PROJECT: HTTPS://GITHUB.COM/MARKFULLER1/BNU 24-SEPT-2019

BCC

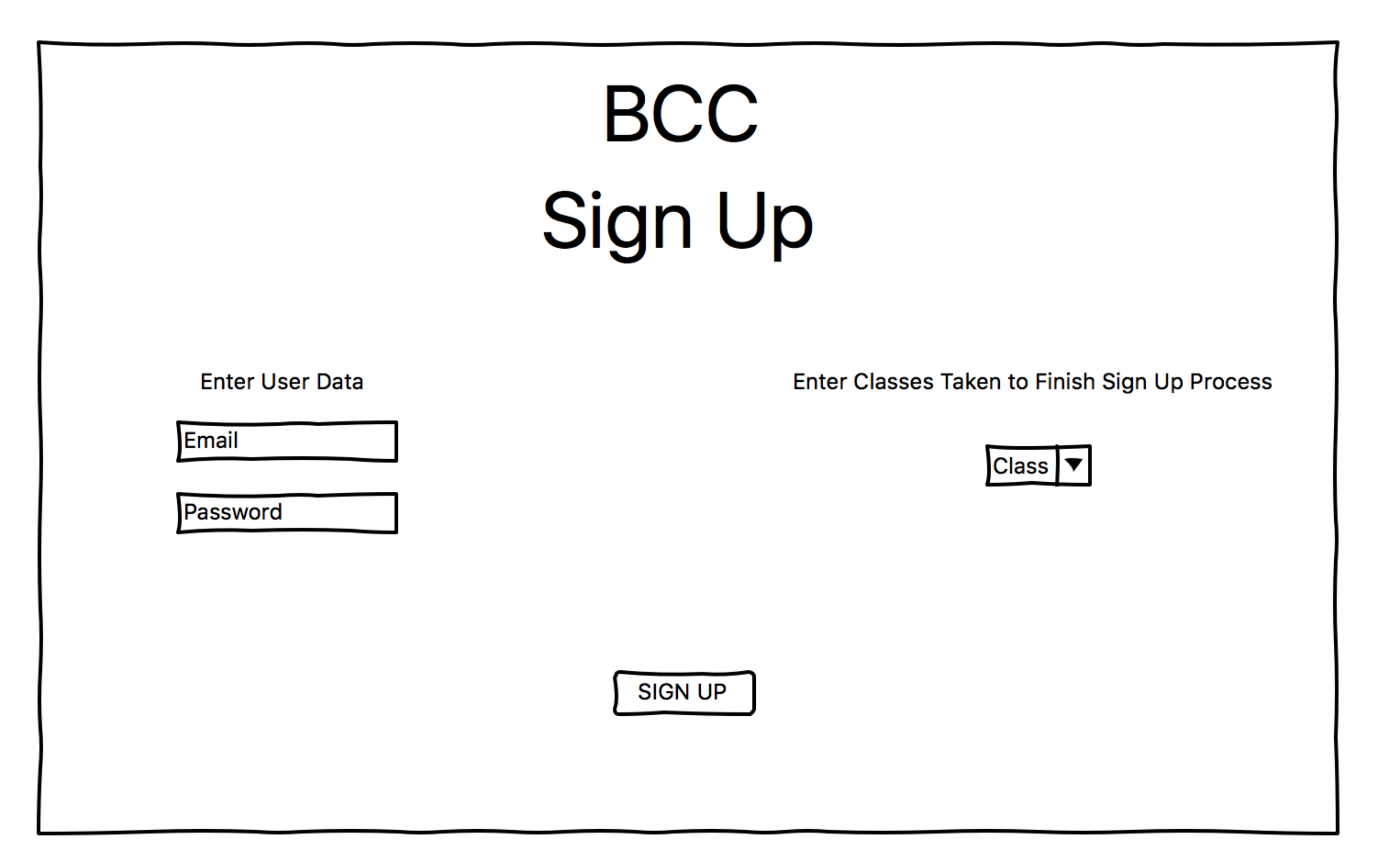
MARK FULLER, KEVIN KULDA, CONNOR WOODAHL

PROFESSOR: DR. CERNY

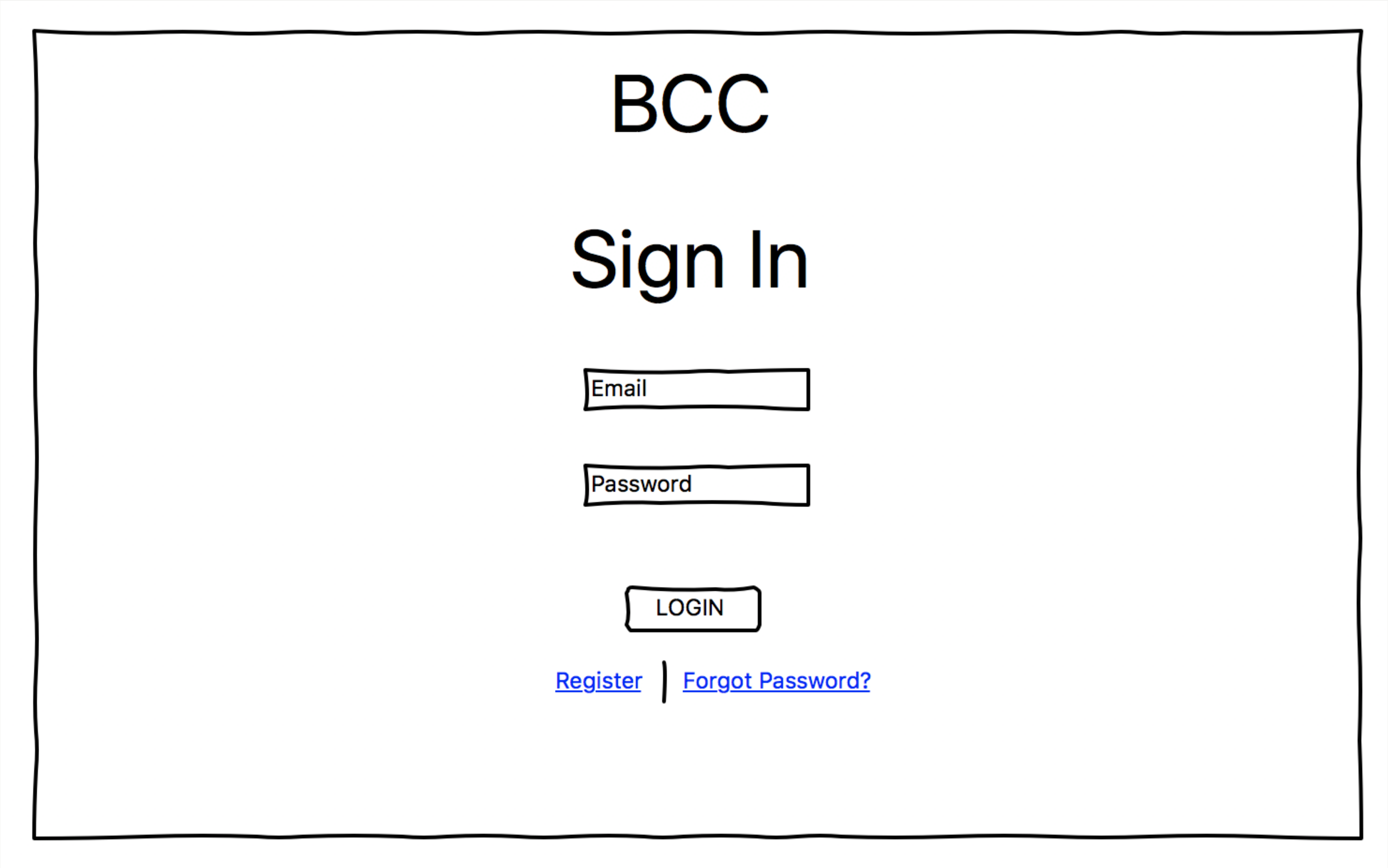
BAYLOR CLASS CONNECT

- BCC is a tool used by students to rate their current teachers, view reviews of future teachers and contact previous students who have had those teachers

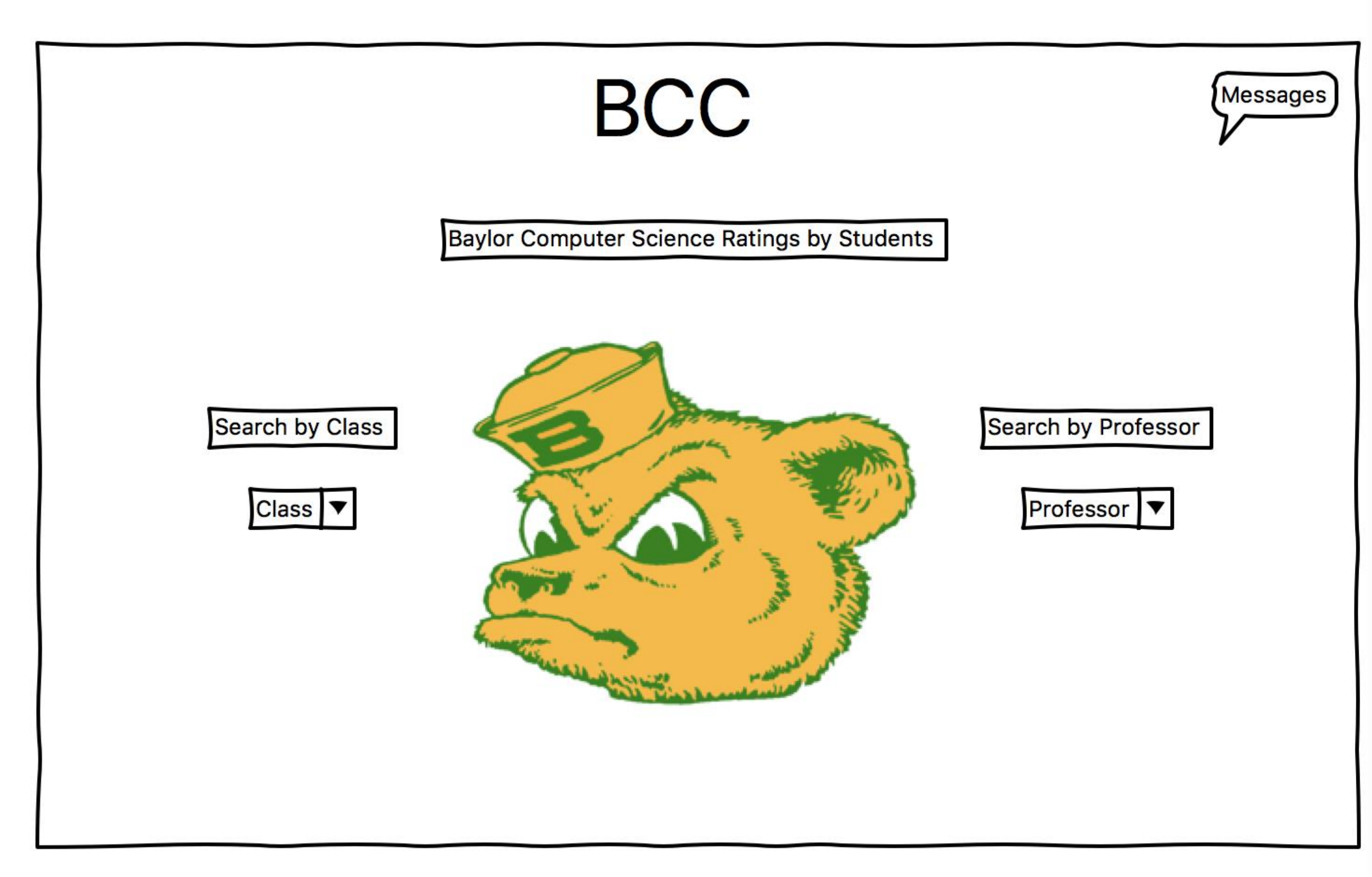
WIREFRAMES


BCC
Sign Up
Enter User Data
Email
Password
Enter Classes Taken to Finish Sign Up Process
Class ▼
SIGN UP

