Final Report

Summary:

The project is a PhD Admission System where the admission committee can review all the applicant profiles and make admission decisions accordingly. The main customer need is a G-Suite Google App Script based system for evaluating all PhD/Master applicants. This project was created and completed in google sheet. More detailed requirements included: (1). Data in EngineeringCAS can be pulled into the system with improved automation. (2). Various filters are added so that users can select a certain range of applicants to review. (3). Existing applicants are updated automatically when new changes are made. (4). The faculty can be notified when the applicants are nominated and when new applicants are added.

After four iterations, we have implemented this project to the customer's expectation. The stakeholders are the admission committee and faculties. First, we have automated the process of downloading all pdf files and converting these files into text format so that they can be uploaded into the Google sheet. Second, we have increased the functionalities in the system by adding more filters such as Nationality, Applied Degree, Research Interests, Potential Faculty, BS University, MS University, GRE, TOEFL and Faculty Contact. Third, we have succeeded in updating the corresponding rows in google sheet without altering the reviews from faculties when new changes are made. Lastly, we have ensured that the faculty can be notified when the applicants are nominated.

Team roles:

Zirui Wang (Product Owner in iteration 1)
Yihao Xie (Scrum Master)
Jinjin Jiang (Product Owner in iteration 3)
Gayathri Krishna (Product Owner in iteration 2)
Ryuki Kobayashi (Product Owner in iteration 4)
Mohit Sharma (Product Owner in iteration 2)

User Stories:

Feature 1: Automatically download pdf files from EngineeringCAS

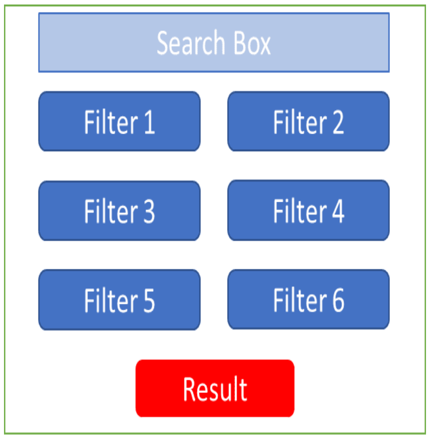
As an admission committee member
So that I can get all the applicant profiles without manual downloading
I want to have an automatic way of downloading all pdf files from the website
Assigned points: 2
Implementation status: not started
Note: Due to lack of authentication, we are not able to access the website.

Feature 2: Automatically add new applicants' information into google sheet

As an admission committee member
So that I can see new applicants' information in the google sheet without manual work
I want to have an automatic way of extracting useful information from files in the google drive and then add them as new rows into google sheet

Assigned points: 3
Implementation status: accepted

Feature 3: Search potential applicants according to one or more filtering criteria.
As an admission committee member
So that I can determine what applicants to review
I want to review a specific range of applicants on the basis of degree applied, nationality, potential faculty, research interests, GRE/TOEFL scores, etc.
Assigned points: 3
Implementation status: accepted



