SW Engineering CSC648-848-05 Spring 2024

Application Name: **Swamp Study**

Seal Team One Members (Team 01)

Lennart Richter lrichter@sfsu.edu — Team Lead
Edmund Huang — Front-End Lead
Conrad Choi — Github Master
Julio Reyes — Docs-Editor
Rafael Fabiani — Backend-Lead

Version Table

Milestone 1	Feb 28, 2024
Milestone 1 V2	Apr 1, 2024
Milestone 2	Apr 4, 2024
Milestone 2 V2	Apr 24, 2024
Milestone 3	Apr 24, 2024
Milestone 3 V2	May 15, 2024
Milestone 4	May 15, 2024

Table of Contents

Product Summary

Name of the product: Swamp Study

URL: https://swamp-study.global.ssl.fastly.net/

Committed Application Functions

1. **User**
   - 1.1. A User shall be able to create an account
   - 1.2. A User shall only have one account
   - 1.3. A User shall be able to view other users' accounts in the system
   - 1.4. A User shall be able to set their availability
   - 1.5. A User shall be able to establish their profile
   - 1.6. A User shall be able to ask many questions in the Forum
   - 1.7. A User shall be able to answer many questions in the Forum
   - 1.8. A User shall be able to like other Users
   - 1.9. A User shall be able to match with other Users
2. **Profile**
   - 2.1. A Profile shall be owned and edited by one and only one user
   - 2.2. A Profile shall be allowed one picture of the owner
   - 2.3. A Profile shall be allowed a short biography
   - 2.4. A Profile shall have many subject attributes
   - 2.5. A Profile shall be able to set a filter for their study partner preferences
   - 2.6. A Profile shall be marked Student or Educator
   - 2.7. A Profile marked Educator shall be able to access the Forum
   - 2.8. A Profile marked Educator shall be allowed to answer questions in the Forum
3. **Account**
   - 3.1. An Account shall have one email associated with it
4. **User Interface**
   - 4.1. The Interface shall allow a User to navigate through a potential Match’s profile
   - 4.2. The Interface shall allow a User to access/edit their Profile
   - 4.3. The Interface shall allow a User to access/edit their Account
5. **Match**
   - 5.1. A Match shall be provided a meeting link
   - 5.2. A Match shall be provided a meeting time
   - 5.3. A Match shall be made based on different criteria
6. **Forum**
   - 6.1. The Forum shall allow one main question per Thread

6.2. The Forum shall allow each class its section
6.3. The Forum shall allow users to post in their class Forum
6.4. The Forum shall allow many questions

Summary/Unique Brag

The crowded field of educational applications is one of little breadth. With the COVID pandemic and teachers' reliance on online tools, the educational process is more intertwined with technology than ever before. The “Legacy” applications in the study space focus on providing a platform in which to ask and answer homework questions. This has led to a rise of unethical behavior to the educational process. These platforms want to sell a subscription that lets you get homework answers quickly for a price. This takes out the most important aspect of education, the process of learning and collaboration.

Swamp Study aims to be the leader in bringing collaborative learning to the online space. A common quandary faced by college students and those of all ages is the process of finding a study partner, someone with whom to share and enhance the learning experience. This is where Swamp Study comes in, we provide a state of the art matching system to give students the access to study partners at their learning institution. We provide users with their most likely matches first ensuring that you get quality partners which will lead to better grades. By letting the students put in their class schedule and availability we can make sure that you’ll find people with commonalities that will surely make the study experience an enjoyable one.

Usability Test Plan

Objective:

Our objective for our usability test aims to evaluate and refine the StudySwamp app's matching feature to deliver an optimal user experience that is both intuitive and satisfying. Our usability test checks whether the integral functions of the matching feature are intuitive, efficient, and user-friendly.

5 Functions To Be Tested:

- Filling user profile
- Matching
- View Matches
- Rating
- Report

Test Descriptions Per Function:

1. Function 1 - Filling User Profile

Test Objective

- We are evaluating the usability and intuitiveness of the profile setup process in the SwampStudyweb service, focusing on user satisfaction, and the time taken to complete the profile. This function is important because users that have setup their profile will get better matches.

