Sprint 1 Retrospective Document

CS307 - Team 23

Arshpreet Singh
Kaushik Ramachandran
Luciano Handal Baracatt
Pratyaksh Motwani
Sadiq Ahmed A Albinalshaikh

What went well?

User Story #1

As a user, I would like to sign up for an *easyA* account.

	Task	Estimated time	Owner
1	Create a sign-up page	3 hrs	Luciano
2	Create a user database	4 hrs	Sadiq
3	Connect the database to add new users	3 hrs	Pratyaksh
4	Test if the sign up page submits the valid user input and records it to the database correctly.	3 hrs	Sadiq
5	Test if the user database handles queries correctly and respond as expected per query.	2 hrs	Kaushik
6	Test if user data and encrypted password are being stored securely in the user database.	2 hrs	Kaushik

Justification for completion:

The signup page was created with the appropriate HTML syntax to send POST requests to the server. Before sending the request, the user interface checks and alerts the user for any invalid inputs. The server (Flask App) receives the request, checks if the data is valid again, translates it to a database format and checks if the user exists. If any error happened, it responds back with the error message where it will be rendered on the web page. If the data was valid, the server sends the data to the database to encrypt the password and store it appropriately. The encrypted passwords are not visible to the Firebase administrators. Afterward, the database responds with success to the server. The server generates and sends a unique email verification link to the user, then lastly redirect them to the log in page with a “You must verify” error message.

User Story #2

As a user, I would like to log in to my account.

	Task	Estimated time	Owner
1	Create a log in page	3 hrs	Arshpreet
2	Setup Firebase Authentication for log in submissions and format (Email and Password)	3 hrs	Sadiq
3	Forbid non-verified accounts to log in	2 hrs	Sadiq
4	Test if the log in page submits the valid user input and checks if it exists in the user database	2 hrs	Pratyaksh
5	Test if logging in compares the credentials correctly the recorded data in the user database.	2 hrs	Pratyaksh
6	Test if a non-verified account could login or not.	3 hrs	Kaushik

Justification for completion:

The log in page was created with the appropriate HTML syntax to send POST requests to the server. Before sending the request, the user interface checks and alerts the user for any invalid inputs. The server (Flask App) receives the request, checks if the data is valid again, translates it to a database format and checks if the user exists. For any error that happens, the server responds back to the user interface with the appropriate error message. If an unverified user tries to log in, the server will generate a new verification email then send an “You must verify” error message to be rendered on the user interface. If the submitted data was valid, the server will send the data to Firebase Authentication to check for the matching credentials then responds with either success or an error message that will be then handled by the server. When it is successful, the server will create a session variable for the user to mark them as logged in. Lastly, they will be redirected to the home page.

User Story #3

As a user, I would like to log out from my account.

	Task	Estimated time	Owner
1	Create a log out button	3 hrs	Arshpreet
2	Set the user as logged out (guest)	3 hrs	Kaushik
3	Refresh the page and re-render the guest user interface after logging out	3 hrs	Kaushik
4	Test if the logout button functions as expected	2 hrs	Pratyaksh
5	Test if the user becomes a guest user after logging out	2 hrs	Sadiq

Justification for completion:

The log out button was created and linked appropriately to the action of logging out. If the user is not logged in (Guest), the log out button will not be visible and the “log in” and “sign up” buttons will be visible. In case a logged in user clicked on log out, the user interface will send a request to the server to execute the signout function. The server will clear the user session to mark them as guests, then they will be redirected back to the home page with the login and signup buttons visible again.

User Story #4

As a user, I would like to change my password after logging in.

	Task	Estimated time	Owner
1	Include “Change Password” option in the account drop down menu	4 hrs	Arshpreet
2	Create a change my password page	5 hrs	Luciano, Arshpreet
3	Check if the old password matches the users’ credentials	2 hrs	Arshpreet, Pratyaksh
4	Update the account’s password in the database with the new encrypted password.	2 hrs	Arshpreet,Pratyaksh
5	Test if the database updates the correct user’s password	5 hrs	Kaushik

Justification for completion:

The change password button was created and linked correctly to the change password page. The button is only visible when the user is marked as logged in, and in case a guest tried to use the URL to go to the change password page, they will be redirected back to the home page. The change password page was created with the appropriate HTML syntax to send the old and new password to the server. In case the user input is not valid, like a short password or unmatching password confirmation, the user interface will alert the user. If the data submitted were valid, the server will receive the data then checks if they are valid again and respond appropriately. The server also checks if the old password is correct. This was done for security reasons e.g. in case a user forgot to log out from a public computer and a stranger tried to change their password. If the old password was correct and the new password matches the confirmed password, the server will translate the data to a database format, gets the user’s ID then send a request to Firebase Authentication to update the user’s data. Firebase will update the user’s data and in case it was a password, it will encrypt it and hide it from Firebase console. The database will respond with success or failure. In case of success, the server will render a success message to the user. The user will not be logged out, and in case they log out and log in again, they need to use their new password to log in successfully.

