to choose the color of user icon

Finally user will be directed to Home View Screen

The Create New User will be using the following functions :

[createUser\(\)](#)

```

createUser() {
  const that = this;
  myserver.postUser({
    name: preferences.getName(),
    color: preferences.getColor(),
    language: preferences.getLanguage(),
    role: 'student',
    password: this.$refs.password.getPassword(),
    options: {
      sync: preferences.getOptions('sync'),
      stats: preferences.getOptions('stats')
    }
  })
  .....
  .....
}
  
```

The `createUser()` creates a new user, it uses **`myserver.postUser()`** method which come **from `lib/server.js`**

This function send a post request to the Sugarizer-Server API with the user's information, including their name, color, language, role, password, and options

```
// Create a new user
server.postUser = function(user, response, error) {
  var ajax = axios.create({
    responseType: "json"
  });
  ajax.post(server.getServerUrl() + constant.signupURL, {user:
JSON.stringify(user)}).then(function(inResponse) {
  if(!user.beforeSignup) {
    var newUser = {"name": user.name, "password": user.password};
    server.loginUser(newUser, function(loginSender, loginResponse) {
      preferences.setToken({'x_key': loginResponse.user._id, 'access_token':
loginResponse.token});
    });
  }
  if (response) response(null, inResponse.data);
}).catch(function(inResponse) {
  var code = getErrorCode(inResponse);
  if (code == 3) {
    if (sessionExpired()) {
      return;
    }
  }
  if (error) error(inResponse, code);
});
}
```

The above function uses **`axios.post`** method is used to make a POST request to `/auth/signup/` with users information.

checkUsername()

Responsible for checking if a given username is valid or not. It will take two arguments name which is the username to be checked, and `createnew` which is a boolean value indicating whether the user is creating a new account or trying to login with an existing account.

This function also uses the above **`server.postUser()`** method to check if the user is already exist or not,

If the user does not exist and `createnew` is true, or if the user exists and `createnew` is false user will be proceeded to next screen, other-wise there will be a warning message

"InvalidUser" to indicate that the user does not exist.

Login

Clicking the login on the First screen will redirect the user to the login screen Which will ask user to enter name , Here **checkUsername()** method will be called to check if the user exist or not . if user exist the user will be directed to password section , Here user will enter password , here **loginUser()** method is called

```

methods: {
  loginUser() {
    const user = {
      name: this.username,
      password: this.password,
    }
    this.showSpinner = true;
    myserver.loginUser(user,
      (loginSender, loginResponse) => {
        preferences.setToken({
          x_key: loginResponse.user._id,
          access_token: loginResponse.token
        });
        preferences.setNetworkId(loginResponse.user._id);
        myserver.getUser(loginResponse.user._id,
          (inSender, inResponse) => {
            ...
          }
        )
      }
    )
  }
}

```

Login function that logs in a user by sending a request to a server. The function starts by retrieving the user's name and password from the preferences and an input field respectively. Then, it creates a user object with the name and password and sends it to the server using the "**myserver.loginUser()**" function.

If the login is successful, the function retrieves the user's data by calling "**myserver.getUser()**" with the user's ID, which is returned in the login response

It then sets various user preferences, such as their private and shared journals, and sets a flag indicating that the user is connected.

If the login is unsuccessful, the function displays a warning message with an error code indicating the reason for the failure.

myserver.loginUser() and **myserver.getUser()** come from **lib/server.js** (**sugar library**)

```

server.loginUser = function(user, response, error) {
  var userValue = user;
  userValue.role = ["student", "teacher"];
  var ajax = axios.create({
    responseType: "json"
  });
  ajax.post(server.getServerUrl() + constant.loginURL, {user:
JSON.stringify(userValue)}).then(function(inResponse) {
  response(null, inResponse.data);
}).catch(function(inResponse) {
  var code = getErrorCode(inResponse);
  if (code == 3) {
    if (sessionExpired()) {
      return;
    }
  }
  if (error) error(inResponse, code);
});
}