Test Description

- Users will fill out their profile in the StudySwamp web service. This includes adding the courses they are currently taking, submitting their available weekdays, and adding a personal bio (not required).

Usability Test Description

- Participants will log in to a pre-established account.
- Participants will then navigate to the profile page.
- Participants will start to edit their profiles.
- Participants will enter a bio.
- Participants will add courses and weekly availability.
- The test will measure the time it takes to complete the profile.

Task	Description
Task	Complete personal profile page
Machine State	Logged In
Successful Completion State	The tester has entered his availability and courses.
Benchmark	5 minutes

2. Function 2 - Matching

Test Objective

- The Goal of this usability test is to determine whether the interface and process for liking and disliking users is an intuitive and user friendly one
- We want to assess the efficiency and accuracy of the process of making matches within the system for a user

Test Description

- The Matching process consists of reading profiles and deciding whether to like their profile as a potential match or to dislike
- Users will be put on the matching page and asked to start interacting with the matching system

Usability Task Description

- For this task a user will go through the process of liking and disliking potential study partners. The ideal user for this is someone of college age who is familiar with the tinder style of interface for making matches
- The user will start their test on the matching page, they will be given a testing account which already has the profile filled out. They should be able to interact with the matching interface right away

Task	Description
Task	Likes/Dislike Users Profiles
Machine State	Logged in on the Matching Page
Successful Completion State	The Tester has liked and disliked at least 10 different profiles
Benchmark	5 Minutes

3. Function 3 - View Matches

Test Objective

- We are testing the navigation and the user interaction with the View Matches function within the application.
- The objective is to evaluate the intuitiveness and the efficiency of navigating from the first page a user sees to the matching page with prior matches. The test is to ensure that students can easily form study groups by navigating and interacting with their matched profiles.

Test Description

- System setup will be conducted on an internet browser.
- Participants will start on a pre-established account on the Forums page.
- The intended audience for this test is students seeking to form study groups.

Usability Test Description

- The starting point will be on the Forums page the users will manually navigate to the Matching page. This is to test the application's navigational design. Also found on (https://swamp-study.global.ssl.fastly.net/)
- Reaching the Matching tab the participants will interact with any element in the matches revealing a detailed profile description of the selected profile.
- The test will measure the time it takes to navigate between tabs and ease of accessing the matched profiles.
- Feedback will be collected based on the clarity, usefulness, and satisfaction of using the View Matches function.

Task	Description
Task	Interact with any profile you matched with.
Machine State	The homepage logged in a pre-established profile.
Successful Completion Criteria	Participants place the mouse cursor on the Your Matches tab.
Benchmark	8.45 Seconds

4. Function 4 - Rating

Test Objective

- We test the rating component with the intent to gauge the efficiency and intuitiveness of the functionality itself as the user interacts with it.
- Moreover the tests aims to assess how accurately and effectively a user can submit a rating for a user.
- The test will also be used to asses and identify potential pitfalls, bugs or problems with the current implementation.

Test Description

- System setup will be conducted on an internet browser.
- Each participant will perform the task of rating other users by selecting a number of stars so as to reflect their perception of said users profi
- The intended audience for this test is users who are familiar with interactive rating systems similar to those seen in e-commerce or other social media platforms.
- Participants will use a pre configured testing account with the intent that they can start rating immediately upon accessing the match details page.

Usability Test Description

- Participants will begin the test from the Meeting Details page within the application. Here, they will click the "Rate User" button, which opens a view displaying the user's current rating and the interface for submitting a new rating.
- Feedback will be collected based on the timeliness, clarity, and effectiveness of using the View Matches function.

Task	Description
Task	Submit a 0 to 5 star rating for a user
Machine State	The users current rating
Successful Completion Criteria	Participants choose a rating and submit it
Benchmark	7 Seconds

5. Function 5 - Report Users

Test Objective

- Main objective is to assess the ease of use, and effectiveness of users being able to Report other users.

Test Description

- Participants will be asked to successfully match with a user and then assess if they can Report that user to us.

Usability Test Description

- The ideal user for this usability test is an individual that will do the community a favor by reporting inappropriate behavior and content. Overall enhancing the user experience of our website.

