CSCE 606 Software Engineering - Final Report "MailSend System"

Chun-Sheng Wu
832001192
CEEN, Texas A&M University
jinsonwu@tamu.edu

Yi-Chia Wu
132006360
CSCE, Texas A&M University
yichia@tamu.edu

Ya-Ru Yang
833003041
CSCE, Texas A&M University
yaruyang@tamu.edu

Yao-Wen Chang
833003095
CEEN, Texas A&M University
yaowen_chang@tamu.edu

Tung-Chi Yeh
531006877
CEEN, Texas A&M University
yehtungchi336@tamu.edu

Summary

Imaging that you are a professor handling tons of business simultaneously, collaboration between labs, conference attendance, research events, and etc., force you to make a quick decision. There must be considerable mails required to be sent on time. Definitely, you are eager to have someone give you a helping hand. MailSend system is accordingly designed to help the executive compose, arrange, and send the mails they need to deal with. The assistant, who assists the executive (professor) with chores, can somehow alleviate your stress through our system. All the executive has to do are just assigning tasks and reviewing the drafts before sending. In that case, the process of mailing becomes a time-efficient work.

The majority of our system is constituted of two roles, the executive and assistant. The executive firstly assigns the drafting tasks to whoever he/she wants. Subsequently, the assistant can start engaging in the draft before the indicated deadline. Review is taken place after the draft is completed. The executive has options of accept&send, reject, and delete functions on the side to operate. Due to the security problem of information leakage concern, Gmail restricts us to directly modulate the content of users' mail account. The executive should follow our instructions to link their mail account with the third-party application, EmailJS[1], in advance to fully activate the functions. Additionally, we designed a notification window on the homepage to remind tasks in the very first sight of users. We will attempt to keep enhancing user experience in the future. In conclusion, MailSend system enables the executive to manage mails with the least effort by distributing the tasks and expediting the process mostly done by ReactJS[2].

Link

Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2596847
Github Repo: https://github.com/yehtungchi336-tamu/MIRC-Code-Verification
Heroku: http://react-login-home.herokuapp.com

User Story

User stories describe the requirements asked from our client. Some of them was split into several pieces of work for staged progress. Therefore, there would be combining bulk stories exemplifying details of iterated status, followed by the revision and primary functionality by far.

Login via Google API and Self-Registration

Points given: 16

In aspect that the client requested to utilize Gmail API, the login method she desired us employing Google authentication login to fetch user information. We should separate the registration in different types of role we had in the website. The executive required additional information after sign-up or first-time login. It turned to the email account linkage page to provide necessary information to enable mailsending function of our application. We also continued enhancing the sign-up flow and fixing the bug we encountered in the following releases. Primary functions of login and sign-up have been appended on our application. We currently dedicate to improve the flow from registration to email linkage.

Fig. 1. stories of login via Google API and self-registration

Database

Points given: 8

Database was constructed for storing user data, drafts, and email linkage information. We decided to form our database by Firebase[3], which can directly use API provided by Google. Realtime database mainly handled the email linkage information and drafts. On the other hand, firestore database recorded the user data in two roles. Database cleanse and column revision was performed while our data grew bigger later on.

Fig. 2. stories of database

Platform Integration and Deployment

Points given: 16

We applied ReactJS, Django, and Firebase in our system. There existed essential platform integration between these three structures. Substantial efforts we put to successfully transfer platform and ensure the functionality worked. In addition, some points were conducted to have front-end and back-end deployment individually in the beginning stage because we tried to firstly divide our work into these two parts. However, as the project became bulky, features and debugging were focused instead of the facets of front-end and back-end. Platform is now steady and do not demand any further revision.

Fig. 3. stories of platform integration and deployment

Mail Sending

Points given: 13

As mentioned above, EmailJS was employed in our system to deal with the task of sending mails. Before activating the function, the executives, who are responsible for the maills, should register their accounts in EmailJS and provide the information, e.g., serviceID, templateID, and publicKey, to us. There are instructions giving examples about how to operate the linkage and obtain the numbers we demand. Currently, we are attempting to merge the executive's registration page with this email linkage setup. The executives that lack of email service will be redirected to the linkage instruction to avoid following mail sending errors. Some tiny optimizations, like jump to the new tab after clicking the button and CC & BCC functions, were conducted in the further testings.

Fig. 4. stories of mail sending

Interface Design

Points given: 30

Many efforts was devoted to the interface design since the MailSend system was purposed to fit with users in different types. The prototype of homepage was decided to be simple and clear after the

discussion with our client, so did the main visualization. Providing that there were two roles, we tried to classify them with colors, homepage, display on the top, and sidebar. So that, users would not be confused about how their status was after logging in. The icons, graphs, and arrangement experienced several revision as the time went by. Collapsed columns, popout alert, and notification window were included to optimize the user experience and for better look. Notification window is expected to have some improvements to indicate more details of draft.

Fig. 5. stories of interface design

Draft Workflow

Points given: 33

Draft was the main character of MailSend system. The executives assign the drafting task to an assistant with deadline involved. Drafts are completed following the commands by the date and sent back to the executive for review. The executives have options of accept&send, reject, and delete on the side to determine the fate of the draft. Drafts required database to store the data, e.g., recipient, subject, content, and etc. To fulfill this function, many stories were considered to smooth the process. Functions were split into pieces to better perform Agile-like development. We have met the consensus with our client that this section is almost done, only some appearance enhancement needed in the future.