```

The above function use **axios.post** method to send a **POST** request to </auth/login/>, it take 3 arguments user , response , error

The user argument is an object containing the user's login credentials, including a name and password property, as well as a role property that is set to an array with two strings, "student" and "teacher"

The response argument is a callback function that will be called when the server responds with a successful login. The function receives two arguments, null and the response data returned from the server

myserver.getUser()

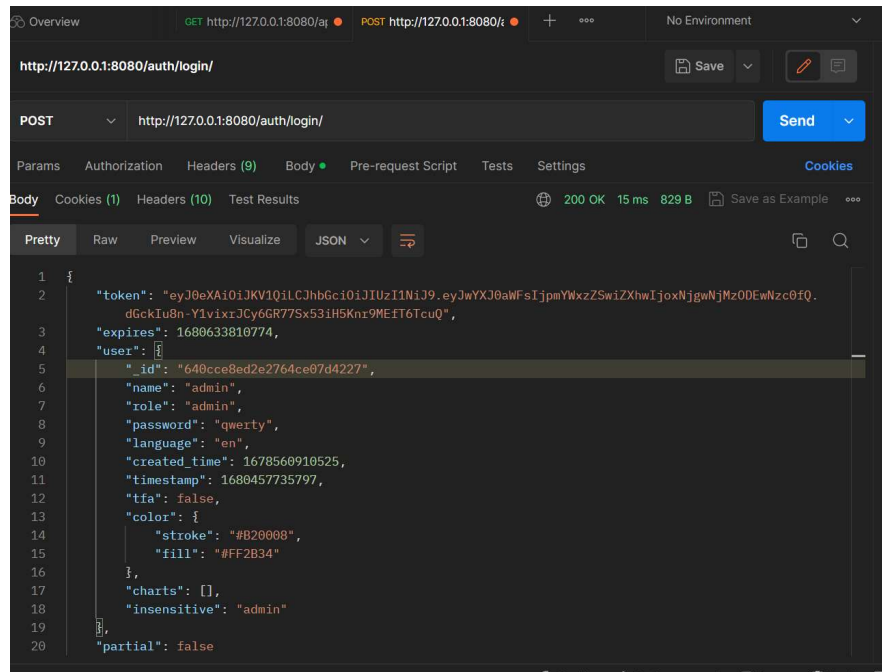
```

server.getUser = function(userId, response, error, optserver) {
  var ajax = axios.create({
    responseType: "json",
    headers: computeHeader(preferences.getToken())
  });
  ajax.get((optserver ? optserver : server.getServerUrl()) + constant.initNetworkURL +
userId).then(function(inResponse) {
  response(null, inResponse.data);
}).catch(function(inResponse) {
  var code = getErrorCode(inResponse);
  if (code == 3) {
    if (sessionExpired()) {
      return;
    }
  }
  if (error) error(inResponse, code);
});
}

```

The function uses Axios to make a get request to </api/v1/users/> to get user data , it uses computerHeader() function to add token to header

Here is the snippet of the response by hitting </auth/login/>



HomeView

It's a class for Sugar Desktop based on the existing kind, enyo.control in EnyoJs Home View has a toolbar which use searchfield component, and uses icon component for rendering icons for journal , listview , owner , Sync Clicking on the Sync icon will trigger **connectToServer()** method which will send a get request to the server for getting user information and preferences. It also synchronize journal.

The sync button and offline button are not always visible, and the show assignments icon is used to display the number of assignments. The three buttons, radial, neighborhood, and list, are used to change between different views. The owner icon is placed at the center of the page, and the journal icon is placed below it. The owner icon and journal are surrounded by favorite activities arranged in a spiral view. To get favorite activities we can make

HTTP GET request to the </api/v1/activities?favorite=true> endpoint

The activities are color-coded based on whether they have been played before, and instances of the activity played are saved, When the user clicks on the settings option, the settings dialog box should be displayed. To log out, the user's data is removed from local storage, and they are navigated to the first screen., and the help button has functionalities similar to those in the first and login screens, When the user clicks on the settings option, the settings dialog box should be displayed. To log out **dologoff()** method is called , the

user's data is removed from local storage, and they are navigated to the first screen.

