Spartan Project Reviewer

Kavya Sahai <kavyamohan.sahai@sjsu.edu>
Miguel Covarrubias <miguel.covarrubias@sjsu.edu>
Priyal Agrawal <priyal.agrawal@sjsu.edu>
Wayne Arnold <wayne.arnold@sjsu.edu>

Computer Engineering Department, San Jose State University
One Washington Square, San Jose, CA 95192, USA

***Abstract* -- Spartan Project Reviewer is a tool developed to streamline the process of finding, submitting, and approving projects for the Enterprise Software Platforms course at San Jose State University, though it can be applied elsewhere. It allows students to organize into groups and submit their ideas to a class section created by the professor. Once project ideas are approved, they can be automatically shipped over to an affiliated Github organization for further development of the aforementioned projects.**

Keywords -- Professor/Instructor, Student, Group, Project/Idea, Mail notification, Github automation, Content-based filtering, Google authentication, APIs, Express, MongoDB, Node.js, React, MERN, Python, Flask.

I. Introduction

It is important for students of software engineering to learn about new and emerging technologies in infrastructure to stay relevant in the field. This must extend beyond reading, so a class project is essential for learning these concepts and tools. In this, the guidance of the professor is crucial to determine meaningful and productive class projects. However, with such large classes, this becomes a herculean task for the professor to give the individual attention and guide students to these projects.

We propose that this system takes the standard process and streamlines these communication channels. This report will give details regarding the architecture, technology, and workflow expected for this process.

II. Architecture

This application is build using the *MERN* architecture (Mongo, Express, React and Node). It uses IBM Bluemix MERN base code line (Figure 2.1) as a containerized application ready for development. In Bluemix, the Git and Continuous Delivery were used (Figure 2.2). In addition, Machine Learning, Github and Google Login tools were incorporated into the application using basic web APIs in Python's Flask library to call functions on demand. In addition, the MongoDB instance of this application was hosted in IBM Cloud as a resource.

One of the web APIs calls the machine learning algorithm which does the content based filtering to suggest similar project ideas based on a particular project idea. The idea is that the professor would be able to find out if similar ideas have already been submitted by other students and can take an informed decision about which idea should be approved/ rejected.

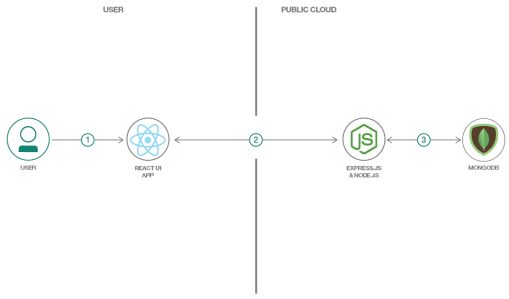

Figure 2.1: IBM cloud-based MERN architecture. [1]

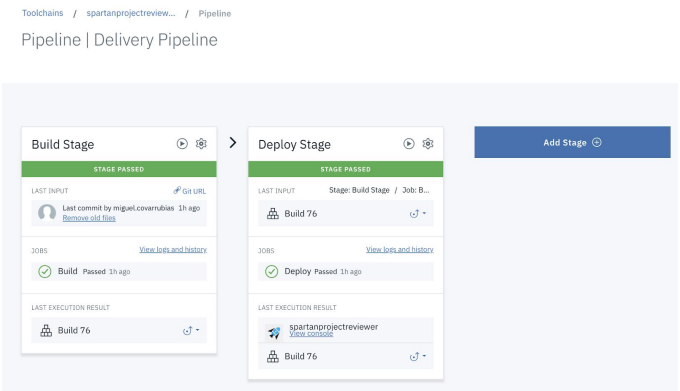

Figure 2.2: IBM pipeline continuous delivery

III. Process Flow

At the beginning of the semester, the professor will create a Github organization relevant to the course and semester. This would be something like “SJSU272LabS19.” He would then create an API key (specifically, a Personal Access Token) with organization access to make the changes. [2]

The professor would then come over to the system and create the class section, Github details in hand. Once the professor creates the course, it will generate a Github repository for each team with a generic name of “Project-Group-X”, with X being the team number. A matching Team will be created with access to that repository.

