CollaBEARate

Your one stop shop for organizing your work life.

Gargi Tawde
Dept. of Computer Science
San Jose State University
San Jose, United States
gargi.tawde@sjsu.edu

Prabhleen Bagri
Dept. of Computer Science
San Jose State University
San Jose, United States
prabhleen.bagri@sjsu.edu

Septarshi Sengupta
Dept. of Computer Science
San Jose State University
San Jose, United States
saptarshi.sengupta@sjsu.edu

***Abstract*—As software developers, it's important to stay in tune with the problems faced in the world, as technology can play a key role in solving them. However, it's easy to forget that the products solving these larger world problems are not possible without the people and systems in place that help get those projects from pen, to paper, to code. That's where agile/group work consolidation comes in. Products involve the coming together of multiple smaller parts, all managed by different teams. Each of these teams need good ways in which they can organize what they do in ways that work across the skillsets of different people (e.g. managers, designers, developers. etc.). There are a number of agile platforms already available, but not many (if any at all) consolidate scrum boarding, scheduling meetings, and project overview/resources into one place.**

Keywords—agile, kanban board, user authentication, .ics file, card, dynamically created rows

I. Introduction

The Problem—Agile/group work consolidation services are often separated into multiple platforms that only perform some of the desired functionality (i.e. you need to use multiple in order to create a fully robust system).

The Solution—Create a platform that combines peoples' desired group work/agile features into one place.

II. Implementation

A. Design

- Given the concept for the project, wanted to use wordplay to come up with a name and logo; settled on **CollaBEARate**
- To come up with a color palette, used the site coolors.co
- Logo was designed and made in Procreate
- There was one main user we had in mind for the current scale: a singular person that wanted to organize projects/work on their own.

B. The Tech Stack

Software: HTML, CSS, JavaScript

Other: Google Suite, coolors.co, Procreate, Adobe XD, GitHub, WebStorm IDE

C. The Features

1. Create Account
    - The user can navigate to this page from the Login Page if it is the first time creating an account
    - This page displays a form, prompting the user for four input fields: username, email, password, and password verification
    - There are certain validation requirements for the fields, such as username length and password matching. If the user clicks away from the field with an invalid input, the corresponding error message is displayed
    - The form cannot be submitted until all fields are validated. Once submitted, the user is directed to the user dashboard page
    - If user has already created an account, there is an option to direct straight to the login page
2. Log In
    - This is the first form shown to the user when the log in button is selected on the launch page
    - This page displays a form, prompting the user for two input fields; username and password
    - If the user clicks submit with an invalid input, the corresponding error message is displayed

- The form cannot be submitted until all fields are validated. Once submitted, the user is directed to the user dashboard page
- If user has not created an account to login with, there is an option to direct to the signup page where user can create an account first

3. Projects
   - A table can be viewed that holds an empty base row that user can edit with information about a project.
   - If user is ever confused on functionality, they can select the ⓘ button next to the Projects title to view use instructions.
   - User is able to designate project name, description, their role, a deadline, and can use a selection menu to set/update the project's status.
   - User can click the 'Add Row' button to generate a new empty row to populate.
   - User can use the 'Select' column to pick one or more rows to then delete by clicking the 'Delete Row' button.
4. Pencil Me In
   - An empty form is viewable upon opening that page.
   - If user is ever confused on functionality, they can select the ⓘ button next to the Pencil Me In title to view use instructions.
   - User must fill in information as specified in order to generate a meeting invitation in .ics format.
   - After form is complete, user must click 'Submit' to trigger the file download.
5. Backlog
   - An empty Kanban board is viewable upon opening the page.
   - Is user is ever confused on functionality, they can select the ⓘ button next to the Backlog title to view use instructions.
   - User can select the '+ Add' button in any column to create an empty card that can be filled out by clicking on it.
   - To delete an item, double click and confirm your change.
   - To move an item between columns, click and drag item into desired area.
     - Then refresh page to see updated board
6. Account
   - A user can view their account information on this page.
   - No alterations to this data are available at current scale.
7. Log out
   - After clicking the 'Log Out' button in the navigation bar, users are redirected back to the launch page.

