Product Backlog

Team 24 - Groupr

Alex Aralis, William Huang, Jeremy Lehman, Zach Perry, Aditya Vaidyam

Problem

At Purdue, computer science students enroll in many courses that require teamwork to build software. The current method of forming groups involves the use of Piazza, a Q&A web service that is not primarily built for team building. Students can only view existing teams, as well as other students looking for teams, by relying on those people to post a comment. Groupr aims to solve this problem by allowing students to easily organize and communicate on one platform, built specifically for forming teams for classes at Purdue.

Background

Many students find it hard to create well organized teams for classes that require group work. For students in CS307 and CS408, team creation options were limited to Piazza team-making, or professors automatically adding students to teams solely by which teams needed more people. We are building Groupr to help future students at Purdue get the most out of their project-based classes. Using Groupr, students and professors can create more valuable teams by matching members with similar interests and skills. This will lead to more efficient teamwork and less conflict between team members as they can build a project that everyone can find interesting and enjoyable.

Environment

The back end system will be built using Swift 3 and the Vapor web framework. We will also use PostgreSQL to access and store data. The front end web application will be built with React, a javascript library for building user interfaces.

Functional Requirements

Backlog Id	Functional Requirement	Hours	Status
1	As a user, I would like to select what class I am in	8	Completed in sprint 2
2	As a user, I would like to list my preferred programming languages	4	Incomplete
3	As a user, I would like to log in and sign out of Groupr	24	Completed in sprint 2
4	As a user, I would like to see other students in a selected class	4	Completed in sprint 2
5	As a user, I would like to be able to sign up	24	Completed in sprint 2
6	As a user, I would like to create a class	6	Completed in sprint 2
7	As a user, if I am a staff member, I would like to delete a class	6	Incomplete
8	As a user, I would like to leave a class	6	Completed in sprint 2
9	As a user, I would like to see profile information of students in my class	6	Completed in sprint 2
10	As a user, I would like to specify a type of expertise (web, mobile, backend)	6	Incomplete
11	As a user, I would like to communicate with class members	8	Completed in Sprint 2
12	As a user, I would like to communicate to the instructor	8	Incomplete
13	As a user, I would like to customize my personal bio.	4	Incomplete
14	As a user, I would like to set my status to be “looking for teammates”	4	Incomplete
15	As a user, I would like to be able to create a team	8	Completed in Sprint 2
16	As a user, I would like to be able to join a team	4	Completed in Sprint 2
17	As a user, I would like to be able to leave a team	4	Completed in Sprint 2

18	As a user, I would like to be able to delete a team	4	Incomplete
19	As a user, I would like to be an owner of a team	4	Incomplete
20	As a user, I would like to be able to group chat with my teammates	8	Incomplete
21	As a user, I would like to be automatically matched with a team	30	Incomplete
22	As a user, I would like to quickly undo my actions if I join or leave a group	2	Completed in Sprint 2
23	As a user, I would like to be able to change my name and career account saved to my account.	4	Completed in Sprint 2

Non-Functional Requirements

1. Security - Authentication with student emails will be used to prevent people from joining classes they aren't in.
2. Reliability - This has to work and be convenient for students to use instead of piazza, if there are problems with the app or downtimes where professors cannot see the groups, it defeats the entire purpose.
3. Scalability - CS classes can contain up to 200 students, a professor, and 5 to 10 TAs. Our app must be able to support this, along with creating groups to whatever the specifications the professor requires.

Use Case #1

for functional requirement 1

Case: Select Class

Action	System Response
1. Perform Use Case #2 procedure	
2. Click “Browse classes”	3. Display appropriate classes
4. Click “Enroll” on desired class	5. Enroll student and navigate to class home page

---

Use Case #2

for functional requirement 3

Case: Login

Action	System Response
1. Open App	2. Login Screen shows
3.1. Login (Student)	4.1. If in group, team members show. If not in group, available groups and members show
3.2. Login (Professor/Admin)	4.2. Active Class Groups Show

Use Case #3

for functional requirement 2

Case: Specify prefered programming languages

Action	System Response
1. Perform Use Case #2 procedure	
2. Click on user name displayed on navbar	3. Display user profile page
4. Click “Add languages” button	5. Display Add/remove language UI
6. Add/remove languages then click save	7. Save languages to the backend

---

Use Case #4 for functional requirement 3

Case: Logout

Action	System Response
1. Perform Use Case #2 procedure	
2. Click on user name displayed on navbar	3. Display user profile
4. Click “Logout” button	5. Log user out and navigate to home page

---

Use Case #5

for functional requirement 4

Case: View classmates

Action	System Response
1. Perform Use Case #1 procedure	
3. Return to home page by clicking logo on navbar	4. Display home page
5. Click “My classes” button	6. Display list of enrolled classes for user
7. Click on a class in the list	8. Display class home page with students listed

