CSE 343 SOFTWARE ENGINEERING

STUDYHILL

INDEX

1. User Stories and Scenarios

1.1 User Stories

1.1.1Group Story

1.1.2Self-Study Story

1.2 Scenarios

1.2.1 Creating a Group Scenario

1.2.2 Joining a Group Scenario

1.2.3 Leaderboard Scenario

1.2.4 Reward Scenario

2. Backlog

3. UML Diagram

3.1 Context Model

3.2 Process Models

3.2.1 Study

3.2.2 Create a Group

3.2.3 Join a Group

3.3 Use Case Diagram and Use Case Tables

3.3.1 Use Case Diagram

3.3.2 Study

3.3.3 Create Group

3.3.4 Join Group

3.3.5 Leave Group

3.3.6 Register

3.3.7 View Group Ranking

3.3.8 View Progress

3.3.9 Collect Badges

INDEX

3.4 Sequence Diagrams

3.4.1 Register Sequence

3.4.2 Join Group Sequence

3.4.3 Leave Group Sequence

3.4.4 Study Sequence

3.4.5 View Progress Sequence

3.5 Class Diagrams

3.5.1 Classes and Associations

3.5.2 Class Diagram

3.5.3 Generalization Hierarchy

3.6 State Diagrams

3.6.1 Group Class State Diagram

3.6.2 Badge Class State Diagram

4. System Architecture Document

4.1 Overview

4.2 System Architecture

4.3 Data Flow

4.4 Deployment

4.5 Maintenance

5. Test Reports

5.1 Register

5.2 Login

5.3 Create Group

5.4 Join Group

5.5 Leave Group

1.USER STORIES AND SCENARIOS

1.1 USER STORIES

1.1.1 Group Story

Rick, Morty and Summer are high schooler friends that wish to study together and increase their performance. Rick enters to the Studyhill website, he creates an account and logs in with his id and password, and he creates a study group by entering a group name. He then shares his group code with his friends, and his friends join the group with the code that Rick shared. Then they start studying with their separate study timer that its break and study session time can be customizable. Studyhill saves the amount of time studied by each member of the group in that week and sorts the members accordingly in the leaderboard, it appears to be Summer is on the top of the list, and she is rewarded with a badge. The amount of time studied by Rick, Morty and Summer reset every week to start a new competition. Rick, Morty and Summer can leave the group whenever they want, and new members can join the group whenever they want if they are not in a group already and if the group is not full.

1.1.2 Self-Study Story

Jerry is a student and wish to study with a program that tracks the time that he spent studying and that with a customizable study timer. Jerry enters the Studyhill website, creates an account and clicks the “Self-study” button. He customizes his study repeat amount, study and break times, and starts studying by pressing the button “Start”, after his study session time is up, he enters the break by clicking “Start”, and after the break timer ends, he enters study session again by pressing “Start” and so on until repeat amount is reached. When he done studying, he closes the website and Studyhill saves the total time Jerry studied.

1.2 SCENARIOS

1.2.1 Creating a Group Scenario

Initial assumption: User is logged in.

Normal: The user is clicked the create a group button, typed a group name and clicked the create button. User created the group, joined the group automatically and received the group code of the group.

What can go wrong: User wrote a group name that includes invalid characters. Inform the user, ask them to write the group name again.
User is already in a group. Inform the user.

System state on completion: The user is a member of the created group. The new group is in the database.

1.2.2 Joining a Group Scenario

Initial assumption: A user created a group, joined the group automatically, received a group code.

Normal: The user logged in to his/her account, clicked the join a group button, entered the group code which is shared with him/her and clicked the join button, then joined the group successfully.

What can go wrong: User wrote a code that is not valid. Inform the user and ask for them to write the code and press join button again.

The group is full. Inform the user.

The user is already in a group. Inform the user.

System state on completion: The user is a member of the group; group members amount is increased by one.

1.2.3 Leaderboard Scenario

Initial assumption: User logged in and he/she is in a group.

Normal: User pressed the leaderboard and leaderboard displayed.

What can go wrong: Group members may have same amount of study time.

System state on completion: No change in the system.

1.2.4 Reward Scenario

Initial assumption: User is in a group, and he/she is at top of the leaderboard in the end of the week.

Normal: The user is rewarded with a badge.

What can go wrong: User is the only one in the group at that time.
There wasn't any active member in that week.

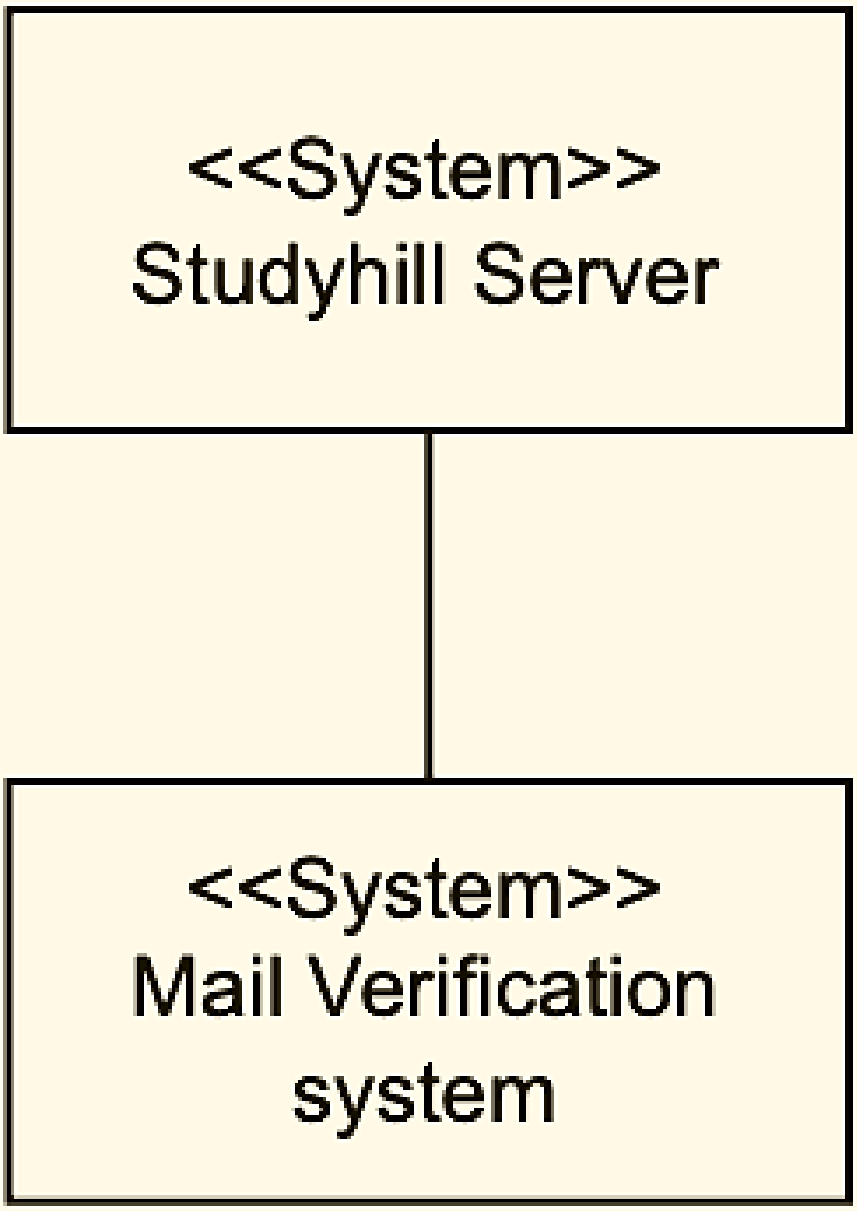
System state on completion: User that is at the top has a badge that with the information of the group that he/she was at the top of and the information of received date of badge.

2. BACKLOG

Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated
1. Home page	Unassigned	boran kurut	=	DONE	Done	07/Nov/22	24/Nov/22
1.1. Implement Create an account	Unassigned	boran kurut	=	DONE	Done	07/Nov/22	24/Nov/22
1.2. Implement Login	Unassigned	boran kurut	=	DONE	Done	07/Nov/22	24/Nov/22
10. Implement preferences data saving for an account (backend)	Unassigned	boran kurut	˅	TO DO	Unresolved	07/Nov/22	28/Nov/22
11. Implement time amount studied saving for an account in that week (backend)	Unassigned	boran kurut	^	TO DO	Unresolved	07/Nov/22	28/Nov/22
12. Implement Database	Unassigned	boran kurut	^	TO DO	Unresolved	24/Nov/22	28/Nov/22
13. Implement communication between frontend and backend	Unassigned	boran kurut	^^	DONE	Done	24/Nov/22	03/Dec/22
2. Implement create a group	Unassigned	boran kurut	=	TO DO	Unresolved	07/Nov/22	24/Nov/22
3. Implement join a group	Unassigned	boran kurut	=	TO DO	Unresolved	07/Nov/22	24/Nov/22
4. Implement main study page	Unassigned	boran kurut	^	TO DO	Unresolved	07/Nov/22	28/Nov/22
4.1. Implement start, pause timer (customizable)	Unassigned	boran kurut	^	TO DO	Unresolved	07/Nov/22	28/Nov/22
4.2.1: Timer preferences (study break times)	Unassigned	boran kurut	=	TO DO	Unresolved	07/Nov/22	07/Nov/22
4.2.2: Repeat amount	Unassigned	boran kurut	=	TO DO	Unresolved	07/Nov/22	07/Nov/22
4.2.3: Sound preferences	Unassigned	boran kurut	˅˅	TO DO	Unresolved	07/Nov/22	28/Nov/22
4.2: Implement preferences	Unassigned	boran kurut	˅	TO DO	Unresolved	07/Nov/22	28/Nov/22
4.3: Implement leaderboard (for groups) and weekly track	Unassigned	boran kurut	=	TO DO	Unresolved	07/Nov/22	07/Nov/22
4.4: Implement badges	Unassigned	boran kurut	˅	TO DO	Unresolved	07/Nov/22	28/Nov/22
5. Implement self-study	Unassigned	boran kurut	˅	TO DO	Unresolved	07/Nov/22	28/Nov/22
6. Create an account (backend)	Erkut Dere	boran kurut	=	DONE	Done	07/Nov/22	24/Nov/22
7. Implement log in account (backend)	Unassigned	boran kurut	=	DONE	Done	07/Nov/22	24/Nov/22
8. Implement create a group (backend)	Unassigned	boran kurut	=	DONE	Done	07/Nov/22	12/Nov/22
9. Implement join a group with a code (backend)	Unassigned	boran kurut	=	DONE	Done	07/Nov/22	12/Nov/22