The radial button, neighborhood button, and list button have click event listeners that implement the **showRadialView()**, **showNeighborView()**, and **showListView()** methods on clicking, respectively.

LIST VIEW



Listview screen has a toolbar , which is used to navigate to homeview , it also uses searchfield.js component to have a searchbar for searching activities

By clicking on the list view icon user will be directed to LIIST VIEW screen

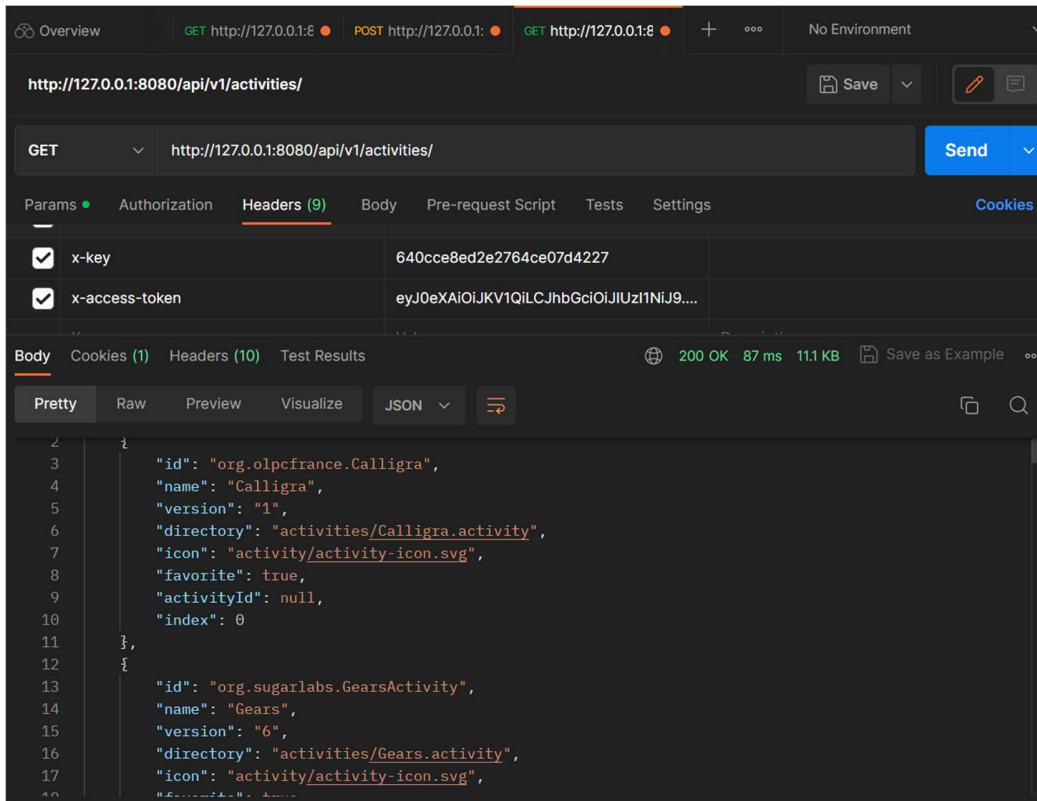
List View is using sugarizer-server API to fetch all the activities,

By using the function:

```
// Get activities
server.getActivities = function(response, error) {
  var ajax = axios.create({
    responseType: "json",
    headers: computeHeader(preferences.getToken())
  });
  ajax.get(server.getServerUrl() + constant.dynamicInitActivitiesURL).then(function(inResponse) {
    var newResponse = {data: inResponse.data};
    if (response) response(null, newResponse);
  }).catch(function(inResponse) {
    var code = getErrorCode(inResponse);
    if (code == 3) {
      if (sessionExpired()) {
        return;
      }
    }
    if (error) error(inResponse, code);
  });
}
```