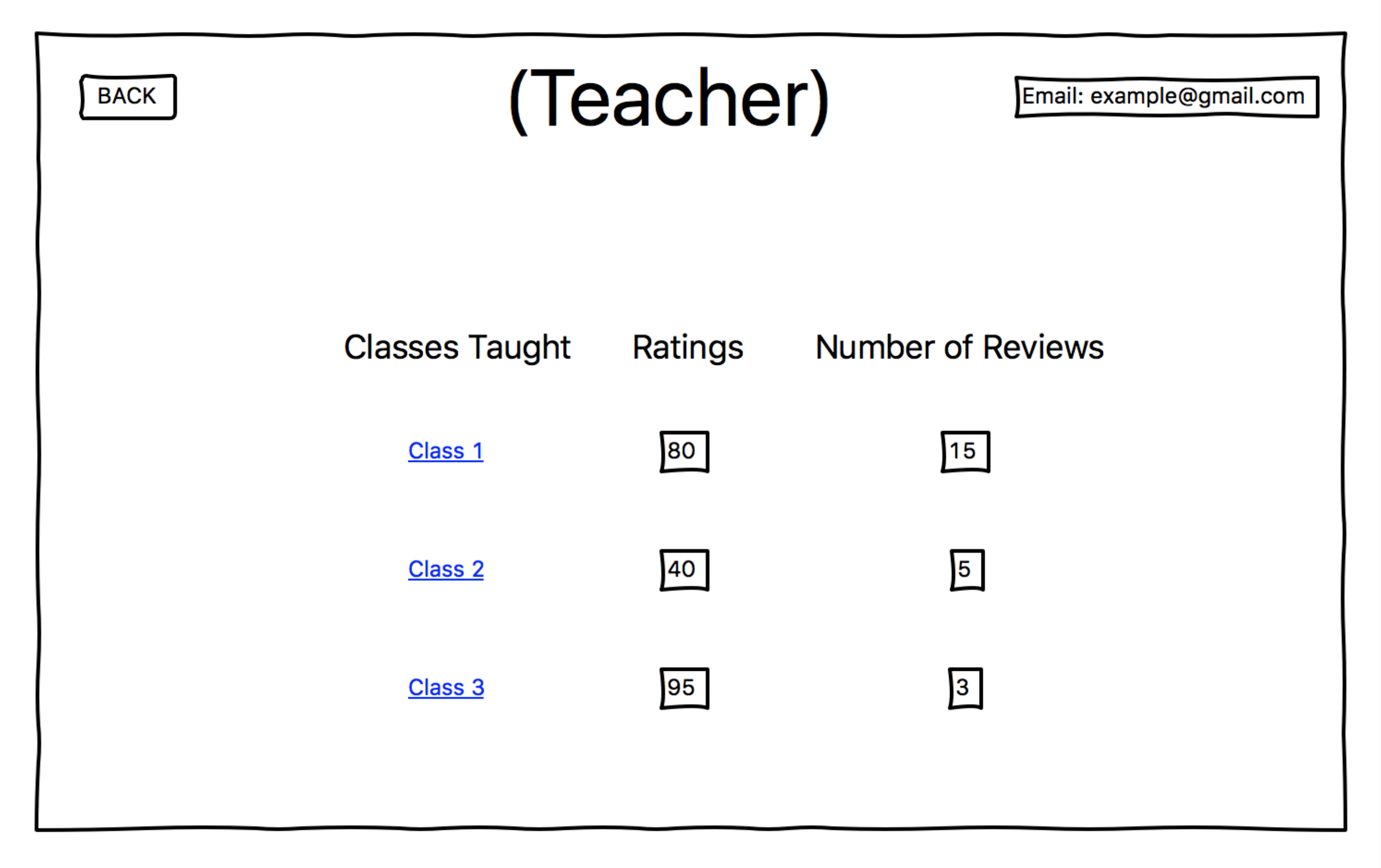
WIREFRAMES



WIREFRAMES


BCC
Messages
Baylor Computer Science Ratings by Students
Search by Class
Class ▼
Search by Professor
Professor ▼

WIREFRAMES

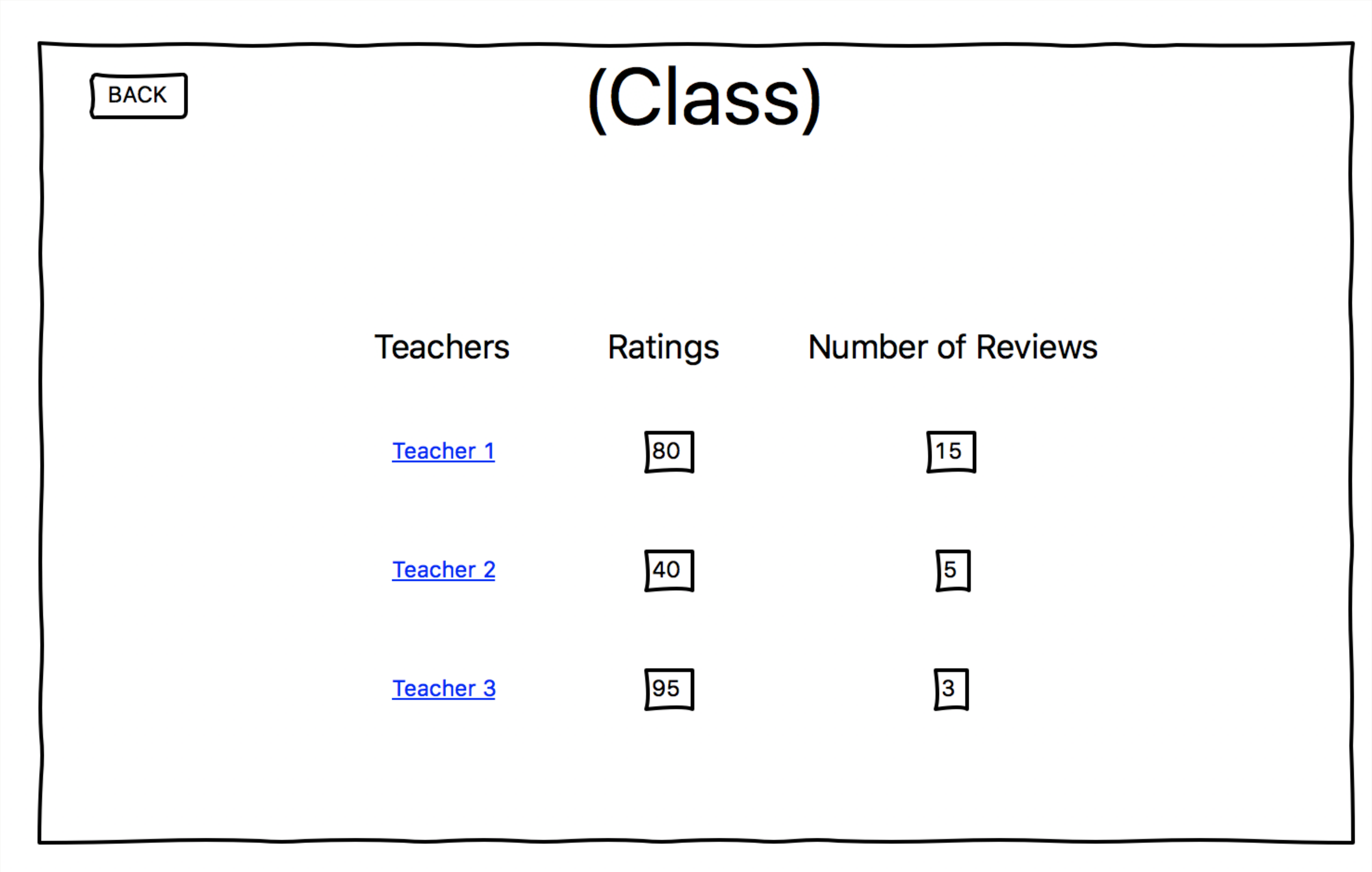


(Teacher)

Email: example@gmail.com

Classes Taught	Ratings	Number of Reviews
Class 1	80	15
Class 2	40	5
Class 3	95	3

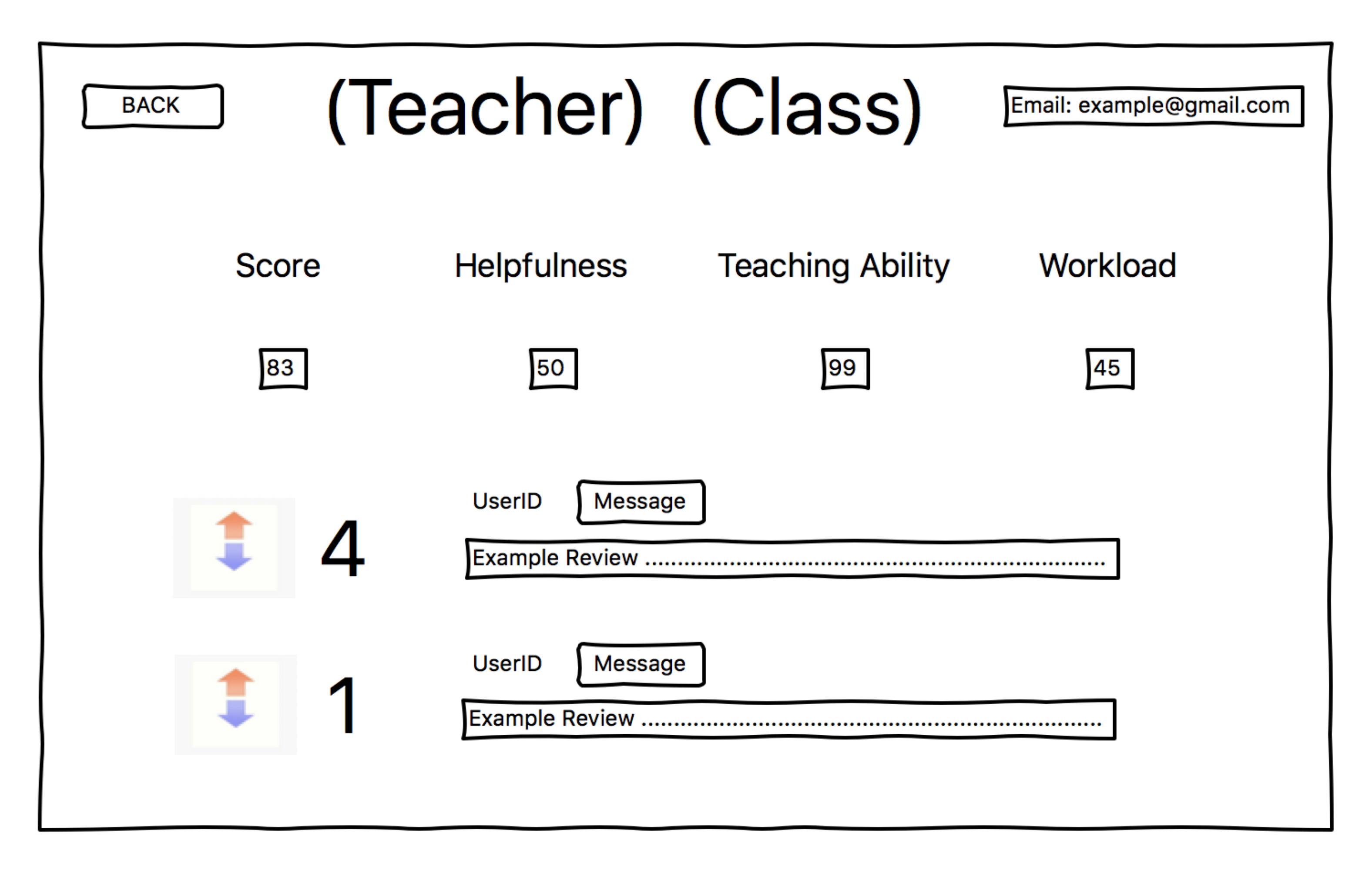
WIREFRAMES



(Class)

Teachers	Ratings	Number of Reviews
Teacher 1	80	15
Teacher 2	40	5
Teacher 3	95	3

WIREFRAMES



(Teacher) (Class)