Distribution & Accomplishment in Iterations

Roles, scrum master, project owner, change, and accomplishments achieved in each iteration will be demonstrated below. Our project owner is Tung-Chi Yeh, who possesses the Github Repository. Release and publish related job is handled by everyone in the team. The scrum master is rotated to equivalently distribute the work in iterations. Our Heroku deployment is under the management of Yao-Wen Chang.

Iteration0

Scrum Master: Tung-Chi Yeh

We were designated to create a code verification judging website in iteration0. It was said to educate the students learning programming at the first time. Correct composing logic was always pleased to develop a good coding habit and thinking. The code judge had a special feature could indicate

4

Fig. 6. stories of draft workflow

the place wherever might execute correct answer but have wrong composing logic. This was able to let the students know how the workable and clear codes look like, thus benefiting them to become an extraordinary programmer. However, the primary work was absolutely different to what we have completed, namely MailSend system. Therefore, many things mentioned in this iteration would be in the discussion stage, not be implemented.
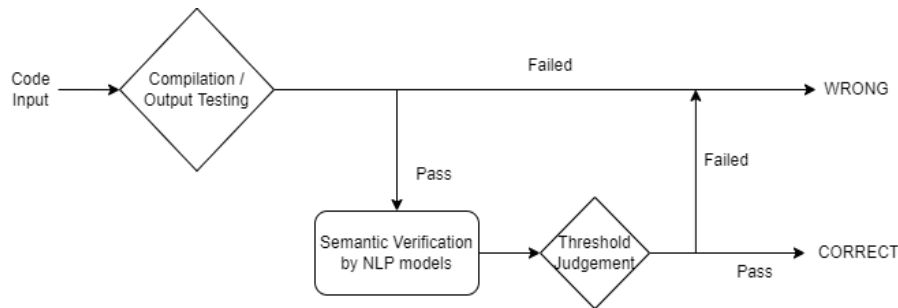

Fig. 7. brief workflow of MIRC code verification judging system

The brief workflow was shown above. The natural language processing (NLP)-based engine was the main judge utilizing seq2seq translation inside the system[4]. NLP was capable of generating segments of word after sufficient training. A threshold was settled as the standard to distinguish whether the input had identical logic with producing codes. The NLP model could tell whether the input codes were semantically qualified with composing logic or not. Input format was limited to a fixed style before transmitted into the engine to regulate inaccessible predicting samples. The result of verification was displayed on the top left corner and errors were placed between lines. Homepage, question database, answering page, and sidebar design were yet deeply discussed in this stage. Thus, they were still serve as prototype in this iteration.

Distribution
Chun-Sheng Wu: NLP model design and back-end prototype construction
Ya-Ru Yang: NLP model design and back-end prototype construction
Yi-Chia Wu: main visualization design

Yao-Wen Chang: front-end construction
Tung-Chi Yeh: database establishment

Iteration1

Scrum Master: Yi-Chia Wu

Starting from this iteration, we were reassigned to another project named as MailSend system along with a start project to build some fundamental application. The major task in this iteration was to create a third-party authentication login method with self-registration in our own database. As for the starter project, it asked us to initiate a web application built with Django. Besides, the landing page was finished with the functions of login and sign-up working perfectly. In case that the project was just turned over at that moment, we spent plenty of time discussing with client to thoroughly her requirements. The missions we accomplished in this iteration were a little bit fewer than other iterations. Fig. 8 is the UML diagram exhibiting the progress status in iteration1.
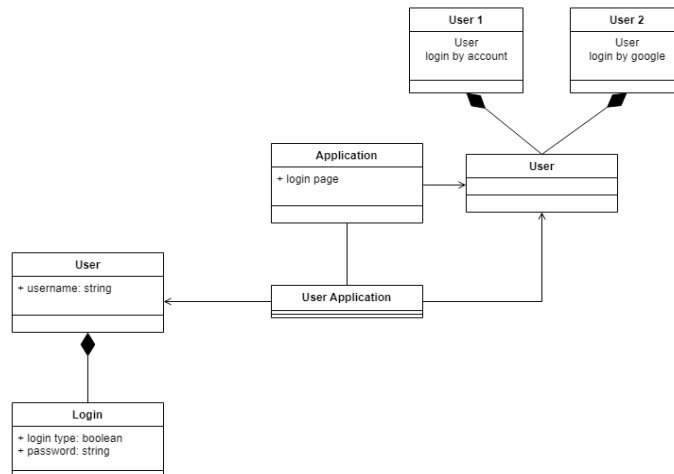

Fig. 8. UML diagram of iteration1

Distribution
Chun-Sheng Wu: Django (back-end) deployment
Ya-Ru Yang: front-end deployment
Yi-Chia Wu: picture and icon design in landing page
Yao-Wen Chang: login via third-party authentication, e.g. Google
Tung-Chi Yeh: database establishment

Iteration2

Scrum Master: Ya-Ru Yang

Mail sending function and dashboard design were what we primarily did in this iteration. Mails were sent through another third-party application, EmailJS, due to the security issue of Google account. Our client and we came up with this strategy to fetch account information and send emails to recipients. The executives were required to register in EmailJS service and provide several numbers, serviceID, templateID, and publicKey, to us. By doing so, our system had the capability to send mails on behalf of the executives. The mail sending function can now dynamically employ the EmailJS service depending on the executives' names. That is, the mails will be sent on the same name of the executives logging in our system. Figures below indicates the design of the landing page and dashboard. Changes in appearance were made in subsequent iterations. UML diagram of iteration2 manifests the completed work briefly.