User Story #5

As a user, I would like to have an account menu for changing my password and logging out.

	Task	Estimated time	Owner
1	Create a drop down menu with “Change Password” and “Log out” options.	4 hrs	Luciano, Arshpreet
2	Make the options in the drop down menu redirect the user accordingly.	2 hrs	Luciano, Arshpreet
3	Create a button that shows up the drop down menu to the user.	4 hrs	Luciano, Arshpreet
4	Test if the “Change Password” Button successfully redirects to change password page	1 hr	Pratyaksh
5	Test if the “Log out” button functions correctly and logs the user out.	1 hrs	Pratyaksh

Justification for completion:

An interactive drop down menu was created and styled appropriately on the user interface. It includes the buttons for logout and change password. The reason for creating this drop down menu is to make the the interface as minimal as possible for ease of use. This menu will only be visible to logged in users and that is when they click on their name on the top right corner. Guests will not be able to see such menu since they are not logged in. The buttons in the menu works as expected according to user stories #4 and #5.

User Story #6

As a user, I would like to know if I am logged in or not.

	Task	Estimated time	Owner
1	Show user's email and account menu button when the user is logged in	3 hrs	Luciano
2	Show the login and sign up buttons when the user is not logged in	3 hrs	Arshpreet
3	Set up the navigation for both the login and sign up button.	2 hrs	Arshpreet
4	Test the implemented buttons and account status visibility on all pages.	3 hrs	Pratyaksh

Justification for completion:

A user status bar was created and styled appropriately on the user interface. The status bar will show the username (anything before the @ in their emails) and "@purdue.edu" is omitted for redundancy. The status bar also acts as a button to show the drop down menu (a.k.a account menu) This bar along with its functionality will only be visible when the user is logged in. The server handles displaying this bar or not, by checking if the user session variable is set or not. The status bar is there on all pages (home page, course pages, and new review page)

User Story #7

As a user, I would like to reset my password if I forget it.

	Task	Estimated time	Owner
1	Create a button in the login page for “Forgot Password?”	1 hrs	Arshpreet
2	Create a password reset page	2 hrs	Luciano
3	Check if the submitted email is in the user database	2 hrs	Pratyaksh
4	Generate a unique link that lets the user reset their account password and make it expire in 1 hour.	4 hrs	Pratyaksh
5	Email the unique link to the user’s email	2 hrs	Pratyaksh

Justification for completion:

A “Forgot Password” link was created and included appropriately in the log in page. A password reset page was created appropriately with the correct HTML syntax for sending a server request. When a guest forgets their password, they will be able to enter their email in the password reset page. The user interface will check if the inputted email ends with “@purrdue.edu” and alerts the user accordingly. Afterward, the server receives the email and checks the database if the it exists or not meaning that the user is signed up or not then responds accordingly. If it exists, a unique link will be generated by the server and emailed to the user’s email. The guest will have to find the email in their inbox and click on it. The link will display a form for entering the new password where it will be updated directly on Firebase Authentication.

User Story #8

As a user, I would like to see a course rating in the course review page.

	Task	Estimated time	Owner
1	Create a page to reflect the current course offered at Purdue University	5 hrs	Luciano
2	Create a UI to reflect the course rating at the course page	3 hrs	Arshpreet
3	Retrieve the course rating from the database.	3 hrs	Kaushik
4	Test if the page correctly updates the the course rating to reflect on the module	1 hrs	Kaushik
5	Test if the page correctly displays the respective course rating	1 hrs	Pratyaksh

Justification for completion:

A course page was created and rendered appropriately from the course data obtained from the database. The course pages are generated dynamically depending on the database where in the project file system there is only one physical HTML page that resembles the template of the course page. The server takes the course ID from the URL e.g. www.example.com/course/CS18000, then queries the database for this specific ID, if it exists it will render the page with the appropriate data otherwise it will display a page not found error to the user. The course page contains a for loop for displaying all the relevant posts and calculates the average ratings. This loop works during the course page loading. The rating will eventually be displayed as a 5-star rating format under the course ID on the page. In case any new post was added, the new rating will be displayed once the page is refreshed.

User Story #9

As a user, I would like to see a course description in the course review page.

