“Chart My Course” Group Project

Team Members:

- Harm Drenth (team leader)
- Warren Burrus
- Anshpreet Kaur
- Ricardo Boone
- Josh Wilson
- Mia Gortney
- Varun Apte

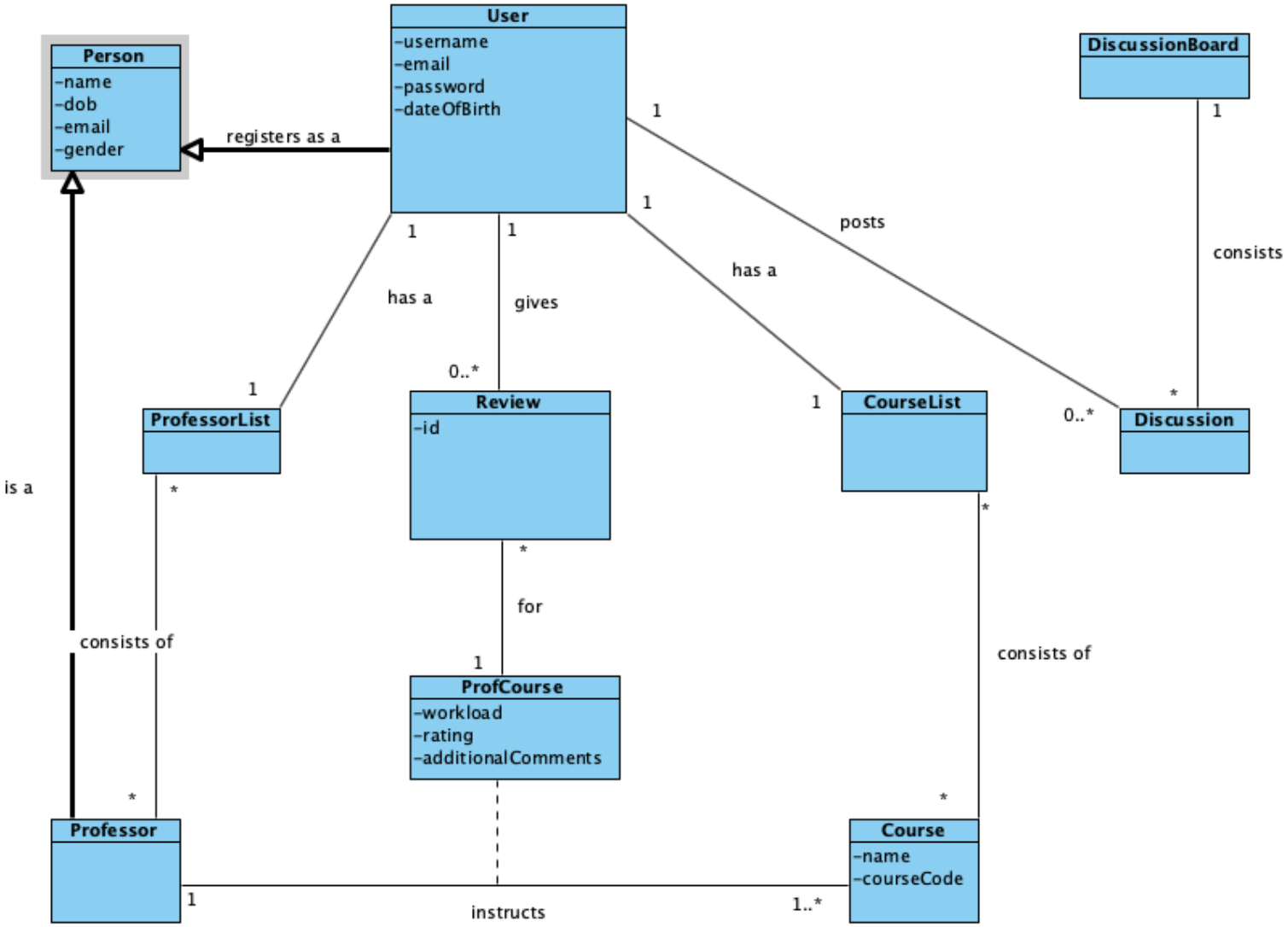
Table of Contents:

1. Project Vision

Navigating college is hard. Navigating the choices of classes is even harder. Chart My Course seeks to simplify a Baylor Computer Science student's choices of courses to take, as well as professors that teach them.

We intend to create a utility similar to the now defunct BUBooks that aggregates reviews of not just the course offerings, but also professor/course pairings. This utility is intended to give potential students a realistic look at what sort of coursework or workload they can expect in a class.

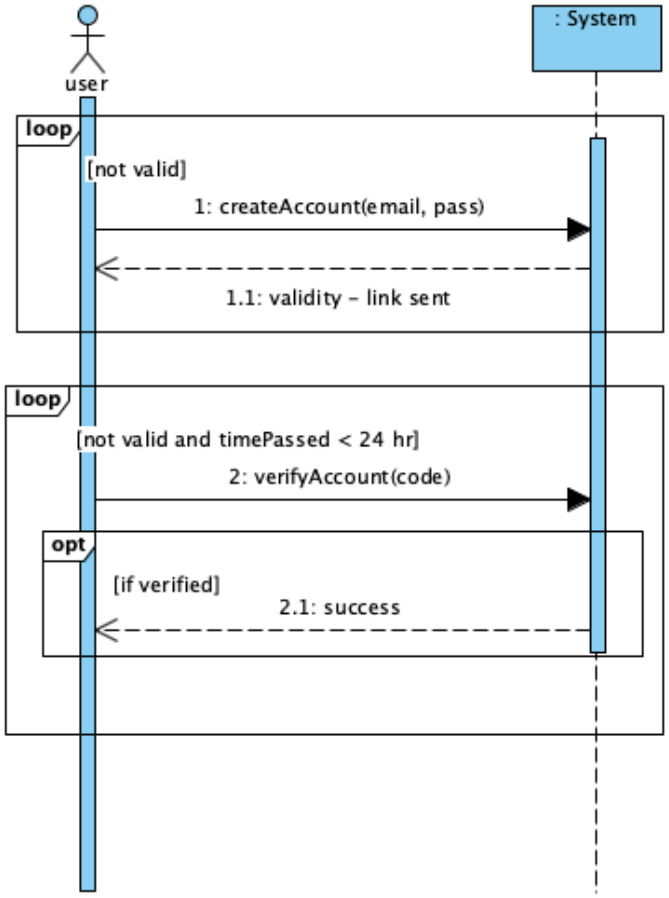
2. Domain Model



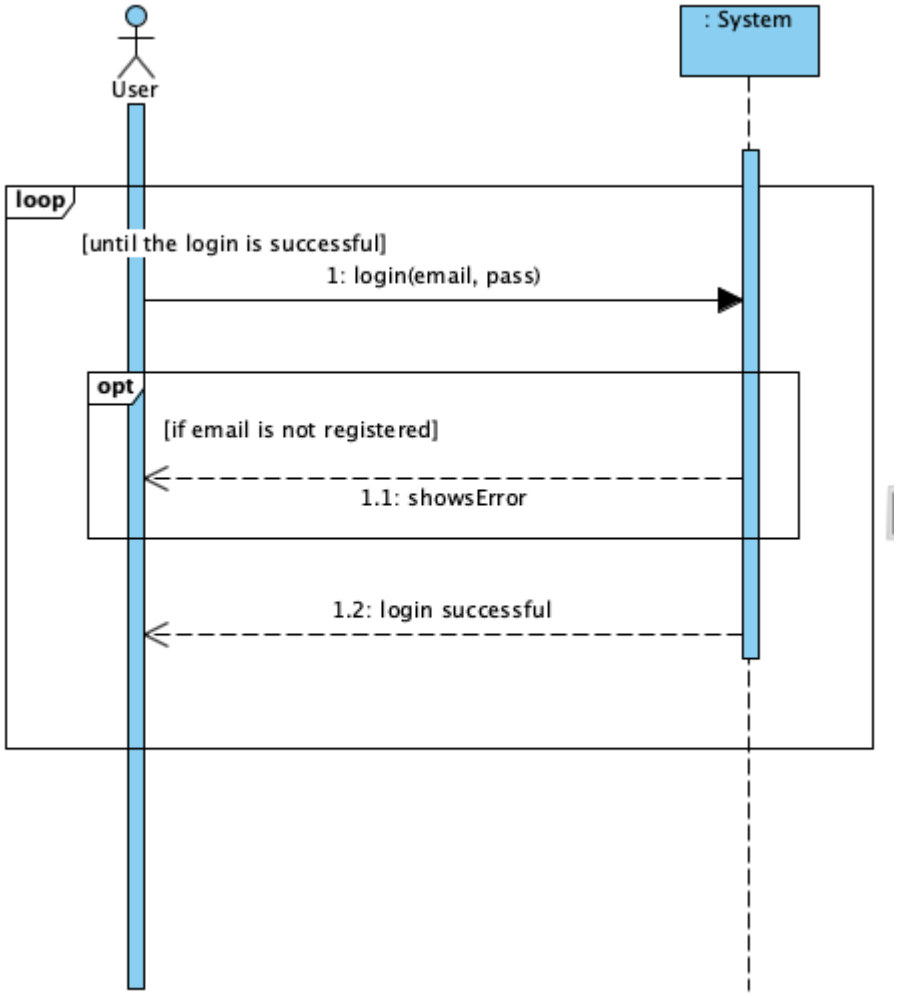
3. Requirements
    a. Functional
        - The user can create a new account
        - The user can log in and out of their account
        - The user can find recommended courses and professors
        - The user can add a review of a professor
        - The user can search reviews of professors/courses
        - The user can filter search results
        - The user can request a review to be removed
        - The user can have a list of saved professors
        - The user can have a list of saved courses
        - The system saves or deletes changes upon logout
        - The user can post questions and answers in the discussion board
        - The user can reply to discussions
        - The user can upvote questions in the discussion board
    b. Non-functional
        - The system shall not share the personal information of the author of posts to other users, only information the operators of the system receive is the author's name
        - All posts and user activity will be user friendly and comply to online community and safety guidelines
        -
        - The courses and professors in the system will be limited to Baylor's Computer Science department
        - The system will adhere to all Baylor guidelines and policies
        - The system will return search results within 30 seconds

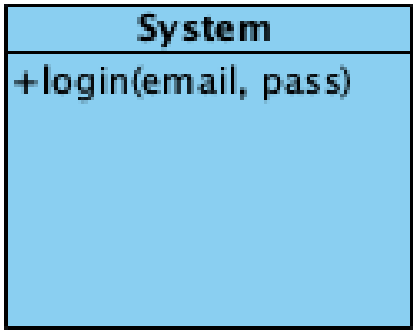
4. Use Cases

ID: UC Create Account
Scope: Course/Professor Planning
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with degree plan info and professor reviews info
Preconditions: The user is not logged in
Flow of events: 1. A new user wants to create an account 2. User clicks on Create Account 3. The system asks for Baylor email 4. The user enters their Baylor email 5. The system sends a verification email 6. The user clicks the link and is brought back to the screen 7. The system asks to enter the password and other information 8. The user enters the information and click submit
Extensions: 4a. The user does not enter Baylor Email 1. System will display "Please enter Baylor Email" 2. System will return to the start of step three 7a. The entered password does not meet the requirements 1. System will display "Please enter a new password"
Postconditions/Success: Account is created successfully



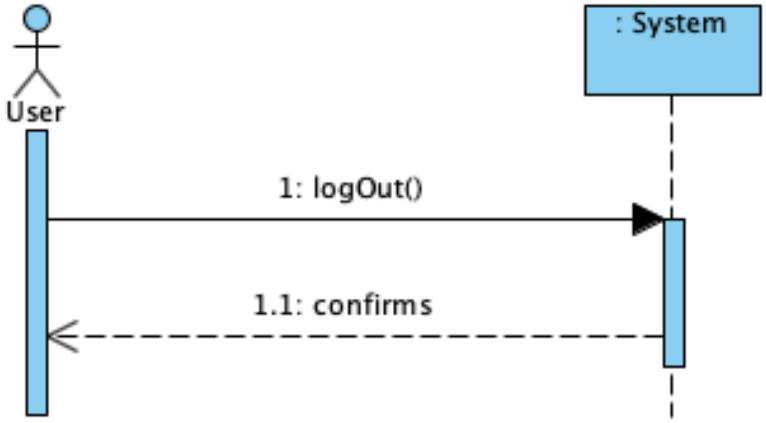
ID: UC Login
Scope: Course/Professor Planning
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with correct login information
Preconditions: The user has an account created
Flow of events: 1. The user clicks on login 2. The system asks for Baylor email and password 3. If the login credentials exist, the user is directed to the home page
Extensions: 2a. The user enters an email which does not exist 1. System will display "Please enter Baylor Email" 2. System will return to the start of step two 2b. The entered password does not match 1. System will display "Please re-enter" 2. If the user has had 3 tries, the system redirects to reset password
Postconditions/Success: User is logged in

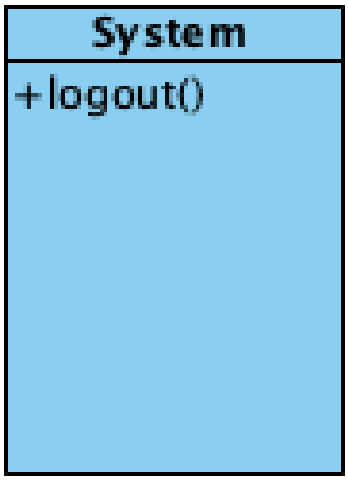

User
: System
loop
[until the login is successful]
1: login(email, pass)
opt
[if email is not registered]
1.1: showsError
1.2: login successful


System
+login(email, pass)

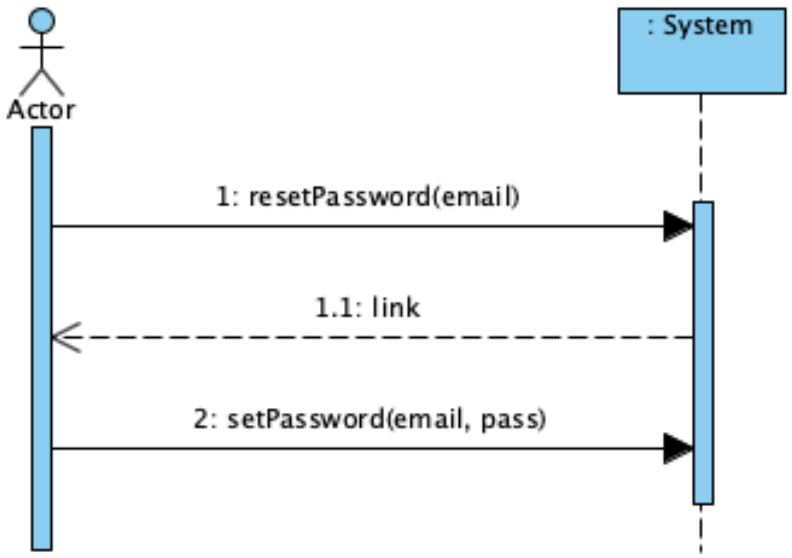
ID: UC Logout
Scope: Course/Professor Planning
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with correct login information
Preconditions: The user is logged in
Flow of events: 1. The user clicks on logout 2. The system confirms from the user about logging out and tells the user to save any changes 3. The user confirms 4. The system directs to the home page
Extensions: a* Anytime the system does not respond 1. User will restart the application
Postconditions/Success: User is logged out

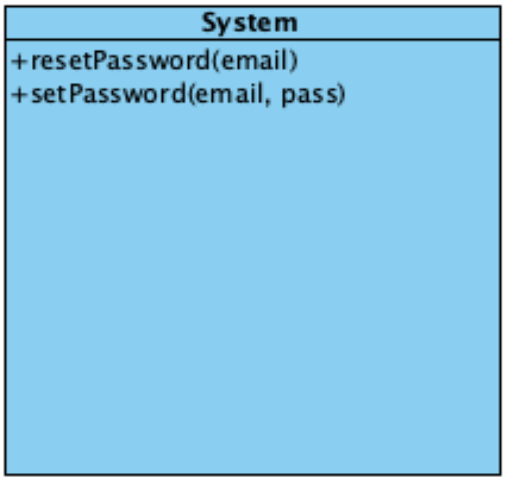
ID: UC Login
Scope: Course/Professor Planning
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with correct login information
Preconditions: The user has an account created
Flow of events: 1. The user clicks on login 2. The system asks for Baylor email and password 3. If the login credentials exist, the user is directed to the home page
Extensions: 2a. The user enters an email which does not exist 1. System will display "Please enter Baylor Email" 2. System will return to the start of step two 2b. The entered password does not match 1. System will display "Please re-enter" 2. If the user has had 3 tries, the system redirects to reset password
Postconditions/Success: User is logged in


User
: System
1: logOut()
1.1: confirms


System
+logout()

ID: UC Reset Password
Scope: Course/Professor Planning
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with correct login information
Preconditions: The user has an account created and is not logged in
Flow of events: 1. The user clicks on "Reset Password" 2. The system asks for Baylor email 3. The system link to their email 4. The user clicks the link and is brought back to reset password page 5. The system asks the user to enter the email and new password 6. The user enters and clicks on submit 7. The system updates the information in the database
Extensions: 2a. The user enters an email which does not exist 1. System will display "Please enter Baylor Email" 2. System will return to the start of step two 4a. The user does not respond to the link in 24 hours 1. The link expires. 5a. The password does not meet the requirements 1. The system asks the user to enter a new password
Postconditions/Success: The password is reset.

