Improve Sugarizer Server Dashboard

25.05.2019

Basic Details

Full Name: Nikhil Mehra

Email: nmehra@ph.iitr.ac.in (College Email),

nikhilmehra998@gmail.com (Personal Email)

GitHub Username: NikhilM98

IRC Nickname: NikhilM98

First Language: Hindi, fluent in English

Location and Timezone: Roorkee, Uttarakhand, India (UTC +5:30)

Institute: Indian Institute of Technology, Roorkee

Degree: Bachelors' in Technology (B. Tech)

Major: Engineering Physics

Graduation: 2020 (Next Year)

Resume: LINK://to.resume

Share links, if any, of your previous work?

1. **Thomso, IIT Roorkee:**
   Thomso is the annual cultural fest of IIT Roorkee. I have been involved as a volunteer, junior team member and now as the technical head responsible for building and maintaining the web presence of Thomso. The website gathered around 1M hits around the globe; Reached 25,000 users. It was built using various technologies including PHP, Node.js and ReactJS over the years.

   Link: **https://github.com/pv-912/Thomso-18**.

2. **Internship at IoTrek:**
   I interned at IoTrek in the summer of 2018 where I converted the dashboard based outdated version of AngularJS to ReactJS, remodeled the functionalities, and integrated new features.
3. **Internship at DocConsult:**
   I did my first development internship at the DocConsult startup during in the winter of 2017 where I gained my first experience of working in a company and learnt to work under deadlines. The project was based on PHP and I worked on fixing bugs and improving the UI.
4. **First open-source issue and PR at Uber Deck.gl**
   During my intern at IoTrek, I was facing issues with the implementation of the features according to the requirements of the field. I was motivated by my mentor to create my first issue **#2012** on the Deck.gl. I tried finding a fix and created PR **#2013** on the repository, though it was turned down.
5. **Contribution to Probot**
   In the winter of 2018, I was looking forward to contribute to open source again. I created a few pull requests on Probot:

   #782 (Merged): [FIX] PRIVATE_KEY_PATH error handling (#778)

   #784 (Pinned): [FIX] Port in use error handling (#767)

Motivation

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

My primary motivation to apply for my first GSoC is to get started with open-source in a way I have yet to experience. I have utilised a lot of open-source technologies in my projects. But I haven't had a chance to work on software that has a tangible effect on millions of people around the globe. The prospect of developing software actually capable of making a difference means a lot to me and I would very much like to be able to give back to the community in some way.

Why did you choose Sugar Labs?

To be very honest, I had no clear idea of what Sugar Labs was or did before I participated in GSoC this year. I have always wanted to do something for society. Sugar Labs presented me with the means by which I can learn and give back to society by providing children with equal opportunities to learn, irrespective of their background.

I have been contributing to Sugar Labs since February and my journey so far with has been a great one. I am fortunate to have met this vibrant community of motivated developers and creators. Every pull request I have made to Sugar Labs has taught me something new and interesting, which was only possible because of the presence and feedback of the community on issues and pull requests.

Even before GSoC has begun, I have learned so much, which only makes me look forward to all the more things I could learn from this endeavour.

Why do you want to work on this particular project?

I've worked on dashboards during my former projects. This project has a familiar tech stack so I will be more comfortable in contributing to the project and thanks to the familiarity of codebase, it will help me focus more on feature implementation rather than figuring out workarounds for bugs in the code.

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

During the progress of the project, I expect at least weekly feedback and suggestions from the mentors. After the successful completion of the project, I plan to continue my contributions to Sugar Labs. Even afterwards I will be looking forward to possible mentorship opportunities from Sugar Labs.

