Student Knowledge System

Final Report

CSCE 606 - Software Engineering

Executive Summary

Project Overview

The Legacy Project Student Knowledge System, built on the robust Ruby on Rails framework, is a good initiative that transforms the professor-student dynamic through an online platform. This innovative platform seeks to meet the needs of professors and integrates interactive games and note-taking features to enhance the way educators engage with their students. Through gamified elements, the system captures and retains essential information about each student's personality, learning style etc . Professors and TAs can seamlessly manage student profiles, incorporating academic achievements, extracurricular activities, through personalized notes. This comprehensive approach not only fosters a deeper understanding of individual students but also streamlines the process of crafting impactful recommendation letters. Teachers can see their active courses, their stats like number of students, course name, semesters, sections etc. Professors can also archive courses when they are not teaching it and unarchive them when needed. They can also see course-sem wise stats of the student face name games

that they play so that they can specifically focus on the particular sections. The user-friendly interface ensures efficient navigation, and the reminder system keeps educators informed about important events and milestones. The Legacy Project represents a technological leap forward, fostering stronger, more personalized connections between educators and students while leaving a lasting legacy in the realm of student-teacher relationships.

Apart from main features, the application can also facilitate stakeholders to search the students by name,sort them via different attributes like email, course etc.The stakeholders of the project are Professors, TAs and students as they are the profiles that the project deals with.

User Stories

Table in the order of stories completed.

Story No	Description	Story points	Developer	Status	Iteration
1	Fixing RSPEC issues for CI/CD Pipeline	1	Vivek Narukurthi	Completed	1
2	Fixing column headers for edit student page	1	Vivek Narukurthi	Completed	1
3	Delete course button on the course page	2	Akshit Bansal	Completed	1
4	Auto Convert Student Emails - from @email.tamu.com to @tamu.com	2	Shubham Mhaske	Completed	1
5	Add Notes Section in student profile	2	Vivek Narukurthi	Completed	1
6	Add the “Edit Course” button	2	Shreshth Kushwaha	Completed	1

	on the course profile				
7	Adding archive - unarchive feature	3	Shreshth Kushwaha	Completed	2
8	Adding Rspec test code for archive unarchive feature	1	Shreshth Kushwaha	Completed	2
9	Email format conversion at upload page	2	Shubham Mhaske	Completed	2
10	Rspec tests for Email format conversion at upload page	1	Shubham Mhaske	Completed	2
11	Input validation for creating new courses	3	Akshit Bansal	Completed	2
12	Adding Rspec and cucumber test code and Improving coverage from 50% to 70%	2	Vivek Narukurthi	Completed	2
13	Adding cucumber scenarios and Rspec tests for Add Notes	1	Vivek Narukurthi	Completed	2
14	Adding Course wise practice dashboard for game	3	Shreshth Kushwaha	Completed	3
15	Add search field for searching students by name	3	Shubham Mhaske	Completed	3

16	Add cucumber tests for Add search field for searching students by name	1	Shubham Mhaske	Completed	3
17	Added Drop Down Lists for selecting course information	3	Akshit Bansal	Completed	3
18	Adding RSpec tests for Addition of New Course page	1	Akshit Bansal	Completed	3
19	Adding RSpec tests for validation checks of adding new courses	1	Akshit Bansal	Completed	3
20	Adding notes atomically	3	Vivek Narukurthi	Completed	3
21	Show student profile by clicking on row on student list	3	Shubham Mhaske	Completed	4
22	Add and Fix cucumber tests - Show student profile by clicking on row on student list	2	Shubham Mhaske	Completed	4
23	Preserving context on clicking back button	3	Shreshth Kushwaha	Completed	4
24	Student Search Bar for the Courses page	2	Akshit Bansal	Completed	4

	has been made case insensitive				
25	Toggle button for the stats dashboard has been altered	1	Akshit Bansal	Completed	4
26	Interface has been changed for the Archive Courses page	1	Akshit Bansal	Completed	4
27	User Interface changes for the Courses page	1	Akshit Bansal	Completed	4
28	Pushed changes to production heroku app	2	Vivek Narukurthi	Completed	4
29	Made coverage 85% for Rspec	3	Vivek Narukurthi	Completed	4
30	Add Sorting in Students based on all columns	3	Shubham Mhaske	Completed	5
31	Improve test coverage for Students Controller	3	Shubham Mhaske	Completed	5
32	Add new student button on course page	2	Shreshth Kushwaha	Completed	5
33	Add student count on course page	2	Shreshth Kushwaha	Completed	5
34	Move archive button to edit course page	2	Shreshth Kushwaha	Completed	5
35	Refactoring code for better	2	Shreshth Kushwaha	Completed	5

	coverage				
36	Made the rows on courses page clickable for all the courses	2	Akshit Bansal	Completed	5
37	Updated the New Course Page with information for adding courses	1	Akshit Bansal	Completed	5
38	Made the Students in Current View redirect to Student Profiles on click	2	Akshit Bansal	Completed	5
39	Rspec tests for notes feature	2	Vivek Narukurthi	Completed	5
40	Modified ReadME to include development steps	1	Vivek Narukurthi	Completed	5
41	Making notes atomic	3	Vivek Narukurthi	Completed	5

Story Descriptions

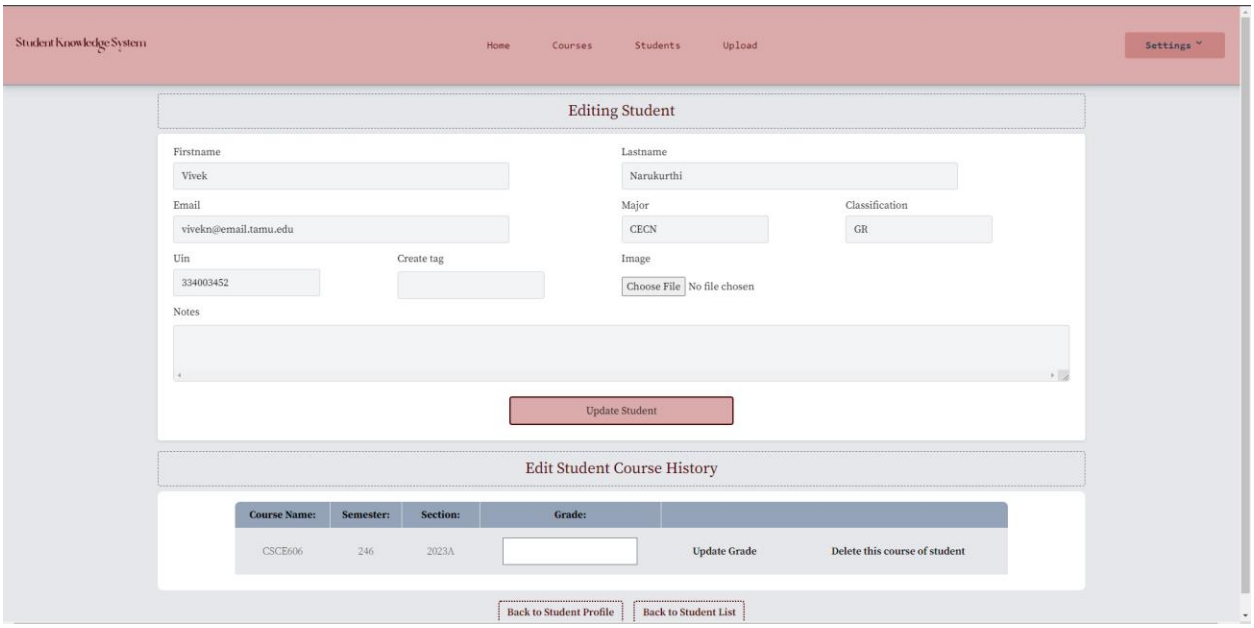
1. Fixing RSPEC issues for CI/CD Pipeline

Resolve RSPEC issues in our CI/CD pipeline to enhance code quality. This involves analyzing static code analysis reports, identifying areas of improvement, and implementing fixes. The goal is to ensure a reliable and efficient pipeline, promoting code adherence to standards before deployment.

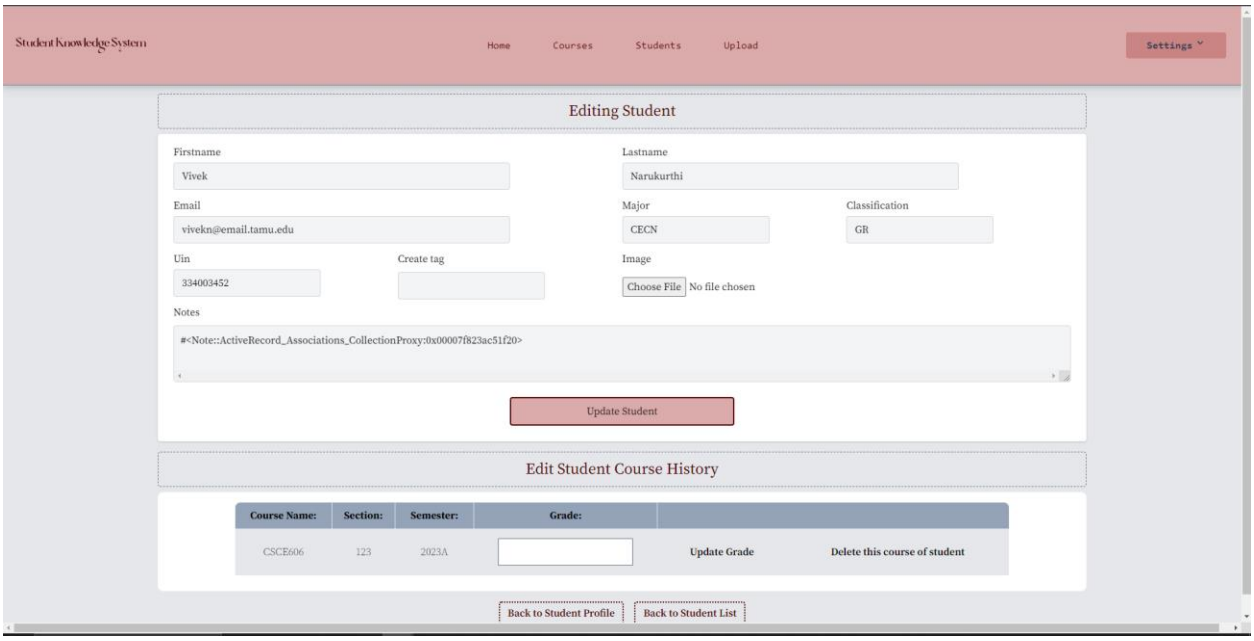
2. Fixing column headers for Edit Student Page

There was a mismatch between the column headers in the Edit Student page which was fixed, this exercise was given a story point of 1 because it involved upskilling about the current codebase.

Previous structure of Edit student page with Bug



Now the Section and Semester have been reversed fixing the UI bug

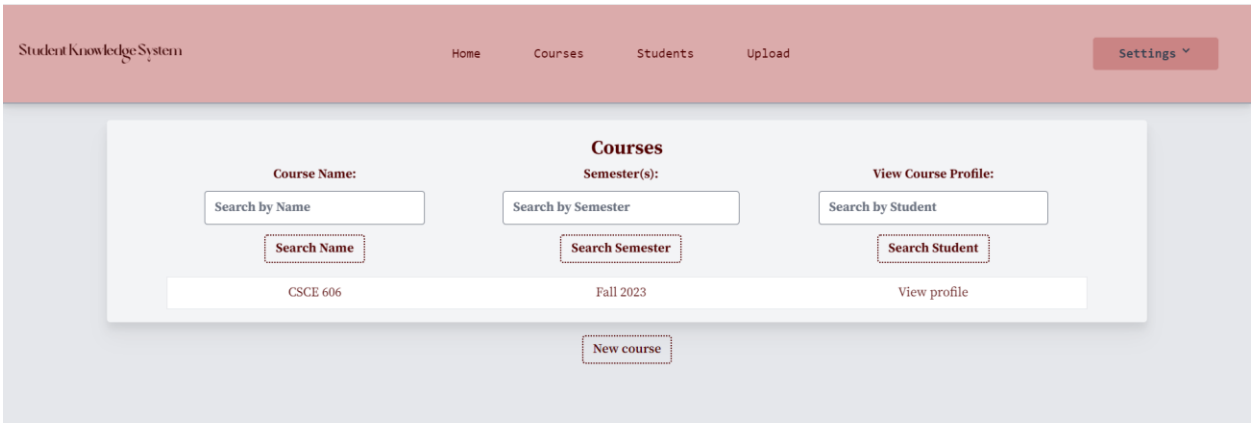


3. Added the 'Delete' button on the 'Courses' page

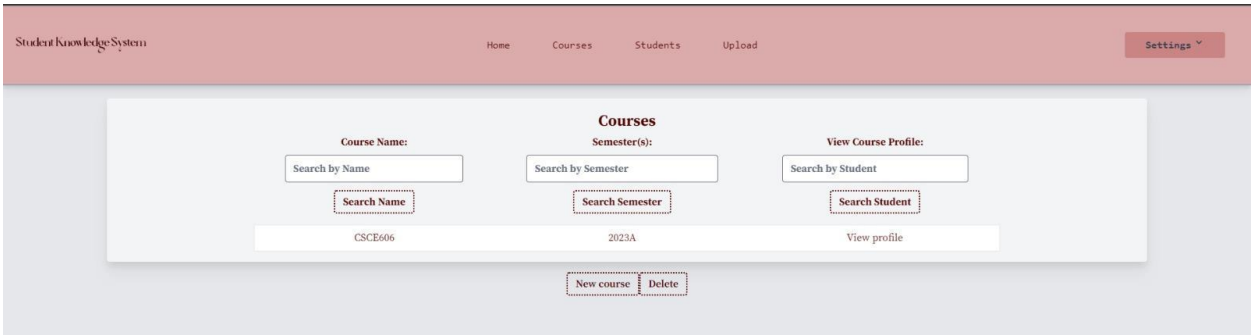
Previously, users were required to navigate through multiple steps, clicking the "View Profile" button and subsequently the "View Course History" button, in order to delete a course. This navigation process posed challenges and complexity for users seeking to delete a course. In an effort to enhance the user experience and make the webpage more user-friendly, we have now incorporated a "Delete Course" button directly on the Course web page itself, streamlining the process for course deletion.

Below are the before and after screenshots:

Before adding the 'Delete' button -

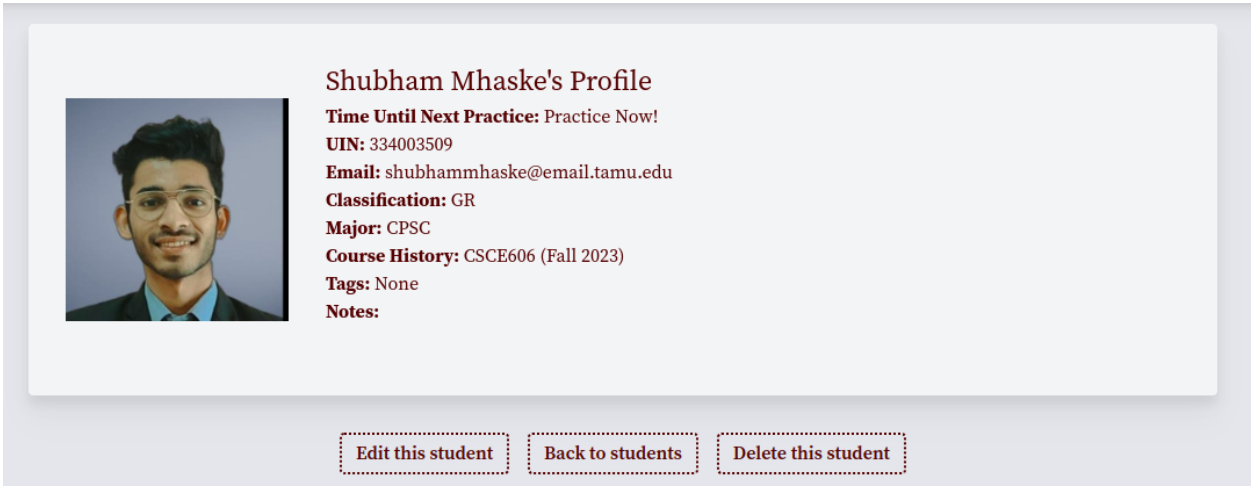


After adding the "Delete" button



4. Auto Convert Emails

The student emails were shown as student_name@email.tamu.edu and now they are changed to student_name@tamu.edu



As we can see the emails of all the students listed as well as the individual students are of the format student_name@email.tamu.com

After the code changes in this story, they will be shown as student_name@tamu.edu. Below are the screenshots after the code change.

Image	Name	Email	Course(s)/Semester(s):	Tags	View Student Profile:
	Bansal, Akshit	akshit.bansal@tamu.edu	CSCE601 - Fall 2023	None	Show this student
	Kushwaha, Shreshth	shreshth.kushwaha@tamu.edu	CSCE601 - Fall 2023	None	Show this student
	Mhaske, Shubham	shubhammhaske@tamu.edu	CSCE601 - Fall 2023	None	Show this student
	Narukurthi, Vivek	vivekn@tamu.edu	CSCE601 - Fall 2023	None	Show this student

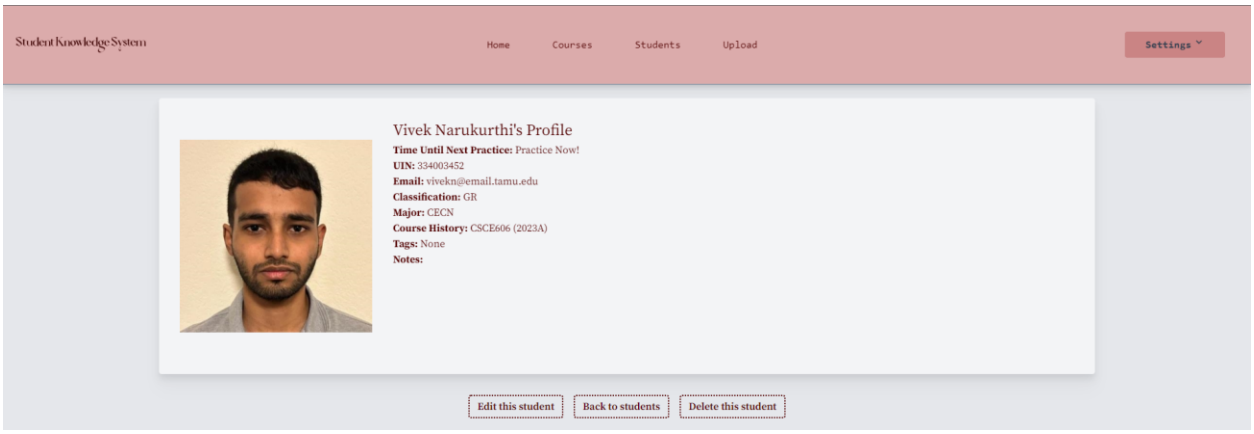
Shubham Mhaske's Profile

Time Until Next Practice: Practice Now!
UIN: 334003509
Email: shubhammhaske@tamu.edu
Classification: GR
Major: CPSC
Course History: CSCE601 (Fall 2023)
Tags: None
Notes:

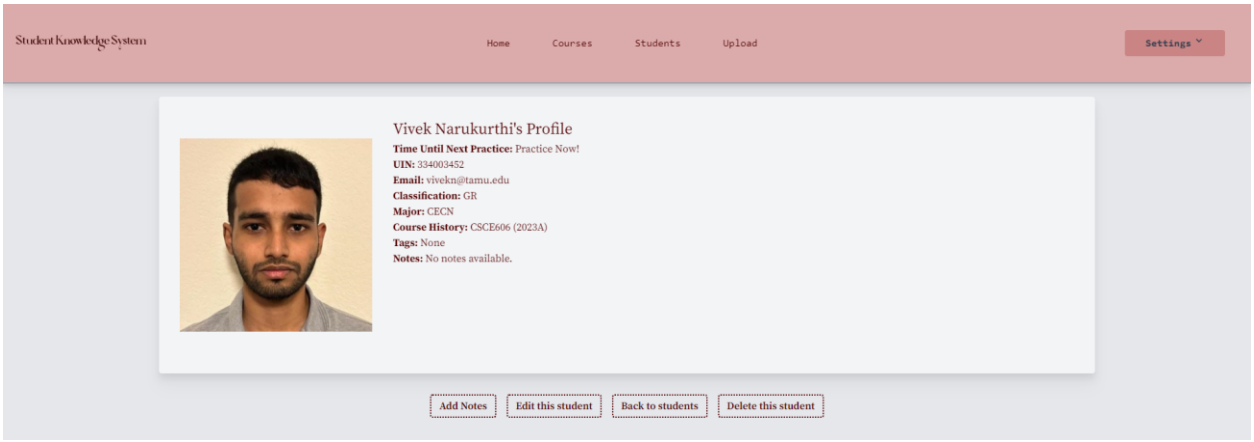
5. Add Notes Section for Student Profile

An Add Notes feature was added to the student profile page for every student. A form would be generated by clicking that button which would append the notes one by one for every new entry.

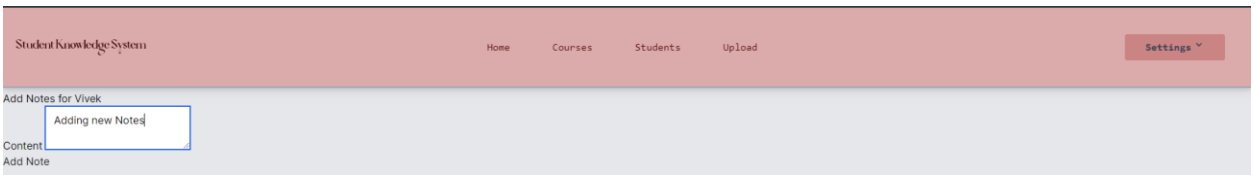
Previous Student Profile Page



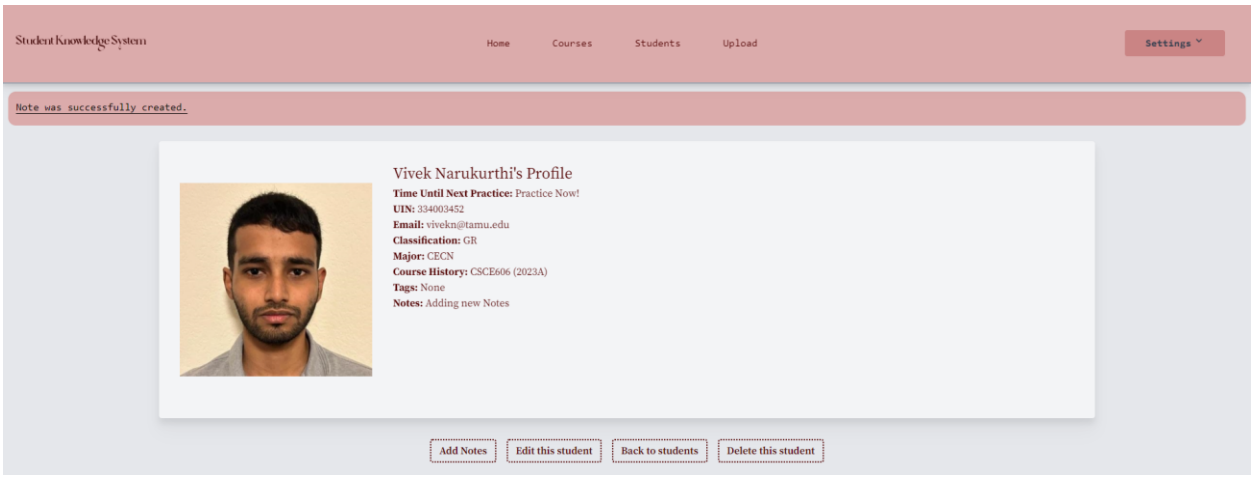
New Student Profile Page



On clicking Add Notes



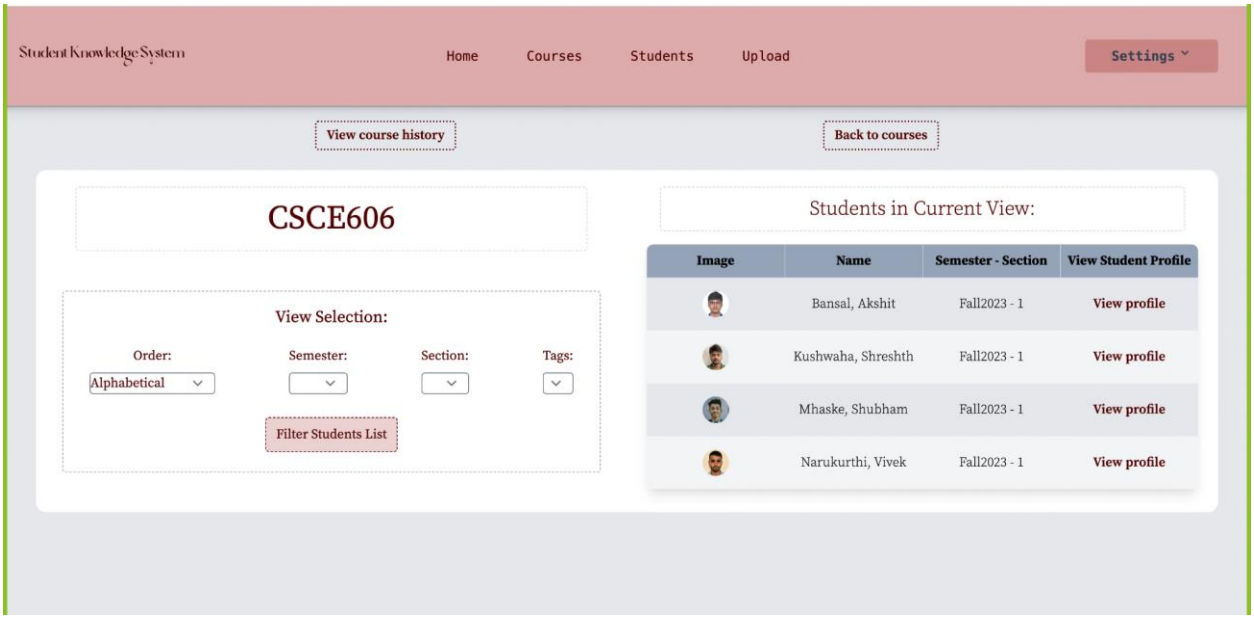
We are adding "Adding new Notes" for the particular student



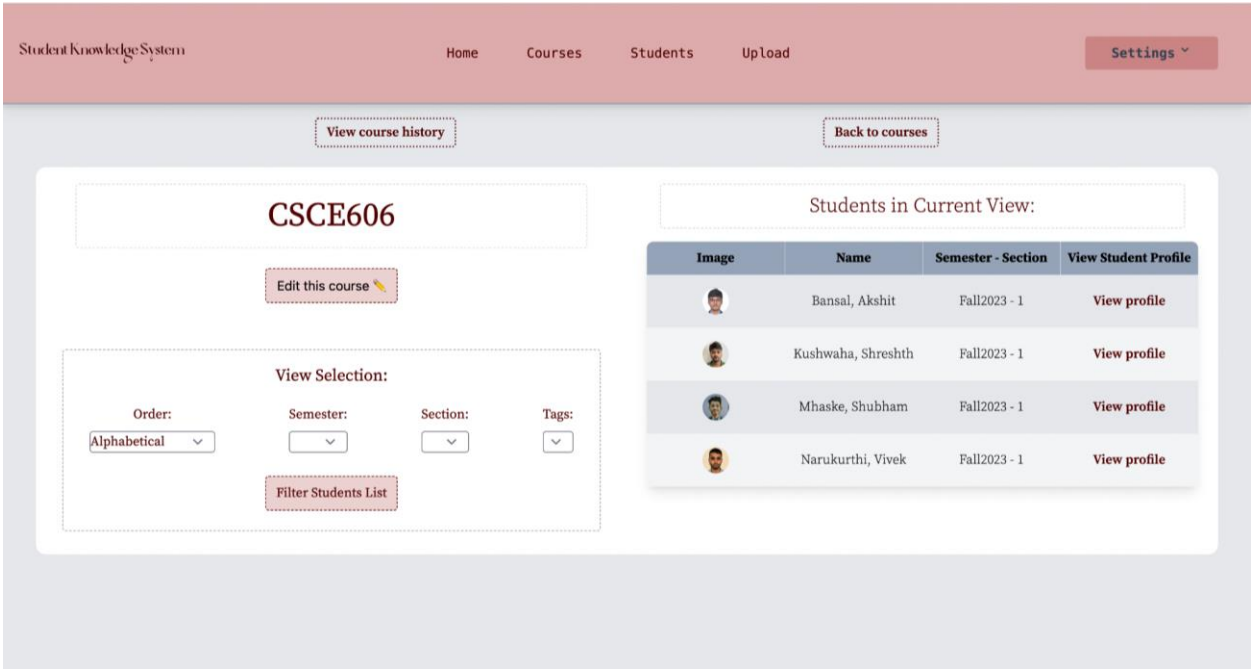
A flash message saying "Note was successfully created" appears and also Adding new Notes has been added to the Notes section of the student.

6. Adding the "Edit Course" button on the course profile

Previously the user had to get to the course, then the course profile, then the course history, and then "edit course" for editing course. This is not a very intuitive flow of steps when someone wants to make changes in the course a better alternative would be to display an "Edit Course" button on the course profile, so that users can intuitively browse the course and then go to edit page. Below are the before and after screenshots.



