SW Engineering CSC648 - 848 Summer 2021

Milestone 4

UniTea

Team 04

Name	**Role**	**Email**
Kiara Gil	Team Lead / Github Master	kgil1@mail.sfsu.edu
Joshua Stone	Front End Lead	
Ostyn Sy	Back End Lead	
Cong Le	Team Member	
Melinda Yee	Team Member	
Miho Shimizu	Team Member	
Vernon Xie	Team Member	

History Table

Milestone / Version	**Date**
M4V2	8/4/2021
M4V1	7/30/2021
M3V2	8/4/2021
M3V1	7/22/2021
M2V2	8/4/2021
M2V1	07/06/2021
M1V2	08/04/2021
M1V1	06/14/2021

Table of Content

Section 1: Product Summary

Name of Product: UniTea

Priority 1:

General Users:

General:

- General users shall be able to access the Registration page.
- General users shall be able to access the Login page.
- General users shall be able to create an account and become a Registered User.
- General users shall be able to access the About Us page.
- General users shall be able to access the Contact page.
- General users shall be able to access the Home page.
- General users shall be able to search for public study groups.
- General users shall be able to browse through study group listings.

Registered Users:

General:

- Registered users shall inherit all privileges of the General User.
- Registered users shall be able to upload profile pictures.
- Registered users shall be able to view other users' profiles.
- Registered users shall be able to join public study groups.
- Registered users shall be able to create public study groups.
- Registered users shall be able to edit information on their Profile Page.
- Registered users shall be able to change their password.
- Registered users shall be able to log in.
- Registered users shall be able to log out.
- Registered users shall be able to delete their account.

Forum:

- Registered users shall be able to create posts.
- Registered users shall be able to add a title to the post.
- Registered users shall be able to add a description to the post.
- Registered users shall be able to comment on posts.
- Registered users shall be able to delete their posts.

- Registered users shall be able to edit their posts.
- Registered users shall be able to edit their comments.

Host:

- Registered users that create a group shall have the status of host.
- Hosts shall be able to rename the group.
- Hosts shall be able to delete study groups that they have created.

Group Members:

- Registered users that join a group shall have the status of a group member.

Study Groups:

- Registered users shall be able to create study groups, which can hold up to 20 students.
- Registered users shall be able to join study groups.
- Registered users shall be able to leave study groups.

Administrative User:

General:

- Administrative users shall inherit all privileges from other users.
- Administrative users shall be able to change users' usernames.
- Administrative users shall be able to delete any study groups.
- Administrative users shall be able to delete any comments.

General:

Forum:

- Forum shall display the titles of the posts and short descriptions.
- Forum shall display posts.
- Forum posts shall display comments.
- Forum shall display usernames of users that post/comment.
- Forum posts shall display the content of the discussion.

Study Group:

- Study group shall be listed in a bulletin displaying the group name.
- Study group shall display the host’s role.
- Study group shall display the subject (and major, if applicable).
- Study group shall display the host username.
- Study group shall display the usernames of group members to other group members.
- Study group shall display the number of users in the group.

- Study group shall display a group forum.

Website:

- Website shall have a Registration page.
- Website shall have a Login page.
- Website shall have an About Us page.
- Website shall have a Contact page.
- Website shall have a Terms of Service page.
- Website shall prompt users to accept the terms and conditions.
- Website shall have the UniTea logo on each page.
- Website shall display the study groups a user has joined.

Unique:

UniTea is unique from other competitors because we offer a sleek dashboard that displays the user's group related info and allows for editing. Another functionality that we offer is the Educator Role. This specific role enables a registered user to create a larger group, thus allowing the host to accommodate more "students".

URL: http://ec2-54-219-170-81.us-west-1.compute.amazonaws.com/landing

Section 2: Usability Test Plan

Test Objectives:

These tests will determine the ease of use of UniTea through the following tasks: searching for a study group, creating a study group, posting a comment in a study group post, interacting with other users in the forums, and user dashboard comprehension. The purpose of UniTea is to create a platform where students can coordinate peer learning for a better learning experience.

While not all students share the same level of tech literacy, UniTea aims to make the site easy to use by using simplistic labels and large buttons on pages, as well as a Navigation bar that has shortcuts to help the User use most of UniTea's functionalities. If a User has been referred to UniTea by another User, they can use the search functionality to quickly locate a specific group they may be trying to join. If the User simply wishes to browse the platform, they can also use the search bar to find groups that may interest them. If the User cannot find any groups, they can either join a recommended group or create their own group for others to join. While within the group they can communicate with other group members by creating posts or leaving comments on postings that other Users have created. If the User wishes to interact with other Users regarding non-academic matters, they can use the forums to interact with one another by creating forum posts or leaving comments on posts by other Users. Over time, a User may lose track of which forums and groups they are participating in, and can view the user dashboard to see which groups and forums they have participated in.

Test Description:

System Setup:
The User will conduct the tests on their personal devices or on a device provided by the Test Administrator. Tests will be conducted on either a Windows Desktop, Windows Laptop, Apple Desktop, Apple Macbook, Apple iPhone or an Android Device. Testers will use one of the three web browsers: Google Chrome, Mozilla Firefox, or Safari.

Starting Point:
The Tester will be provided with a URL that will take them to the landing page of UniTea, unless stated otherwise in the Machine State of a specific test. The Tester will not be provided with any login credentials, and will be logged out before performing any task. In the event of a hosting issue or an outage of the service provider, the Test Administrator will deploy a local instance of the website on their machine and administer the test there, or wait for the outage to be resolved.

Intended Users:
The intended user is a college student who has basic technical literacy, and is comfortable using a web browser.

URL of system:
http://ec2-54-219-170-81.us-west-1.compute.amazonaws.com/landing

Metrics to be measured:

- Amount of time required to complete each task
- Level of frustration upon completion of each task
- Level of confusion to complete each task
- Level of difficulty in learning how to accomplish each task
- Amount of times the Test Administrator was asked to clarify a function

Usability Task Description:

The tester will be tasked with searching for a study group, creating a study group, posting a comment in a study group post, interacting with other users in the forums, and reading the user dashboard. Throughout the test, the tester will also be expected to register an account and login.