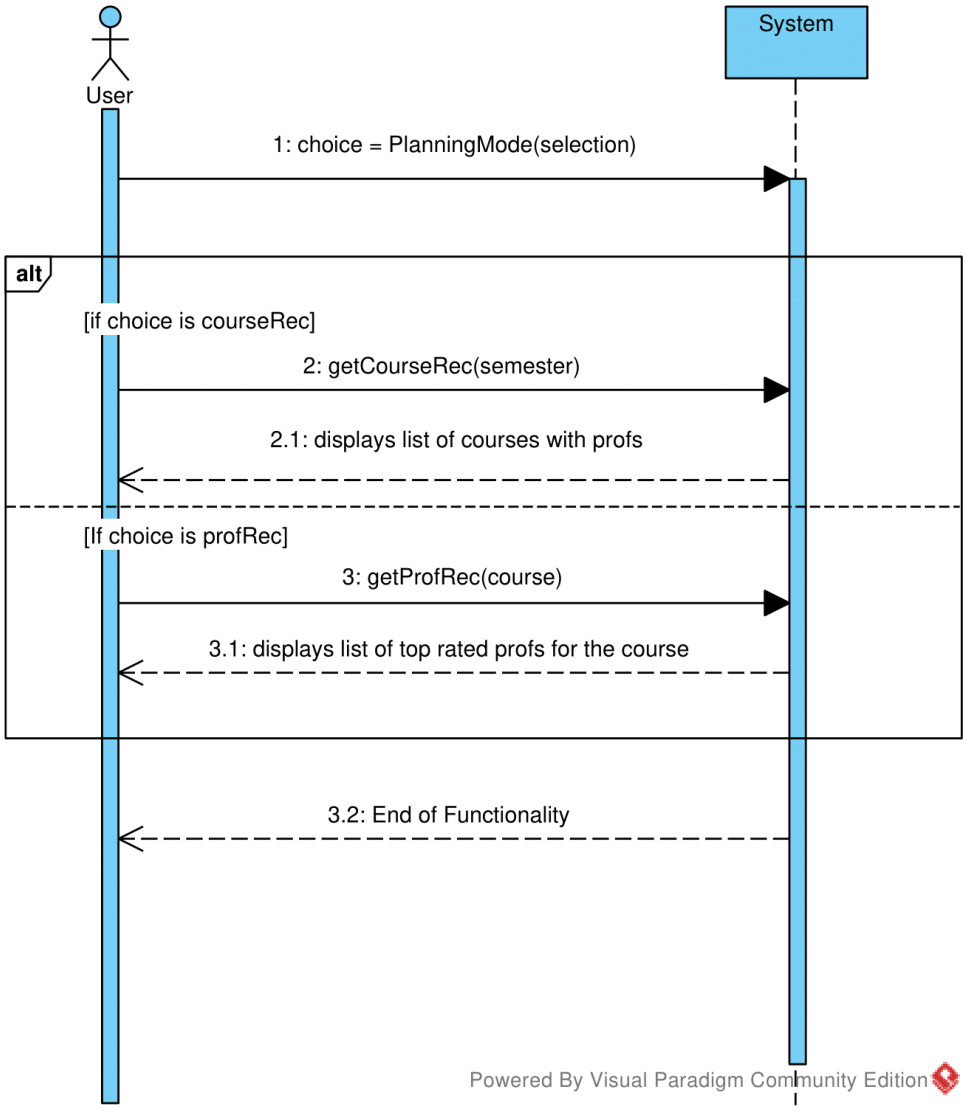


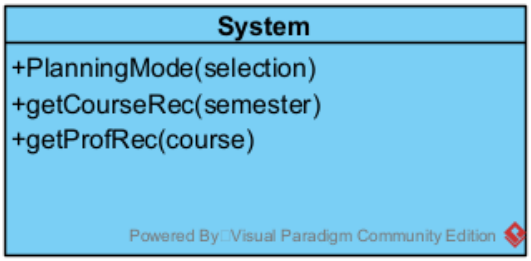


ID: UC Recommendations
Scope: Course/Professor Planning
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with degree plan info and professor reviews info
Preconditions: The user is on the planning page of the site

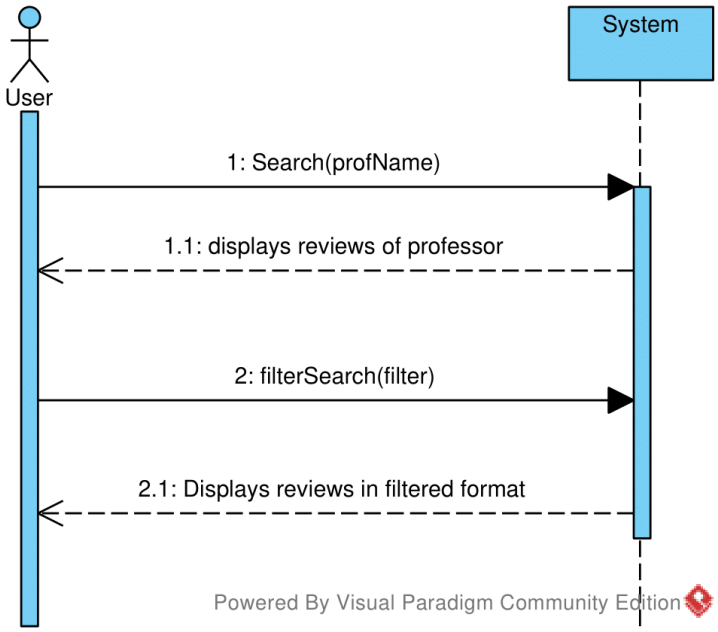
Flow of events:
1. User selects type of recommendation
2. System records the type of rec selected
3. If user choses course rec
    1. User enters the semester they are in
    2. Displays list of courses with professors
4. If user chooses prof rec
    1. User enters the course they are in
    2. Displays list of top rated profs for the selected course

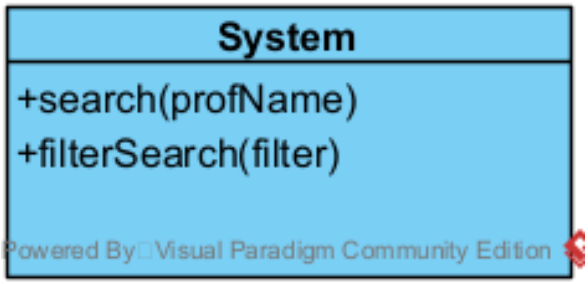
Postconditions/Success: Site displays recommended courses along with the associated professors


System
User
1: choice = PlanningMode(selection)
alt
[if choice is courseRec]
2: getCourseRec(semester)
2.1: displays list of courses with profs
[If choice is profRec]
3: getProfRec(course)
3.1: displays list of top rated profs for the course
3.2: End of Functionality
Powered By Visual Paradigm Community Edition

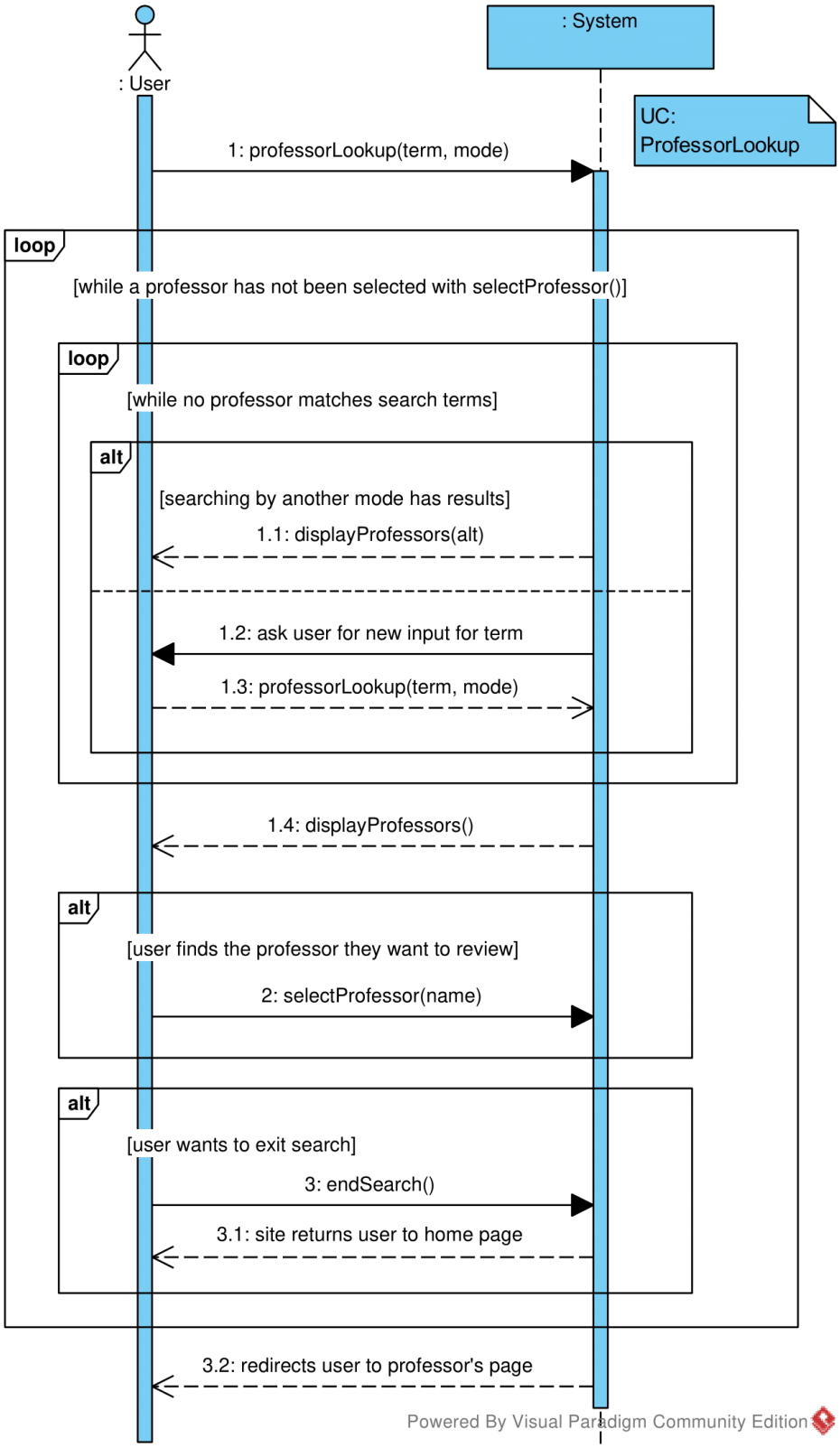

System
+PlanningMode(selection)
+getCourseRec(semester)
+getProfRec(course)
Powered By Visual Paradigm Community Edition

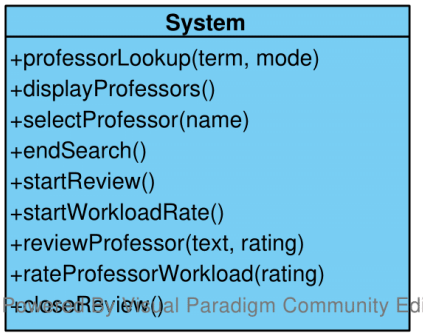
ID: UC Filter Results
Scope: Professor Reviews
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with reviews
Preconditions: The user is on the search professor page of the site
Flow of events: 1. User selects a professor's name from dropdown menu to search for reviews 2. Site displays the reviews of the given professor 3. If user wants to filter results the user can select a filter from the dropdown menu 4. The system filters the results, and the site displays filtered info
Postconditions/Success: Site displays the reviews with the given input from the user



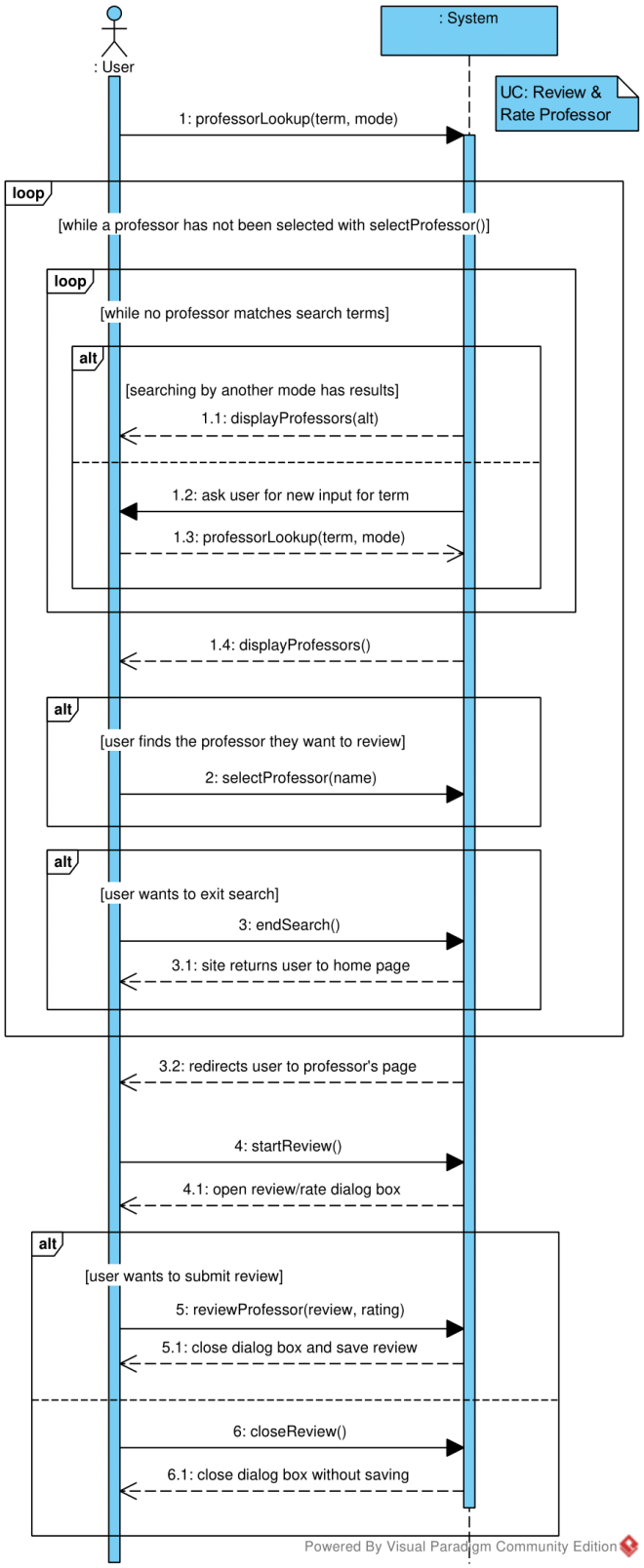


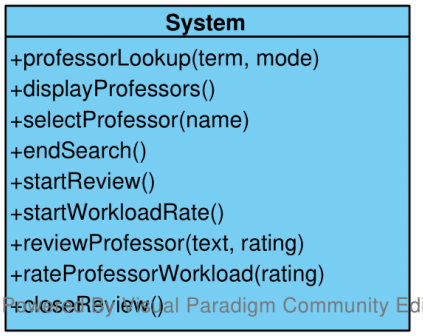
ID: UC Professor Lookup
Scope: Professor Lookup & Rating System
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with professor info
Preconditions: User must be logged into the website
Flow of events: 1. User opens the site main page 2. User enters search terms into the search box, and chooses a search mode (by professor name or by class) 3. Site redirects user to a list of all professors matching search terms 4. User selects the desired professor 5. Site redirects user to selected professor's page
Extensions: 3a. User exits search without choosing a professor 1. Site redirects user back to main page 3b. If no professors match search terms used 1. Site switches search mode and displays any matching results 2. If there are no results, site asks user for new search terms 3. Site continues this until a search provides valid results
Postconditions/Success: User is directed to professor's page by lookup system Site displays professor's reviews and ratings to user


