Final Report - CSCE 606 Software Engineering

1. Our group was tasked with continuing the implementation of a Teacher Assistant (TA) Management application for the Engineering Department at Texas A&M. Currently the Department uses a single Google sheets spreadsheet to manage the hiring and assignment process which has been noted as time-consuming. The TA management application seeks to streamline the process by providing a hub for all users (i.e. TA's, professors, prospective students, coordinators) to receive information.

Our primary stakeholders are professors of the Engineering Department at Texas A&M, current & prospective student TA's and coordinators. We sought to make the hiring and assignment process for TA's to be simple and functional by implementing a minimalist sign up and applying user interface for students. Professors and coordinators were also provided with a minimalist user interface allowing for easy checking of assigned TA's.

2.

Iteration 0:

Super Admin needs to upload the Master Schedule to the System
Professor can view details of Teacher Assistant
Professors can press button to email their TA's
Option for professors to Rate/Comment on TA's

Iteration 1:

Super Admin needs to upload the Master Schedule to the System: 3 points (completed)

Iteration 2:

View schedule and Upload schedule UI: 4 points (completed)

- The user interface for the view schedule and upload schedule was designed for better interaction with the admin. The admin is able to easily upload the schedule and also remove the schedule because of the obvious colors of the buttons used to prompt those actions.

User authentication: 1 point (completed)

- We implemented a user authentication system to enable users to signup, login, and be directed to the right view. With this authentication system, we added sessions that keep track of the access_level of the users thereby restricting their access to the right pages. The admin has access to all of the pages on the system, the professor can only view their profile and TAs assigned to them. The coordinator can create subjects and assign subjects to TAs. While the TA has access to their profile page, TA application form page, and application status.

- Iteration 3 > Opeyemi
  - We completed 3 user stories by the end of this iteration. The professors can now view the information of their TAs. Also, professors can now easily contact their TAs by clicking on a button to send an email. Super admin can delete past schedules to allow for clean upload of current schedules.

Final Report - CSCE 606 Software Engineering

Iteration 3:

Professors view: 3 points (completed)

- We fixed the view for the professor to see the TAs assigned to them, and be able to send feedback based on their performances.

Professor emails TAs: 2 points (completed)

- For convenience, we added a feature for the professor to be able to send an email to the TAs assigned to them. The button directly opens the professor's mail application and prepends their emails to the message.

Delete upload schedules : 2 points (completed)

- We added a feature for the admin to remove the schedules before uploading a new one at the beginning of each semester.

Iteration 4:

Fix signup functionality: 2 points (completed)

- This fix is really crucial for our team to make sure that teacher assistants can login after they sign up with our platforms. With this iteration, students can access the platform and see the views that are related to their tasks. Before this iteration, we made some changes to the app, and it caused login issues for newly signed up teaching assistants, which created problems in their onboarding process. Teaching assistants now can access all resources which can help to increase productivity. We gave 2 points for this user story because this requires us to create a new function with the TA access level in the app, and give some test cases for the method.