Task Description

Task	Description
Task	Successfully use the Report function
Machine State	Users own “previous matches” page
Successful Completion Criteria	Pop Up stating Report was successful
Benchmark	1 minutes

Usability Test Table

Tests/Use Cases	Total Completed (in Percentage)	Errors	Comments	Efficiency
Filling User Profile	100%	Added the wrong class but was able to fix it	Hard to find specific classes without reading all the text.	4 minutes
Matching	100%	N/A	If you aren't familiar with the "Swiping" it's difficult to understand what exactly is happening. I didn't find any matches as well so it's hard to test	45 Seconds
View Matches	100%	Didn't notice the Matching tab to move to the matching page.	"It can be easy to miss."	8.93 Seconds
Rating	100%	N/A	"Its quick, i like that the button is big / yellow its not hard to find"	~7 seconds
Report Users	100%	none	"I think it is a little confusing finding your matches. I had to click around to figure out I had to do previous matches. It would be nice to have a separate section that can show all matches. When I think of previous matches I think that those are the most recent and not all" "Also you have to have a match in order to report them, is that intentional? Like what if I see someone's profile and I think they are sus, do i need to match with them to report."	2-3 min

Questionnaire

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Is filling out the profile straight forward?				X	
Is the clear separation of fields beneficial?					X
Did you learn what you could edit quickly?					X
Potential Match Profile info is displayed in an easily digestible manner				X	
The Like Mechanism is intuitive				X	
The Dislike Mechanism is intuitive			X		
Easy to Navigate to Matching Tab					X

Information displayed in Your Matches was clear and easy to locate				X	
I understood the functionality of Your Matches at first glance.					X
Rating a user is simple and straight forward					X
Selecting the rating i want was easy					X
I understood that the stars in the rating page represent the rating i want to submit					X
Easily found a Previous Match		X			
Easily found the Report button		X			
UI was not confusing, user did not get lost trying to Report a user			X		

QA Test Plan

1. Performance: The site shall load in less than 7 seconds on a wireless connection with a speed of 350 Mbps

- Test Objective:
  - The Test will ensure that the website loads in a reasonable amount of time
- HW and SW Setup:
  - The Website needs to be successfully deployed to the EC2 instance for this test
- Test Environment:
  - The Test will utilize 3 popular and reputable tools for website testing
    - https://pagespeed.web.dev/
    - Google Chrome built in Lighthouse analyzer
    - https://www.webpagetest.org/

Number	Description	Test Input	Expected Output	Actual Output	Pass/Fail
1	Running the test on pagespeed.web.dev	The home page link, ran on a throttled connection of 10.2 Mbps	Speed Score of <= 7s	Speed Score of 2.2s	Pass
2	Running the test on lighthouse-metrics	The home page link, ran on a throttled connection of 10.2 Mbps	Speed score of <= 7s	Speed Score of 0.7s	Pass
3	Running the test on webpagetest.org	The home page link, ran on a throttled connection of 5Mbps from an East Coast location	Speed score of <= 7s	Speed Score of 1.159s	Pass

2. Scalability: The system shall be containerized allowing for simple scaling

- Test Objective:
  - To test the containerization of out application
- HW and SW Setup:
  - Testing can be done when the container is running on the EC2 instance. Docker Hub needs to be available, as well as GitHub Workflows
- Test Environment:
  - 1 Person with access to all the required resources, 1 hour

Number	Description	Test Input	Expected Output	Actual Output	Pass/Fail
1	Running the GitHub workflow for automated pushing to DockerHub	Manually running the GitHub Workflow for Container creation and push to DockerHub	A new version of the container in DockerHub	A new version of the container in DockerHub	Pass
2	Running the DockerHub Webhook for automatic deployment on the EC2 instance	Sending a new version of the container to DockerHub using the above workflow, and having the webhook send a signal to the EC2 instance	Checking the uptime and version of the Docker Container running on the EC2 to make sure the process worked successfully	A new version of the container was pulled and started running on the instance	Pass
3	Rolling back to another version of the Docker container	Running the workflow for creating a new version of the container with issues and then rolling back to a previous	We expect the broken/buggy version to stop running and the “new” reverted version to take over	The Docker container successfully stopped and was rolled back to a previous version	Pass

		version of the container on the instance			

3. Data Integrity: The Application shall allow only registered users to access their data

- Test Objective:
  - To determine whether data integrity is being upheld by allowing users to only access data pertinent to them
- HW and SW Setup:
  - The EC2 instance needs to be running, the S3 bucket active, and the Fastly CDN active
- Test Environment:
  - 1 person required, and 30 minutes