: User
: System
UC: ProfessorLookup
1: professorLookup(term, mode)
loop
[while a professor has not been selected with selectProfessor()]
loop
[while no professor matches search terms]
alt
[searching by another mode has results]
1.1: displayProfessors(alt)
1.2: ask user for new input for term
1.3: professorLookup(term, mode)
1.4: displayProfessors()
alt
[user finds the professor they want to review]
2: selectProfessor(name)
alt
[user wants to exit search]
3: endSearch()
3.1: site returns user to home page
3.2: redirects user to professor's page
Powered By Visual Paradigm Community Edition

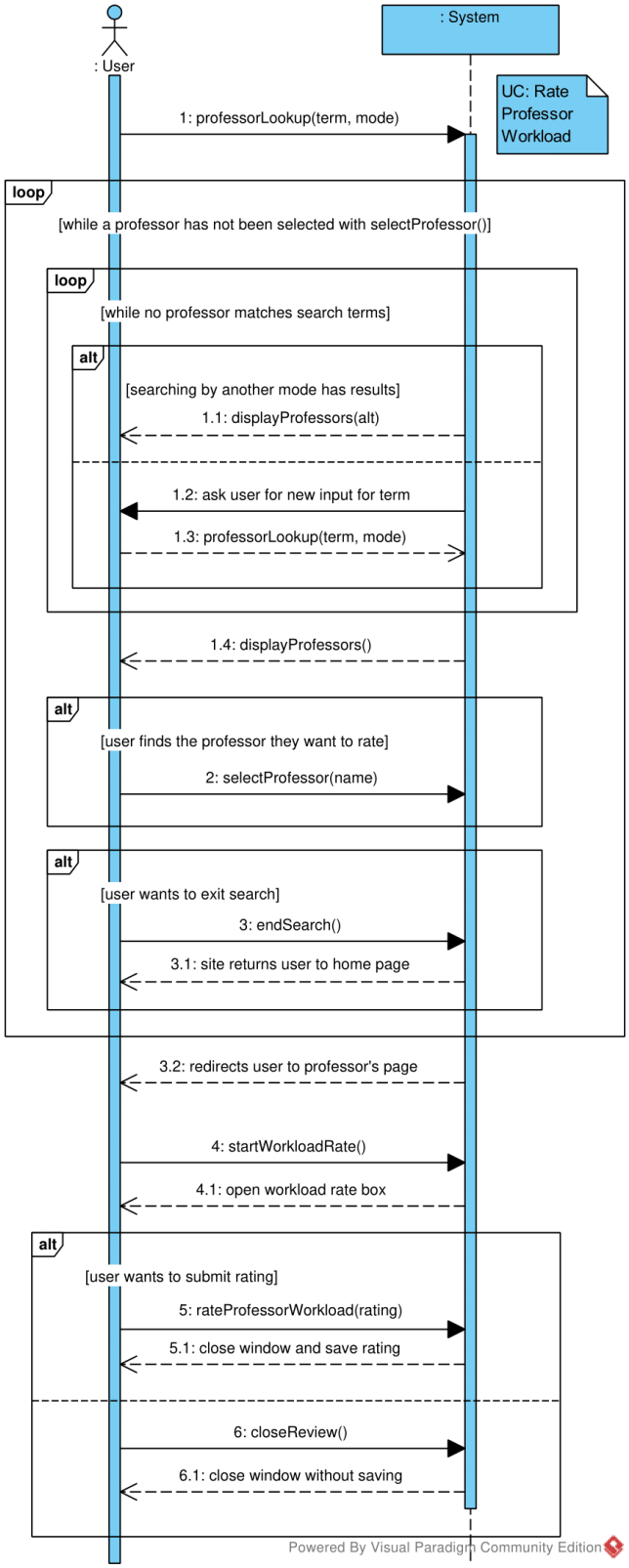


ID: UC Review & Rate Professor
Scope: Professor Lookup & Rating System
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with professor info
Preconditions: User must be logged into the website
Flow of events: 1. User uses professor lookup system to navigate to desired professor's page 2. Site displays professor's page 3. User selects "Rate and Review Professor" 4. Site opens popup with text box and rating selection buttons 5. User types review into text box 6. User selects review rating with rating selection buttons 7. Site saves review and returns user to the professor's page
Extensions: 5a. User wants to exit review without posting 1. User clicks "X" button in review panel 2. Site returns user to professor's page and ends review/rating
Postconditions/Success: User's review and rating are stored in the site's database User's review and rating are now visible on the professor's page on the site


: User
: System
UC: Review & Rate Professor
1: professorLookup(term, mode)
loop
[while a professor has not been selected with selectProfessor()]
loop
[while no professor matches search terms]
alt
[searching by another mode has results]
1.1: displayProfessors(alt)
1.2: ask user for new input for term
1.3: professorLookup(term, mode)
1.4: displayProfessors()
alt
[user finds the professor they want to review]
2: selectProfessor(name)
alt
[user wants to exit search]
3: endSearch()
3.1: site returns user to home page
3.2: redirects user to professor's page
4: startReview()
4.1: open review/rate dialog box
alt
[user wants to submit review]
5: reviewProfessor(review, rating)
5.1: close dialog box and save review
6: closeReview()
6.1: close dialog box without saving
Powered By Visual Paradigm Community Edition



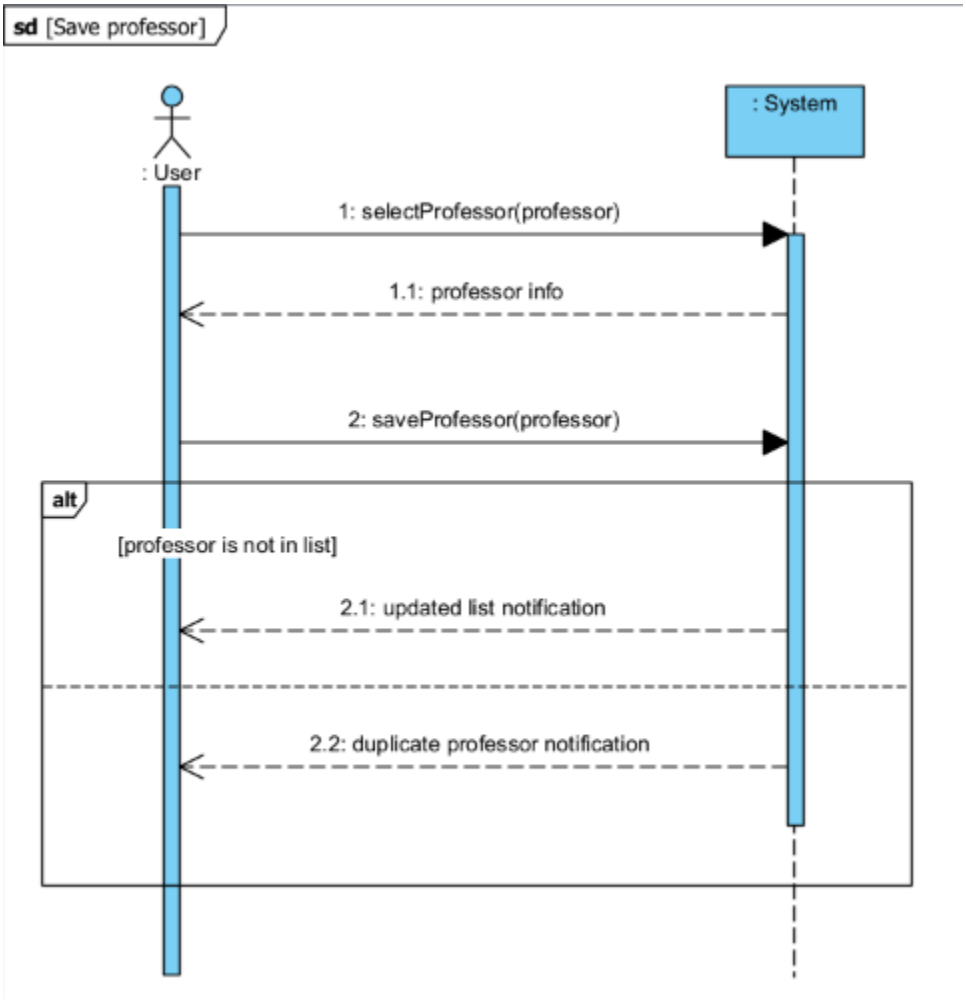
ID: UC Rate Professor Workload
Scope: Professor Lookup & Rating System
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with professor info
Preconditions: User must be logged into the website
Flow of events: 1. User uses professor lookup system to navigate to desired professor's page 2. Site displays professor's page 3. User selects "Rate Professor Workload" 4. Site opens popup with rating selection buttons 5. User selects rating with rating selection buttons 6. Site saves rating and returns user to the professor's page
Extensions: 5a. User wants to exit rating without posting 1. User clicks "X" button in review panel 2. Site returns user to professor's page and ends rating
Postconditions/Success: User's workload rating is stored in the site's database User's workload rating is now visible on the professor's page on the site

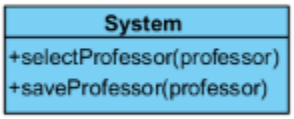

: User
: System
UC: Rate Professor Workload
1: professorLookup(term, mode)
loop
[while a professor has not been selected with selectProfessor()]
loop
[while no professor matches search terms]
alt
[searching by another mode has results]
1.1: displayProfessors(alt)
1.2: ask user for new input for term
1.3: professorLookup(term, mode)
1.4: displayProfessors()
alt
[user finds the professor they want to rate]
2: selectProfessor(name)
alt
[user wants to exit search]
3: endSearch()
3.1: site returns user to home page
3.2: redirects user to professor's page
4: startWorkloadRate()
4.1: open workload rate box
alt
[user wants to submit rating]
5: rateProfessorWorkload(rating)
5.1: close window and save rating
6: closeReview()
6.1: close window without saving
Powered By Visual Paradigm Community Edition

System
+professorLookup(term, mode) +displayProfessors() +selectProfessor(name) +endSearch() +startReview() +startWorkloadRate() +reviewProfessor(text, rating) +rateProfessorWorkload(rating) +closeReview()

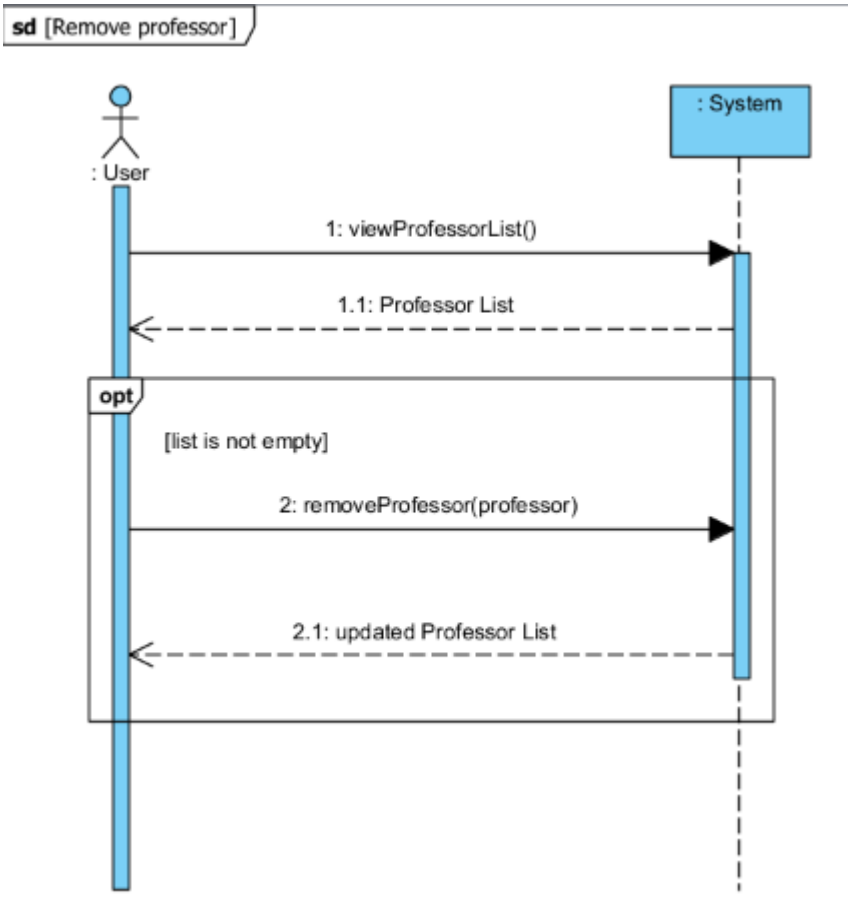
Produced by Visual Paradigm Community Edition

ID: UC Save Professor
Scope: Professor Lookup and List of saved professors
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User is logged into the website, and has successfully looked up a list of professors
Flow of events: 1. The user selects a professor from the search results 2. The system displays a page with that professor's info 3. The user selects the "Save Professor" icon on the professor's window 4. If the professor is not currently stored in the user's "Saved Professors" list, then a. The system displays a notification that the professor has been added to the end of the user's list 5. Else, a. The system displays a notification that the professor has already been saved to the list
Postconditions/Success: The user's "Saved Professors" list has one more professor at the end of the list, if it is not already in the list


sd [Save professor]
: User
: System
1: selectProfessor(professor)
1.1: professor info
2: saveProfessor(professor)
alt
[professor is not in list]
2.1: updated list notification
2.2: duplicate professor notification


System
+selectProfessor(professor)
+saveProfessor(professor)

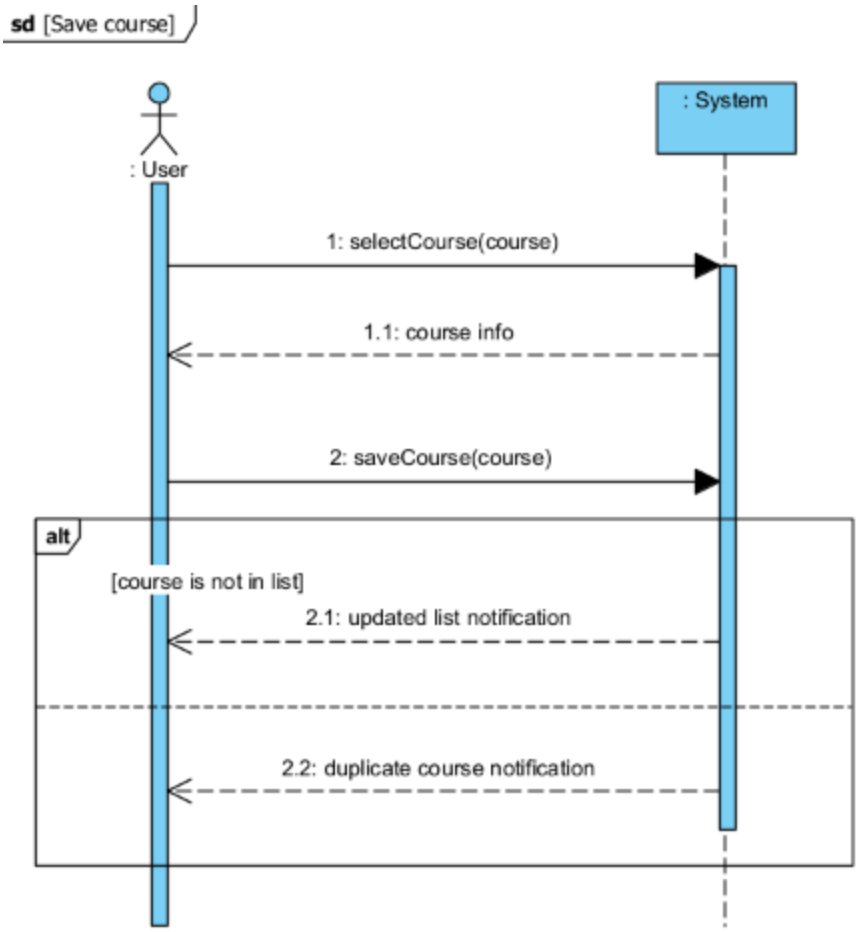
ID: UC Remove Professor
Scope: List of saved professors and Professor Lookup
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User must be logged into the website
Flow of events: 1. The user opens up their "Saved Professors" list 2. The system displays the user's list 3. If their list is not empty, then a. The user selects the "Remove Professor" icon next to a professor in their list b. The system removes that professor from the user's list and displays the updated list to the user
Postconditions/Success: The user's "Saved Professors" list has one less professor, or remains empty if originally empty