Email: example@gmail.com

Score	Helpfulness	Teaching Ability	Workload
83	50	99	45

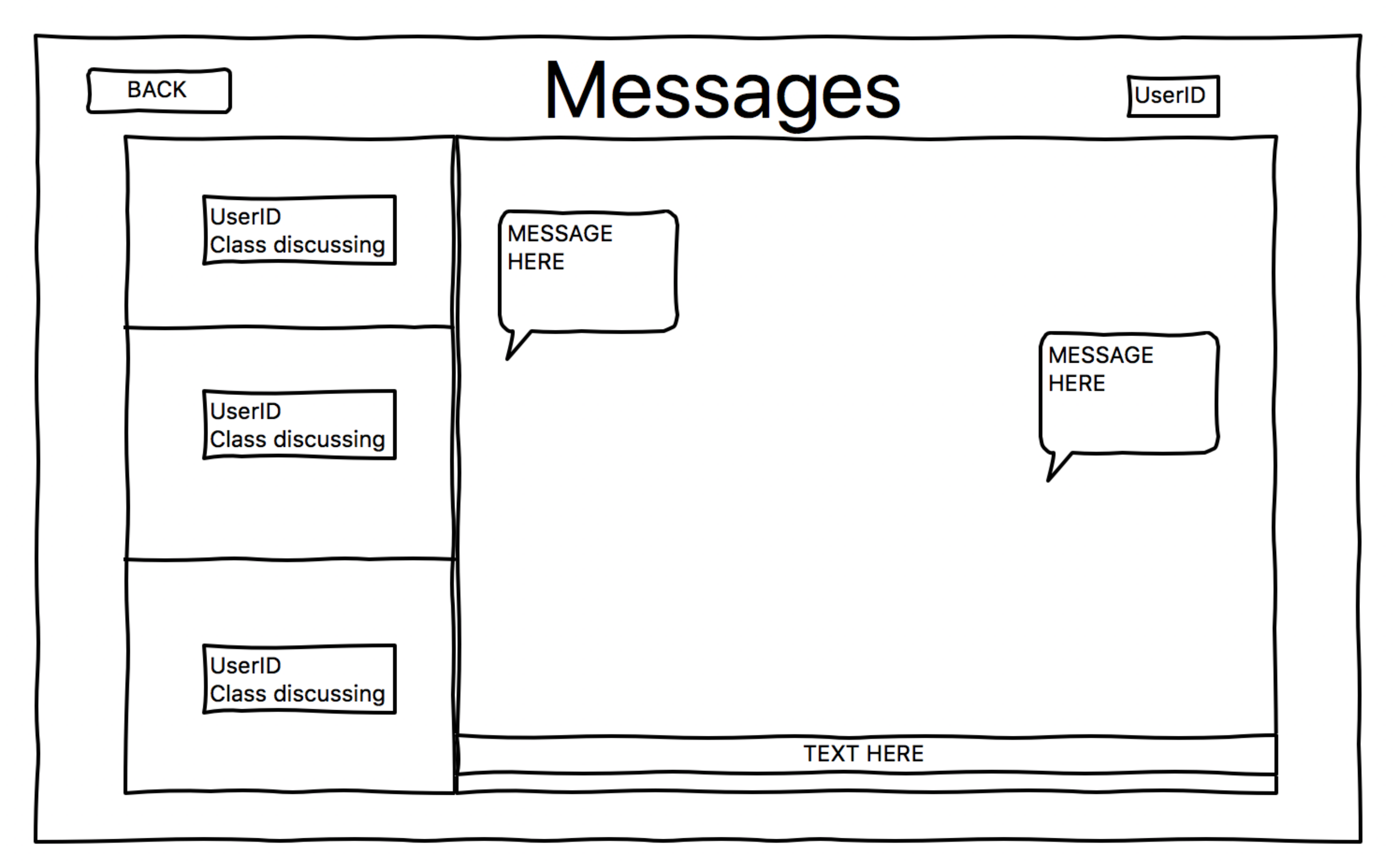
4 UserID

Example Review ....

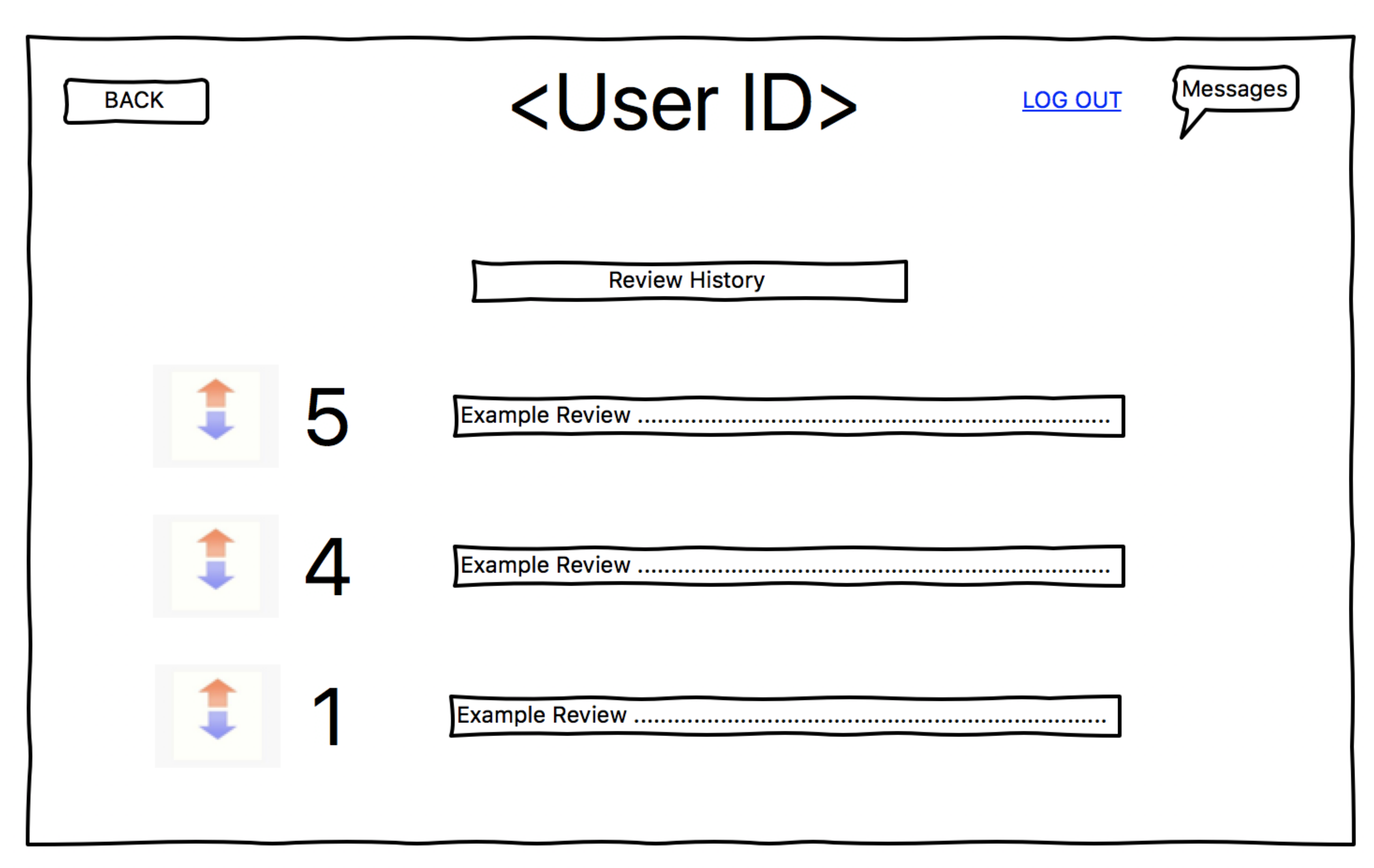
1 UserID

Example Review ....

WIREFRAMES


BACK
Messages
UserID
UserID
Class discussing
UserID
Class discussing
UserID
Class discussing
MESSAGE
HERE
MESSAGE
HERE
TEXT HERE

WIREFRAMES


BACK
<User ID>
LOG OUT
Messages
Review History
5
Example Review ....
4
Example Review ....
1
Example Review ....

REQUIREMENTS

Functional Requirements

- Users need to be able to register an account.
- Users must be able to search based on class.
- Users must be able to see professor options after searching the class.
- Users must be able to search based on professor.
- Users must be able to write a review on a professor.
- Users must be able to upvote or downvote a review.
- Users must be able to send a message to the author of the review.
- The reviewer must be able to respond to a message.
- User must be able to log out.
- The application must be able to sanitize user input.
- User must be able to navigate bidirectionally in this application.
- Users need to be able to delete an account.
- Users must be able edit a review.

Non-Functional Requirements

- Messages must be able to be sent anonymously
- User data must be kept private
- Application must be fast for the average user
- Application must handle traffic of at most 20 users at once
- User interface should be intuitive
- User interface should be graphically pleasing

ACTORS

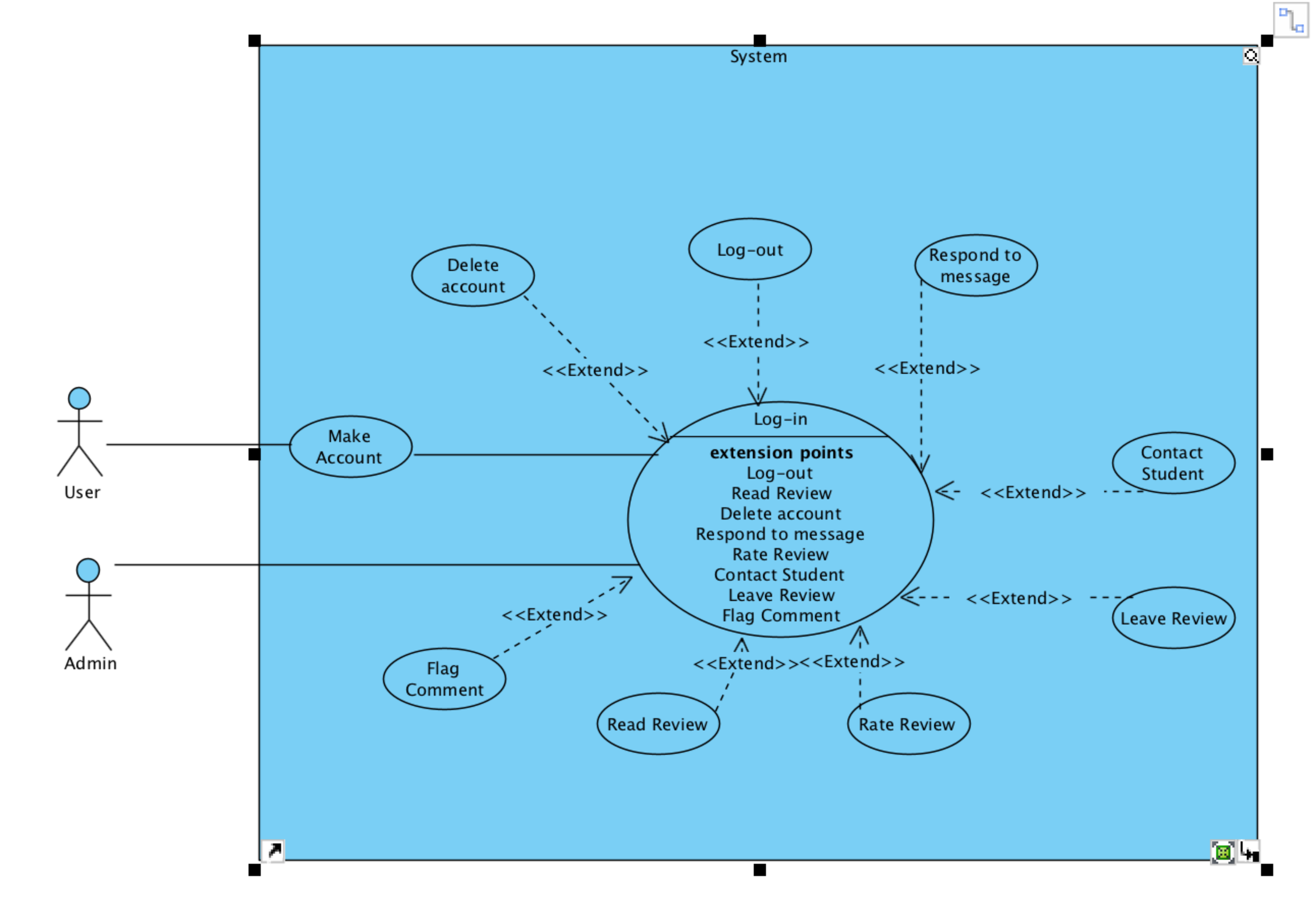
User

Admin

System

APPLICATION
MONITOR
OPERATING SYSTEMS
HARD DRIVE
KEYBOARD & MOUSE
PRINTER

USE CASE DIAGRAM



USE CASE DIVISION

- Use Cases handled by Kevin– Make Account, Respond to a Message, Flag Comment
- Use Cases handled by Mark – Leave Review, Rate Review, Read Review
- Use Cases handled by Connor – Contact Student, Edit Review, Delete Account, Log Out

MAKE AN ACCOUNT

Make an Account

Scope: GCC Application

Level: User Goal

Primary Actor: Student

Stakeholders and Interests:

Student User: Wants username and password stored and hashed for security.