Distribution
Chun-Sheng Wu: mail sending function and linkage
Ya-Ru Yang: draft workflow (in progress)
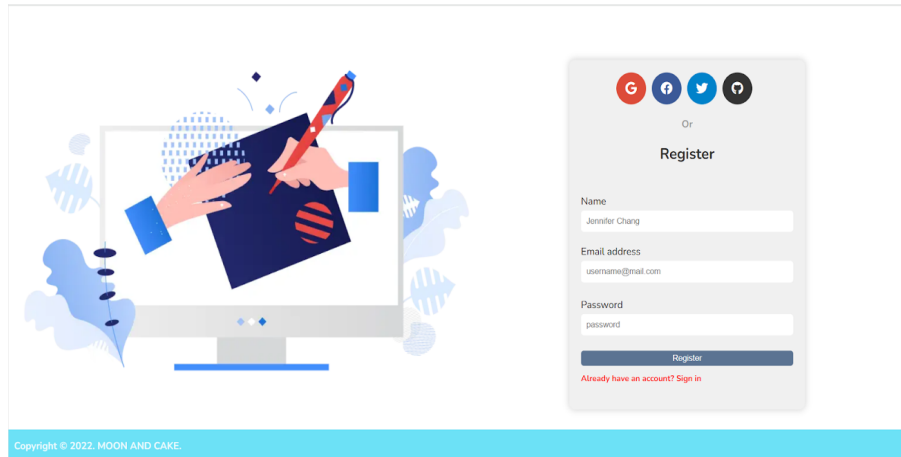*Yi-Chia Wu: landing page and dashboard design*
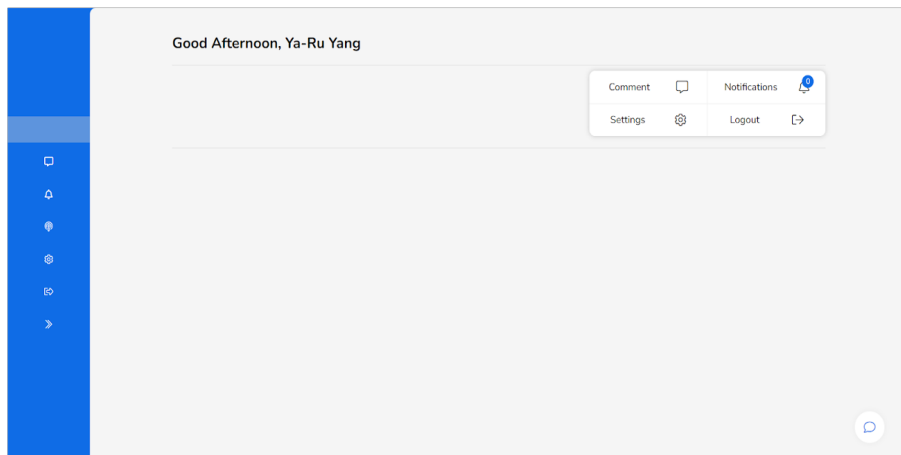
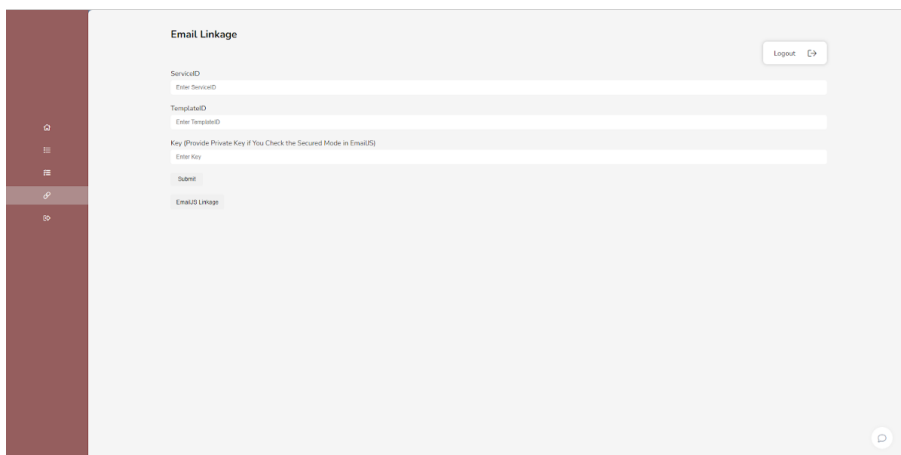Fig. 9. landing page