"Edit this course" button:

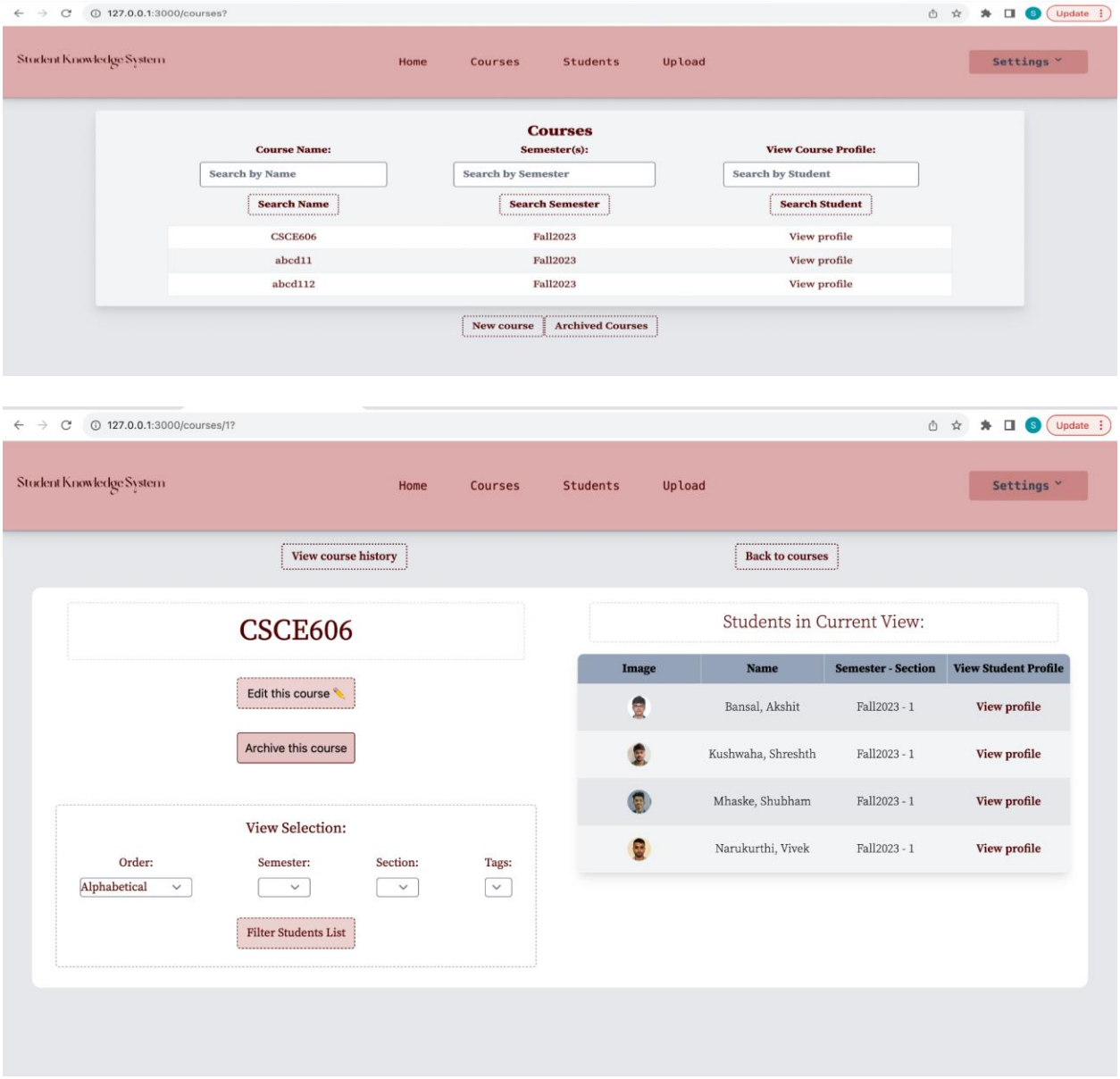


7. Archive and Unarchive Courses

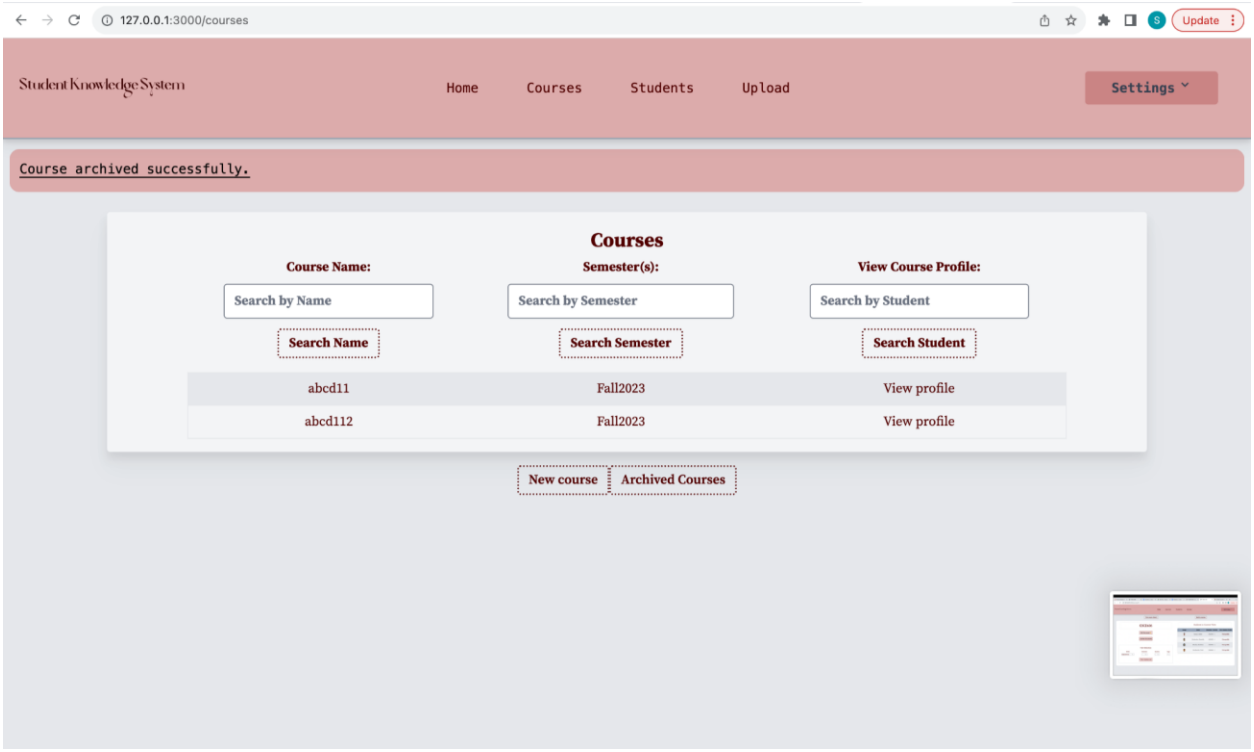
The client asked us for a feature to archive - unarchive the courses that he is not currently teaching or were transferred from his schedule. Previously there was no such functionality , there was only the "delete this course" feature. In this iteration, we implemented archiving and unarchiving features.

I created a separate route and a separate webpage view for the archived courses. Once the course is marked archived in the database the course is visible in the archived view and is not visible in the courses view due to filtering. Changes to code and different MVC files can be visible in the GitHub repository. Users can render these archived courses and then again mark and unarchive them in case something changes later. Here we have attached screenshots for reference.

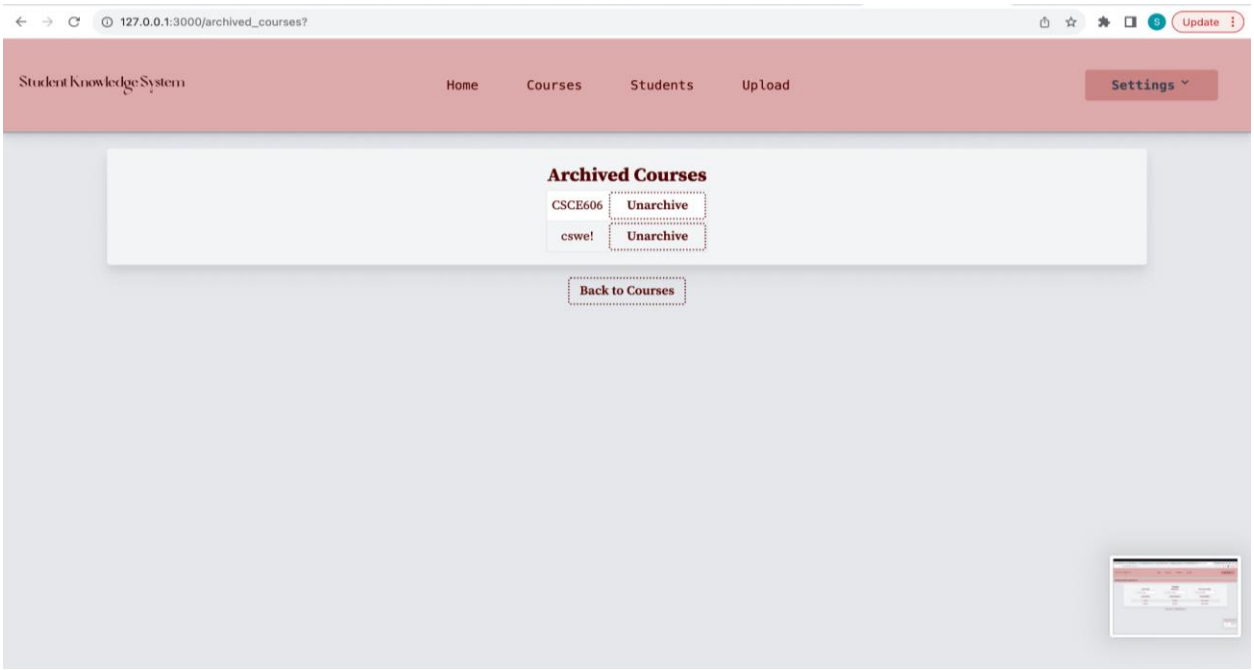
User can click View profile of the course he wants to archive



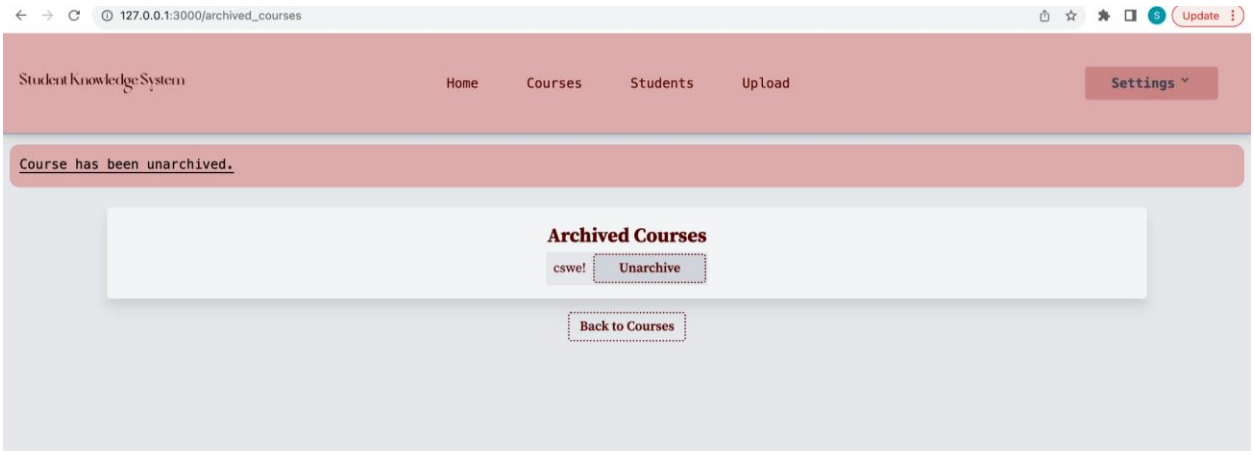
After archiving the course user is directed back to the course page



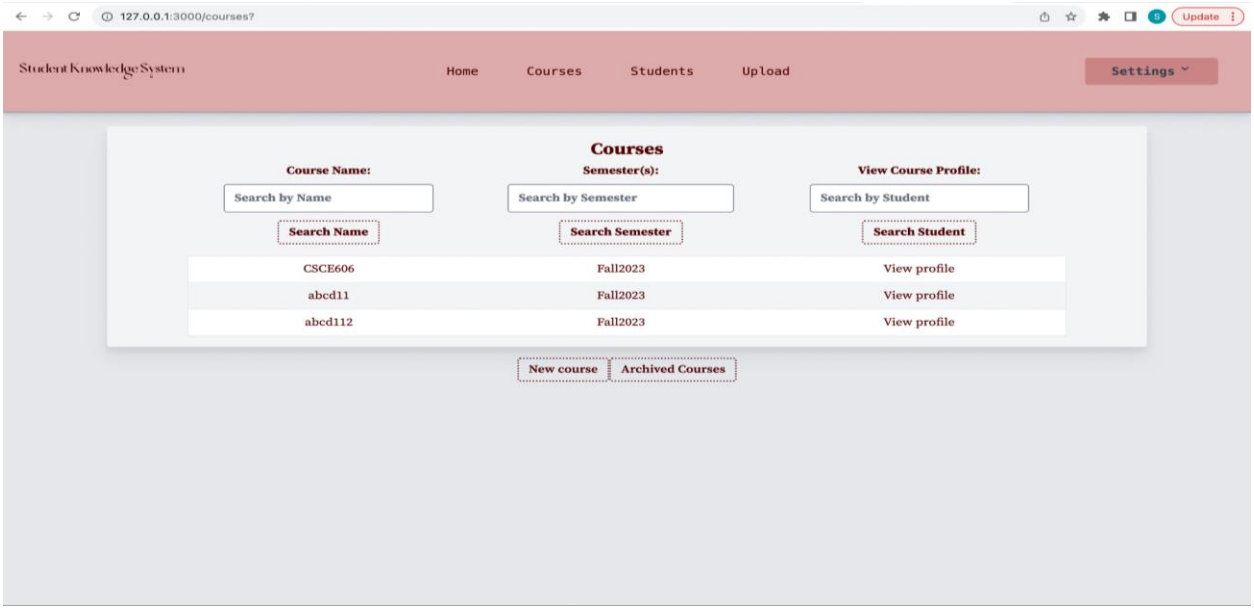
If the user clicks on archived courses he can see all the courses he archived till now and also can unarchive them



When a user clicks unarchive, the course gets removed from the list.



He can go back to the course page to see his past courses back in the courses



8. Adding Rspec test code for archive unarchive feature

Implement RSpec tests for the archive and unarchive feature. This task involves writing test code to ensure the robustness and correctness of the archive and unarchive functionalities. The tests will cover various scenarios, validating the behavior and functionality of these key features within the application.

9. Email format conversion at upload page

Client asked - Whenever the new data is uploaded , the emails should be converted to "@tamu.edu" format from original "@email.tamu.edu" format. And this change should happen before storing the data to the database and should be visible on students and student details page.

If a student already exists with the original email format and we upload new data for the same student, the email should get updated to the "@tamu.edu".

If student does not exist, then student should email should be stored as "@tamu.edu"

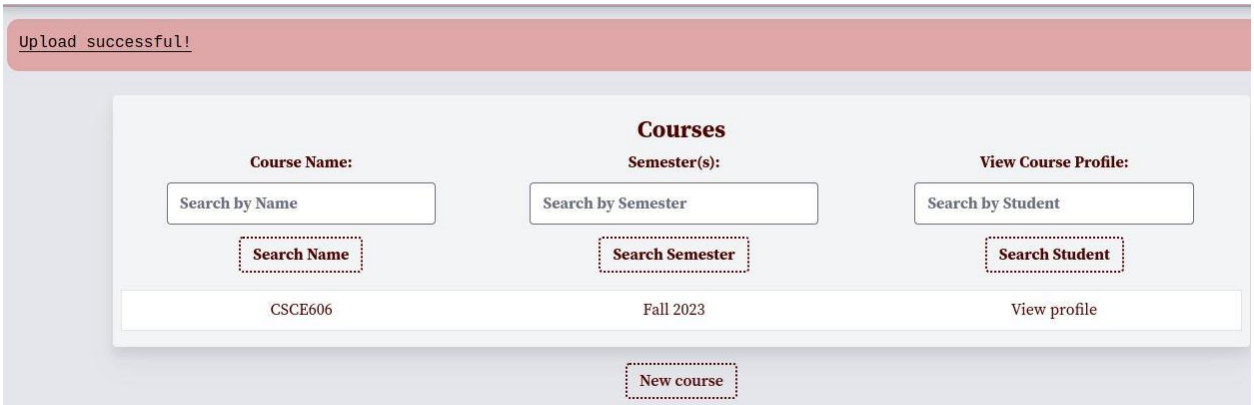
Case 1: No student data is present and we upload new student data.

Before:

No student data present.

After:

We upload student data.



Student data with updated email at index as well as student detail level.

Image	Name	Email	Course(s)/Semester(s):	Tags	View Student Profile:
	Bansal, Akshit	akshit.bansal@tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Kushwaha, Shreshth	shreshth.kushwaha@tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Mhaske, Shubham	shubhammhaske@tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Narukurthi, Vivek	vivekn@tamu.edu	CSCE606 - Fall 2023	None	Show this student

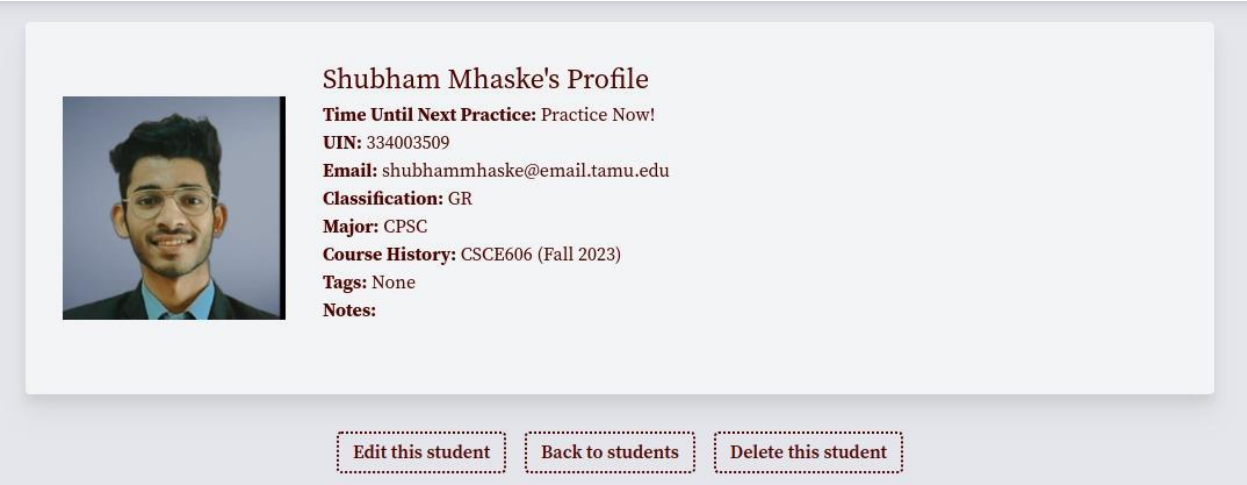
Case 2: Student data already present and we upload the data for the same students. The emails will be updated to the required format.

Before:

Filter Students List

Yearbook View

Image	Name	Email	Course(s)/Semester(s):	Tags	View Student Profile:
	Bansal, Akshit	akshit.bansal@email.tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Kushwaha, Shreshth	shreshth.kushwaha@email.tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Mhaske, Shubham	shubhammhaske@email.tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Narukurthi, Vivek	vivekn@email.tamu.edu	CSCE606 - Fall 2023	None	Show this student



Student emails are in original format at both index and student detail level.

After:

Image	Name	Email	Course(s)/Semester(s):	Tags	View Student Profile:
	Bansal, Akshit	akshit.bansal@tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Kushwaha, Shreshth	shreshth.kushwaha@tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Mhaske, Shubham	shubhammhaske@tamu.edu	CSCE606 - Fall 2023	None	Show this student
	Narukurthi, Vivek	vivekn@tamu.edu	CSCE606 - Fall 2023	None	Show this student

Emails are updated to the modified required format.

10. Rspec tests for Email format conversion at upload page.

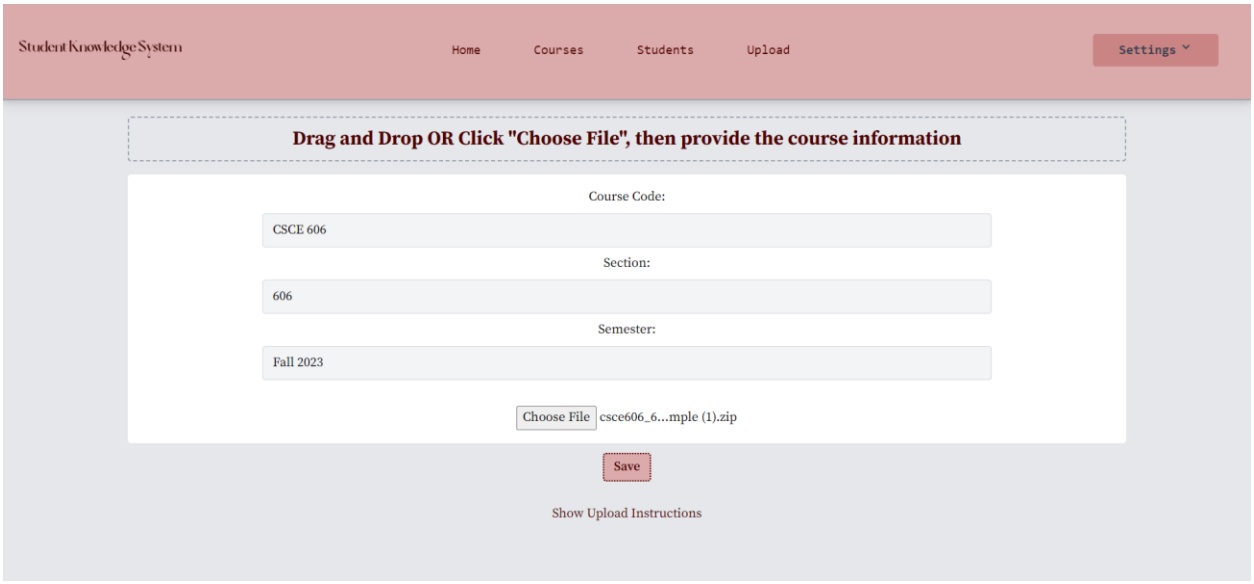
Added RSpec tests for email format conversion at the upload page. The tests verify the correct functionality of email format conversion, ensuring accurate processing and handling of email-related features during file uploads.

11. Input Validation Checks added to Create New Courses

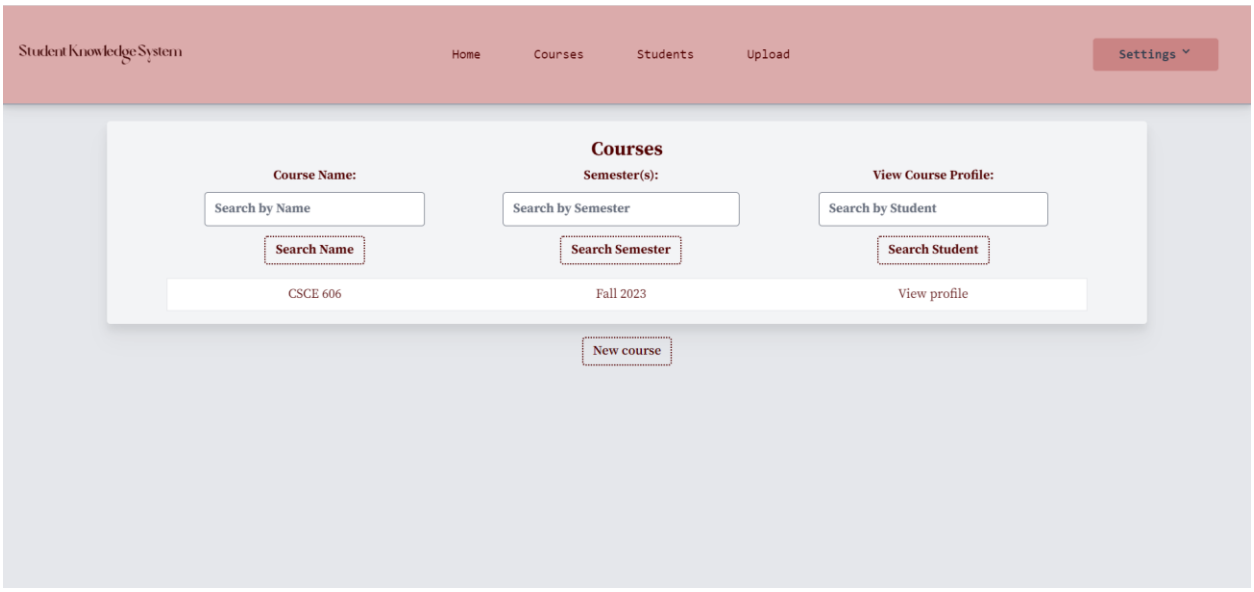
The client has explicitly outlined a crucial requirement to facilitate the smooth integration of new course listings into the webpage. This requirement mandates strict adherence to a predefined data entry format, encompassing specific details such as the course code, course section, and the corresponding semester. This format is meticulously designed to ensure the uniformity and lucidity of data entry procedures. By adhering to this structured format, the user experience is significantly improved, and the overall organization of course-related information is enhanced.

Furthermore, as a means to enforce this format, a failsafe mechanism is being developed. In the event that a user submits course information that does not align with the prescribed format, the system will detect this deviation and trigger an error message, alerting the user to the incorrect formatting. This proactive approach not only safeguards the quality and consistency of the data but also helps users maintain compliance with the required standards, thereby contributing to an efficient and error-free data entry process.

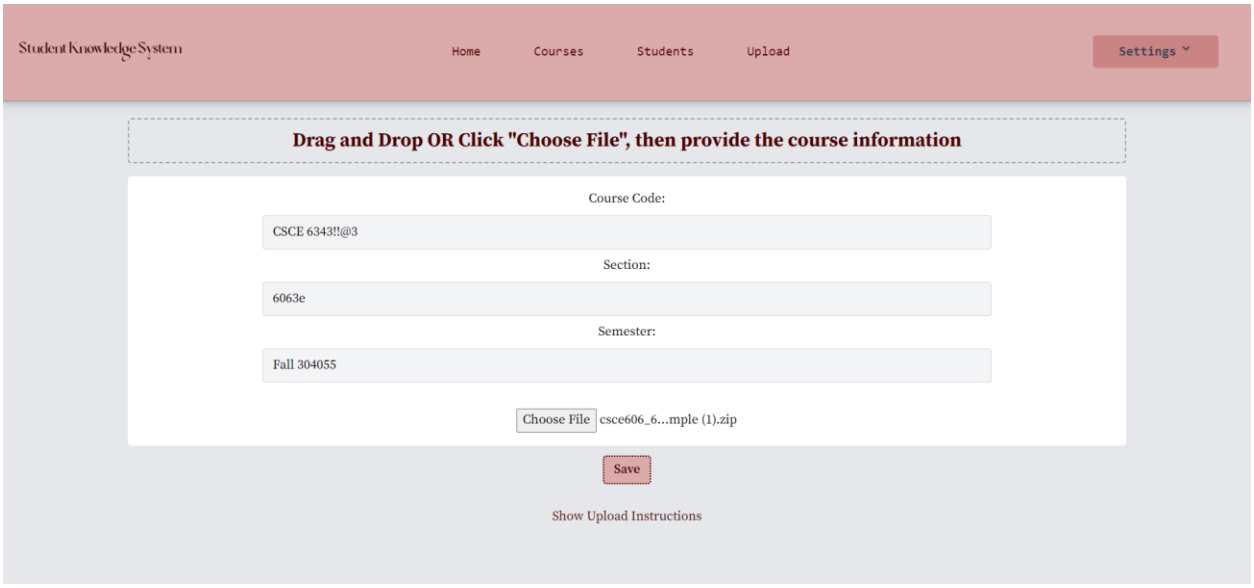
When the user tried to add course information in the correct format i.e. Course Code, Course Section and Semester are added in the correct format (as shown below):



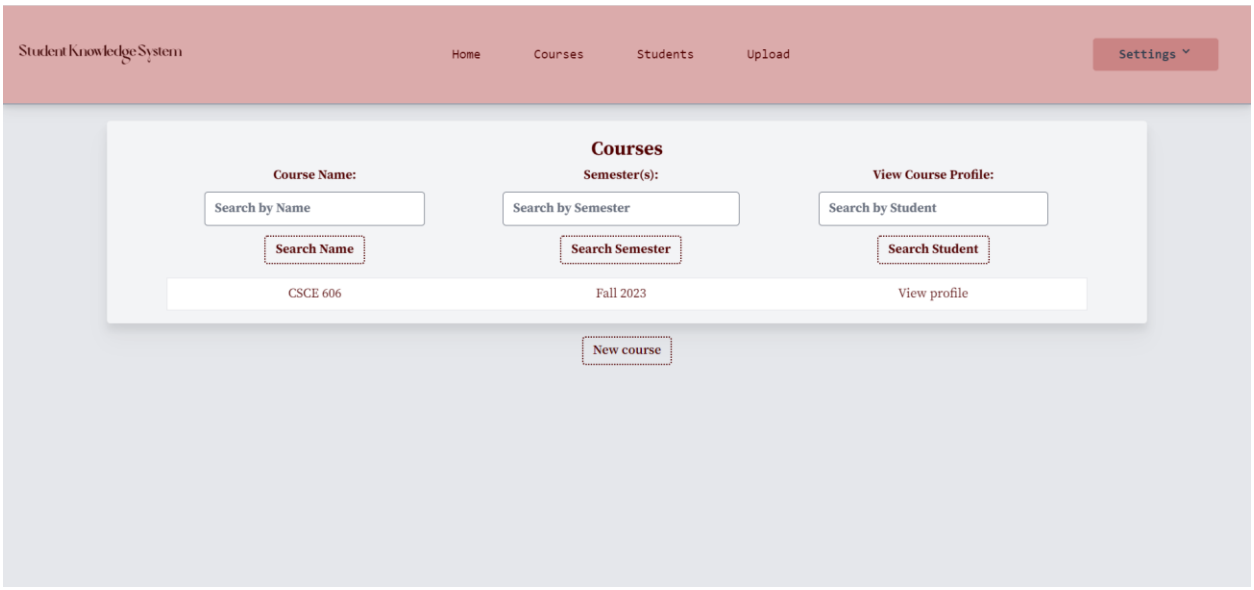
Since the entered information is in the correct format, the course gets added to the courses page as shown below:



Now, let's see the scenario where we enter information in the incorrect format (image below)



Now when the user tries to save this course, it won't get added and it won't display on the Courses page



12. Adding Rspec and cucumber test code and Improving coverage from 50% to 70%

Enhance test coverage from 50% to 70% by adding RSpec and Cucumber test code. This effort involves writing comprehensive tests to validate various functionalities, ensuring a more robust and reliable codebase. The goal is to improve overall test coverage and strengthen the quality assurance process for the application.

13. Adding cucumber scenarios and Rspec tests for Add Notes.

Added 2 cucumber scenarios and 12 steps (6 each) along with the step definitions to test the Add Notes feature.

Rspec tests mentioned below were also added for the Add notes feature.