D. How Each Feature Was Developed

1. Create Account
   - The validation and form routing were implemented using event listeners and functions to check error cases in JS
     - This set of functions ensured that legitimate information was inputted, the correct error messages were displayed and the user was routed to the correct page
   - The user authentication was implemented using cookies, REST API and Express.js framework
     - JS was used to create a connection to server using Express.js
     - Next, API endpoints were created to handle all the user logins. The users created were authenticated by checking against of a list of already created users. For new signups, users were assigned a generated cookie which is used to validate future login attempts.
2. Projects
   - A JavaScript script was programmed to trigger an alert containing usage instructions after clicking the ⓘ button.
   - The function addRow() is triggered when the 'Add Row' button is clicked.
     - The function proceeds to dynamically generate code for each part of the new row in the table (checkbox, text input, select menu)

- The function deleteRow() is triggered when the 'Delete Row' button is clicked.
    - Since this is a table element, can simply delete a row in its entirety rather than having to access each part of the row and deleting each one at a time.

3. Pencil Me In
    - A JavaScript script was programmed to trigger an alert containing usage instructions after clicking the ⓘ button.
    - On the HTML side, code was written out to create a form, each text input area attached to a unique ID.
    - On the JavaScript side, each input provided was pulled using getElementById(), then assigned to a variable after making any necessary adjustments to their format.
        - These variables were used as parameters for a function called CreateDownloadICSFile() that would generate a meeting invitation with that custom information.
        - After the meeting is generated, waits for a button click in order to begin the download.

4. Backlog
    - A JavaScript script was programmed to trigger an alert containing usage instructions after clicking the ⓘ button.
    - A custom API handles insertion and deletion to the kanban, as well as what should happen when an item is updated.
    - Items in the board are stored locally, and can be moved between columns, updating in real time and requiring a refresh of the page to make sure they are properly viewable.
        - After an item is dropped into a new column, updateItem() is called so that its designated column is updated.
    - After deleting or updating an item, table's values stay updated even after a page refresh.
    - Upon creating a new item, code for a new card is dynamically generated and inserted into the preexisting page.
    - Styling was also very important for this page for things such as displaying a drop zone (i.e. on hovering a dragged item over an area, it would highlight where the item would be inserted into the column).

5. Account
    - HTML and CSS are used to display the user account information as well as a form to submit zoom meeting room information
    - Future development of this feature will use the REST API and CRUD operations to fetch the user information and update the username or password as indicated by user. See section VI.

III. USAGE

1. Start by opening the launch page.
2. Create an account by clicking 'Sign Up' and entering information into the provided form.
3. Click 'Login' to begin using the profile you just created.
4. Click on the 'Projects' tab in the navigation bar at the top of the screen.
5. Select the ⓘ to view usage instructions and proceed to use the page as instructed.
6. Click on the 'Pencil Me In' tab in the navigation bar at the top of the screen.
7. Select the ⓘ to view usage instructions and proceed to use the page as instructed.
8. Click on the 'Backlog' tab in the navigation bar at the top of the screen.
9. Select the ⓘ to view usage instructions and proceed to use the page as instructed.
10. Click on the 'Account' tab in the navigation bar at the top of the screen; can view account information.
11. Click on 'Log Out' in the navigation bar in order to log out of the service and return to the launch page automatically.

IV. TESTING

When it came to testing the product, it was done by repeatedly going through the motions of users, both new and current.

It was imperative not just to test whether or not a user could sign in, but if they could log in using those account credentials after the fact. A few factors and edge cases needed to be considered in this form validation. Firstly, in the event that there is a large set of users signing up, there needs to be enough varied usernames. To ensure this, there was a minimum length requirement implemented for the username which is validated. This way, there minimizes risks of repeated usernames. Testing this feature was done through logging in with several cases of usernames, emails, and passwords. The form was working as anticipated.

Another important aspect of the login/signup feature was user authentication. This was to ensure that each user could have an account associated with their name and access the data when they logged in. This was made possible through the use of storing the user data using cookies. Testing this feature involved launching the server and creating mock accounts to save to the list. The next step was attempting to log in with the same credentials. When this process was done locally using a REST client file to send a POST request, the console showed the data being saved and fetched successfully.

However, when this feature was implemented along with the rest of the application, there arose an issue with inconsistencies in server ports. As a result, proper testing could not be conducted. This issue will be discussed further in section VI, which pertains to future developments.

From there, it was time to test each feature of the product. When considering Projects, testing was focused on the ability to add and delete rows, as well as edit information that was already inputted into the table. It was also important that users could change their project status, so testing that was crucial as well.

The Pencil Me In feature required the most testing, as it was important to make sure that users were provided with the proper information on how to input desired information. For example, the time had to be in a 24-hour format, so it was crucial that we tested times after 12 to make sure that information was used correctly in the meeting generation. We also had to test to make sure the zoom meeting link was being used as a meeting location, rather than just being a part of the event description. By testing the form multiple times, we were also able to ensure that this meeting invite not only downloaded, but was able to be used to generate an actual meeting.

