CSCE 606 - Project Assignment System

Team: FellowshipOfTheRing

Semester: Spring 2020

Members: Abishalini Sivaraman, Abraham Gerard Sebastian, Akansha Agarwal, Aurosmita Khansama, Harish Kumar, Sahan Suresh Alva, Vaishnavi Chauhan

Two paragraph summary of the project as implemented, including the main customer need and how the application meets it, including who the stakeholders are. This will contrast to what you wrote in Iteration 0

Customer - Dr. Hank Walker (Instructor of CSCE 606 - Software Engineering)

The main customer need is to have a software system that allows the course instructor and TAs to manage and facilitate formation of teams, create and list all available projects, permit teams to provide their preferences for the listed projects, assign these projects to teams in accordance with their preferences, and to make pertinent stored information available to users who should have access to them. These users include the course instructor, the TA and the students who are taking the course.
The application, as delivered now, supports this need through a website with a set of pages that allow for user, team and project information to be entered, backend logic that retrieves relevant information from a database, views to render the retrieved information to the user (admins or otherwise), search functionalities to identify specific students through different attributes, delineation of privileges between users who are admins and those who are not, a method to move users between these two categories, and a migration tool to move over legacy projects to the next semester while cleaning up the database by purging outdated user and team records. Every user has an account that they use to log in to the website, and all operations that users can do on the site are associated with their account identity.

Stakeholders -

- Professor and Course TA (Admin)
- Students of CSCE 606 (Current and Future)

Application Objectives -

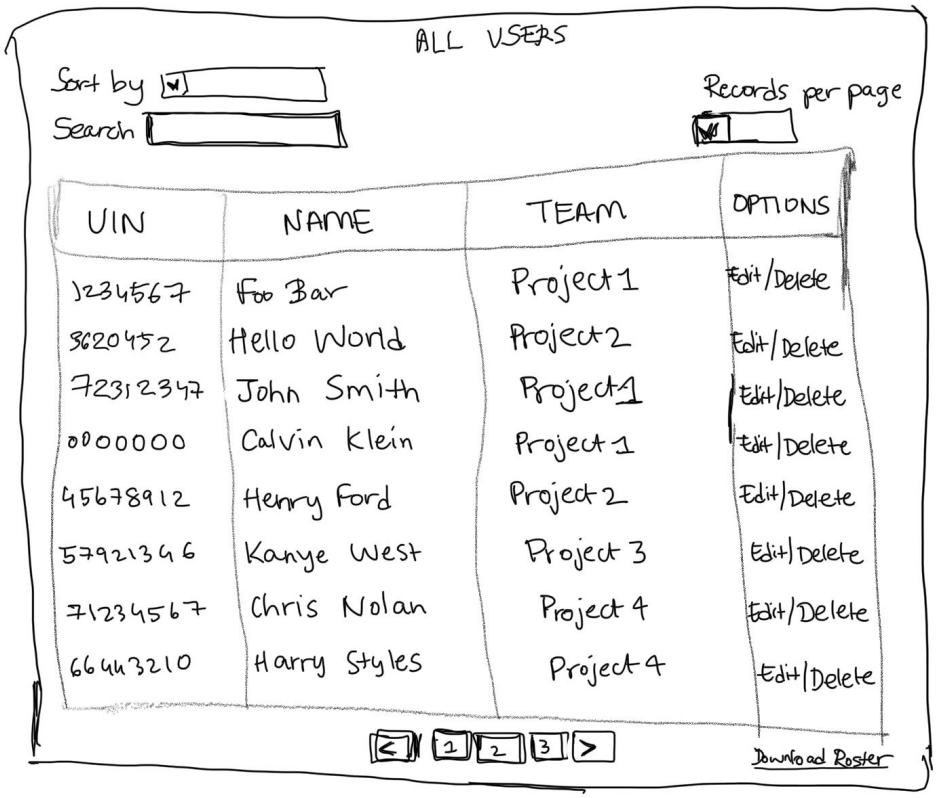
- For students: Manage formation of teams, list all projects, permit team-wise project preferences ranking and perform team project assignment.
- For admins: Allow deletion of users, teams and projects, maintain the projects database, allow the creation and removal of admins.