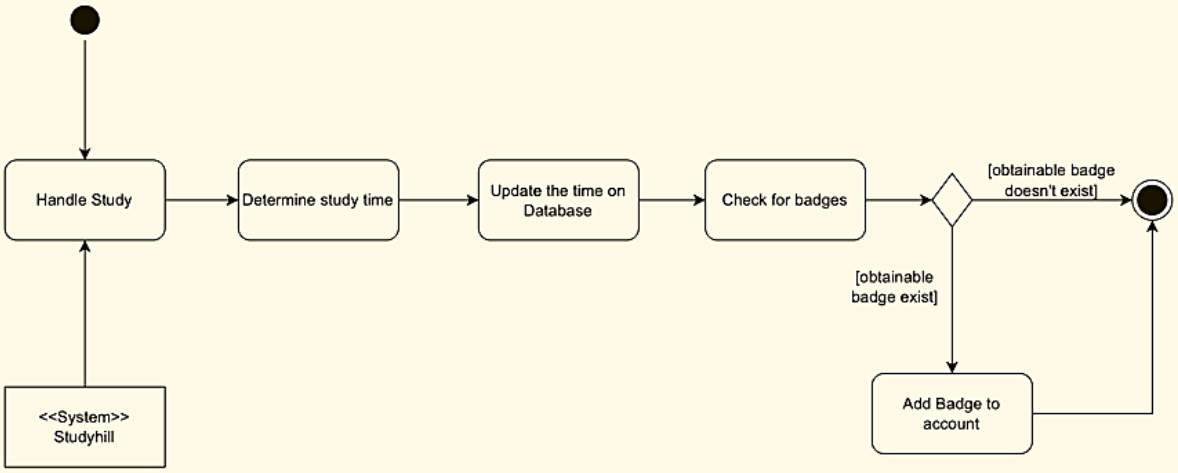
3. UML DIAGRAMS

3.1 CONTEXT MODEL

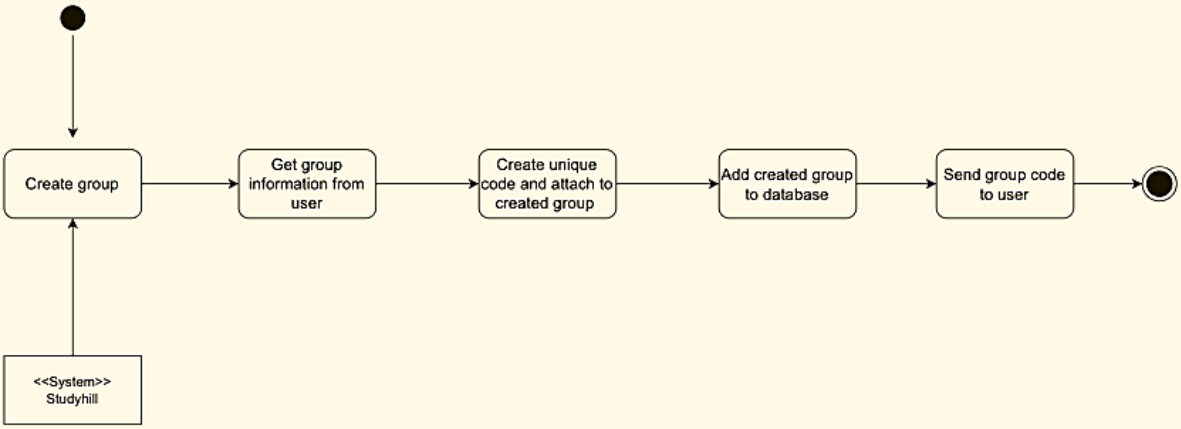


3.2 PROCESS MODELS

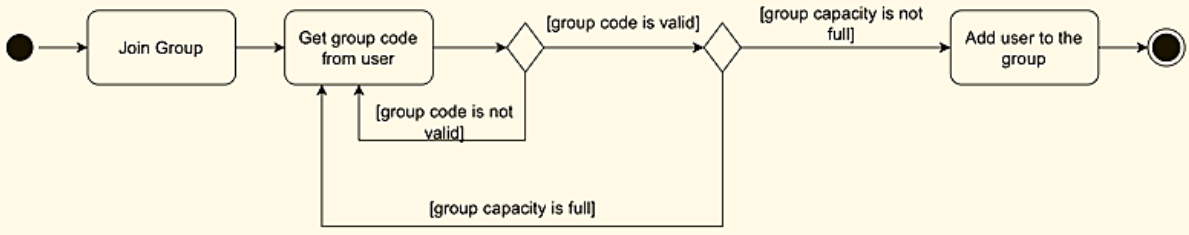
3.2.1 Study



3.2.2 Create a Group

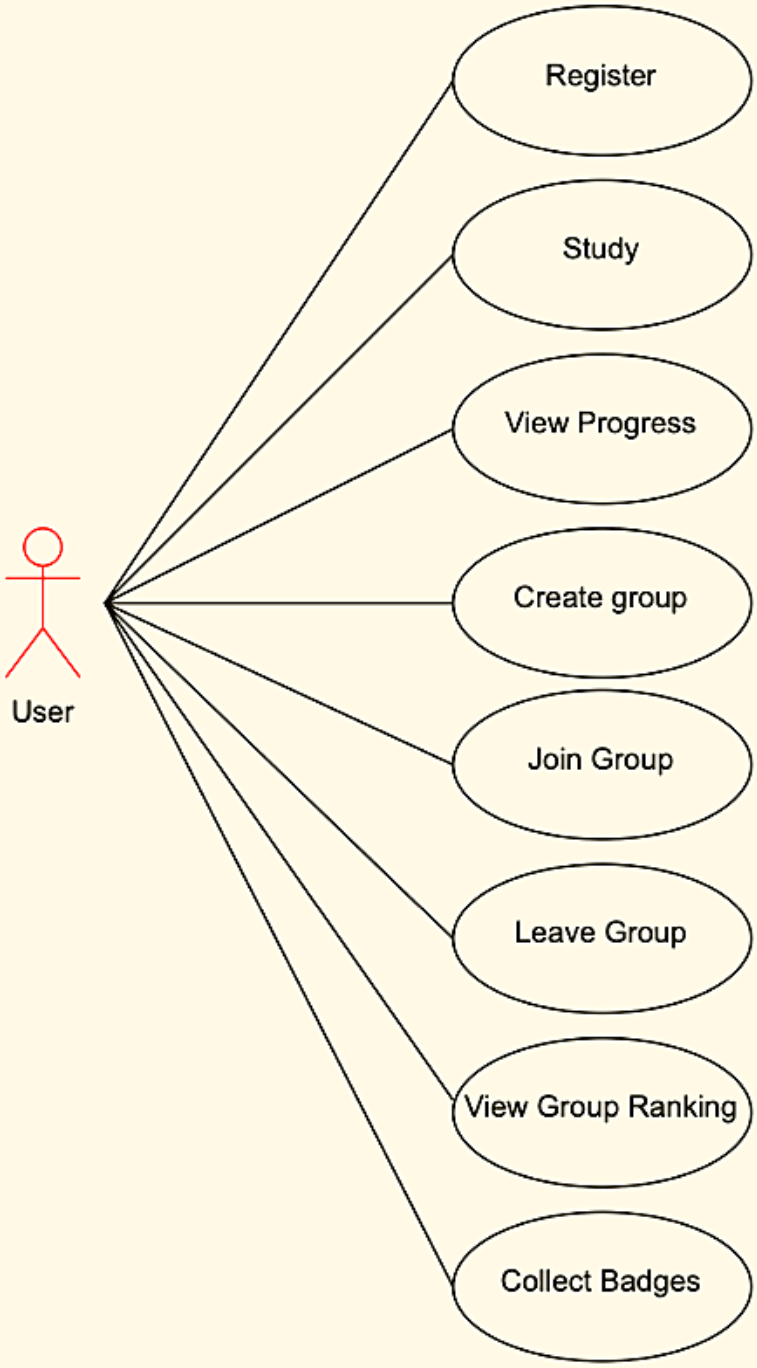


3.2.3 Join a Group



3.3 USE CASE DIAGRAM AND USE CASE TABLES

3.3.1 Use Case Diagram



3.3.2 Study

System: Studyhill Website
Use Case: Study
Actors: User, database
Data: User's timer preferences data that created by user earlier from database or if it does not exist default preferences data
Stimulus: Start timer, break timer.
Response: Studied time is saved to database with user id and date.
Comments: This data will be used to track users' progress.

3.3.3 Create Group

System: Studyhill Website
Use Case: Create Group
Actors: User, database
Data: Group name provided by user, max group size
Stimulus: User creates a group
Response: Group code of the created group, group is saved to the database. The creator is the admin.
Comments: Group code is a six-digit string, more people than max group size provided by the user cannot join the group.

3.3.4 Join Group

System: Studyhill Website
Use Case: Join Group
Actors: User, database
Data: Group code
Stimulus: User enters the group code in join a group page
Response: User joins the group that group code references.
Comments: Group members can view the leaderboard, if a user is already in a group, he/she cannot join another group.

3.3.5 Leave Group

System: Studyhill Website
Use Case: Leave Group
Actors: User, database
Data: User id
Stimulus: User leaves the group
Response: User and users information removed from group
Comments: When admin leaves the group a group member becomes the admin

3.3.6 Register

System: Studyhill Website
Use Case: Register
Actors: User, database
Data: Username, email, password
Stimulus: User provides the data and registers
Response: The email provided by the user receives a verification mail, to users to verify their account. The user is saved to the database.
Comments: Same email address cannot used by multiple account

3.3.7 View Group Ranking

System: Studyhill Website
Use Case: View Group Ranking
Actors: User, database
Data: The studied time of the members of the group that the user is in. (Provided by database)
Stimulus: User views group ranking
Response: Group ranking is showed with members study time
Comments: Can only be used by group members

3.3.8 View Progress

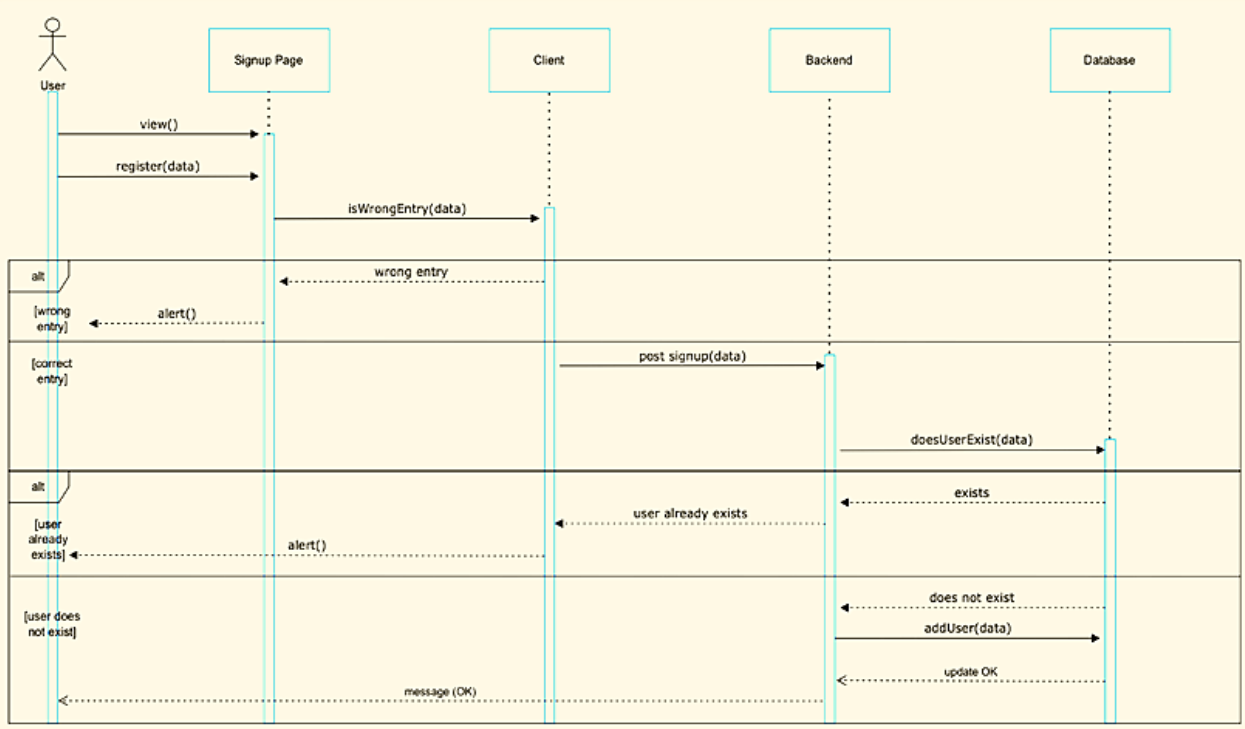
System: Studyhill Website
Use Case: View Progress
Actors: User, database
Data: Users' studied time provided by database.
Stimulus: User views progress.
Response: Users' progress is shown.
Comments: Users can see their predetermined daily study time goal in the progress to compare their progress.

