TEAM: ChefBatMan
Group Members:

1. Chenyang Fang(cf2843)
2. Wan Xu(wx2227)
3. Guancheng Ren(gr2625)
4. Tian Niu(tn2415)

Part 1:

Link: https://github.com/wx2227/4156
git reset --hard d70b01bda58b2ee492be23b79a86719419e7d7ed

Part2:

User stories:

1. As a student of a particular course, I want to view notes uploaded by the professor or other students in this class, so that I can better understand course materials and prepare for homeworks and exams.
    a. Common cases:
        i. A logged-in user can search a class by course number.
        ii. The user can click a particular file that was previously uploaded to view.
    b. Special cases:
        i. A logged-in user enters a course number combo that does not exist. The page shows no notes(no data).
2. As a student/professor of a particular course, I want to upload my lecture notes to share with other students in this class, so that I can help others to study.
    a. Common cases:
        i. A logged-in user can upload his class notes in pdf type by indicating which department and course number the file belongs to.
        ii. The uploaded file is then available to all users to view and download.
    b. Special cases:
        i. A logged-in user enters course number that does not exist. The system pops up an error message.
        ii. A logged-in user enters a valid combo but tries to upload a file that has invalid file type. The system pops up an error message.
3. As a student of a particular course, I want to download notes uploaded by the professor or other students in this class, so that I can view the notes offline.
    a. Common cases:

        i. A logged-in user can search a class by department name and course number.
        ii. The user can click a particular file that was previously uploaded, then choose to download.
    b. Special cases:
        i. A logged-in user enters a course number that does not exist. The system shows no notes(no data).
4. As a student of a particular course, I want to upvote or downvote a note so that I have a way to tell others students that the file is of high or low quality.
    a. Common cases:
        i. A logged-in user can search a class by department name and course number.
        ii. The user can upvote or downvote this file.
        iii. The upvote and downvote number increments by 1 accordingly.
        iv. The user can undo his action by clicking the button again and the upvote/downvote number decrements by 1 accordingly.
    b. Special cases:
        i. A logged-in user tries to upvote (or downvote) a file that has been downvoted (or upvoted) by himself. The system will not decrement or increment the votes.
5. As a student of a particular course, I want to post comments and read comments posted by other students to a particular file belonging to that class, so that I can ask questions, discuss my thoughts, and answer questions related to the course materials.
    a. Common cases:
        i. A logged-in user can search a class by department name and course number.
        ii. The user can choose a note to view explicitly.
        iii. The user can read comments in the discussion forum of that file.
        iv. The user can post his new comments to the forum.
        v. The post is then available to view for all users.
    b. Special cases:
        i. The user enters a blank comment and tries to post it. The system detects that the post is blank and pops up an error message.

Acceptance testing plan:
We manually simulated a user going through all the user stories and see if we can satisfy all the branches of the user story.

1. **View Course Notes**
   Common Case #1:
    a. The user opens the airNote website and login through Google
    b. User searches a course number

c. The course is found and course notes are displayed, the user can see the upvotes and downvotes about the note to get better insight about the quality of the note.
d. User clicks into a note and can view a page with the detailed information about the note. The user can preview the note without downloading it. The user can view the comments about this note and add his/her own comment.

DISCUSSION & BUG FIXED

a. In the initial commit, we just use the fields defined in the note schema like name, related course number to render the UI. But we later want to include other information like the total number of upvotes and downvotes into the note. So during the serialization process of the queryset object in the backend, we calculate the votes corresponding to the note, which can be returned back to the frontend when the API is called. The same pattern followed by the design of comment.
b. Sometimes the preview window doesn't display the pdf file but prompts the user to download the file. Then we realized that we need to fix the Content-Disposition header to contain the "inline" string. So the browser won't attempt to download the file.

RESULT (Passed):

The user clicks the login button at the homepage, a Google login window pops up, the user logs in using one account and is redirected to the course index page. Then the user enters a course number in the search bar and clicks the search button, and the user is redirected to the results page. The result page contains a list of course notes and the user clicks on one of the notes, then the user is redirected to the preview page. The preview page contains a PDF preview window where the user can scroll and view the note file.