Task 1	**Description**
Task	Count how many study groups have the word “test” in their name
Machine State	At Landing Page
Successful Completion Criteria	Returns an accurate count of groups that have the word “test” in their name
Benchmark	30 seconds

Task 2	**Description**
Task	Join the group that your friend Josh2 made. It’s called “123”
Machine State	At Landing Page
Successful Completion Criteria	Registers and Logs into an account. Searches the group “123” and joins.
Benchmark	2 minutes

Task 3	**Description**
Task	Make a study group for a Spanish 1 Class
Machine State	At Home Page
Successful Completion Criteria	Successful creation of a study group for Spanish 1 found in the language section.
Benchmark	3 minutes

Task 4	**Description**
Task	Make a post to ask for help on the notional Assignment 4
Machine State	At Home Page
Successful Completion Criteria	Successful creation of a study group forum post for the study group made in the previous task
Benchmark	3 minutes

Task 5	**Description**
Task	Post a comment in a study group forum post made previously for the notional Assignment 4
Machine State	At Home Page
Successful Completion Criteria	Comment is successfully posted in the comment section of the correct study group forum post.
Benchmark	2 minutes

Task 6	**Description**
Task	Update the comment made in the previous step to include your favorite color.
Machine State	At Study Group Post Page from previous step
Successful Completion Criteria	Comment is successfully updated in the comment section of the correct study group forum post.
Benchmark	1 minute

Task 7	**Description**
Task	Delete the Study Group Post regarding the notional Assignment 4
Machine State	At Study Group Post Page from previous step
Successful Completion Criteria	Study Group Forum Post is successfully deleted from the study group section.
Benchmark	1 minute

Task 8	**Description**
Task	Count the number of study groups you belong in
Machine State	At User Dashboard
Successful Completion Criteria	Accurately returns the count of study groups the tester has joined by using the user dashboard
Benchmark	2 minutes

Task 9	**Description**
Task	Count the number of study groups you are the host of
Machine State	At User Dashboard
Successful Completion Criteria	Accurately returns the count of study groups the tester has joined is the host of by using the dashboard
Benchmark	2 minutes

Task 10	**Description**
Task	Share something about yourself in your profile
Machine State	At User Dashboard
Successful Completion Criteria	Modify their account bio
Benchmark	2 minutes

Task 11	**Description**
Task	Edit your username
Machine State	At User Dashboard
Successful Completion Criteria	Modify their username
Benchmark	1 minute

Task 12	**Description**
Task	Make use of the Forums and Introduce yourself
Machine State	At User Dashboard
Successful Completion Criteria	Create a Forum Post
Benchmark	3 minutes

Task 13	**Description**
Task	Find the user who made the forum post “test92 forum post”
Machine State	At Landing Page
Successful Completion Criteria	Return of the correct username who created the “test92 forum post”
Benchmark	2 minutes

Task 14	**Description**
Task	Go to the forum and find an educator’s post
Machine State	At User Dashboard
Successful Completion Criteria	Successfully find the educator’s user name
Benchmark	2 minutes

Task 15	**Description**
Task	Count the number of groups hosted by miho2_edu
Machine State	Main Forum Page
Successful Completion Criteria	Successfully return the number of study groups that are hosted by miho2_edu
Benchmark	2 minutes

Usability Test Table:

Test / Use Case	% Completed	Errors	Comments / Feedback	Average Time To Complete Test
1	100%	• Searches with no results sometimes return random strings	• Poor layout and design • "About" should be after "Forum" • Search is not robust enough as it cannot handle fuzzy searches • If you search for something and can't find it, the sections are dumped and have weird text • Search results don't show enough information • Search bar should be on the right • Search results do not return in a logical order • If you click on a subject to look for a group then you have to then go return to create a group. Maybe it would be good to have a create group option at the top of that page as well in case you can't find a group and then need to make one.	40 seconds
2	100%		• After making an account it doesn't automatically log me in	2 minutes
3	100%	• Changing group name to a name that already exists causes an error • You can create a group without selecting a subject • Making a new group with an existing name says successful but does not create the new group	• I was able to make a group without a subject • Why doesn't it auto select the subject for me if I'm already viewing one	3 minutes
4	100%			2 minutes
5	100%		• Comment text window was off to the left and hard to see/find	1 minute
6	100%			1 minute
7	100%			1 minute

Usability Test Table (Continued):

Test / Use Case	% Completed	Errors	Comments / Feedback	Average Time To Complete Test
8	100%			1 minute
9	100%		● The tabs aren't very apparent	1 minute
10	100%			2 minutes
11	100%	● Cannot add/modify profile picture	● I can't add or change my profile picture	1 minute
12	100%			2 minutes
13	90%		● What if I want to search for a forum and not a study group? ● It doesn't seem good that I can see everyone's study group posts ● It was a little challenging to find text92. No easy way to find forum posts ● When looking at the creator of a post it is not easily distinguishable who is ● the creator because it is not labeled	2 minutes 30 seconds One case not counted as tester could not complete the task
14	100%			1 minute 30 seconds
15	100%			1 minute 30 seconds

Overall Questionnaire Results:

#	Prompt	Average Rating
1	The search bar was easy to find	5
2	The search bar was easy to interact with	5
3	The search bar responded in a natural way	5
4	Search results were laid out in an easy to read manner	4
5	The search bar provided the results I was looking for or recommended something similar.	4
6	The search results were consistent and as expected	5
7	It was easy to create a group.	4
8	It was easy to figure out where to create a group	4
9	It was easy to make changes to my study group	5
10	It was easy to create a study group post	5
11	It was easy to modify my post	5
12	The study group forum posts are organized in a neat manner	3
13	It was easy to make a comment	5
14	It was easy to find my own comment	5
15	The comments are displayed in a easy to read manner	5
16	It was easy to edit my own comment	5
17	It was easy to see who wrote which comments	5
18	The comments are sorted properly	3
19	It was easy to find my study group	4
20	It was easy to find my study group post	5
21	It was easy to delete my study group post	5
22	It was easy to read the User Dashboard	4
23	It was easy to navigate the User Dashboard	4