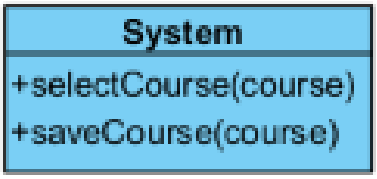


System
+viewProfessorList() +removeProfessor(professor)

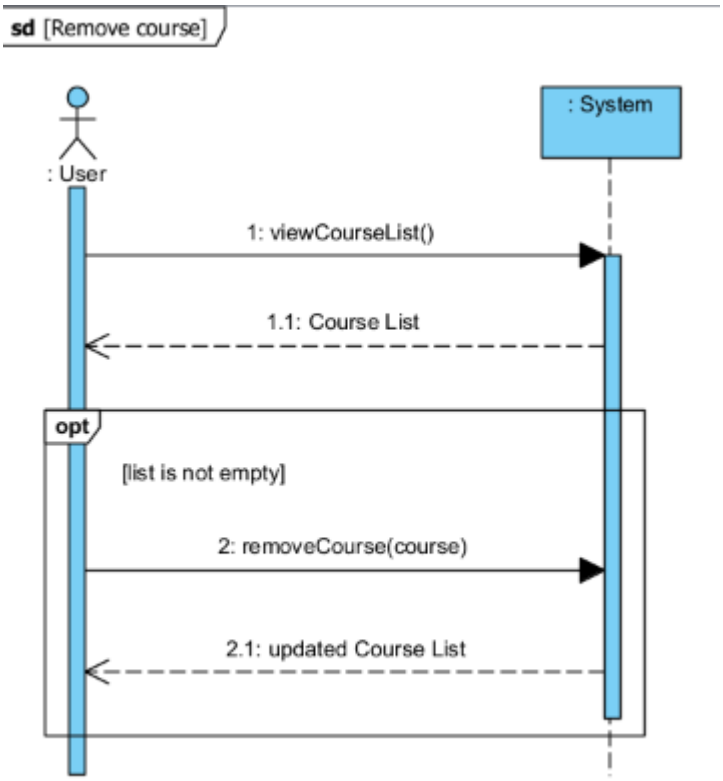
ID: UC Save Changes Exit Message
Scope: User Account Changes
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User must be logged into the website
Flow of events: 1. User clicks the "Save Changes?" button 2. System checks if any changes made to "Saved Professors" list a. If yes, overwrites the previous "Saved Professors" list and saves for user 3. System checks if any changes made to "Saved Courses" list a. If yes, overwrites the previous "Saved Courses" list and saves for user
Postconditions/Success: User's changes to "Saved Professors" and "Saved Courses" lists are saved by the system System displays a "Save Successful" message

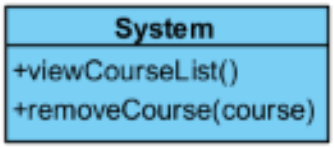
ID: UC Save Course
Scope: Course Lookup and List of saved courses
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User is logged into the website, and has successfully looked up a list of courses
Flow of events: 1. The user selects a course from the search results 2. The system displays a page with that course's info 3. The user selects the "Save Course" icon on the course's window 4. If the course is not currently in the user's "Saved Courses" list, then a. The system displays a notification that the professor has been added to the end of the user's list 5. Else, a. The system displays a notification that the course has already been saved to the list
Postconditions/Success: The user's "Saved Courses" list has one more course at the end of the list, if it is not already in the list


sd [Save course]
: User
: System
1: selectCourse(course)
1.1: course info
2: saveCourse(course)
alt
[course is not in list]
2.1: updated list notification
2.2: duplicate course notification

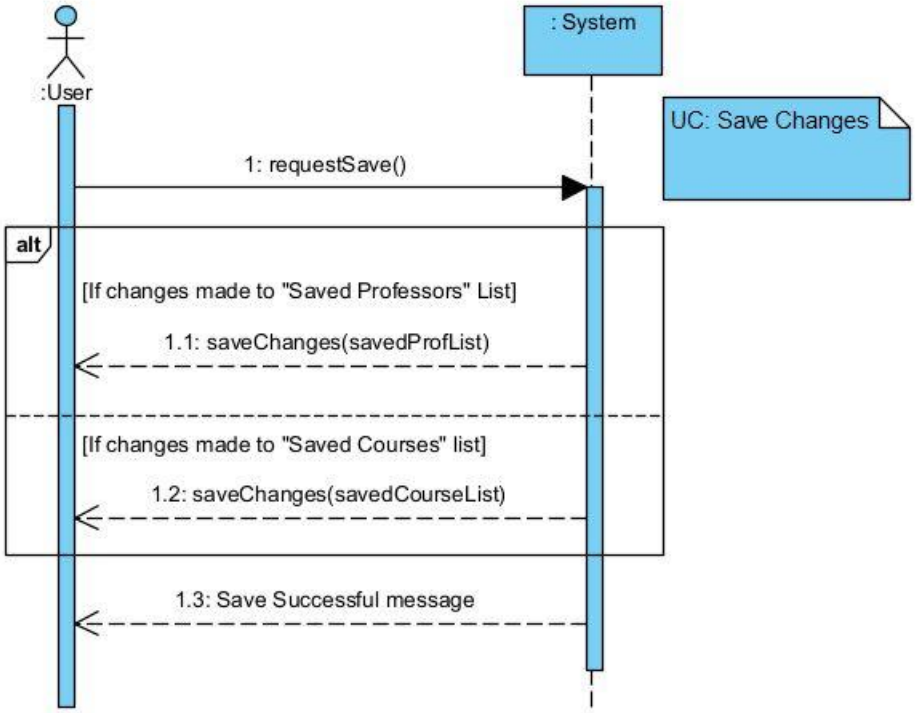

System
+selectCourse(course)
+saveCourse(course)

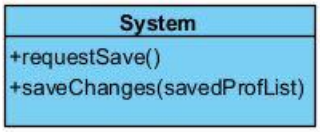
ID: UC Remove Course
Scope: List of saved courses and Course Lookup
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User must be logged into the website
Flow of events: 1. The user opens up their "Saved Courses" list 2. The system displays the user's list 3. If their list is not empty, then a. The user selects the "Remove Course" icon next to a course in their list b. The system removes that course from the user's list and displays the updated list to the user
Postconditions/Success: The user's "Saved Courses" list has one less course, or remains empty if originally empty



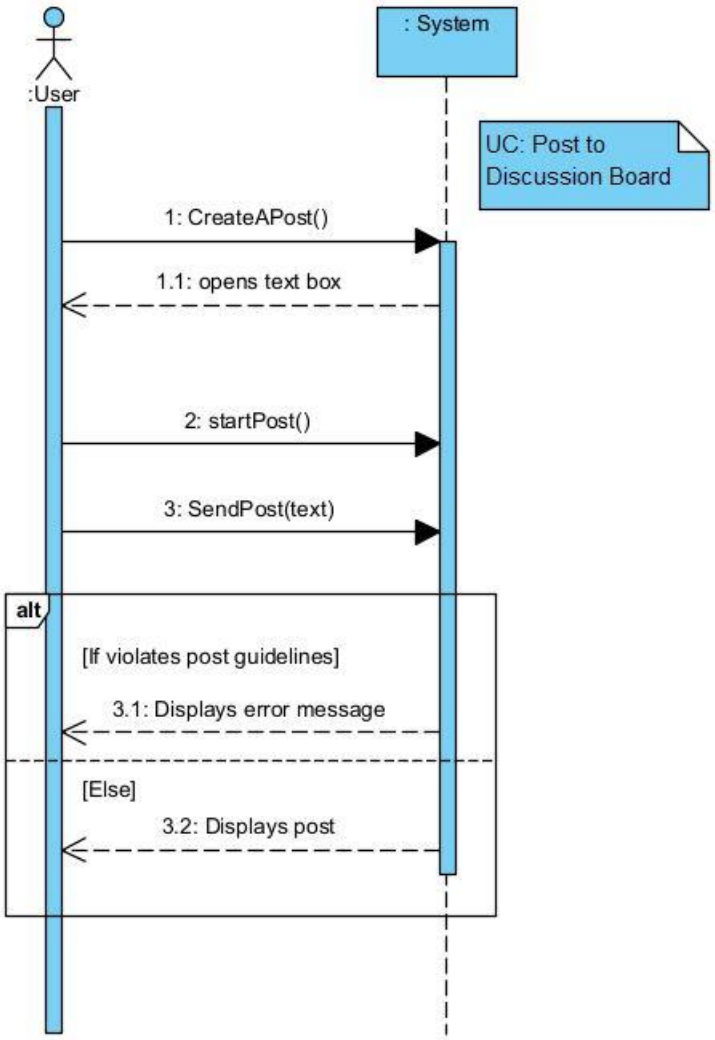


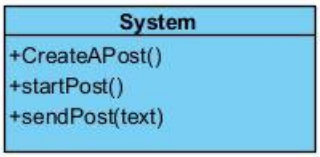
ID: UC Save Changes Exit Message
Scope: User Account Changes
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User must be logged into the website
Flow of events: 1. User clicks the "Save Changes?" button 2. System checks if any changes made to "Saved Professors" list a. If yes, overwrites the previous "Saved Professors" list and saves for user 3. System checks if any changes made to "Saved Courses" list a. If yes, overwrites the previous "Saved Courses" list and saves for user
Postconditions/Success: User's changes to "Saved Professors" and "Saved Courses" lists are saved by the system System displays a "Save Successful" message



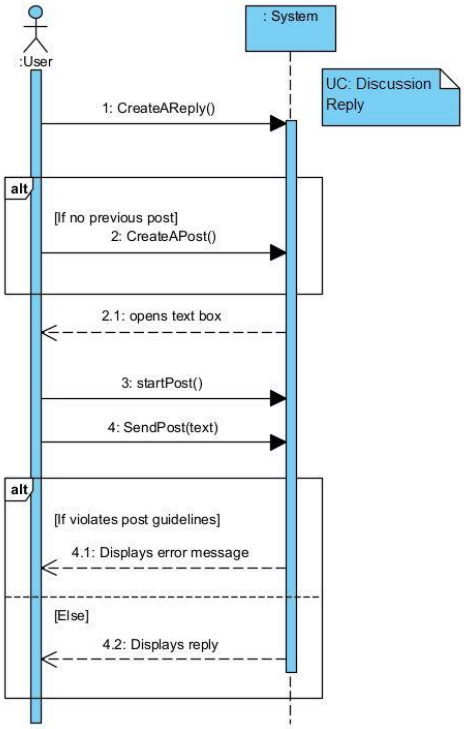


ID: UC Post to a Discussion Board
Scope: Discussion Board
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User must be logged into the website User is on the Discussion Board page
Flow of events: **1.** User clicks the "Create a Post" button **2.** System opens a text box **3.** User inputs text in the text box **4.** User clicks the "Send Post" button a. If a post does not follow post guidelines, system will display an error message **5.** System displays the post
Postconditions/Success: System displays the post for all users to see



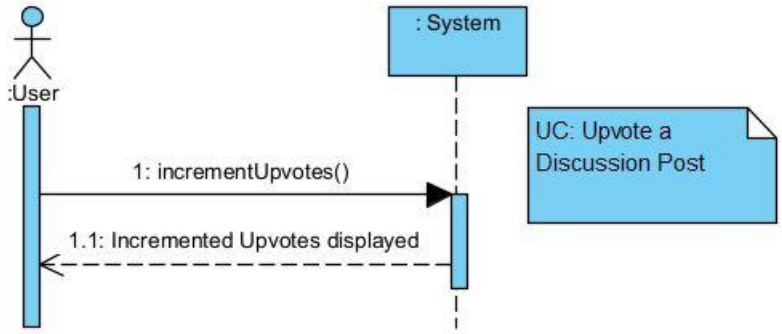


ID: UC Discussion Reply
Scope: Discussion Board
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User must be logged into the website
Flow of events: **1.** User clicks the "Reply" button below a post a. If a previous post does not exist, only button to click will be "Create New Post" **2.** System opens a text box **3.** User inputs text in the text box **4.** User clicks the "Send Reply" button a. If a reply does not follow post guidelines, system will display an error message **5.** System displays the reply below the original post
Postconditions/Success: System displays the reply under the original post for all users to see



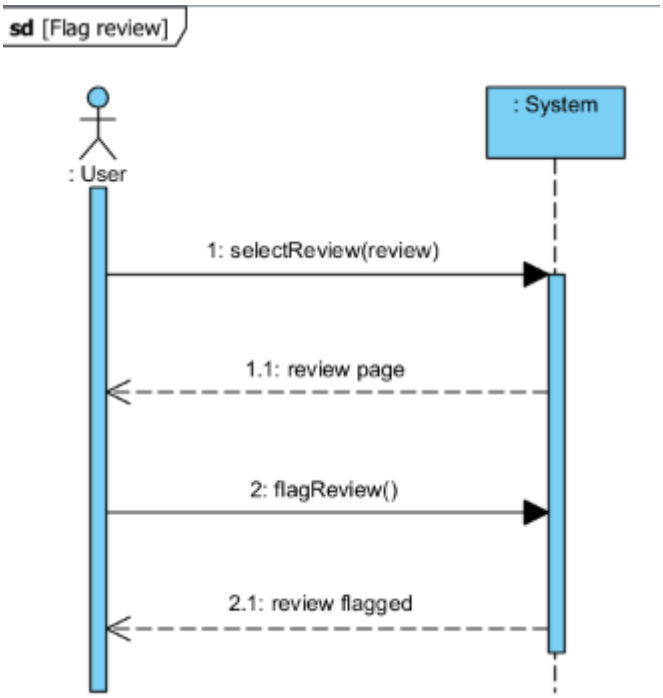
System
+CreateAReply() +CreateAPost() +startPost() +SendPost(text)

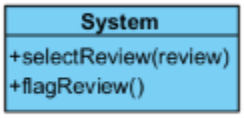
ID: UC Upvote a Discussion
Scope: Discussion Board
Level: User Goal
Actors: User - Student interacting with system
Preconditions: User must be logged into the website A post must exist
Flow of events: 1. User clicks the "Upvote" button under a discussion post 2. System increments the amount of Upvotes on the post
Postconditions/Success: System displays the amount of Upvotes under a discussion post



System
+incrementUpvotes()

ID: UC Flag Review
Scope: Professor reviews
Level: User Goal
Actors: User - Student interacting with system Database Service - Site database with reviews
Preconditions: The user has searched for professor reviews from the search professor page of the site
Flow of events: 1. The user selects a review from the professor review search results 2. The system displays a page with the selected review 3. The user selects the "Flag Review" icon on the review page 4. The system updates the flag icon to reflect that the review has been reported to admins
Postconditions/Success: The review is marked as flagged and is reported to the admins