Preconditions: Student has loaded the site

Success Guarantee: Students data is hashed and saved into database. Immediate access to the application is granted.

Main Success Scenario:

Student loads application.

Student requests a new account.

Student enters Username, Password and, email, etc.

System updates valid students list with new student.

Student Logs in.

Alternative Flows:

a. At any time the student closes the application.

1. Interaction ends.

b. At any time system fails.

1. Student will restart application.

c. While student is on register page, they would like to login with valid credentials(the realized they had an account already)

1. Student clicks `return to login` and logs in with valid username and password.

d. Username and password from account creation do not work.

1. Prompt student to reset password with security question.

1.1. Student answers security question correctly

1.1.1. Student resets password.

1.2. Student does not answer security question correctly

1.2.1 Student is returned to login page.

LEAVE REVIEW

Leave Review

Scope: GCC Application

Level: User Goal

Primary Actor: Student

Stakeholders and Interests:

- Student User:
  - Wants Username and Password to be recognized.
  - Wants review to be saved for others to view
  - Student has valid login credentials
- Preconditions:
  - Student has loaded the program.

Success Guarantee:

- Student review is made available to other users on the site.

Main Success Scenario:

- Student loads application.
- Student logs in.
- Student selects professor they would like to review from drop-down menu
- Student is brought to professors review page
- Student writes review
- Student saves review

Alternative Flows:

a. At any time the student closes the application.

1. Interaction ends.

b. At any time system fails.

1. Student will restart application.

c. Student picks wrong professor

1. Student clicks `back`

2. Selects correct professor from drop-down list

d. Username and password do not work.

1. Prompt student to reset password with security question.

 1.1. Student answers security question correctly

  1.1.1. Student resets password.

 1.2. Student does not answer security question correctly

  1.2.1 Student is returned to login page.

CONTACT STUDENT

Contact Student

Scope: GCC Application

Level: User Goal

Primary Actor: Student

Stakeholders and Interests:

Student User:

Wants Username and Password to be recognized.

Wants review to be saved for others to view

Wants contact with other Student to be over message board

Preconditions:

Student has loaded the program.

Student has valid login credentials

Success Guarantee:

Student is able to message other Student user (two-way anonymity)

Main Success Scenario:

Student loads application.

Student logs in.

Student selects professor they would like to view reviews of

Student is brought to professors review page

Student selects, contact reviewer button

Student creates message

Student Sends message

Opposing student responds to message

Message Notification shows up on login of user

Alternative Flows:

a. At any time the student closes the application.

1. Interaction ends.

b. At any time system fails.

1. Student will restart application.

c. Student picks wrong professor

1. Student clicks `back`

2. Selects correct professor from drop-down list

d. Username and password do not work.

1. Prompt student to reset password with security question.

1.1. Student answers security question correctly

1.1.1. Student resets password.

1.2. Student does not answer security question correctly

1.2.1 Student is returned to login page.

CASUAL USE CASES

Respond to a Message

Main Success Scenario

The user clicks on the message board icon on the main page. Selecting the message they want to respond to, they use the messing capability of the application to respond to the message.

Alternative Success Scenario

Edit Review

Main Success Scenario

The user logs in and navigates to their profile where all their reviews are listed. Using the editor, they change the content of their review and save changes.

Alternative Success Scenario The user logs in and navigates to their review by selecting the professor and finding their review under the professors page. Using the editor, they change the content of their review and save changes.

Delete Account

Main Success Scenario

While the user is logged in, the user navigates to their user profile settings and select the delete account button. After confirming your selection they are logged out.

Alternative Success Scenario

Rate Review

Main Success Scenario

User logs in with valid credentials. Selects the professor and class to read reviews for. Clicking the up-vote button if the review is accurate will increase the reviews score.

Alternative Success Scenario

User logs in with valid credentials. Selects the professor and class to read reviews for. Clicking the down-vote button if the review is inaccurate will decrease the reviews score.

Flag Comment

Main Success Scenario

User logs in with valid credentials. Selects the professor and class to read reviews for. If a review is inappropriate or uses vulgar language, the users flags it.

Alternative Success Scenario

Read Review

Main Success Scenario

The user will login with valid credentials. Using the navigation page, selects a professor and the desired class to load reviews for. Reviews load in chronological order.

Alternative Success Scenario

User logs in with valid credentials. Using the navigation page, selects a class and the desired professor to load reviews for. Reviews load in chronological order.

Log Out

Main Success Scenario

While the user is logged in, the user navigates to their user profile settings and logs out.

Alternative Success Scenario

The user is automatically logged out when they close the application.

GIT HUB PAGE

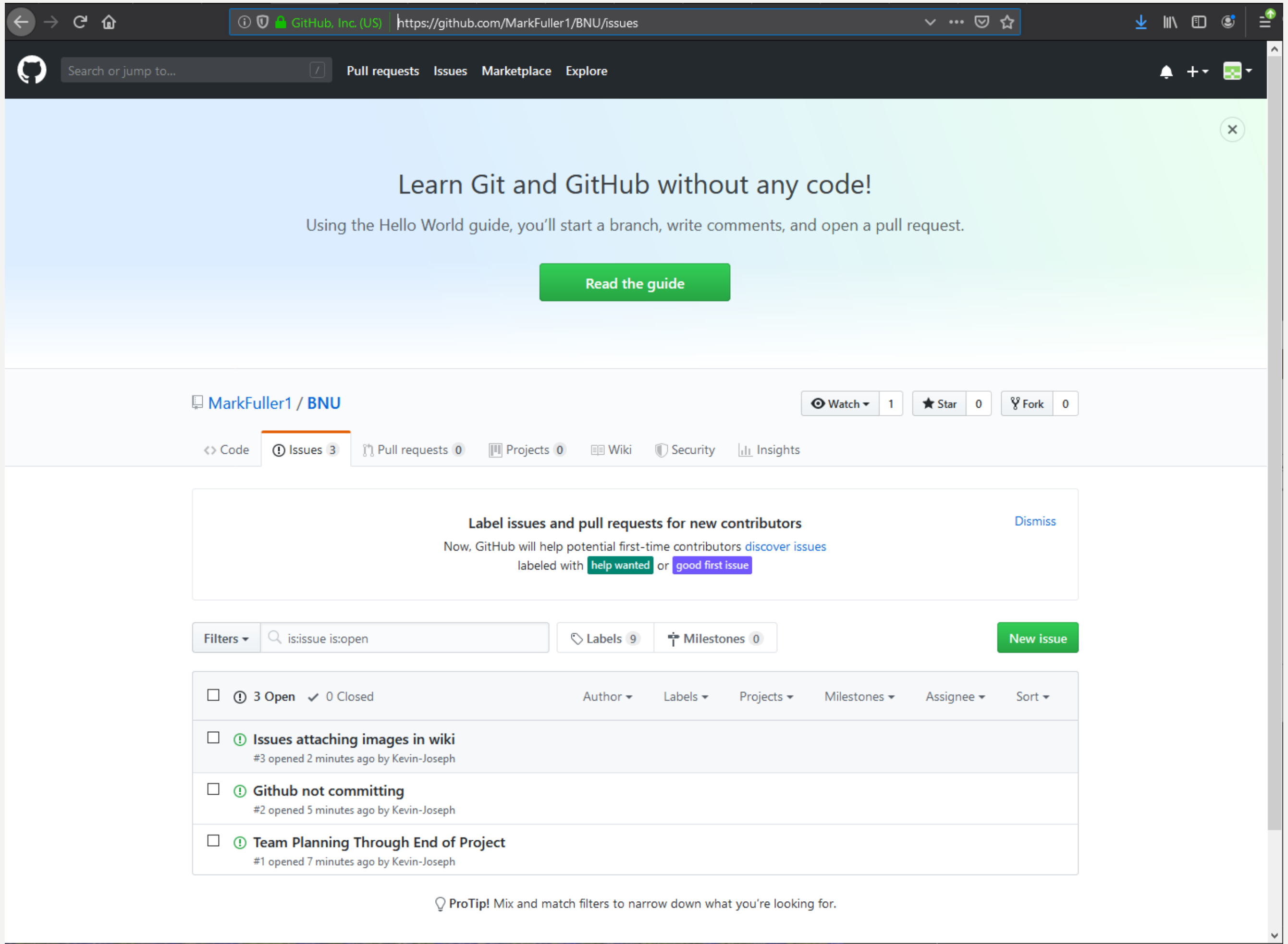
Please see our Github page for the current versions of our development materials and the full repository of our project artifacts.

https://github.com/MarkFuller1/BNU/

ISSUE TRACKING

- Please see our issue tracking repository on Github at the link below:

https://github.com/MarkFuller1/BNU/issues



Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with help wanted or good first issue

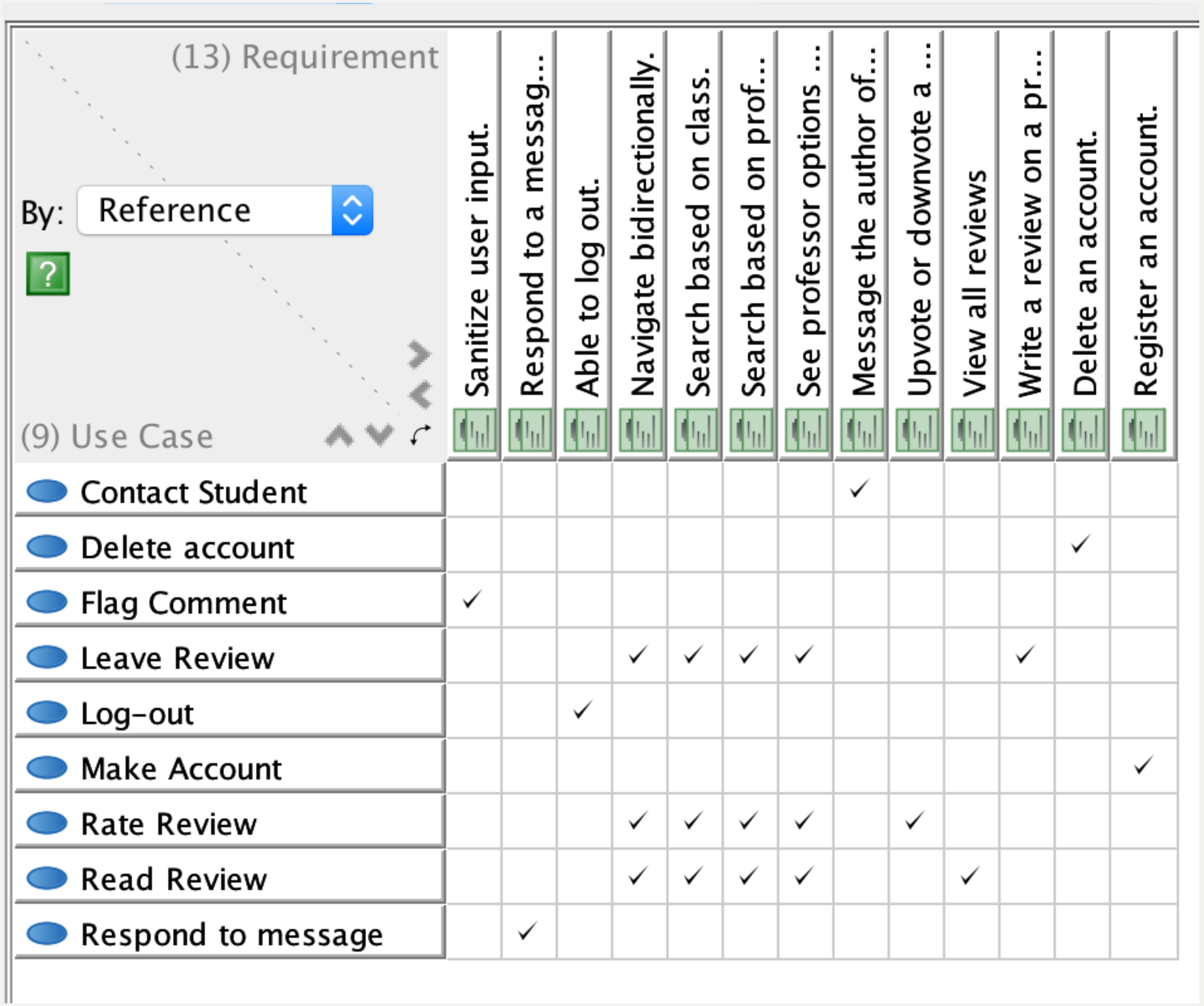
□ Issues attaching images in wiki
#3 opened 2 minutes ago by Kevin-Joseph

□ Github not committing
#2 opened 5 minutes ago by Kevin-Joseph

□ Team Planning Through End of Project
#1 opened 7 minutes ago by Kevin-Joseph

ProTip! Mix and match filters to narrow down what you're looking for.