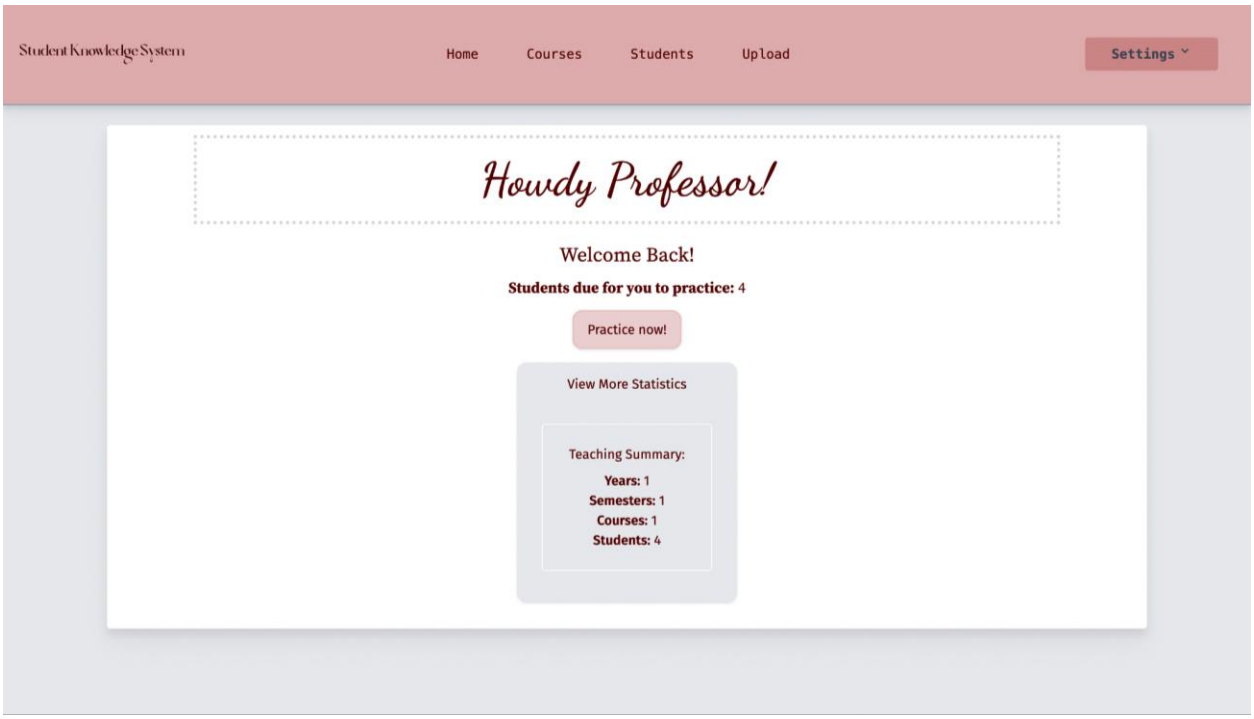
4 examples -> 1) Renders the new note form, 2) Creating a note for existing student, 3) Creating a note for non-existent student, 4) Create a note with missing content

14. Adding course wise practice dashboard on Home page

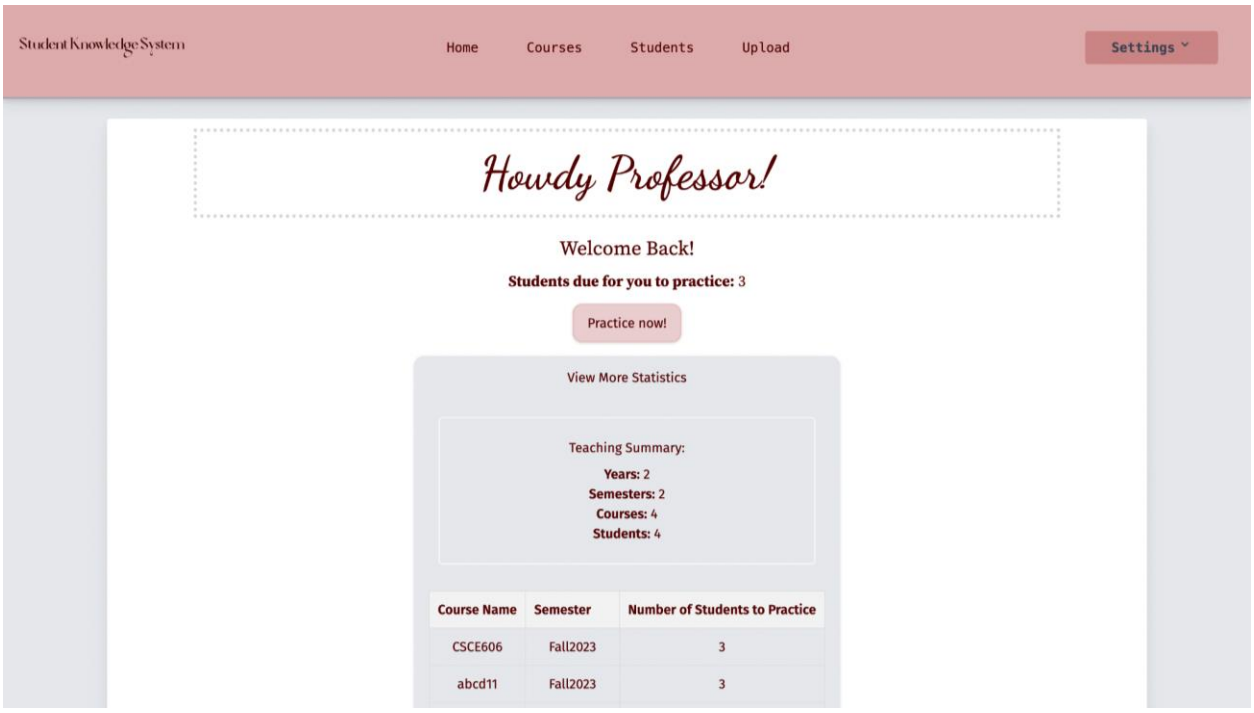
Client wants us to include a comprehensive dashboard on the home page so that he can see the statistics of the students he does not know and have not practiced through game. The dashboard should show him what are the number of students he does not know for a particular course, section, and semester. The idea is that once client know about his progress he will focus on these particular courses while playing the game. We implemented this dashboard in this iteration and might change it as per the client requirements for the next iterations.

We majorly modified the home controller and home view for adding this functionality. Here we are attaching the screenshots for the reference

Before screenshots:

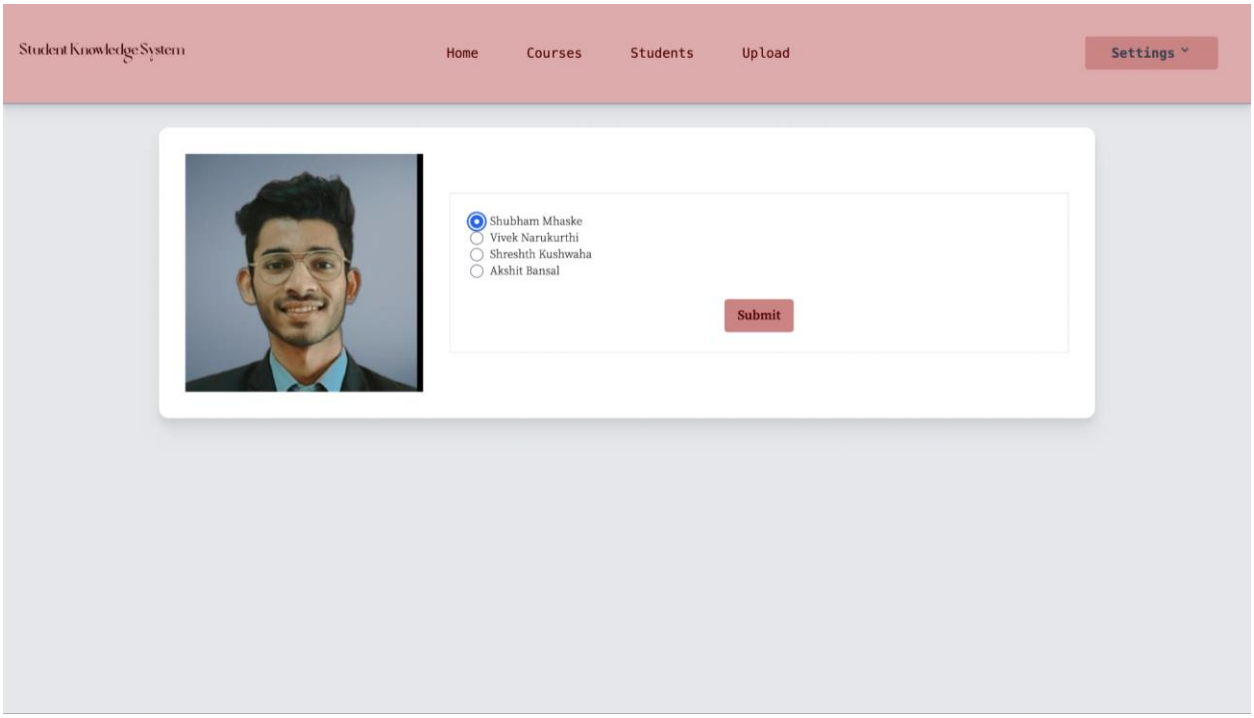


After implementing the feature we can see this:

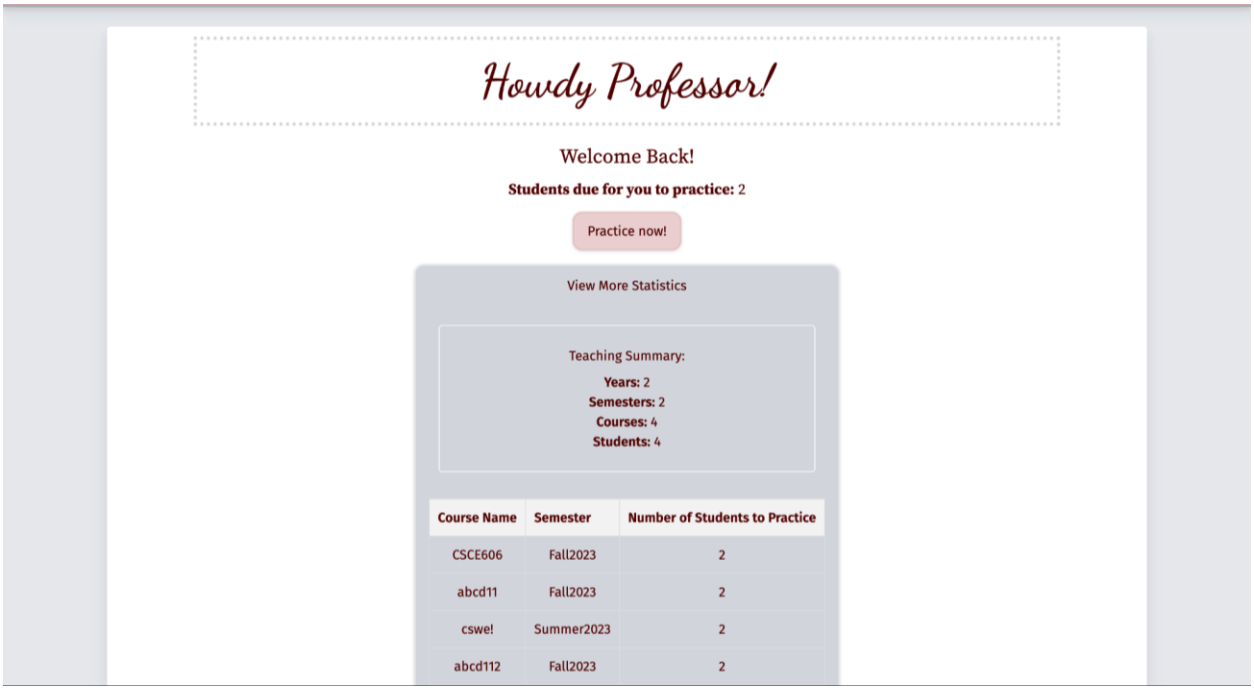


We can see that number of students are 3 which the client/professor does not know. As these three students are common between all the courses so it is same across.

Now we will play the game as client and will also know another student by answering the question correctly.



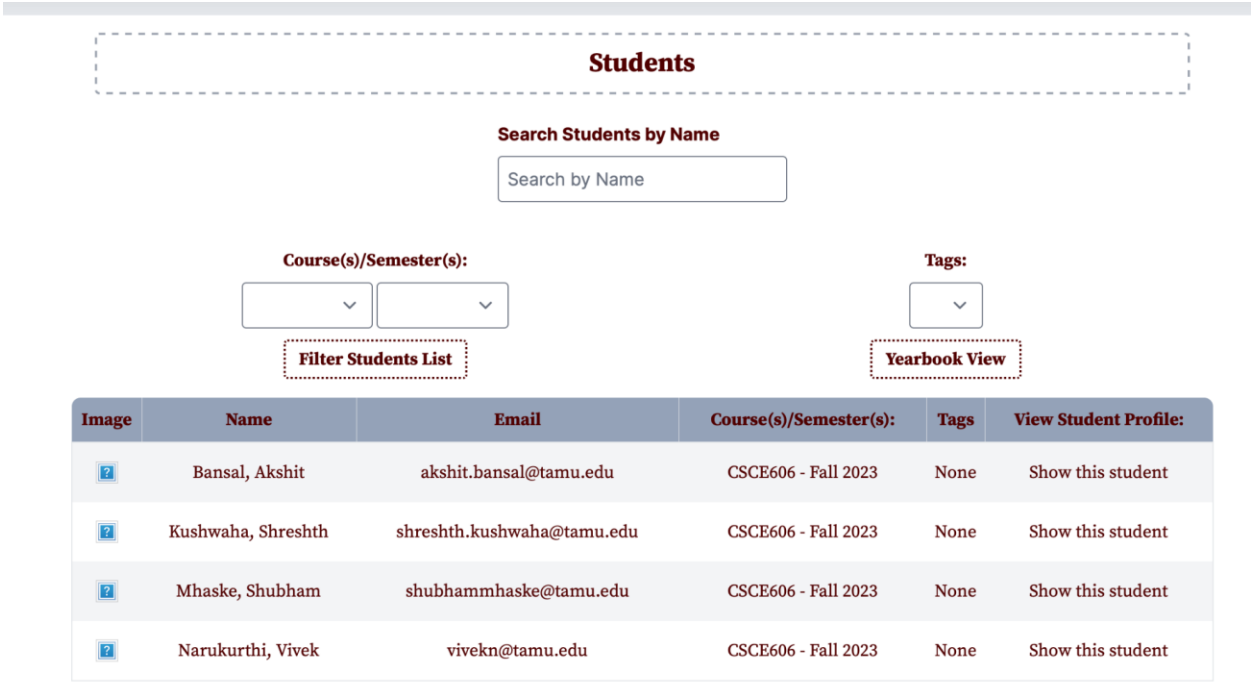
Now we answered correctly and the number of students in the dashboard should decrease.



We can see that the dashboard is working perfectly and it shows updated numbers as we answered question correctly.

15. Add search field for searching students by name

Client wants us to include a search feature on the student index page. Using the search, we can search the students using their names. This search bar works on the go without an explicit search button and searches the students with partial input as well. There is no need to give full student names for search. We implemented this search fun in this iteration and might change it as per the client requirements. Below are the screenshots for the search operation.



16. Add cucumber tests for Add search field for searching students by name

Implement Cucumber tests for the addition of a search field for student names. These tests will validate the functionality of the newly added search feature, ensuring accurate and efficient

searching of students by name. The goal is to enhance the reliability of the search functionality and provide comprehensive test coverage for this specific feature.

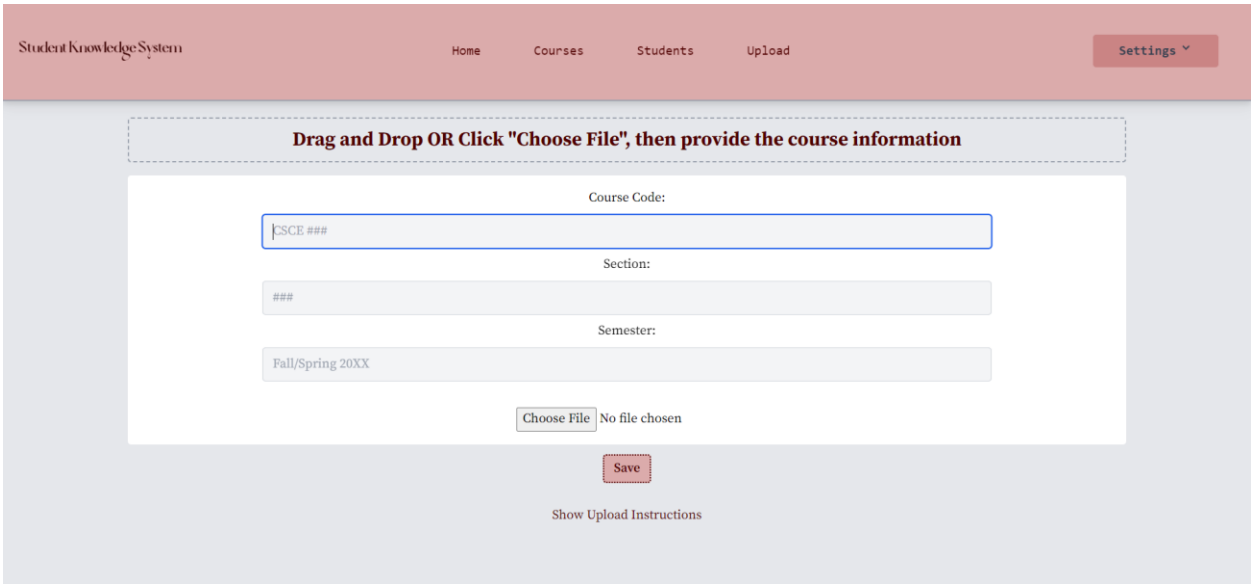
17. Drop Down Lists added for entering New Course information

Client wants us to include a feature where the user has the option to select course information from a drop down list rather than entering it manually. By offering a predefined list of courses through a drop-down menu, users will be less likely to enter course information in an incorrect format. This will help maintain data integrity and reduce the likelihood of data entry errors.

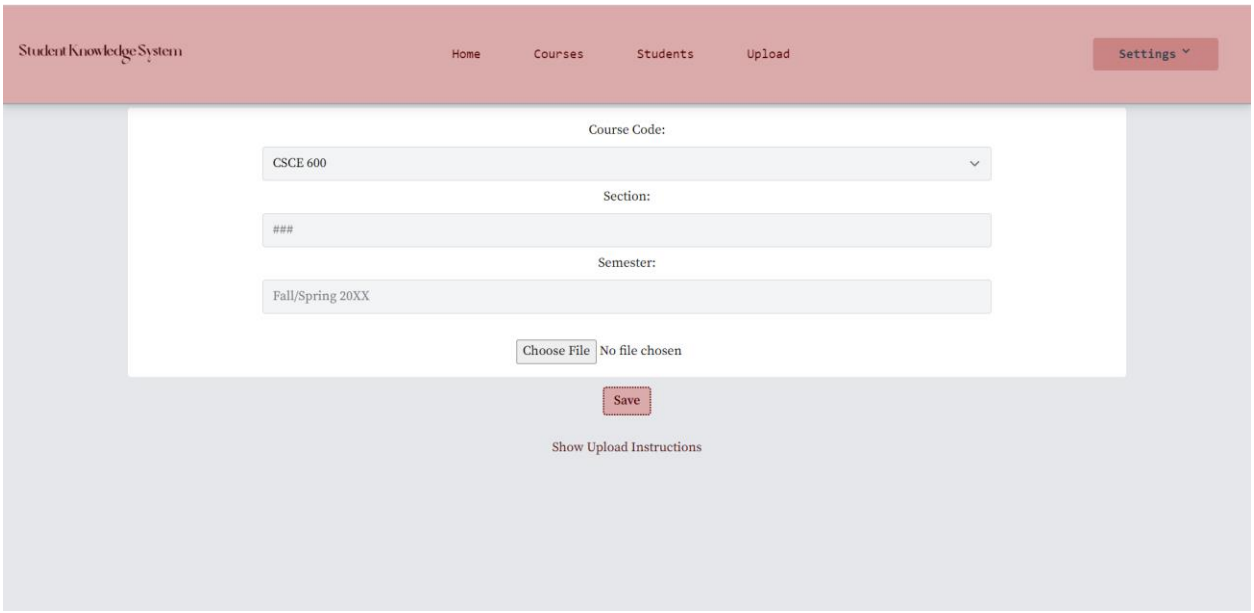
Users will no longer need to recall and manually input course names, which can be challenging, especially for courses with long or complex names. This reduces the cognitive load on users and makes the application more user-friendly.

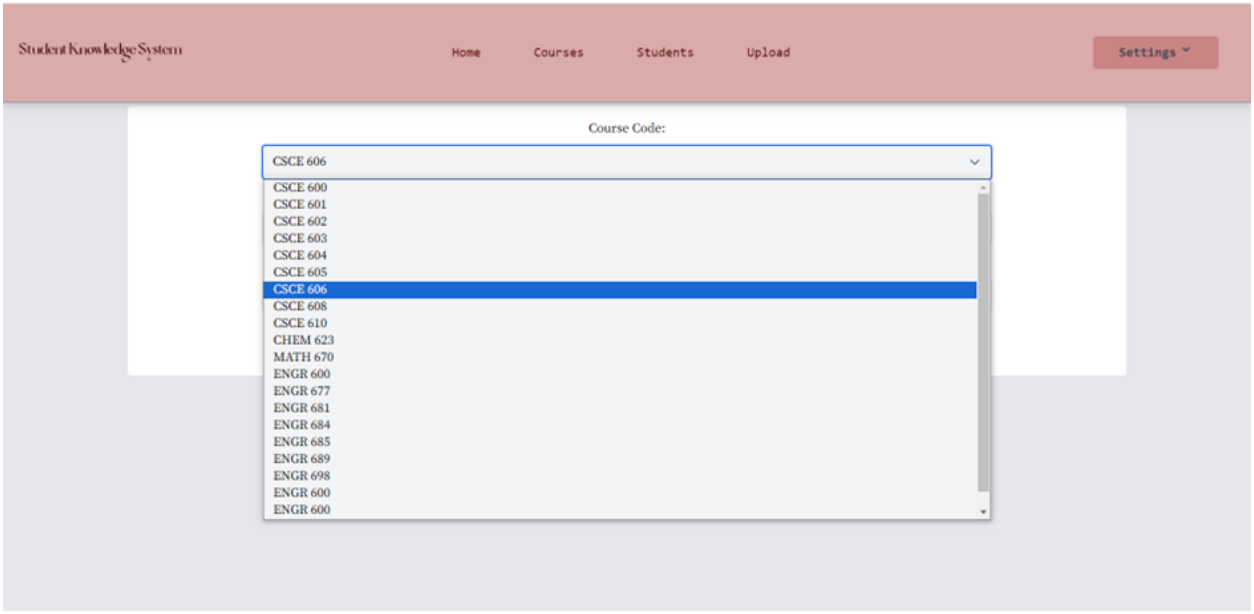
The drop-down list simplifies the user experience by presenting a structured and easy-to-navigate list of available courses. Users can quickly locate their desired course without manual input, thus improving the overall user journey.

This is how the webpage for adding a new course looked before the drop down lists (where the user was to free enter anything in the columns):



Now the user gets to choose the courses from a drop-down list rather than having to enter it manually:





We can observe that the process has been made more streamlined by giving the user an option to choose course information from drop-down lists.

18. Rspec tests for Drop Down List Verification

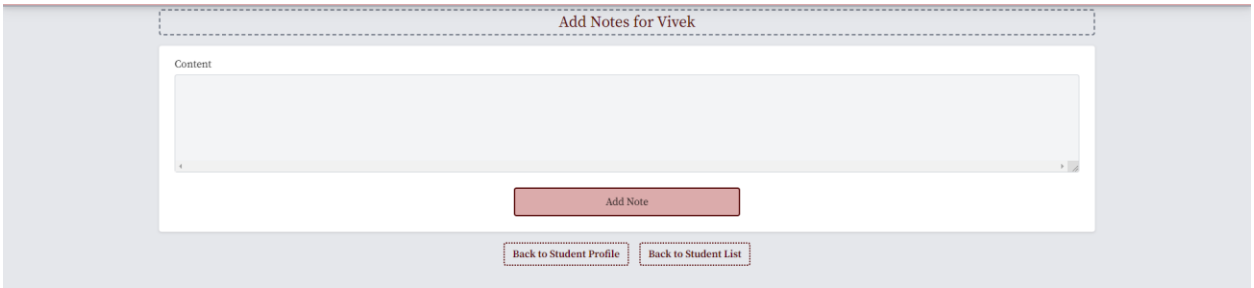
RSpec tests have been added to verify if the course code is being selected from the drop-down list and the information is being submitted correctly for creation of a new course.

19. Adding RSpec tests for validation checks of adding new courses

Integrate RSpec tests to validate checks for adding new courses. These tests aim to ensure robust validation mechanisms for the addition of new courses, covering various scenarios to enhance the reliability and integrity of the course creation process.

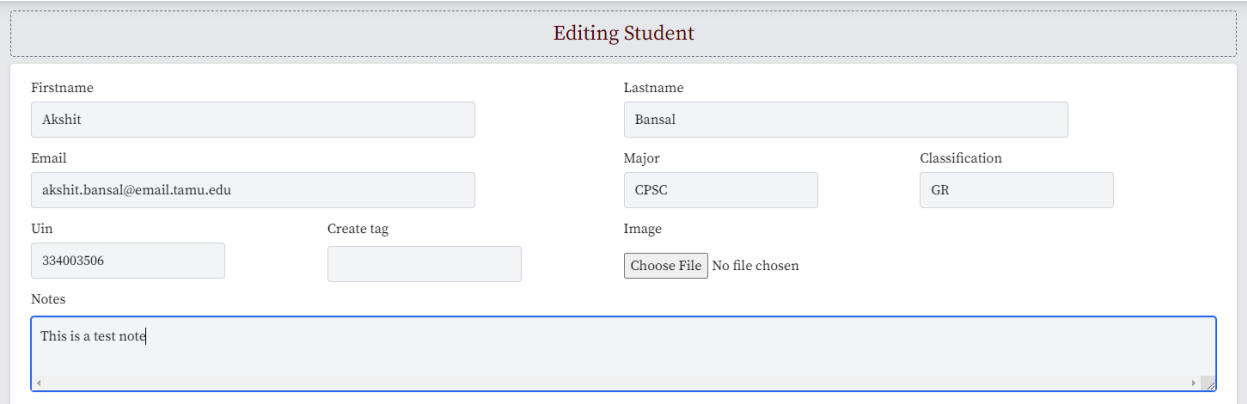
20. Atomic Notes Feature

Updated the Add Note form to match the current design of the web application as shown below.

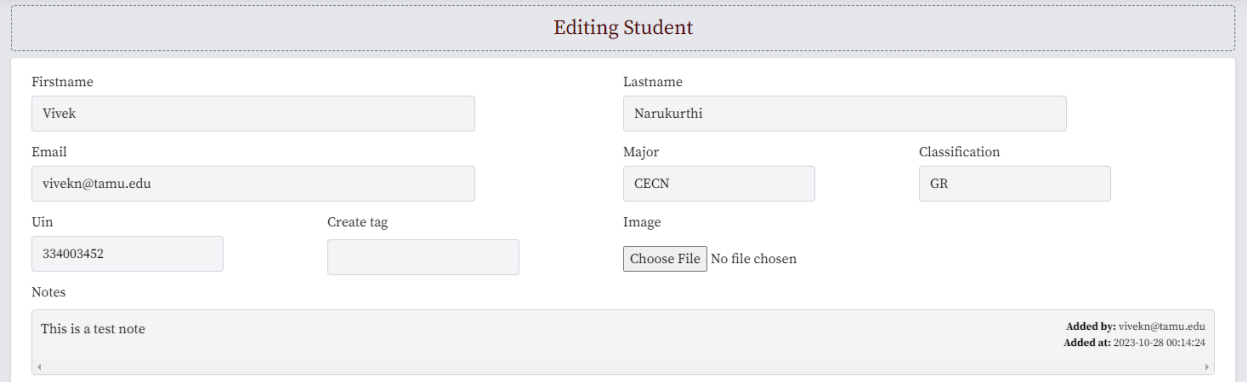


We also have the back to student profile and list buttons below it to go back easily. This time we add notes atomically, everytime we come to this page and click on Add Note it stores the time stamp and also email id of the user who created the note.

