Group 14 Sprint 2 Retrospective

OURPURDUEPLAN

Team Members: Neil Van Eikema Hommes, Cooper Chiappetta, Michael Dimitrov, Jack Roscoe

What went well

Over this sprint, we were able to successfully integrate almost all of the features contained in our backlog. Similarly to the previous sprint, a lot of our features were able to be implemented on the backend first and then integrated at the time of creation on the frontend, which minimized the integration time troubles we had in sprint 1. Each member was very quick to respond to the needs of other members, and we were able to keep major communication mishaps to a minimum. Most of us were able to finish our tasks with time to spare before the review, but we finished with a comfortable window of time in case a final issue arose.

User Story #1:

As a user, I would like to be able to be able to branch off of semesters to avoid creating plans with duplicate starting semesters.

#	Description	Time	Team	Owner
1	Update plan editor to be able to create branches	3hr	Frontend	Cooper
2	Update plan editor to be able to cycle through branches with the arrow buttons	3hr	Frontend	Cooper
3	Update plan editor with a delete button that deletes an entire branch	2hr	Frontend	Cooper
4	Branching API integration	3hr	Frontend	Cooper
5	Backend API for creating branches	3hr	Backend	Jack
6	Store branches in database	4hr	Backend	Jack
7	Unit tests for branching	2hr	Backend	Neil

Completed

Upon visiting the plan editor, each semester UI element has a branch button which creates one or two new semesters depending on the context and one new branch to support it. When a split occurs, the element spawns arrow buttons which can be used to cycle between the available branches at that level. In the bottom right corner, there is a branch bubble element containing a trash can which deletes the current branch when clicked.

User Story #2:

As a user, I would like to be able to post a plan to the forum that has branches by choosing which branches to display.

#	Description	Time	Team	Owner
1	Update post creation form with an option to select branches if there are any	3hr	Frontend	Cooper
2	API integration update to send the branch IDs when creating a new post	2hr	Frontend	Cooper
3	API logic to create a post with a hard copy of a plan	2hr	Backend	Jack
4	API logic to choose specific branches of that plan to create the hard copy	1hr	Backend	Jack
5	API logic to update copying a plan with these rules	1hr	Backend	Jack
6	Unit tests for creating hard copies of plans from choosing branches	2hr	Backend	Neil

Completed

Upon navigating to the new post page, selecting a plan that contains multiple branches spawns a new dropdown that lets the user choose which branch to post, and the preview updates accordingly. When the post is created, the API creates a post with a hard copy plan of the branch that was chosen. When a user copies a plan from a post, that hard copy will be copied and the other branches from the original plan are not carried over.

User Story #3:

As an admin, I would like to be able to moderate forum posts and comments and delete unwanted ones.

#	Description	Time	Team	Owner
1	Update post page to have a delete button for admins	1hr	Front	Cooper
2	Update comments to have delete buttons for admins	1hr	Front	Cooper
3	API integration to send delete requests for posts and comments	2hr	Front	Cooper
4	API logic to delete a post or comment	2hr	Backend	Jack
5	API logic to only allow deletion if the user is an admin	2hr	Backend	Jack
6	API logic to send permissions data to the frontend when getting your own profile data to know if the current user is an admin	2hr	Backend	Jack
7	Unit tests for deleting posts and comments with/without admin permissions	2hr	Backend	Neil

Completed

Upon navigating to a post, admins will see trash can icons contained in the post information card and each comment card. Deleting a comment will change the comment to say “[deleted]” and deleting a post deletes the post and takes the user back to the forum page. Non-admins do not see the icons, and sending requests to the API will not work because of the permission rules.

User Story #4:

As an admin, I would like to be able to add, edit, and delete courses from the database.

#	Description	Time	Team	Owner
1	Update course page with a delete button that only appears for admins	1hr	Front	Cooper
2	Update course page with an edit button that only appears for admins	1hr	Front	Cooper
3	Update the courses page with an add course button that only appears for admins	1hr	Front	Cooper
4	Craft a create course form	2hr	Front	Cooper
5	Update the course page to be editable after an admin clicks the edit button	3hr	Front	Cooper
6	API integration to support adding editing and deleting courses	3hr	Front	Cooper
7	API integration to support resetting a course's difficulty	1hr	Backend	Neil
8	API logic to support create, update, delete on courses	2hr	Backend	Neil
9	Unit tests for creating, updating, and deleting courses with and without admin permissions	2hr	Backend	Neil

Completed

Upon navigating to the course search page, admins see an add button that takes them to the new course page, where they can add a course to the database. Upon navigating to a course page, admins have buttons to edit or delete courses. Deleting a course makes it unavailable.
Non-admins do not see this functionality and API requests are rejected because of permissions rules.