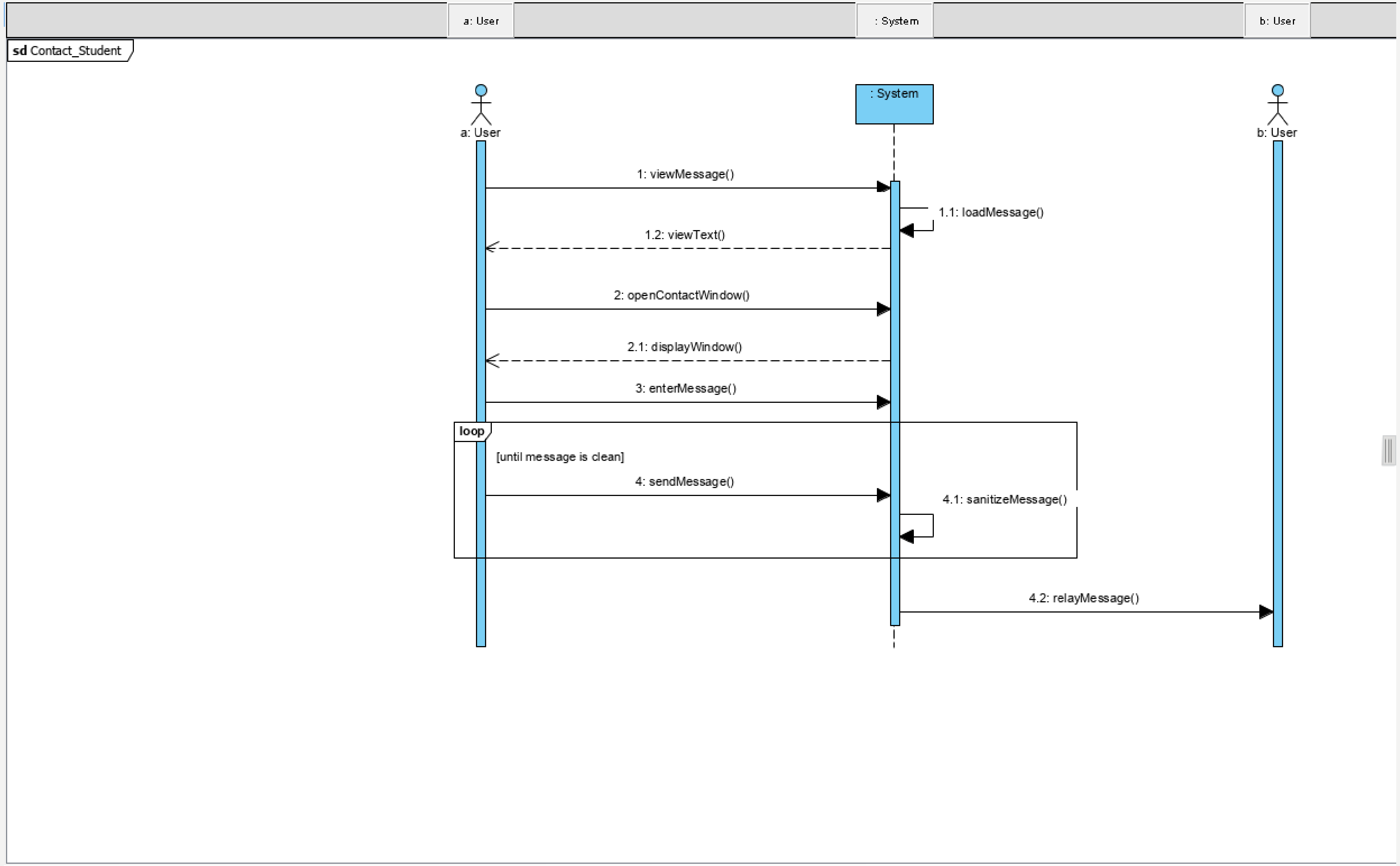
TRACEABILITY MATRIX



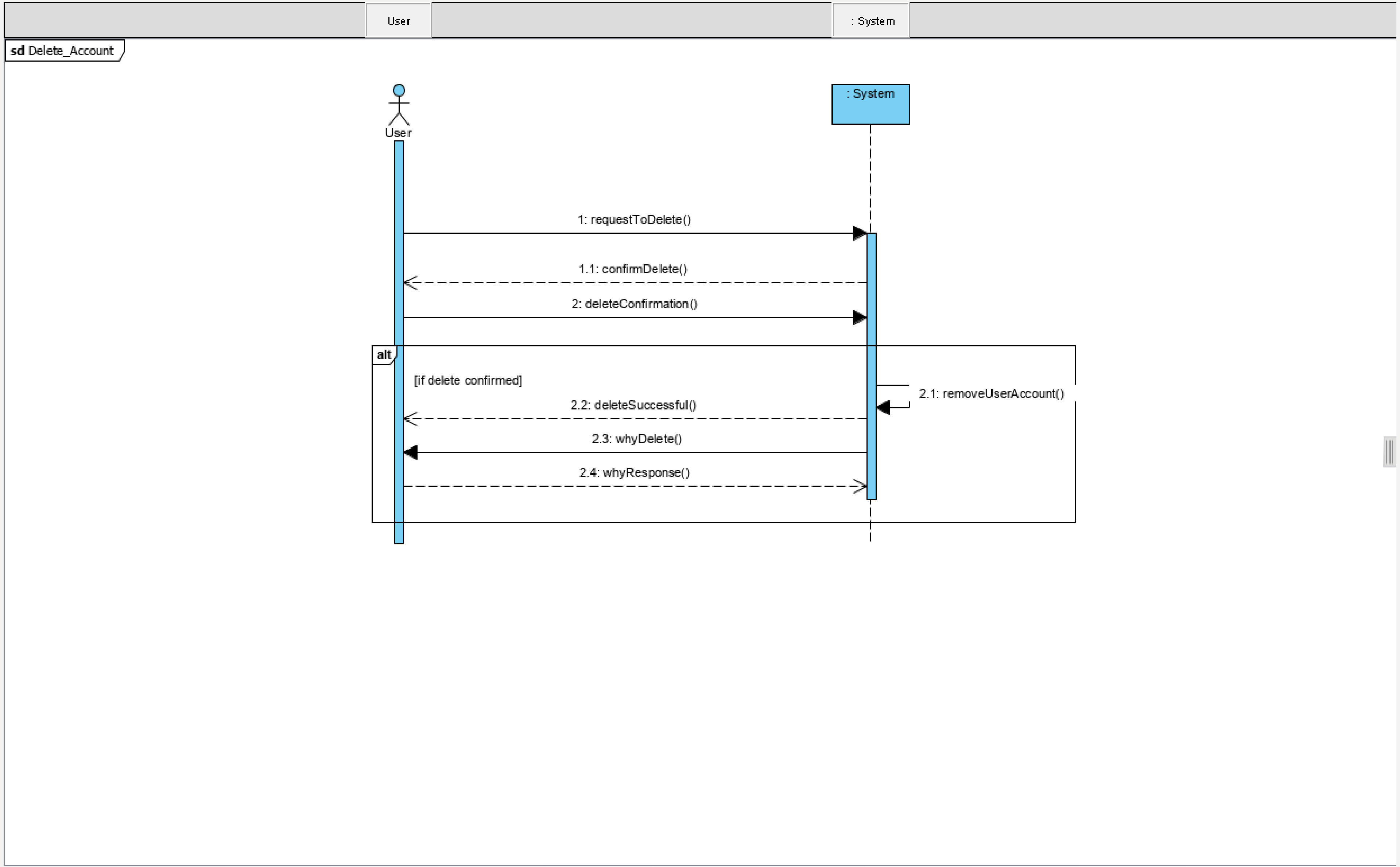
By: Reference

(9) Use Case \ (13) Requirement	Sanitize user input.	Respond to a messag...	Able to log out.	Navigate bidirectionally.	Search based on class.	Search based on prof...	See professor options ...	Message the author of...	Upvote or downvote a ...	View all reviews	Write a review on a pr...	Delete an account.	Register an account.
Contact Student								✓					
Delete account												✓	
Flag Comment	✓												
Leave Review				✓	✓	✓	✓				✓		
Log-out			✓										
Make Account													✓
Rate Review				✓	✓	✓	✓		✓				
Read Review				✓	✓	✓	✓			✓			
Respond to message		✓											

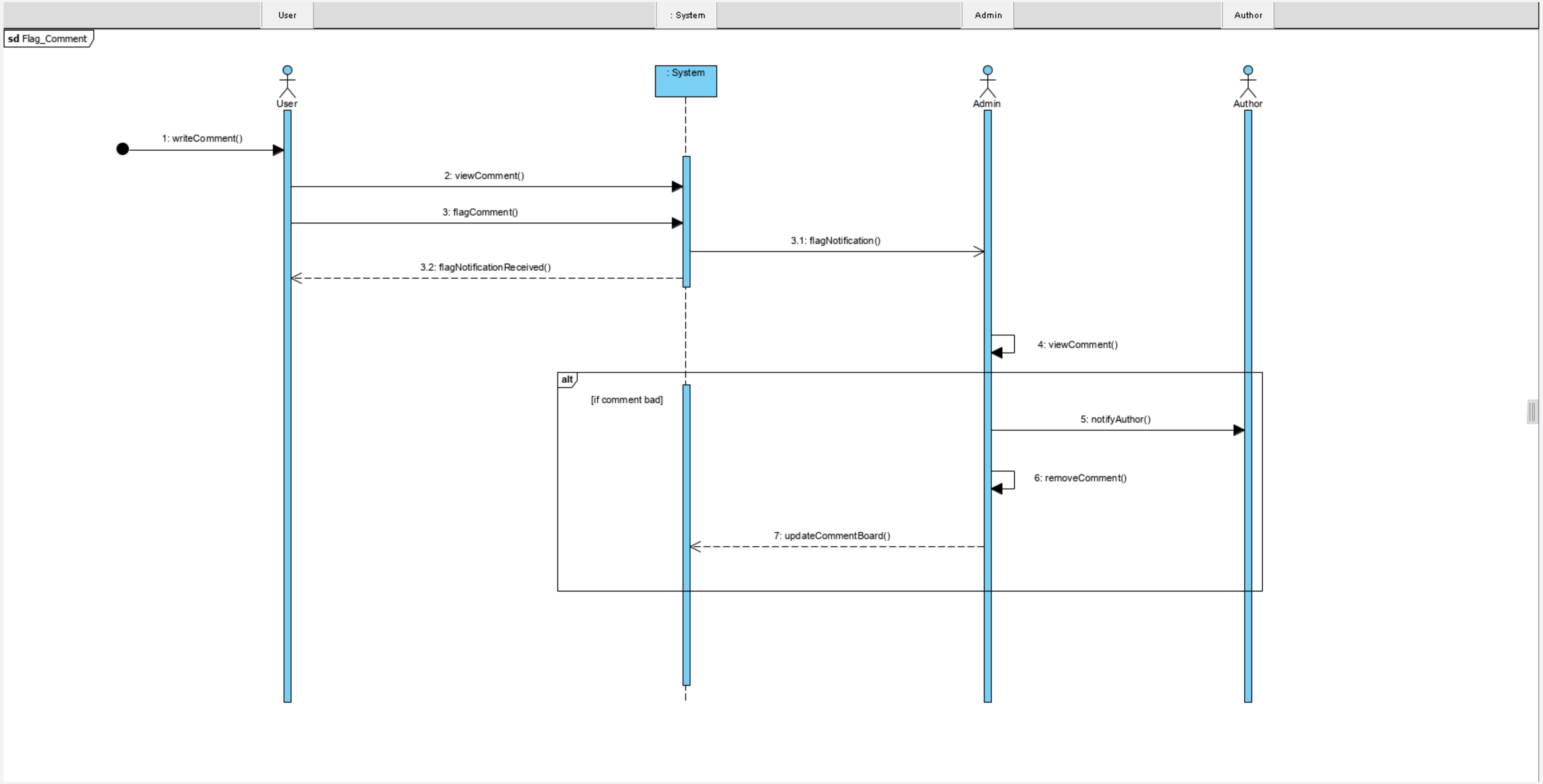
SSD


a: User
: System
b: User
sd Contact_Student
a: User
: System
b: User
1: viewMessage()
1.1: loadMessage()
1.2: viewText()
2: openContactWindow()
2.1: displayWindow()
3: enterMessage()
loop
[until message is clean]
4: sendMessage()
4.1: sanitizeMessage()
4.2: relayMessage()