Overall Questionnaire Results (continued):

#	Prompt	Average Rating
24	It was easy to find my study groups in the User Dashboard	5
25	It was easy to see which groups I host in the User Dashboard	5
26	It was easy to find my Forum Posts in the User Dashboard	5
27	The User Dashboard layout feels intuitive (Things were where I expected them to be)	4
28	It was easy to modify my account information	3
29	It was easy to add a user biography	3
30	It was easy to add a profile picture	1
31	It was intuitive to change my username	3
32	It was intuitive to change my password	3
33	It was easy to delete my account	5
34	It was easy to navigate to the forums	5
35	The Forums was easy to find	4
36	It was easy to make a forum post	5
37	It was easy to edit a forum post	5
38	The forums display posts in an organized manner	4
39	It was easy to view forum posts	4
40	It was easy to find the forum post author	5
41	It was easy to find the Educator’s forum post	5
42	It was easy to find forum posts	1
43	It was easy to view other Users profiles	5
44	It was easy to see what groups another User belongs to	4
45	It was easy to see posts another User has made	4

Section 3: QA Test Plan

40. The website shall allow users to interact with other users using the forum.

Test Objectives:

Check post creation functionalities.

HW and SW setup:

- Macbook, Desktop PC, iPhone.
- Web Browsers: Chrome, Firefox, Safari.

Feature to be tested:

Main forum and study group forum post and comment.

Test cases:

- Create a main forum post.
- Create a study group forum post.
- Create a comment to a study group forum post.

50. The website shall be capable of being updated in a timely manner.

Test Objectives:

Check if the website reflects changes.

HW and SW setup:

- Macbook, Desktop PC, iPhone.
- Web Browsers: Chrome, Firefox, Safari.

Feature to be tested:

Post and comment editing functionalities.

Test cases:

- Edit a main forum post and check if another user can see the updated post.
- Edit a comment to a study group post and check if another user can see the updated post.
- Delete a comment to a study group post and check if another user can see changes.

51. The website shall be capable of interacting efficiently with the users.

Test Objectives:

Check if a user can navigate the website using buttons.

HW and SW setup:

- Macbook, Desktop PC, iPhone.
- Web Browsers: Chrome, Firefox, Safari.

Feature to be tested:

Buttons to navigate the website.

Test cases:

- Test if the Home button on the Nav Bar will direct the user to the study group page while logged in.

- Test if the Home button on the Nav Bar will direct the user to the landing page while not logged in.
- Test if the Search Bar will properly search “test" and display groups with “test” in their names.

82. The website shall generate error messages when users try to do unauthorized operations.

Test Objectives:

To check if the website displays error messages when users try to do unauthorized operations.

HW and SW setup:

- Macbook, Desktop PC, iPhone.
- Web Browsers: Chrome, Firefox, Safari.

Feature to be tested:

Check for Error messages.

Test cases:

- Try to create a main forum post without logging in.
- Try to create a study group without logging in.
- Try to create a comment to a study group forum post without logging in.

63. Each webpage shall not display unnecessary buttons and forms.

Test Objectives:

To check that the website layout and design choices follow a uncluttered and intuitive design.

HW and SW setup:

- Macbook, Desktop PC, iPhone.
- Web Browsers: Chrome, Firefox, Safari.

Feature to be tested:

Form design, Button Layouts, and Button Placement.

Test cases:

- Create a study group post, check for unnecessary buttons, form fields, or other artifacts.
- Edit the user profile, check for unnecessary buttons, form fields, or other artifacts.
- Make a comment on a forum post, edit that comment. check for unnecessary buttons, form fields, or other artifacts.

QA Testing Table:

Test #	NFR #	Description	Test Input	Expected Output	Chrome Result	Firefox Result	Safari Result
1	40	Create a main forum post.	title: 91test forum post post: post 91 test	Post is created	Pass	Pass	Pass
2	40	Create a study group forum post.	title: 91test sg post post: post 91 test	Post is created	Pass	Pass	Pass
3	40	Create a comment on the study group forum post from the previous step.	“This is a 91test study group comment”	Comment is created	Pass	Pass	Pass
4	50	Edit a main forum post and check if another user can see the updated post.	title: test92 forum post post: 92post 92 test	Post is updated	Pass	Pass	Pass
5	50	Join a Study Group, create a study group forum post comment, then edit the comment.	comment: test91 edited comment: 92test	Comment is updated	Pass	Pass	Pass
6	50	Delete the comment from the previous step	Delete comment	Comment is removed	Fail	Fail	Fail
7	51	Test if the Home button on the Nav Bar will direct the user to the homepage while logged in.	Click the home button on the nav bar	Direct to Home Page / /home	Pass	Pass	Pass
8	51	Test if the Home button on the Nav Bar will direct the user to the landing page while not logged in.	Click the home button on the nav bar	Direct to Home Page / /home	Pass	Pass	Pass
9	51	Test if the Search Bar will properly search “test” and display groups with “group” in their names.	“test”	List of “test” groups /search	Pass	Pass	Pass
10	63	Create a study group post, check for unnecessary buttons, form fields, or other artifacts.	Title: “M4p3 test” Desc.: “haha”	No unnecessary buttons in focus, no artifacts on screen	Pass	Pass	Pass
11	63	Edit the user profile, check for unnecessary buttons, form fields, or other artifacts.	New username: newvernontest New description: I like pickleball 2	No unnecessary buttons in focus, no artifacts on screen	Pass	Pass	Pass

QA Testing Table (Continued):

Test #	NFR #	Description	Test Input	Expected Output	Chrome Result	Firefox Result	Safari Result
12	63	Make a comment on a forum post, edit that comment. check for unnecessary buttons, form fields, or other artifacts.	Comment 1: haha New Comment: ha 2	No unnecessary buttons in focus, no artifacts on screen	Pass	Pass	Pass
13	82	Try to create a main forum post without logging in.	Post Title: “Test Title” Post: “Test Post”	Error, Direct to Login Page /login	Pass	Pass	Pass
14	82	Try to create a study group without logging in	Post Title: “Test Group” Post: “Test Post” Subject: Math	Error, Direct to Login Page /login	Pass	Pass	Pass
15	82	Try to create a comment to a study group forum post without logging in.	Comment: “Test Comment”	Error, Direct to Login Page /login	Pass	Pass	Pass

Section 4: Code Review

Coding Style:

- Files will be named using camel case naming convention.
- Files will have headers with the filename, the team name, the URL of the github repository, and a brief file description.
- A simple naming convention will be adopted for all files, methods, classes, functions, and variables.
- Code will be written in a neat and organized manner.
- The code shall have proper comments.