Server.getActivites is a function in the **lib/server.js** , This function is using Axios to make asynchronous get request to the server by using route **/api/v1/activities/** to retrieve activities data.The function creates an Axios instance using `axios.create()`, with the `responseType` set to "json" and headers set to `computeHeader(preferences.getToken())`, which calculates the authentication headers needed for the request.If the request is successful the response will look like

x-key and **x-access-token** are also shown in the snippet



As we can see the response that came is not in sorted order so we need a sorting function

```

function sorted(activities) {
  return Object.values(activities).sort((a, b) => a.name.localeCompare(b.name));
}

```

The activity container will look like this



Having a Favorite icon , Activity icon , Activity name, Help button , all of them will be using icon.js component for rendering icons.

Clicking on Favorite icon it will trigger **doSwitchFavorite()** method , responsible for changing the color of icon and updating the user preferences in Datastore and server


```

methods: {
  doSwitchFavorite(inSender, inEvent) {
    const activitiesList = this.activities.sort((a, b) => a.name.localeCompare(b.name));
    const activity = activitiesList[inEvent.index];
    this.switchFavorite(inEvent.dispatchTarget.container, activity);
  },
}

```

Setting

For implementing Setting we will use **js/dialog.js** Vue component

There are 5 icons

1. About Me
2. About My Computer
3. About My server
4. My Privacy
5. My Security
6. Language

For **About Me** dialog:

It is used to change the color of current user-icon, which is done by **setcolor()** method

```

methods: {
  setNewColor(icon) {
    const newColor = icon.getColorizedColor();
    if (newColor === this.currentColor) {
      return;
    }
    this.currentColor = newColor;
    this.render();
  },
}

```

My Security

This dialog is used by user if user want to change password

This dialog uses next() method which sends a login request to the server using myserver.loginUser () with the user object as a parameter. If the login request is successful, the user's access token is saved to local preferences, a warning message is hidden, the password field is cleared, and the "Next" button's text is changed to "Done"

If the request is failed it displays a error message

Here We have use Sugarizer-server API to make the changes, We have made PUT request using AXIOS to the **/api/v1/users/:id**

My Privacy

My privacy dialog is used when user wants to delete his/her account, This dialog will have **confirmRemove()** method which delete the user data from Datastore as well as from server

For clearing data from Datastore the functions uses **historic.removeUser()** method and for deleting account from server it uses **myserver.deleteUser()** method which send DELETE request to server via [/api/v1/users/:id](#) route

```
myserver.deleteUser(
  networkId,
  function(inSender, inResponse) {
    if (util.getClientType() == constant.appType) {
      preferences.setServer(null);
      preferences.save();
      util.restartApp();
    } else {
      util.cleanDatastore(null, function() {
        util.restartApp();
      });
    }
  },
  function(response, error) {
    if (util.getClientType() == constant.appType) {
      preferences.setServer(null);
      preferences.save();
    }
    humane.log(l10n.get("ServerError", {code: error}));
    that.hide();
    that.owner.show();
  }
);
},
```

My Server

This dialog is used when user wants to know server information, it uses **myserver.getServerInformation()** method described in **lib/server.js** , It send a GET request to the server via route [/api/](#)

Language

This dialog is used when user want to choose or change the language
It uses **js/selectbox.js** Vue component, When the language setting is clicked, a dialog box will appear with a toolbar that offers comparable features to the previously mentioned settings. The toolbar will contain an icon for selecting the language, and underneath it, a div tag displaying the text "Please select your preferred language." Below the text, there will be a select box component allowing users to choose from various language options.