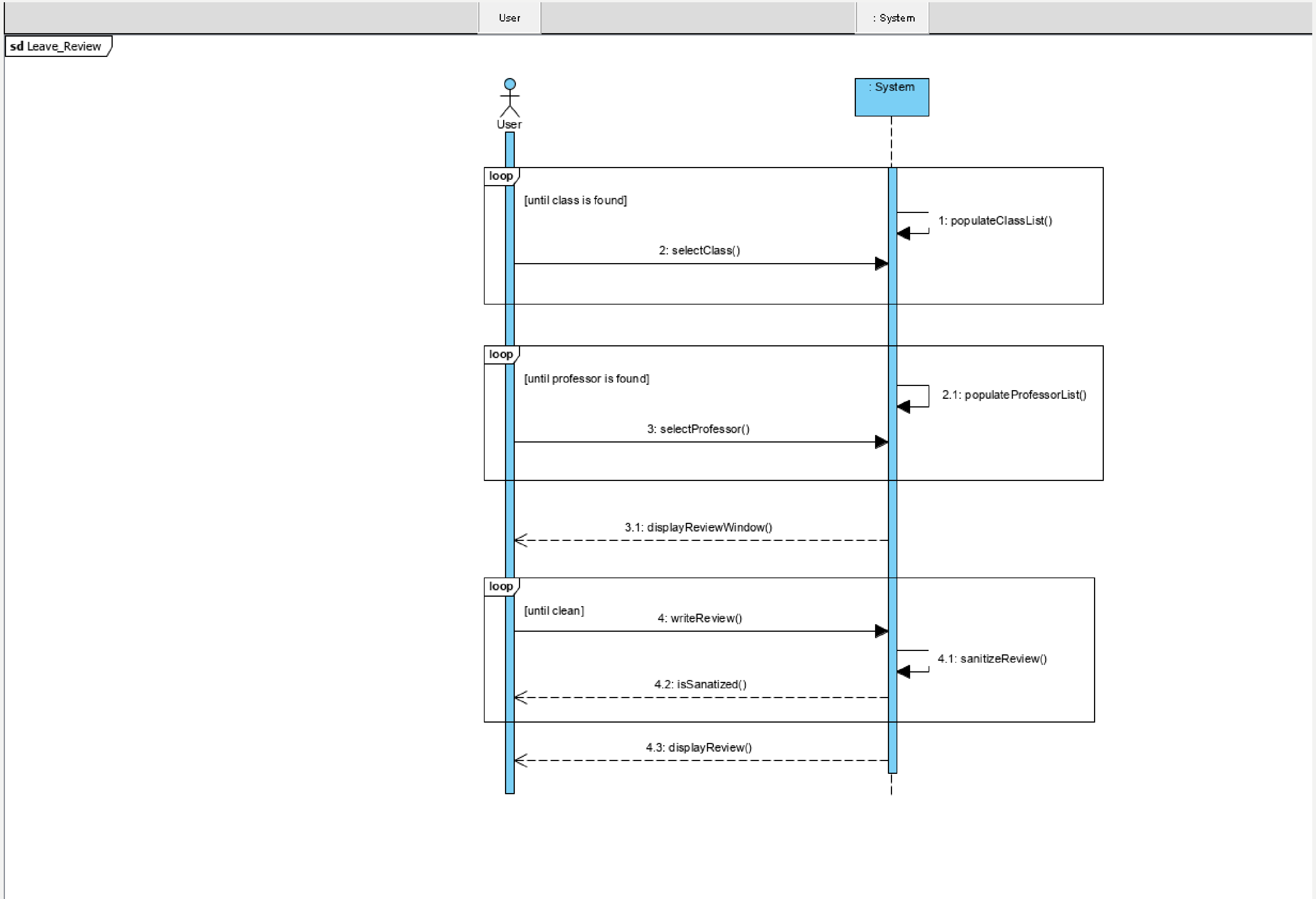
SSD


User
: System
sd Delete_Account
User
: System
1: requestToDelete()
1.1: confirmDelete()
2: deleteConfirmation()
alt
[if delete confirmed]
2.1: removeUserAccount()
2.2: deleteSuccessful()
2.3: whyDelete()
2.4: whyResponse()

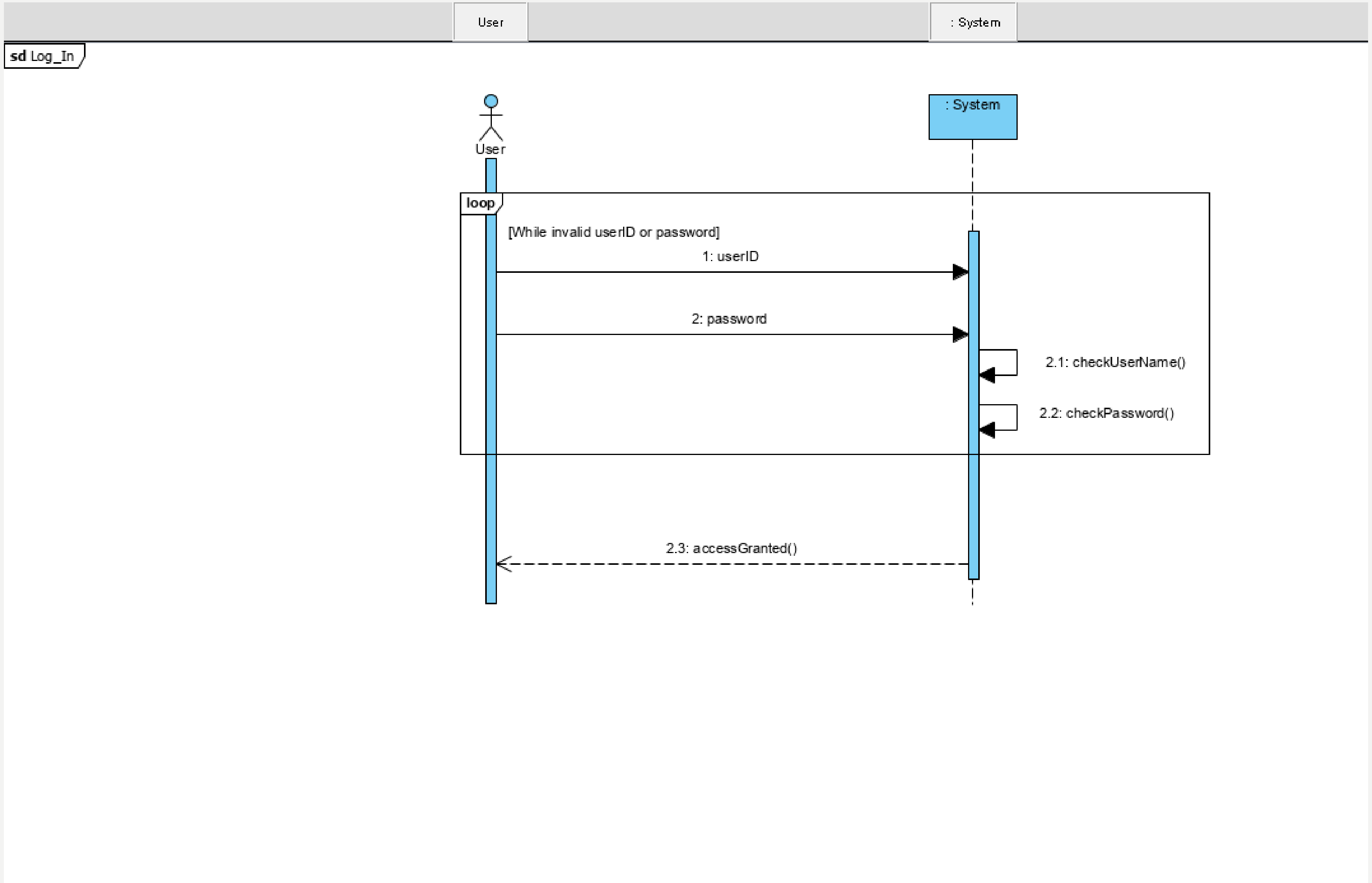
SSD


User
: System
Admin
Author
sd Flag_Comment
: System
User
Admin
Author
1: writeComment()
2: viewComment()
3: flagComment()
3.1: flagNotification()
3.2: flagNotificationReceived()
4: viewComment()
alt
[if comment bad]
5: notifyAuthor()
6: removeComment()
7: updateCommentBoard()

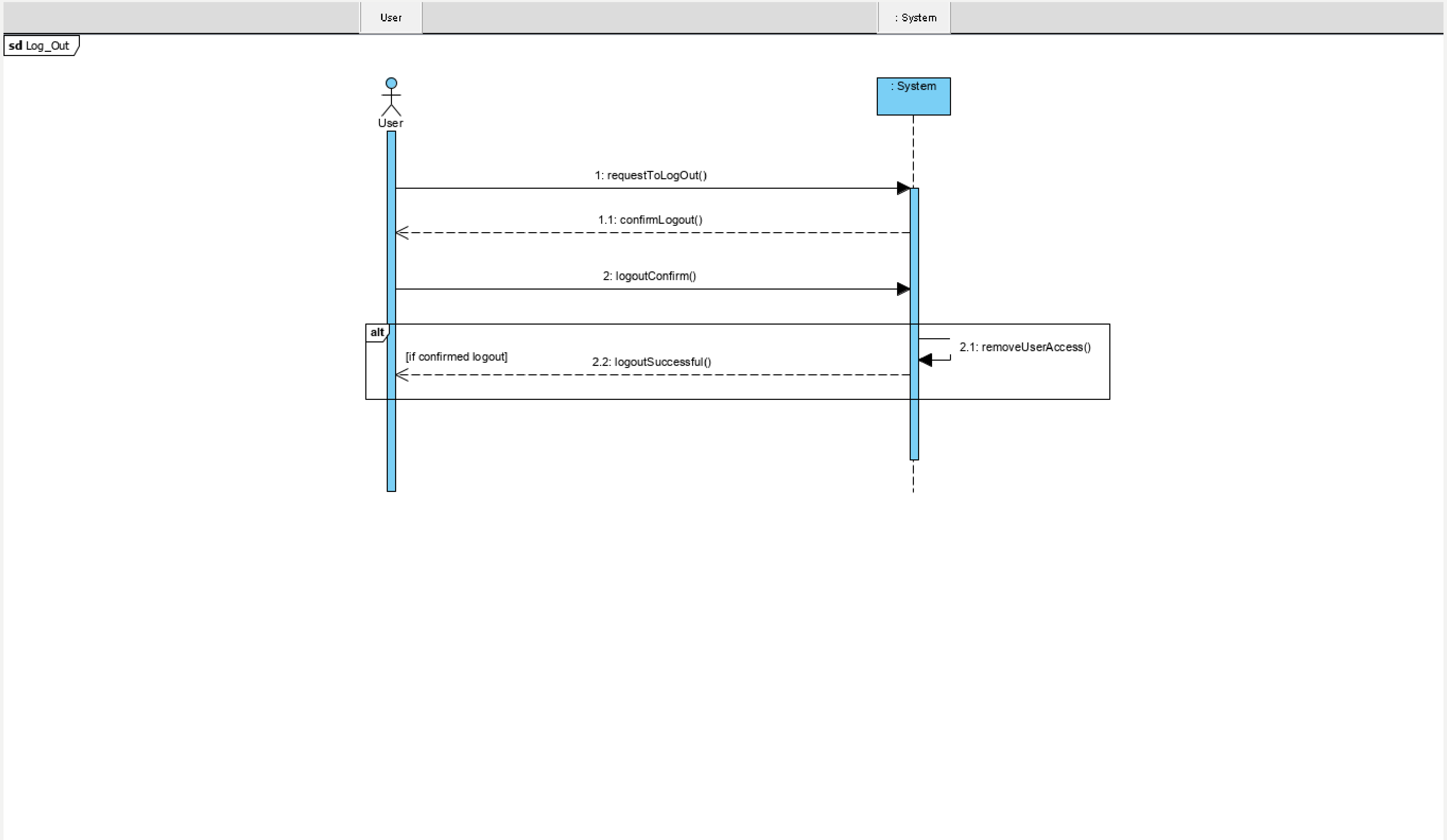
SSD


User
: System
sd Leave_Review
User
: System
loop
[until class is found]
1: populateClassList()
2: selectClass()
loop
[until professor is found]
2.1: populateProfessorList()
3: selectProfessor()
3.1: displayReviewWindow()
loop
[until clean]
4: writeReview()
4.1: sanitizeReview()
4.2: isSanatized()
4.3: displayReview()