Before Screenshot

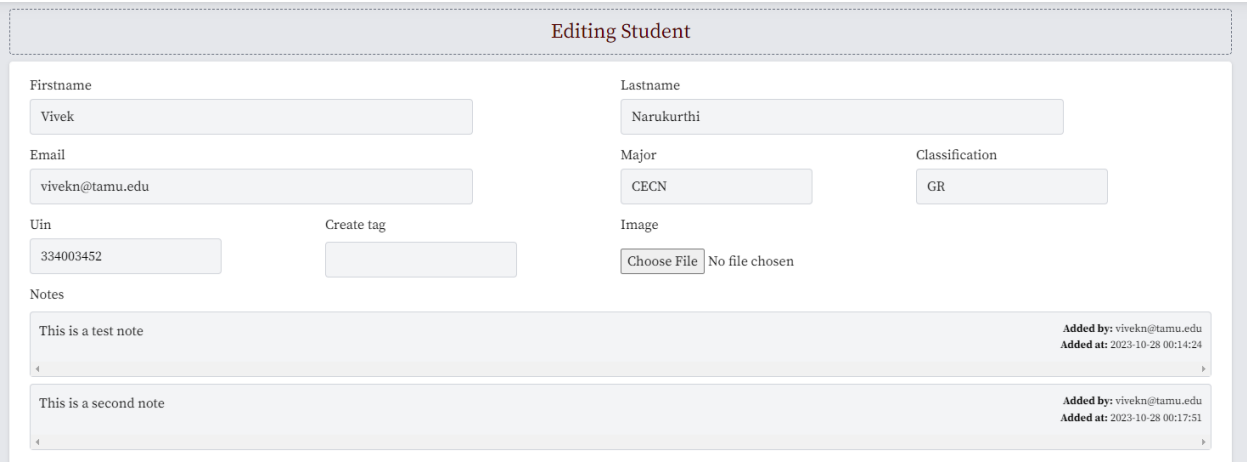


Current Status



When we add multiple notes we get multiple such elements making it atomic in nature and the timestamp makes it easy to remember when we added it

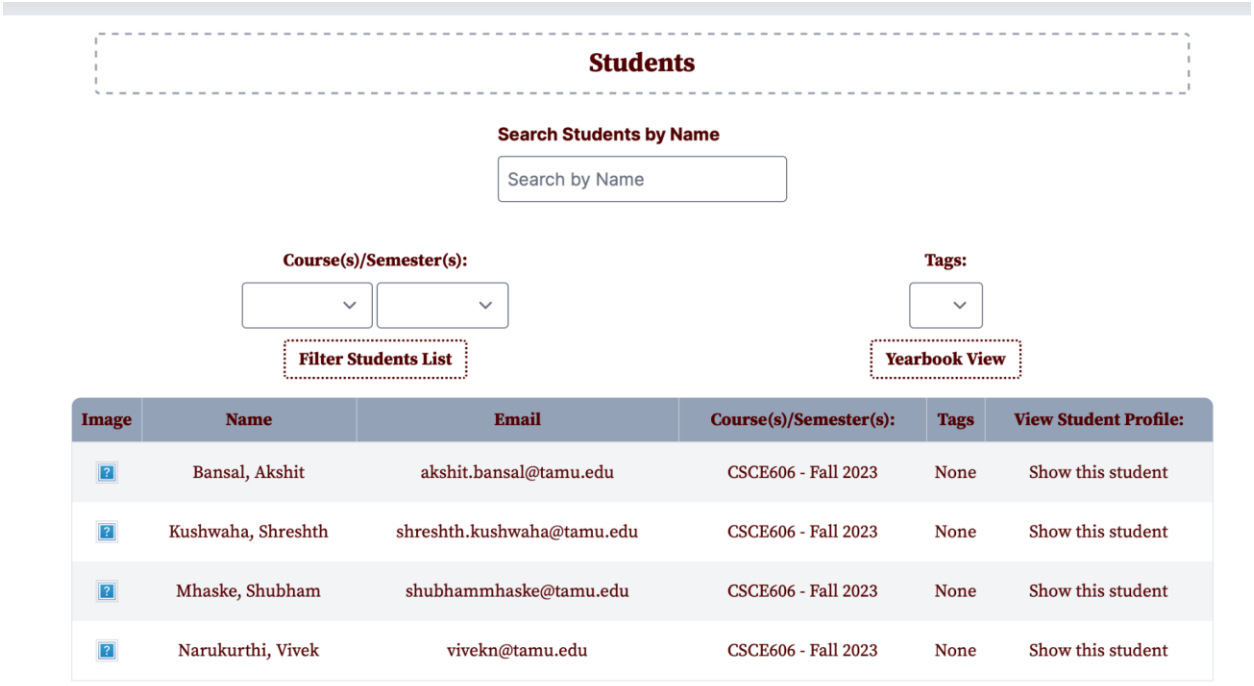
Multiple Notes



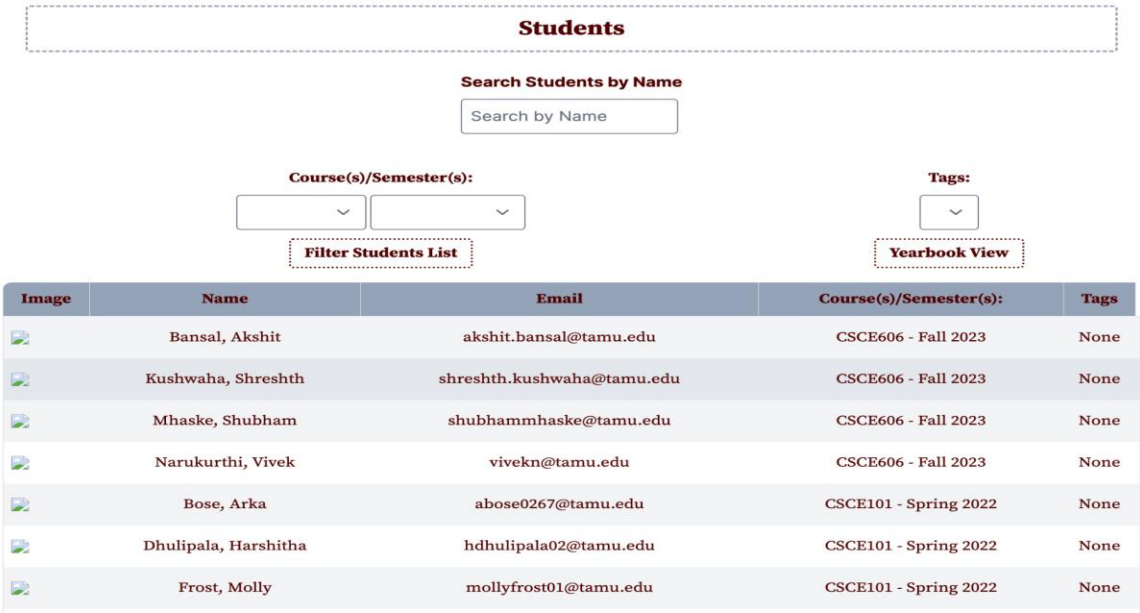
21. Show student profile by clicking on row on student list

Client wants to be able to view individual student profile by simply clicking on the the student row instead of having to click explicitly on the show this student button. Implemented the expected functionality enabling client to open profile without explicit button.

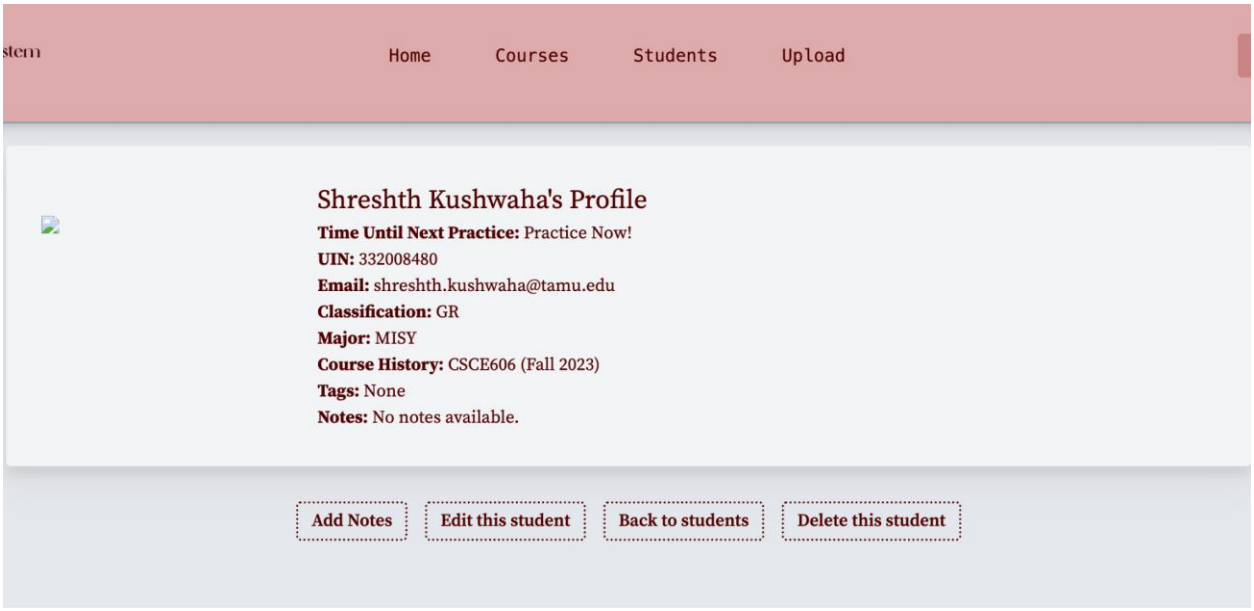
Before:



After:



Able to open student profile by clicking on row -



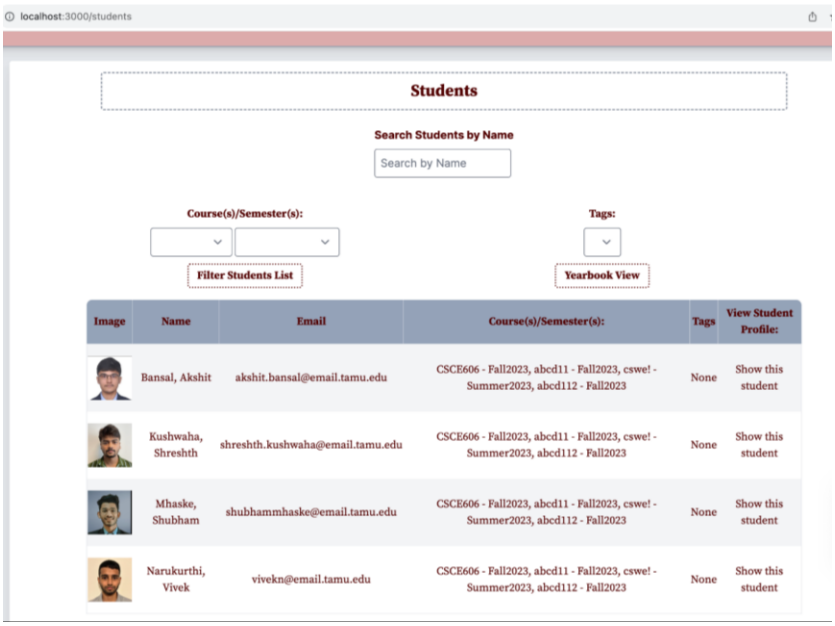
22. Add and Fix cucumber tests - Show student profile by clicking on row on student list

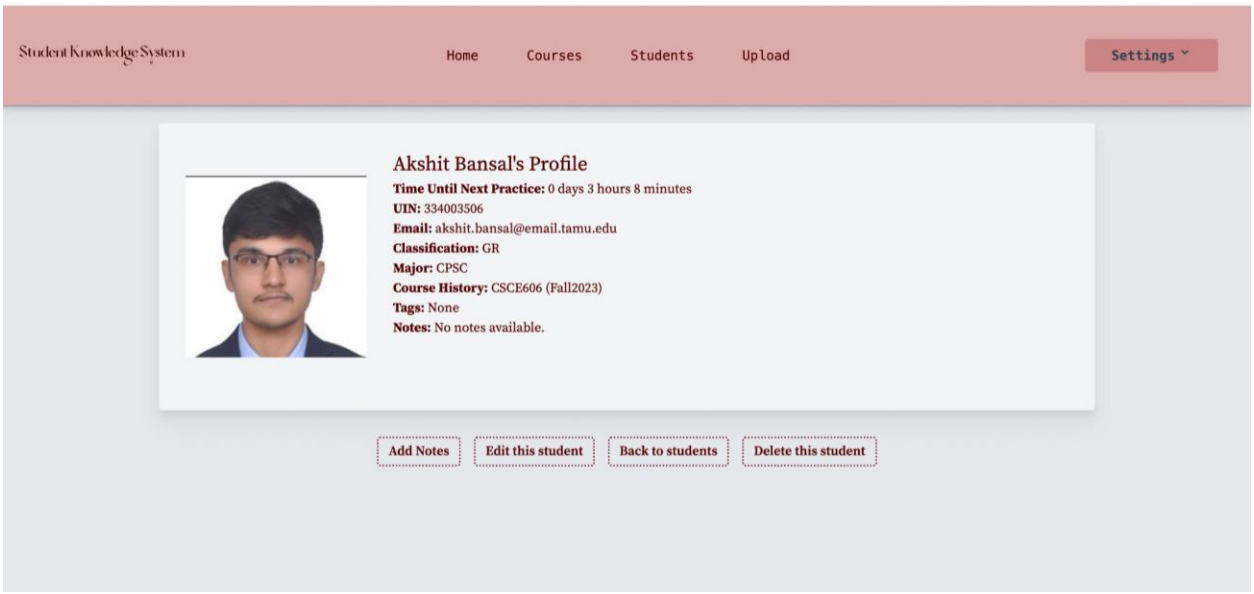
Develop and rectify Cucumber tests for displaying student profiles upon clicking a row in the student list. This task involves adding new tests to cover the feature comprehensively and addressing any existing issues for a seamless and accurate presentation of student profiles during user interaction.

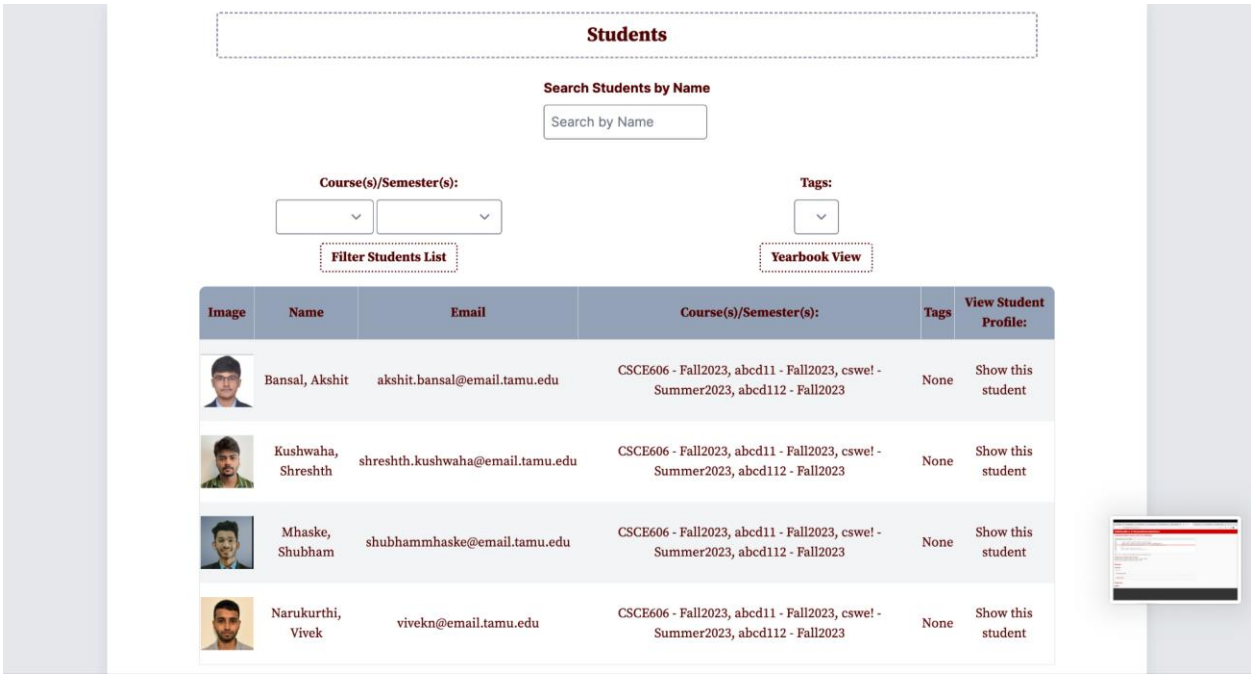
23. Preserving context in "Back to students" button

Client wants us to preserve context when we get on the student detail page and click "Back to students" button. In this iteration, we have corrected the back to students button to only take user where the user came from and preserve any filters/sorting if there.

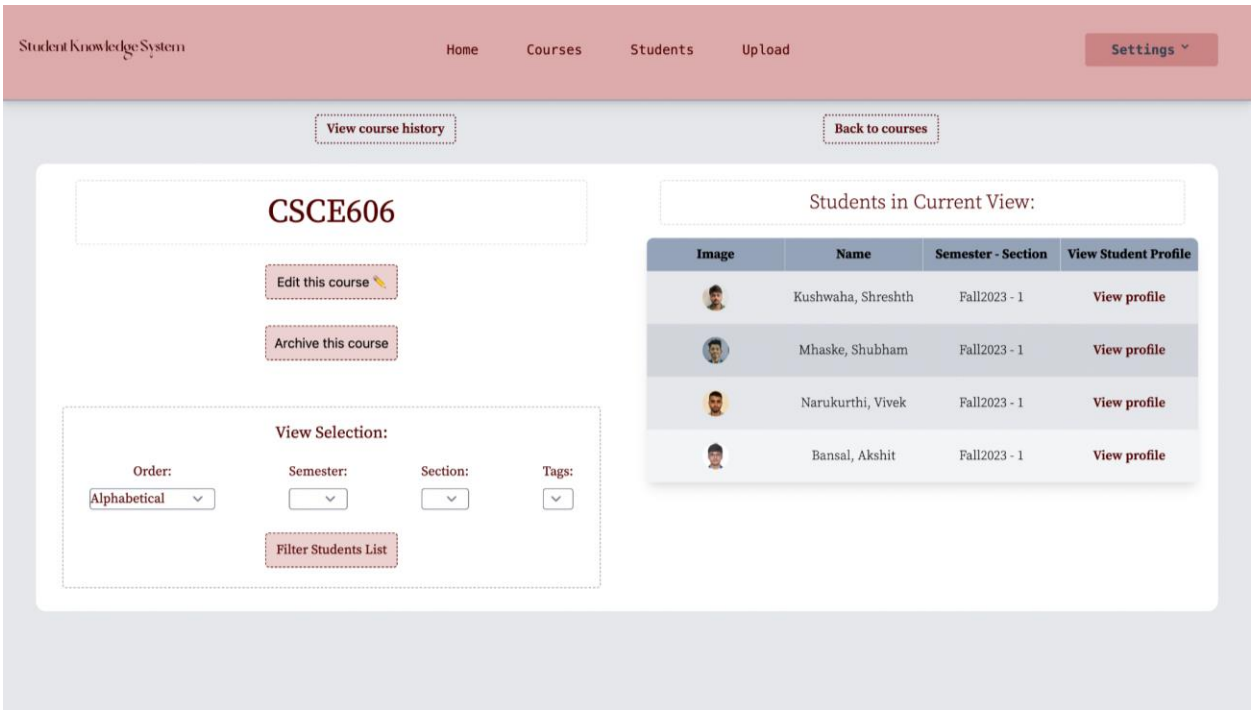
Here are the screenshots:

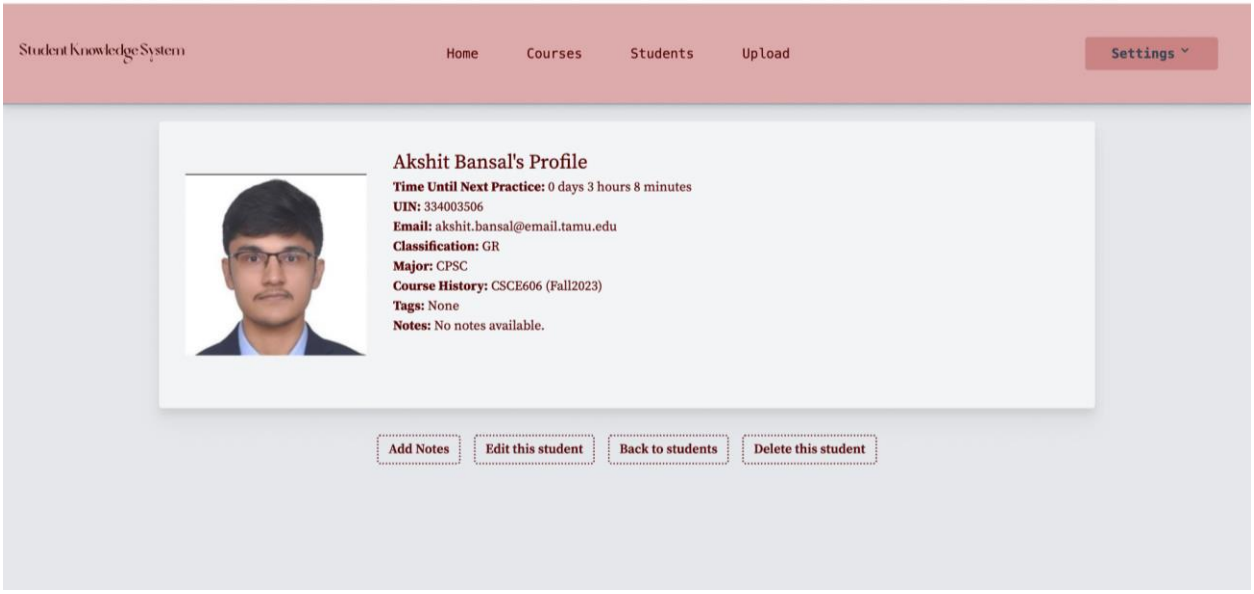

localhost:3000/students
Students
Search Students by Name
Search by Name
Course(s)/Semester(s):
Tags:
Filter Students List
Yearbook View
Image
Name
Email
Course(s)/Semester(s):
Tags
View Student Profile:
Bansal, Akshit
akshit.bansal@email.tamu.edu
CSCE606 - Fall2023, abcd11 - Fall2023, cswe! - Summer2023, abcd112 - Fall2023
None
Show this student
Kushwaha, Shreshth
shreshth.kushwaha@email.tamu.edu
CSCE606 - Fall2023, abcd11 - Fall2023, cswe! - Summer2023, abcd112 - Fall2023
None
Show this student
Mhaske, Shubham
shubhammhaske@email.tamu.edu
CSCE606 - Fall2023, abcd11 - Fall2023, cswe! - Summer2023, abcd112 - Fall2023
None
Show this student
Narukurthi, Vivek
vivekn@email.tamu.edu
CSCE606 - Fall2023, abcd11 - Fall2023, cswe! - Summer2023, abcd112 - Fall2023
None
Show this student

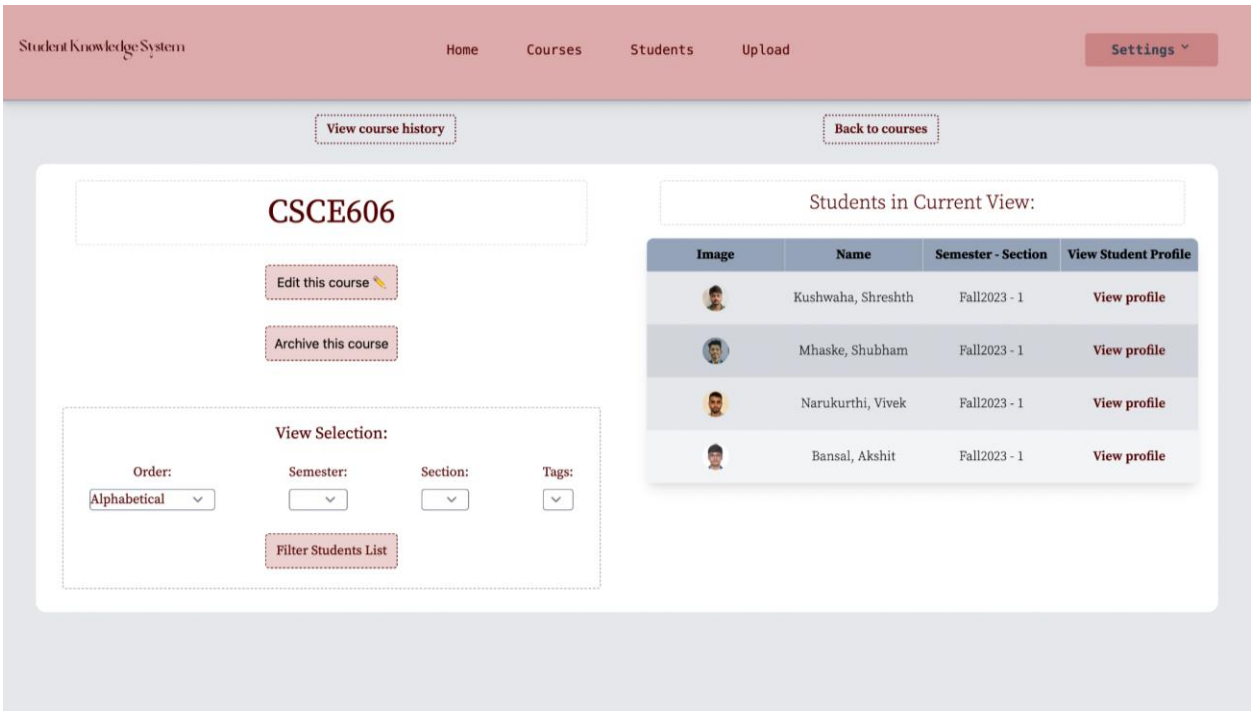

Student Knowledge System
Home
Courses
Students
Upload
Settings
Akshit Bansal's Profile
Time Until Next Practice: 0 days 3 hours 8 minutes
UIN: 334003506
Email: akshit.bansal@email.tamu.edu
Classification: GR
Major: CPSC
Course History: CSCE606 (Fall2023)
Tags: None
Notes: No notes available.
Add Notes
Edit this student
Back to students
Delete this student



So we went back to the all students view along with the filters we applied when we click Back to students button. Now we will try to go to student profile from the course view.







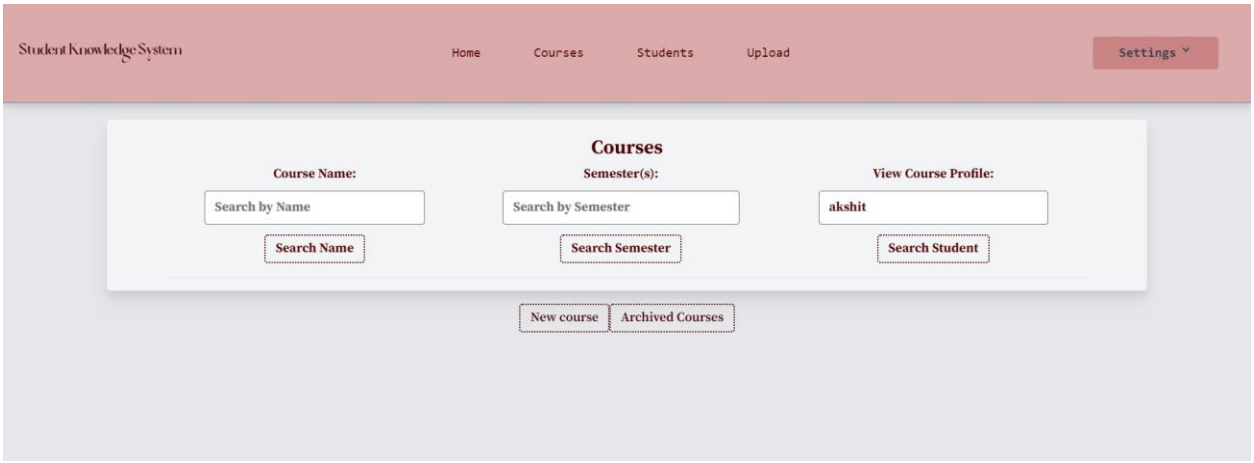
After clicking on "Back to students" button we get back to the course view not the all students view, which was a defect in previous repositories. We also added a check for the null object error while clicking the student detail so that always object is checked if it is not null.

24. Student Search Bar has been made case insensitive

In response to client requirements, an enhancement has been introduced to the system's search functionality for students enrolled in specific courses. Previously, the search process was case-sensitive, resulting in instances where searches for names like "akshit" failed to yield results when the stored data was in a different case, such as "Akshit."

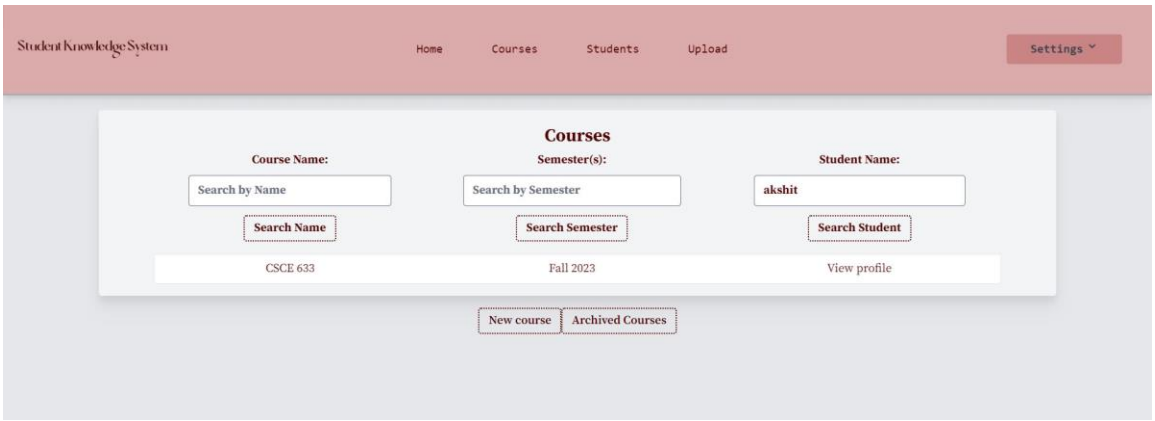
With the implemented changes, the search bar has been rendered case-insensitive. This modification ensures a more user-friendly experience by allowing searches to be conducted without requiring precise case matching. As a result, searching for "akshit" will now yield accurate results, irrespective of the case used in the database records.

Before making the changes:



We can see that no results are showing up for the courses when we type in “akshit”

After making the changes:



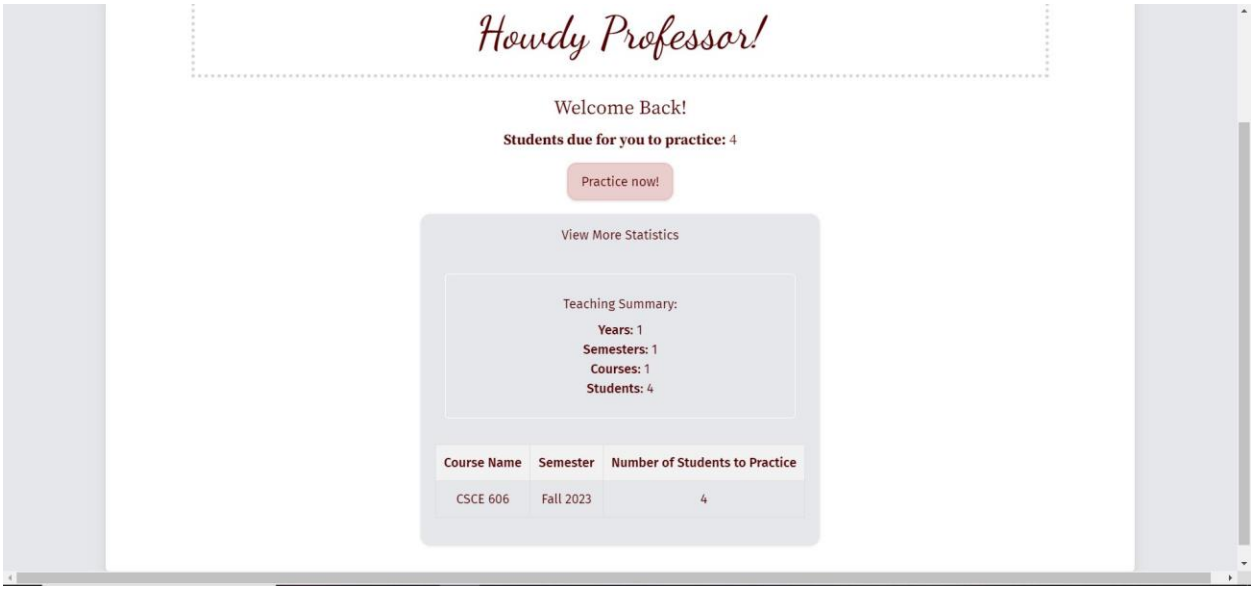
We can now see that even if we type “akshit”, the results for the courses are showing up correctly.