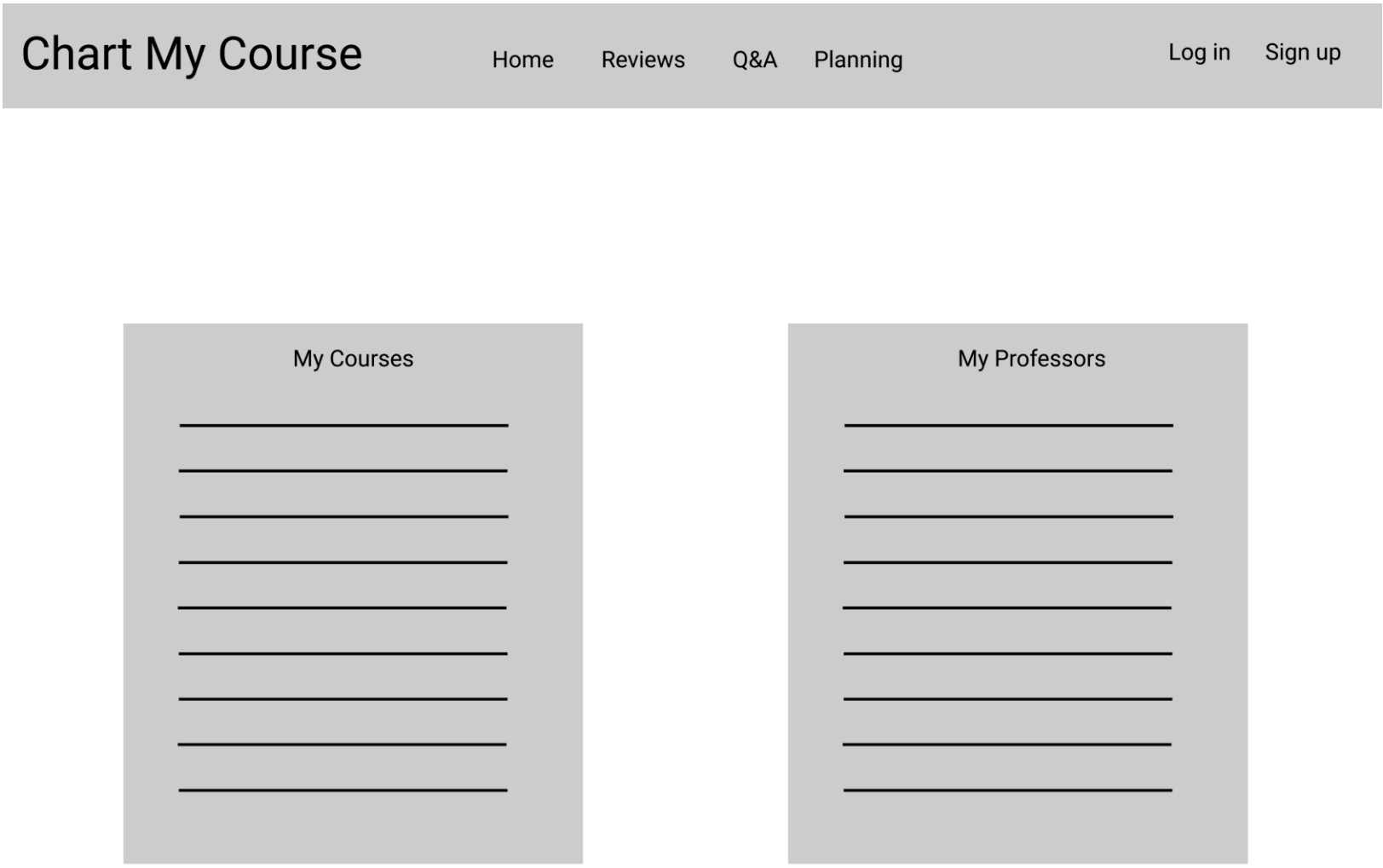




5. Traceability Matrix

Traceability Matrix	The user can create a new account	The user can log in an out of their account	The user can find recommended courses and professors	The user can add a review of a professor	The user can search reviews of professors/courses	The user can filter search results	The user can request a review to be removed	The user can have a list of saved professors	The user can have a list of saved courses	The system saves or deletes changes upon logout	The user can post questions and answers in the discussion board	The user can reply to discussions	The user can upvote questions in the discussion board
UC Create Account	x												
UC Log-in		x											
UC Log out		x											
UC Reset Password		x											
UC Reccommendations			x										
UC Filter Results						x							
UC Professor Lookup					x		x						
UC Review Professor				x									
UC Rate Professor Workload				x									
UC Add Professor								x		x			
UC Remove Professor								x		x			
UC Add Course									x	x			
UC Remove Course									x	x			
UC Save Changes								x	x				
UC Post Discussion											x		
UC Reply Discussion												x	
UC Upvote Discussion													x
UC Flag Review							x						

6. Wireframes



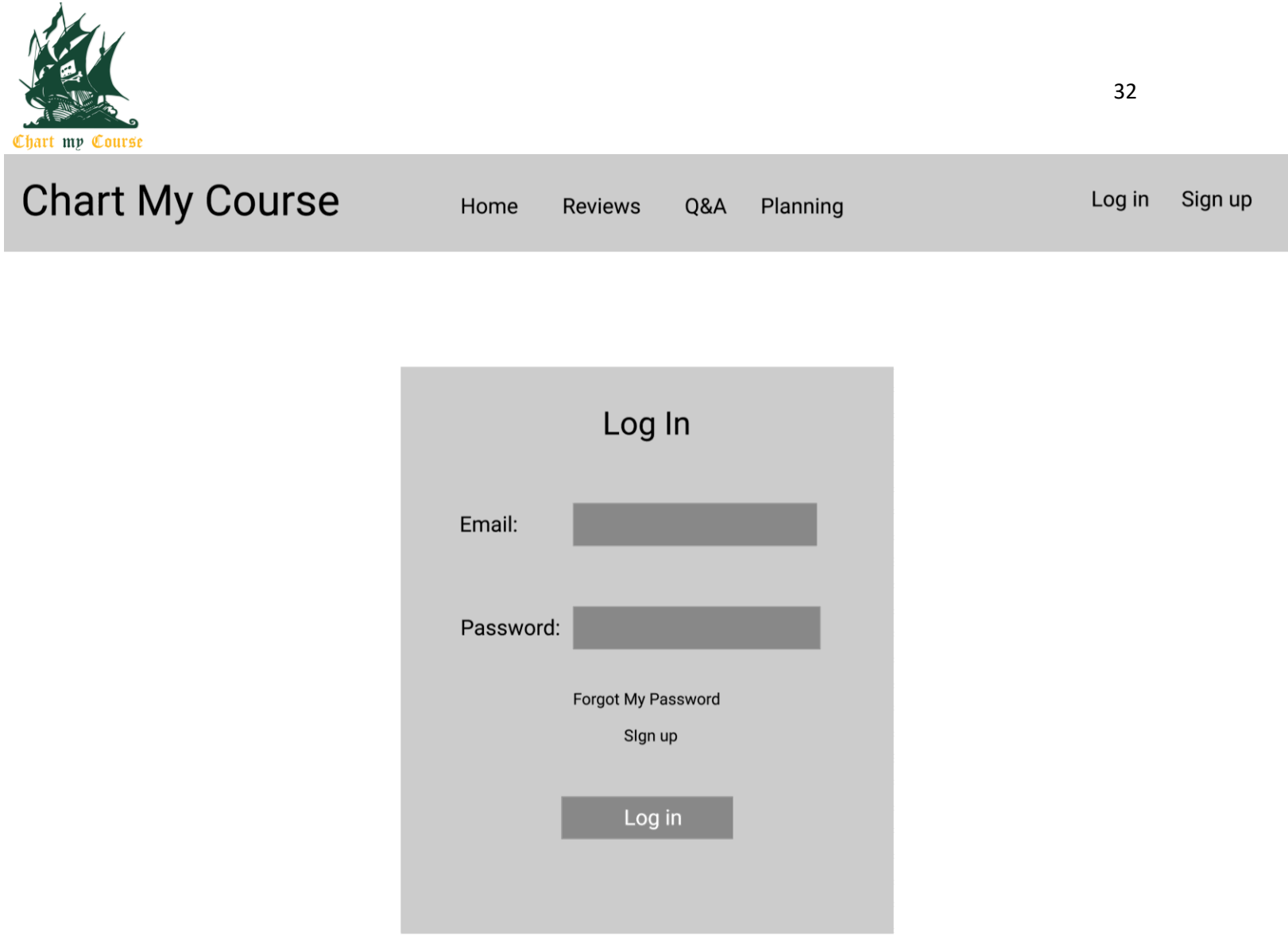


Chart my Course
Chart My Course
Home
Reviews
Q&A
Planning
Log in
Sign up
Log In
Email:
Password:
Forgot My Password
SIgn up
Log in


Chart my Course

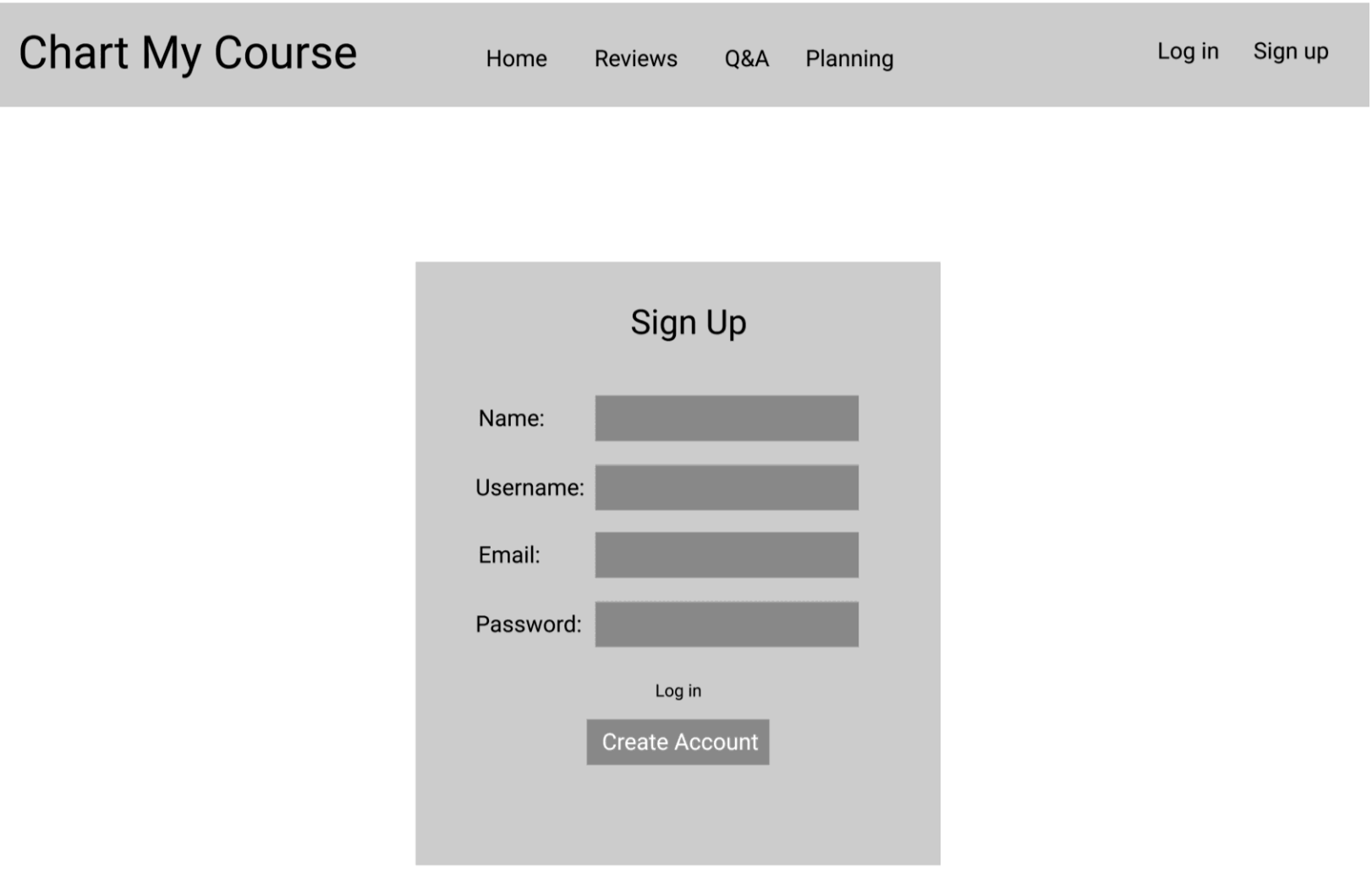


Chart My Course
Home
Reviews
Q&A
Planning
Log in
Sign up
Sign Up
Name:
Username:
Email:
Password:
Log in
Create Account

Chart My Course

Home Reviews Q&A Planning

Log in Sign up

Select Professor

Select Filter

Chart My Course

Home Reviews Q&A Planning

Log in Sign up

Search

Chart My Course

Home Reviews Q&A Planning

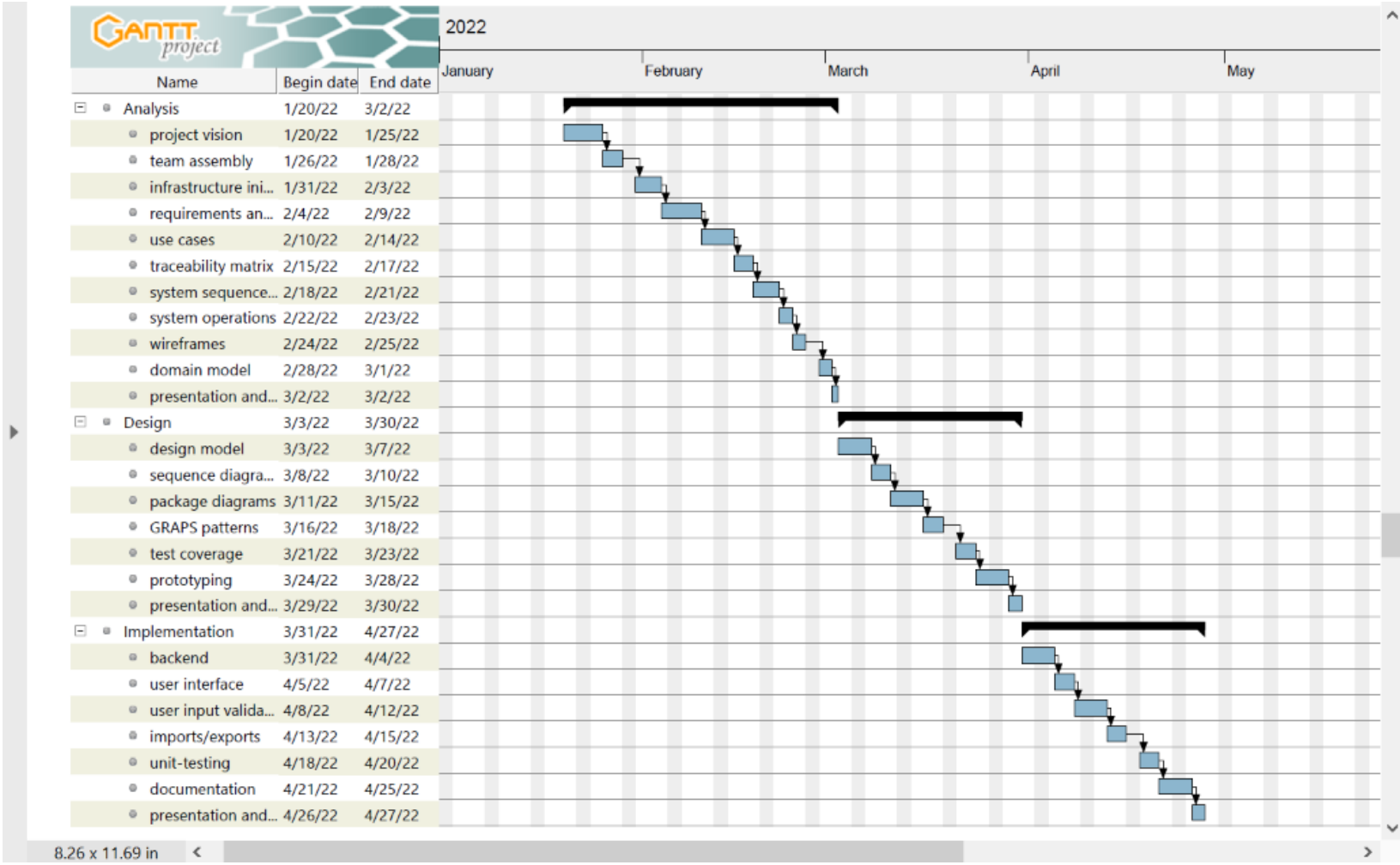
Log in Sign up

Planning

Recommended Courses

Recommended Professors

7. Gantt Diagram



Name	Begin date	End date
Analysis	1/20/22	3/2/22
project vision	1/20/22	1/25/22
team assembly	1/26/22	1/28/22
infrastructure ini...	1/31/22	2/3/22
requirements an...	2/4/22	2/9/22
use cases	2/10/22	2/14/22
traceability matrix	2/15/22	2/17/22
system sequence...	2/18/22	2/21/22
system operations	2/22/22	2/23/22
wireframes	2/24/22	2/25/22
domain model	2/28/22	3/1/22
presentation and...	3/2/22	3/2/22
Design	3/3/22	3/30/22
design model	3/3/22	3/7/22
sequence diagra...	3/8/22	3/10/22
package diagrams	3/11/22	3/15/22
GRAPS patterns	3/16/22	3/18/22
test coverage	3/21/22	3/23/22
prototyping	3/24/22	3/28/22
presentation and...	3/29/22	3/30/22
Implementation	3/31/22	4/27/22
backend	3/31/22	4/4/22
user interface	4/5/22	4/7/22
user input valida...	4/8/22	4/12/22
imports/exports	4/13/22	4/15/22
unit-testing	4/18/22	4/20/22
documentation	4/21/22	4/25/22
presentation and...	4/26/22	4/27/22

8. Jira

Our team uses Jira for task and time tracking – it is accessible at https://chartmycourse.atlassian.net.