Fig. 10. dashboard
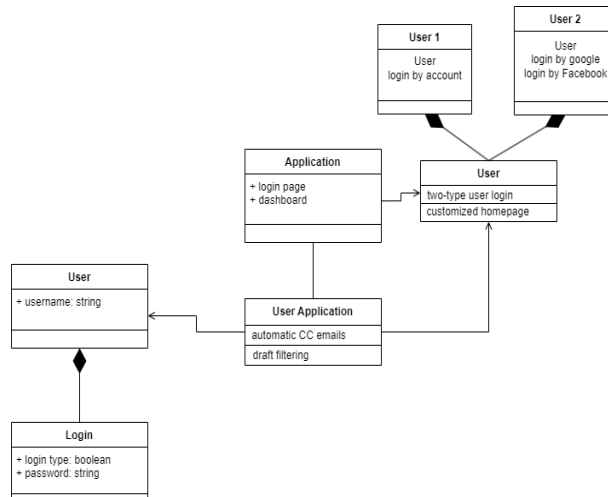

Fig. 11. page of email linkage

Fig. 12. UML diagram of iteration2

Iteration3

Scrum Master: Yao-Wen Chang

Draft function was incorporated into MailSend system in this iteration. Users could write a draft in the block and send email directly in our website. Subject, recipient, CC&BCC, and message listed by the user were transmitted to the expected recipients. The figure underneath showed how the draft page looked like. Draft workflow had additional optimization in the next two iterations since it did still not fulfill the desired communication between the executives and assistants. Furthermore, the discrepancies of two roles were created. The executives and assistants receive different theme colors, display words on the top, and sidebar. Users could clearly know which role they were in the first sight landing in the website. Instructions of email linkage joined the system for the better understanding of linkage process. We also attempted to concatenate the front-end designed by ReactJS with Django to reach the expectation of our client. Nonetheless, we encountered some abstruse problems unable to resolve in a short time. Plus, the transferring task was a static transferring, which needed continuous maintenance. Hence, the transferring job was postponed to a later stage until most of the requirements were achieved.
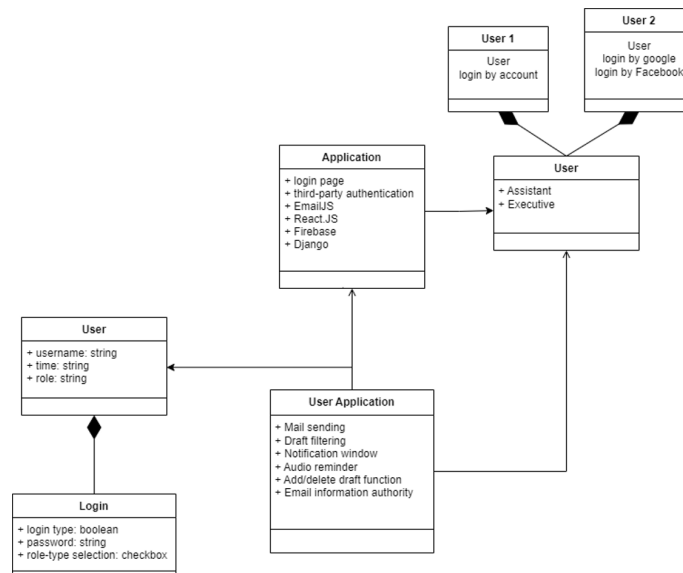

Fig. 13. draft composing page

Fig. 14. UML diagram of iteration3

Distribution
Chun-Sheng Wu: draft form and sidebar simplification (in progress)
Ya-Ru Yang: draft workflow and database reconstruction
Yi-Chia Wu: dashboard design and icons of sidebar
Yao-Wen Chang: UI difference by user role
Tung-Chi Yeh: audio file to transmit content (in progress)