TESTING PLAN

For testing I will be using-

- Vue Test Utils
- Jest
- Postman

Login Screen

- 1) Test if the username and password input fields are rendered using `wrapper.find()`
- 2) Test if the login API returns an error when invalid credentials are entered using `mock axios` or `jest.mock()`

ListView

- 1) Test if the list of items is sorted by date or name using `wrapper.setData()`
- 2) Test if the List View screen is rendered using `wrapper.find()`
- 3) Test the activities list API using Postman

First Screen

- 1) Test if the Screen is mounting correctly
- 2) Test Create User and Login icons render correctly

Setting

- 1) My security should update the password correctly
- 2) Component and child settings should mount correctly
- 3) Connection to server steps should work properly in sequence

Localization

To localize an application in Sugarizer or Sugar-Web using the `webL10n` JavaScript library, the first step is to identify the strings that need to be localized. This involves replacing hard-coded strings in HTML or JavaScript files with localization resources. In `webL10n`, all strings are defined in a specific file called `locale.ini`, where translations for each string should be set.

To create the `locale.ini` file, use a text editor. The `webL10n` library is capable of automatically detecting the browser language. However, in Sugarizer, the user decides the language, so it is different. To determine the default language for the browser, check the `navigator.language` variable, except for Chrome OS where a `chrome.i18n.getUILanguage()` call is required. To get the user language, check the `environment.user.language` variable, unless it is not set. In that case, the `defaultLanguage` should be used.

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

Major part of the project will be using Vue Js , with this I will also be using HTML/CSS for implementing screens , Vue Test utils and Jest, Postman for Unit testing and API testing . Axios for API calls, Intro js for tutorial tour , and i18Next for creating localization component

Timeline

Time Period	Plan
Community Bonding Period	<ul style="list-style-type: none"> ➤ Discuss the creation of the testing app for this project and learn about it ➤ Go through documentation of EnyoJs and VueJs for code understanding about the integration to android and iOS platform ➤ Get familiar with codebase and the development environment ➤ Setting Up environment
Week - 1 (May 29 - June 4)	<ul style="list-style-type: none"> ➤ Adding localization component ➤ Implementing files from lib folder, As server.js for Server API calls

(Week 2) June 5 - June 11	<ul style="list-style-type: none"> ➤ Implementing First Screen, (Login Button, Create user button , help button) ➤ Creating history list for previous logins, ➤ Unit Testing
June 12 - June 25 (Week 3 & Week 4)	<ul style="list-style-type: none"> ➤ Implementing Login screen ➤ Creating methods for create user and integrating with Server API ➤ Creating methods for login (login user , checkusername) integrating it with server and password js
June 26 – July 9 (Week 5 & Week 6)	<ul style="list-style-type: none"> ➤ Implementing Home screen ➤ Unit Testing
July 10 - July 16 (Mid-term evaluation)	<ul style="list-style-type: none"> ➤ Writing Documentation for the previous work ➤ Complete Previous Pending work if any
Week - 8 (July 17 - July 23)	<ul style="list-style-type: none"> ➤ Implementing ListView ➤ Implementing methods (doswtichfavorite, dorunactivity) ➤ Integrating with API
Week - 9 (July 24 - July 30)	<ul style="list-style-type: none"> ➤ Writing Unit Testing for Listview and Login Screens ➤ Complete Any pending work
Week -10 & 11 (July 31 – Aug 14)	<ul style="list-style-type: none"> ➤ Implementing Setting Screen ➤ Integrating Dialogs with APIs ➤ Write Unit Test for the following
Week – 12 (Aug 14 – Aug 20)	<ul style="list-style-type: none"> ➤ Complete all the pending work ➤ Write Documentstion for the work

Convince us, in 5-15 sentences, that you will be able to successfully

complete your project in the timeline you have described. This is usually where people describe their past experiences, credentials, prior projects, schoolwork, and that sort of thing, but be creative. Link to prior work or other resources as relevant.

Being a developer and an enthusiast, I enjoy coding and creating solutions that are not only functional but also innovative and user-friendly. My goal is to improve the user experience by implementing new ideas in my work.

I am an active contributor of Sugar Labs with over **20+ commits** merged in various repositories. Spending this much time with the codebase helped me to understand it in a better way.