Number	Description	Test Input	Expected Output	Actual Output	Pass/Fail
1	Test whether the settings page can be accessed without being logged in to change a password	Trying to access the https://swamp-study.global.ssl.fastly.net/settings url without being logged in. And curling the endpoint without authorization	A Redirect in the case of the URL and a forbidden on the Curl to the endpoint	When trying to access the URL we get a redirect. When curling the endpoint we get a json response saying “User not authenticated ”	Pass
2	Test whether we can delete a User by hitting the API Endpoint	Curling the Endpoint responsible for deleting user accounts	A message stating it’s forbidden to access that feature without being authenticated	Json response message saying “User is not authenticated ”	Pass
3	Checking whether authorized	Logging in and changing the password	We expect the change to go through	Message saying the account was	We receive a popup stating that the

	changes work	on the settings page for the logged in user	and change the user password	updated	account was successfully updated

4. Privacy: Only Registered Users shall be able to access the application

- Test Objective:
  - To determine whether Privacy is being upheld by only allowing registered users to access any parts of the site besides the about and login
- HW and SW Setup:
  - The Application has to be running on the EC2 instance
- Test Environment:
  - 1 Person, 30 Minutes

Number	Description	Test Input	Expected Output	Actual Output	Pass/Fail
1	Accesing the Forum Page	Trying to access https://swamp-study.global.ssl.fastly.net/forum	To not be on the Forum page	An Empty page	Pass
2	Accessing the Matching Page	Trying to access https://swamp-study.global.ssl.fastly.net/matching	To not be on the Matching page	An Empty Page	Pass
3	Accessing the Settings page	Trying to access https://swamp-study.global.ssl.fastly.net/login	To not be on the settings page	The Login Page	Pass

5. Utility: The site shall support concurrent usage by many users without negligible performance reduction

- Test Objective:
    - Stress testing the website to make sure concurrent usage does not negatively impact other users of the system. The tests will be compares to a benchmark of 100 requests
- HW and SW setup
    - The application needs to be on the EC2 instance and the Fastly CDN needs to be configured
- Test Environment:
    - 1 Person, 30 minutes. Using Apache Benchmark on WSL to perform the requests
    - Benchmark: Sending 1000 requests, 50 concurrently
        - Requests per second: 219.24
        - 99% of Requests served within 0.508s

Number	Description	Test Input	Expected Output	Actual Output	Pass/Fail
1	Light Load	Sending 1000 requests 100 concurrently		227.83 Requests Per Second, and 99% served within 0.553s	Pass
2	Medium Load	Sending 1000 Requests, 250 concurrently		195.72 Requests Per Second, and 99% served within 2.064s	Pass
3	Heavy Load	Sending 10000 requests, 1000 concurrently		374 Requests Per Second, and 99% served within 3.701s	Pass

Security Self Check

1. Using the express-session middleware we have implemented token based sessions. Any requests that try accessing the API or Data that is in a password protected area will need a valid session to access.
2. Passwords are all encrypted and once enterres by the user will not be able to get accesses in clear text again. We are using the bcryptjs library within our program to encrypt passwords as they are entered into our application. When logging into the application the encrypted password which is stored in the DB gets retrieved and compared to the password the user answered and if the two match a user can login. This allows us to effectively store user passwords.
3. All data and text fields which are entered into the application have front-end and back-end validation. This double sided approach allows us to catch any type of invalid or dangerous data that users may try to input into the application. In our Service Layer we also validate all the data and only pass requests to the Database once we can be sure the request will have no adverse effect on the Application, and other users data.