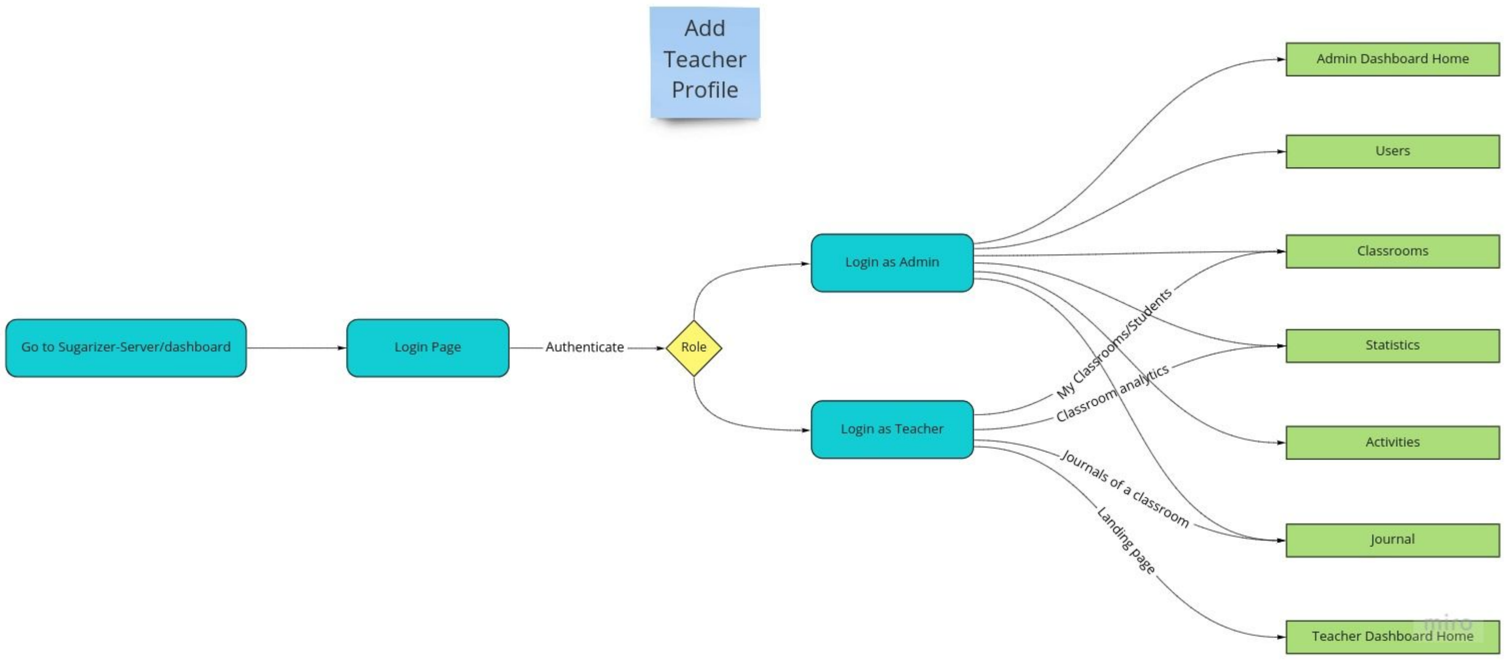
Project Details

What are you making?

The goal of my project is to improve the existing Sugarizer-Server Dashboard. My proposed enhancements include (The list may grow as discussion progresses):

- A profile for teachers. The teacher would have access to
    - View the profile information of students of their classroom.
    - Access journals of students of their classroom and launch them exactly as it was on the student's device.
    - View and compare the statistics of the students related to their classroom such as active students, favourite activity, assignment submissions, etc.

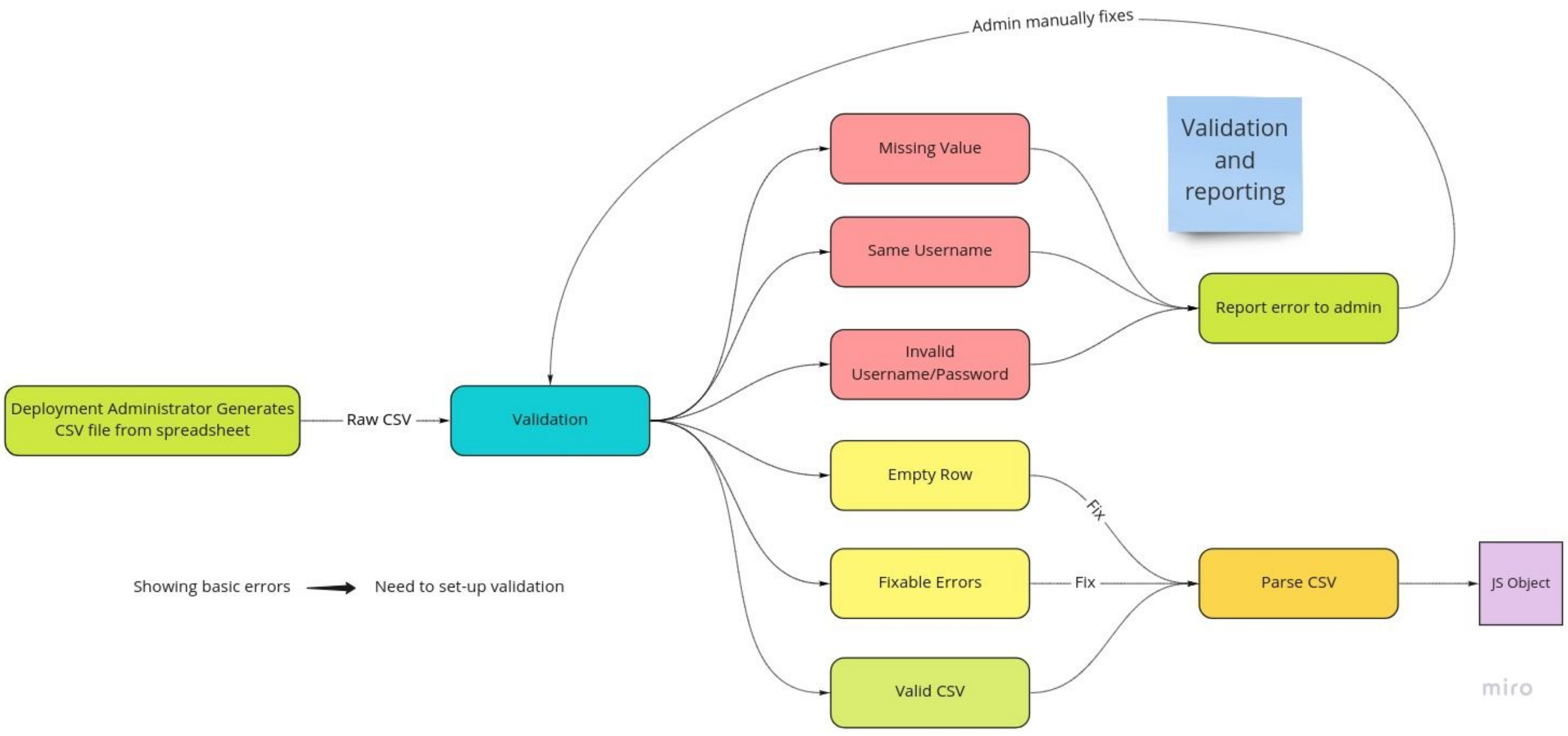
Creating a profile for the teacher would require updating the existing API, creating and updating the functionality and structure of the views, and updating routes to handle the new teacher role.