Iteration4

Scrum Master: Chun-Sheng Wu

We made a gigantic leap in this iteration. Draft was optimized to nearly perfectness. The executives could assign the drafting task to the desired helper alongside audio files containing the details in Fig.15. The audio file was then eliminated since the client told us the executives and assistants were tend to use external communication applications to discuss the content. The draft would be sent back to wait review once the assistant finalized it in Fig.16. The executives had the rights to accept&send the draft or decline it back to the writer. Or, Fig.17 stated that the executives were enabled to modify the draft by themselves and sent it when they were satisfied. We designed a draftlist shown in Fig.18 for either the executives or assistants to quickly understand the status of drafts. Appearance optimizations were performed afterwards. The colors representing roles were changed to more distinguishable ones. The main picture of landing page was switched to a more suitable one. Sidebar was simplified for better recognition and accessibility. Some debugging and maintenance work began from this moment because the major requirements were accomplished.
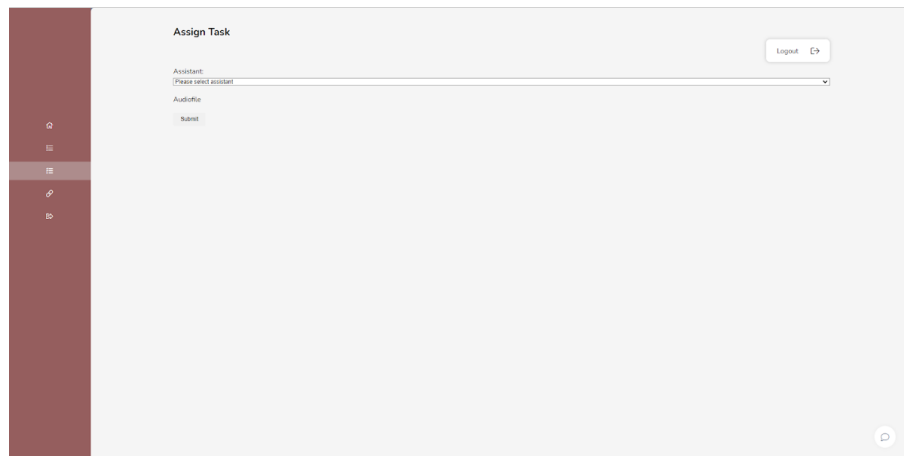
Distribution
Chun-Sheng Wu: redirection between pages and debugging
Ya-Ru Yang: majority of draft workflow and draftlist
Yi-Chia Wu: picture of landing page and arrangement of dashboard
Yao-Wen Chang: login limit and release
Tung-Chi Yeh: audio file to transmit content (paused)
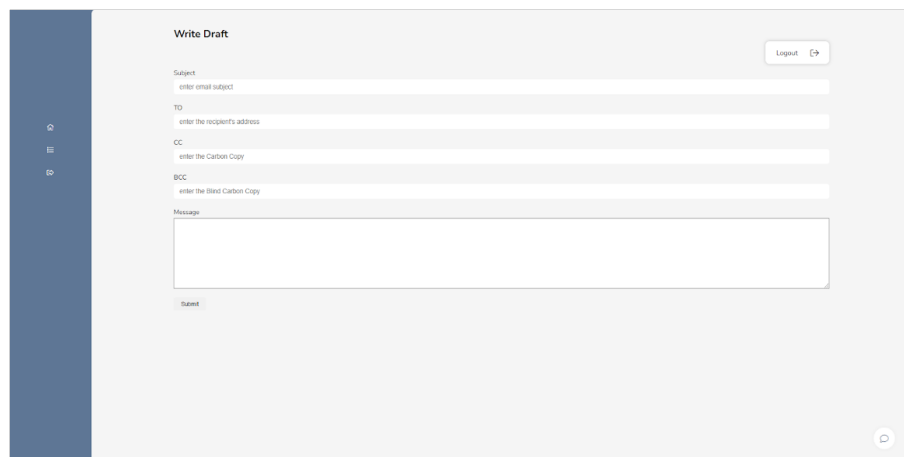
Fig. 15. page of assigning task
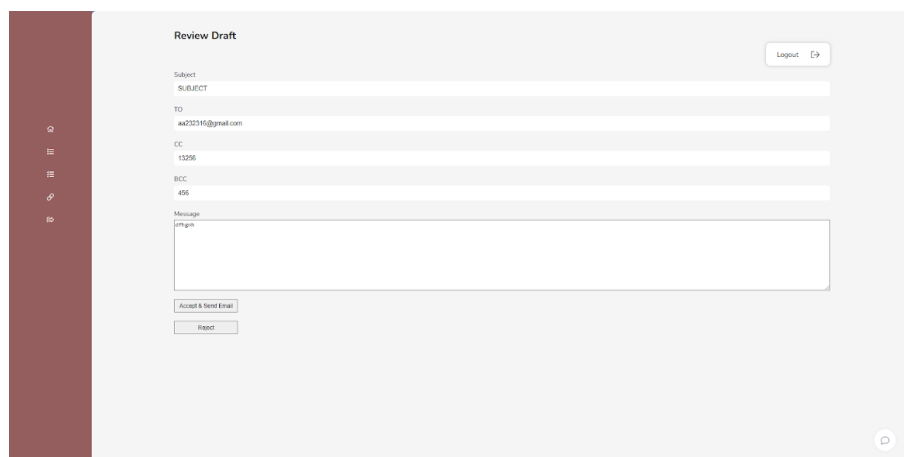
Fig. 16. page of writing draft on the side of assistant