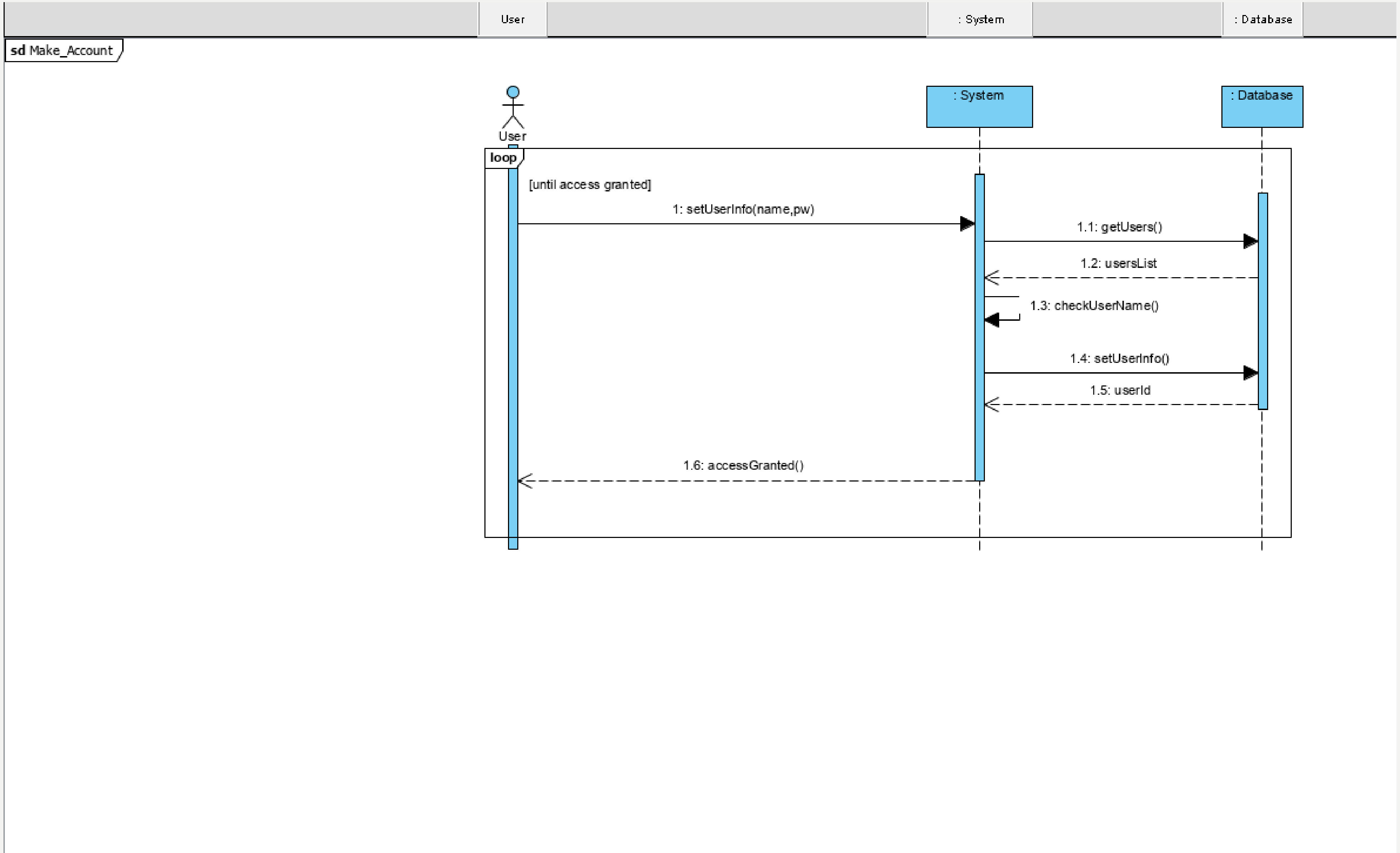
SSD


User
: System
sd Log_In
User
: System
loop
[While invalid userID or password]
1: userID
2: password
2.1: checkUserName()
2.2: checkPassword()
2.3: accessGranted()

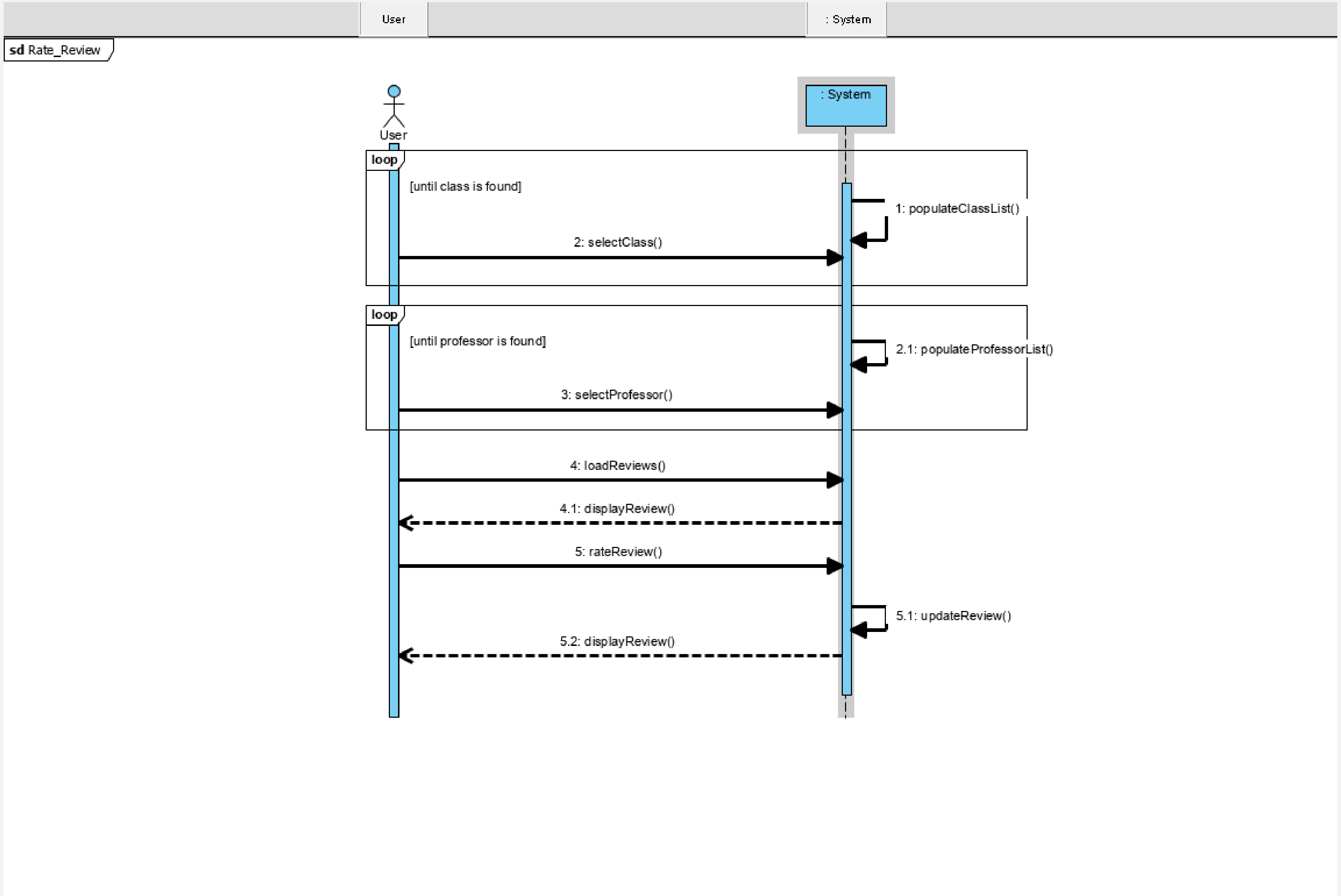
SSD


User
: System
sd Log_Out
User
: System
1: requestToLogOut()
1.1: confirmLogout()
2: logoutConfirm()
alt
[if confirmed logout]
2.1: removeUserAccess()
2.2: logoutSuccessful()

SSD

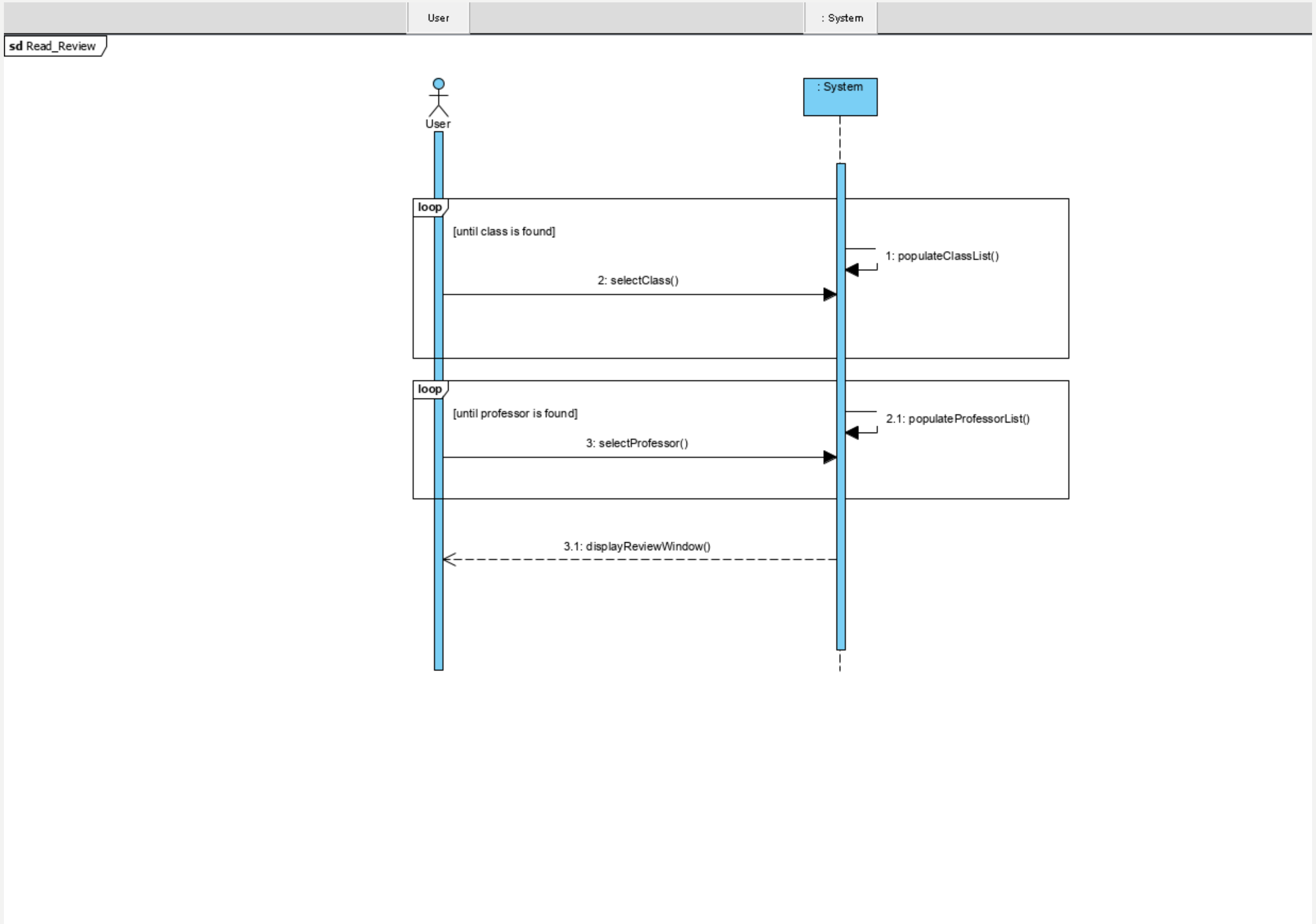

User
: System
: Database
sd Make_Account
User
: System
: Database
loop
[until access granted]
1: setUserInfo(name,pw)
1.1: getUsers()
1.2: usersList
1.3: checkUserName()
1.4: setUserInfo()
1.5: userId
1.6: accessGranted()