25. The Toggle button for the statistics dashboard has been corrected

Upon client request, a modification was made to the functionality of the button responsible for expanding the statistics dashboard on the homepage. Specifically, the button now dynamically displays "View More Statistics" when the dashboard is not expanded and switches to "View Less Statistics" when the table is expanded and subsequently minimized.

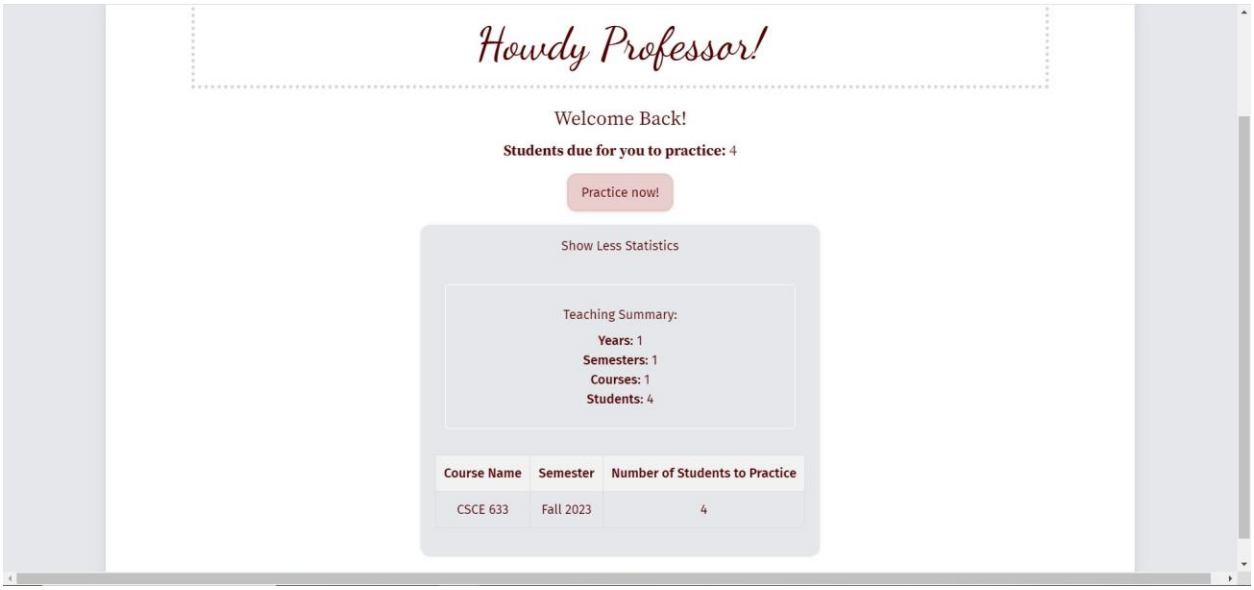
Previously, an inconsistency existed where the dashboard consistently presented the label "View More Statistics" irrespective of its expanded or minimized state. This adjustment ensures a more intuitive and context-aware representation, providing users with accurate and descriptive button labeling based on the current state of the statistics dashboard.

Before making the changes, the dashboard used to look like this:



As we can see, the dashboard is showing “View More Statistics” even when the dashboard is already expanded.

After making the changes, the dashboard now toggles between “View More Statistics” and “Show Less Statistics” button. As we can see, it displays “Show Less Statistics” when the dashboard is expanded.

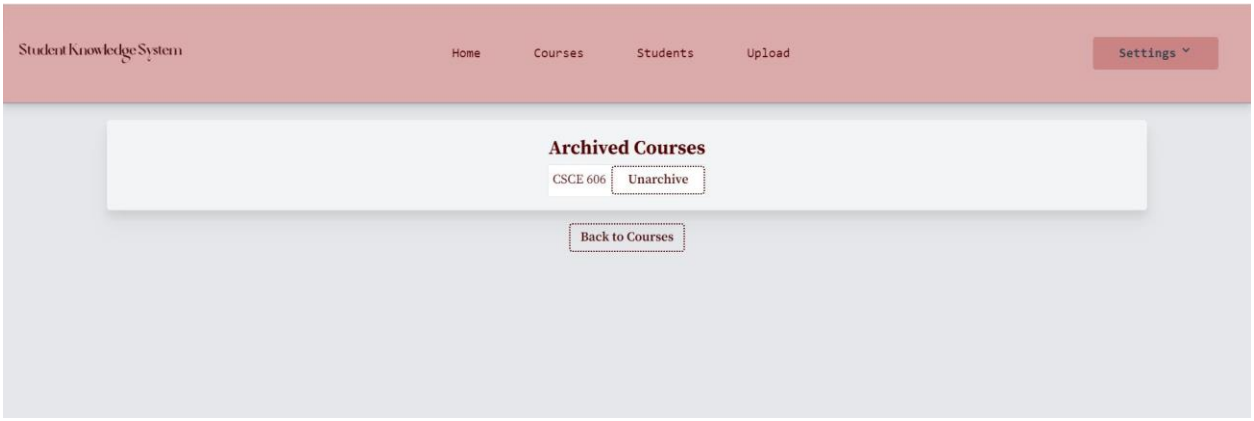


26. User Interface has been modified for the Archive Courses Page

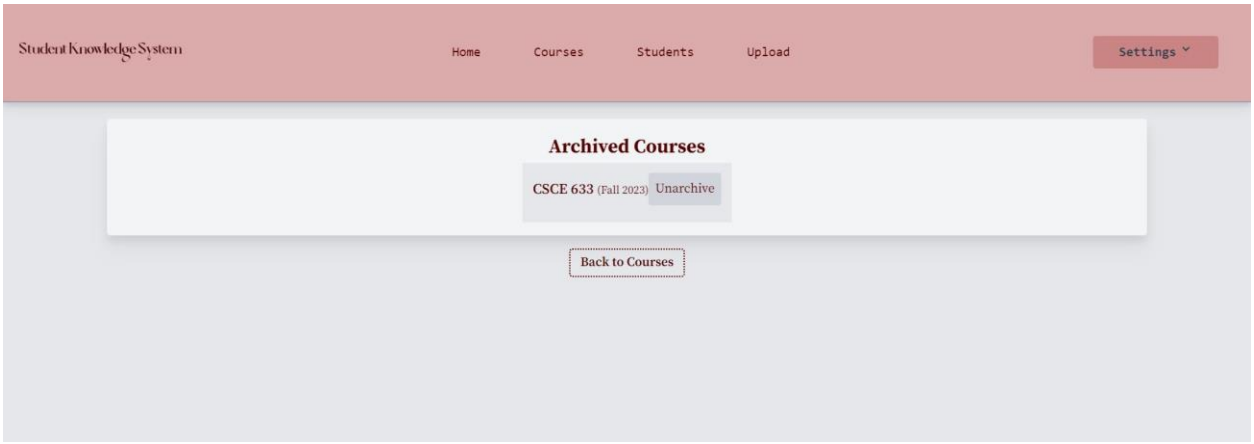
The client identified a need for the Archived Courses page to exhibit both the Semester and Course Code information of the archived courses. The existing presentation only featured the course name, leading to a lack of comprehensive information. Additionally, the page's formatting was deemed disorderly, impeding user navigation and interaction.

In response, modifications were implemented to incorporate the Semester details alongside the Course Code on the Archived Courses page. Simultaneously, user interface enhancements were introduced to rectify the previously identified formatting issues, thereby ensuring a more user-friendly and efficient experience for website visitors.

Before making the changes:



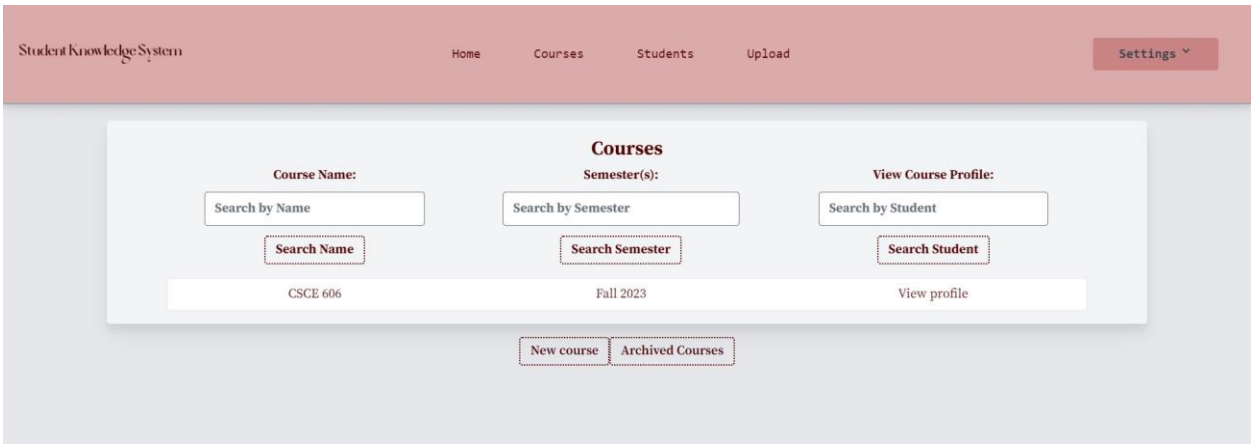
After making the changes, the page displays the archived course's semester and in a much more user-friendly format.



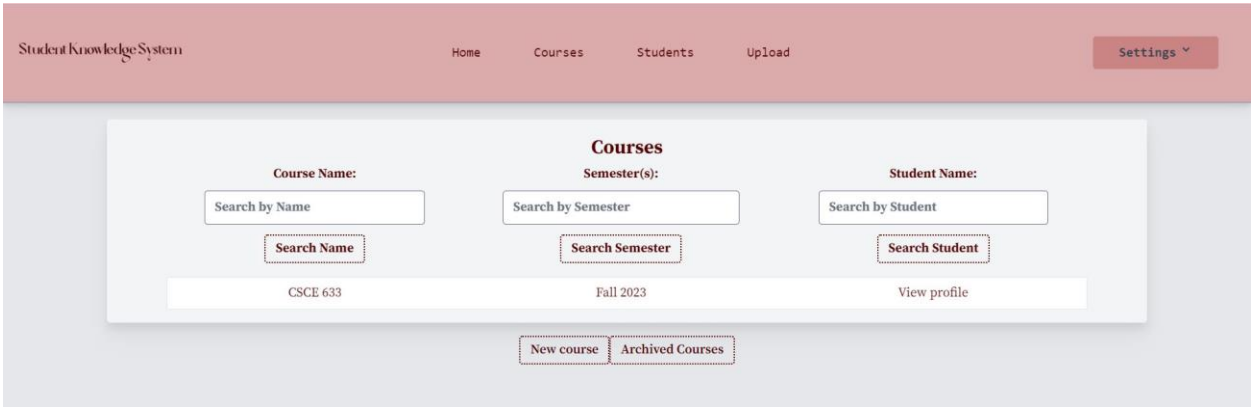
27. User Interface of the Courses Page has been modified

The client expressed a requirement for a refinement of the user interface on the Courses page, citing concerns with the accuracy of labels associated with search bars. The initial labelling was identified as ambiguous, potentially causing user confusion during data entry. Consequently, adjustments were made to the label descriptions to accurately convey the purpose of each respective search bar. This enhancement aims to improve user experience by providing clear and concise guidance, ensuring users can readily discern the intended content for input within the search bars.

Before making the changes, the courses page looked like the following:



After making the changes, the page looks as follows:



28. Pushed changes to production heroku app

Successfully deployed changes to the production Heroku app. This includes the latest updates and enhancements, ensuring that the live application reflects the most recent changes made to the codebase.

29. Made coverage 85% for Rspec

Achieved 85% test coverage for RSpec, enhancing the codebase's robustness and reliability. This milestone ensures a comprehensive suite of tests to validate various scenarios, contributing to the overall quality and stability of the application.

30. Add Sorting in Student based on all columns

Client wanted sorting functionality on the student index page to be able to sort students based on their Name, Email, Courses and Tags.

Implemented the sorting functionality so that users will be able to sort ascending and descending by clicking on respective columns.

Sorting Ascending with Names:

Click on column header to sort

Image	Name	Email	Course(s)/Semester(s):	Tags
	Bansal, Akshit	akshit.bansal@tamu.edu	CSCE606 - Fall 2023	None
	Bose, Arka	abose0267@tamu.edu	CSCE406 - Spring 2022	None
	Dhulipala, Harshitha	hdhulipala02@tamu.edu	CSCE406 - Spring 2022	None
	Frost, Molly	mollyfrost01@tamu.edu	CSCE406 - Spring 2022	None
	Imwalle, Andrew	andrew.imwalle@tamu.edu	CSCE406 - Spring 2022	None
	Ketkar, Anuj	anujketkar@tamu.edu	CSCE406 - Spring 2022	None

Sorting descending based on Email:

Click on column header to sort

Image	Name	Email	Course(s)/Semester(s):	Tags
	Narukurthi, Vivek	vivekn@tamu.edu	CSCE606 - Fall 2023	None
	Sarkar, Uma	umasarkar@tamu.edu	CSCE406 - Spring 2022	None
	Tran, Tuong	t2tran@tamu.edu	CSCE406 - Spring 2022	None
	Sotelo, Sarah Eilene	slsotelo@tamu.edu	CSCE406 - Spring 2022	None
	Mhaske, Shubham	shubhammhaske@tamu.edu	CSCE606 - Fall 2023	None
	Kushwaha, Shreshth	shreshth.kushwaha@tamu.edu	CSCE606 - Fall 2023	None

Sorting ascending based on Courses:

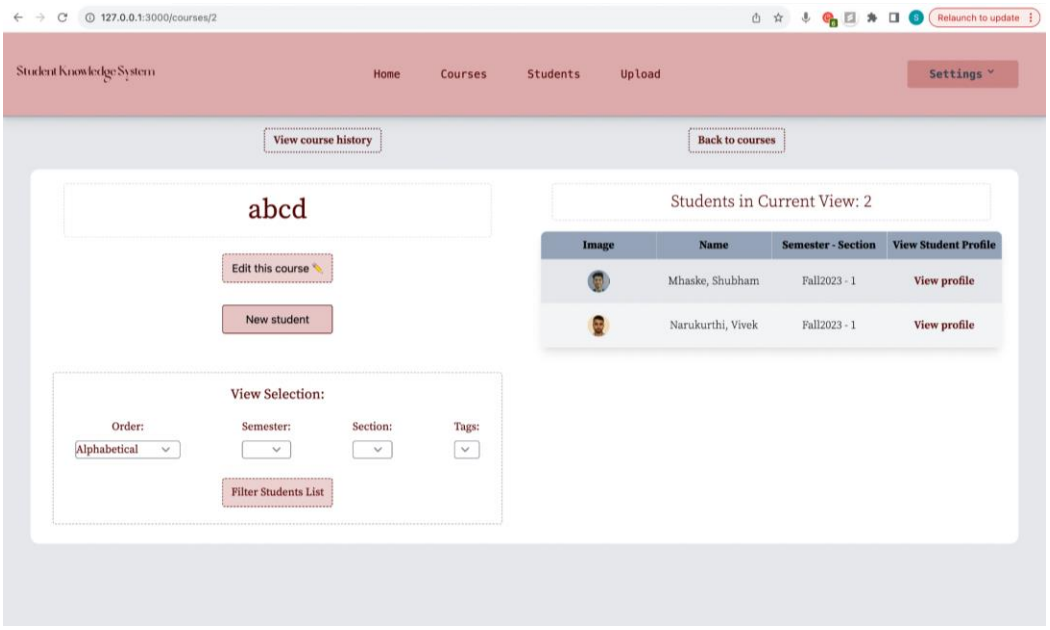
Click on column header to sort

Image	Name	Email	Course(s)/Semester(s):	Tags
	Sarkar, Uma	umasarkar@tamu.edu	CSCE406 - Spring 2022	None
	Tran, Tuong	t2tran@tamu.edu	CSCE406 - Spring 2022	None
	Sotelo, Sarah Eilene	slsotelo@tamu.edu	CSCE406 - Spring 2022	None
	Oliphant, Caleb	oliphcal000@tamu.edu	CSCE406 - Spring 2022	None
	McDonough, Nick	nimcd@tamu.edu	CSCE406 - Spring 2022	None
	Manuel, Neha	nehaam02@tamu.edu	CSCE406 - Spring 2022	None

31. Improve test coverage for Students Controller

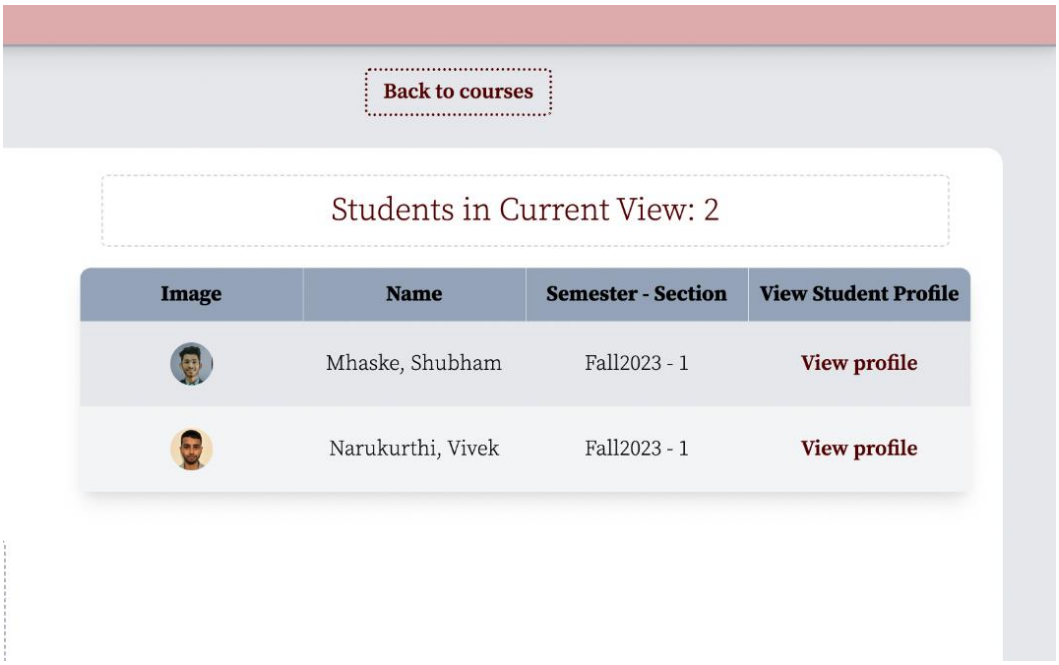
Increase test coverage for the Students Controller to enhance the reliability and thoroughness of the testing suite. This improvement involves writing additional tests to cover various functionalities and scenarios within the controller, ensuring comprehensive validation of the Students Controller's behavior.

32. New Student button on course page



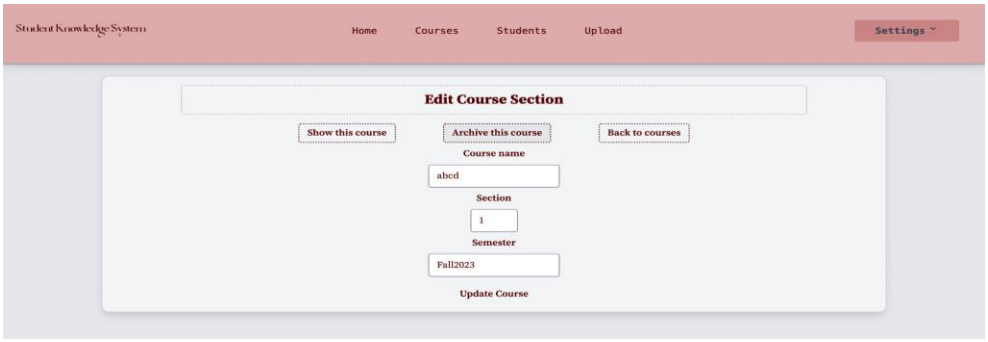
Previously the new student button was only on the students section but now it is present on the course page too so that the instructor can add a new student from there itself.

33. Added student count on the course page



With help of this count instructors can see total number of students present in the current view, ie: total number of students in the course. This detail can help them to know what course is most important with respect to number of students and which course they ned to focus on more.

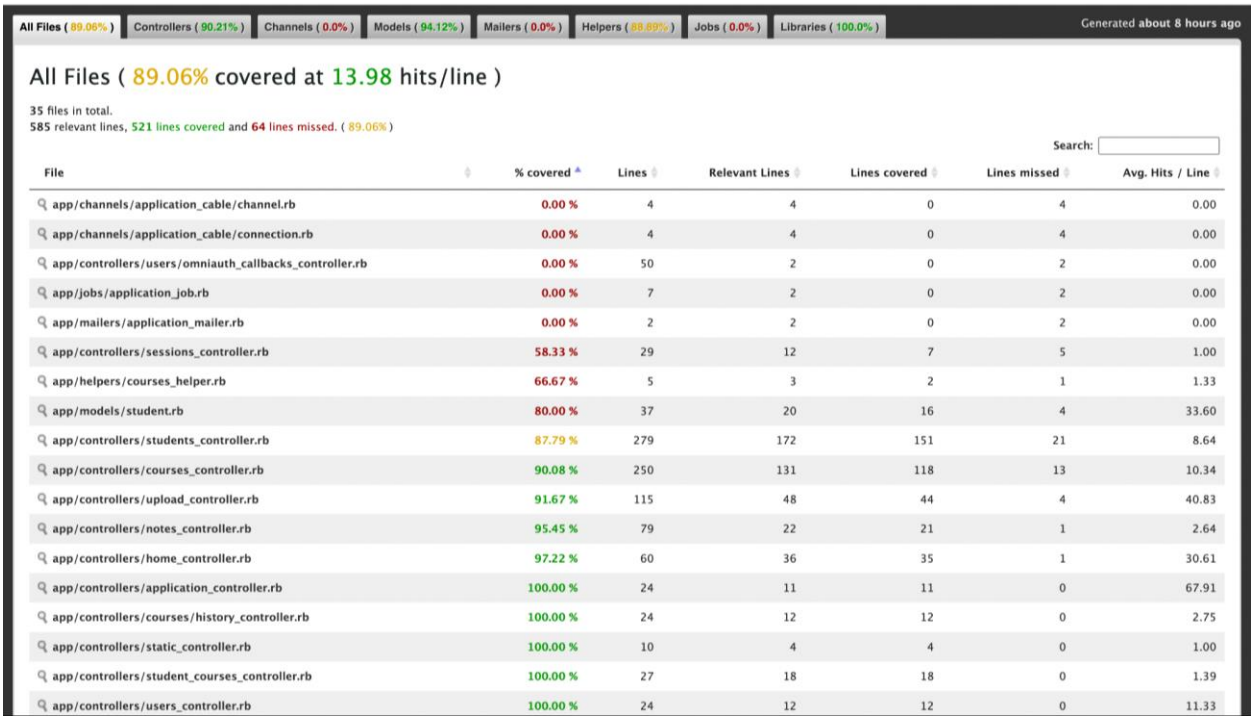
34. Moved Archive button to edit student page with improved style



Moving this button to edit the course makes more sense as this would prevent accidental archiving while viewing a course. Also the user interface looks better when the button is in the edit page.

35. Refactored code for better coverage

Some methods in some controllers were arbitrarily defined with unnecessary if else and other conditional by previous teams. This iteration we identified them and commented on those lines of code which resulted in better coverage.

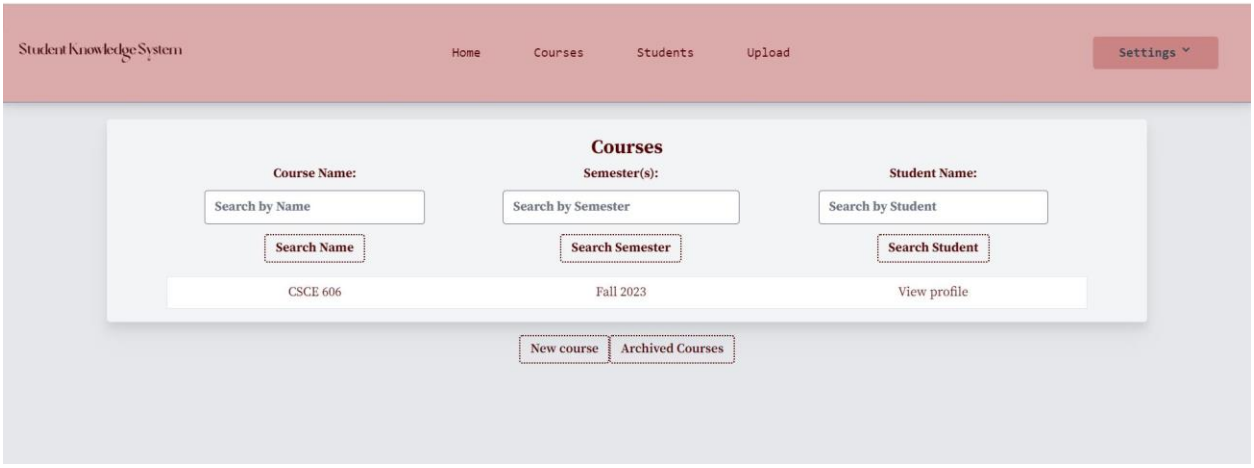


File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/channels/application_cable/channel.rb	0.00 %	4	4	0	4	0.00
app/channels/application_cable/connection.rb	0.00 %	4	4	0	4	0.00
app/controllers/users/omniauth_callbacks_controller.rb	0.00 %	50	2	0	2	0.00
app/jobs/application_job.rb	0.00 %	7	2	0	2	0.00
app/mailers/application_mailer.rb	0.00 %	2	2	0	2	0.00
app/controllers/sessions_controller.rb	58.33 %	29	12	7	5	1.00
app/helpers/courses_helper.rb	66.67 %	5	3	2	1	1.33
app/models/student.rb	80.00 %	37	20	16	4	33.60
app/controllers/students_controller.rb	87.79 %	279	172	151	21	8.64
app/controllers/courses_controller.rb	90.08 %	250	131	118	13	10.34
app/controllers/upload_controller.rb	91.67 %	115	48	44	4	40.83
app/controllers/notes_controller.rb	95.45 %	79	22	21	1	2.64
app/controllers/home_controller.rb	97.22 %	60	36	35	1	30.61
app/controllers/application_controller.rb	100.00 %	24	11	11	0	67.91
app/controllers/courses/history_controller.rb	100.00 %	24	12	12	0	2.75
app/controllers/static_controller.rb	100.00 %	10	4	4	0	1.00
app/controllers/student_courses_controller.rb	100.00 %	27	18	18	0	1.39
app/controllers/users_controller.rb	100.00 %	24	12	12	0	11.33

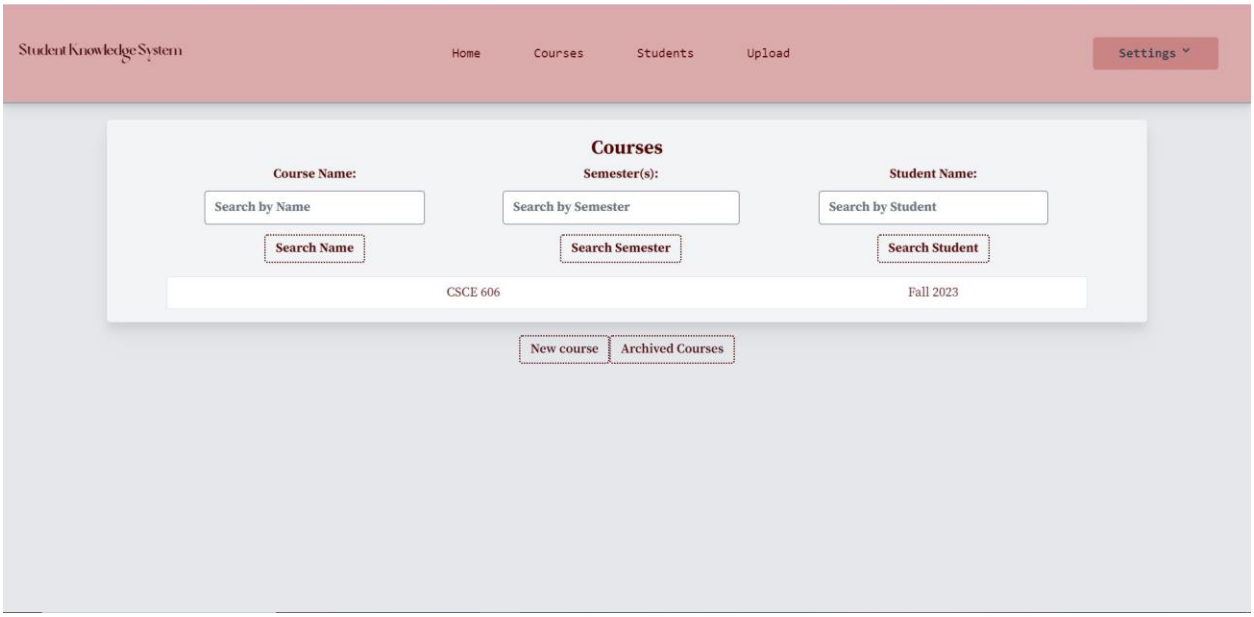
36. Rows on Courses page have been made clickable for all the courses

The courses that were visible on the Courses page could only be opened by clicking the "View Profile" button. The client wanted that the user should be able to view the course information by clicking anywhere on the tab of a course. Hence the tabs have been made clickable and the "View Profile" button has been removed.