3.3.9 Collect Badges

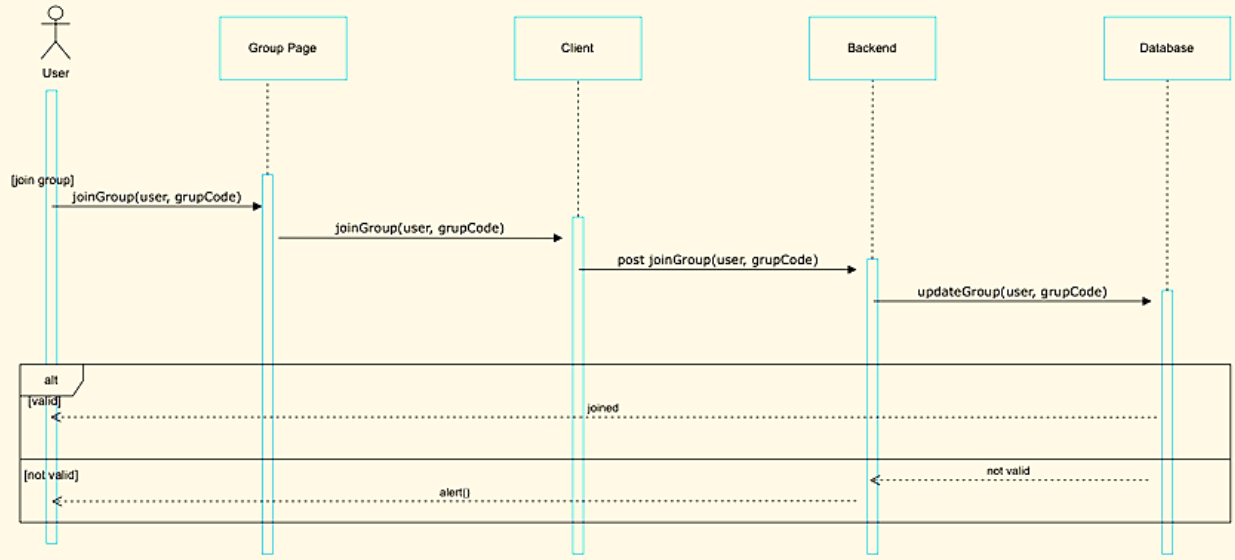
System: Studyhill Website
Use Case: Collect Badges
Actors: User, database
Data: Users data provided by database.
Stimulus: Update in users study data.
Response: If a badge can be collectable by user, badge is added to the user badges in database.
Comments: Users can see their badges in collected badges.