Flowchart for user login

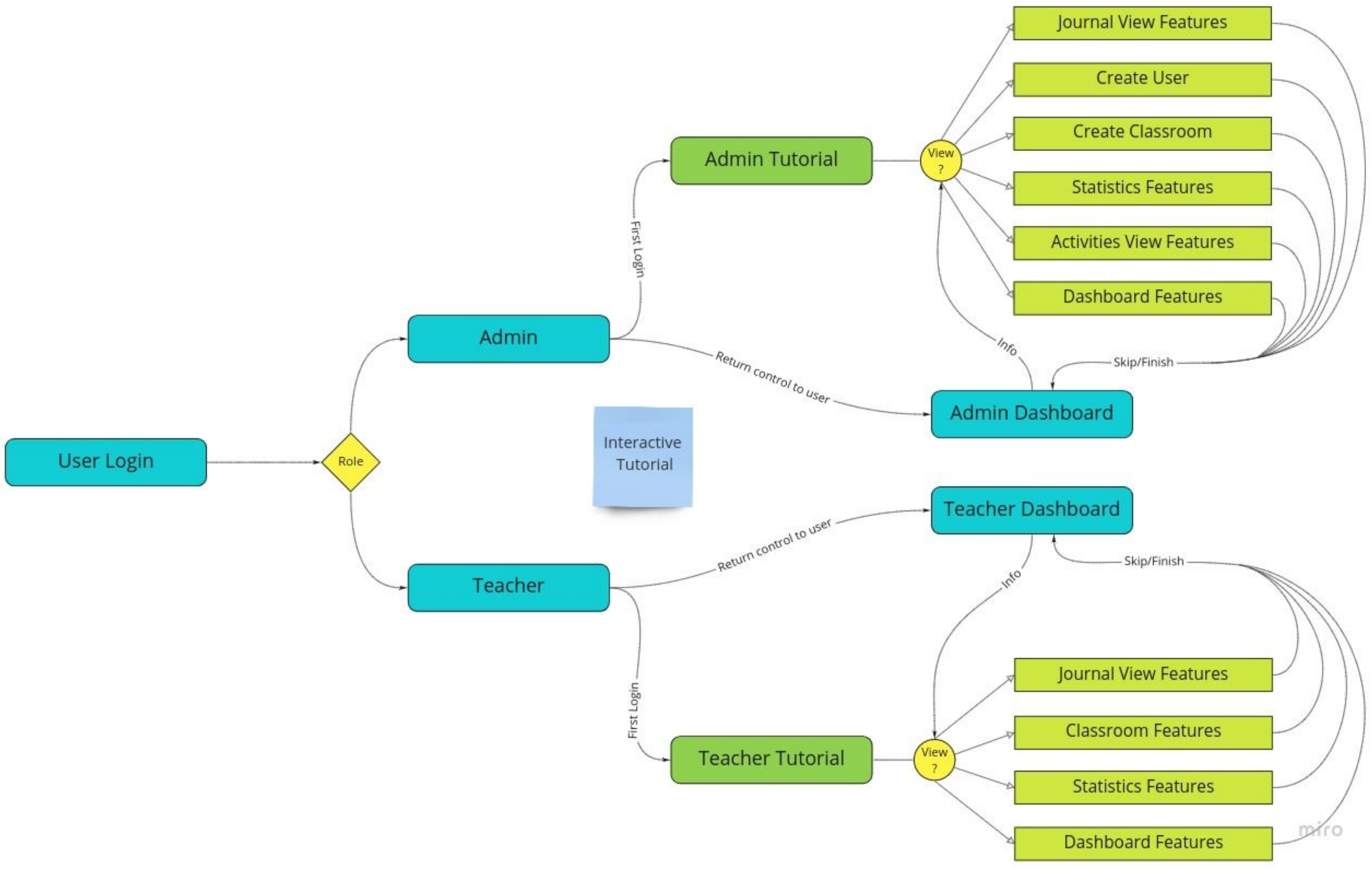
- Extend the Sugarizer-Server Dashboard interface.
  - Make the application UI fully responsive over the progress of the project so that it can be accessed from mobile and tablet devices. It includes improving the efficiency of the existing features and making sure that the application can run smoothly on low performance devices.

  - Add a feature to import and export the users and classroom data to CSV files. The CSV data generated by the school faculty has high possibility of errors and may not be inserted directly to the database, it requires parsing and validating the CSV data, comparing the data with existing database data, and identifying and reporting the issues with the CSV file to the administrator so that he can request changes from the teachers.



Flowchart for import users from CSV

- Add an interactive (step by step) overlay tutorial in the application to guide the users through the functionality of the dashboard depending on user role. The tutorial will run when a user logs in the application for the first time. In the later stages, the user will get an *info* icon on his dashboard, clicking on which will give user the workaround for the active view.



Flowchart for interactive tutorial

- Extend Sugarizer-Server Analytics:
  - Generate more analytics from the collected data. Currently, the statistics data collected is primarily used to generate analytics related to Sugarizer deployment. The analytics can be extended to derive the relations between the behavioral patterns of the students. The current analytics also lacks time series comparison because of which it is not possible for the institute to monitor the progress with respect to time.