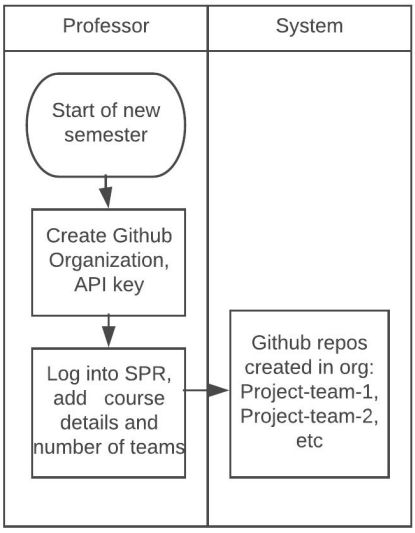

Figure 3.1: Beginning of semester workflow

A student will initially interact with this system as a generic user that is not logged in. In this view, the student will be presented with a dashboard view with some quick statistics about the system, and a search feature allowing them to search for ideas. The search feature will allow them to find ideas that are relevant to them.

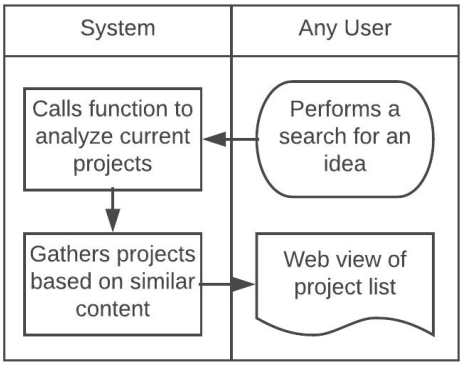

Figure 3.2 - Search feature

When the student decides they are ready to submit an idea, they will need to join a team, and then they will be prompted for user details. With those details, an invite will be sent using the student’s github username to the team that is associated with their repo.

Next, the student will need to provide an idea to be reviewed. They can choose from the list of ideas provided by the professor or submit one of their own.

Once the ideas are submitted, at some point the professor will log in to review ideas. He will have a few actions to choose from when looking at an idea. He can “star” the idea, reject it, or approve it. He can also like it, dislike it or add comments/suggestions. In each of these cases, the students of that group will be sent a notification regarding the professor’s feedback.

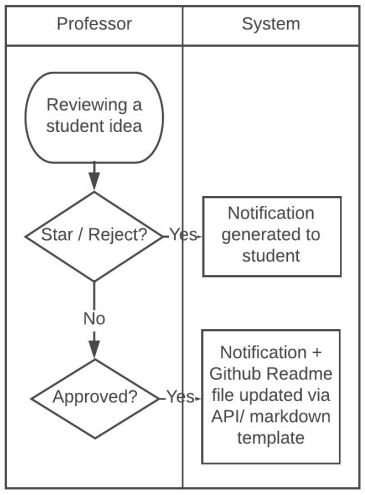

Figure 3.3: Professor/idea interactions

Based on the professor’s action, if the idea is approved, then that means that the students can begin working within their assign repo on Github. For this reason, the “approval” action will take the details (Title, Abstract, and Summary) and push them to the README file for that team’s repository.

IV. Application User Interface

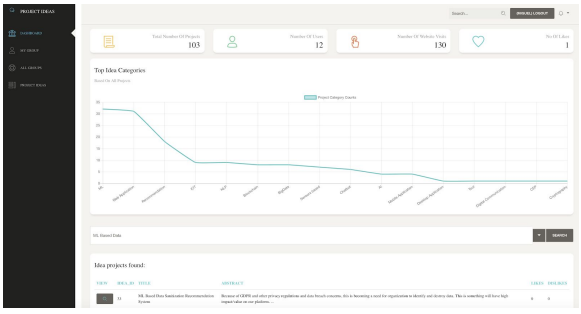

Figure 5.1: User Dashboard

Figure 5.1 shows the view of the application dashboard as the user logs in. This page shows aggregated statistics about all the projects in the application. This includes the number of projects, number of users and number of visits to the site. The page also shows a line chart of the top project categories. At the bottom of the page, the user can search for projects that are public or from the same class.