3.4 SEQUENCE DIAGRAMS

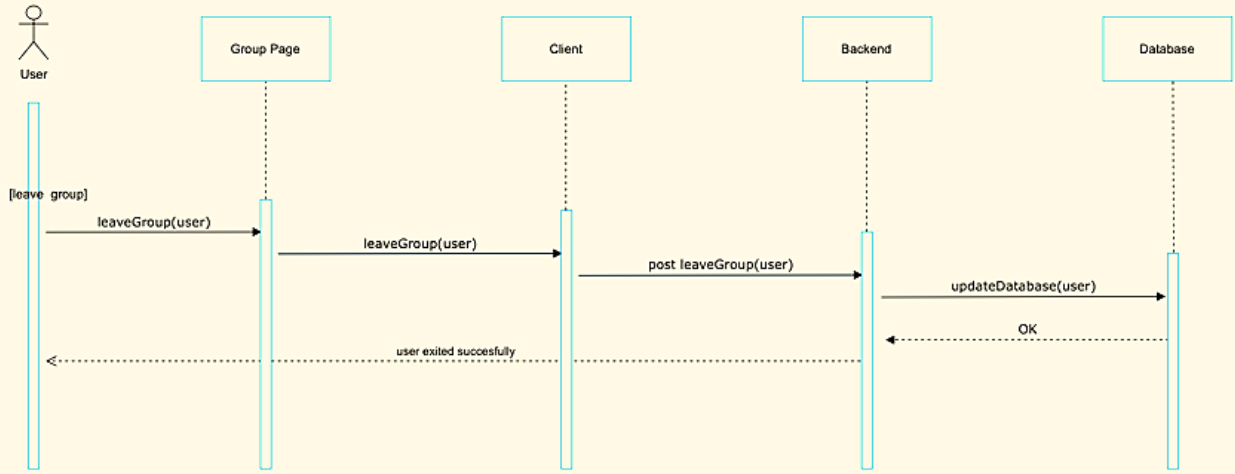
3.4.1 Register Sequence



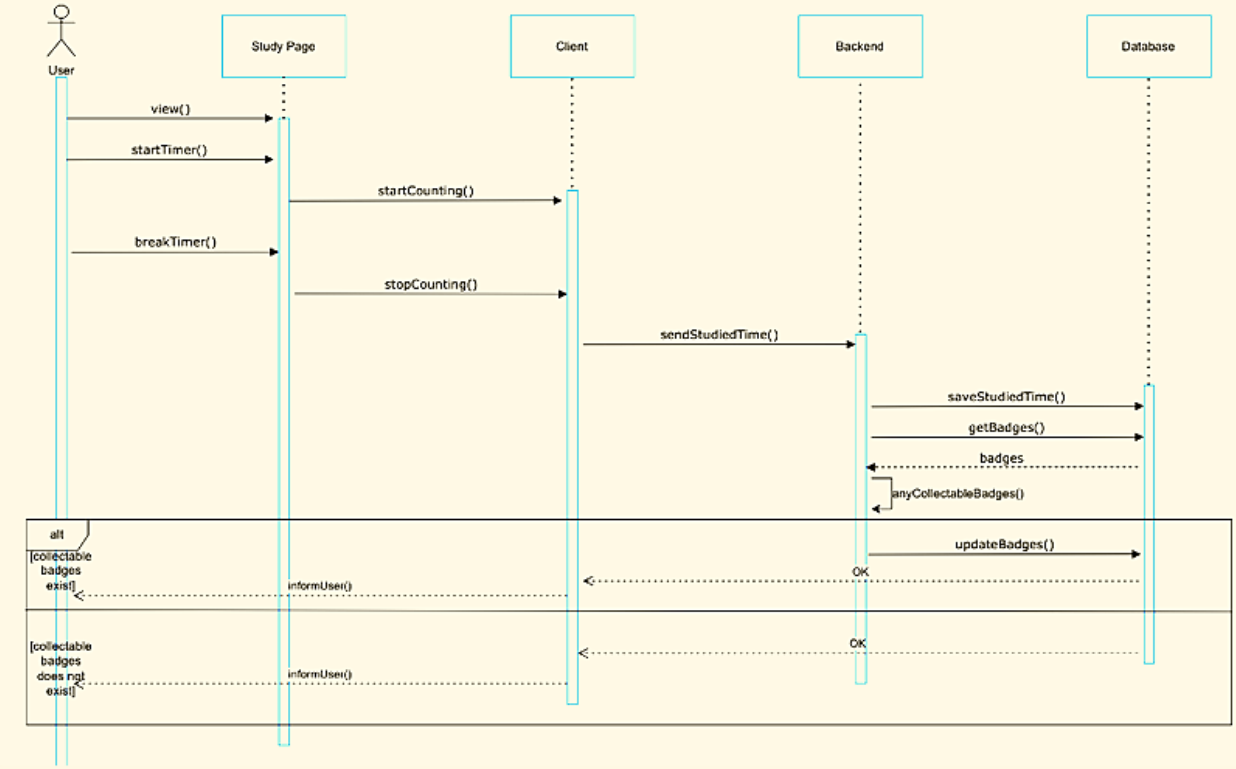
3.4.2 Join Group Sequence



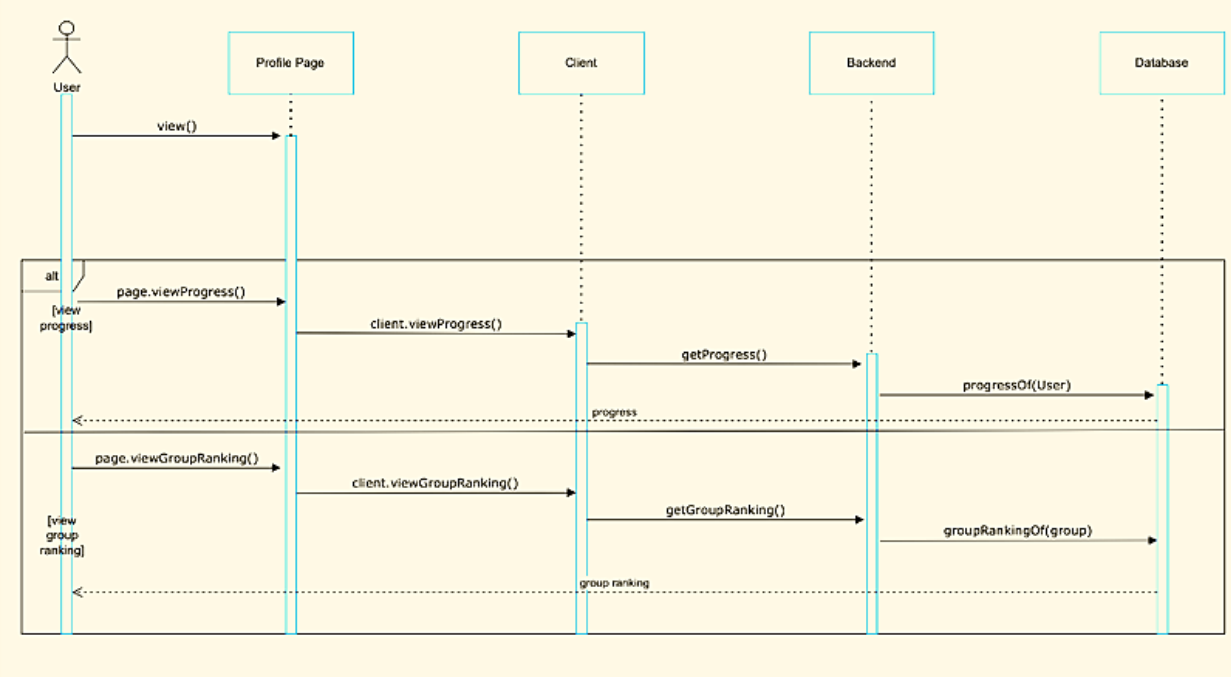
3.4.3 Leave Group Sequence



3.4.4 Study Sequence

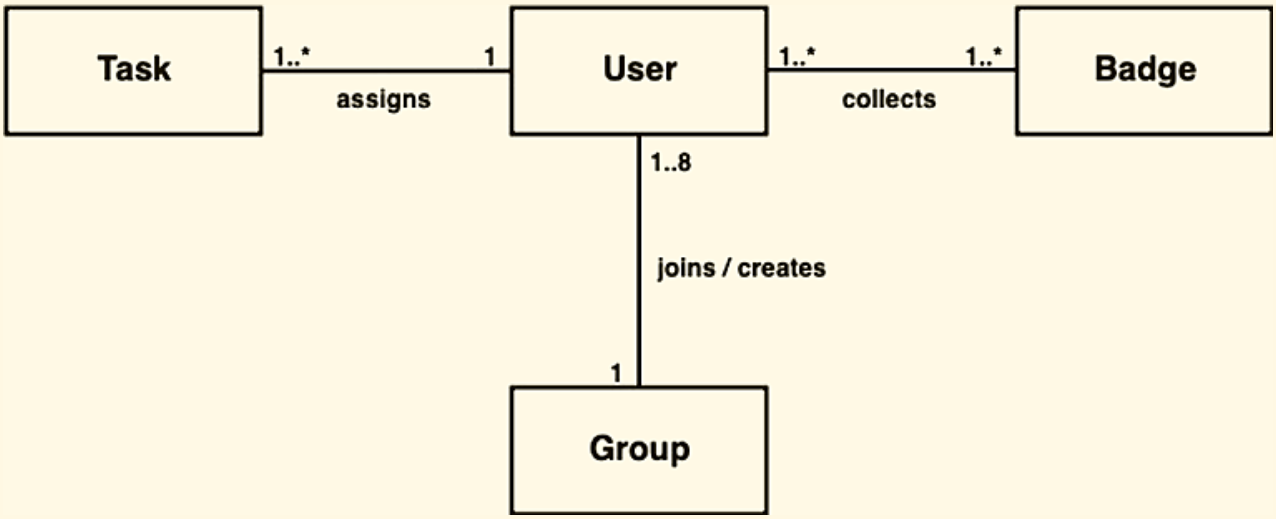


3.4.5 View Progress Sequence

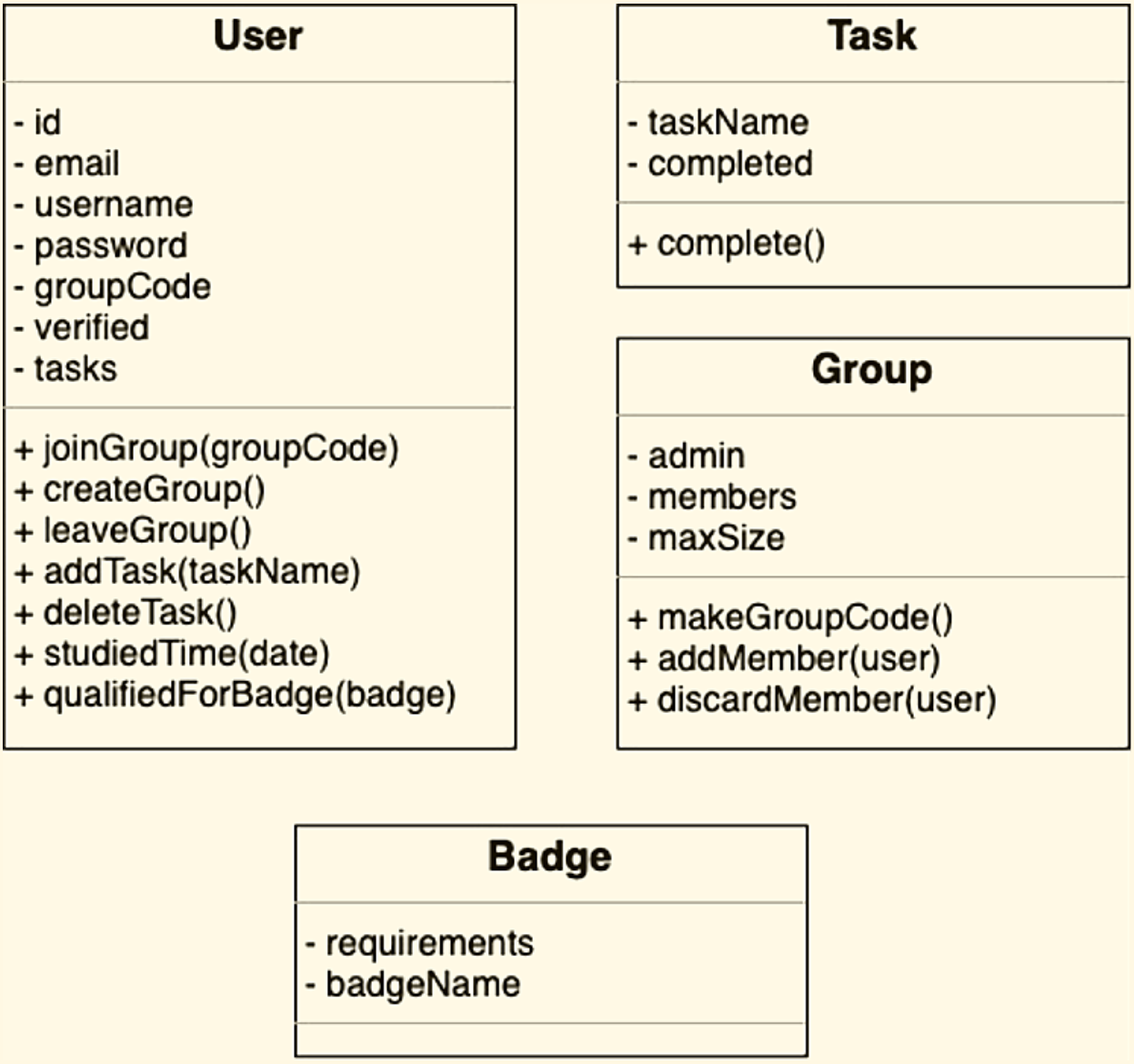


3.5 CLASS DIAGRAMS

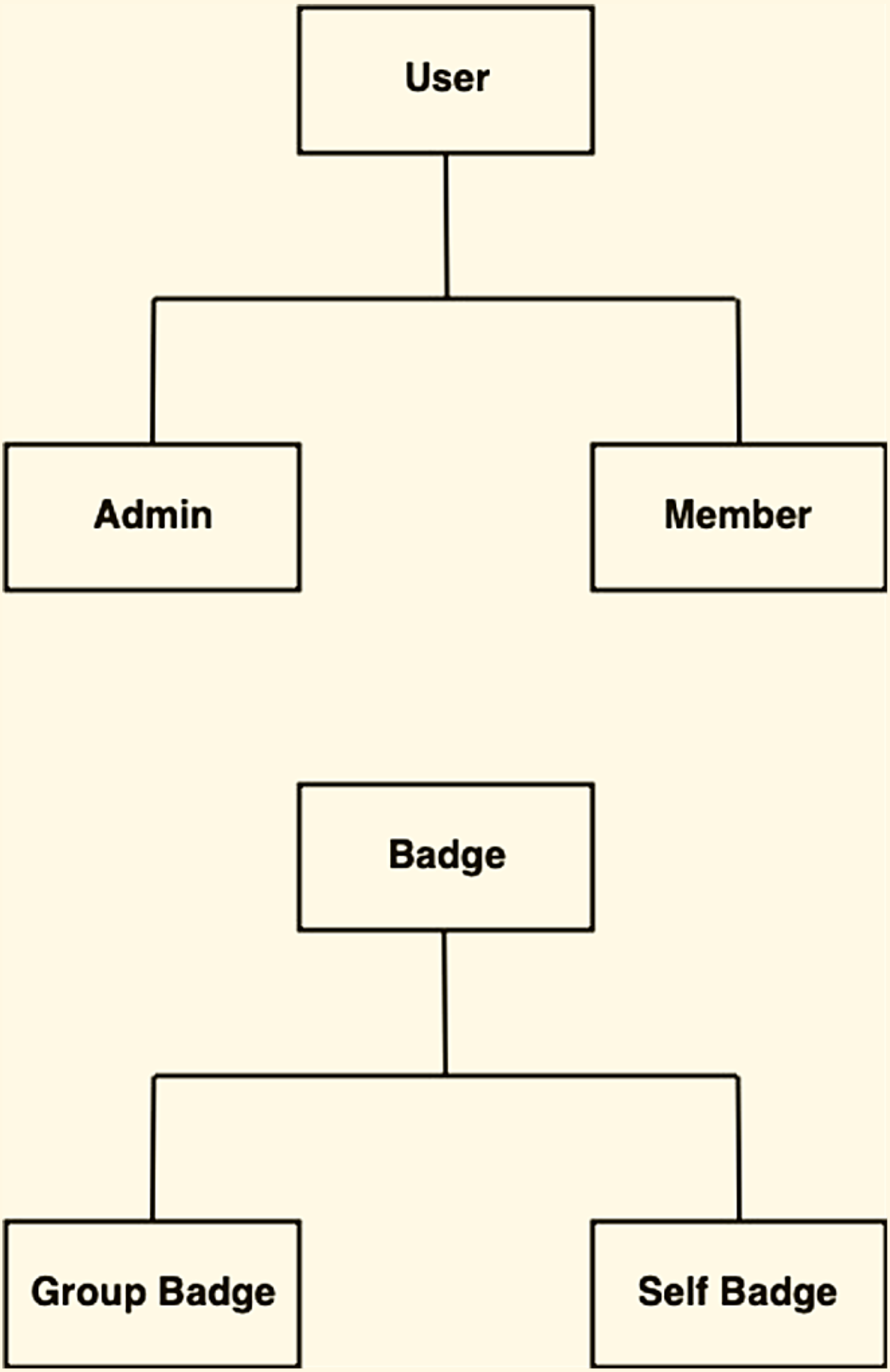
3.5.1 Classes and Associations



3.5.2 Class Diagram

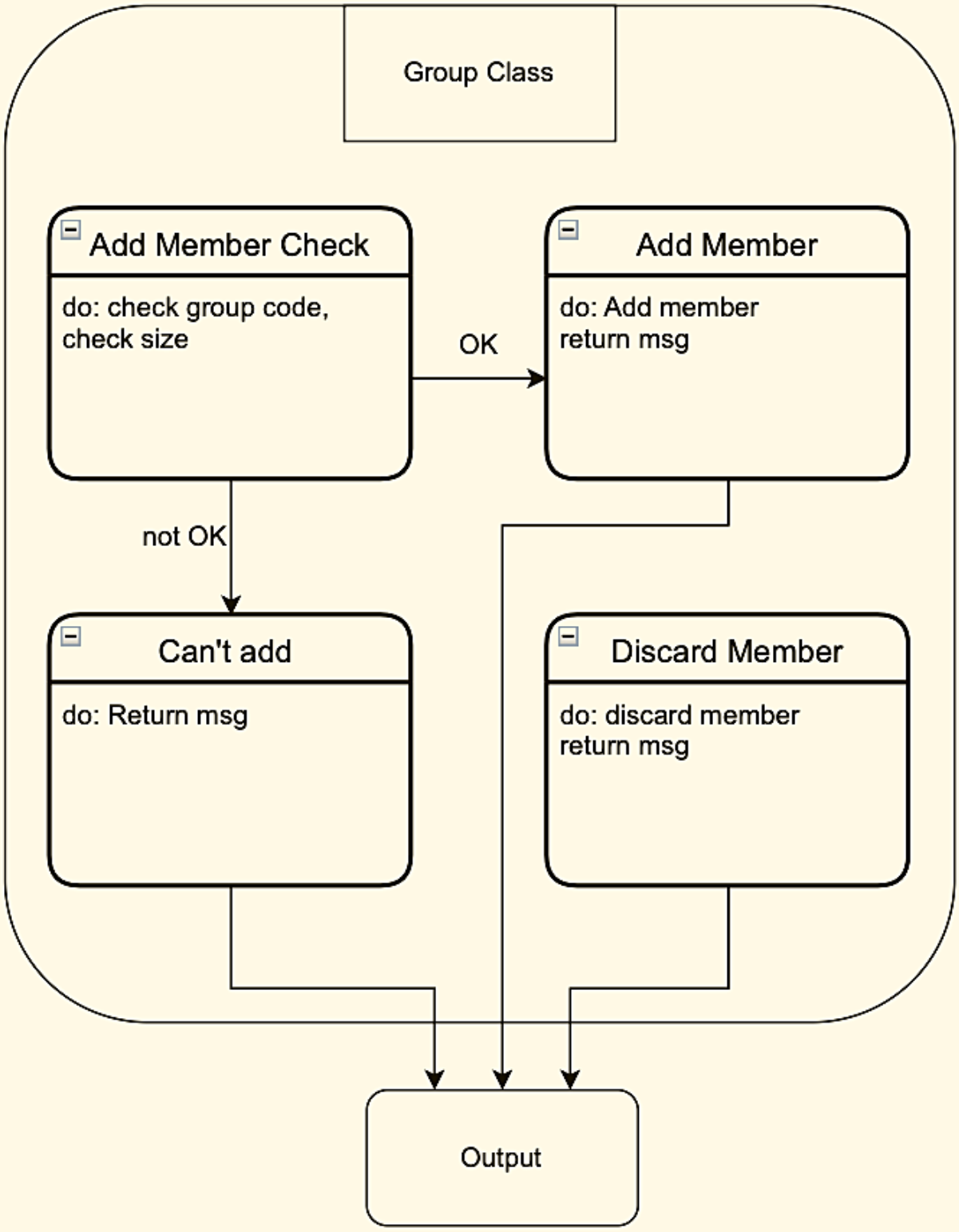


3.5.3 Generalization Hierarchy

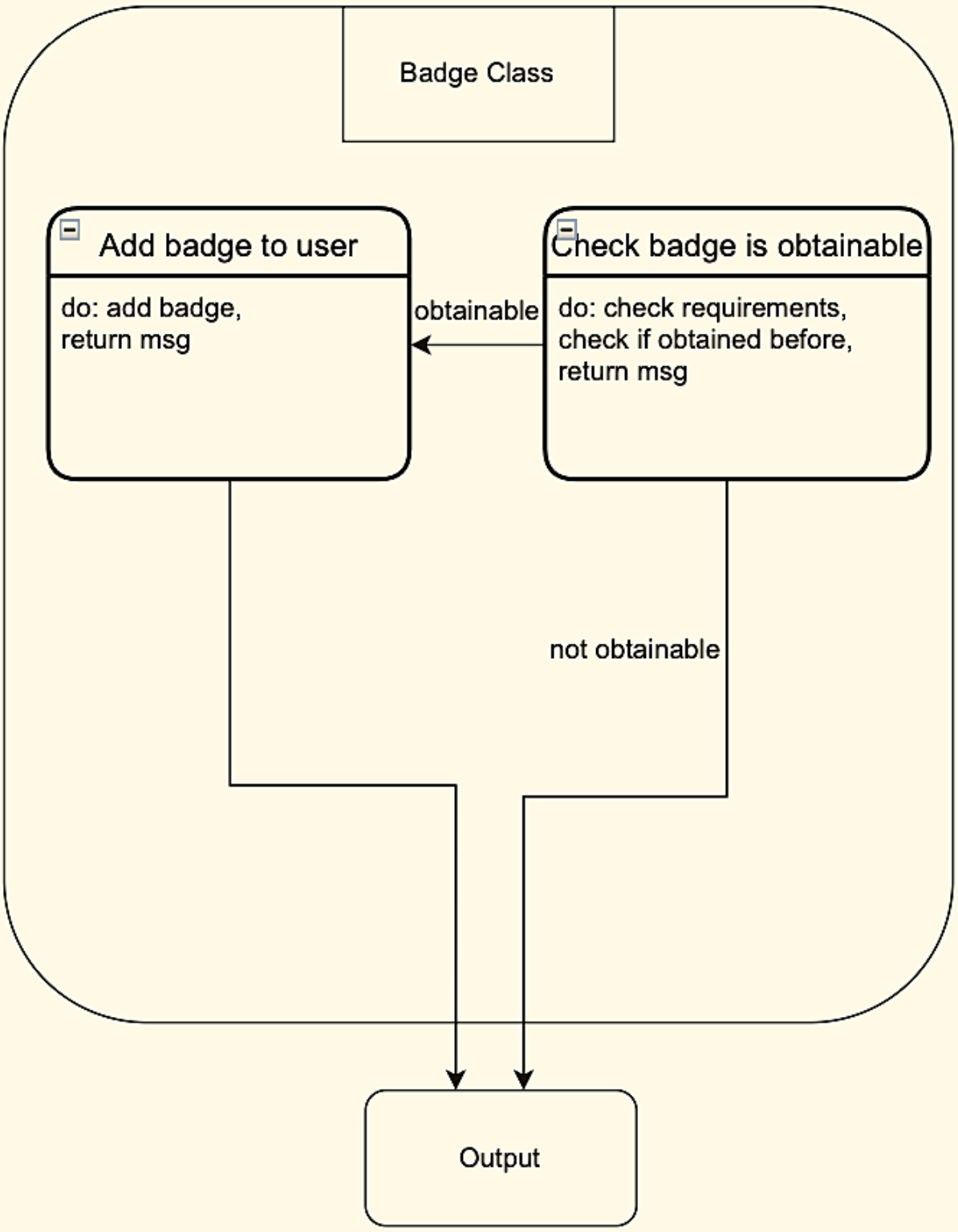


3.6 STATE DIAGRAMS

3.6.1 Group Class State Diagram



3.6.2 Badge Class State Diagram



4. SYSTEM ARCHITECTURE DOCUMENT

4.1 OVERVIEW

Studyhill is a web application that allows users to study with their friends using the Pomodoro technique. It allows users to create groups and track their studied time, which is saved to a database. The studied time data is then used to generate a leaderboard for each group, showing the total studied time for each member for a given week. The application also includes a badge system that rewards users with badges for various accomplishments, such as being the group winner for a week or reaching a certain number of hours studied. Users must register with their email address and verify their account by clicking a link sent to their email before they can use the application. Groups are created with a unique code that is generated by the backend and can be used by other users to join the group. When creating a group, users must specify a name and a size between 2 and 8. Users who are in a group are directed to the group-profile page upon login, while users who are not in a group are directed to the profile page. The Pomodoro timer used by users is customizable, with adjustable study and break times.

Architecture Diagram

4.2 SYSTEM ARCHITECTURE

The Studyhill system consists of the following components:

- **Frontend:** The frontend of the Studyhill application is built using React and Node.js. It is responsible for rendering the user interface and handling user interactions with the application. The frontend includes the customizable Pomodoro timer and allows users to adjust the study and break times.

- **Client:** The client is a Node.js component that sits between the frontend and the backend. Its purpose is to handle any tasks that need to be performed on the client side, such as making HTTP requests or processing data.

- **Backend:** The backend of the Studyhill application is built using Express.js and Node.js. It is responsible for handling HTTP requests, interacting with the database, and performing business logic. The backend generates unique codes for each group, handles group membership updates when users join or leave a group, and tracks studied time data for each user and group.

- **Database:** The Studyhill application uses a MySQL database to store user and group data, as well as studied time data and badge information.

4.3 DATA FLOW

1. The user accesses the Studyhill application through a web browser, which sends a request to the frontend.
2. The frontend sends a request to the client to perform any necessary tasks on the client side.
3. The client sends a request to the backend to retrieve the necessary data from the database.
4. The backend retrieves the data from the database and sends it back to the client.
5. The client processes the data and sends it to the frontend.
6. The frontend renders the data for the user to view, including the customizable Pomodoro timer.
7. When the user adjusts the study or break times on the timer, the frontend updates the timer settings and saves them to the client.