  - Make the statistics page more flexible by allowing the user to plot the desired parameters on different possible types of charts. The most commonly used charts will be automatically seeded for the users when the user is created (So that inexperienced users don't face any issues). The user can add more charts using an interactive interface which will look like this:

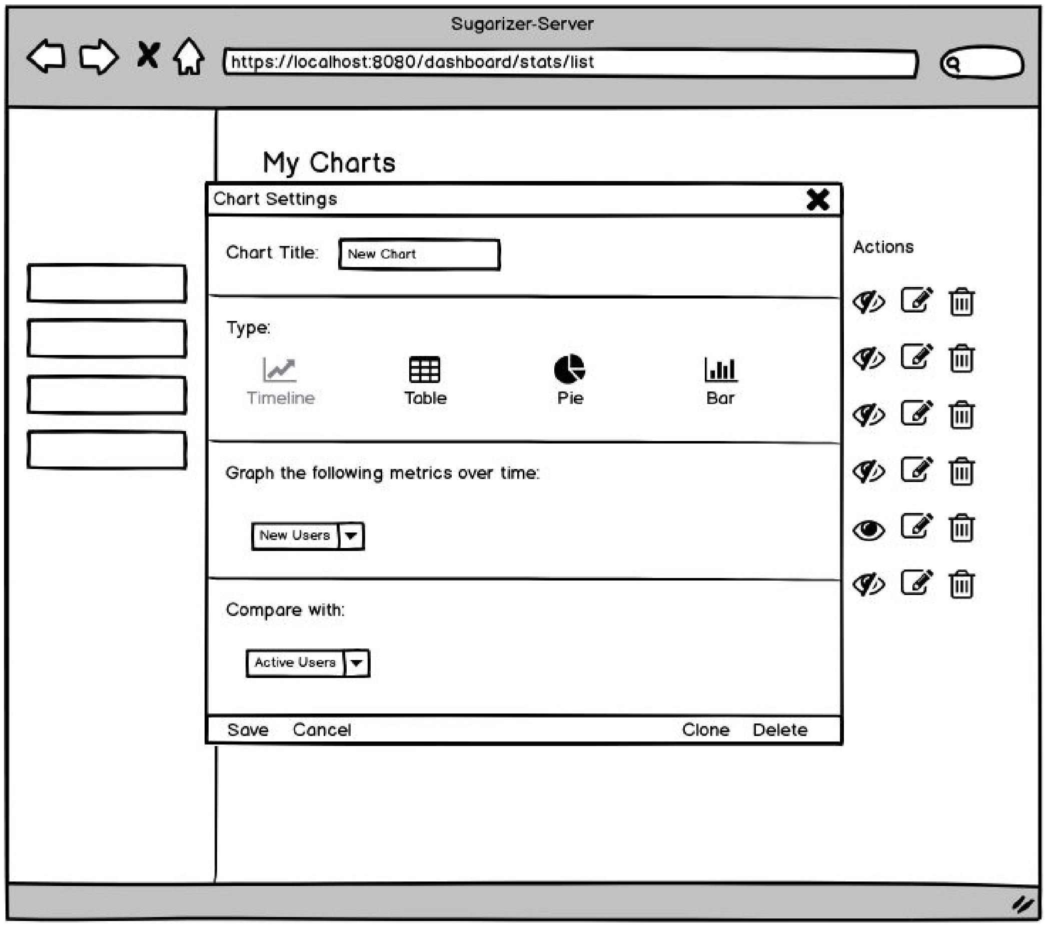


Chart Title: New Chart

Graph the following metrics over time: New Users

Compare with: Active Users

Modal box for add new chart (Type: Timeline)

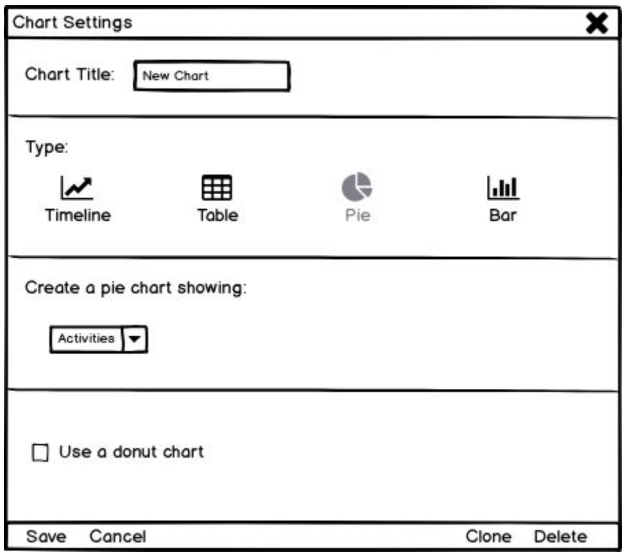


Chart Settings

Chart Title: New Chart

Type:

Create a pie chart showing:

Activities

☐ Use a donut chart

Save Cancel Clone Delete

Type: Pie

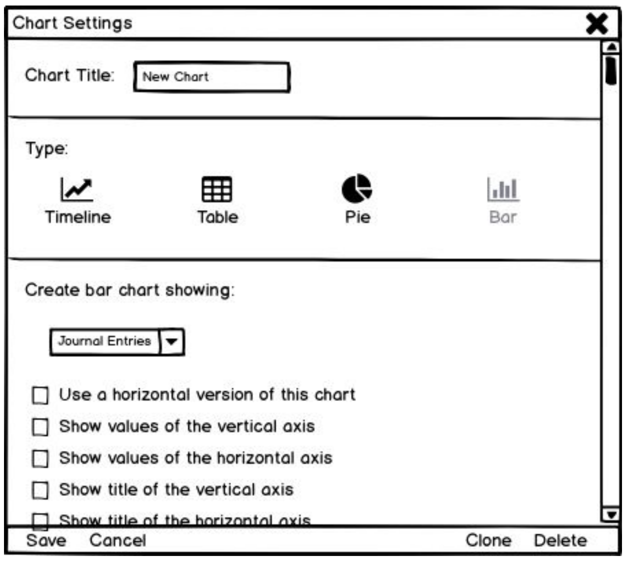


Chart Settings

Chart Title: New Chart

Type:

Create bar chart showing:

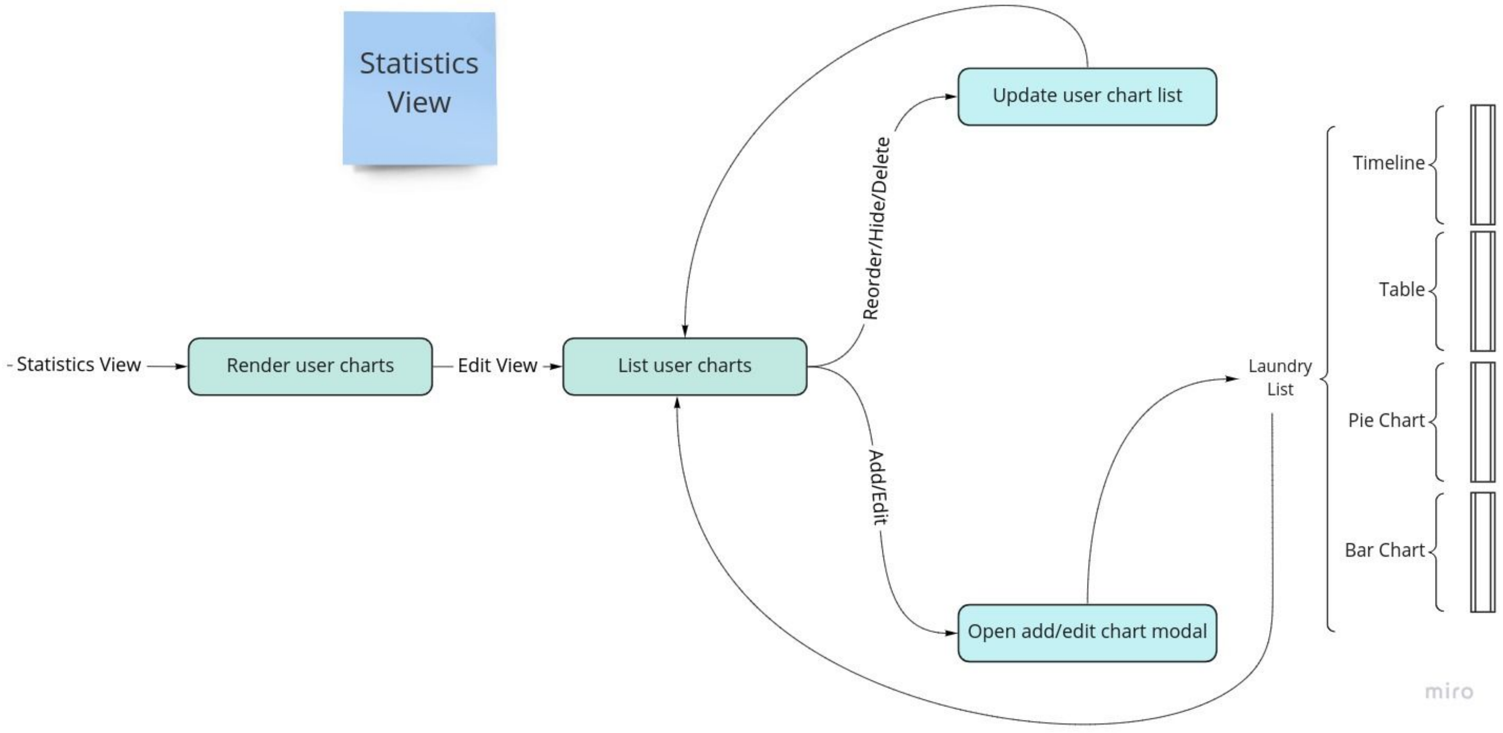
Journal Entries

☐ Use a horizontal version of this chart
☐ Show values of the vertical axis
☐ Show values of the horizontal axis
☐ Show title of the vertical axis
☐ Show title of the horizontal axis