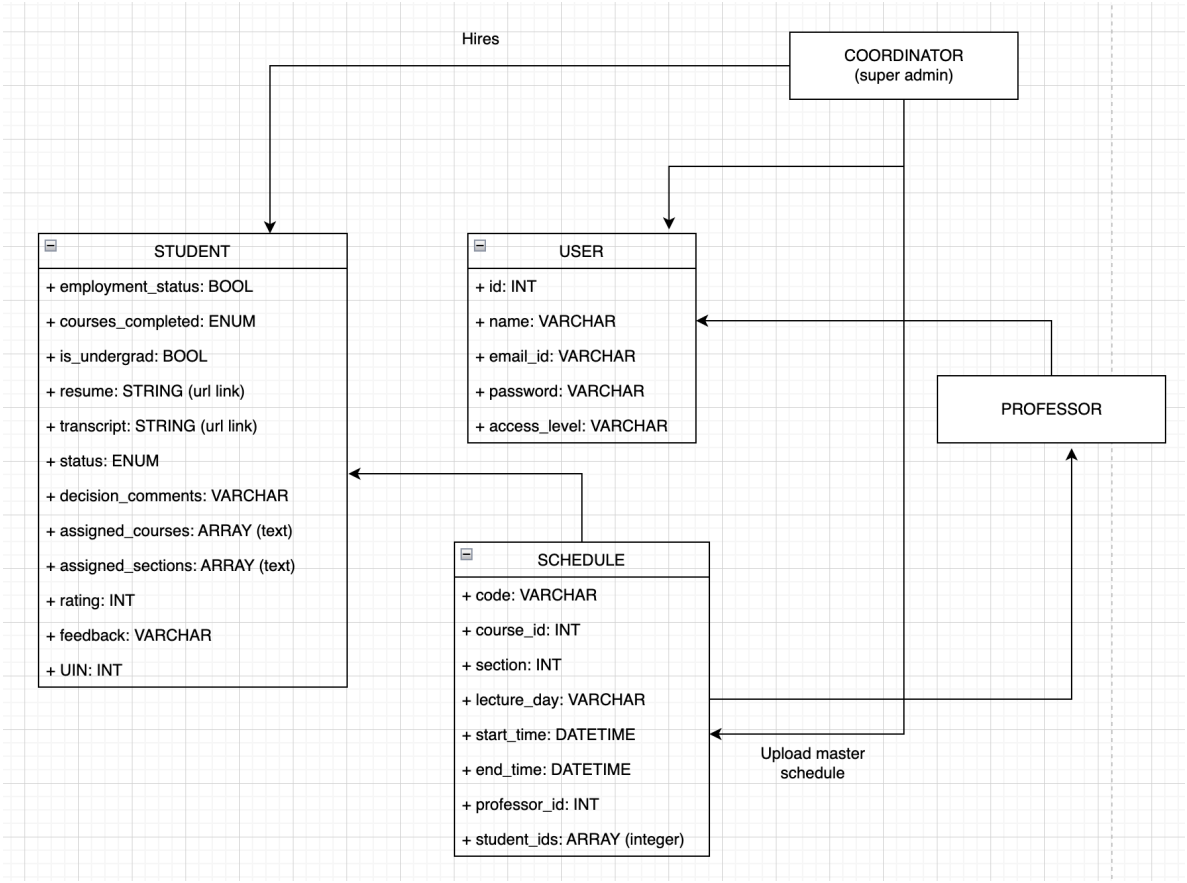
Setting up RSpec for schedule page: 1 point (completed)

- Our team created a test case to cover the schedule page functionality. We created mock data to test the page's ability to show schedules and handle data. Once the mock data is generated, we have a test case which verifies that the index page returns a status code of 200. This is written to ensure that the page exists and can be accessed without any issues. Next, we tested if the schedule can be reached. This test can verify that the page can display the correct schedule data and confirm the view page (schedule page) is linked correctly to the index page. Lastly, we tested if the schedule can be removed. This test can ensure that the destroyed functionality in rails is functioning properly and correctly. The test shows that we can remove all schedules without any errors. This setup can help us to develop the schedule page by identifying issues early in the development stage. We gave 1 point to this user story.

Setting up Cucumber: 1 point (completed)

- Our team created test scenarios that cover various aspects of the schedule page functionality. We tested some functionalities that related to the schedule; such as upload, view and remove. We tested by trying to upload an unsupported file format and trying to remove schedules that have already been uploaded. Secondly, we tested if the application submission is working properly. We

tested when trying to submit an application with incomplete data, duplicate submissions and invalid inputs. Lastly, we tested the signup functionality. This involves creating new accounts, verifying that passwords are matched between password and confirm password, and other invalid inputs validation. We gave 1 point for this user story.



We changed a little bit for our design diagram this iteration which can reduce the redundancy. We take out the role of hiring manager in our diagram because we notice that the super admin can actually do hiring manager's work. Then we change the relationship between schedules and students and between schedules and professors. For students, we use student's email to identify each student. Once a student signs up for an account, he will see the student portal and have a button for him to apply for a job. Once the super admin assigns this student to a course's TA, then the student will see his schedule when he logs in again. And for the professor, he can login and see his schedules based on his professor ID that the super admin assigns to him. He will also see his TAs' emails in all his classes in the schedule.

Iteration 5

Fix the view for TA status screen: 2 points (completed)

- The default status for TA is not applied. After a TA applies for a position, his/her status will become applied. If a TA already applied for a position, he/she will see a message saying that "You will

be notified once you are accepted". If rejected, TA will see the message "Sorry, you are rejected". If hired, TA will be able to see the status and the schedules that are assigned to him/her. We also make sure that TAs can not apply again if their applications already exist.

Add professor email attribute as foreign key of the schedule: 2 points (completed)

- Before this, the professor cannot see all schedules that are related to him/her. It is because the foreign key is missing in the schedule table. Professor can only query one schedule. With this iteration, we added professor email as foreign key because it is unique to each one of the professors. With this key, we can just query all schedules that have that professor email.