8. When the user registers for an account, the frontend sends their email address to the backend.
9. The backend sends a verification email to the user's email address with a link to verify their account.
10. When the user clicks the verification link, the backend marks their account as verified in the database.
11. When the user creates a group, the backend generates a unique code for the group and stores it in the database.
12. When the user wants to join a group, they enter the group code on the frontend. The frontend sends a request to the client to join the group.
13. The client sends a request to the backend to update the group membership in the database.
14. The backend updates the database and sends a response back to the client.
15. The client processes the response and sends it to the frontend.

16. The frontend updates the user interface to reflect the changes made to the database and directs the user to the appropriate profile page (group-profile or profile).
17. When the user starts the Pomodoro timer, the frontend sends a request to the client to start tracking studied time.
18. The client sends a request to the backend to update the studied time data in the database.
19. The backend updates the database and sends a response back to the client.
20. The client processes the response and sends it to the frontend.
21. When the user completes a Pomodoro, the frontend sends a request to the client to stop tracking studied time and update the studied time.
22. The client sends a request to the backend to update the studied time data in the database.
23. The backend updates the database and sends a response back to the client.
24. The client processes the response and sends it to the frontend.
25. The frontend updates the user interface to reflect the changes made to the database and displays the appropriate badge if any badge criteria have been met.
26. When the user interacts with the application (e.g. viewing the leaderboard), the frontend sends a request to the client to perform any necessary tasks on the client side.
27. The client sends a request to the backend to retrieve the necessary data from the database.
28. The backend retrieves the data from the database and sends it back to the client.
29. The client processes the data and sends it to the frontend.
30. The frontend renders the data for the user to view.

4.4 DEPLOYMENT

The Studyhill application can be deployed to a web server or cloud platform such as AWS or Azure. Project size is about 250mb. It can be accessed by users through a web browser. In the case of high traffic and low performance, load balancer of aws will be used and depending on revenue, PaaS of aws (elastic Beanstalk) will be an option.

4.5 MAINTENANCE

Regular maintenance tasks for the Studyhill system include:

- Monitoring the performance of the application and identifying and fixing any issues that arise.
- Updating the database schema as necessary to support new features or data types.
- Backing up the database regularly to ensure data integrity and availability.
- Updating the frontend and backend code to fix bugs or add new features.
- Adding new badges to the badge system as needed.
- Ensuring the security of the application by implementing proper authentication and authorization measures and regularly testing for vulnerabilities.

5. TEST REPORTS
5.1 REGISTER

Description

User enters an email address, username, password and password-confirm to register.

Tests

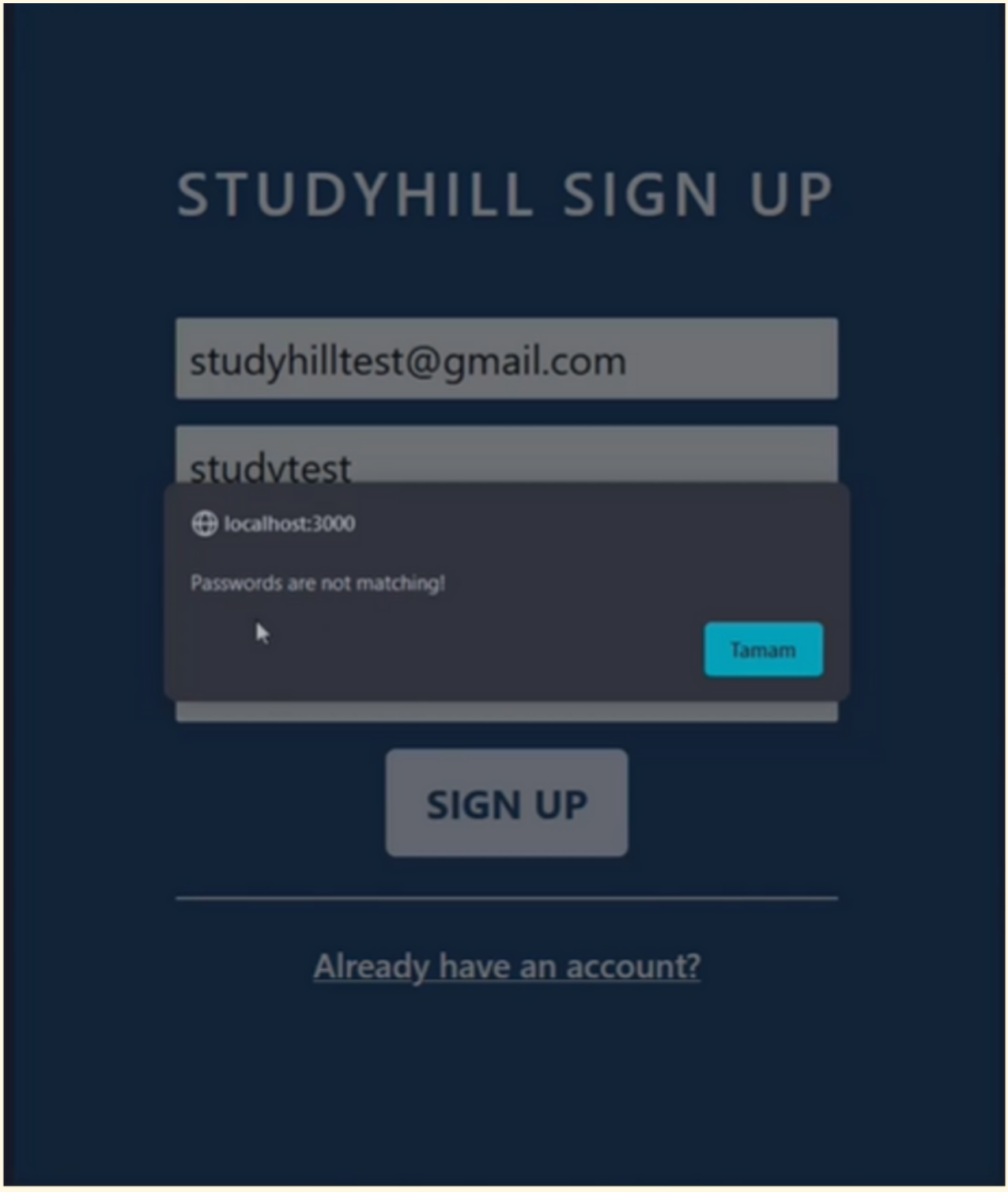
1. Password and password-confirm should match.
2. Email address shouldn’t be used for another account.