Some of my contributions

- 1) <https://github.com/llaske/sugarizer/pull/1235>
- 2) <https://github.com/llaske/sugarizer/pull/1292>
- 3) <https://github.com/llaske/sugarizer/pull/1328>

In addition, I have participated in several hackathons since my second year of college and have been fortunate enough to win some of them. As a team player, I have taken part in these competitions both as a leader and as a member of a team. I have helped guide my juniors and have learned from my seniors, all while working together effectively to build projects within strict deadlines. Through these experiences, I have developed valuable time management skills and learned the importance of communication, thus strengthening my soft skills.

I completed the Sugarizer Vue.js activity development tutorial. This helps me understand how the Sugarizer core works and how the activity is implemented. Following is the link to my repository of the implemented tutorials.

<https://github.com/NischayGoyal1/Pawn-Vuejs>

Prerequisites for Project:

As given in Idealist. I already have experience in HTML5, Javascript, and Vue.js framework development. Other than this I am already familiar with the Sugarizer codebase and have merged commits to the repository.

Following are some of my major projects

Twitter Clone:

Twitter clone replicates the basic features and functionalities of the popular social media platform, typically includes the ability to create and customize a user profile, post short messages or "tweets," follow other users and view their tweets, like and retweet tweets,

Tech: Vue3 framework, CSS, HTML, Javascript,



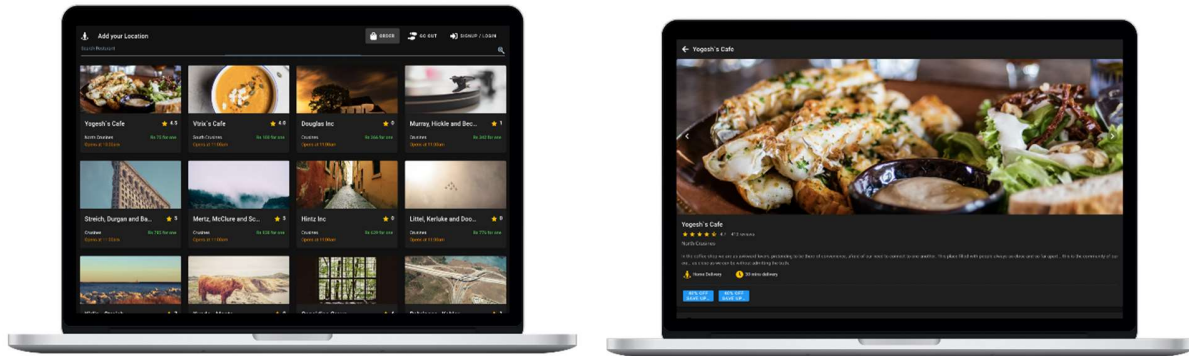
Food Delivery App

A UI/UX design for a Food delivery application based on a real life scenario made with Vue.js and Vuetify.

Key Features

- 1) Progressive Web App (PWA) supported.
- 2) Search among restaurants.
- 3) A responsive and mobile friendly design.
- 4) Bottom navigation bar for fast navigation.

Tech: **Vue3 framework**, CSS, HTML, Javascript, Vuetify



You and Community

What will you do if you get stuck on your project and your mentor isn't around?

When faced with such a situation, my first step would be to search for viable solutions on the internet. If that proves to be fruitless, I will turn to the Sugar channel on Element to contact other developers. The community there has always been very helpful and responsive, and I have no doubt that they will lend their expertise to assist me. Furthermore, I have access to senior developers in my college and industry experts I met during my internship to seek guidance from

How do you propose you will be keeping the community informed of your progress and any problems or questions you might have over the course of the project?

To keep everyone informed about the project, I will maintain a blog where I will post regular updates on the progress made, as well as any obstacles encountered and how they were overcome. For weekly progress reports, I will submit pull requests to sugarizer for anyone in the organization to view. In case of any problems or queries, I will rely on IRC, which I used during the proposal period, and keep in touch with my mentors via email.

Commitments after Google Summer Of Code

I would like to contribute to the project even after this summer. I would like to contribute to the community by solving queries and reporting issues. I wish to be an active member of this GSoC community and add value to the group by exploring various opportunities and mentoring my juniors.

~~End of Proposal~~