Special Case #1 (login through email not from Columbia University):

a. User enters the homepage of airNote
b. Login through Google with email not from Columbia University
Sample input: superman@gmail.com
c. The system reject the login request

RESULT (Passed):

The user can't login using emails outside those from Columbia University

Special Case #2 (invalid login credential):

a. (Suppose the user already has an account)
b. User enters airNote and login through Google with wrong password
c. The system reject the request and ask the user to try again

RESULT (Passed):

The user attempts to login with the wrong password. The Google login panel rejects the password and prompt the user to try again

Special Case #3 (course not exist):

a. The user opens the airNote website and login
b. The user searches a course number corresponding to a department name
c. The system notifies the user that the specified course does not exist.

RESULT (Passed):

The user enters a course that doesn't exist in the database and clicks the search button. The webpage shows "No Data".

2. **Upload Course Notes**

Common Case #1 (user already have an account):

a. The user opens the airNote website and login
b. The user clicks upload note button and is redirected to file upload page
c. The user upload the note and fills in necessary information about the course
   Sample input: a pdf file
d. The user clicks submit button and the note is available to all users

DISCUSSION & BUG FIXED

The upload functionality was implemented with S3 pre-signed urls. The user requests a pre-signed url from S3 and uploads the file through that url. In our first implementation the files uploaded to S3 were all corrupted and not viewable. Then we realized that we need to include the file type "application/pdf" in the request header for AWS to correctly recognize and decode the file.

RESULT (Passed):

The user login through Google, and clicks the upload button on the navigation bar, then the user is redirected to the upload page. The user selects the file he/she wishes to upload, and fills in the necessary information about the course. Then the user clicks upload and the file is uploaded to the server and the user is redirected to the note page.

Special Case #1 (login through email not from Columbia University):

a. User enters the homepage of airNote
b. Login through Google with email not from Columbia University
   Sample input: superman@gmail.com
c. The system reject the login request

RESULT (Passed):

The user can't login using emails outside those from Columbia University

Special Case #2 (user didn't fill in all the required information):

a. The user opens the airNote website and login
b. The user clicks upload note button and is redirected to file upload page
c. The user upload the note but fails to fill in some information
   Sample: forget to fill in title, label, course number of the note
d. The system prompt the user to fill in those fields

RESULT (Passed):

The user didn't fill in all the necessary fields and clicks submit. A window pops up and prompts the user to fill in missing information.

Special Case #3 (invalid file type):

a. The user opens the airNote website and login
b. The user clicks upload note button and is redirected to file upload page
c. The user uploads the file but the file type is not expected.
   Sample input: exe file, bash file

RESULT (Passed):

The user fills in all the necessary fields and selects a non-PDF file the clicks submit. A window pops up and prompts the user to upload a PDF file.

3. **Download Notes**

Common Case #1:

a. The user opens the airNote website and login through Google
b. User searches a course number with a corresponding department name
c. The course is found and course notes are displayed
d. User clicks into a note
e. A download button is visible on the page and the use clicks it
f. A download window pops up and the user saves the file on his/her own computer

RESULT (Passed):

The user clicks the login button at the homepage, a Google login window pops up, the user logs in using one account and is redirected to the course index page. Then the user enters a course number in the search bar and clicks the search button, and the user is redirected to the results page. The result page contains a list of courses and the user clicks on one of the courses, then the user is redirected to the preview page. A “Download File” button is visible on the page and the file downloading window pops up when the user clicks on the button.

Special Case #1 (login through email not from Columbia University):

a. User enters the homepage of airNote
b. Login through Google with email not from Columbia University
   Sample input: superman@gmail.com
c. The system reject the login request

RESULT (Passed):

The user can login using emails outside those from Columbia University

Special Case #2 (invalid login credential):

a. (Suppose the user already has an account)
b. User enters airNote and login with wrong password
c. The system reject the request and ask the user to try again

RESULT (Passed):

The user attempts to login with the wrong password. The Google login panel rejects the password and prompt the user to try again

Special Case #3 (course does not exist):

a. The user opens the airNote website and login
b. The user searches a course number corresponding to a department name
c. The system notifies the user that the specified course does not exist.

RESULT (Passed):

The user enters a course that doesn't exist in the database and clicks the search button. The webpage shows "The course does not exist".

4. **Upvote and Downvote**

Common Case #1 (Upvote):

a. User enters the homepage of airNote with credentials
   Sample credential: cookie, session storage
b. User searches a note and clicks into the preview page
c. User clicks the upvote button
d. The system updates the upvotes by incrementing it by 1

RESULT (Passed):

The user enters the homepage and then enters a course number in the search bar and clicks the search button. Then the user is redirected to the results page. The result page contains a list of courses and the user clicks on one of the courses, then the user is redirected to the preview page. The user clicks on the upvote button, and the upvote button becomes grey and the upvote counter is incremented.