For the Backlog, testing was centered around making sure the usage was simple, effective, and user-friendly. In order to make the product feel intuitive, it was apparent that it should feel like not just adding but deleting tasks is an easy action for the user to take. It also made sense to test as many possible combinations of actions as possible; not every user is going to interact with the product using the same tasks or order of tasks. Users should be able to add, delete, or reorganize cards in any order they see fit, and that is what was tested for.

The account tab was intended to show the user their account information including the username, password, and zoom meeting room. This feature did not have as much room for error as it heavily relies on the users input, but there were a few features which needed testing such as changing the password. For the current status of the program, the change account info features just change the info as displayed on the screen, but there are no modifications to a database of user info. Testing was simply to ensure that the input fields and buttons changed the relevant info and that routing was correct. This would be important so that with the integration of a database, the feature would work and allow users to modify their account data.

Finally, the logout feature is straightforward and allows the user to leave the program and sign in again later, or with a new account. This feature does not need to make any modifications to the account info or database of users. It simply redirects the user back to the login page.

V. CollaBEARate and How it Connects to CS 152

Functional programming and object oriented programming was crucial to making this project work. Web design is code-heavy in many ways and concepts such as the aforementioned streamline it quite a bit.

Consider the Projects page and the Backlog, for example. For both pages, HTML code was dynamically added through the use of JavaScript every time 'Add Row' or '+ Add' was clicked by the user. This was made possible by functions that would perform a desired set of actions repeatedly whenever those buttons were clicked. Rather than hard-coding hundreds of lines and dealing with enabling/disabling their view on the page, it made far more practical sense to just trigger a function that would "write the code" for us, making the product by consequence truly customizable in the amount of cards/ rows a user could add now being more than a set number.

Another way in which the project was benefitted by material from the course was in its use of OOP: object oriented programming. By creating classes and programming functionality into methods within them, we were better able to organize the project. Parts of functionality could be separated into sub-problems, such as displaying items from a column and adding/deleting items. Then, different subparts would be called upon in other places to create code that is far easier to understand and much shorter in length as well.

Overall, CS 152 provided very useful concepts/ knowledge when tackling this semester-long project. From helping better organize the code, to shortening the amount of hard-coding in and of itself, functional and

object oriented programming were very useful for making CollaBEARate work.

VI. Future Work and Concluding Statements

As the project comes to a close, it is important to consider where one could take the product as it currently stands from here. After going through the many stages of our project, there are some ideas that immediately come to mind:

1. Add functionality that would allow user to store data under each project (e.g. README, assets, its own Kanban board, team members, etc.)
2. Allow for the meeting invitation generator to create invites in formats other than .ics files.
3. Allow users to specify event timezones other than PST.
4. For the Backlog, add a 'Set Priority' feature to each card.
5. For accounts, allow users to store a zoom link to their personal meeting rooms so it can be pulled for meeting generation rather than them having to input it manually into the form every time they wish to generate a meeting invitation.
6. Complete the user authentication which can save user account information for accessing data in future login attempts. Fix server issue.

Acknowledgment *(Heading 5)*

CollaBEARate was a project developed as a final project for our CS 152 course at San Jose State University. We'd like to thank our instructor Saptarshi Sengupta for giving us the necessary knowledge and support to come up with and create this product over the course of the semester.

References

The following references cite different sources we used in order to develop our desired product and its features.

1. dcode. *How to Build a Kanban Board with JavaScript (No Frameworks)*. (Sept 20, 2021). [Online Video]. Available: https://www.youtube.com/watch?v=ijQ6dCughW8
2. SouthBridge. *Javascript Tutorial: Make HTML Table content editable NO Jquery.* (Oct 29, 2020). [Online Video]. Available: https://www.youtube.com/watch?v=uPBxzvSGIiA&t=125s
3. W3Schools. *HTML, CSS, and JavaScript resource.* [Website] Available: https://www.w3schools.com
4. Web Dev Simplified. *Build Node.js User Authentication - Password Login.* (June 22, 2019) [Online Video]. Available: https://www.youtube.com/watch?v=Ud5xKCYQTjM
5. dcode. *How to Build a Login & Sign Up Form with HTML, CSS & JavaScript - Web Development Tutorial.* (Aug 31, 2020). [Online Video]. Available: https://www.youtube.com/watch?v=3GsKEtBcGTk