Outputs

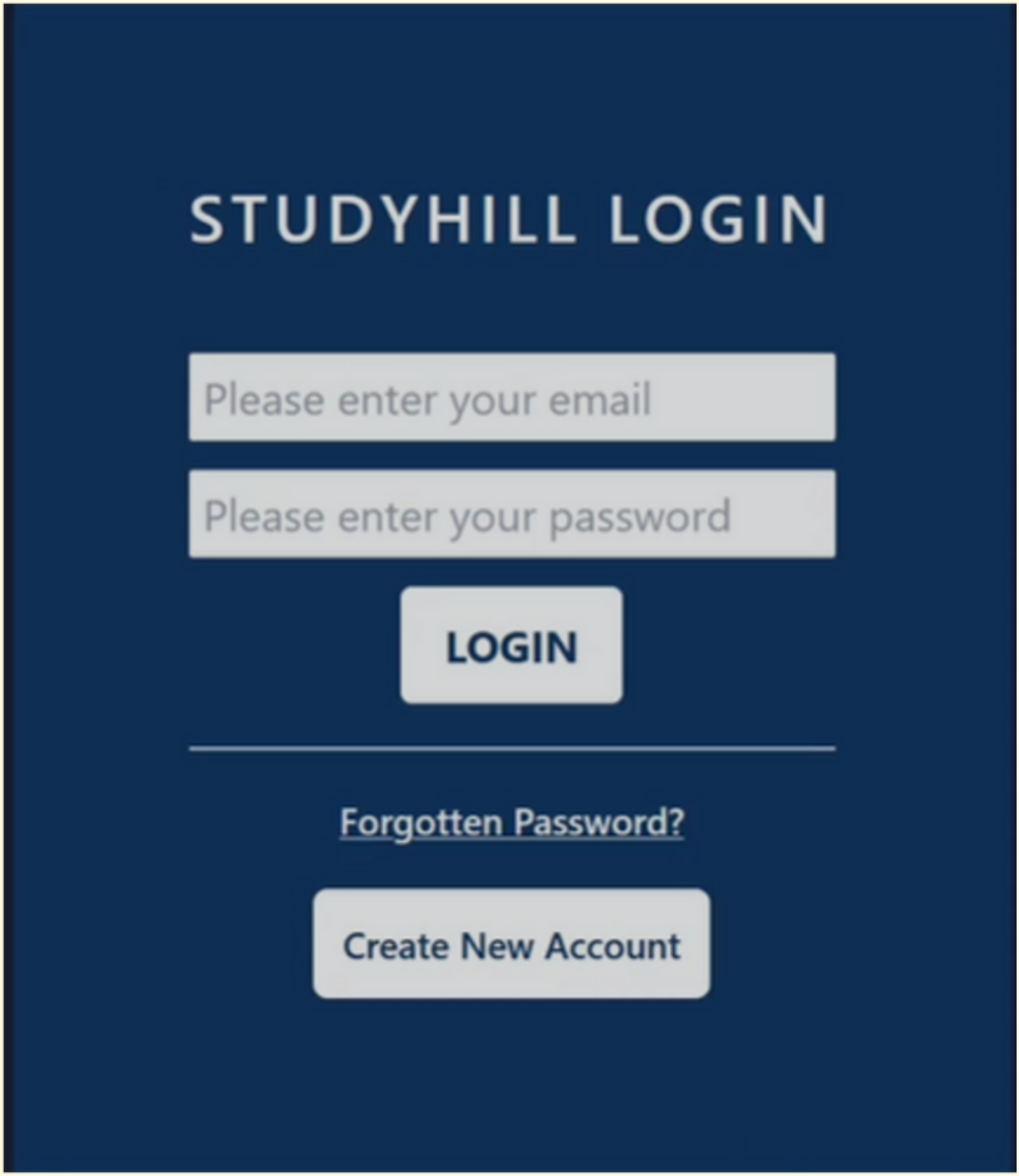
1. Go sign-in page successfully registered.
2. Show alert window if tests are not passed.

Result: Successful

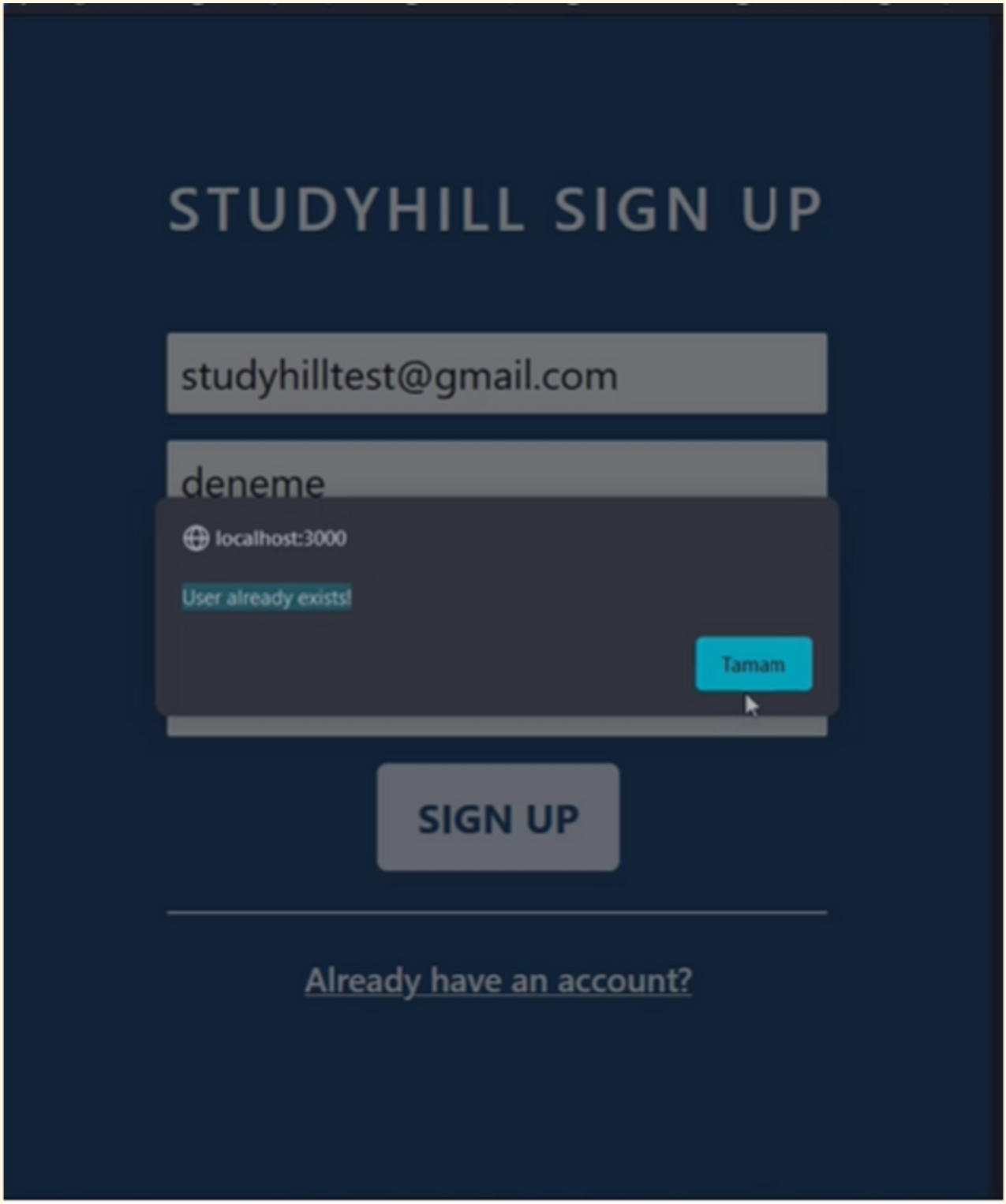
Inputs
-Email: studyhilltest@gmail.com
-Username: studystest
-Password: 123
-Password-Confirm: 1234



Inputs
-Email: studyhilltest@gmail.com (not registered before)
-Username: studystest
-Password: 123
-Password-Confirm: 123



Inputs
-Email: studyhilltest@gmail.com (registered before)
-Username: deneme
-Password: 123
-Password-Confirm: 123



5.2 LOGIN

Description

User enters his/her email address and password to login.