Front End:

- Frontend code will utilize kebab case naming convention.
- Avoid inline styling when possible.

Back End:

- Backend code will utilize camel case naming convention.
- Avoid using global variables when possible.
- Do not use variables before their assignment.

User Dashboard Email Chain:

From: Miho Shimizu <mshimizu2@mail.sfsu.edu>
Date: Wednesday, July 28, 2021 at 10:47 PM
To: Vernon Chuan Xie <vxie1@mail.sfsu.edu>
Subject: T4 Code Review - User Dashboard

Hello Vernon,

Please review these file(s) under Commit ID: 1e59578. This is for the User Dashboard.

- views.py 338-358

Best Regards,

Miho Shimizu

From: Vernon Chuan Xie <vxie1@mail.sfsu.edu>
Sent: Thursday, July 29, 2021 2:56 PM
To: Miho Shimizu <mshimizu2@mail.sfsu.edu>
Subject: Re: T4 Code Review - User Dashboard

Hello Miho,

I've reviewed the code and have tested it. Everything looks good. Where is the StudyGroup Function and StudyGroupMember Function defined?

Respectfully Sent,

Vernon Xie

From: Miho Shimizu <mshimizu2@mail.sfsu.edu>
Date: Wednesday, July 28, 2021 at 11:02 PM
To: Vernon Chuan Xie <vxie1@mail.sfsu.edu>
Subject: Re: T4 Code Review - User Dashboard

Hello,

Do you mean StudyGroup class and StudyGroupMember class? They are defined in models.py, line 84 - 97 and 126-131, respectively.

Best Regards,

Miho Shimizu

From: Miho Shimizu <mshimizu2@mail.sfsu.edu>
Date: Wednesday, July 28, 2021 at 11:07:51 PM
To: Vernon Chuan Xie <vxie1@mail.sfsu.edu>
Subject: Re: T4 Code Review - User Dashboard

Hello Miho,

Everything looks good! Keep up the good work

Respectfully Sent,

Vernon Xie

views.py 338-358

```
337  # show a specified user's profile
338  def showUserProfile(request, userId):
339      #fetch user information
340      user = User.objects.get(userId=userId)
341
342      #fetch information of groups the user joined
343      studyGroupMembers = StudyGroupMember.objects.filter(userId=request.user.userId)
344      studyGroupIds = []
345      for studyGroupMember in studyGroupMembers:
346          studyGroupIds.append(studyGroupMember.studyGroupId.studyGroupId)
347      studyGroupsAsMember = StudyGroup.objects.filter(studyGroupId__in=studyGroupIds).order_by('-studyGroupId')
348
349      #fetch information of groups hosted by the user
350      studyGroupsAsHost = StudyGroup.objects.filter(ownerId=user).order_by('-studyGroupId')
351
352      #fetch information of the user's main forum posts
353      mainPosts = MainPost.objects.filter(userId=user).order_by('-postDateTime')
354
355      #fetch information of the user's study group forum posts
356      studyGroupPosts = StudyGroupPost.objects.filter(userId=userId).order_by('-postDateTime')
357
358      return render(request, 'userProfile.html', {'userprofile': user, 'studyGroupsAsMember': studyGroupsAsMember, 'studyGroupsAsHost': studyGroupsAsHost, 'mainPo
359
```

models.py lines 83-132

```
83  # Study Group Model
84  class StudyGroup(models.Model):
85          studyGroupId = models.AutoField(primary_key=True, unique=True)
86          groupName = models.CharField(max_length=100)
87          description = models.CharField(max_length=5000)
88          groupType = models.CharField(max_length=10, choices=GROUP_TYPE_CHOICES, default='general')
89          memberCount = models.IntegerField(default=0)
90          subject = models.CharField(max_length=100, blank=True, null=True)
91          ownerId = models.ForeignKey(User, on_delete=models.CASCADE)
92
93          def isFull(self):
94                  return True if self.memberCount >= STUDY_GROUP_CAPACITY else False
95
96          class Meta:
97                  db_table = "studygroups";
98
99
100 # Study Group Post Model
101 class StudyGroupPost(models.Model):
102         postId = models.AutoField(primary_key=True, unique=True)
103         postTitle = models.CharField(max_length=100)
104         post = models.CharField(max_length=5000)
105         postDateTime = models.DateTimeField(auto_now=True)
106         userId = models.ForeignKey(User, on_delete=models.CASCADE)
107         studyGroupId = models.ForeignKey(StudyGroup, on_delete=models.CASCADE)
108
109         class Meta:
110                 db_table = "studygroupposts";
111
112
113 # Study Group Comment Model
114 class StudyGroupComment(models.Model):
115         commentId = models.AutoField(primary_key=True, unique=True)
116         comment = models.CharField(max_length=5000)
117         commentDateTime = models.DateTimeField(auto_now=True)
118         userId = models.ForeignKey(User, on_delete=models.CASCADE)
119         postId = models.ForeignKey(StudyGroupPost, on_delete=models.CASCADE)
120
121         class Meta:
122                 db_table = "studygroupcomments";
123
124
125 # Study Group Member Model
126 class StudyGroupMember(models.Model):
127         userId = models.ForeignKey(User, on_delete=models.CASCADE)
128         studyGroupId = models.ForeignKey(StudyGroup, on_delete=models.CASCADE)
129
130         class Meta:
131                 db_table = "studygroupmembers";
132
```

Home Dashboard Email Chain:

From: Miho Shimizu <mshimizu2@mail.sfsu.edu>
Date: Wednesday, July 28, 2021 at 10:53 PM
To: Vernon Chuan Xie <vxie1@mail.sfsu.edu>
Subject: T4 Code Review - Home Dashboard

Hello Vernon,

Please review these file(s) under Commit ID: 1e59578. This is for the Home Dashboard.