Code Review

Coding Style

- All Frontend Folders in lowercase
- All React Components Completely Capitalized
- We use spaces instead of tabs
- All Controllers are camelCase
- All entities are lowercase
- All Repositories are Completely Capitalized
- All Routers are in camelCase
- All Services are Completely Capitalized
- Use Consistent Coding Style Throughout the Application
- Explain any Difficult Sections
- Provide Consistent Comments

Code Review

We used GitHub to make pull requests and review each others code. Below are some excerpts from those reviews.

implement requested changes and include a screen shot.

`application/frontend/src/App.jsx`

rf922 last week

added faqq route correctly

Reply...

Resolve conversation

`application/frontend/src/components/faq/FAQ.jsx`

rf922 last week

there are multiple components within `FAQ.jsx` namely `FaqItem` and `FAQ` itself , the convention we use is one component per jsx, each jsx file should live then in a folder whose name is the name of the jsx file in all lower case., seperate the components to their own jsx files and import or retool to only use one component. this is done with the intent that each component be modular and easier to maintain., also consider moving faqData to a seperate .js file `faqData.js` and importing it

Reply...

rf922 reviewed last week — View reviewed changes

rf922 left a comment

nice restyle for the registration component using consistent styling and addresses what is bein fixed, in the future make the pull request to Development branch

`application/frontend/src/components/register/Register.jsx`

rf922 last week

good job addressing the concern of new users being able to upload files

Reply...

Resolve conversation

rf922 closed this last week

len898 left a comment

Looks Great. The rewrite was done with our code style and looks to be bug free.

rf922 commented 2 weeks ago

good consistent code style with well chosen color scheme to align with site colors while adding contrast for visibility,

rf922 approved these changes 2 weeks ago — View reviewed changes

rf922 left a comment

code looks clean and fixes the issue as intended,

`application/frontend/src/components/footer/footer.jsx`

rf922 2 weeks ago

solid use of tailwind directly addresses the concern of the footer being too "thick", consistent styling changes to the margins `mr-4 md:mr-6` to `mr-2 md:mr-3`. good use of break points

Reply...

External Code Review

We performed a code review with Team 5. The code review was for one of our Controller files from the backend. This is the feedback we received. We implemented all the changes that made sense within the scope of our application

I checked out the LikeController code you sent over. Here are a few pointers:

Constructor: You don't need to assign the services inside the constructor since TypeScript does this automatically when you list them as private in the constructor itself.

Error Handling: The way you check if userId2 is a number is good. For errors, the message you send back could be clearer or more specific based on what went wrong.

Status Codes: Make sure you use CREATED only when something new is actually made. For other errors, 500 Internal Server Error might be more suitable than 422 Unprocessable Entity.

Business Logic: The part where you check if both users liked each other could perhaps go into the LikeService or MatchService. This would keep your controller simpler and just focused on handling the request and response.

Clean Up: If there's any code you're not using anymore, like // const userId1 = 1; it's a good idea to remove it to keep things neat.

I'm also attaching a file with some code I need help reviewing. Could you take a look when you have a chance?

Let me know if you have any questions or need more help with this!

Cheers,

The Feedback we sent to Team 5 is pictured below

We've reviewed your code and had a few pieces of feedback.

- there's a typo "const preferencess = await userService.getUserPreferences(req.params.userId);" there's an extra S in "preferences"
- i don't see validation but I'm assuming there is one somewhere else in their code.
- not a big thing but changing the order of parameters in the "try{ const user" to match the order of the "const createUser" or vice versa
- Everything else looks really good

Adherence to Non-Functional Requirements

1. **Usability**
   - 1.1. The U.I. will be designed to be as intuitive as possible allowing use from all technical backgrounds. - Done
   - 1.2. The Application shall be compatible with web browsers, and mobile devices - Done
   - 1.3. The Application shall conduct surveys on features that could improve usability - Issues
   - 1.4. The Application U.I. shall be streamlined and responsive - Done
   - 1.5. The system shall send notifications to the user's device - Moved to P2
   - 1.6. A user shall be allowed to match with many partners - Done
   - 1.7. A high rating shall make the user more visible - Done
   - 1.8. The system shall show a user their ideal match - Issues
   - 1.9. The system shall notify users when they have a match - Moved to P2
   - 1.10. A high rating shall gift users one free super like - Moved to P2
2. **Reliability**
   - 2.1. The application shall achieve 90% uptime - Done
   - 2.2. The Application shall minimize the occurrence of bugs and crashes - Done
3. **Scalability**
   - 3.1. The System shall be containerized allowing for simple scaling - Done
   - 3.2. The System shall automatically provision resources to handle fluctuating demand - Done
4. **Maintainability**
   - 4.1. The Application shall be formed in a manner that supports hot fixes to minimize downtime - Done
   - 4.2. The Restoration period shall be less than 24 hours - Done
   - 4.3. The Application shall be able to update and upgrade without downtime - Done