Tests

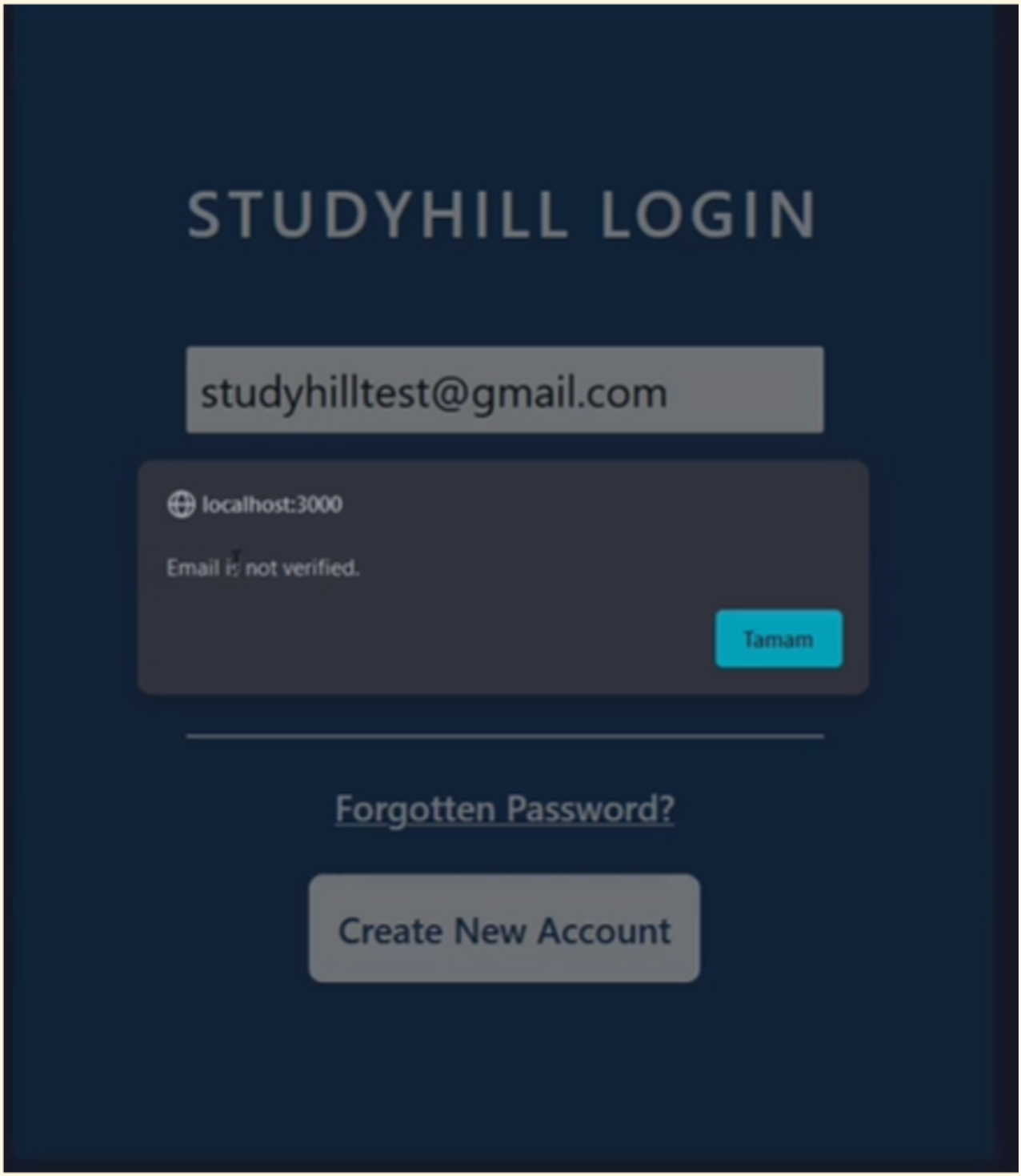
1. User should enter the correct inputs.
2. User should verify his/her account to login.

Outputs

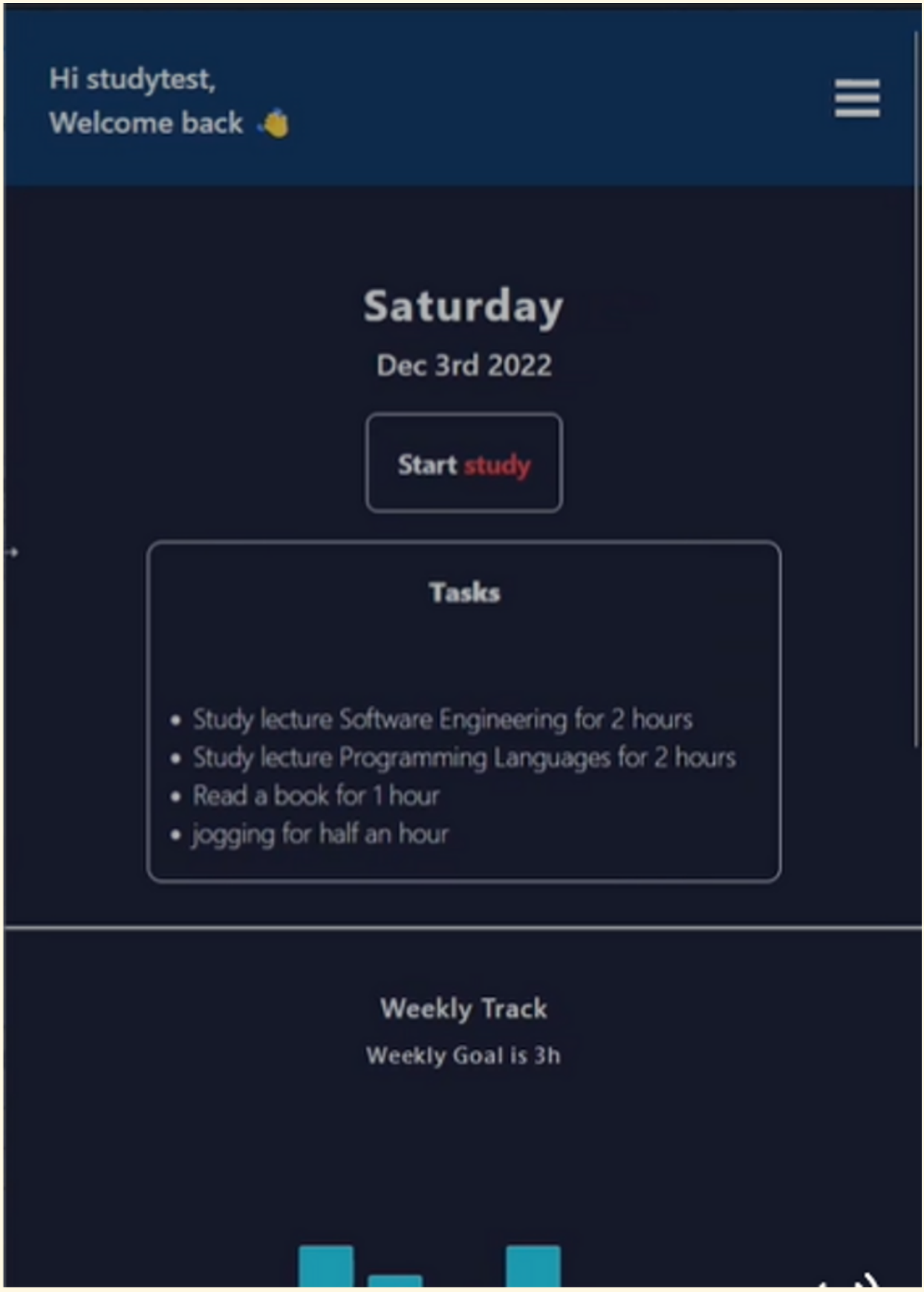
1. Go profile page if successfully logged in.
2. Show alert window if tests are not passed.

Result: Successful

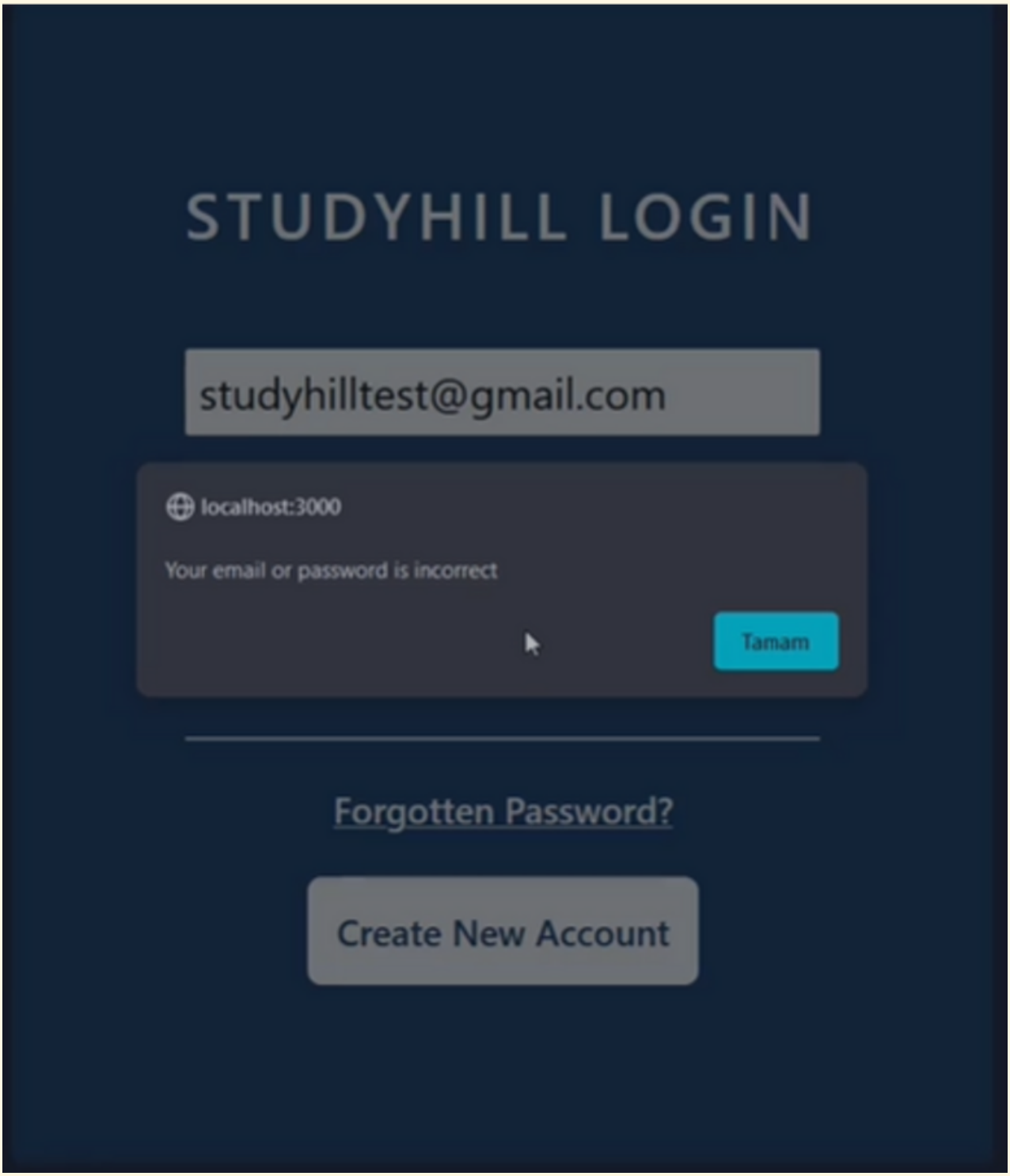
Inputs
-Email: studyhilltest@gmail.com (not verified)
-Password: 123



Inputs
-Email: studyhilltest@gmail.com (verified)
-Password: 123 (correct)



Inputs
-Email: studyhilltest@gmail.com (verified)
-Password: 123456789 (not correct)



5.3 CREATE GROUP

Description

User enters a group name and size.