Before:



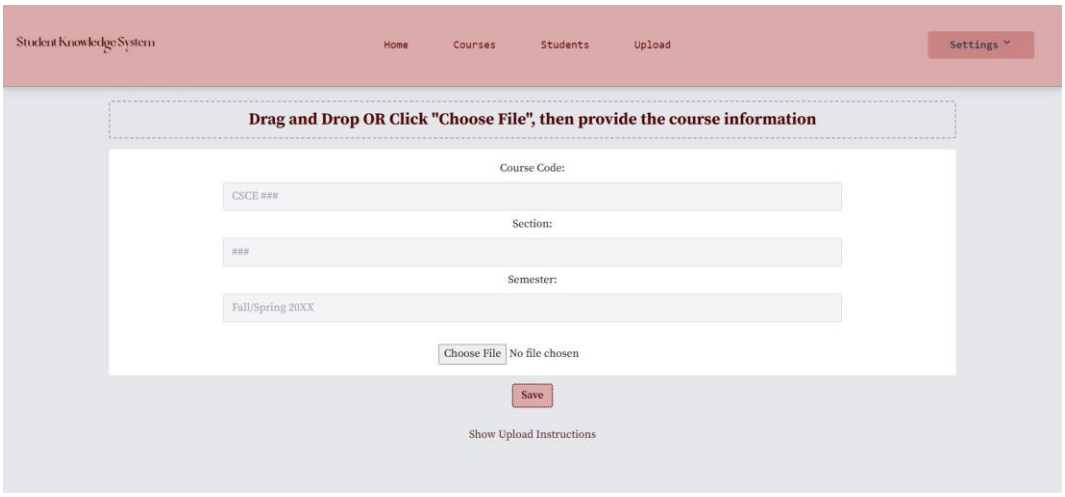
After:

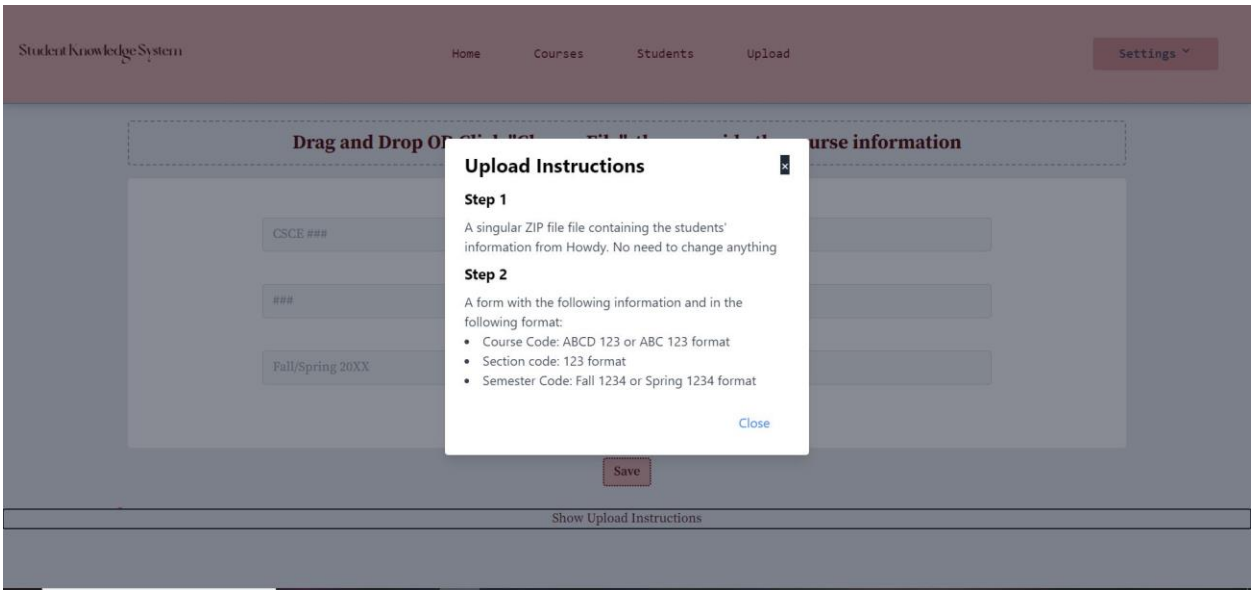


37. Updated the New Course Page with information for adding courses

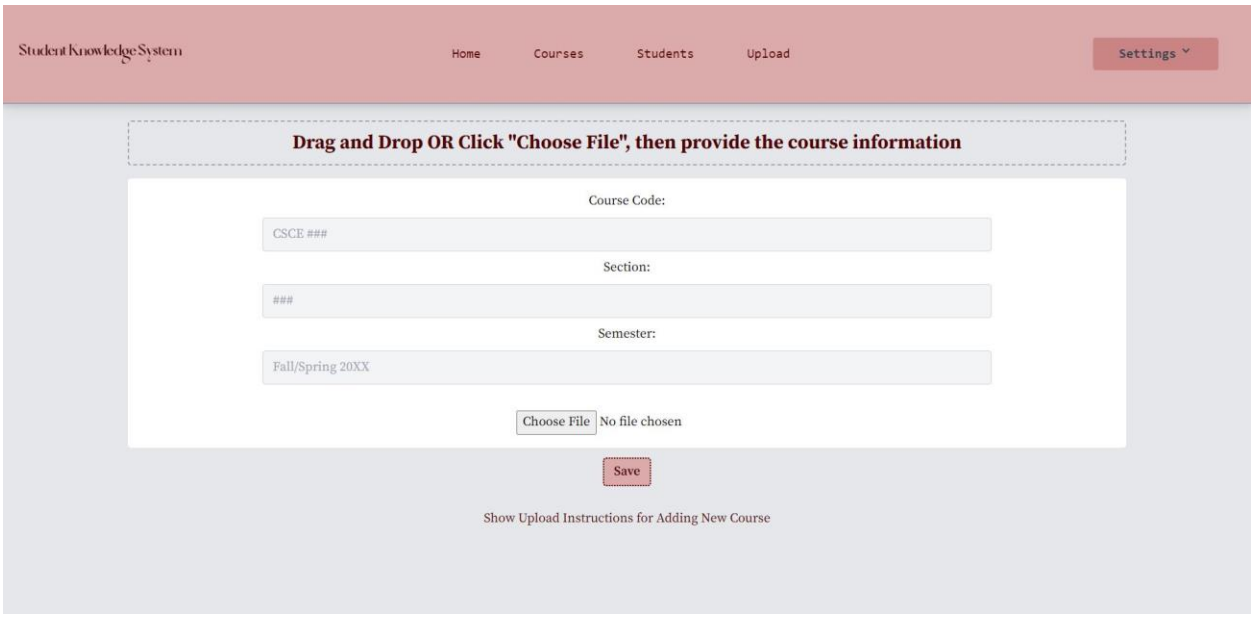
When adding a new course to the website, the user has to follow certain steps which include filling out a form with the information and uploading a zip file with the information of the course. Earlier the instructions were quite unclear making it difficult for the user to interpret what to do. Hence, according to the client's requirements, we have modified the instructions and made changes to the layout of the button to view instructions.

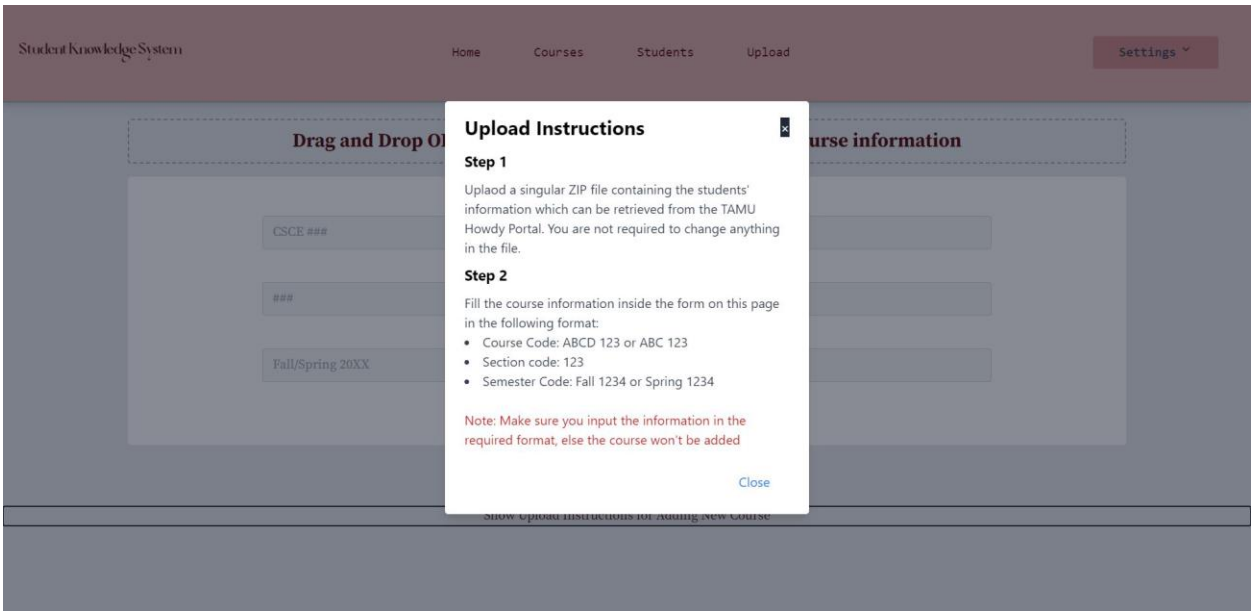
Before:





After:

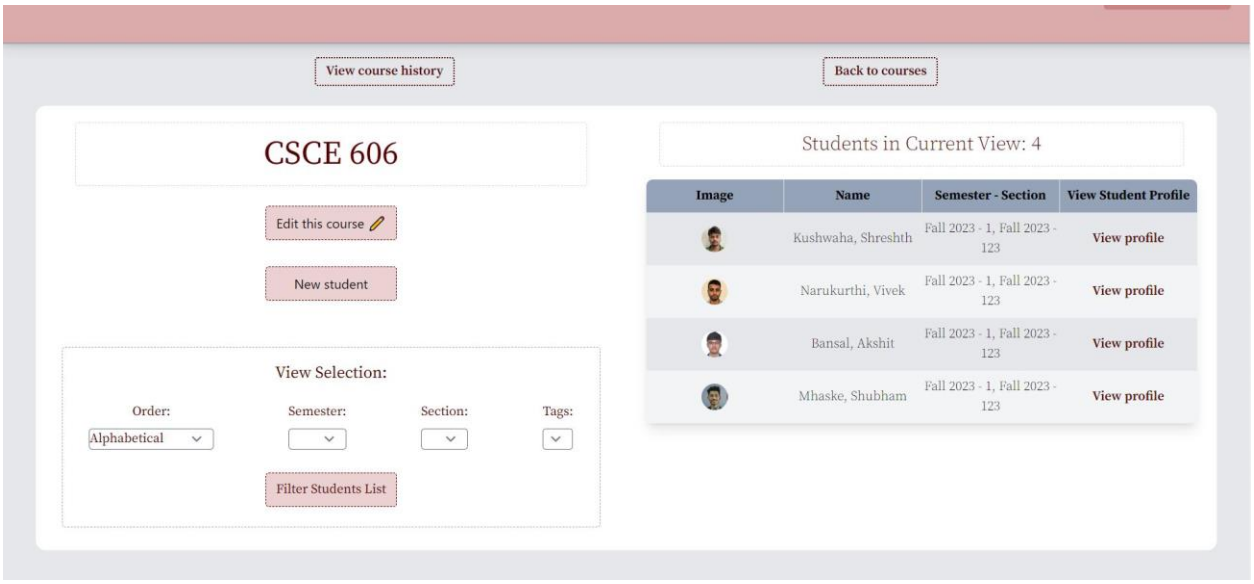




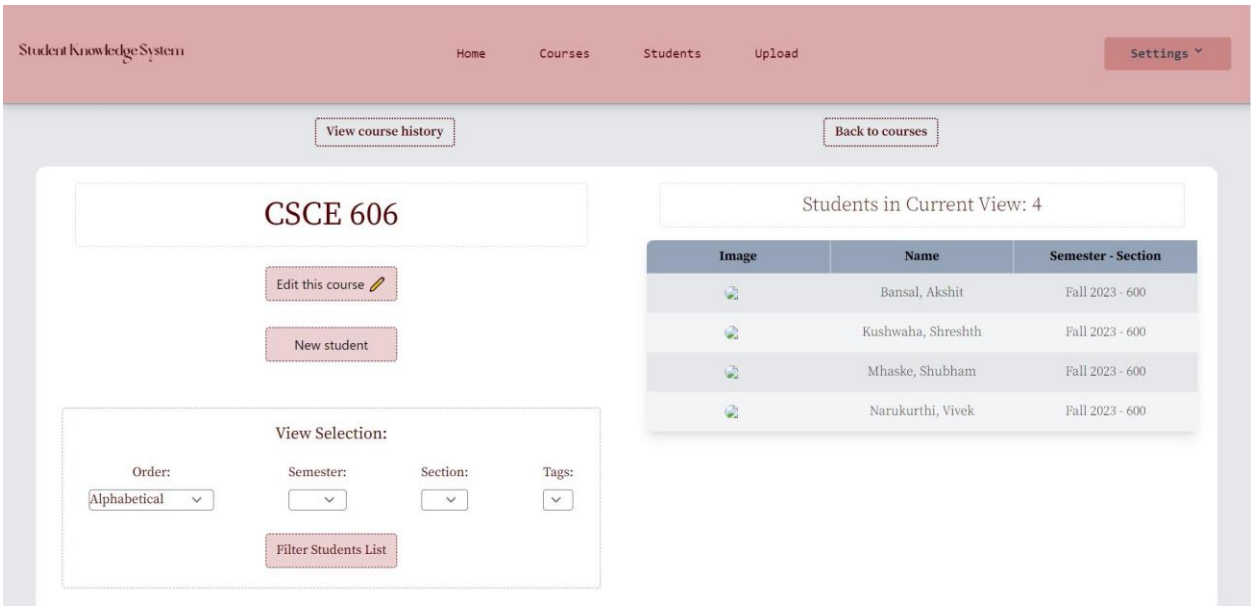
38. Made the Students in Current View redirect to Student Profiles on clicking

On the course information page, the table displaying the students had a link to view the information about the students. The student info could only be displayed on clicking the View Profile button. According to the client's request, we have made the whole tab to be clickable to view the student information, redirecting the user to the student's information page. Also, the view Student Profile column has been removed from the table.

Before:



After:

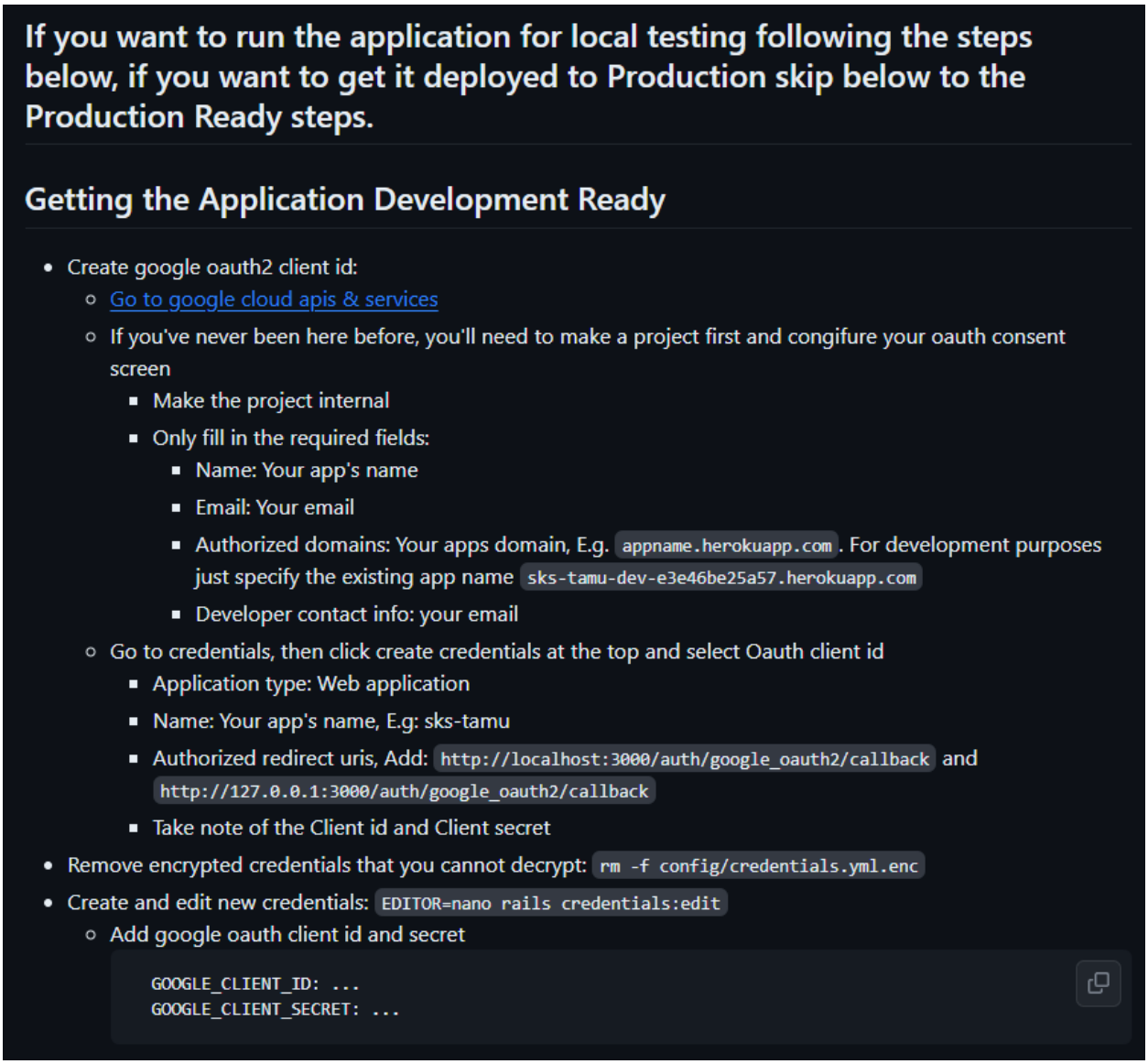


39. Added rspec tests for notes feature

Rspec tests were added for the above mentioned features to ensure coverage isn't affected. Around 14/16 lines were covered with the tests written which was above the excepted standard of 90%.

40. Updated readme and instructions for project deployment

The readme was updated to reflect the correct dependencies needed to get the project working, also on top of that we added steps to run it in local environment and production as well as shown below.



If you want to run the application for local testing following the steps below, if you want to get it deployed to Production skip below to the Production Ready steps.

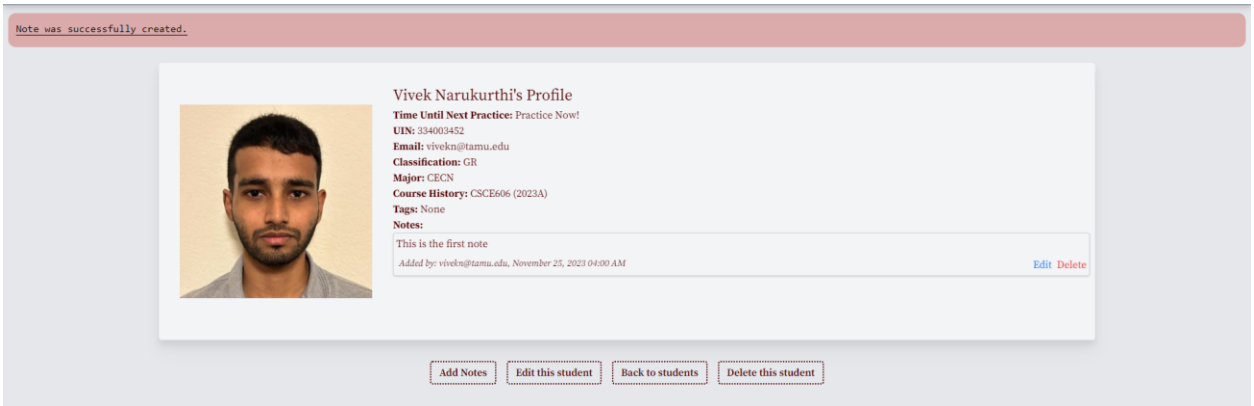
Getting the Application Development Ready

- Create google oauth2 client id:
  - Go to google cloud apis & services
  - If you've never been here before, you'll need to make a project first and congifure your oauth consent screen
    - Make the project internal
    - Only fill in the required fields:
      - Name: Your app's name
      - Email: Your email
      - Authorized domains: Your apps domain, E.g. `appname.herokuapp.com`. For development purposes just specify the existing app name `sks-tamu-dev-e3e46be25a57.herokuapp.com`
      - Developer contact info: your email
  - Go to credentials, then click create credentials at the top and select Oauth client id
    - Application type: Web application
    - Name: Your app's name, E.g: sks-tamu
    - Authorized redirect uris, Add: `http://localhost:3000/auth/google_oauth2/callback` and `http://127.0.0.1:3000/auth/google_oauth2/callback`
    - Take note of the Client id and Client secret
- Remove encrypted credentials that you cannot decrypt: `rm -f config/credentials.yml.enc`
- Create and edit new credentials: `EDITOR=nano rails credentials:edit`
  - Add google oauth client id and secret

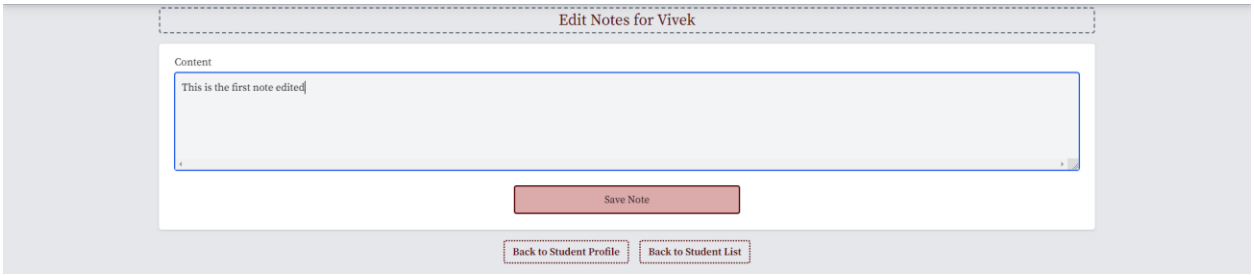
```
GOOGLE_CLIENT_ID: ...
GOOGLE_CLIENT_SECRET: ...
```

41. Making notes Atomic

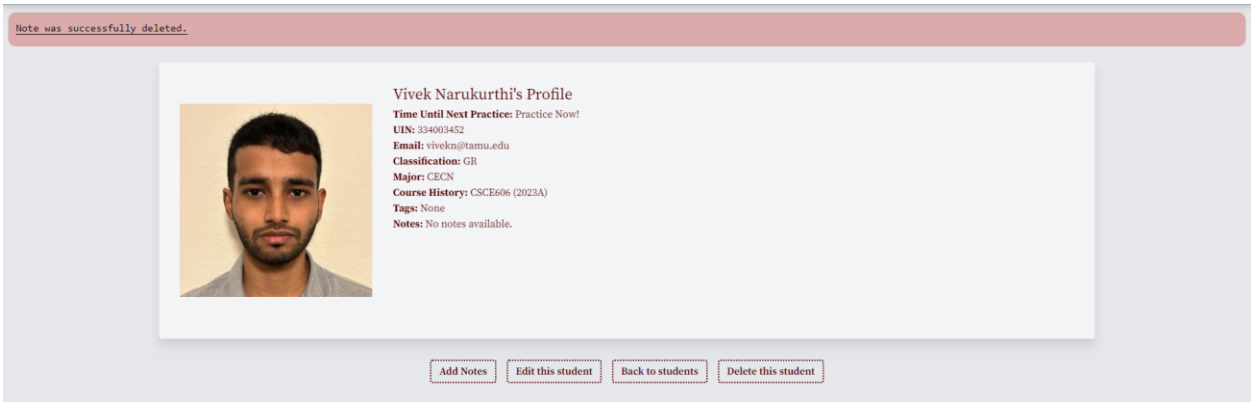
The notes feature has been made atomic by including timestamp and user who made the note, also as requested by the client we have added an edit and delete button for each note as opposed to having a common place to edit the notes



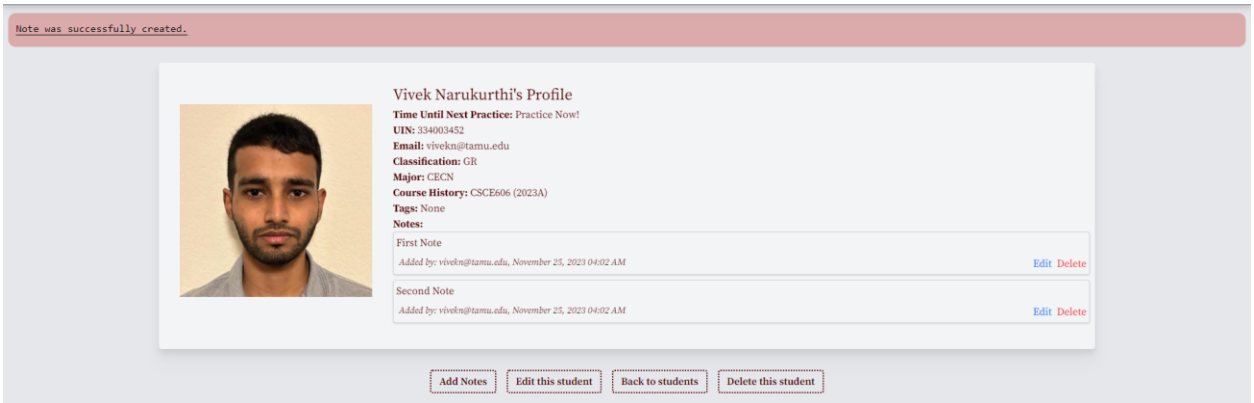
Edit opens a form which lets us edit the existing note as shown below



Delete removes the note altogether and gives a message on successful deletion



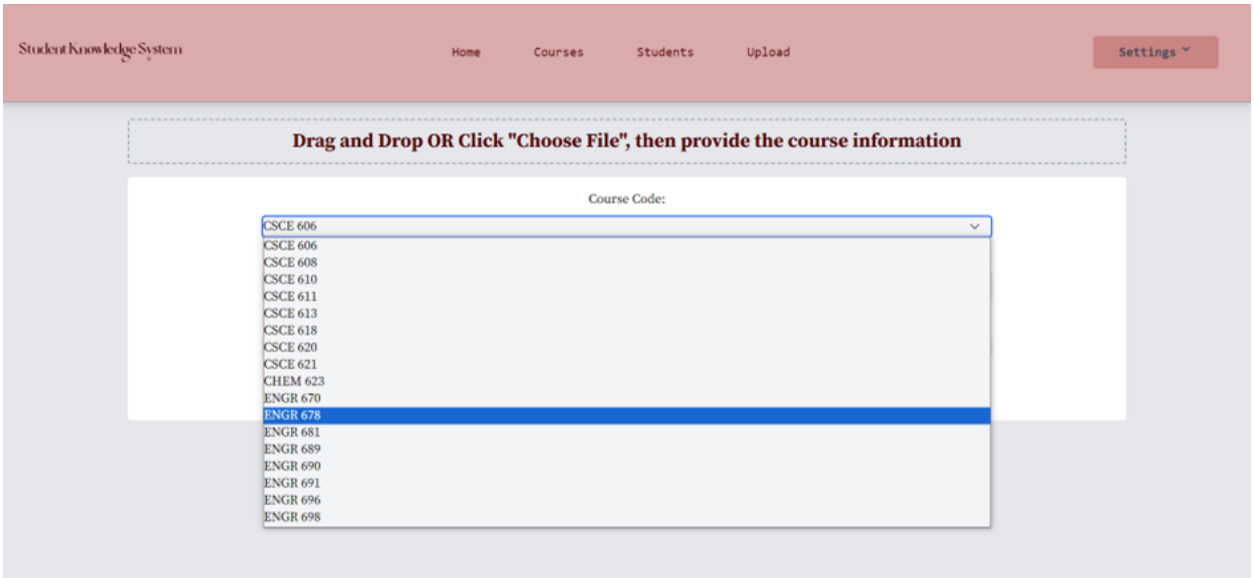
When we have multiple notes we have separate cards for each note to differentiate between them as shown below

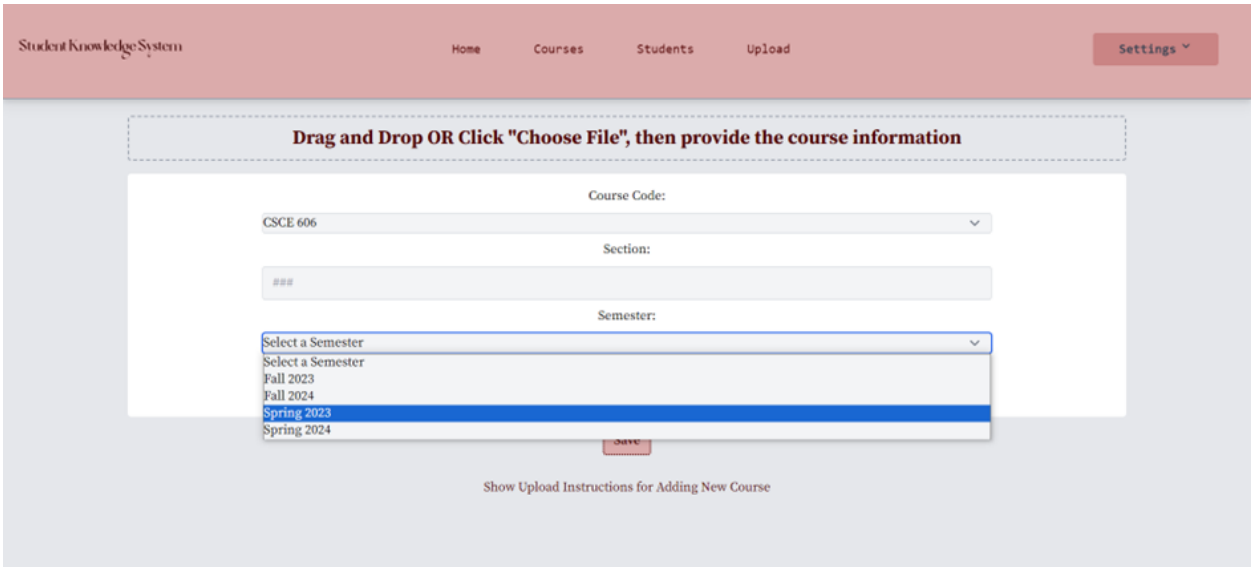


Revised the Drop Down Lists for Adding New Courses

Drop down lists have been added with additional courses on the New courses page to make it easier for the user to choose courses and to fill them in the correct format. This creates a uniform pattern for all the courses making it easier to understand and view for the professors. The addition of drop-down lists on the New Courses page represents a significant enhancement to the user experience, providing a streamlined and user-friendly interface for selecting courses. From the perspective of professors and administrative staff, the use of drop-down lists streamlines the data entry process. This efficiency not only saves time but also reduces the likelihood of discrepancies or misinterpretations of course information.