SSD

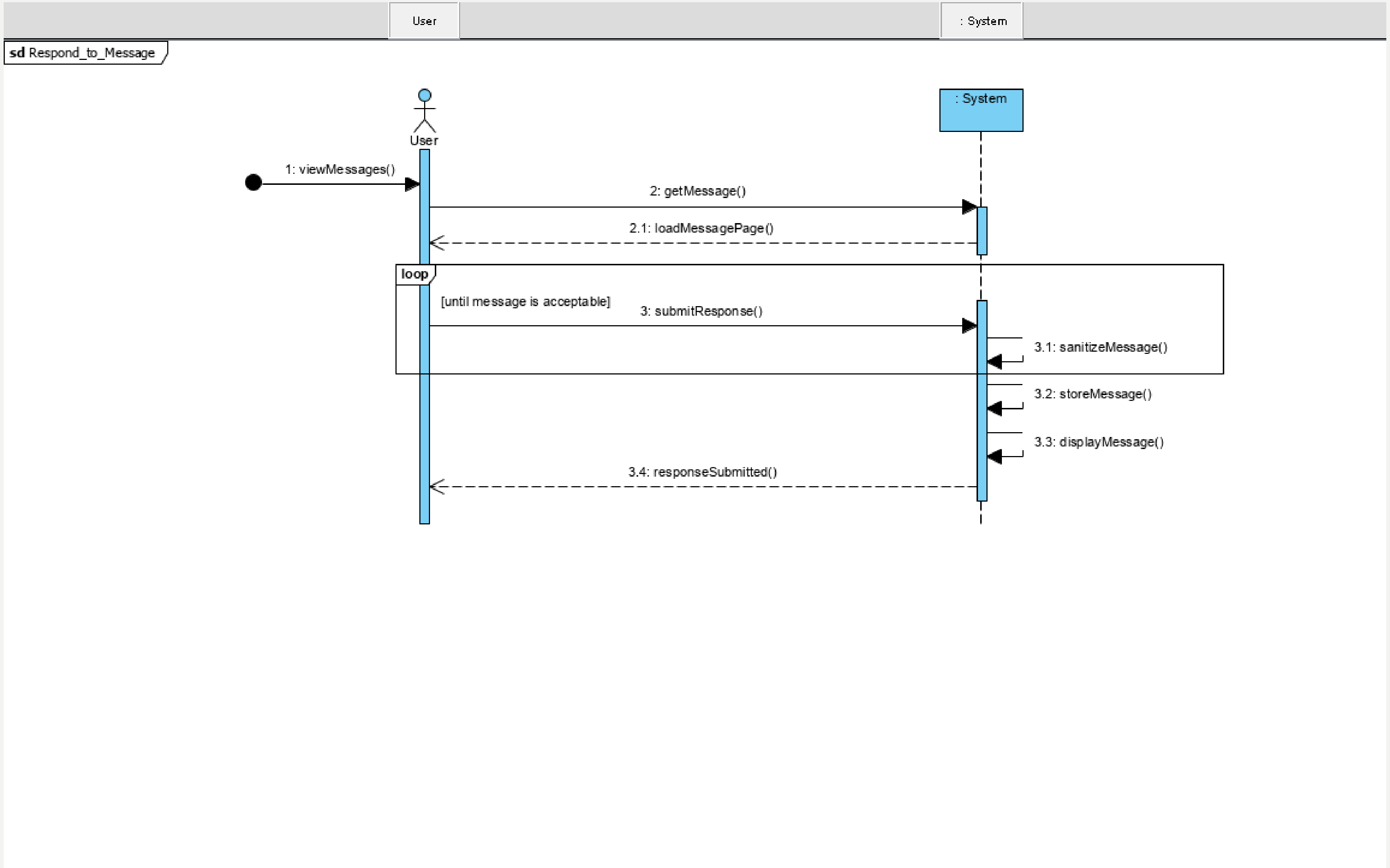


User
: System
sd Rate_Review
User
: System
loop
[until class is found]
1: populateClassList()
2: selectClass()
loop
[until professor is found]
2.1: populateProfessorList()
3: selectProfessor()
4: loadReviews()
4.1: displayReview()
5: rateReview()
5.1: updateReview()
5.2: displayReview()

SSD


User
: System
sd Read_Review
User
: System
loop
[until class is found]
1: populateClassList()
2: selectClass()
loop
[until professor is found]
2.1: populateProfessorList()
3: selectProfessor()
3.1: displayReviewWindow()

SSD


User
: System
sd Respond_to_Message
User
: System
1: viewMessages()
2: getMessage()
2.1: loadMessagePage()
loop
[until message is acceptable]
3: submitResponse()
3.1: sanitizeMessage()
3.2: storeMessage()
3.3: displayMessage()
3.4: responseSubmitted()

OPERATIONS

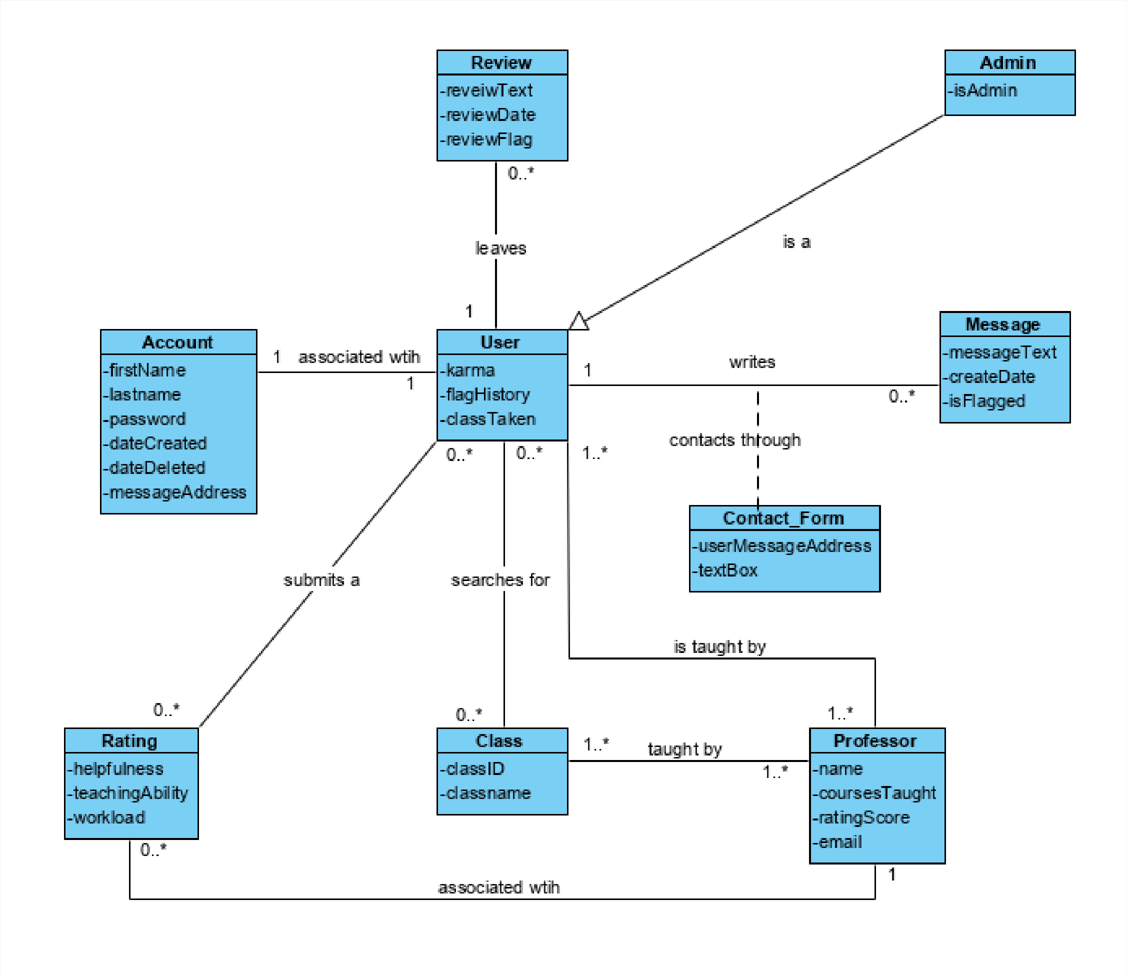
Defining the system operations helps us plan how we will implement the BCC application.

Examples

- checkUsername()
- checkPassword()
- populateClassList()
- populateProfessorList()
- sendMessage()
- viewMessage()

There are ~70 more ☺

DOMAIN MODEL



Review
-reveiwText
-reviewDate
-reviewFlag

Admin
-isAdmin

0..*
leaves
1

is a

Account
-firstName
-lastname
-password
-dateCreated
-dateDeleted
-messageAddress

1 associated wtih 1

User
-karma
-flagHistory
-classTaken

1 writes 0..*

Message
-messageText
-createDate
-isFlagged

contacts through

Contact_Form
-userMessageAddress
-textBox

0..* 0..* 1..*

submits a

searches for

is taught by

0..*

Rating
-helpfulness
-teachingAbility
-workload

0..*

Class
-classID
-classname

1..* taught by 1..*

1..*

Professor
-name
-coursesTaught
-ratingScore
-email

0..* associated wtih 1

QUESTIONS?

TIMECARD

Mark Fuller

- 09/10 - 1 Hour Use Case && Requirements List
- 09/11 - 1 Hour Use Case
- 09/12 - 1 Hour Use Case
- 09/13 - 1 Hour Use Case
- 09/14 - 1 Hour Use Case
- 09/15 - 1 Hour Use Case
- 09/16 - 1 Hour Use Case
- 09/17 - 1 Hour Use Case
- 09/18 - 1 Hour Use Case
- 09/19 - 1 Hour Use Case
- 09/20 - Domain Model

Connor Woodahl

- 09/10 - 1 Requirements List
- 09/11 - 1 Wireframes
- 09/12 - 1 Wireframes
- 09/13 - 1 Wireframes
- 09/14 - 1 Wireframes
- 09/15 - 1 Wireframes
- 09/16 - 1 Wireframes
- 09/17 - 1 Wireframes
- 09/18 - 1 Wireframes
- 09/19 - 1 Use Case Diagram && Traceability Matrix
- 09/20 - 1 Use Case Diagram && Domain Model
- 09/21 - 1 Use Case Diagram

Kevin Kulda

- 09/10 - 1 Requirements List
- 09/11 -
- 09/12 -
- 09/13 -
- 09/14 -
- 09/15 -
- 09/16 -
- 09/17 -
- 09/18 -
- 09/19 - 6 SSD
- 09/20 - 6 Domain Model && System Operations
- 09/21 - Presentation