Description of all user stories (including revised/refactored stories in the case of legacy projects). For each story, explain how many points you gave it, explain the implementation status, including those that did not get implemented. Discuss changes to each story as they went.
Show lo-fi UI mockups/storyboards you created and then the corresponding screen shots, as needed to explain to stories

Main User Stories :

1) Feature: Reset Password (3)
   As a student, I want to reset my password when I forget it, So that I have an alternative way to login when I forget my password.

2) Feature: Search and Sort Students (3)
   As an admin, I want to easily search for a student's information from the database and sort any results thus obtained, So that I can find (and subsequently) edit information of a particular student.



3) Feature: Semester Migration (moves all legacy projects to the next semester and delete old users) (3)
   As an admin, I want to download all the information for the current semester and migrate legacy projects to a new semester, So that I can save time and easily ready the application for the upcoming semester.

4) Feature: Having multiple admins and being able to grant admin privileges to users
   As an admin, I want to grant/revoke admin privileges to new users, So that I can onboard other members of the instructor team onto the platform. (2)

Other User Stories (Smaller features implemented/fixed) :

5) Feature: Remove login using NetID (1)
   As an admin, I want all logins to be through accounts created on the platform, So that we do not depend on CAS.
6) Feature: Fix the excel download feature for users (only the first page is downloading currently) (1)
   As an admin, I want to download information of all users and teams into a spreadsheet, So that I can archive their information or maintain a backup.
7) Feature: Add personal email attribute to users so there is a non-tamu email in record
   As an admin, I want to add a personal email attribute to all users, So that I can have alternate contact information for all users when their TAMU email becomes inactive. (2)
8) Feature: Team Leader can change project preferences after initial selection
   As a Team Leader, I want the capability to modify already entered project preferences, So that I change my preferences after the initial selection if deemed necessary. (2)
9) Feature: Team members can see project selections made by their team leaders (2)
   As a Team Member, I want to see the project selections made by the team leader, So that I can validate the choices and propose changes if needed.
10) Feature: After deleting a user without team, redirect to list of users without team

As an Admin, I want to be redirected to the list of users without a team after deleting one
So that I can view other users without a team after deleting one. (0.5)

11) After editing a team, redirect to list of all teams (0.5)
As an Admin, I want to be redirected to the list of teams after making an edit to a team, So that I can continue to edit other teams.

12) Feature: Team leader cannot delete team if members exist in the team (1)
As an admin, I want team leaders to be unable to delete their team if it has other members, so that they do not take unilateral actions that do not have the team's approval.

13) Feature: A team with assigned projects cannot be deleted (1)
As an admin, I want a check on whether a team has an assigned project before deletion, So that I do not accidentally delete a team from the current semester that is responsible for a project.

For legacy projects, include a discussion of the process for understanding the existing code, and what refactoring/modification was performed on the code, in addition to the user stories listed above.

We started by forking the legacy repo and creating our development stack. We first identified the process to deploy the project locally, and then on Heroku, and documented these in the documentation folder. After this, we listed several small tasks (e.g. remove the NetID button or fix the download XLSX bug) that each of us could take up and finish in the first sprint. By working towards the completion of these relatively simple tasks, we were able to understand how the codebase was structured.
Further, by writing tests for the small tasks that we were working on, we were also able to understand the existing test infrastructure, the helper functions, and shortcomings with it.
We used the built-in debug method to understand which controller and action were triggered for various user inputs into the application by rendering this to the web application (only in development).

Refactoring on Legacy code:
Apart from implementing features and bugfixes, we saw that the test helpers were severely outdated, causing most of the tests in the code to fail. Coverage was below 35%. One of the main issues here was that the test helpers were still assuming that names of users are stored as one column, while in reality they are constituted of Firstname and Lastname components. We rewrote several test helpers that were making this incorrect assumption. Along with other improvements that we made to the tests code, and the new tests that we added, the coverage now exceeds 70%.

List who held each team role (Scrum Master, Product Owner, etc.)
Describe any changes in roles during the project.

Scrum Master - Abishalini Sivaraman
Product Owner - Harish Kumar

There were no role changes during the course of the project.