Fig. 17. page of reviewing draft on the side of executive
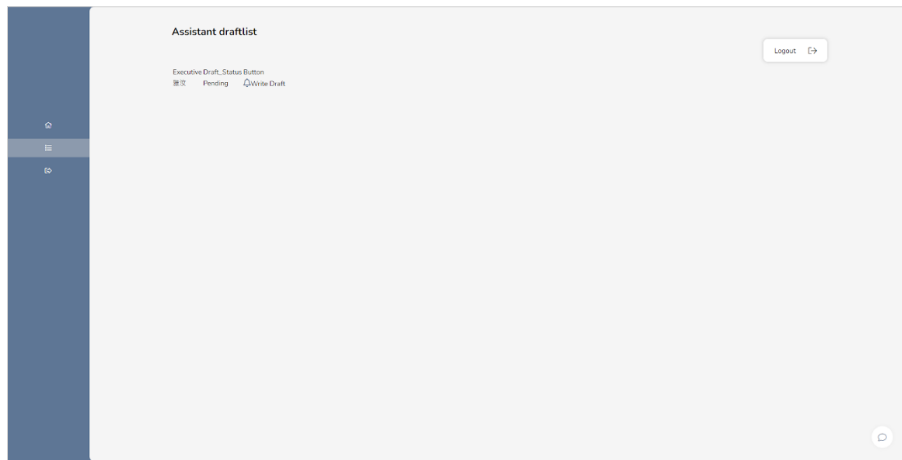
Fig. 18. draftlist
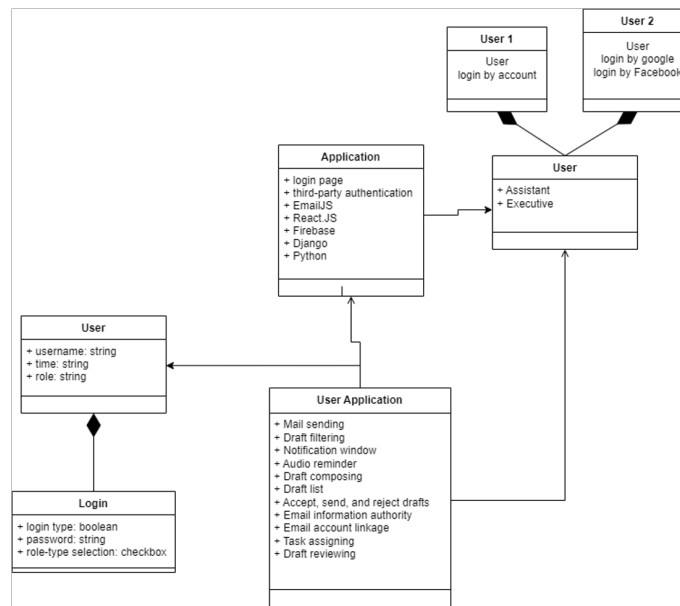
Fig. 19. UML diagram of iteration4

11

Iteration5

Scrum Master: Tung-Chi Yeh

We concentrated on optimizations in this iteration. Delete draft function was added to remove redundant tasks. Deadline was put on the assigning task page to notify when the assistants should submit their drafts in Fig.20. The executives could also see it to send mails on time. Draftlist was categorized to finished and unfinished in both ends of executive and assistant in Fig.21. There was an collapse button to fold the table as well. Redirection was enhanced for better user experience. Fig.22 pointed that more thorough instruction of linkage was involved. The css settings were updated to erase strange blocks on the website. A notification window indicating the status of draft was placed on the homepage exhibited in Fig.23. Some profiles were set to be fixed or static used for verification, they were reset to be dynamically callable. The system was processed with database cleanse while most of the columns were configured. We further implemented more BDD/TDD process to figure out potential bugs and inconvenient features.

Distribution
Chun-Sheng Wu: draftlist appearance optimization, debugging, and feature maintenance
Ya-Ru Yang: draftlist filter and delete draft function
Yi-Chia Wu: css setting maintenance and arrangement
Yao-Wen Chang: notification window and merge email linkage into registration (in progress)
Tung-Chi Yeh: draft filter and collapsed table

Fig. 20. revised draftlist

Fig. 21. revised draftlist

Fig. 22. revised mail linkage page

Fig. 23. notification window in homepage
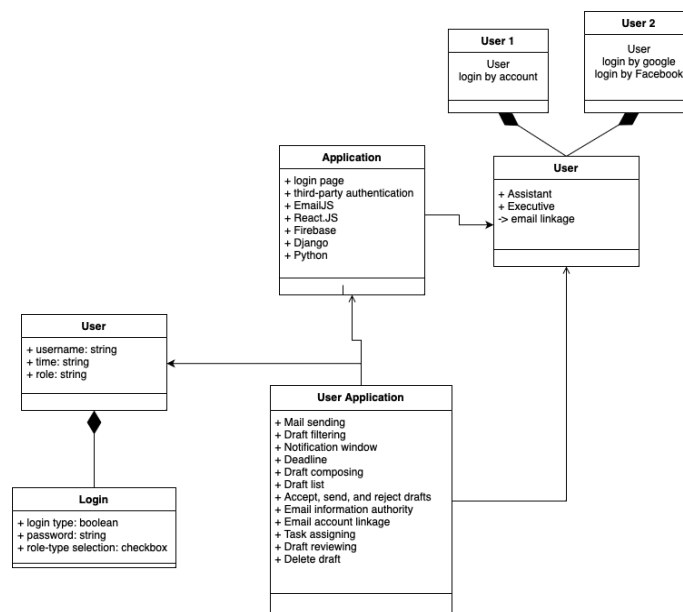
Fig. 24. UML diagram of iteration5

Meeting with Client

The meetings are ordered by the date it happened. Time, location, and its iteration come with the details of the meeting.

1. 9/21 at WEB 324 from 1:00-1:30pm - iteration0
Discussed the concept of MIRC code verification judging system. Our client gave a situation how this logic-semantic verification system could benefit freshman-level programmer with good coding habits. We decided to utilize NLP-based engine to judge the code with a threshold classifying between lines. There were two types of role, instructor and student respectively. He also desired a database storing quizzes and user history queue to record the answers of students. Customized homepage was expected to show pending quizzes and the results of verification.

2. 9/29 via Zoom from 10:00-11:00am - iteration1
The client asked us to firstly construct a basic website enabling third-party authentication login. Homepage and landing page were totally free to design. Should have admins that could check appropriate questions for visitors to test themselves. Required a database to collect user history and results of verification.