- views.py lines 35-50

Best Regards,

Miho Shimizu

From: Vernon Chuan Xie <vxie1@mail.sfsu.edu>
Sent: Thursday, July 29, 2021 3:13 PM
To: Miho Shimizu <mshimizu2@mail.sfsu.edu>
Subject: Re: T4 Code Review - Home Dashboard

Hello Miho,

The code looks good, I've tested it and confirmed its functionality. Can you further elaborate on what you are doing from lines 44 to 48?

Respectfully Sent,

Vernon Xie

From: Miho Shimizu <mshimizu2@mail.sfsu.edu>
Date: Wednesday, July 28, 2021 at 11:36 PM
To: Vernon Chuan Xie <vxie1@mail.sfsu.edu>
Subject: Re: T4 Code Review - Home Dashboard

Hello,

The purpose of those lines of code is to show the groups that the user has joined or are hosting. If the user has joined 4 groups or more, the dashboard will show the 3 most recently joined

groups, and then the 2 most recent groups that they are hosting. Otherwise, it will display up to 5 of the groups that they are currently hosting.

Best Regards,

Miho Shimizu

From: Vernon Chuan Xie <vxie1@mail.sfsu.edu>
Date: Wednesday, July 28, 2021 at 11:41:21 PM
To: Miho Shimizu <mshimizu2@mail.sfsu.edu>
Subject: Re: T4 Code Review - Home Dashboard

Hello Miho,

Okay, please include some comments into the code that will reflect the functionality in those lines of code. Keep up the good work!

Respectfully Sent,

Vernon Xie

views.py lines 35-50

```
34  # show homepage
35  def home(request):
36      #fetch groups the user has joined
37      studyGroupMembers = StudyGroupMember.objects.filter(userId=request.user.userId)[:5]
38      studyGroupIds = []
39      for studyGroupMember in studyGroupMembers:
40          studyGroupIds.append(studyGroupMember.studyGroupId.studyGroupId)
41      studyGroupsAsMember = StudyGroup.objects.filter(studyGroupId__in=studyGroupIds).order_by('-studyGroupId')
42
43      #fetch groups whose host is the user
44      if len(studyGroupIds) > 3:
45          studyGroupsAsHost = StudyGroup.objects.filter(ownerId=request.user).order_by('-studyGroupId')[:2]
46          studyGroupsAsMember = studyGroupsAsMember[:(5-len(studyGroupsAsHost))]
47      else:
48          studyGroupsAsHost = StudyGroup.objects.filter(ownerId=request.user).order_by('-studyGroupId')[:(5-len(studyGroupIds))]
49
50      return render(request, 'index.html', {'studyGroupsAsMember': studyGroupsAsMember, 'studyGroupsAsHost': studyGroupsAsHost})
```

views.py lines 43-52 (revised with new comments)

```
43      #if the user belongs to more than 3 groups, display the 3 most recent groups, then 2 most
        recent they host
44      if len(studyGroupIds) > 3:
45          studyGroupsAsHost = StudyGroup.objects.filter(ownerId=request.user).order_by('
                -studyGroupId')[:2]
46          studyGroupsAsMember = studyGroupsAsMember[:(5-len(studyGroupsAsHost))]
47      #else display up to the 5 most recent groups they host
48      else:
49          studyGroupsAsHost = StudyGroup.objects.filter(ownerId=request.user).order_by('
                -studyGroupId')[:(5-len(studyGroupIds))]
50
51      return render(request, 'index.html', {'studyGroupsAsMember': studyGroupsAsMember, '
            studyGroupsAsHost': studyGroupsAsHost})
52
```

Section 5: Self-check on Best Practices for Security

On UniTeam we are protecting the following:

- User passwords

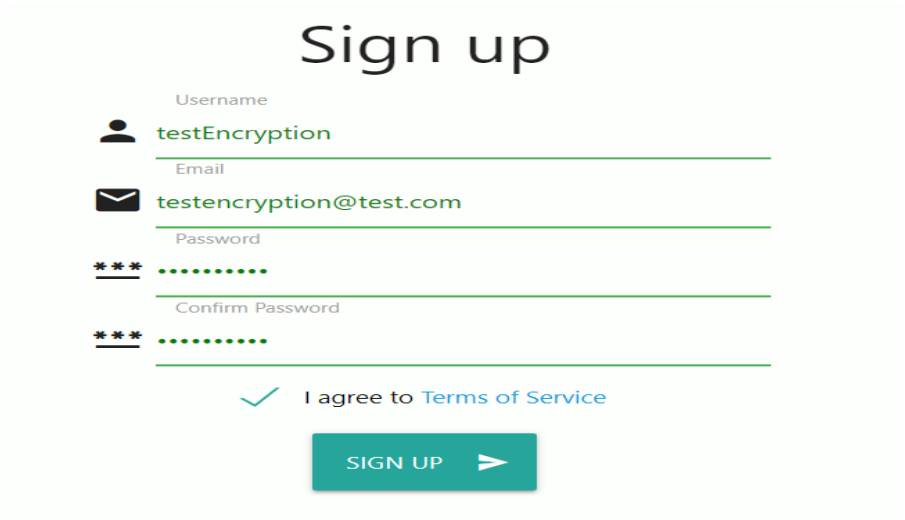
For the passwords, we are using Django's built-in encryption which includes hashing and random salting. Django uses the PBKDF2 algorithm with a SHA256 hash and a password stretching mechanism.