For each scrum iteration, summarize what was accomplished and points completed
Iteration 1 - Update at 03/23/2020. Points completed: 5

- Reset Password to UIN (temporary fix to Reset Password feature) (1)
- Removed login using NETID (1)
- Fixed download XLSX feature which previously downloaded the first page of users (2)
- After deleting a user without a team, the admin is redirected to the list of users without a team (0.5)
- After editing a team, the admin is redirected back to the list of all teams (0.5)

Iteration 2 - Update at 03/04/2020. Points completed: 7

- Implemented Reset Password using link sent via email (2)
- Implemented Search Bar to find students using UIN, Name or Team Name (3)
- Added personal email attribute to users so there is a non-tamu email on the record (2)

Iteration 3 - Update at 04/17/2020. Points completed: 6

- Admins can grant admin privileges to other users (2)
- Team Leader can change project preferences after entering it once. (2)
- Other members of the team can see the project preferences selected by the team leader. (2)

Iteration 4 - Update at 5/01/2020. Points completed: 7

- Semester Migration page - moves all legacy projects to semester specified by admin. (3)
- Team leaders cannot delete team if there are members (1)
- A team with assigned project cannot be deleted (1)
- Admins can see project selections made by each team (2)

List of customer meeting dates, and description of what happened at the meetings (Eg: what software/stories did you demo)

Initial meeting with the customer (D. Walker) - 2/21/20, 3:30pm
TA Meeting 1 - 04/23/2020, 12.45pm
TA Meeting 2 - 04/30/2020, 12.45pm

Regular email updates to the customer were sent on 3/24, 4/6, 4/30 and finally on 5/4.

Explain your BDD/TDD process, and any benefits/problems from it.

Our process was to first understand the existing code that may be relevant to our feature, look for whether they are any tests for the existing code, and to either update or write new cucumber/rspec tests to cover the requirements of the new feature. These often required bigger updates to the test helpers since many of those were broken or outdated. We decided the structure of the rspec tests using what we thought the low-level design would look like.

After this, we implemented the feature incrementally while manually testing via local deployment. Once we believed that what we were seeing on the local website was meeting requirements, we ran the rspec and the cucumber tests. Rarely, we noted that the cucumber tests themselves had a few residual errors that needed to be fixed, but in most cases, the cucumber tests succeeded when the manual inspection on the local server passed, or they failed and pointed out something that we had overlooked during manual testing.

The benefit from such a process is that we were able to think of all the different possibilities while writing the tests, and we did not have to repeatedly worry about forgetting those possibilities during implementation. Since we had put in a lot of thought into the tests, we could assume reasonably that passing those meant that we had covered most of the required paths, probably all of them. The downside is that this is a long process that requires several iterations before resulting in a "green" result for all tests. In addition, the previously broken test infrastructure required significant amounts of work before we could write our own tests.

Discuss your configuration management approach. Did you need to do any spikes? How many branches and releases did you have?

The only instance in our case was a technical spike where we did not initially know whether the password reset via email feature would work correctly, since we had to work with a third-party email API. We instead took a different initial approach where the password would instead just be reset temporarily to the user's UIN. This way, we were able to build the rest of the feature and have a working product in place. In the next sprint, we got the email API to work and completed the feature as required.
We followed a one-branch-per-feature approach where developers created a branch for every new feature that they were building, and raised a pull-request to be merged into master when they were done with implementing the feature. Upon approval and merger of the pull-request, we rolled out the code from master into Heroku and this served as our latest instance of the project. We did not maintain separate release branches or versions since we deemed this unnecessary - there are few advantages in maintaining multiple release versions.

Discuss any issues you had in the production release process to Heroku

We initially had no documentation on the process to deploy the project to Heroku. We took some time to figure this out, but after doing that, we created a readme in the documentation folder the details the full process.

Discuss any issues you had using AWS Cloud9 and GitHub and other tools

No issues. This was much smoother than we expected.

Describe the other tools/GEMs you used (such as CodeClimate or SimpleCov) and their benefits

FactoryBot - Create test objects in Ruby.
SimpleCov - For evaluating code coverage.

Make sure all code (including Cucumber and RSpec) is pushed to your public GitHub repo. Links to your Pivotal Tracker, GitHub repo and Heroku deployment. Make sure these are up-to-date

Pivotal Tracker - https://www.pivotaltracker.com/n/projects/2436883
Github - https://github.com/harishk1908/Project_Collection_and_Assignment.git
Heroku - https://cryptic-ocean-45494.herokuapp.com/