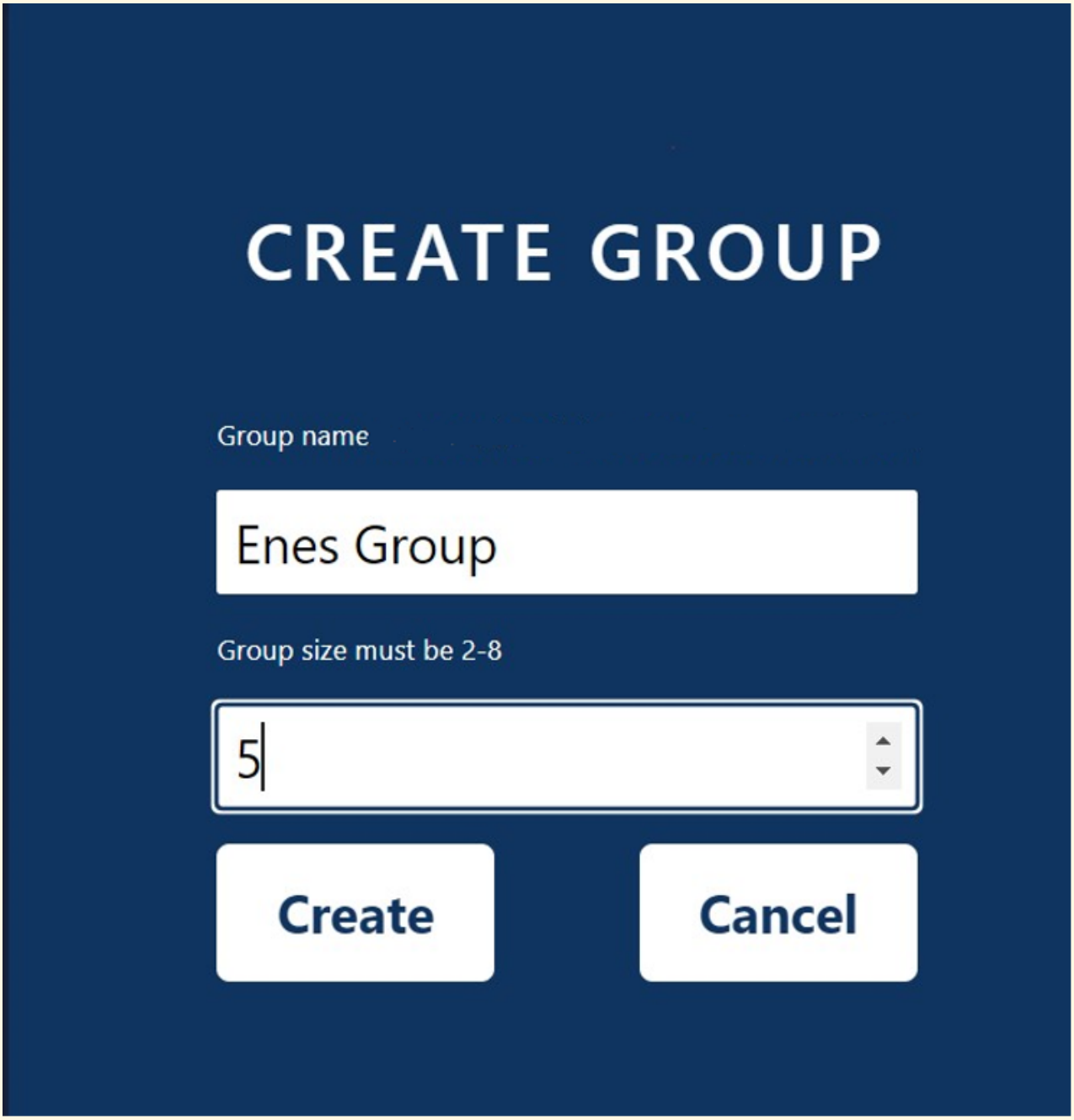
Tests

Size need to be within 2-8.

Outputs

1. Create a group
2. Return to profile page.

Result: Successful



Create button not active when size is not within 2-8.

After creating the group:

	3	vj1n	Enes Group	5	1	2023-01-02
	NULL	NULL	NULL	NULL	NULL	NULL

5.4 JOIN GROUP

Description

User enters the code and joins the group.

Tests

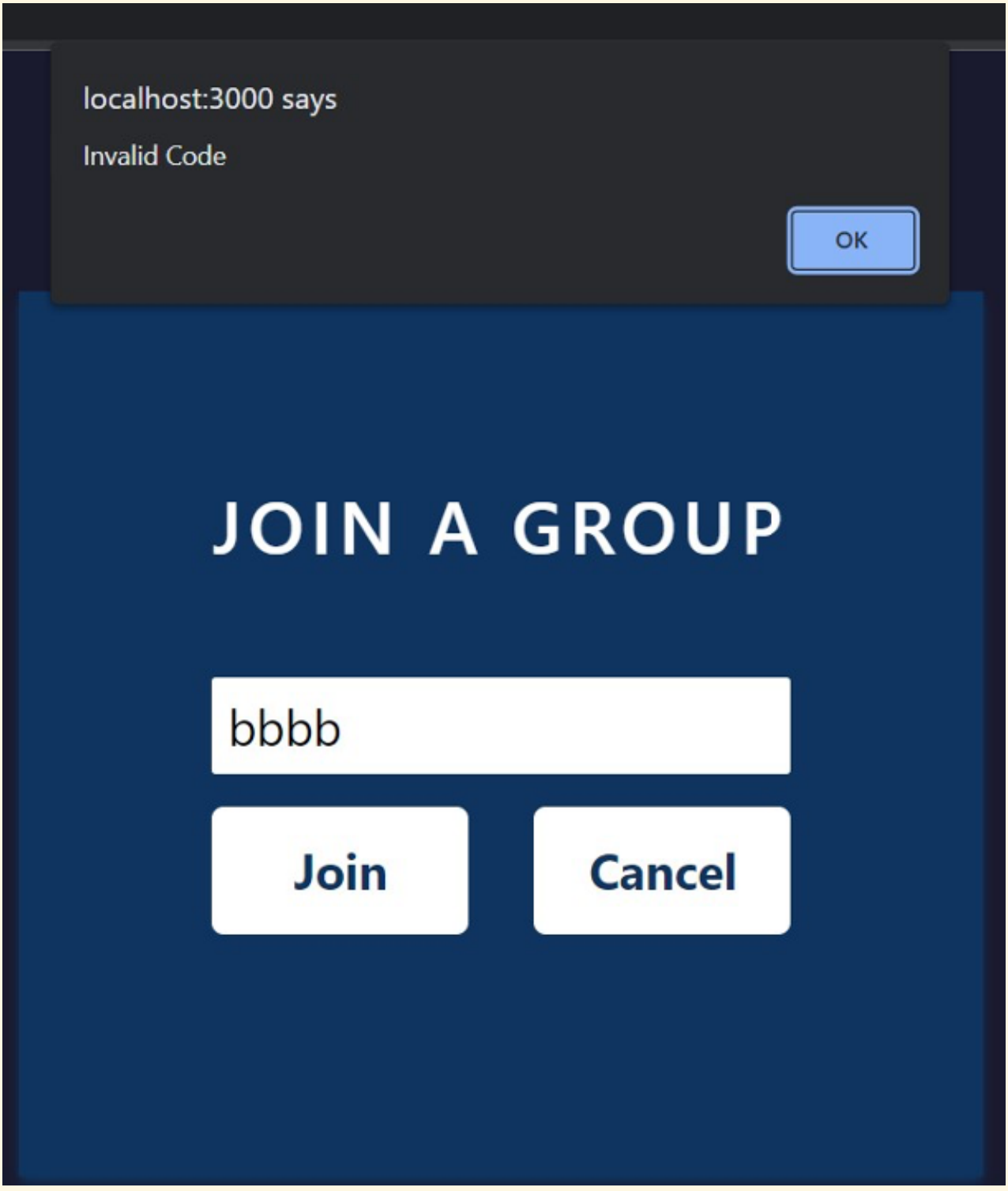
1. Group code should be valid.
2. Group shouldn’t be full.

Outputs

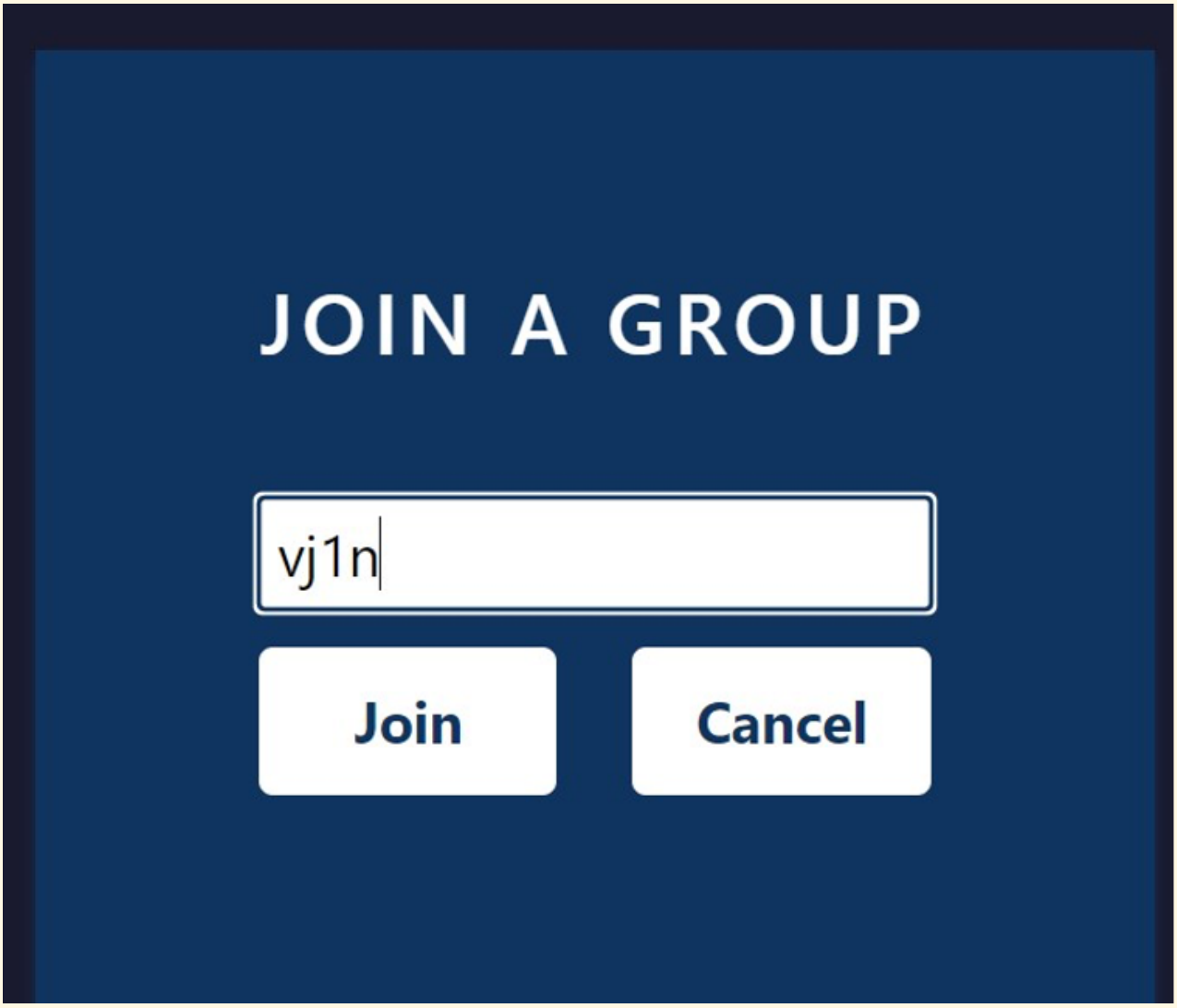
1. Alert if tests not passed.
2. Add user to the group and return to profile page.

Result: Successful

Input: bbbb (not valid)



Input: vj1n (valid)



Output: Page returned to profile page.

Database:

	id_groups	groupCode	groupName	maxSize	memberCount	mondayDate
▶	1	AAAA	firstGroup	1	1	2022-01-01
	2	axln	borangroup	3	1	2023-01-02
	3	vj1n	Enes Group	5	2	2023-01-02
*	NULL	NULL	NULL	NULL	NULL	NULL

Group table of vj1n:

	id_group_table_code	member_user_id	studyTime	username
▶	1	4	0	enes123
	3	2	0	boran
*	NULL	NULL	NULL	NULL

5.5 LEAVE GROUP

Description

User leaves the group from profile page.

Tests

-

Outputs

1. Discard user from his/her group.
2. Return to profile page.

Result: Successful

After leave the group button pressed:

Database:

id_groups	groupCode	groupName	maxSize	memberCount	mondayDate
1	AAAA	firstGroup	1	1	2022-01-01
2	axln	borangroup	3	1	2023-01-02
3	vj1n	Enes Group	5	1	2023-01-02
NULL	NULL	NULL	NULL	NULL	NULL

Group table of vj1n:

id_group_table_code	member_user_id	studyTime	username
1	4	0	enes123
NULL	NULL	NULL	NULL