User Story #5

As a user I would like to be able to see prerequisites for courses on the course page, and click a link to visit that prerequisites course page.

#	Description	Time	Team	Owner
1	Add links to the course page of each prerequisite	1hr	Frontend	Michael
2	Remove the “prerequisites” section if there are no prerequisites	1hr	Frontend	Michael
3	Request the prerequisites for each course from the server	2hrs	Frontend	Michael
4	Backend API logic to return the prerequisites of a course	3hrs	Backend	Jack
5	Unit tests for course prerequisites	2hrs	Backend	Neil

Completed

Upon navigating to a course page, there is a card which shows the prerequisites of the course. Clicking a prerequisite will navigate to its course page.

User Story #6

As an advisor, I would like to convert my account to an advisor account, and know when my account is approved via an email.

#	Description	Time	Team	Owner
1	Add an option in settings to convert to an advisor account, if they are not already an admin	1hr	Frontend	Michael
2	If user is an admin put a badge next to their profile picture	1hr	Frontend	Michael
3	Have a confirmation message when the user tries to convert advisor account	1hr	Frontend	Michael
4	Send a request to the server to change account to advisor account	1hr	Frontend	Michael
5	Backend API logic to convert an account to an advisor account	2hr	Backend	Neil
6	Send an email confirmation when an advisor account is approved/denied	2hr	Backend	Neil
7	Unit tests for advisor accounts.	2hr	Backend	Neil
8	Admin dashboard to view pending users and accept them	1hr	Frontend	Cooper
9	Fetching/accepting pending users API integration	1hr	Frontend	Cooper

Completed

Upon navigating to the user settings, the user now has the ability to request to convert to an advisor account (provided they are not already an advisor). This sends a request to the admin accounts, who now have a dedicated page where they can approve advisors. When the advisor is confirmed or denied by an admin, an email is sent to them with the appropriate message.

User Story #7

As an advisor, I would like to save a student's plan to "My Student's Plans".

#	Description	Time	Team	Owner
1	Create a new section of the plans page for plans an advisor has saved	2hrs	Frontend	Michael
2	Add error message on saved plans page for when the advisor has saved no plans	1hr	Frontend	Michael
3	Add a button on posts that advisors only can see, so they can save plans	2hr	Frontend	Michael
4	Add functionality to save plans when the user taps the save plan button	2hrs	Frontend	Michael
5	Navigate to the advisors "Saved plans" page	1hr	Frontend	Michael
6	API route to fetch all an advisors student plans	2hr	Backend	Jack
7	API route to save a student's plan to my students plans	2hr	Backend	Jack
8	Store which plans are in the advisors student plans in the database	2hr	Backend	Jack
9	Unit tests for advisors storing plans	2hr	Backend	Neil

Completed

Upon navigating to the forum page, if the user is an advisor, they can now save plans, which maintains all the original details of the plan. The plan is added to a new section of the plans page which is dedicated for advisors. Saving the plan automatically redirects the advisor to the new "Saved Plans" section of the plans page. Lastly, if the advisor has not saved any plans, they are shown a message which explains that the plans can be saved from the forum and will appear in the "Saved Plans" list.

User Story #8

As an advisor I would like to be able to search for users to request to add them as my student.

#	Description	Time	Team	Owner
1	Add a user search page to search for users by username	2hrs	Frontend	Michael
2	Display an appropriate error message if no users are found	1hr	Frontend	Michael
3	Show a list of usernames that match the search text	2hrs	Frontend	Michael
4	Frontend logic to search for users	2hrs	Frontend	Michael
5	Button to request user to be advisor's student	2hrs	Frontend	Michael
6	API route to fetch and search for users	2hrs	Backend	Neil
7	API route to add user to advisors student list	2hrs	Backend	Neil
8	Unit tests for adding students to advisor's list	2hrs	Backend	Neil

Completed

Upon navigating to the settings page, advisors can see a new section where they can lookup users and add them to their "Student List". There is a search bar which searches for users that match the search term, and displays relevant results in a dropdown. If no results match, an error message which says that no students match, is shown. If the advisor adds the student to their list of students, the list UI is updated to reflect the change.