Common Case #2 (downvote):

a. User enters the homepage of airNote with credentials
   Sample credential: cookie, session storage
b. User searches a note and clicks into the preview page
c. User clicks the downvote button
d. The system updates the downvotes by incrementing it by 1

DISCUSSION & BUG FIXED

a. In the initial commit, we maintained upvote and downvote in two data tables. Because it would be more efficient when we are trying to count the total number of upvotes and downvotes. But we found that it may cause the frontend to call the API many times when we are trying to make sure the data updated correctly in both tables. We then combine these two tables into one table, using 1, 0, -1 to represent the upvote, no vote and downvote.
b. In the second commit, we found we need to first check whether or not the note has been voted by the user (GET), then decide to use POST or PUT method to create or update a vote item. We optimize this problem by combining the POST and PUT method together in the backend. We can always use the POST method, and the backend will decide to create or update the vote item in the database accordingly.

RESULT (Passed):

The user enters the homepage and then enters a course number in the search bar and clicks the search button. Then the user is redirected to the results page. The result page contains a list of notes and the user clicks on one of the notes, then the user is redirected to the note page. The user clicks on the upvote button, and the upvote button becomes grey and the upvote counter is incremented.

Special Case #1 (the file has already been upvoted/downvoted by user):

a. User enters the homepage of airNote with credentials of non file owner
   Sample credential: cookie, session storage
b. User searches a note that has been upvoted by him/her and clicks into the preview page
c. The user cannot upvote this note, the upvote button will cancel his/her last action of upvoting instead of to upvote again

DISCUSSION & BUGS FIXED:

a. In the initial commit, users can upvote the note even if the note has been voted by the user. After discussion, we maintained an action state in the component. If the note has been upvoted, we set the action to 'liked'. Based on the action state, we decide whether to increment the upvotes counter or not. The same pattern followed by downvote.
b. In the second commit, based on the action state, we implement part c. But after the user cancels the last action, and refreshes the page, the votes are in the same state as the user did not cancel the action. We fixed this problem by updating the database timely.

RESULT (Passed):

The user can undo the upvote/downvote action by clicking the button again.

Special Case #2 (course not exist):

a. The user login into the airNotes with credentials
b. The user searches for specific course
c. The system pops up messages suggesting that course does not exist

RESULT (Passed):

The user enters a course that doesn't exist in the database and clicks the search button. The webpage shows "The course does not exist".

5. Comment

Common Case #1(comment):

a. User in the note page which shows the detail information about the note
b. User views the document and the comments in the discussion forum
c. User edits the comment in the comment editor located after the discussion list.
d. The user clicks add and the comment is available in the discussion forum

RESULT (Passed):

After the user edits the comment, and clicks the add-comment button, the comment can be shown at the top of the comment list.

DISCUSSION & BUGS FIXED:

a. In the initial commit, after the user clicks the add-comment button, the comment cannot be shown at the top of the comment list. We found that a callback function should be added to the comment list component (parent component of comment editor).
b. In the second commit, the user cannot add the comment twice without refreshing the page. The problem is that loading state within the editor component is not set back to false after the user adds one comment.

Special Case #2 (user enters empty comment)

a. User edits the comment in the comment editor located after the discussion list.
b. The user enters nothing and clicks the add-comment button
c. The system pops up error message suggesting that comment cannot be empty

DISCUSSION & BUGS FIXED:

a. In the initial commit, the user can add a comment like “ ”.

RESULT (Passed):

Users cannot add an empty comment or a comment filled with space.

Bugs found/fix:

Currently the login functionality is still under development and sometimes fails to correctly authenticate the user. And the comment display is displaying incorrectly because of the wrong React configuration. We should be able to fix that before the next demo.

Part3:

Link to all test cases:
https://github.com/wx2227/4156/blob/master/backend/sharednote/tests.py

Link to Configuration of build tool and automated unit testing tool:
https://github.com/wx2227/4156/blob/master/backend/manage.py

Part4:

Front-end:

1. Link to style report: https://github.com/wx2227/4156/tree/master/frontend/styleReport
2. Link to bug report: https://github.com/wx2227/4156/tree/master/frontend/bugReport

Backend (style checker and bug finder in one report):

1. https://github.com/wx2227/4156/tree/master/backend/stylereport