5. **Utility**
    5.1. The site shall support concurrent usage by many users with negligible performance reduction - Done
    5.2. The System shall scale to accommodate a growing user base - Done
    5.3. The System shall support Firefox version 123.0 - Done
    5.4. The System shall support Google Chrome Version 122.0.6261.70 - Done
6. **Performance**
    6.1. The site shall load in less than 7 seconds on a wireless connection with a speed of 350Mbps. - Done
7. **Privacy**
    7.1. Users shall have access to privacy settings - Done
    7.2. Users shall be able to hide their profile - Done
    7.3. Users shall be able to view the privacy agreement - Done
    7.4. The Platform shall treat customer data as sensitive - Done
    7.5. Only Registered Users shall be able to access the application - Done
8. **Data Integrity**
    8.1. The Application shall ensure that all billable data is accurate - Moved to P2
    8.2. The Application shall only allow registered users to access data and the sites - Done
9. **Security**
    9.1. The Application shall encrypt sensitive user data - Done
    9.2. The Application shall comply with regional data safety regulations - Done
    9.3. Payment Information shall be routed through a trusted third party service provider using state of the art security measures - Moved to P2
10. **Storage**
    10.1. The Application shall assign 10MB of Memory per table - Done
    10.2. The Application shall make automatic backups to prevent data loss - Done
11. **Deployment**
    11.1. The Application shall use github integrations to build the Docker container - Done
    11.2. The Cloud instance shall automatically transition to the newest version of the application when available - Done
12. **Coding Standards**
    12.1. The Team shall use the same styling for all written code - Done
    12.2. The Team shall use spaces for indentation - Done
    12.3. The Team shall write thorough documentation and comments - Done
    12.4. The Team shall own the codebase and act responsibly - Done
    12.5. The Team shall write tests that ensure a good working product - Done

13. Environmental Sustainability

13.1. The Application shall use as few resources as needed - Done

13.2. The Application shall work with responsible service providers - Done

13.3. The Application shall minimize its footprint and make continuous efforts to improve further on sustainability - Done

Team Evaluation

Lennart (Team Lead) 10

- Project Planning
- Finished the M3V2 Revision
- Organize All The Team Meetings
- Wrote a Test Plan
- Wrote the QA Test Plans
- Wrote the Security Check
- Participated in the Code Review
- Attended All Team Meetings
- Responsive On Discord

Rafael (Backend Lead) 10

- Wrote the Matching Algorithm
- Implemented QOL Improvements for the Platform
- Participated in the Code Review
- Reviewed all internal Pull Requests
- Wrote a Test Plan
- Lots of bug fixes
- Tech Support for the Team
- Attended All Team Meetings
- Responsive on Discord

Edmund (Frontend Lead) 10

- Helped revise the Footer
- Multiple frontend improvements
- Participated in the Code Review
- Revised the Register Layout
- Spaced out the Navbar
- Attended All Team Meetings
- Responsive on Discord

Julio (Docs-Editor) 10

- Change the paths on the navbar to be more accurate
- Worked on the FAQ
- Implemented other routing changes
- Recorded all the meetings
- Uploaded Feedback video from Review session
- Made a Test Plan
- Attended All Team Meetings
- Responsive on Discord

Conrad (Github-Admin) 10

- Wrote the Overarching Test Plan
- Wrote the Tables in which we organized the Testing
- Implemented small fixes on the footer
- Various bug fixes
- Wrote a Testing Plan
- Attended All Team Meetings
- Responsive on Discord