Use Case #6

for functional requirement 5

Case: Sign up as a staff member

Action	**System Response**
1. Open web page	2. Display home page
3. Click sign-up on navbar	4. Display sign-up page
5. Fill out sign-up page, check “Staff” box and click submit	6. Create staff user account log user and navigate to home page

Use Case #7

for functional requirement 5

Case: Sign up as a student

Action	**System Response**
1. Open web page	2. Display home page
3. Click sign-up on navbar	4. Display sign-up page
5. Fill out sign-up page, leave “Staff” checkbox unchecked and click submit	6. Create student user account log user and navigate to home page

---

Use Case #8

for functional requirement 6

Case: Create a class as a staff member

Action	System Response
1. Perform Use Case #6 procedure	
2. Click “create a class”	3. Display the create a class page
4. Fill out create a class form and click submit	5. Create class and navigate to class home page

---

Use Case #9

for functional requirement 7

Case: Delete class as a staff member

Action	System Response
1. Perform Use Case #5 procedure	
2. Click “Delete Class” button	3. Ask for confirmation
4. Confirm class deletion	5. Delete class and navigate to home page

---

Use Case #10

for functional requirement 8

Case: Leave class as student

Action	System Response
1. Perform Use Case #5 procedure	
2. Click “Leave Class” button	3. Ask for confirmation
4. Confirm leaving	5. Remove student from class and navigate to home page.

Use Case #11

for functional requirement 9

Case: Navigate to User Profile Page

Action	System Response
1. Perform Use Case #7 procedure	
2. Click user name on navbar	3. Display profile page

Use Case #12

for functional requirement 10

Case: Specify Expertise

Action	System Response
1. Perform Use Case #11 procedure	
2. Click “Set Expertise” button	3. Display expertise form
4. Specify expertise and submit	5. Update expertise and return to profile page

Use Case #13

for functional requirement 11

Case: Communicate with class member

Action	System Response
1. User 1 Perform Use Case #5 procedure	
2. User 1 Click “Chat” button next to desired class member User 2	3. Display to chat page for User 1
4. User 1 Communicate with class member via chat interface	5. Notify User 2 they are being contacted
6. User 2 clicks notification	7. Open chat page for User 2
8. Proceed with 2-way communication	

Use Case #14

for functional requirement 12

Case: Communicate with class Instructor

Action	System Response
1. User 1 Perform Use Case #5 procedure	
2. User 1 Click “Chat” button next to desired class Instructor User 2	3. Display to chat page for User 1
4. User 1 Communicate with Instructor via chat interface	5. Notify User 2 they are being contacted
6. User 2 clicks notification	7. Open chat page for User 2
8. Proceed with 2-way communication	

Use Case #15

for functional requirement 13

Case: Personalize Bio

Action	System Response
1. Perform Use Case #11 procedure	
2. Click “Edit Bio” button	3. Display edit bio form
4. Fill out bio form and submit	5. Update user bio and return to profile page

---

Use Case #16

for functional requirement 14

Case: Set Status

Action	System Response
1. Perform Use Case #11 procedure	
2. Click “Set status” button	3. Display Set status form
4. Specify status and submit	5. Update user status and return to profile page

---

Use Case #17

for functional requirement 15

Case: Create team in class

Action	System Response
1. Perform Use Case #5 procedure	
2. Click “Create team” button	3. Display create team form
4. Fill out create team form	5. Create team, put user in new team as creator, and display class page to user

---

Use Case #18

for functional requirement 16

Case: Join team

Action	System Response
1. Perform Use Case #5 procedure	
2. Click desired team	3. Display team page
4. Click “Join Team” button	5. Add user to team

Use Case #19

for functional requirement 17

Case: Leave Team

Action	System Response
1. Perform Use Case #7 procedure	
2. Click desired team in “My Teams” list	3. Display team page
4. Click “Leave Team” button	5. remove user from team

Use Case #20

for functional requirement 18

Case: Delete team

Action	System Response
1. Perform Use Case #17 procedure	
2. Click “Delete Team” button	3. Ask for confirmation
4. Confirm Deletion	5. Delete team and display home page

Use Case #21

for functional requirement 19

Case: Designate Team Owners

Action	System Response
1. Perform Use Case #17 procedure	
2. Click “Make Owner” next to desired user	3. Toggle Owner flag and provide feedback

---

Use Case #22

for functional requirement 20

Case: Chat with teammates

Action	System Response
1. Perform Use Case #18 procedure	
2. Use chat on team page to communicate	3. Propagate messages to everyone on team

with everyone on team	

Use Case #23

for functional requirement 21

Case: Automatic team matching

Action	**System Response**
1. Perform no actions past team-matching deadline for a given course	2. Add student to a team with team members who share similar skills

---