Below is the call to Django's encryption function where our users passwords are encrypted:

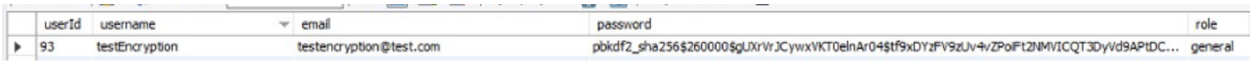
```
-  6|  from django.contrib.auth.hashers import make_password
-  140|      user.password = make_password(user.password)
```

For confirmation that our password encryption is effective, we provide the screenshots below. The first is the form a new user fills out when they register for UniTea and the second is an image of the user's encrypted password stored in our database.

Registration form:



Encrypted password:



userId	username	email	password	role
93	testEncryption	testencryption@test.com	pbkdf2_sha256$260000$gUXrVrJCywxVKT0elnAr04$tf9xDYzFV9zUv4vZPoiFt2NMVICQT3DyVd9APtDC...	general

For the registration information, we are validating the inputs using the validators provided in the Django framework and a few of our own written in python.

The email, username, and passwords are all being validated.

Emails provided by users during registration are validated to make sure they are proper emails - i.e they have the '@' and a '.<domain>'. and that they do not currently exist in our database. The Django API let's us specify a textfield as being an email input in models.py. Django automatically handles all things to do with identifying an email address in an input in model.py.

```
    email = models.EmailField(max_length=100, unique=True)
```

In our view.py, we have a boolean that holds the result of a reference to our database that checks if the email is already in our database.

```
email_exist = User.objects.filter(email=form.cleaned_data['email'])
```

Usernames are validated to make sure that they do not currently exist in our database in views.py. They can be alphanumeric and include symbols. In our view.py, we have a boolean that holds the result of a reference to our database that checks if the username is already in our database.

```
username_exist = User.objects.filter(email=form.cleaned_data['username'])
```

The segment below is where all the actual email and username validation occurs in views.py. The booleans are stored and used by registered.html to control error messages.

```
                # if make it here, means password is good, but either
username/email is taken

                if username_exist or email_exist:

                    return render(request, 'register.html',

                                  {'form': form,

                                   'error': True,

                                   'valMessages': False,

                                   'user_error': username_exist,

                                   'email_error': email_exist})
```

Passwords are validated to make sure they have a minimum of 8 characters, 1 upper-case, 1 lower-case, 1 digit, and 1 symbol. We created our own validators in python for the latter 4 and used a built-in Django function for the minimum length. Below, we made sure to raise a ValidationError in Django so we can use the messages in views.py and eventually our registration.html for the user to see on the frontend.

```
import re

from django.core.exceptions import ValidationError

from django.utils.translation import ugettext as _

class NumberValidator(object):

    def validate(self, password, user=None):

        if not re.findall('\d', password):

            raise ValidationError(

                _("The password must contain at least 1 number."),

                code='password_no_number',

            )

class UppercaseValidator(object):

    def validate(self, password, user=None):

        if not re.findall('[A-Z]', password):

            raise ValidationError(

                _("The password must contain at least 1 uppercase letter."),

                code='password_no_upper',

            )

class LowercaseValidator(object):

    def validate(self, password, user=None):

        if not re.findall('[a-z]', password):
```

```
            raise ValidationError(
                _("The password must contain at least 1 lowercase letter."),
                code='password_no_lower',
            )
class SymbolValidator(object):
    def validate(self, password, user=None):
        if not re.findall('[()[\]{}|\\`~!@#$%^&*_\-+=;:\'",<>./?]', password):
            raise ValidationError(
                _("The password must contain at least 1 symbol."),
                code='password_no_symbol',
            )
```

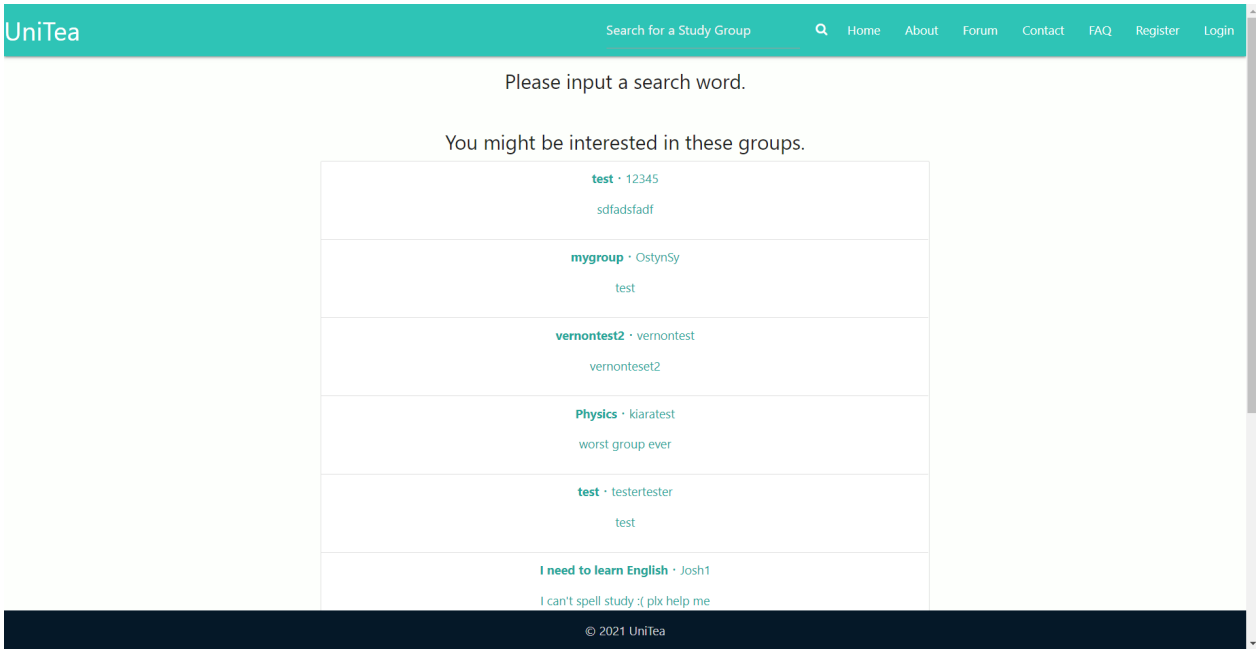
In settings.py, we had to make sure we specify a correct path to our validators in validators.py in the AUTH_PASSWORD_VALIDATORS section. All of these validators will be invoked by simply calling validate_password(<string>,<object>) on the text being input as a password. Django then runs the list of specified functions on the string provided.

```
AUTH_PASSWORD_VALIDATORS = [
    {   "NAME":"django.contrib.auth.password_validation.MinimumLengthValidator",
        "OPTIONS": { "min_length": 8,}, },
    {   "NAME": "study_app.validators.UppercaseValidator",},
    {   "NAME": "study_app.validators.LowercaseValidator",},
    {   "NAME": "study_app.validators.SymbolValidator",},
    {   "NAME": "study_app.validators.NumberValidator",},
]
```