Save Cancel Clone Delete

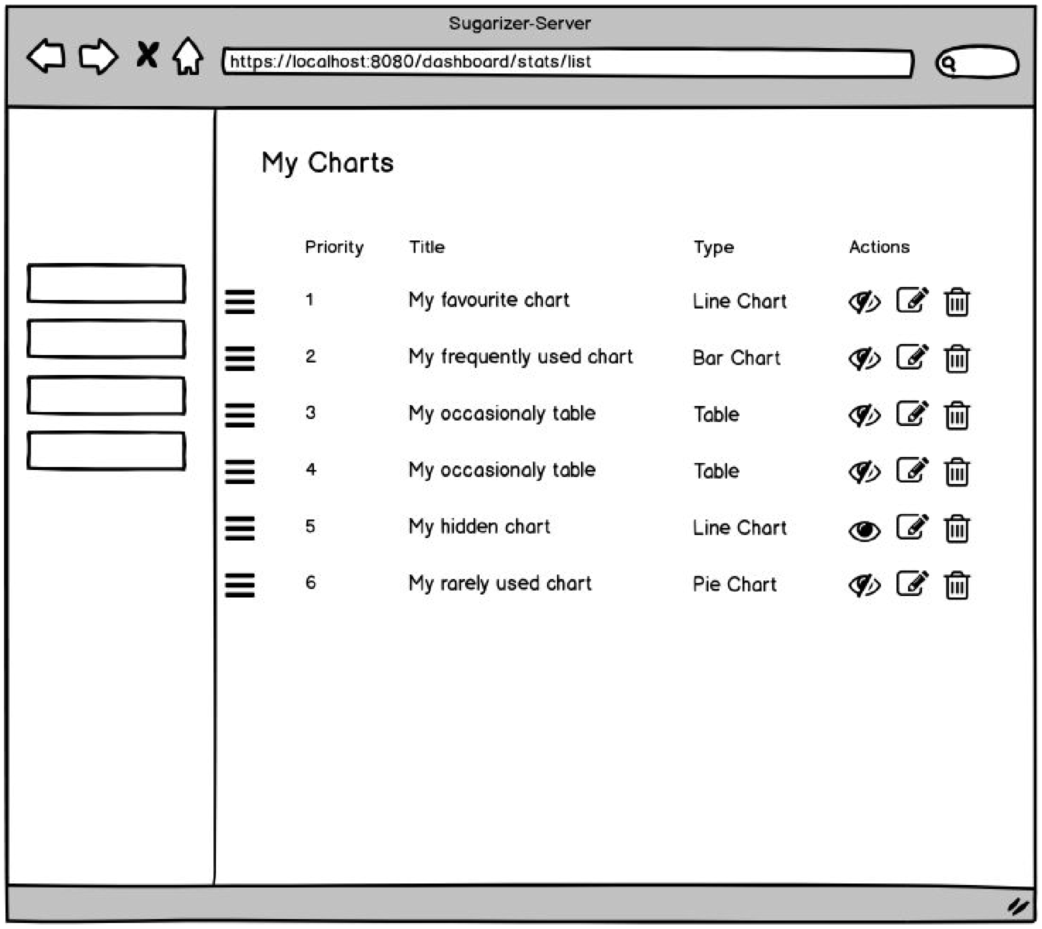
Type: Bar

The basic idea is to have a laundry list from which the user can select the charts he/she needs. However, it will be difficult for the user to look for the desired charts in a list view. This modal would help the user to look for the desired chart in the laundry list in a user-friendly way (Inspired by Google Analytics). The laundry list would actually be a list of JS Objects which would contain information about type of chart, parameters, and the name of the method used to generate the chart.



Flowchart for Statistics View

  - Make the statistics page customizable so that the user can select, reorder, and hide the charts that he/she wants to display on his/her statistics page.



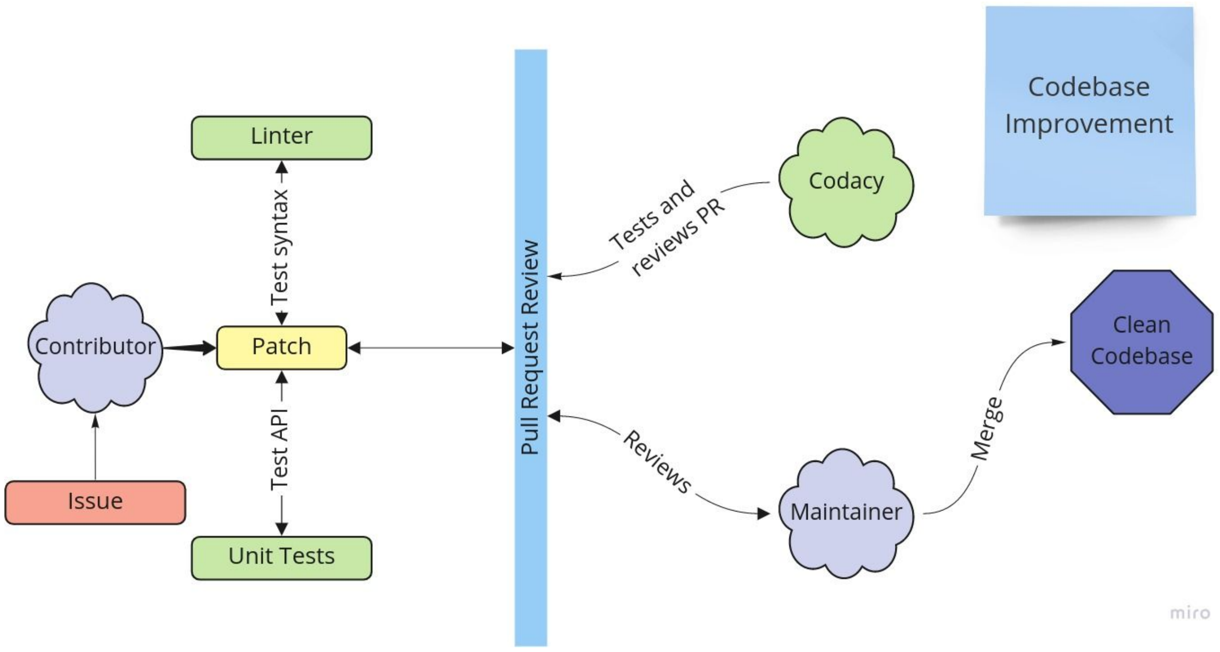
My Charts

Priority	Title	Type	Actions
1	My favourite chart	Line Chart	
2	My frequently used chart	Bar Chart	
3	My occasionaly table	Table	
4	My occasionaly table	Table	
5	My hidden chart	Line Chart	
6	My rarely used chart	Pie Chart	

List user charts view

- Add shell scripts with the functionality to:
  - Generate the settings file and initialize the database content.
  - Create complete backup of the database in a backup folder and restore data from the backup folder by simply running a script. It would greatly enhance the *Backup and Restore* functionality of the application.