*Our project was replaced with MailSend system between these two meetings!

3. 10/14 via Zoom from 9:00-10:00am - iteration2
Primary tasks were changed since project was substituted. She wanted us to establish a dashboard at first containing the email sending function. Database was switched to store email details, not results of verification.

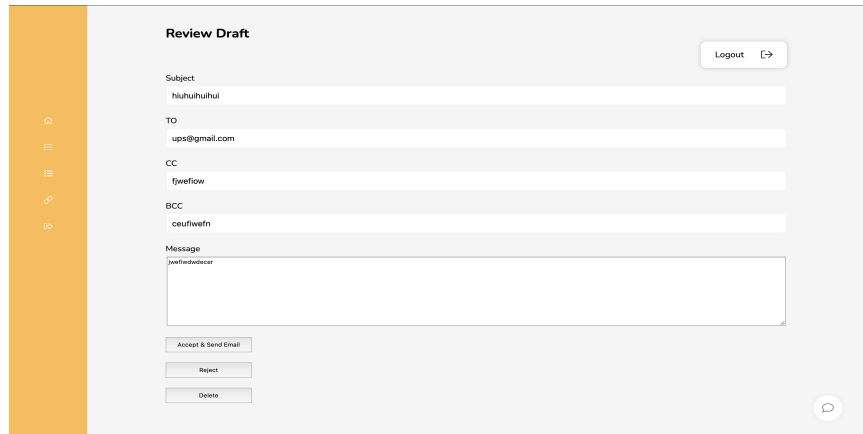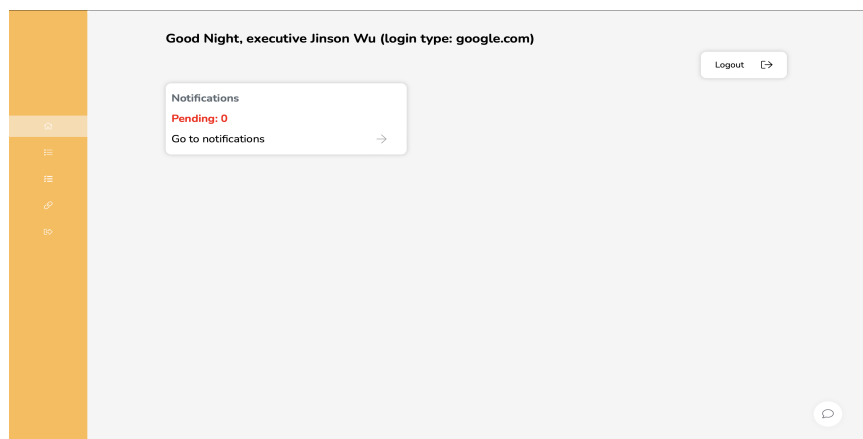*4. 10/21 via Zoom from 9:00-10:00am - iteration2*
Testing of homepage design, login/sign-up page, and two-type login worked well. Our client was satisfied with the dashboard along with draft form. The mail sending function was examined and it went well.

5. 11/04 via Zoom from 9:00-10:00am - iteration3
Researched about how to separate executives' and assistants' interfaces. Our client wished to have audio files that could record details from executives while assigning a task. All the functions of mail sending was tested and dedicated to the draft workflow.

6. 11/11 via Zoom from 9:00-10:00am - iteration3
Classified user roles with theme color, display words on the top, and sidebar design. Presented the draft prototype with draft assigning, composing, and review. Further discussed about what should be included in the draft workflow. Interface related design was checked and continued to optimize in the following.

7. 11/18 via Zoom from 9:00-10:00am - iteration3
Exhibited the whole draft workflow, which met the expectation of our client. She also suggested to add the feature of deleting draft if the mail was cancelled. The separation of user roles was completed with enough differences. Achieved registration limitation, EmailJS linkage, and main visualization enhancement. Debugging was initiated to improve user experience.

8. 12/9 via Zoom from 9:00-10:00am - iteration3
Final meeting in this semester. Gave a thorough demo from head to toe. Included new features of deleting draft, deadline, draftlist filter and collapsed table. Redirection and jumping problems were fixed. Alerts when submitting were all introduced in the system. Bugs and optimization were dealt with to smooth the operation. Documentation was written if there was a team continuing this project.

BDD/TDD Process

BDD: Test an application's behavior from the end user

- There will be two ways for users to log in, one is logged by a google account and the other is by an email-signup account.
- User can choose his login role type on the checkbox
- Registration and login is prohibited if the user does not select the role

- EmailJS linkage with executives' accounts
- Check out the notification window and jump to drafts
- Send emails with executive's account
- Different selected roles will lead to different homepage after login
- Function varies depending on the roles
- Task assigning by the executive, including deadline
- Draftlist checkout and status indication
- Drop-down table of the list
- Filter drafts with status (pending, completed, and finished)
- Draft composing by the assistant and submission to the executive
- Executives review and modify drafts (accept & decline)
- Send the email once the draft is perfect
- Log out and redirect to login page
- Randomly browse the pages
- Collapse the draftlist after reviewing all the pending tasks
- Durability test

By involving all the potential behaviors operated by users, we can clearly know where the bugs are and work on optimizations leveraging the user experience. Many added features were come up with BDD process to make our application more general to users.