How the page looks after adding the changes:

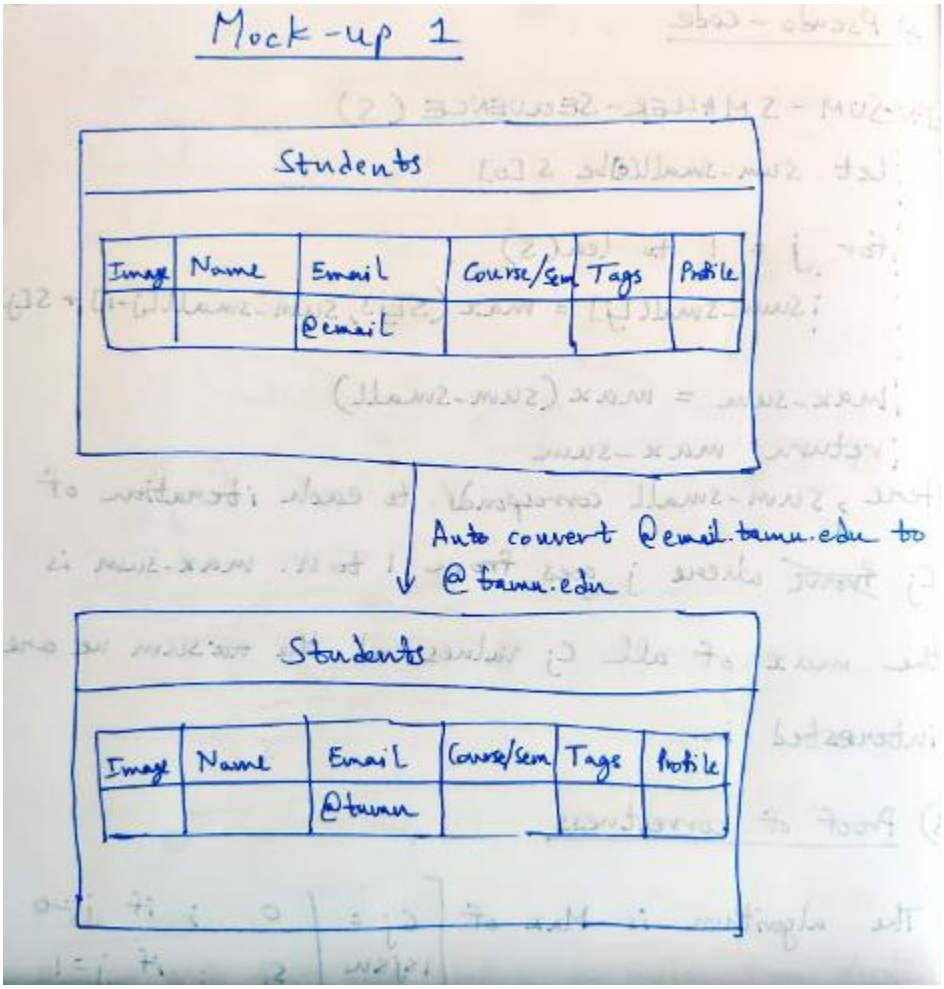




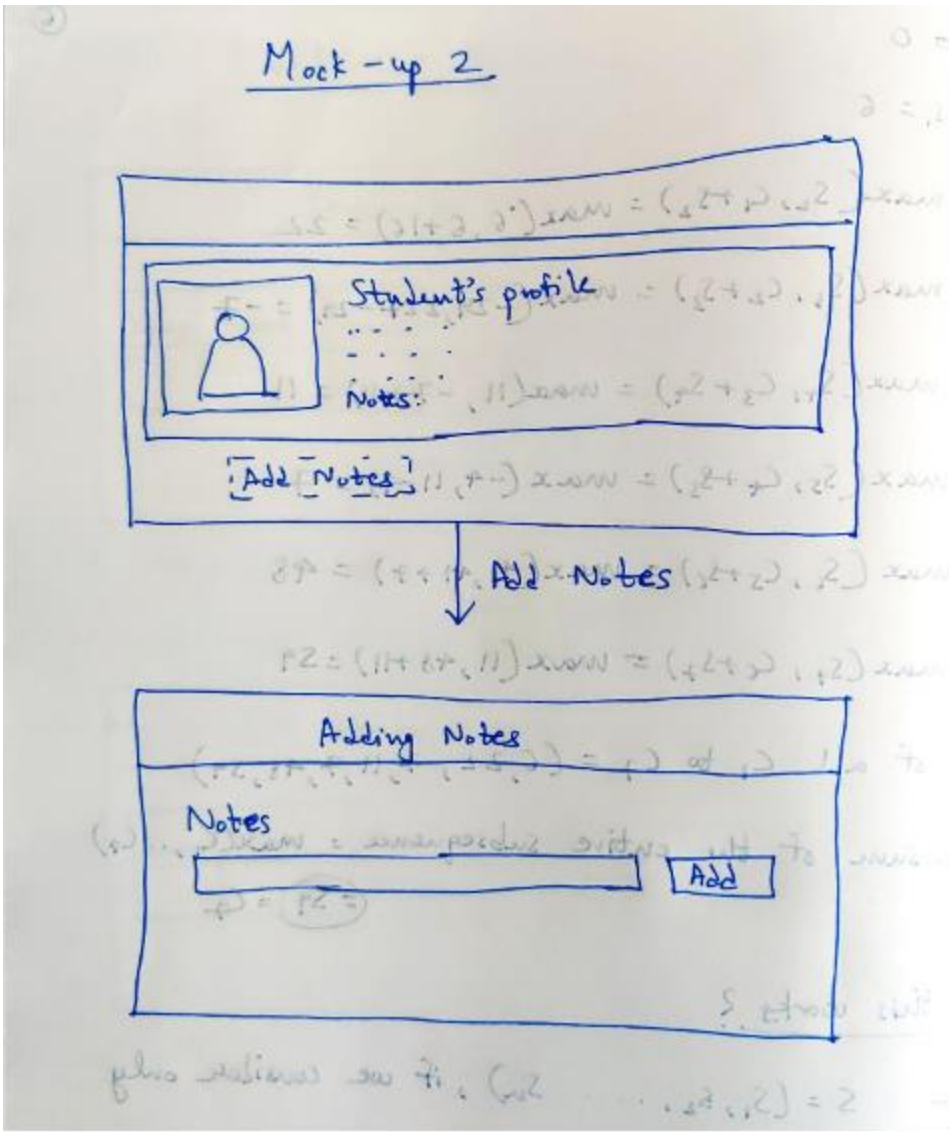
Unfinished User Stories:

- Improving the space repetition algorithm for the practice game
- Displaying error message when the user enters info about new course in incorrect format
- Improving coverage from 90% to 95%
- Adding new student in a course with that particular course
- TA role
- Course-section specific gameplay
- Adding site admin role
- Private - shared notes

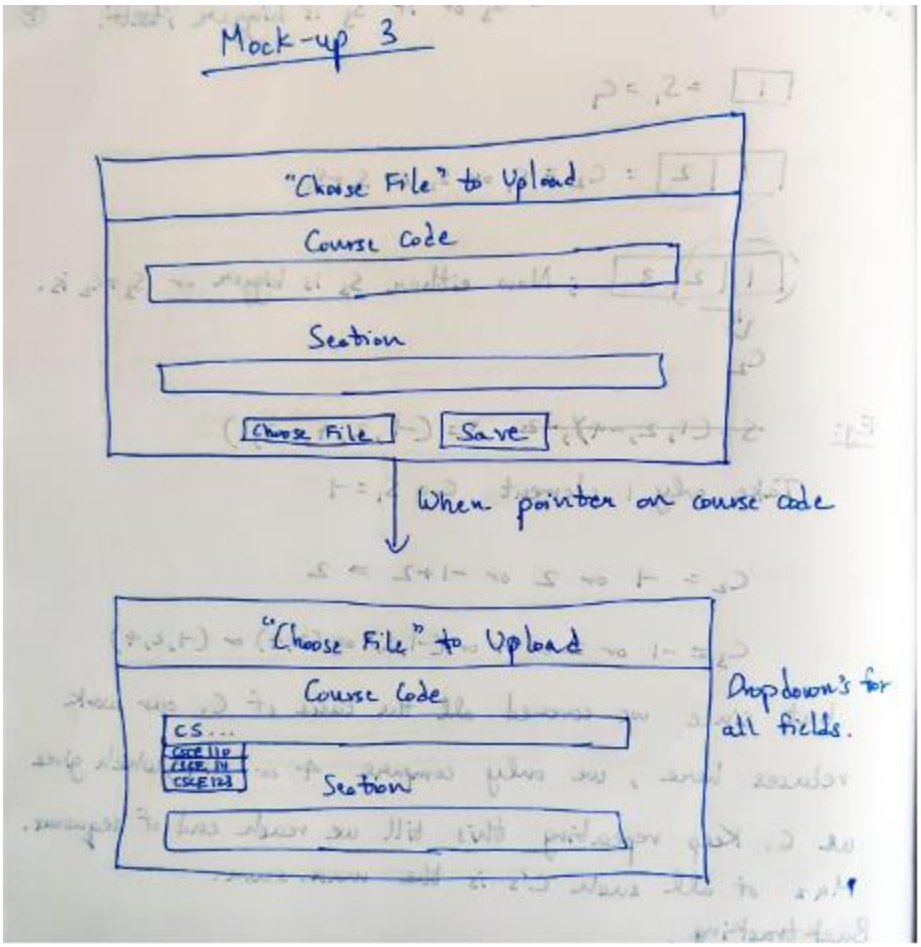
UI Mockups/Storyboards



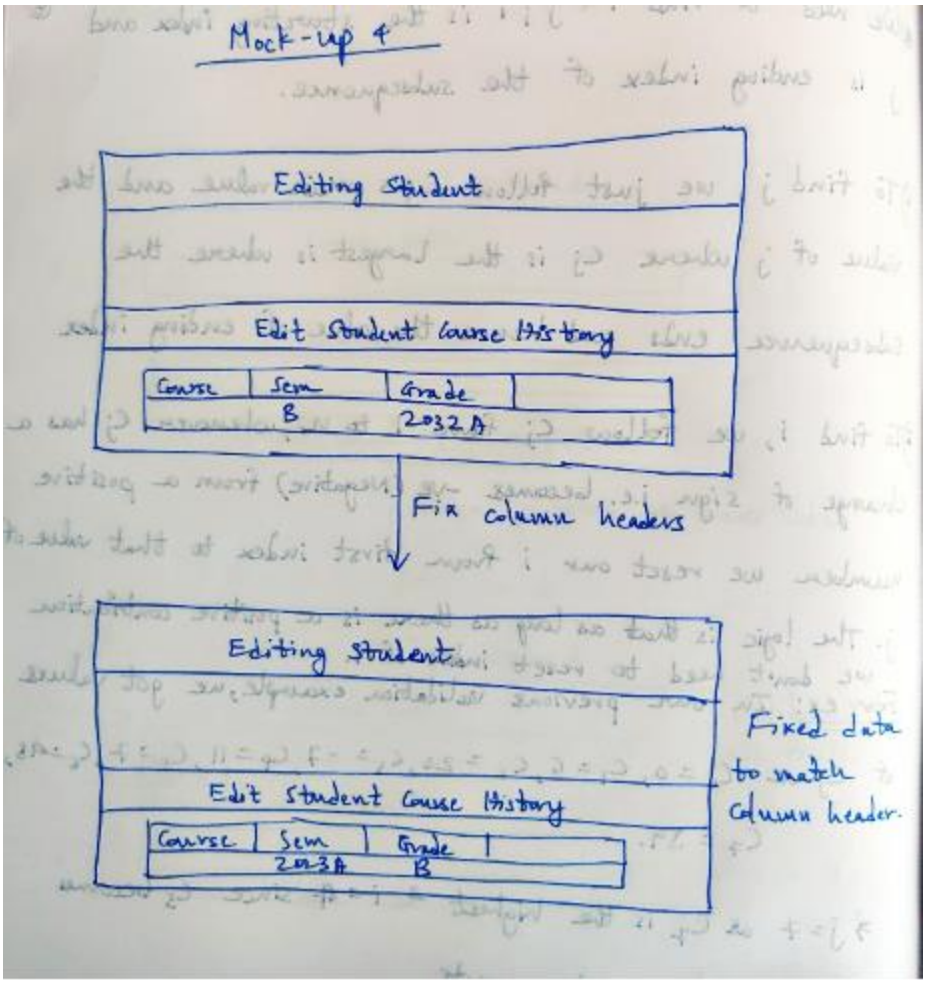
This explains the flow of how the students email address should appear on the students profile page.



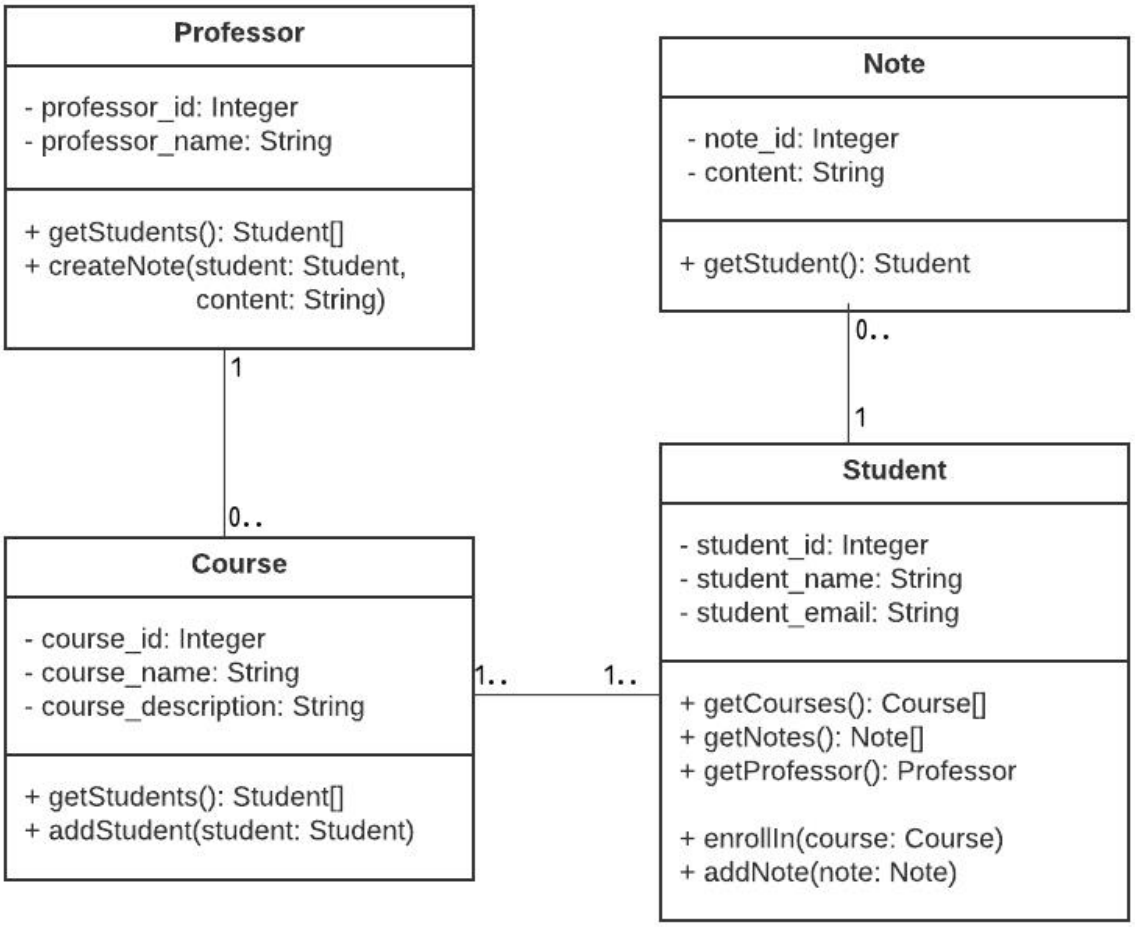
This a high level mockup of the notes feature which should have functionality to add,delete and edit notes individually.



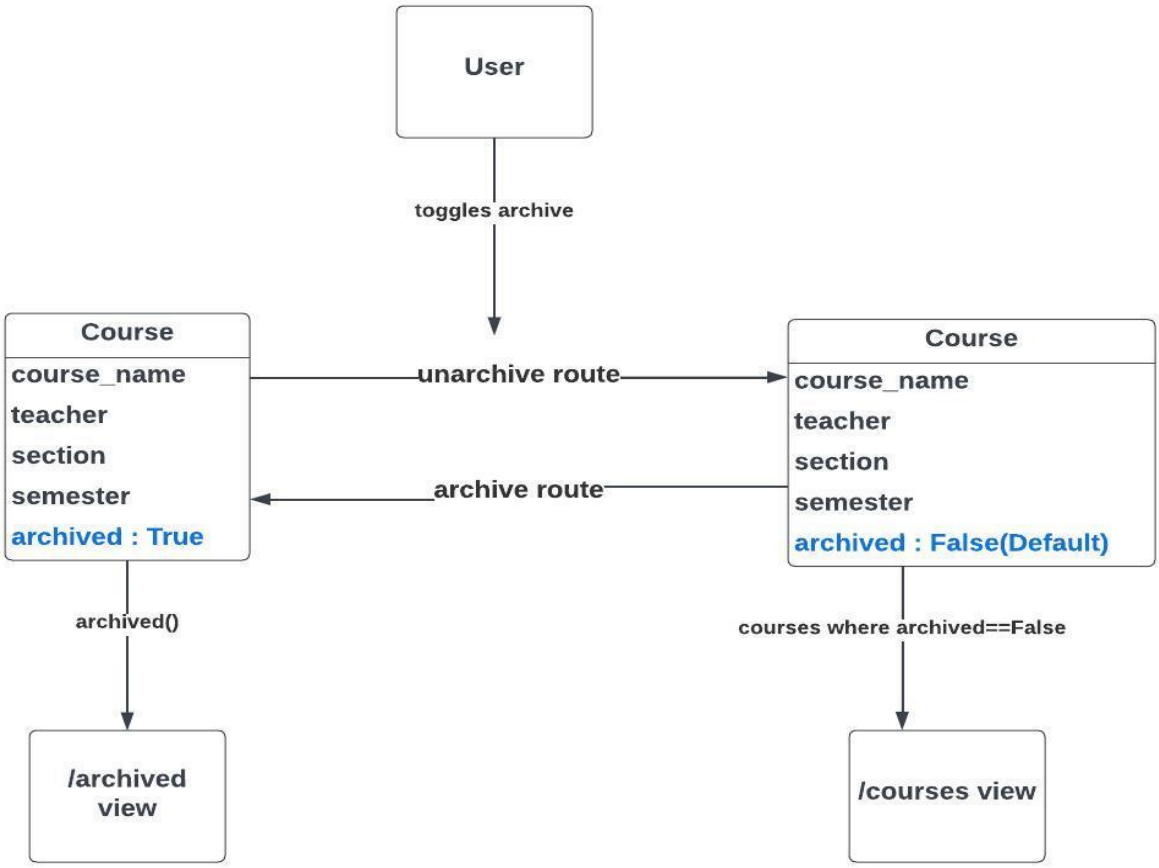
Instead of having to fill the fields manually we need to be able to select from a dropdown to minimize human errors.



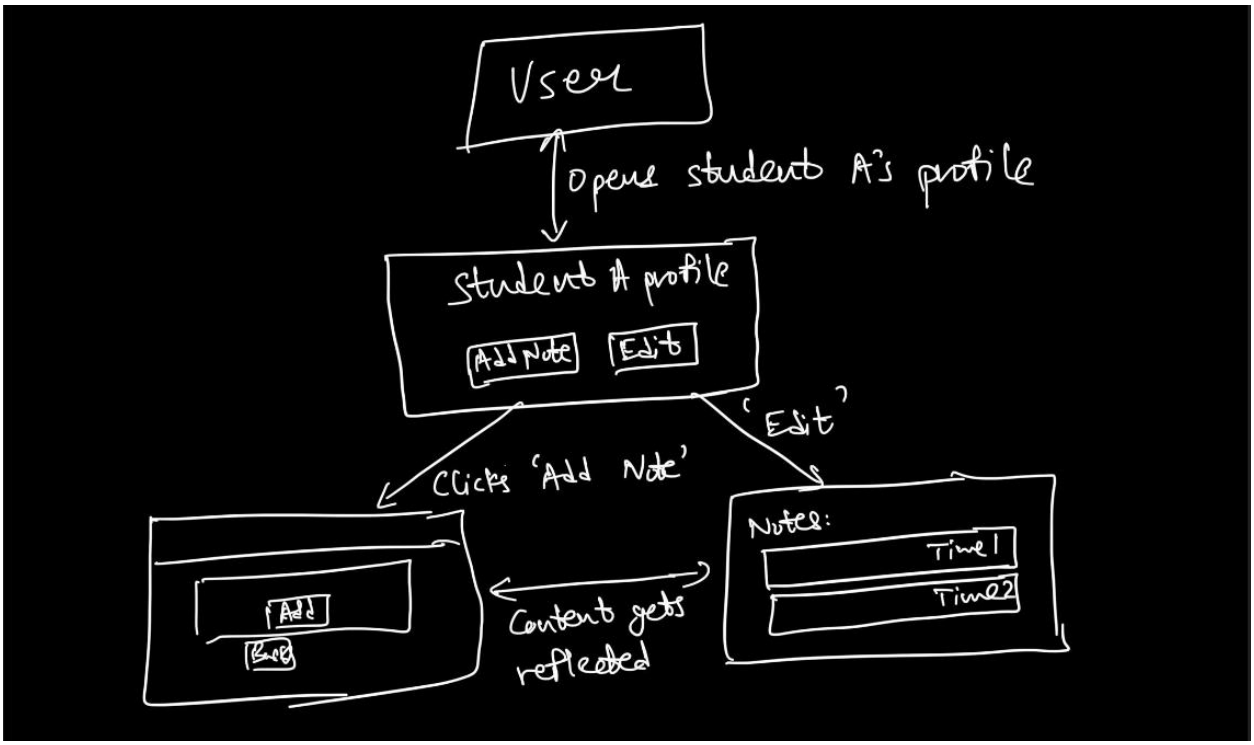
Fix UI by interchanging the column headers, a bug fix.



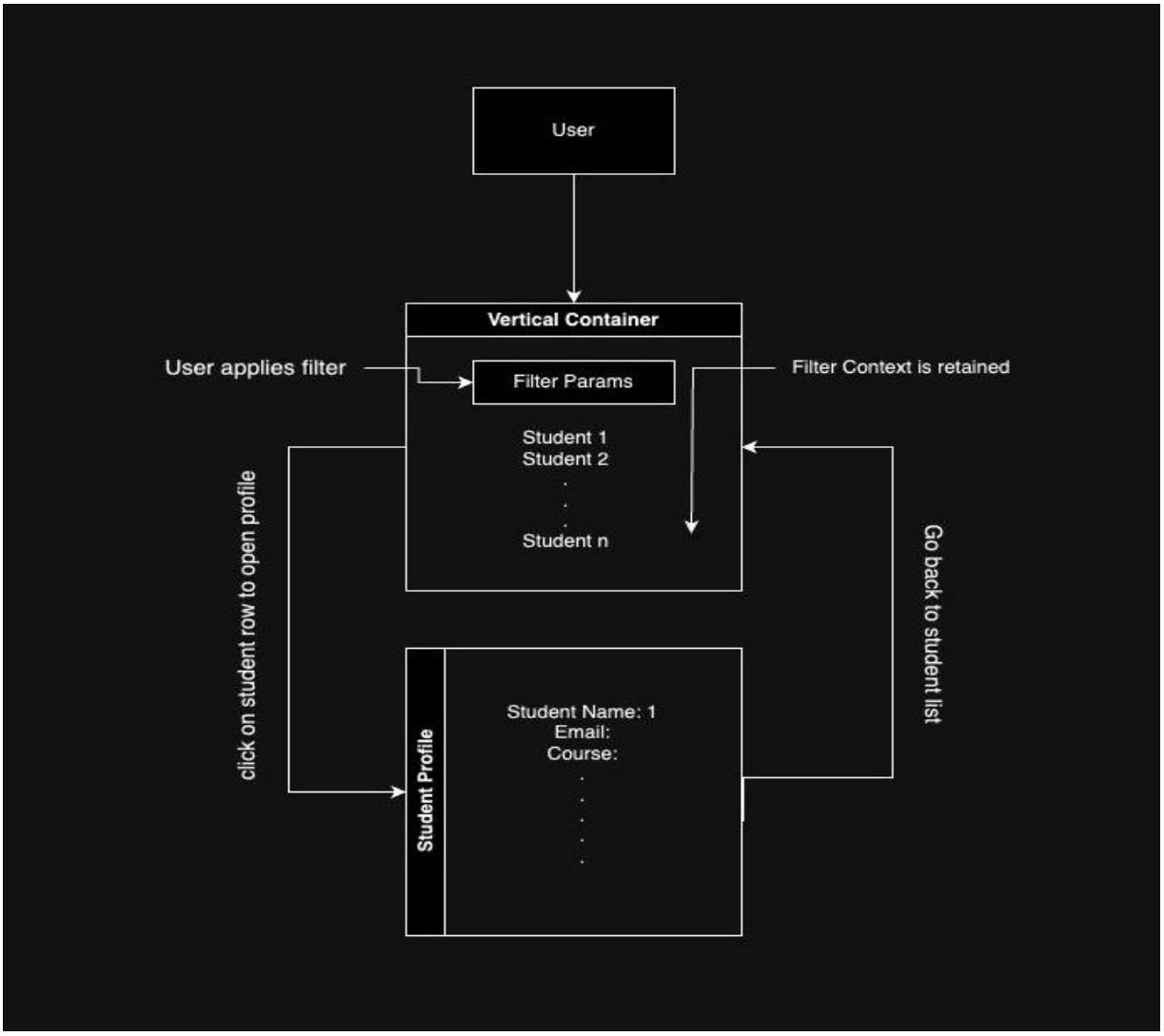
A class diagram showing all the methods inside each controller (Professor, Course, Note, Student)



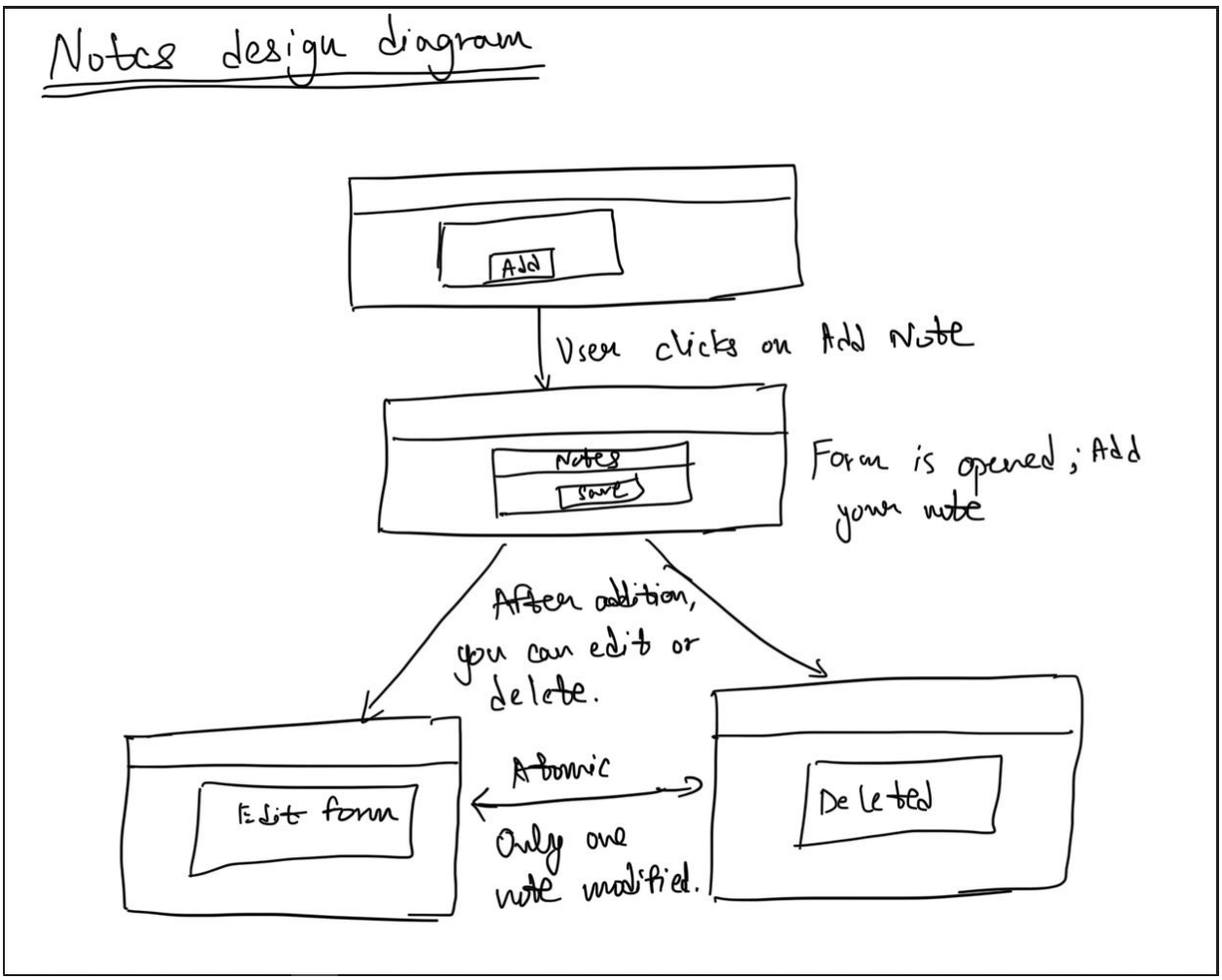
This is the workflow for the archive section which toggles between the archived courses and non archived courses based on the boolean “archived”



A high level workflow of the notes feature which has Add Note by using a form to Add Notes



The filters need to retain history once applied instead of resetting on page refresh, this is the workflow behind how the filters were retained.