- Improve code efficiency, application performance and codebase structure.
    - Some public resources for API docs and sugarizer server (Like JQuery and Bootstrap) are redundantly imported and some resources are outdated (Like moment). Non-minified versions for several resources are transferred over the network.
      Use Grunt to minify and manage public resources. It involves restructuring the build and deploy process of the dashboard.
    - Setup a linter (Specifically ESLint) in the project and eliminate the linting errors over the progress of the project.
    - Increase and maintain the coverage of the unit tests.
    - Integrate Codacy (free for open source) in the project which is used to check code coverage, code duplication and code complexity in every commit and pull request.



Flowchart for merging a clean patch

As far as code quality is concerned, I prefer elegance over hacks for long term codes. It takes some time to understand a code base and write code fitting the same and usually I invest this time and I hope that this will lead to satisfactory code quality. I have already read some part of the code base while fixing some bugs. This shall help me mirror and improve the code quality.

How will it impact Sugar Labs?

Abstract:

With the successful implementation of the project, the teachers will be able to remotely access the assignments and view real time statistics of the students. Improved efficiency and elimination of the device dependence would greatly increase the accessibility in a budget friendly way. Server setup and deployment would be optimized. The need for training to use the dashboard would be eliminated. With improved statistics, Institutes will be able to adapt and improve their teaching methodologies in a way never before possible. Improved codebase and automated code reviews would make contribution and maintenance easier.

- Currently the dashboard can only be accessed with administrator privileges by the deployment admin who has the permission to view all server statistics and manage all the users and their data.

  Current mechanism is missing a tier - the involvement of the teachers. In current scenario, the teachers require either the physical access to a student's device or the credentials of a student in order to access the students' journals. None of the mentioned ways is efficient and there is also a chance for the teacher of unintentionally modifying the student's journal.

  I want to fill the gap by introducing of a profile called teacher which will have the permission to view the of the students of the their assigned classroom, access their journals and launch the activities exactly as it was on the student's device. The ability to view the analytics of the classroom would enhance the ability of the teacher to monitor the performance of the students and help them improve their teaching methodology. This will greatly help in increasing the engagement on both student and teacher side.

- Extending the interface of Sugarizer-Server would improve overall user interface of the application.
    - Making the application mobile responsive will help improve the mobility of the users and help eliminate device dependence. The use case include:
        - Teachers would not need to worry about buying and bring a computer to the classroom. They can simply access the dashboard from their smartphones or tablets.
        - The improved mobility will give the teachers the ability to increase their reach in the classroom and give on spot illustrations.
    - In schools, the student's data is generally stored in spreadsheets. The feature to import CSV will make it easy for the deployment administrator to create bulk user accounts. The user friendly validation will also help the administrator to easily detect issues and make changes with the imports.
    - The overlay tutorial and instructions will eliminate the need to read documentation and training for non-technical users. This would help save time and eliminate the risk of teacher being confused with the interface during the class.

- Extending the Sugarizer-Server analytics would allow the educational institutes to understand the impact of their teaching methodology using behavioral pattern and the time series comparison of students. It will allow the institutes to adapt to the change in behavior of the students and accordingly improve their teaching methodology.
  Customizable statistics page would make it efficient for the user work with multiple charts and it will improve the application performance on low performance device like mobile phones by loading only few selected charts.

- Using scripts will automate the task of server initialization, backup and restore will make server migrations easier for the deployment administrators.

- Improvement in the codebase has long term benefits.
    - It will make it easier to manage and update public resources. Minification would improve network efficiency so that it will take less time to load sugarizer server on weak network connection.
    - Linter would enforce good coding practices to the project and would greatly eliminate the possibility of the errors related to variable scope and declaration. Uniform code structure would make finding and eliminating bugs easier and it would also help in attracting future contributions.
    - Increasing and maintaining the code coverage and automatically checking for the errors on each pull request would make it easier for the maintainers to test and detect possible bugs and suggest improvements.

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

I will be using Node.js on the server side as a runtime environment, Express as web application framework and EJS as the templating engine. JQuery, Chart.js and Bootstrap will be used to design and make an interactive and responsive user interface with a bunch of customizable analytics. Apart from that, I will be using Shell for scripting. Most of the programming would be done in Javascript, HTML and CSS.

Timeline

Timeframe	Start Date - End Date	Task
Phase 0: Community Bonding	May 6 - May 12	- Interact with the mentors of the project and set up feedback loops. - Get familiarised with the code base. - Explore the technologies to be used in the project.
	May 13 - May 19	- Continue to refine the plans for the project in consultation with the mentors. - Discuss the structure of the project for the implementation of the teacher profile.
	May 20 - May 26	- Discuss the build and deploy process of the server for the integration of grunt. - FInalize the idea and structure of the project.
Phase 1: The goal of this phase is to implement the teacher profile feature.	May 27 - June 2	- Integrate Grunt considering possible project restructure discussed during bonding period. - Update the API to handle teacher creation and deletion. - Handle admin view for the creation of teacher profile. - Write the unit tests for creating teacher profile.