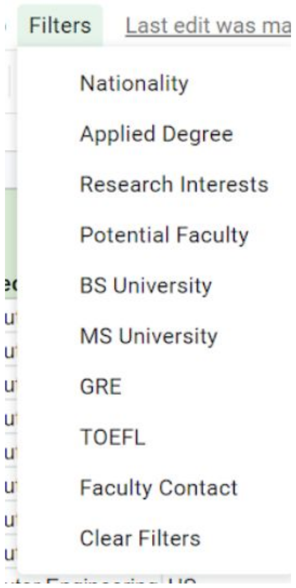


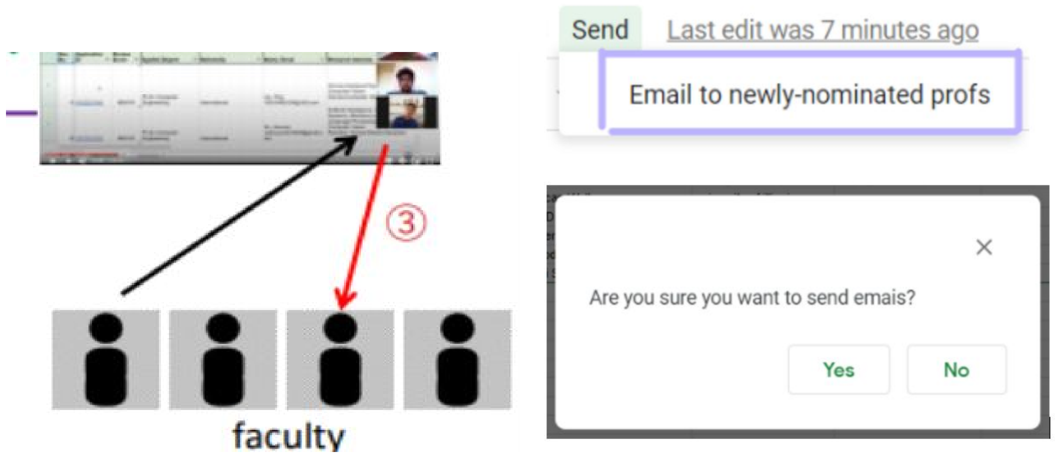
Figure 1: [feature 3]Left: lo-fi UI mockup   Right: corresponding final UI

Feature 4: Automatically Navigate the decision results back to Webadmit.org
As an admission committee member
So that I can make decisions in the system without the need of going back to EngineeringCAS
I want to make decisions in the system and the results can be navigated back automatically
Assigned points: 3
Implementation status: not started
Note: Due to lack of authentication, we are not able to access the website.

Feature 5: Automatically update existing applicants' information in google sheet
As an admission committee member
So that I can update existing applicants' data in google sheet without manual work
I want to have an automatic way of detecting those changes and update the corresponding rows in google sheet without changing other cells filled by the reviewers
Assigned points: 3
Implementation status: accepted

Feature 6: Automatically send email reminder to the corresponding faculty members

As an admission committee member
So that I can remind the faculty members of the nominated applicants
I want to have an automatic way of sending email reminders to the faculty members.
Assigned points: 3
Implementation status: accepted



Are you sure you want to send emais?

Figure 2: [feature 6] Left: lo-fi UI mockup Right: corresponding final UI

Feature 7: Keep track of the data on the Google Sheet
As an admin
So that I can get informed of any changes and prevent the data from getting modified by mistake
I want to keep track of every version of the system and have access to all versions.
Assigned points: 0
Implementation status: no action
Note: Such functionality already exists in Google Sheet.

Iteration summaries:

Iteration 1:
1. Accomplished the initial customer requirement about the automation part.
(1) Scanner completed. The scanner will first read the log file which contains all processed file names. Then scan the whole folder and mark those files whose names are not in the log file as new files. Then call the downloader.
(2) Downloader completed. The downloader will download all the new files into the user's local pc. Then call the local parser.
(3) Parser will be finished by the customer. Due to some privacy issues, we are not allowed to access those application files. After all useful information has been extracted. The parser will call the uploader and updater.
(4) Uploader will upload all extracted information into google drive for backup. Besides, it will update the log file.
(5) Updater will add all extracted information into google sheet.

(6) Notifier will send an email to notify the administrators about these updates.
2. Three points completed in this iteration.
3. New customer needs came after this iteration. The customer wanted to add a new feature: when some existing applicants' pdfs are replaced with new ones, the program should ask the user whether he/she wants to to update the corresponding rows, if yes, the corresponding rows should be updated without changing other cells, for example, reviews and scores given by the reviewers; if no, do not update the corresponding rows.

Iteration 2:
1. Accomplished the customer requirement about the Filters functionality in the Google sheet.
(1) Using the google apps script, a corresponding sidebar gets populated using the contents of the filter's index.html file.
(2) Index.html file contains the links to the skeleton (form.html) of that particular filter and also to the associated javascript (javascript.html) file containing the call to the "processForm" function back in the parent script.
(3) After the user makes the selection on filter values and clicks submit, the corresponding processForm function will get called.
(4) Using the parameter passed, we set the "column filter criteria" on the corresponding column and it filters out the sheet entries according to the query composed on that parameter.
(5) To further apply another filter on the filtered data, users can simply select other filters and the sheet will be filtered on top of the previous filter's results.
(6) To remove all the filters, users can simply select "clear filters" from the filter menu which will remove all the filters present and return the original sheet.
2. Three points completed in this iteration.
3. No new customer needs for this part. The filter functionality part is finished.