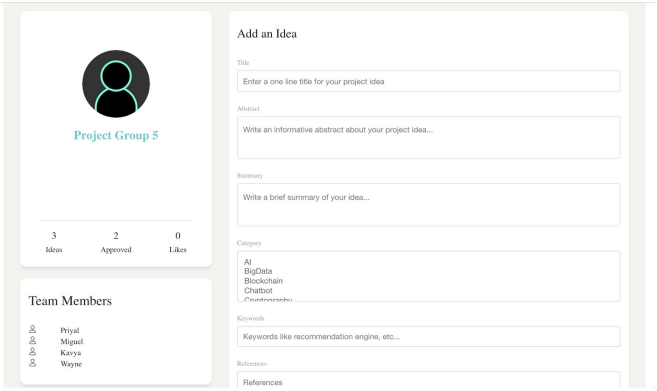

Figure 5.2: User Groups

Figure 5.2 shows the user groups page. This page shows the group associated with the current user for a given course. Information such as project name, project ideas, approved ideas and team members are displayed. In this page, the team members are able to create new ideas that will be associated with this group.

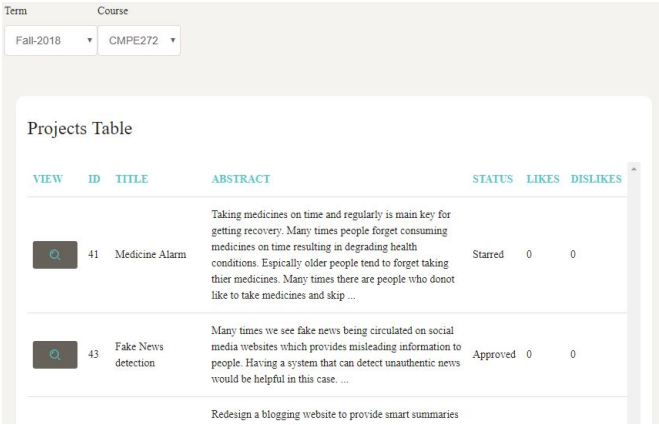

Figure 5.3: Project Ideas

Figure 5.3 shows the project ideas the user has access to. Each project in the table has a view action button and this is used to view the full project details. If the user owns one of the projects in the list then the user will be able to edit the project. If the user's role is a professor then the user will be able to approve or reject any of the projects.

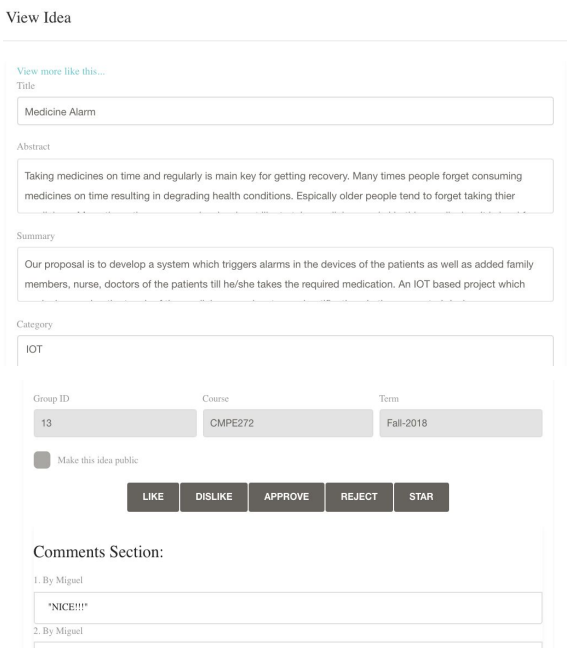

Figure 5.4: View Idea

Figure 5.4 shows the view of an idea. In this page, a normal user can view the idea in detail and it is able to perform actions such as "like", "dislike" and "comment". Only users with professor role for this class have the ability to "approve", "reject" and "star" the project.