	June 3 - June 9	- Update the API and views to handle assigning the classroom to the teacher. - Restructure the global view to handle admin and teacher profile. - Write the unit tests.
	June 10 - June 16	- Work on the dashboard for the teacher. - Make the journals of a classroom accessible to the teacher. - Make basic student information of a classroom available to the teacher. - Make classroom statistics available to the teacher.
	June 17 - June 23	- Test teacher profile, fix bugs and finalize teacher dashboard interface. - Wrap up the teacher profile feature and prepare for first evaluation.

Phase 2:
The goal is this phase is to:
- Add a feature to import/export users/classrooms as CSV
- Extend the server analytics.
	June 24 - June 30	// First Evaluation
		
		// Start working on import/export using CSV feature.
		
		- Write the code to export the classroom and users as CSV.
		
		- Write the code to import and validate the CSV.
		
		- Identify and report issues with invalid CSV to the user.
		
		- Restructure the imported CSV data such as to make it efficient to do bulk queries.

	July 1 - July 7	- Modify the API to handle bulk user/classroom creation and finalise the import from CSV feature. - Write unit tests for import/export // Start working on extending analytics - Create API to handle the list of analytics for a user.
	July 8 - July 14	- Make the statistics view customizable. - Add more analytics. - Write the tests and fix bugs.

	July 15 - July 21	// Finalize the previous work before the evaluation.
		
		// Start working on overlay tutorial

Phase 3:
The goal of this phase is to:
- Add overlay tutorial
- Add shell scripts
- Finish codebase improvements
- Review the documentation
	July 22 - July 28	// Second Evaluation.
		
		// Finalize the overlay tutorial.

	July 29 - August 4	// Start working on scripting.
		
		- Write shell script to initialize the database.
		
		- Write shell script for database backup and restore.
		
		// Finalize the linter configuration and set-up Codacy.

	August 5 - August 11	- Buffer Week: Finish any remaining work.
	August 12 - August 19	- Polish up the UI and review the documentation. - Prepare for the final evaluation.
Finalize	August 19 - August 26	// Final Evaluation
Post-GSoC Period	-	Continue to work on the project, finishing items that have been put on an if condition in the last two weeks. Improve the codebase in aspects that will only come to light when integrations are getting merged.
Distant Future	-	In the distant future, I would love to work with Sugar Labs as they explore creative ways to provide educational opportunities to the children. I would also love to mentor new-coming contributors to the project in any way I can.

Contribution to Sugar Labs

To familiarise myself with the contribution and review process at Sugar Labs, I have contributed to two repositories. I made a number of pull requests to these repositories with invaluable input from Lionel Laské and Tarun K. Singhal. I have also been assisting the maintainers by reviewing pull requests by other users.

My significant contributions to Sugar Labs includes:

Sugarizer-Server:

- **#144 (Open):** [FIX] Re-initiated locale on non-index views
- **#143 (Open):** [FIX] Fix classroom for similar students
- **#141 (Merged):** [FEAT] Show view journal button only for students in users view
- **#140 (Merged):** [FIX] Fix API inconsistency for asynchronous calls
- **#138 (Merged):** [FEAT] Display name of the activity while deleting journal entry
- **#137 (Merged):** [FEAT] Display name in notification after create/update/delete
- **#135 (Merged):** [FIX] Fixed placeholder translation in addEditClassroom
- **#134 (Merged):** [FIX] Re-init l10n in stats controller
- **#129 (Merged):** [FIX] Fix launching activity with null text
- **#128 (Merged):** [FIX] Fix console error on bar chart onClick
- **#124 (Merged):** [FIX] Fix incorrect value in journal select field
- **#121 (Merged):** [FIX] Dropdown language change fix
- **#115 (Open):** [FIX] Delete private journal on delete user
- **#110 (Merged):** [FEAT] Translate stats labels. Corrected and added translations
- **#108 (Merged):** [FIX] Updated deprecated mongodb functions
- **#107 (Merged):** [FIX] Search for stroke and fill fixed
- **#105 (Open):** [FEAT] [WIP] Script to seed db with test data
- **#103 (Merged):** [FEAT] List admins after create/update/delete admin
- **#101 (Merged):** [FIX] Fixes breaking tests on dev. Handling admin deletion
- **#93 (Merged):** [FIX] Fix the ordering of activities while sorting
- **#92 (Merged):** [FIX] Classroom user count fix
- **#89 (Merged):** [FEAT] UX Improvement. Hide non editable empty fields
- **#86 (Merged):** [FIX] Fix add another classroom/user issue
- **#76 (Merged):** [FIX] Updated locale
- **#75 (Merged):** [FIX] Creating classroom with single student
- **#73 (Merged):** [FEAT] UX improvement
- **#72 (Merged):** [FIX] Added missing translations
- **#58 (Open):** [FEAT] [WIP] Installing linter in the project
- **#55 (Merged):** [FIX] Fixes broken dev branch
- **#52 (Merged):** [FIX] Server crash fix
- **#48 (Merged):** [FIX] Added missing Hindi and Spanish translations
- **#46 (Merged):** [FIX] Variable scope and typos issue fixed
- **#45 (Open):** [FEAT] Use single instance of db connection

ExerciserReact:

- **#2 (Merged):** [FIX] Fixes crashes in development mode
- **#1 (Merged):** [REFACTOR] Unused parameters removed from routes