A similar workflow for the Edit notes was added which involved editing the existing note using a form on a new page, delete just removes it after asking for confirmation.

Legacy Projects

Understanding the Existing Project

Our team embarked on the journey of working with a legacy project that had been previously developed by another group. The first step in this endeavor was gaining a comprehensive understanding of the existing codebase. This involved conducting a thorough code review, system analysis, and documentation examination to grasp the project's architecture, design patterns, and overall functionality.

Our initial step in assimilating the legacy project involved a meticulous review of the project's documentation. This process was instrumental in gaining insights into the project's architecture, data flow, and key components. By delving into the documentation, we developed a nuanced understanding of the controllers, database structure, and other pivotal aspects that form the backbone of the application.

Building upon the foundation laid by the documentation, we conducted an in-depth analysis of the controllers and database schema. This allowed us to unravel the intricacies of how data is processed, managed, and presented within the application. Our focus encompassed comprehending the logic embedded in the controllers and understanding the data models that constitute the core of the project.

A critical facet of our exploration involved a detailed examination of the Ruby tests associated with the project. By scrutinizing the test suite, we not only validated the correctness of existing functionalities but also gained valuable insights into the intended behavior of various components. This process facilitated a deeper comprehension of the codebase and provided a solid foundation for subsequent modifications and enhancements.

Understanding the deployment procedures was paramount to ensuring the project's accessibility and availability. We dedicated time to familiarizing ourselves with the intricacies of deploying the application, specifically on the Heroku platform. This involved configuring deployment settings, addressing dependencies, and streamlining the deployment pipeline. Our objective was to fortify our proficiency in efficiently managing the project's lifecycle from development to production.

Recognizing the critical role of deployment in the project's success, we prioritized gaining a comprehensive understanding of deployment procedures. A robust deployment

strategy is fundamental to ensuring the accessibility and availability of the project, directly impacting the user experience.

Given that our project utilizes the Heroku platform for hosting, we devoted substantial time to delving into its nuances. Heroku's Platform-as-a-Service (PaaS) model simplifies deployment, yet grasping its intricacies was crucial for optimizing our deployment processes. This involved not only understanding the platform's features but also aligning our application with its best practices.

Configuring deployment settings was a meticulous process aimed at tailoring the application for its intended production environment. This encompassed defining runtime configurations, managing environment variables, and fine-tuning settings to align with Heroku's deployment requirements. The goal was to create a deployment setup that seamlessly translated our development efforts into a robust and scalable production environment.

Effective deployment goes beyond the application code itself; it involves ensuring that all dependencies, including libraries and external services, are seamlessly integrated. We meticulously addressed dependencies during deployment, verifying compatibility, and optimizing configurations to guarantee a smooth operation in the production environment. This included version management and meticulous package installations to eliminate potential runtime issues.

A key focus was on streamlining the deployment pipeline, which represents the end-to-end process of delivering code changes from development to production. Automation played a central role in this, with an emphasis on continuous integration and continuous deployment (CI/CD) practices. By automating repetitive tasks and introducing robust testing mechanisms, we aimed to enhance the reliability and efficiency of our deployment pipeline.

Our overarching objective was to fortify our proficiency in managing the project's complete lifecycle. This extended beyond mere deployment to encompass the entire spectrum from development to production. We sought to establish not only a one-time deployment success but a sustainable, adaptable framework for future iterations, updates, and improvements.

The holistic understanding gained through documentation review, controller and database analysis, Ruby test examination, and deployment procedures laid a robust groundwork for our team. This knowledge not only empowered us to address current requirements but also positioned us strategically for future modifications, optimizations, and feature enhancements. Our thorough onboarding process ensures that we are well-equipped to navigate the complexities of the project and contribute meaningfully to its ongoing development and evolution.

Refactoring and Modifications for students.rb model for increasing coverage:

Refactoring `Student.search`:

- Simplified the method by removing unnecessary code.
- Replaced commented-out lines with a more concise and readable query.

Refactoring `Student.getDue`:

- Used ActiveRecord queries for a more efficient implementation.
- Enhanced readability by eliminating the manual iteration.

Refactoring `Course.search_course` and `Course.search_semester`:

- Simplified and improved these methods using a ternary expression.
- Eliminated unnecessary commented-out code.

Refactoring `Course.search_student`:

- Streamlined the method by removing redundant code.
- Improved variable names for better clarity.
- Enhanced efficiency using ActiveRecord queries.

Code Comments:

- Removed commented-out lines that were no longer needed.
- Commented sections related to error handling and alternative logic were preserved.

Variable Naming:

- The variable `@courses_db_result` might benefit from a clearer name for better readability.

Console Output:

- Removed `puts` statements that were used for debugging.

By following these steps, the codebase has been refined, making it more maintainable and adhering to best practices. The modifications aim to improve readability, eliminate redundancy, and enhance overall code quality.

Team Roles

We took turns ensuring that each of us got a taste of what it feels like to be a Scrum Master or Product Owner and once we were all done getting a turn we just chose randomly since there are only 4 of us and 5 iterations.

Role Assignment - Iteration 1

1. Shubham Mhaske - Scrum Master
2. Vivek Narukurthi - Product Owner
3. Akshit Bansal - Developer
4. Shreshth Kushwaha - Developer

Role Assignment - Iteration 2

1. Shubham Mhaske - Developer
2. Vivek Narukurthi - Developer
3. Akshit Bansal - Product Owner
4. Shreshth Kushwaha - Scrum Master

Role Assignment - Iteration 3

1. Shubham Mhaske - Product Owner
2. Vivek Narukurthi - Developer
3. Akshit Bansal - Developer
4. Shreshth Kushwaha - Scrum Master

Role Assignment - Iteration 4

1. Shubham Mhaske - Developer
2. Vivek Narukurthi - Developer
3. Akshit Bansal - Scrum Master
4. Shreshth Kushwaha - Product Owner

Role Assignment - Iteration 5

1. Shubham Mhaske - Product Owner
2. Vivek Narukurthi - Scrum Master
3. Akshit Bansal - Developer
4. Shreshth Kushwaha - Developer

Scrum Iterations

The below is a high level overview of what was achieved during each iteration along with the number of story points completed per individual during that iteration. Iteration 0 is excluded here because that was used just for planning purposes.

Iteration 1 Summary and Effort

This iteration we focused mainly on bug fixes to get a feel of the workflow of the project since it was legacy. We also added some basic functionality like Adding Notes, Edit Course to the Student profile and converting emails from email.tamu to tamu.edu.

Below you can find the number of story points completed by each member of the team for the iteration.

Vivek Narukurthi	4
Shubham Mhaske	2
Akshit Bansal	2
Shreshth Kushwaha	2

Iteration 2 Summary and Effort

- Modifications in model to include archive:False by default
- Modification in database schema to include archive column
- Including route and view for archive courses
- Making methods for filtering archived courses
- Modification in the model to add verification checks for adding new courses

Below you can find the number of story points completed by each member of the team for the iteration.

Vivek Narukurthi	3
Shubham Mhaske	4

Akshit Bansal	3
Shreshth Kushwaha	4

Iteration 3 Summary and Effort

● Modified the Add notes feature to have timestamp and user email

● The edit student page now shows each note instead of the object reference

● Added Drop Down Lists for entering course information while creating new courses

● Comprehensive dashboard added on the home page so that the user can see the

statistics of the students

Below you can find the number of story points completed by each member of the team for the iteration.

Vivek Narukurthi	4
Shubham Mhaske	4
Akshit Bansal	4
Shreshth Kushwaha	5

Iteration 4 Summary and Effort

● Improved coverage to almost acceptable standard by writing Rspec for quiz, sessions, etc
● Student Search Bar for the Courses page has been made case insensitive
● Toggle button for the stats dashboard has been modified to display appropriate options
when the dashboard is minimized and expanded
● User Interface changes for the Courses page have been made to remove ambiguity for
the users while using the search bars

● Interface has been changed for the Archive Courses page to display the semester of the archived course, and in a much more user friendly format

Below you can find the number of story points completed by each member of the team for the iteration.

Vivek Narukurthi	4
Shubham Mhaske	5
Akshit Bansal	5
Shreshth Kushwaha	5

Iteration 5 Summary and Effort

● Notes feature was made atomic by adding the timestamp, edit and delete buttons for each note. Also each note was represented using cards which differentiated them.
● Coverage has been increased to 90%.
● Documentation has been updated to reflect all missing dependencies.
● Rows have been made clickable

Below you can find the number of story points completed by each member of the team for the iteration.

Vivek Narukurthi	6
Shubham Mhaske	6
Akshit Bansal	6
Shreshth Kushwaha	5

Total Effort (All Iterations)

This is the total effort contributed by each team member at the end of 6 iterations. We have assumed that one story point is worth **4 hours** of effort for our project.

Team Member	Story Points
Vivek Narukurthi	21
Shubham Mhaske	21
Akshit Bansal	20
Shreshth Kushwaha	20

Customer Meetings

Iteration 1 Meeting Summary

The demo meeting for iteration 1 is on Sep 29, 2023, 3:00 pm to 3:30 pm

- Customer Feedback:
    - Making Add Notes Atomic
    - Make the flash message with 2 options to close and one to disappear
    - Prioritize the filters section on the students and courses page as well for the next iteration
    - Add “edit course” button on the course profile page
    - Add archive course option
    - Make UI intuitive

Iteration 2 Meeting Summary

The demo meeting for iteration 2 was on October 13, 2023, 1:30 pm to 3:00 pm

- Customer Feedback:
    - We mentioned to the customer that we didn’t implement many features because we were focused on getting the coverage up till 70% at least

○ Making Add Notes Atomic with the user who created and timestamp is top priority for next iteration

○ Implement search bar feature is also a priority

○ Get coverage upto 90% next iteration

Iteration 3 Meeting Summary

● The demo meeting for iteration 3 was on October 25, 2023, 1:30 pm to 3:00 pm

● Customer Feedback:

○ Customer asked us to perform minor UI changes in archive unarchive feature

○ Customer also asked us to make a practice dashboard on the homepage

○ There were some discussions on the Atomic notes functionality, Adding TA, new students to a course, dropdowns while adding a new course, and some other use cases

○ Implement search bar feature is also a priority

○ Get coverage upto 90% next iteration

Iteration 4 Meeting Summary

● The demo meeting for iteration 4 was on November 10, 2023, 1:30 pm to 2:30 pm

● Customer Feedback:

○ Customer asked us to come up with student practice game feature

○ Took feedback from customer for implementing features and customer feedback was positive.

○ Customer asked for UI changes in the notes feature.

○ Customer asked to implement sorting on the student list.

○ There were some discussions on the Atomic notes functionality, and making it

more readable and retaining context in searching students.

Iteration 5 Meeting Summary

● The demo meeting for iteration 5 was postponed to next Monday at 1:30 pm because of holidays. We reached out to the client via email.

● Customer Feedback:

  ○ Customer asked us to come up with student practice game feature

  ○ Will meet next week to have final discussions and demo

  ○ Next steps regarding migrations to be discussed as well

BDD/TDD Process

Testing Processes

Explain the BDD/TDD (Behavior-Driven Development/Test-Driven Development) process used, highlighting any benefits or problems encountered.

Test-Driven Development (TDD) in our Project:

- Writing Tests First: In our project, TDD is integral to our development process. Before implementing any new feature or modifying existing code, our developers write unit tests that outline the expected behavior of the functionality.
- Red-Green-Refactor Cycle: We adhere to the Red-Green-Refactor cycle, ensuring that each iteration begins with a failing test (Red), followed by the minimum code required to pass the test (Green), and concluding with code refinement without altering external behavior (Refactor).

Behavior-Driven Development (BDD) Integration:

- User-Centric Approach: BDD is seamlessly integrated into our development workflow. We prioritize a user-centric approach, emphasizing collaboration among developers and the client.

Fixing existing issues with tests

Our team actively addressed and rectified issues within the students controller, particularly those related to the quiz and tags features. Through a meticulous debugging and refactoring process, we ensured the seamless functionality of these critical components, bolstering the overall reliability of the students module.

Specific attention was dedicated to refining the quiz feature, ironing out any inconsistencies or bugs within the students controller. This involved not only fixing existing issues but also optimizing the codebase to enhance the efficiency and maintainability of the quiz-related functionalities.

The tags feature, a vital aspect of our application's organizational structure, received special consideration. Our team implemented refinements and modifications to augment the tags functionality within the students controller. This not only resolved existing issues but also laid the foundation for future scalability and adaptability.

Benefits and Considerations of BDD and TDD:

Early Detection of Issues:

Embracing Test-Driven Development (TDD) with RSpec allowed us to proactively identify and rectify potential issues in the early stages of development. Writing tests as a precursor to coding acts as a safety net, averting the accumulation of bugs and ensuring a robust codebase.

Cucumber, as our Behavior-Driven Development (BDD) tool, proved invaluable in collaborative scenarios. By engaging stakeholders early in the process, we can identify and resolve misunderstandings swiftly, significantly reducing the likelihood of issues surfacing later in the development lifecycle.

Improved Code Quality:

Our codebase witnesses a marked improvement in reliability and maintainability through the disciplined practice of TDD with RSpec. Each piece of code is complemented by a thorough suite of tests, providing a safety net against regressions and contributing to a more dependable software foundation.

Adopting a BDD approach with Cucumber ensured that our development efforts align closely with user expectations. Focusing on delivering features in line with user needs enhanced the overall quality of our software, fostering a user-centric development philosophy.

To fortify the robustness of our application, we introduced a set of new tests for the sessions controller. This proactive testing approach ensures that the sessions-related functionalities are thoroughly validated, promoting a more resilient and error-resistant codebase.

Collaboration and Communication:

Cucumber, with its Gherkin syntax, significantly enhanced collaboration by providing a shared language for describing system behavior. This shared vocabulary not only streamlined communication among developers but also bridged the gap between technical and non-technical stakeholders. The result is a more cohesive and cooperative development environment.

Internally, RSpec as our testing framework improved communication within our team. Each unit of code is not only a functional implementation but also a documented validation of its expected behavior. This internal validation process fostered a shared understanding among team members, contributing to a more cohesive and aligned development team.

Challenges and Considerations:

The continuous maintenance of our RSpec and Cucumber test suites demanded ongoing effort. However, we recognize that this maintenance effort is a worthwhile investment in the long-term stability of our project.

The initial investment in writing tests before code with RSpec and Cucumber may seem time-consuming, but we observed that the long-term benefits in terms of improved project stability and reduced bug accumulation justify this up-front time investment.

It's crucial to use RSpec and Cucumber judiciously. Understanding the capabilities of these tools and ensuring they serve the specific needs of our project efficiently is essential.

In conclusion, the strategic integration of RSpec for TDD and Cucumber for BDD into our project's development process has proven instrumental in fortifying our codebase's reliability, aligning our efforts with user expectations, and fostering effective collaboration. While acknowledging the learning curve and maintenance considerations, the enduring benefits affirm the wisdom of embracing these testing frameworks in our unique project context.

Configuration Management

Version Control (Git) and Releases (Approval Process)

For our project, we employed GitHub as our version control system and Pivotal Tracker to monitor the project's status. In the initial iterations, we strategically conducted several spikes to gain a comprehensive understanding of the codebase, especially as part of bug-fixing initiatives.

To facilitate a streamlined development process, we implemented a well-defined configuration management approach. This included the creation of **44 new branches**, each serving a specific purpose or feature. Notably, we established a stringent workflow wherein any branch intended for merging into the production environment required approval from at least one team member. This meticulous approval process was instituted to minimize the risk of introducing bugs into the production codebase.

Our use of branches allowed us to work on distinct features or bug fixes concurrently, ensuring a parallel development workflow. Each branch was a focused space for development, promoting code isolation and facilitating more effective collaboration among team members. Furthermore, this approach significantly reduced the likelihood of conflicts arising from simultaneous changes to the codebase.

In terms of releases, we followed a systematic approach to ensure the stability and reliability of our production environment. The development and testing of features occurred within specific branches, and only after thorough testing and peer review did these branches undergo the approval process for merging into the production branch.

This meticulous configuration management strategy, involving spikes, branches, and releases, played a pivotal role in maintaining code quality and project organization. It provided a structured framework that not only enhanced collaboration but also safeguarded the integrity of our production codebase.

Production Release Issues

Heroku Deployment

We faced a couple of issues with Heroku

1) Make sure you are using the proper PostgreSql plugin on Heroku
2) Make sure the Dynos you are using are Basic using anything less than that might not work
3) You need to make sure to push the credentials file after feeding Heroku with the right master key value, we faced some issues with this
4) There was a slight gap in the documentation when it came to pushing changes to heroku we have fixed those gaps promptly

Development Tools

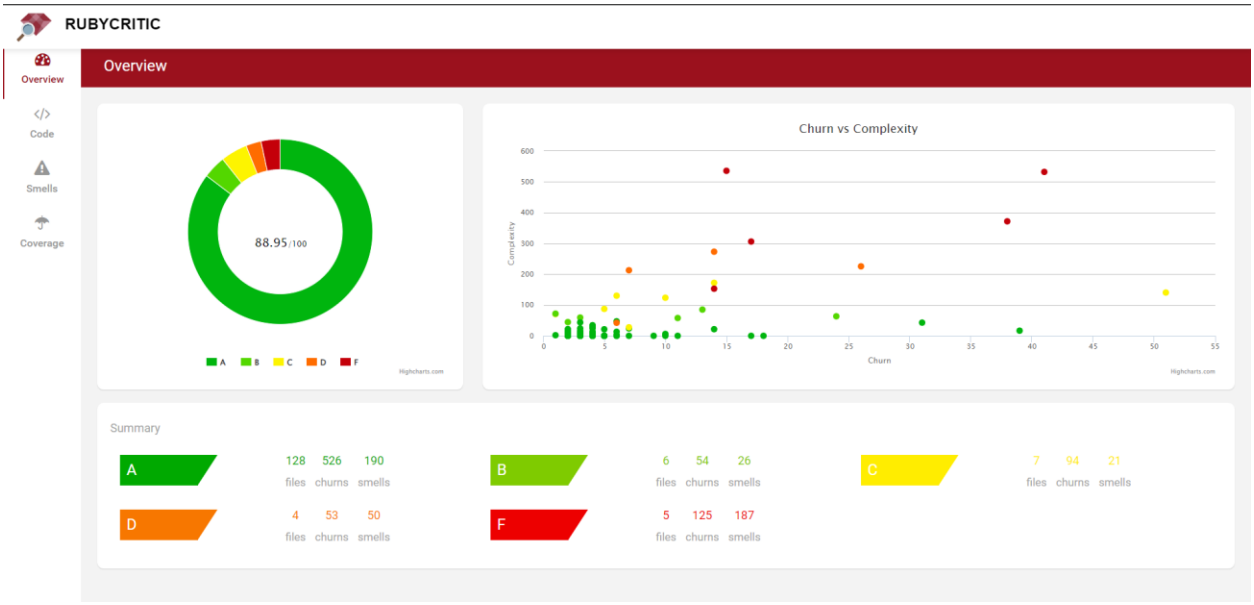
GitHub, Actions

We learned a lot on how to become better at using Git, GitHub thanks to a few challenges below:

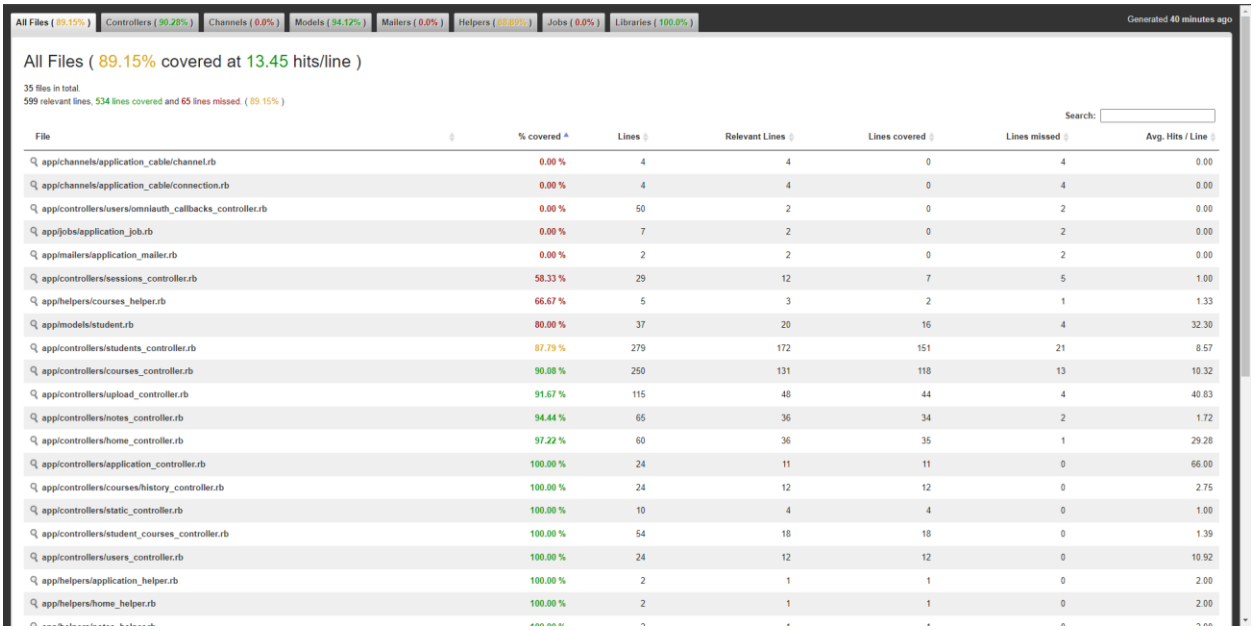
1) Oftentimes we had difficulty rolling back changes without affecting the rest of the code so we would recommend creating new branches and having protections for both the dev and master branches.
2) Github Actions posed some challenges during initial setup but that was mostly because of the environment variables being configured incorrectly so ensure your environment variables and secrets are correct and up to date.

Additional Tools/Gems (SimpleCov and RubyCritic)

SimpleCov and RubyCritic were extensively used across all iterations to help us improve our code quality and coverage. They were very easy to use as shown below screenshots from Iteration 5.



You can click on the Code blade and see which part of your code needs fixing which really helped make our lives much easier.



SimpleCov helped us improve the quality of our tests. Clicking on any file above will open the lines of code not covered by tests which made adding new tests hassle free and ensured we didn't break already working tests.

One advice while using RubyCritic would be to create a new virtual environment since we require a ruby version of 3.2.0 or higher and our project runs on 3.1.3.

Deployment Process

Repository Contents and Deployment

These are the dependencies we need to install beforehand

These are the steps to follow for spinning up development server

Getting the Application Development Ready

- Create google oauth2 client id:
  - Go to google cloud apis & services
  - If you've never been here before, you'll need to make a project first and congifure your oauth consent screen
    - Make the project internal
    - Only fill in the required fields:
      - Name: Your app's name
      - Email: Your email
      - Authorized domains: Your apps domain, E.g. `appname.herokuapp.com`. For development purposes just specify the existing app name `sks-tamu-dev-e3e46be25a57.herokuapp.com`
      - Developer contact info: your email
  - Go to credentials, then click create credentials at the top and select Oauth client id
    - Application type: Web application
    - Name: Your app's name, E.g: sks-tamu
    - Authorized redirect uris, Add: `http://localhost:3000/auth/google_oauth2/callback` and `http://127.0.0.1:3000/auth/google_oauth2/callback`
    - Take note of the Client id and Client secret
- Remove encrypted credentials that you cannot decrypt: `rm -f config/credentials.yml.enc`
- Create and edit new credentials: `EDITOR=nano rails credentials:edit`
  - Add google oauth client id and secret

```
GOOGLE_CLIENT_ID: ...
GOOGLE_CLIENT_SECRET: ...
```

  - Save and exit: `Ctrl+X`, Press Y and Enter.
- It may take about 5 minutes for the effects to take place
- Run the web application locally using: `rails server`
- Create a new account using your TAMU email id and have fun :).
- Do leave us some feedback if you find any difficulties in following along or for other queries at our email id's below.

These are the steps needed for deploying on Heroku i.e. your prod environment

Getting the Application Production Ready

- Install heroku cli: `curl https://cli-assets.heroku.com/install-ubuntu.sh | sh`
- Login to heroku: `heroku login -i`
  - `username: <your username>`
  - `password: <your API key>`
    - get your API key from your heroku account
- Create an app on heroku: `heroku create [appname]`, where `[appname]` is an optional name for the app
- Create an s3 bucket
- Create iam role for app to access s3 bucket
  - Take note of the access key id and secret access key
  - Create access policy: in your iam s3 user, under permissions, click add permission, then create inline policy
    - Choose `s3` as the service
    - Specify the actions allowed:
      - `ListBucket`
      - `PutObject`
      - `GetObject`
      - `DeleteObject`
    - Specify bucket resource ARN for the ListBucket action: click add ARN to restrict access
      - Put name of your s3 bucket in the bucket name field
    - Specify object resource ARN for the PutObject and 2 more actions:
      - Put name of your s3 bucket in the bucket name field
      - Click any next to object
    - Click review policy at the bottom
    - Make sure it looks right and then create it
- In `config/storage.yml`, make syre `region` and `bucket` fields match yourt bucket's region and name

- Create google oauth2 client id:
  - Go to google cloud apis & services
  - If you've never been here before, you'll need to make a project first and congifure your oauth consent screen
    - Make the project internal
    - Only fill in the required fields:
      - Name: your app's name
      - Email: your email
      - Authorized domains: your apps domain, e.g. `appname.herokuapp.com`
      - Developer contact info: your email
  - Go to credentials, then click create credentials at the top and select oauth client id
    - Application type: web application
    - Name: your app's name
    - Authorized redirect uris, add: `https://appname.herokuapp.com/auth/google_oauth2/callback`
    - Take note of the client id and client secret
- Remove encrypted credentials that you cannot decrypt: `rm -f config/credentials.yml.enc`
- Create and edit new credentials: `EDITOR=nano rails credentials:edit`
  - Add AWS access key and secret (the iam s3 user access key id and secret access key)

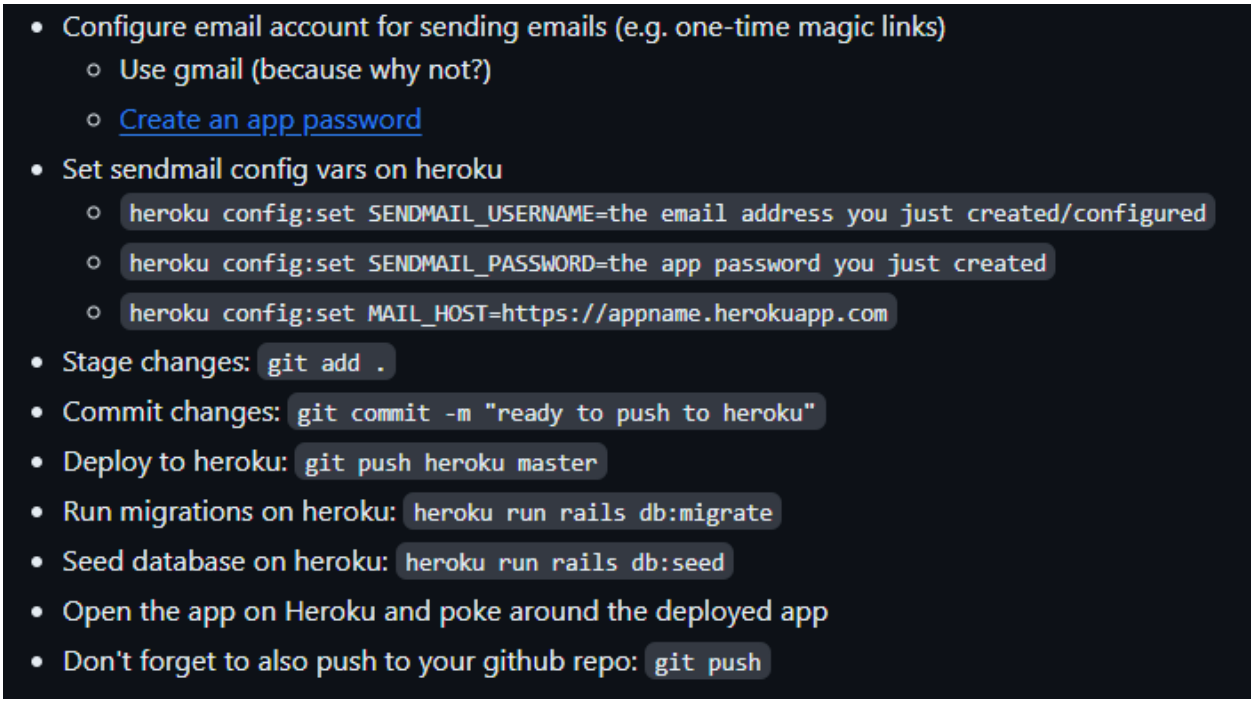
```
aws:
  access_key_id: ...
  secret_access_key: ...
```

  - Add google oauth client id and secret

```
GOOGLE_CLIENT_ID: ...
GOOGLE_CLIENT_SECRET: ...
```

  - Save and exit: `ctrl+o` , `ctrl+x`
  - Take note of the master key in the console
- Save master key to heroku as config var (for security): `heroku config:set RAILS_MASTER_KEY=...`

Please ensure the master key matches with what shows on the console



The above instructions have been updated in the repository ReadMe as well along with details of our team in case you run into any issues.

Project Links

Important Links

● Heroku deployment: https://sks-tamu-dev-e3e46be25a57.herokuapp.com/

● Pivotal Tracker: https://www.pivotaltracker.com/n/projects/2595510

● Github: https://github.com/Knowledge-Ninjas/student_knowledge_system

● Slack: https://app.slack.com/client/T05SA7MHMDW/C05RJA72C5B

Presentation and Demo

Video Links

Presentation Video : https://www.youtube.com/watch?v=GgcevCFxIDU
Demo Video : https://www.youtube.com/watch?v=B9LijDSXdgI