9. Website

Our team maintains a website for content relevant to the project, found at https://hkaase.github.io/ChartMyCourse/

10. Git Link

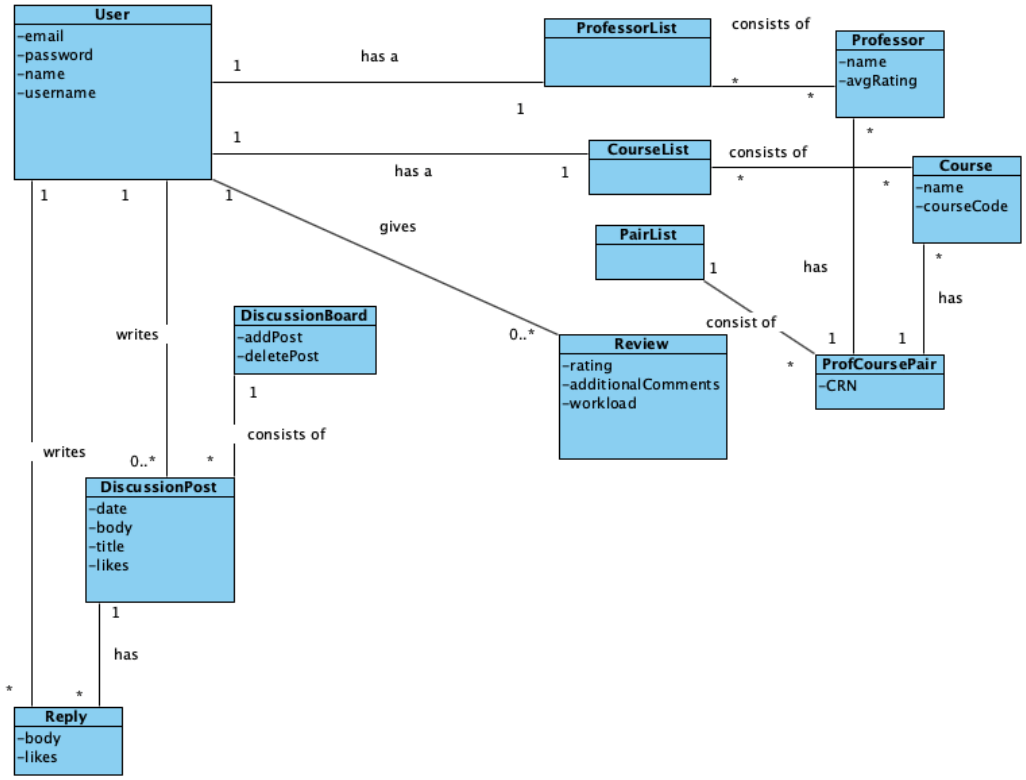
Our Git repository can be found at https://github.com/hkaase/ChartMyCourse.

11. UI Demo

Our UI Demo is on the GitHub. The JAR file can be downloaded from here. The source code and ensuing project can be found here.

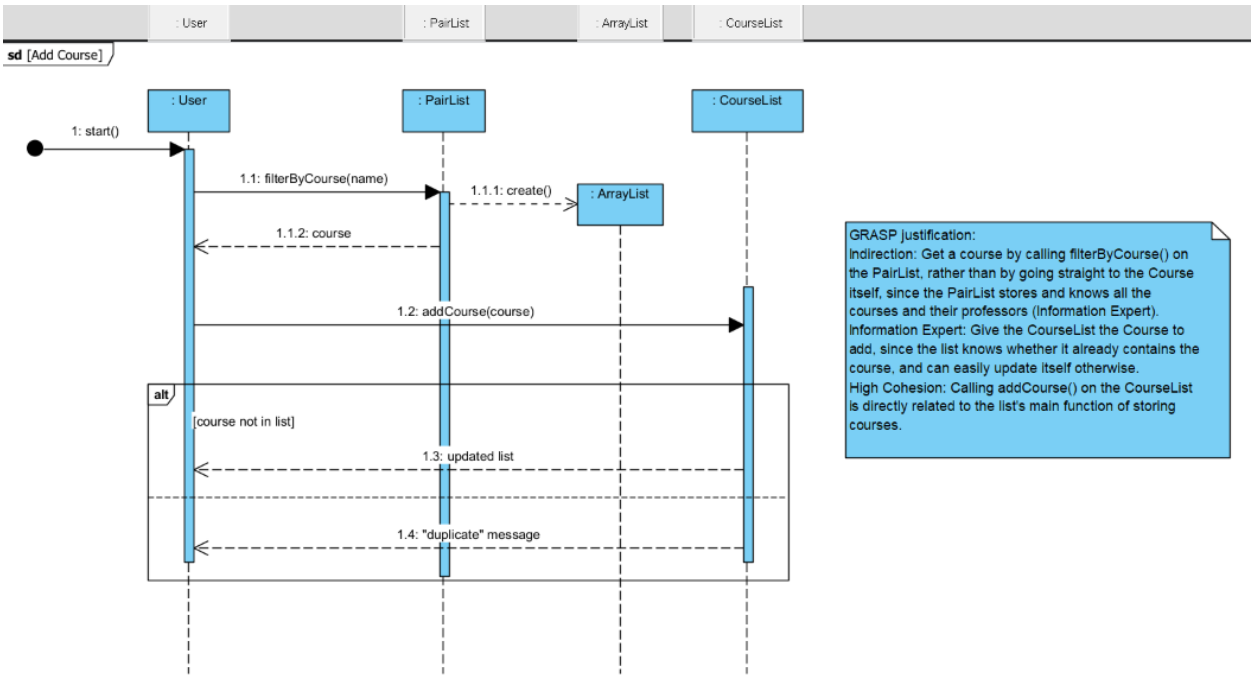
12. Corrections from Iteration 1

From our iteration 1, we changed two things. This is our new domain model:

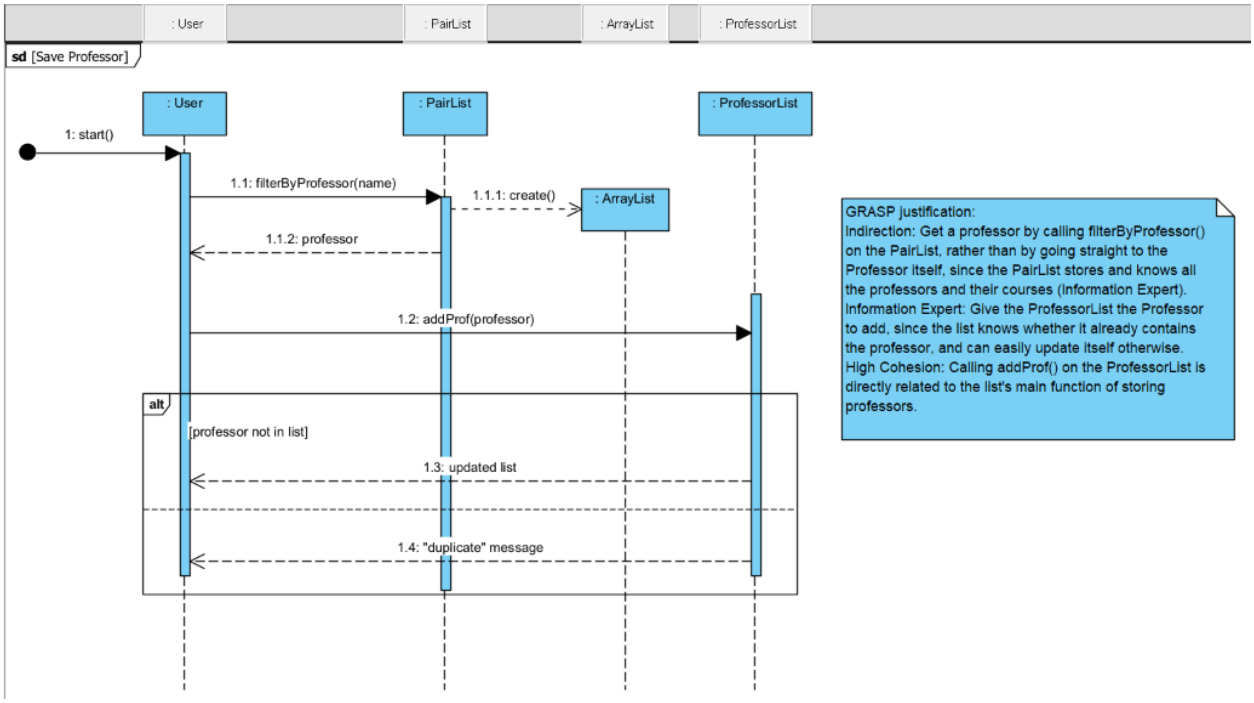


And this is our updated class diagram. It is now three separate

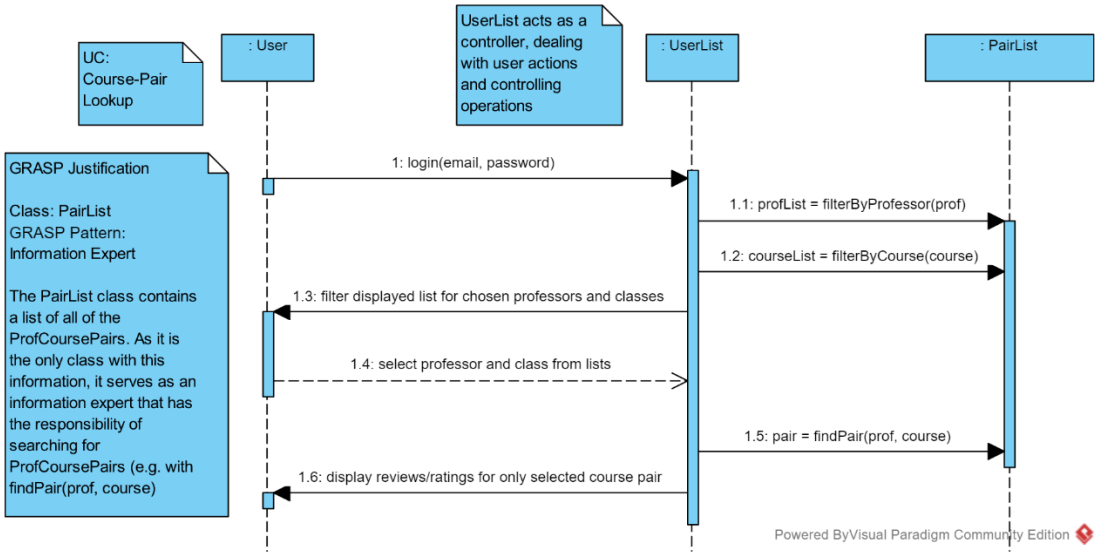
13. Design Diagrams

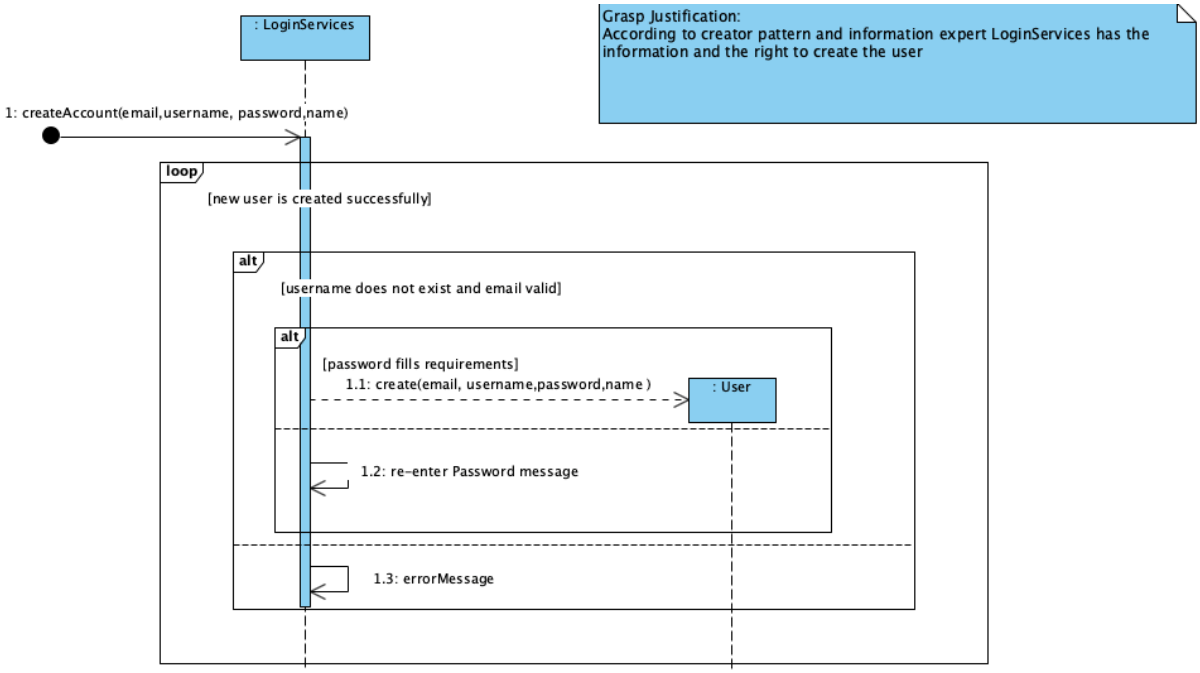


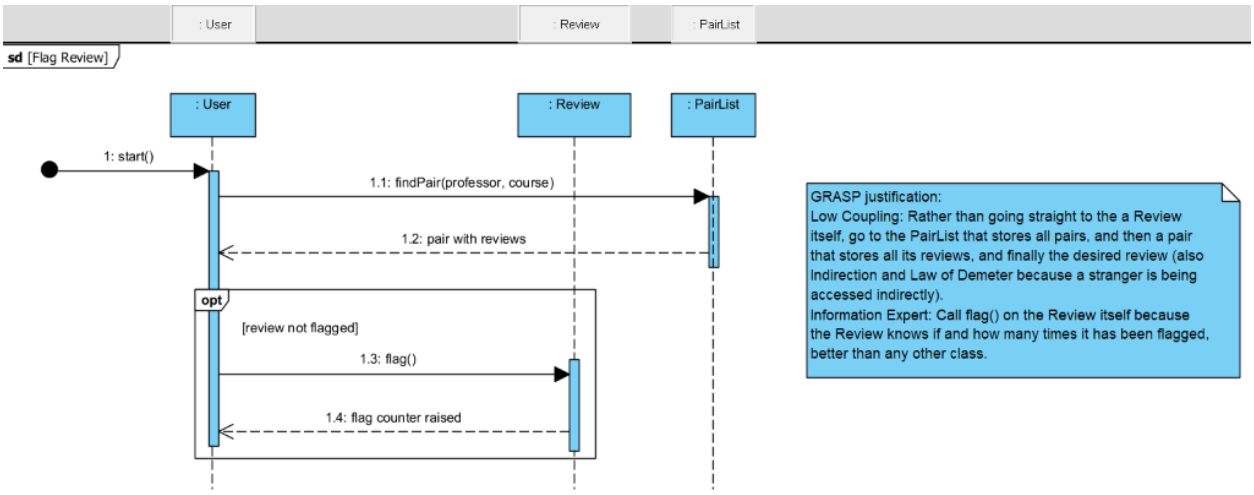
GRASP justification:
Indirection: Get a course by calling filterByCourse() on the PairList, rather than by going straight to the Course itself, since the PairList stores and knows all the courses and their professors (Information Expert).
Information Expert: Give the CourseList the Course to add, since the list knows whether it already contains the course, and can easily update itself otherwise.
High Cohesion: Calling addCourse() on the CourseList is directly related to the list's main function of storing courses.