	Task	Estimated time	Owner
1	Create the course database	5 hrs	Sadiq
2	Create a UI to contain the course description for the respective course	2 hrs	Arshpreet
3	Record the course description in the database	2 hrs	Sadiq
4	Test if the page correctly displays the course description for the respective course	1 hrs	Kaushik

Justification for completion:

The course database contains the all the fields needed for the course information. It has: course_id, course_name, and description. As explained in user story #8, the course page is rendered dynamically once the user requests a specific page. During the loading time, the server will pull the needed course data from he database including the description then renders them in the appropriate positions in the template. The course fields are entered manually by the administrators from the Firebase console UI.

User Story #10

As a user, I would like to know that all reviews are from Purdue students.

	Task	Estimated time	Owner
1	Create an email verification system to verify that a signed up user owns a functional Purdue email.	12 hrs	Luciano, Arshpreet, Sadiq
2	Create a system to forbid non-Purdue users from signing up.	12 hrs	Luciano, Arshpreet, Sadiq
3	Limit unverified users to login until they are verified.	4 hrs	Kaushik, Pratyaksh
4	Show “Purdue Student” as the author name of all review posts.	2 hrs	Kaushik, Pratyaksh

Justification for completion:

This user story is dependent on the user stories #11 and #12 where the idea of the project is to make it exclusive to Purdue students. To make that possible, all signed up users should sign up with a valid Purdue email and they also need to prove that they can still access that email by completing the email verification process. Any newly signed up user who tries to use a fake Purdue email will not be able to log in which prevents any outsiders from contributing to the course review platform. Lastly, to give the reviews a watermark and a reminder that the reviews are from Purdue students, the author of each post will show “Purdue Student.”

User Story #11

As an administrator, I would like only Purdue students with "@purdue.edu" emails to be able to sign up.

	Task	Estimated time	Owner
1	From a front-end point of view: Use regular expressions to only accept "XXX@purdue.edu" formats in sign up, log in, and password reset.	2 hrs	Sadiq, Arshpreet
2	Implement a backend input sanitization to verify the input again and avoid web attacks that would involve code injection.	4 hrs	Pratyaksh, Kaushik
3	Give an error to the user in case the input fails in either front-end or back-end verification,	2 hrs	Luciano

Justification for completion:

Upon signing up or resetting the password, the user input for the email only permits to submit emails that end with "@purdue.edu" When a user tries to enter an email with a different domain, the front-end will give an alert saying that the email should be a Purdue email. To prevent outsiders to sign up, the back-end server will also check if the email is a Purdue email and respond with an error accordingly.

User Story #12

As an administrator, I would like newly signed up users to verify their ownership of their Purdue emails.

	Task	Estimated time	Owner
1	Verify that the email exists and working.	4 hrs	Sadiq
2	Send an email confirmation to the newly signed up user with a verification link that expires in 1 hour.	8 hrs	Arshpreet, Sadiq

Justification for completion:

When the user signs up with a valid Purdue email, the server will generate a unique verification link and send it to this email. The user will not be able to log in unless they verify their email and if they try to do so, the server will regenerate a new verification link then produces an error to the user saying “You must verify.” The verification link will expire in one hour, and to regenerate a new link the user should try to log in or sign up again.

User Story #13

As a user, I would like to have an easy navigitable user interface throughout the website.

	Task	Estimated time	Owner
1	Design the web and file storage necessary for an intuitive flow	2 hrs	Luciano, Arshpreet
2	Set up the HTML skeleton and navigation between the different pages	4 hrs	Luciano
3	Add necessary CSS to allow user to have an identifiable visual contrast	2 hrs	Luciano
4	Add necessary JavaScript to make the website more responsive for to the user	3 hrs	Luciano
5	Debug and test the responsiveness and syntax of the web pages	4 hrs	Luciano

Justification for completion:

The user interface follows a user experience logic where it maintains a minimal style with the only needed elements for every page. The main page has three main elements: search bar, log in and sign up buttons, or user status if the user is logged in. The course page display the information about a course in neat and a stylish theme. Many pages follow the same uniform code skeleton which gave us the opportunity to debug and add features to it easily. For a dynamic page flow, we incorporated the features of templates and static files in Flask. This way, repetitive elements in pages are automatically included upon rendering the page. All relative hyperlinks are not hardcoded rather it uses the url_for() function that gets the correct paths no matter which page. Also, we used the powerful feature of templates to add for loops, if statements, and variables to HTML pages. All this embedded code executes once the user loads the page.

User Story #14

As a user, I would like to log in/sign up from any webpage in *easyA*.

	Task	Estimated time	Owner
1	Set up a navigation bar that contains the login and sign up buttons	2 hrs	Arshpreet
2	Set up necessary forms for the user to submit their information	4 hrs	Arshpreet
3	Make buttons disappear according the user's status if they were logged in or not	4 hrs	Arshpreet
4	Test that the buttons work as expected on all pages	1 hr	Luciano, Arshpreet

Justification for completion:

As a follow-up approach of the user friendly interface, we included the login and signup button to all pages to minimize the user clicks on reaching them. Minimizing user clicks is user experience approach to avoid confusion in navigating the website which is one of our goals in this project.