Fix the schedule upload to import with the instructor email: 2 points (completed)

- We made some changes to the csv file. For this program to work perfectly, the administrator must also add the professor email into the csv file. It is because we need professor email to be a foreign key of the schedule. Now, in the schedule table view page, it has another column called instructor email.

Fix the view of user portal : 1 points (completed)

- Once the TA login, he/she will see the TA Info on the top right corner. Before that the user can only see "User Info" on the top right corner, we want to change the view of it to make it clear based on our client's requirements

Coordinators can successfully login and re-assign TA for each course: 2 points (completed)

- Before this, once the admin assigned a coordinator account, the coordinators cannot successfully login. Now, the coordinators can login and redirect to the subject's view where they can do their job.

Fix access level : 2 points (completed)

- Before this, the access level for different users was messed up. We changed the access level to make sure that some pages are only accessible for admin and coordinators. And these pages won't be accessible for TA.

Fix the schedule upload: 2 points (completed)

- When the admin uploads the schedule, if the TA email is not registered or exists in the database, the program will still upload the schedule but with "TA is not registered" in the column PT1 and PT2. Admin can reupload the schedule when the TA is registered. Once the admin uploads the schedule again, the old one will be deleted.

3. 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.

- Each team member reviewed the documentation section, paying special attention to the final report and final powerpoint for understanding. Any outdated gems were updated and corresponding code updated to reflect current version. Database changes were considered, but ultimately declined as any changes would fundamentally change the application. Team members worked with client, Dr.

Ritchey, to come to an understanding of the present state of the application and what the client would like to see implemented.

4. List who held each team role, e.g. Scrum Master, Product Owner. Describe any changes in roles during the project.

- Scrum Master: Alexander Muyshondt
- Product Owner: Opeyemi Alabi
- Members: Kimleng Hor, Boxuan Hu

5. For each scrum iteration, summarize what was accomplished and points completed.

- Iteration 0
  - Met with Dr. Ritchey and derived 4 user stories. Will choose one user story to focus on for the next iteration. Created Lo-Fi user mockups for stories.
- Iteration 1
  - Selected to implement upload of master schedule by admin user for 3 points.
- Iteration 2
  - Completed the UI design for the upload page and also updated the login functionality.

- Iteration 3
  - Revised the professor's view to show the information of their TAs, and also added a send email feature.
- Iteration 4
  - We completed our 3 user stories which are fixing sign up functionality, writing RSpec tests and writing cucumber tests. In this iteration, we earned 4 points.
- Iteration 5
  - We completed 7 user stories which are fix the view for TA status, add professor email attribute as foreign key of the schedule, fix the schedule upload to import with the instructor email, fix the view of user portal, coordinators can successfully login and re-assign TA for each course, fix access level, and fix the schedule upload. And we also do cucumber tests and RSpec tests for all the functions. In total we have 13 points.
  -

6. List of customer meeting dates, and description of what happened at the meetings, e.g. what software/stories did you demo.

- 2/3/2023: Discussed with client current implementation of app and came up with four possible user stories.
- 2/24/2023: Demoed master upload .csv function.
- 3/10/2023: Demoed increased security measures for app, users cannot edit other user profiles or access admin page. Admin can edit user information. Update UI of .csv upload and schedule page.
- 3/31/2023: Demoed Professor can now view TA information when logged in. Professors can press a button to email their TA's. Super Admin can delete past schedules.
- 4/14/2023: Demoed improved sign up and TA application process. Discussed implementation of testing measures.

- 4/28/2023: Demoed improved TA application status. Added professor email to master .csv upload. Simplified user portal view. Coordinators can re-assign TA's.

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

- BDD
- Identify the user stories and requirements for the schedule page.
- Write some basic cucumber tests to check if the page is available and reachable.
- Before we come up with these tests, we also discuss what the requirements are with our stakeholders so that we can ensure that everyone is on the same page.
- We then write the test code based on the requirements and everytime we finish the functionality, we run the test case to see if it is working properly.
- Then, we keep adding more tests and keep repeating the process for each test case.

What we found during this development is that BDD is really time consuming. It slows down our development process but it can ensure the quality of the code. We could not really finish lots of tasks because we are not really familiar with the BDD. But, we learn during the development process and put it into practice. The benefits of BDD is that it promotes better collaboration between developers and stakeholders by ensuring that everyone is on the same page. It is really good to work in a team because we know exactly what breaks the code when we add new functionalities.