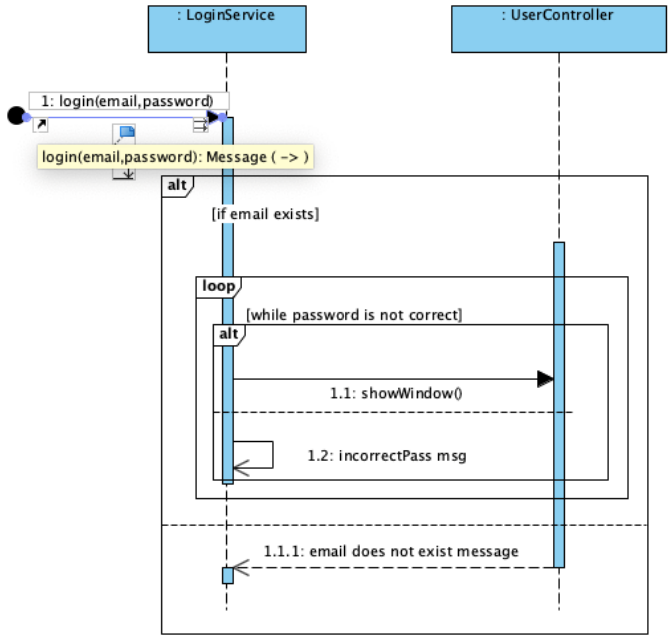
GRASP justification:
Indirection: Get a professor by calling filterByProfessor() on the PairList, rather than by going straight to the Professor itself, since the PairList stores and knows all the professors and their courses (Information Expert).
Information Expert: Give the ProfessorList the Professor to add, since the list knows whether it already contains the professor, and can easily update itself otherwise.
High Cohesion: Calling addProf() on the ProfessorList is directly related to the list's main function of storing professors.


UC:
Course-Pair
Lookup
: User
UserList acts as a
controller, dealing
with user actions
and controlling
operations
: UserList
: PairList
GRASP Justification
Class: PairList
GRASP Pattern:
Information Expert
The PairList class contains
a list of all of the
ProfCoursePairs. As it is
the only class with this
information, it serves as an
information expert that has
the responsibility of
searching for
ProfCoursePairs (e.g. with
findPair(prof, course)
1: login(email, password)
1.1: profList = filterByProfessor(prof)
1.2: courseList = filterByCourse(course)
1.3: filter displayed list for chosen professors and classes
1.4: select professor and class from lists
1.5: pair = findPair(prof, course)
1.6: display reviews/ratings for only selected course pair
Powered ByVisual Paradigm Community Edition

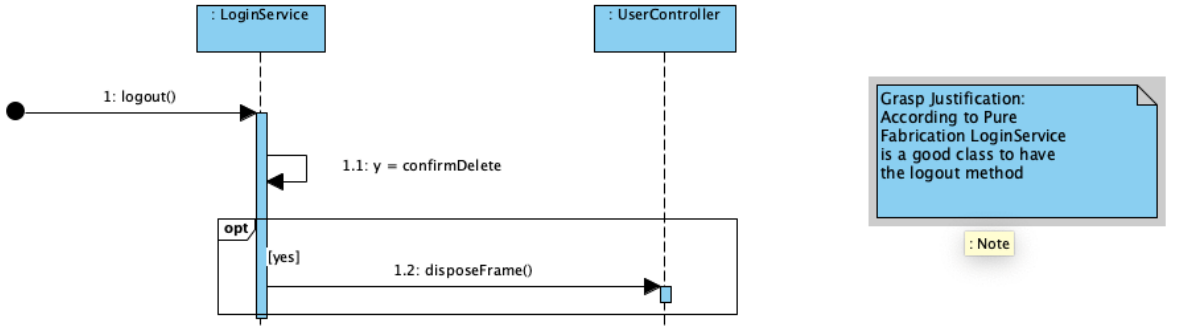

: LoginServices
Grasp Justification:
According to creator pattern and information expert LoginServices has the
information and the right to create the user
1: createAccount(email,username, password,name)
loop
[new user is created successfully]
alt
[username does not exist and email valid]
alt
[password fills requirements]
1.1: create(email, username,password,name)
: User
1.2: re-enter Password message
1.3: errorMessage

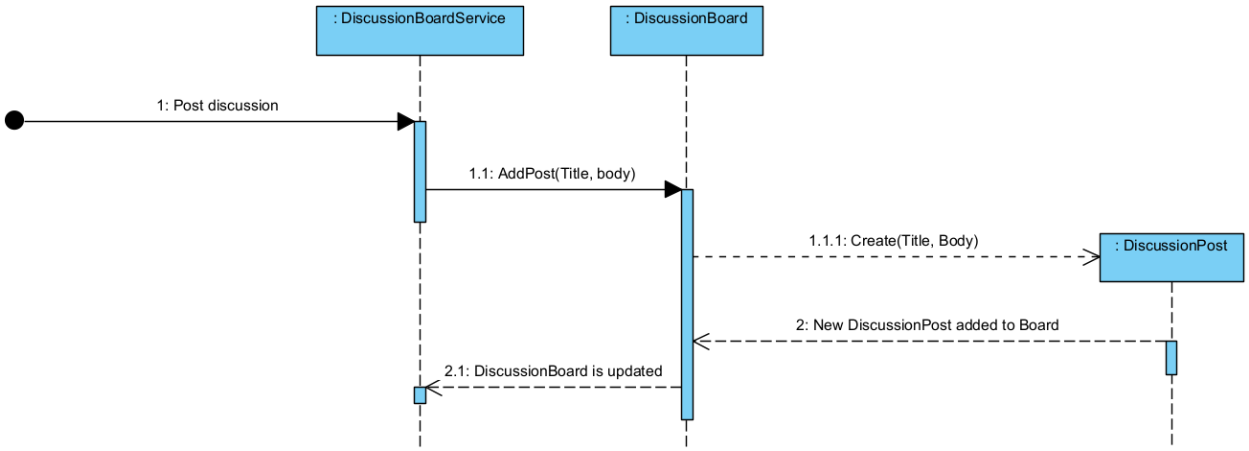


GRASP justification:
Low Coupling: Rather than going straight to the a Review itself, go to the PairList that stores all pairs, and then a pair that stores all its reviews, and finally the desired review (also Indirection and Law of Demeter because a stranger is being accessed indirectly).
Information Expert: Call flag() on the Review itself because the Review knows if and how many times it has been flagged, better than any other class.



Grasp Justification:
According to Information expert since all the collection of user is in Login Service hence login is a part of LoginService.

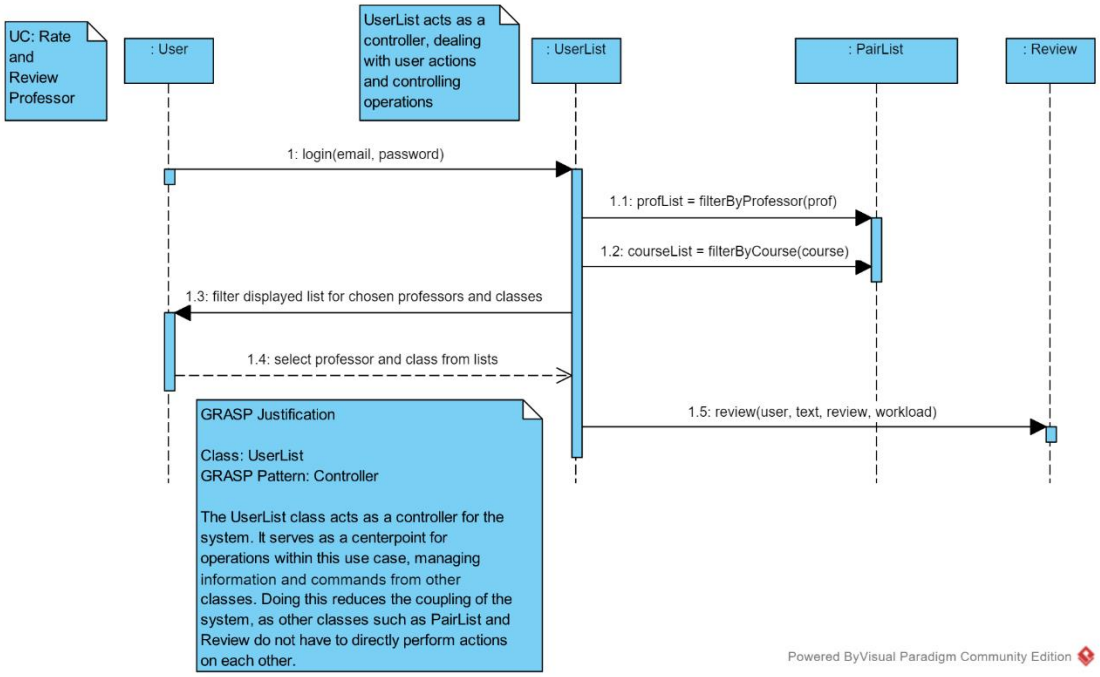


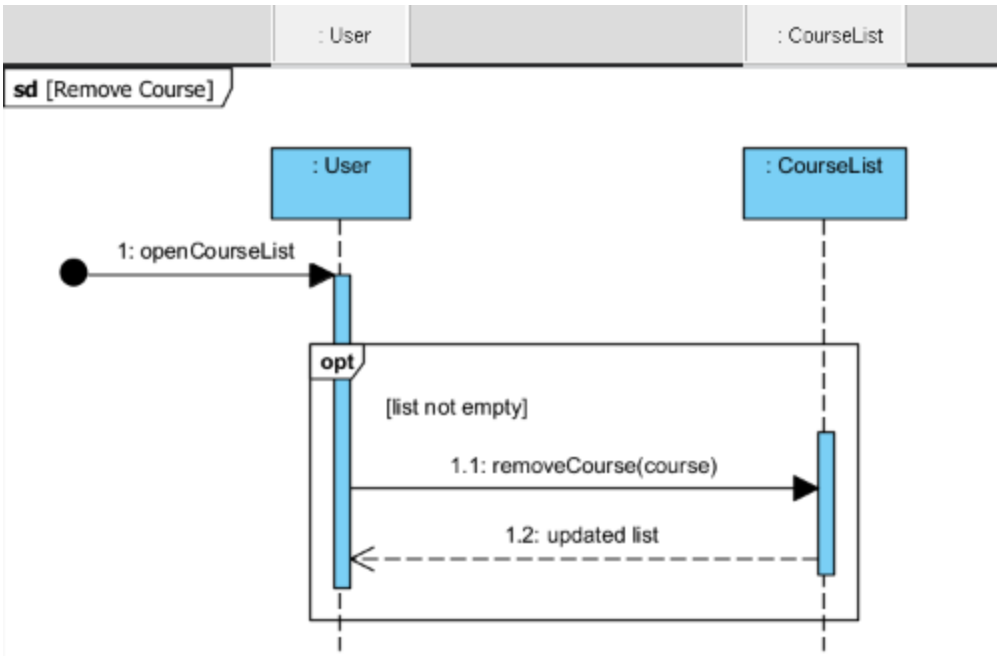


Grasp Justification

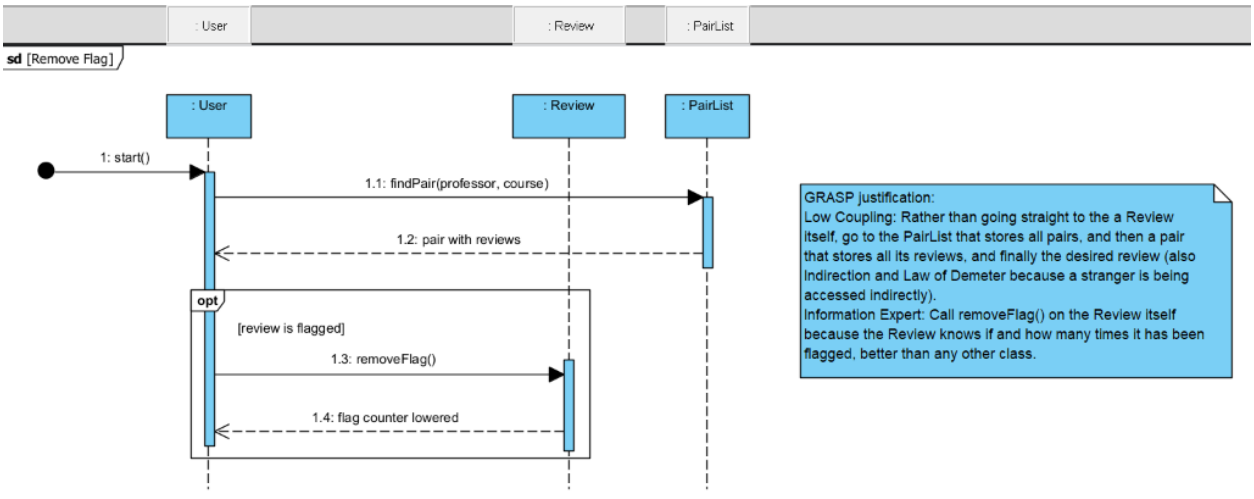
Operation: addPost()
Class: DiscussionBoard
GraspPattern: Creator
The DiscussionBoard class is in charge of creating
a new instance of a DiscussionPost that it will add to
its list of Posts.

Powered By: Visual Paradigm Community Edition

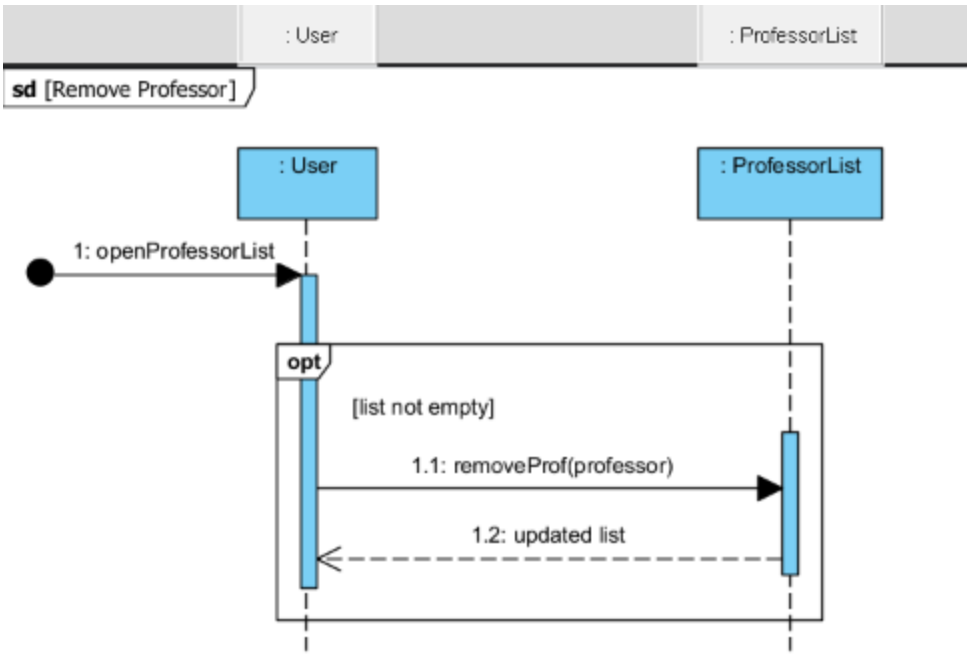

UC: Rate and Review Professor
: User
UserList acts as a controller, dealing with user actions and controlling operations
: UserList
: PairList
: Review
1: login(email, password)
1.1: profList = filterByProfessor(prof)
1.2: courseList = filterByCourse(course)
1.3: filter displayed list for chosen professors and classes
1.4: select professor and class from lists
1.5: review(user, text, review, workload)
GRASP Justification
Class: UserList
GRASP Pattern: Controller
The UserList class acts as a controller for the system. It serves as a centerpoint for operations within this use case, managing information and commands from other classes. Doing this reduces the coupling of the system, as other classes such as PairList and Review do not have to directly perform actions on each other.
Powered ByVisual Paradigm Community Edition



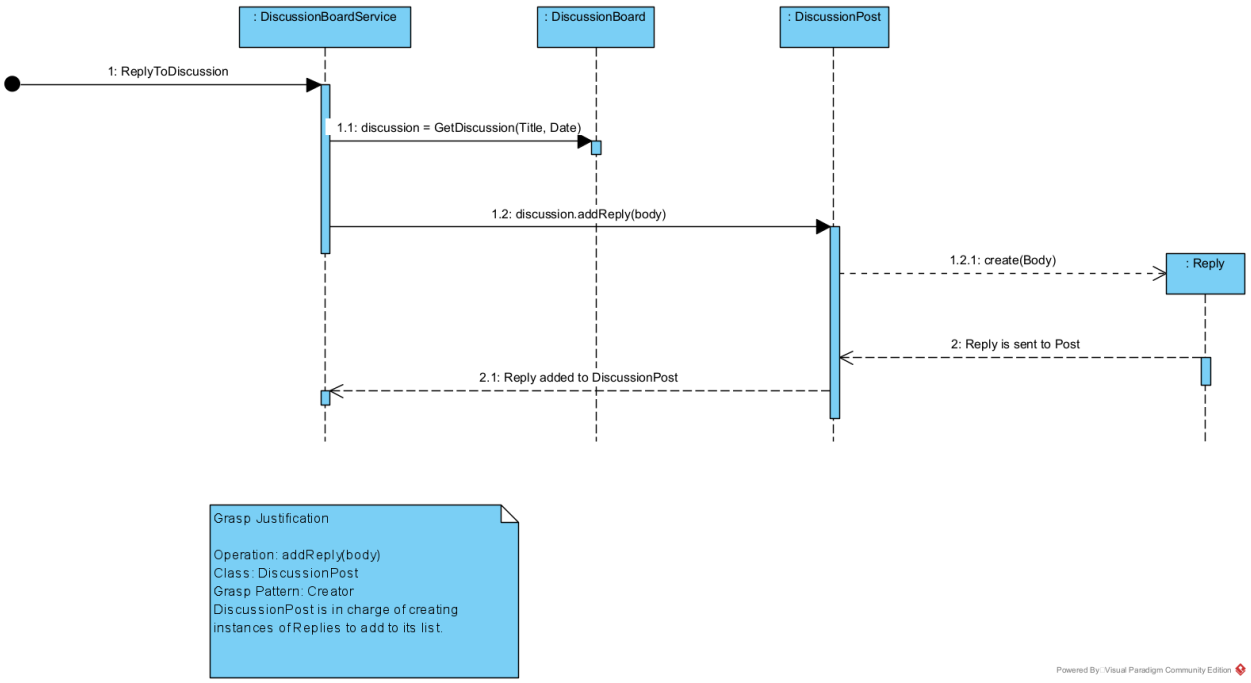
GRASP justification:
Indirection: Rather than going straight to the course itself, go to the CourseList, which stores all saved courses together.
High Cohesion: Calling removeCourse() on the CourseList is directly related to the list's main function of storing courses.