User Story #9

As an advisor I would like to be able to share plans with my students.

#	Description	Time	Team	Owner
1	Add a button on the advisor's page to share plans with a selected student from a dropdown of advisor's students	1hr	Frontend	Michael
2	Add a dropdown selector for advisor's students	1hr	Frontend	Michael
3	Frontend logic to add a plan to users shared plans section	2hrs	Frontend	Michael
4	Backend logic to add a plan to users shared plans	3hrs	Backend	Jack
5	Unit tests for adding a plan to users shared plans	2hrs	Backend	Jack

Completed

Upon navigating to the plans page, an advisor is able to share any of their plans or their saved plans. When they click the share button, they are presented with a window where they can search/select to share with any of their students from a dropdown. Once they have selected a student, the share button is enabled, and when tapped, will share the plan with the selected student and close the popup.

User Story #10

As a student I would like to be able to view plans that my advisor shared with me

#	Description	Time	Team	Owner
1	Frontend page to view all plans that an advisor has shared with a student.	2hrs	Frontend	Michael
2	Frontend logic to display the recommended plan	1hr	Frontend	Michael
3	Frontend button to accept / reject the recommended plan	1hr	Frontend	Michael
4	API route to get a list of plans that an advisor shared with a student	2hr	Backend	Neil
5	API route for a student to accept a plan that an advisor shared	2hr	Backend	Neil
6	Unit testing for students accepting advisor recommended plans	2hr	Backend	Neil

Completed

Upon visiting the plans page, students can see a new section for “Shared Plans”. This section displays a list of plans an advisor has shared with them. The student is able to accept or reject these plans. If accepted, the plan will be moved to the students “My Plans” list, and if rejected, it will be deleted.

What did not go well

Overall, this sprint went pretty smoothly, with only a couple minor problems that we ran into. One of the most notable problems had to do with advisors. In our planning, we pushed almost all work involving advisors to the last week of the sprint. This meant that both frontend and backend were being worked on at this time, and the frontend UI that we needed heavily relied on the backend being complete, leading to a mad dash to finish advisors right before the review, which consisted of a couple large stories with lots of tasks. Additionally, we ran into problems where the frontend contained some very unlikely bugs that required a good amount of time to fix during the final week, perhaps due to a lack of thorough frontend testing once tasks were completed.

Additionally, we ran into fewer problems than sprint 2, but we still managed to omit acceptance criteria that just made sense to be there, and some of our criteria were hyper-specific and probably not necessary. We were still able to make the app usable and add features that we didn't specify in the criteria, but the fact that we were aiming to accomplish some criteria that didn't make complete sense got in the way somewhat. Additionally, we forgot a crucial criteria/task and had to add it in the middle of the sprint, but were still able to get it all done.

Not Completed

Everything that we said we were going to do was again completed this time around.

How we can improve

In our final sprint, we continued to improve as a team to accomplish all of our tasks and integrate cleanly. However, if there was another sprint, we could still make some improvements. There were a couple tasks which seemed like more features could benefit the user experience.

We did better this sprint in regards to team communication, but ultimately there were some delays in getting tasks done due to external factors. This messed up a lot of our task scheduling, and next time our scheduling should focus on more broad tasks rather than specific things, as it is much easier to redo.

Next time, we should also plan out our task dependencies better. There were issues with tasks being dependent on other tasks, which caused some group members to have to wait until another group member finished. This ultimately made the people who had to wait have a big

time crunch to get things done before the deadline at the end. We could have done better in either assigning tasks and the tasks dependent on those tasks better, or assigning people tasks that are mutually exclusive to do simultaneously so everyone has something to do at all times.

Lastly, we should test our code better before immediately pushing and merging with main. A few times during this sprint, code that had major flaws was pushed, and those flaws were only found through manual testing, or while the test cases were being written. Next time, the test cases should have been written earlier and adequate testing should have been done by the author of each story before pushing.