TDD: Test smaller pieces of functionality

- Type 1: enter username, password and select login role on the checkbox
- Type 1: front-end passes the information and interacts with database via back-end service
- Type 1: store user information in database
- Type 2: able to access google account (into account selection page)
- Type 2: use google account to login and fetch the basic information from the platform
- Different role types and username displays different user type
- Login information is stored alongside role type
- Sidebar hiding based on roles
- Email linkage by the provided information
- Send emails with the keys given by the users
- Database stores drafts, email linkage information, and draftlist
- deadline setting
- Communication between roles through table
- Draft status on the notification window
- Jump to drafts by clicking the href on the window
- Draft rejection and acceptance
- Task assigning with details including
- Scroll-down function for the table
- Draftlist status indication
- Mail sending test (whether the mails correctly deliver/receive)

TDD process is mainly responsible for debugging and maintenance. The functionality of application is split into pieces to ensure they all work well individually. Errors will be marked and uploaded to pivotal tracker to have a hot fix.

15

Configuration Management Approach

Everyone in the team shares the authority to modify codes on Github repository. We have Develop branch to keep up with the latest version. About seventy commits had been made to branch Develop so far. Everything checked to be correct will be published to Develop afterwards. Besides, there are branches named by members' name to design new functions locally. Once a feature was completed, we all would switch to that branch and gave advice. Content in branch Develop was merged to branch main every time about to submit an iteration report. For Heroku deployment, one of us managing the deployed stuff in Heroku. The newest revision is uploaded to Heroku in a while. We already published approximately thirty releases on Heroku.

Issues

Issues encountering in development, release, and deployment will be discussed in this section. Some of them have been resolved, but the rest of them still not will be mentioned in Future Work, accompanied with possible solutions.

Development

The very first obstacle we faced was the website establishment. We decided to exploit ReactJS as the backbone to implement this application. ReactJS coordinated with css could bring about convenient working environment when devising a website. The next perplexing problem was the security concern of Gmail API. We could not directly modulate the information fetched from users' Google account due to profile leakage doubt. Linked with EmailJS was the method to solve this problem. This third-party application could send mails for us although there were additional steps required. In order to connect the service smoothly, we put efforts here to promulgate a tutorial instructing the executives how to link.

Bugs emerged crazily after features published on the website. The most severe one we could not remove yet was the draftlist show-up problem. It did not display the content every first time jumping to the page of draftlist. we have checked out the data passed across, nothing wrong was realized though. It became normal when redirecting back from other pages of the website. Investigation would be kept until the resolution was found. Except that one, some accidental jumps happened sometimes. These needed time to locate where the problems at and how to address. Infrequently the website dead in first-time requesting. This might be caused by the internal problem from Heroku, which could not be resolved easily.

Production Release

The release situation did not go well in the beginning actually. We all were unfamiliar with GitHub when the project was just on the outset. Versions were crossing around and some improper commits overwrote the work had been done. Things went back on the right track when we tried to sync everyone's progress and set a rule of pushing commits after a thorough cleanse. More branches were later created to leave own space to forge and test new functions.

Deployment

There was no big deal when we deployed the application to Heroku. Only the website would seldom shut down or dead after requesting. Redeployment was beneficial to conquer this situation since the reason might be the internal publishing errors in Heroku itself.

Description of the Repository

```
│  documentation
│  └── Fall2022
│      └── *.tar
├── iteration
│  └── *.doc
├── venv -> virtual environment
├── backend -> Django structure
│  ├── backend -> Django back-end
│  └── frontend -> ReactJS front-end
├── react-login
│  ├── node_modules
│  ├── public
│  ├── package.json
│  ├── package-lock.json
│  └── src
│      ├── index.js -> head file
│      ├── App.js -> render file
│      ├── Fire.js -> database connection
│      ├── style.css -> appearance design
│      └── components
│          └── *.js -> major functions
```

Directory tree above indicating to how our repository looks like. Except the directories containing documents, most of the files are javascript files. **/venv** is the virtual environment used for Django. The strike-out directories handle the Django structure, but we currently remove them. They will be put back since all the designs by ReactJS are completed. Json files store the package and required environment to run npm start. **index.js** is the head file initiating the project. **App.js** renders the configuration. **Fire.js** contains the connection information with Firebase. All appearance related settings are written in **style.css**. Most functions are located in components callable with **index.js** & **App.js**.

Future Work

Most of the requirements from our client were met so far. What we have to do is to generalize our applications. This application is expected to cater to users from different cases having draft composing needs. Draftlist requires further optimizations to enhance the user experience. Appearance, filter, and status indication are directions to bolster. Moreover, more detailed information of draft or functions can be included in the notification window on the homepage. The draftlist show-up bug can probably be eliminated by rewriting a wrap-up subroutine to return effective values. Decorations on the homepage beautifying the interface are also likely to color our application. In conclusion, the essential features are almost carried out in our work, it still thirst for endeavors to make the application better.

Reference

[1] URL: https://www.emailjs.com.

[2] URL: https://reactjs.org/.

[3] URL: https://firebase.google.com.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.