The segment below is where all the actual password validation occurs in views.py. We have a try- except block. We try the validations, and if the password doesn't mean one of them, the function sends back a validation error. All the errors are stored in val_err and then the list of errors is stored in valMessages. In register.html, we have a loop that prints out all the messages in valMessages onto our front end for the user to see.

```
            try:
                validation.validate_password(user.password, user)
            except ValidationError as val_err:
                return render(request, 'register.html',
                              {'form': form,
                               'error': True,
                               'valMessages': val_err.messages,
                               'user_error': username_exist,
                               'email_error': email_exist})
```

A well known security flaw in websites is submitting an empty search query in something like a search bar and getting back a lot more data than one should. At UniTea, we not only protect against empty queries in our search bar, but we also guide our users to relevant study groups when they attempt an empty search or any search that doesn't return results. Below is a screenshot of what a user will see when they attempt an empty search in our search bar:



We not only felt that search bar validation was important, but also from a UX standpoint, we knew that we needed a way to give a user some suggestions for study groups when their search doesn't bring back any results. We achieve this with our searchStudyGroups() function which I will further elaborate on.

In this function, if the search input matches a group name already in our database, our function will make found = true and return the matched results. If we can not find a study group whose name matches the user's search, then we try to match the search input to a study group description instead. If our algorithm is successful at this, it will make maybe = true and return the results. This process can be seen on the next page:

```python
# search study groups
def searchStudyGroups(request):
    found = False
    maybe = False
    foundStudyGroups = None
    maybeStudyGroups = None
    suggestStudyGroups = None

    if request.method == "POST":
        searched = request.POST['searched']
        if searched:
            #find groups whose name contains the searched word
            foundStudyGroups = StudyGroup.objects.filter(groupName__icontains=searched)
            if foundStudyGroups:
                found = True
            else:
                #find groups whose description contains the searched word
                maybeStudyGroups = StudyGroup.objects.filter(description__icontains=searched)
                if len(maybeStudyGroups) > 10:
                    maybe = True
```

Lastly, if the user's search input doesn't match a study group name or study group description, our function will suggest popular study groups that still have room for new members. This way, the user always has an option to join a study group after every search, regardless of the results. This can be seen in our function below:

```python
if not found and not maybe:
    #no search word was inputted. suggest not-full popular groups
    suggestStudyGroups = StudyGroup.objects.filter(memberCount__lt=20).order_by('memberCount').reverse()[:10]

return render(request, 'searchResults.html', {'searched': searched, 'foundStudyGroups': foundStudyGroups, 'maybeStudyGroups': maybeStudyGroups, 'suggestStudyGroups': suggestStudyGroups})
```

Section 6: Self-check: Adherence to original Non-functional Specs

Functionality:

1. The website shall be implemented with the tools and frameworks that were approved by the CTO - **DONE**
2. The website shall be using Amazon Web Services for deployment. - **DONE**
3. The website shall be user friendly. - **DONE**
4. The website shall be easy to navigate. - **DONE**

Security:

5. Email and password shall be required upon login verification. - **DONE**
6. Username, password, and email shall be required to create an account. - **DONE**
7. Email and password shall be used to authenticate users. - **DONE**
8. The passwords shall be saved as encrypted. - **DONE**

Privacy:

9. Users will be required to accept policies before creating an account. - **DONE**
10. The Terms of Service shall be easily accessible. - **DONE**
11. Passwords and other personal information shall be kept hidden. - **DONE**
12. Study groups shall be visible to all users. - **DONE**
13. Study group members shall be visible to all users. - **DONE**
14. Comments shall be visible to all users. - **DONE**
15. The comment section shall display who writes the comment with date and time (PST). - **DONE**

Performance:

16. A user request on the website should be completed within 5 seconds. - **DONE**
17. The website should load within 5 seconds. - **DONE**
18. The website shall display study groups search results within 5 seconds. - **DONE**

System Requirements:

19. The website shall be implemented with the architecture and technologies that the CTO has approved. - **DONE**
20. The website shall support up to version 91.0.4472.106 of Google chrome. - **DONE**
21. The website shall support up to version 89.0.1 of Firefox. - **DONE**
22. The website shall support up to version 14.1.1 of Safari. - **DONE**
23. The website shall support desktops, and laptops operating on Windows 10, Ubuntu 20.04 LTS, or macOS 10.15 or newer, using the supported web browsers. - **DONE**

24. The website shall support mobile devices such as smartphones and tablets operating on AndroidOS 8.0 or newer, or Apple iOS 12 or newer, with a device resolution greater than 400px x 700px, using the supported browsers. - **DONE**

Marketing:

25. Each webpage shall display the application logo in the upper left corner. - **DONE**
26. Each webpage shall be clear and easy to understand for first time visitors. - **DONE**
27. There shall be a Contact Us page. - **DONE**
28. The Contact Us page shall be accessible from each webpage. - **DONE**
29. The footer shall have copyright information. - **DONE**
30. The signup page shall be easily accessible from any page while the user is not logged in. - **DONE**