Iteration 3:
1. Accomplished the new requirement from the customer for the automation part.
(1) Scanner and log file structure updated. The scanner will first read the log file which contains all processed file names and their version numbers. Then scan the whole folder and mark those files whose names are not in the log file as new files, mark those files whose names are in the log file but with different version numbers as updated files. Then call the downloader to download all these files.
(2) Updater updated. For extracted information from new files, add new rows into google sheet. For extracted information from updated files, for each file, ask the user whether to continue the updating process, if yes, only update relevant fields for the corresponding row; if no, do not update this one.
(3) Notifier updated. For new files, send an email to notify the administrators that new applicants' have been added; for updated files, send an email to notify that existing applicants' info has been updated.
2. Three points completed in this iteration.
3. No new customer needs for this part. The automation part is finished.

Iteration 4:
1. Accomplished the customer requirement about the automatic email notification function in the Google sheet.
(1)Using the Google App Script, we added an extra column in R that records if the application is already notified to the potential faculty or not.
(2)By pressing Send->Email to newly-nominated profs, you are asked if you are sure you're sending emails. This is important because the user does not want to send emails to professors by mistake.
(3)If you are sure you're sending emails, then the program scans the column R to find out which application is not yet notified. If the cell is empty, it means the application is not yet notified.
(4)Then, the program creates an email recipients' list that contains applicants' IDs, potential faculties' names.
(5)The program sends emails to the addresses in the list. While sending, the cells are marked as "Sending...". This lets the user know the status. The user should not interrupt the operation while sending.
(6)After sending, the cells are marked as "Sent." This lets the user know that the emails are successfully sent.
2. Three points accomplished in this iteration.
3. This is the final iteration.

Customer Meetings:

- 09/26/2020, 11:00 am – 11:59 am, Zoom meeting
  Iteration 0: Discuss initial customer needs
- 10/02/2020, 5:15 pm – 6:00 pm, Zoom meeting
  Confirmed the user story for iteration 1: implement the automation process of uploading the pdf files data into the google sheet
- 10/13/2020, 8:15 pm – 9:00 pm, Zoom meeting
  Demo for iteration 1
- 10/19/2020, 8:00 pm – 9:00 pm, Zoom meeting
  Confirmed the user story for iteration 2: implement filters towards the admission system
- 10/26/2020, 8:00 pm – 8:30 pm, Zoom meeting
  Demo for iteration 2
  Also confirmed the user story for iteration 3: google sheet can automatically update the applicant profiles
- 11/9/2020, 7:00 pm – 7:30 pm, Zoom meeting
  Demo for iteration 3
  Also confirmed the user story for iteration 4: notify the professors when new applicants are nominated
- 11/21/2020, 7:00 pm – 8:00 pm, Zoom meeting
  Demo for iteration 4
  Also help the customer go through the whole project setup

BDD/TDD:

Our development is more BDD-wise. The product owner wrote each customer need in simple English, which later was converted into programming testing. The programmer implemented the functional code underlying the behavior. The behavior would be checked in each iteration to determine if there was any need to refactor or reorganize the code.
The most significant benefit of BDD for us is that it allows our developers to walk in the customer's shoes. It is easy for the developer, testers and business users to collaborate.

Configuration Management Approach:

We took advantage of the Github and Pivotal tracker as the configuration management approach to document the system from requirements and include design and code, as well as tracking every version we developed for the system. We didn't do any spike because the customer needs were straightforward and thus it was easy for us to estimate how much work would be required to solve or work around a software issue. We have 4 releases, each with only one branch.

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

No, we did not use Heroku.

Describe any issues you had using AWS Cloud9 and GitHub and other tools.

No issues

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

No we did not use any other tools/GEMs

Links:

Pivotal tracker: https://www.pivotaltracker.com/n/projects/2467770
GitHub repo: https://github.com/ziruiwang1997/PhD-Admission-System
Heroku: no deploy to heroku. Google sheet is the working environment.
Poster and demo video: https://www.youtube.com/watch?v=FBhVGpwHQ00&feature=youtu.be