- TDD
- Write the test that defines basic features of the page to make sure that the RSpec is working with the page.
- With every functionality, the test case is written first to ensure that the actual code is working properly.
- Run the test code every time the functionality is completed.
- Keep repeating these steps whenever we add a new functionality.

For TDD, it is difficult to implement and fix when we make changes to the code. To implement it, we need to install a gem file and learn at the same time during the development phase because it is new to us. Because of this, it slows down our development. Sometimes, our method is working fine but our test case is wrong, so we need to fix the test case. When we make changes to a user session, we need to make sure our RSpec is working well with that which is a headache. However, TDD ensures the quality of our code in the long run and actually reduces the amount of time spent on debugging if we implement it well.

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

- In our application, we have two different config files. One is for development and the other is for production. Each member of the team is using development either in local or AWS Cloud 9. Our production config is only used when we push it to Heroku.

Final Report - CSCE 606 Software Engineering

- To collaborate with each other in the team, we are using GitHub. It allows us to manage versions of the code, and create and track changes to code. It is also useful when we want to roll back to a specific version in order to fix some bugs.
- We usually pick a user story from pivotal tracker and update the team in Slack the user story that we are working on. After that, we started implementing our part and once we completed, we created our own branch and pushed to GitHub. By doing this, we guarantee that the code does not have conflict with other branches. Then, we created a pull request for our team to review. Once it is reviewed and approved, we merge it into the main branch.
- We have one branch for one user story and we have 5 releases in total for each iteration that we accomplished.

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

- We had problems deploying the legacy code base to Heroku because the Ruby version and other gems were outdated and incompatible with Heroku. So, updating Ruby and the gems to the latest version broke our code and literally showed us hell. We also ran into issues with our CI/CD workflow, where our code was not merging properly and deployed to Heroku.

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

- Alex:Big issues using Slack. Standard features are locked behind Pro Access subscription making it difficult to use reliably. AWS Cloud9 had issues finding commands for Github and Heroku.

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

- Semantic-ui-sass: This gem provides the Sass version of the Semantic UI CSS framework. Semantic UI is a popular front-end development framework that allows developers to easily create responsive and modern user interfaces using pre-defined CSS classes and JavaScript components.
- Jquery-rails: This gem provides the jQuery JavaScript library for use in Ruby on Rails projects. It makes it easier for developers to use jQuery in their Ruby on Rails applications by automatically adding the jQuery library to the asset pipeline.
- Rspec-rails: This gem provides integration between the RSpec testing framework and the Ruby on Rails web application framework. It provides an easy way for developers to use RSpec for testing their applications, with a set of Rails-specific tools and helpers to simplify testing of Rails functionality.
- Factory_bot_rails: This gem provides a convenient way to create test data for Ruby on Rails applications using the Factory Bot gem.
- Cucumber-rails: This gem provides integration between the Cucumber testing tool and the Ruby on Rails web application framework. It can help developers to easily use Cucumber for testing their applications, with a set of Rails-specific tools and helpers to simplify testing of Rails functionality.

12. Make sure all code (including Cucumber and RSpec!) is pushed to your public GitHub repo.

13. Make a separate section discussing your repo contents and the process, scripts, etc., you use to deploy your code. Make very sure that everything you need to deploy your code is in the repo. We have had problems with legacy projects missing libraries. We will verify that everything is in the repo.

Final Report - CSCE 606 Software Engineering

Our repository can be divided into three main categories because Ruby adheres to the Model-View-Controller Architecture: the model is in charge of the application logic, the view is in charge of rendering content, and the controller establishes a connection between the model and the user (user requests). We have one of these for each of our various use cases, and we give them names accordingly (students, subjects, users, managements,professors). Additionally, we have files for our imports, which are also a component of the Ruby architecture and are known as GemFiles. The Cucumber testing files that were used to test our features are also available. The documentation folder, which contains all six of our iteration reports, is the last item. On our Github repository, we have also included a link to our Heroku deployment. There is no additional script needed to deploy as we have added automatic deployment to our repository.

14. Links to your Pivotal Tracker, public GitHub repo, and Heroku deployment, as appropriate. Make sure these are up-to-date.

- Pivotal tracker: https://www.pivotaltracker.com/n/projects/2629140
- GitHub: https://github.com/MuyshondtTAMU/ta-management/
- Heroku deployment: TaManagement (csce-636-ta-management.herokuapp.com)

15. Links to your presentation video and demo video.
    i. FYE TA Mgmt - YouTube