Content:

31. The website shall have a navigation bar on top of a webpage. - **DONE**
32. The website shall have a search bar to search study groups. - **DONE**
33. The website shall present the most populated group in the search page. - **DONE**
34. The website shall have user profile pages for each user. - **DONE**
35. The user profile page shall present the username, the profile picture, and the study groups the user has posted and partake in. - **DONE**
36. The website shall have separate pages for each study group. - **DONE**
37. The website shall allow users to interact with other users using the forum. - **DONE**
38. The website shall allow users to upload a profile image. - **DONE**
39. The study group pages shall present group members. - **DONE**
40. Posted pictures shall be resized appropriately. - **DONE**
41. The website shall have a footer in each page at the bottom. - **DONE**

Scalability:

42. The website shall be able to handle 30,000 users. - **DONE**
43. The website shall be able to handle 50,000 study groups. - **DONE**
44. The website shall be developed using microservice architecture, contributing to the maintenance of the application. - **DONE**

Capability:

45. The website shall be capable of providing requested data by the user. - **DONE**
46. The website shall be updated in real time. - **DONE**
47. The website shall utiliza an intuitive layout and is easy to navigate. - **DONE**
48. The website shall be capable of recovering from failures. - **DONE**

Look and Feel:

49. The website shall have subtle colors. - **DONE**
50. The website shall have readable fonts. - **DONE**
51. The website shall have a simple layout. - **DONE**
52. The website shall be easy to navigate. - **DONE**
53. Study group content shall be immediately recognizable. - **DONE**
54. The usage of the website is intuitive. - **DONE**
55. The subject for which the study group is created shall be easily identifiable. - **DONE**
56. Users can easily recognize if they are members of the study group on the study group pages. - **DONE**
57. Users can easily identify the study groups they are in on their profile page. - **DONE**
58. The signup page, the login page, and the logout button shall be easily accessible. - **DONE**
59. Users can easily join in and leave from study groups. - **DONE**
60. Each webpage shall not display unnecessary buttons and forms. - **DONE**

Coding Standards:

61. The code shall use proper naming conventions, camel case and kebab case. - **DONE**
62. The code shall be understandable and have proper commenting. - **DONE**
63. The source code file shall have a header on top. - **DONE**
64. The source code header contains the filename, the team name, the URL of the github repository, and the brief description. - **DONE**
65. The code shall be maintained with one coding style. - **DONE**
66. The code shall have proper formatting. - **DONE**

Development:

67. The code shall be pushed and pulled from branches in git properly. - **DONE**
68. The code shall be maintained in the relevant branches. - **DONE**
69. The final code shall be maintained in the master branch. - **DONE**
70. The code in each branch in the remote repository must be successfully compiled and runnable. - **DONE**
71. The major update/changes of the code shall be reviewed by team members using the github functionalities. - **DONE**

Availability:

72. The website shall be active even if no users are active on the website. - **DONE**
73. The website shall generate error messages when the connection is lost. - **DONE**
74. The website shall generate error messages when users try to do unauthorized operations. - **DONE**

Fault tolerance:

75. The website shall be able to handle the display errors. - **DONE**

76. The website shall load all the content upon refresh after reconnecting to the server. - **DONE**
77. The website shall verify user input data both on the front end and the server side. - **DONE**
78. The website shall check if the user is authorized for each operation. - **DONE**

Storage:

79. Store study groups in the database. - **DONE**
80. Remove study groups from the database if the study group is deleted. - **DONE**
81. Store study group members in the database. - **DONE**
82. Remove all the study group members from the study group when the study group is deleted. - **DONE**
83. Remove users from a study group when the users leave the study group. - **DONE**
84. Store comments in the database. - **DONE**
85. Remove comments from the database when the study group is deleted. - **DONE**
86. Store usernames in the database. - **DONE**
87. Store emails in the database. - **DONE**
88. Store passwords in the database. - **DONE**
89. Store user types in the database. - **DONE**
90. Store profile pictures in the Amazon Web Services database. - **DONE**
91. Store the login user's cookie in the database. - **DONE**
92. Remove the login user's cookie session from the database. - **DONE**

Expected Load:

93. The website shall support 50,000 study groups. - **DONE**
94. The website shall support 30,000 users - **DONE**
95. The website shall support 5,000 concurrent users. - **DONE**
96. The website shall support 300,000 comments. - **DONE**

Legal:

97. The website shall have Terms and Conditions that users will be required to accept before they log in. - **DONE**
98. The website shall have a privacy policies section in the Terms of Service. - **DONE**
99. The website shall have copyright notice. - **DONE**
100. The website shall have user guidelines in the Terms of Service. - **DONE**

Section 7: List of Contributions

Kiara:

- Security Review
- Minor frontend fixes on several pages
- Production Server Maintenance
- Password Validation

Josh:

- Security Review
- FAQ Page
- Usability Test Plan
- QoL fixes such as adding forum to mobile nav bar, making home page buttons more clickable, and moving navbar logo slightly towards the left

Vernon:

- QA Testing
- Code Review
- Cleaning Up Previous Milestones
- Usability Test Plan
- Fixed navigation github issues

Miho:

- Code Review
- QA Testing
- User Group Dashboard
- Backend fixes
- Search Results Improvements
- Cleaning Up Previous Milestones
- Usability Test Plan
- Dashboard Implementation
- Educator Sign Up page and features

Melinda:

- Product Review
- FAQ Page
- Footer fixes
- Usability Test Plan
- Fixed Search Results Listings
- Overall front end fixes

Cong:

- Password Validation
- Frontend validation Messages
- Non-FR Checklist
- Cleaning Up Previous Milestones
- Dashboard idea
- Educator Sign Up page and features
- Backend fixes for several pages

Ostyn:

- Product Review
- Usability Test Plan
- Profile Page fixes
- Fixed backend search results
- Backend fixes for several pages