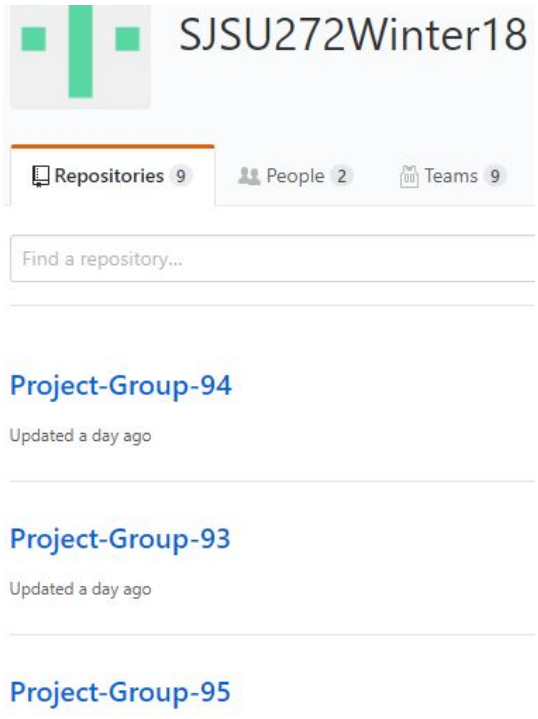

Figure 5.5: Automatically generated project groups

Once the section has been created, the teams are automatically generated in the pre-created Github repo. Initially, there will be a matching number of repositories and teams, and the members will be added as they are invited. Members added to the teams are also added to the organization.

V. Future Enhancement

Scoring system - this would be an instantaneous feedback for students to receive a "score" for their proposed project. The score would represent the professor's preferences, i.e. how likely the project is to be approved. Once this tool has gotten more use and the user has interacted with the system, one could leverage that data by using collaborative filtering. This would use NLP to find similarities to a set of past projects, and assign a weighted score based on how many of those projects were approved by the professor. This score would be positive by default, as we would want an innovative idea to be viewed favorably.

Canvas integration - this system already has integration with Google authentication, so it can be used to allow only university students to log in. However, this would be even more useful if it can integrate with Canvas and pull a list of students for the class. This could be used to automatically create sections and assign students to the view.

Code modularization and security access improvements. In order to scale this project, the base code should be more modular to allow for easy code reuse and management. The database and user authentication need to be revised to secure our data even further.

VI. Conclusion

The web application developed provides a single platform for professors and students to view/create the project ideas. It reduces professor's/TA's overhead of creating the repository for each group manually and sending invites to hundreds of students. It helps in the better organization of university-wide project ideas. It also helps professors to make an informed decision on whether an idea is unique or not by providing facilities to check if similar ideas exist. Also, provides general public facility to look for the public ideas.

VII. Acknowledgements

Thanks to Professor Ranjan for providing the intuitive IBM cloud resources that made this project possible and for allowing us to select a less conventional project topic that would directly benefit future classes.

VIII. Project Repository

Application URL: https://spartanprojectreviewer.mybluemix.net
Git Repository URL (private):
https://git.ng.bluemix.net/miguel.covarrubias/spartanprojectreviewer
Application Metrics URL:
https://spartanprojectreviewer.mybluemix.net/appmetrics-dash/

IX. References

[1] Build a MERN Web App on IBM:
https://developer.ibm.com/patterns/build-a-mern-web-app/
[2] Github API v3: Github Developer Guide:
https://developer.github.com/v3/
[3] Node.js: https://www.w3schools.com/nodejs/nodejs_intro.asp
[4] Flask: http://flask.pocoo.org/