User Story #15

As a user, I would like to log out from any webpage in *easyA*.

	Task	Estimated time	Owner
1	For logged in users, make the account menu that contains the log out button visible on all pages	2 hrs	Arshpreet
2	Check the status of the users if they are logged in or not	2 hrs	Arshpreet
3	Make the account email and account menu button disappear after logging out	2 hrs	Arshpreet
4	Test that logging out works properly on all pages	1 hr	Luciano, Ashpreet

Justification for completion:

As a follow-up on user story #14, if the user is logged in, the user status and the account menu is viewed from all pages in the website. We incorporated user sessions on the page templates to display the user status accordingly if the user is in session (logged in) or not. This user story is different that user story #5 since this user story focuses on the front-end functionality and ease of use; including the log out button on all pages will avoid unnecessary navigation between pages to reach the log out button e.g. going back to the homepage just to click on log out.

What did not go well?

User Story #7

As a user, I would like to reset my password if I forget it.

6	Debug and test implementations using manual testing and unit tests.	5 hrs	Kaushik

User Story #10

As a user, I would like to know that all reviews are from Purdue students.

5	Debug and test the implementations using unit testing for each task.	27 hrs	Sadiq, Pratyaksh, Kaushik

User Story #11

As an administrator, I would like only Purdue students with "@purdue.edu" emails to be able to sign up.

4	Debug and test the implementations using unit testing for each task.	12 hrs	Sadiq, Pratyaksh, Kaushik

User Story #12

As an administrator, I would like newly signed up users to verify their ownership of their Purdue emails.

5	Debug and test the implementations using unit testing for each task.	16 hrs	Sadiq, Pratyaksh, Kaushik

Justification for not completion:

All testing in Sprint 1 was done manually and exhaustively. These tasks were partly completed because we did not implement unit testing due to spending most of our time on researching and implementing the tasks. Implementing unit testing for Flask or Firebase requires extra researching since they provide their own APIs to create a testing environment.

User Story #12

As an administrator, I would like newly signed up users to verify their ownership of their Purdue emails.

	Task	Estimated time	Owner
3	Give an error to the user in case they use an non-existing email, login unverified, or use an expired verification link.	4 hrs	Luciano
4	Delete unverified users after their confirmation links expire.	4 hrs	Arshpreet, Sadiq

Justification for not completion:

Most of task 3 is implemented except for checking if the email exists. Our intentions were to use an API that tests any inputted email by connecting to its mail server and tests if the mailbox exists there or not. The reasons we did not implement this feature is because of how inconsistent its results are which is probably due to the qualities mail server of Purdue. In addition, the API takes around 5 seconds to produce a results which only slows down the process of signing up.

How should we improve?

Our goal in sprint 1 was setting up the main foundations of the project. That is by setting up the file organization, the needed environment, version control, etc. All these steps need close communication between the team members because they are major tasks that affect everyone. In the first week, due to our lack of experience in Python and Flask, we changed our file organization twice. In the first organization, we thought we needed to use Flask in a single Python module. Unfortunately, the initial documentation of Flask suggested using this approach. We were wrong because this project is a contribution of five members and will require many lines of code. Therefore, we needed to use a Python package of multiple scripts interconnected together as the whole program. So, we followed this approach: https://exploreflask.com/en/latest/organizing.html. With these decisions being tentative in the first week, some team members did not understand the file hierarchy and where to get started. This is because the decisions were made without proper communication.

Another result of miscommunications was during all the three weeks. Although each team member understands their roles, we still worked on others' bugs without informing the task owner with the changes being made. This way, we worked on the same bugs/tasks at the same time redundantly. For the next sprint, we hope to communicate our progress and tasks more frequently on our Slack channel by announcing what task we are working on. In addition, for any major decision that would affect everyone, we should discuss it either on Slack or in meetings and bring every teammate up to date with the justifications behind such a decision.

Another issue that we could improve in the next sprint is to use branches in Github. In sprint 1, some team members worked directly on the master branch while others incorporated a branch for each step or task. Unsolved conflicts happened when the latter merged their branch with the master branch before the former pushed their changes. In each meeting, we had to solve these conflict issues and spend some time recloning and reviewing the code. Some implementations get lost because of this which made us spend more time re-implementing them back. In addition, when we worked on branches, we merged the changes once multiple features were implemented. Because of this, some teammates wait for others' to push their changes because their task depends on them. For sprint 1, we could improve by understanding version control and communicate more clearly through it. We hope to work on branches and maintain the agile methodology of releasing frequently once a feature is implemented.