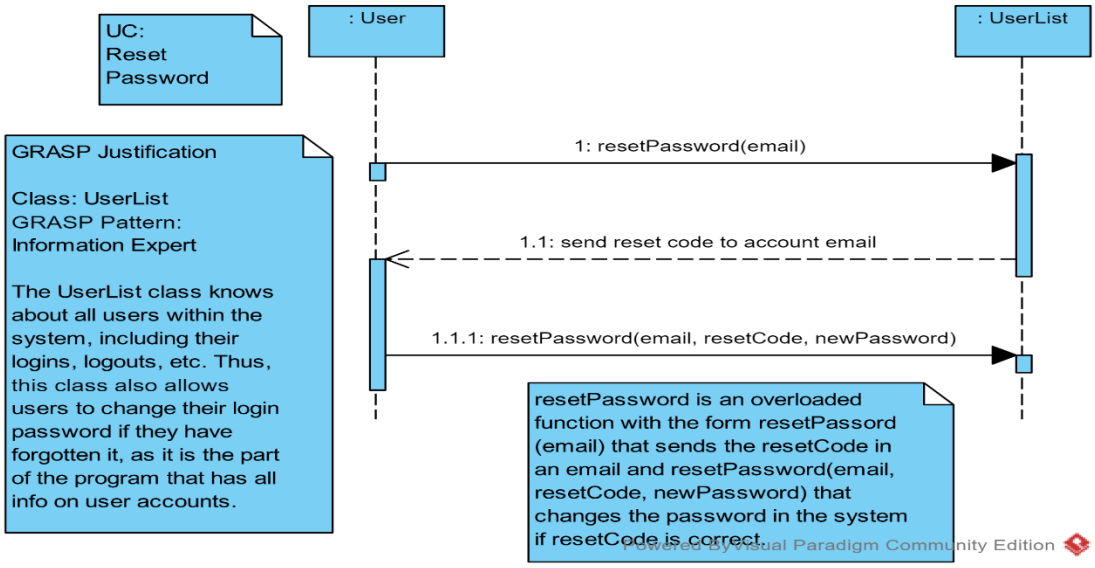


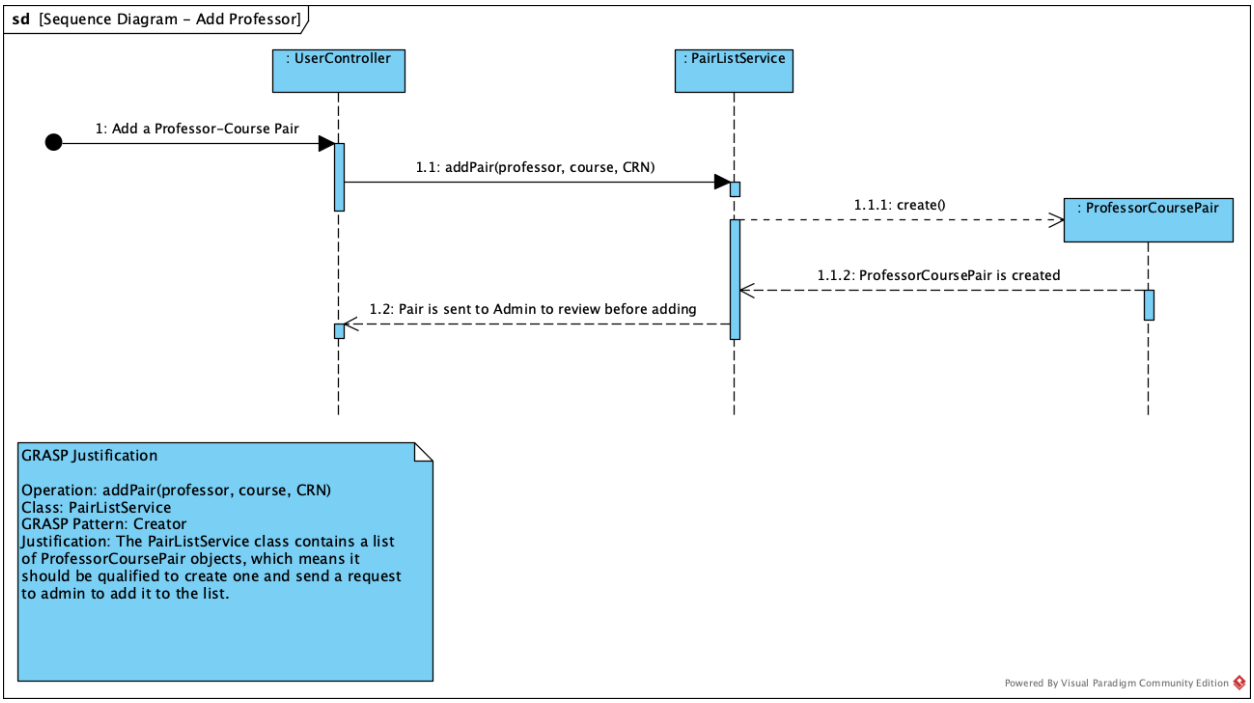
GRASP justification:
Low Coupling: Rather than going straight to the a Review itself, go to the PairList that stores all pairs, and then a pair that stores all its reviews, and finally the desired review (also Indirection and Law of Demeter because a stranger is being accessed indirectly).
Information Expert: Call removeFlag() on the Review itself because the Review knows if and how many times it has been flagged, better than any other class.

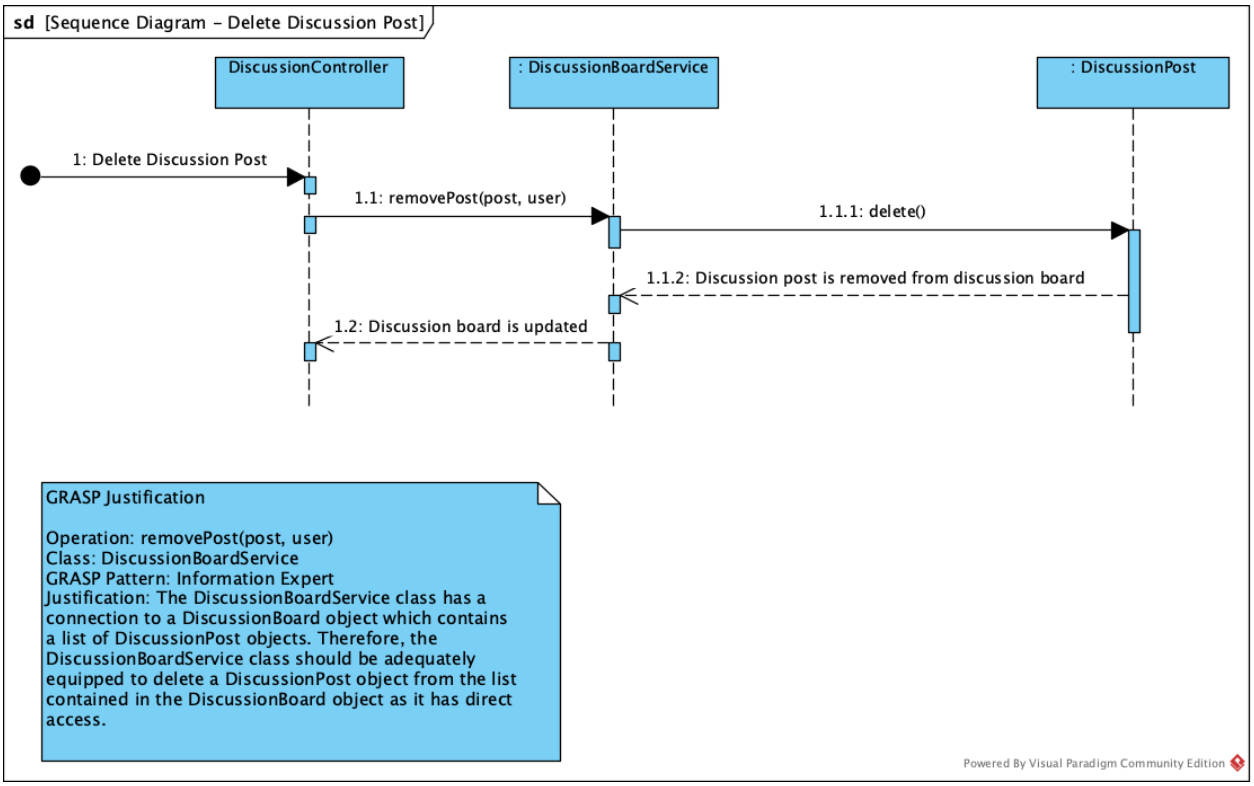


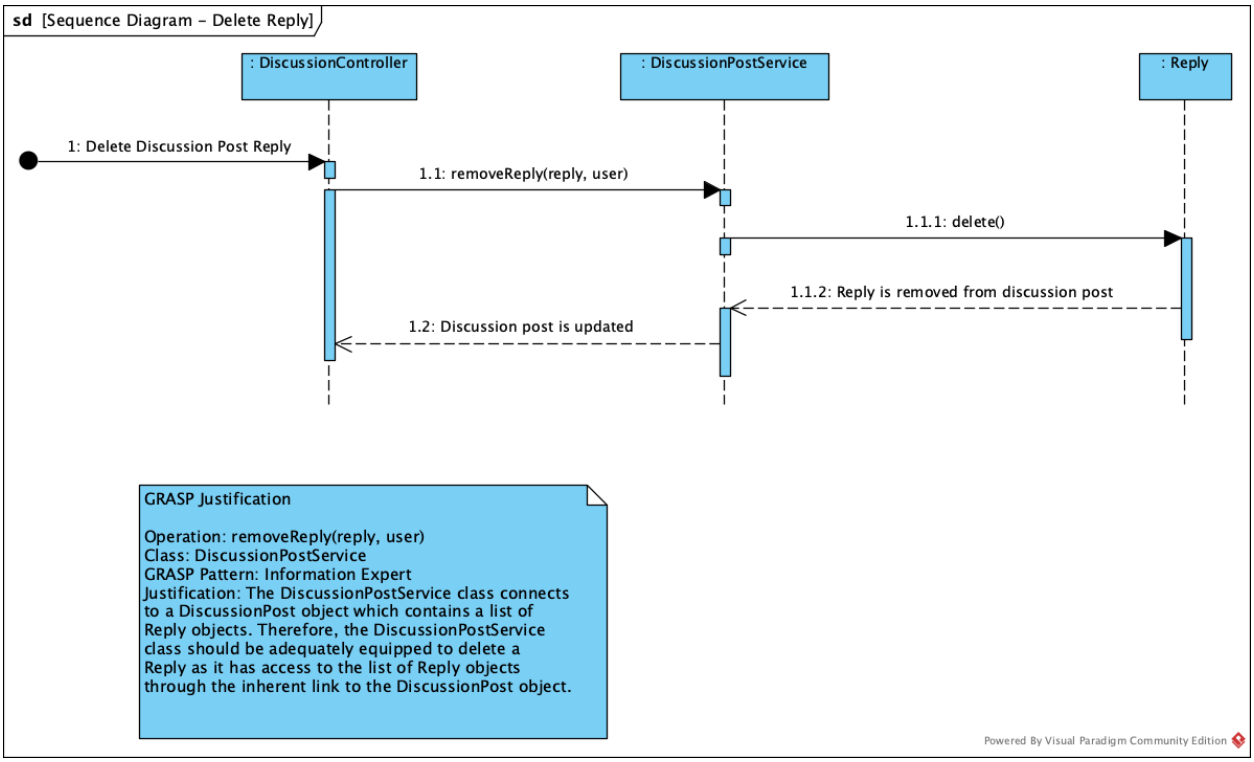
GRASP justification:
Indirection: Rather than going straight to the Professor itself, go to the ProfessorList, which stores all saved Professors together.
High Cohesion: Calling removeProf() on the ProfessorList is directly related to the list's main function of storing Professors.

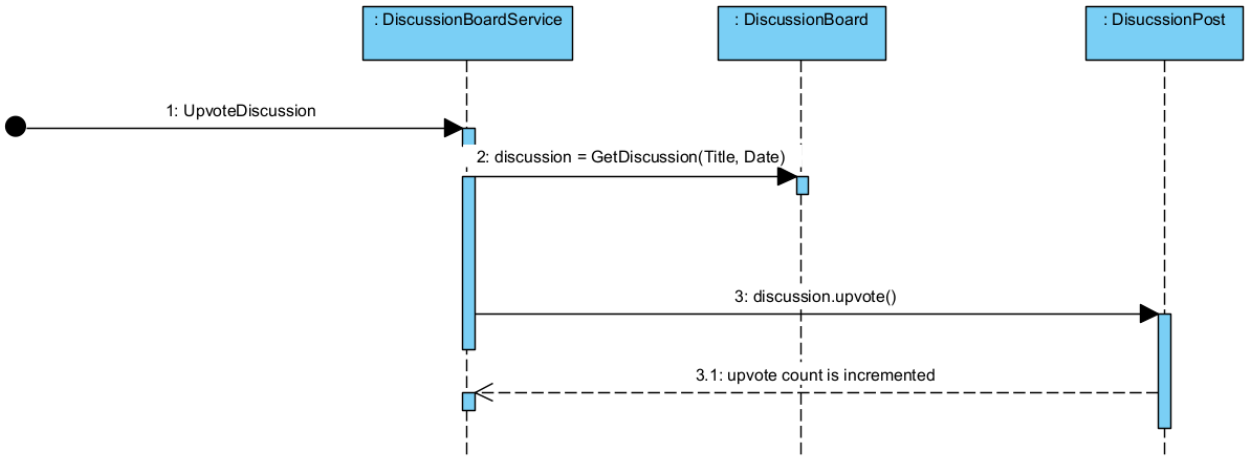

: DiscussionBoardService
: DiscussionBoard
: DiscussionPost
1: ReplyToDiscussion
1.1: discussion = GetDiscussion(Title, Date)
1.2: discussion.addReply(body)
1.2.1: create(Body)
: Reply
2: Reply is sent to Post
2.1: Reply added to DiscussionPost
Grasp Justification
Operation: addReply(body)
Class: DiscussionPost
Grasp Pattern: Creator
DiscussionPost is in charge of creating instances of Replies to add to its list.
Powered By: Visual Paradigm Community Edition


UC:
Reset
Password
: User
: UserList
GRASP Justification
Class: UserList
GRASP Pattern:
Information Expert
The UserList class knows about all users within the system, including their logins, logouts, etc. Thus, this class also allows users to change their login password if they have forgotten it, as it is the part of the program that has all info on user accounts.
1: resetPassword(email)
1.1: send reset code to account email
1.1.1: resetPassword(email, resetCode, newPassword)
resetPassword is an overloaded function with the form resetPassord (email) that sends the resetCode in an email and resetPassword(email, resetCode, newPassword) that changes the password in the system if resetCode is correct.
Powered By: Visual Paradigm Community Edition


sd [Sequence Diagram – Add Professor]
: UserController
: PairListService
1: Add a Professor–Course Pair
1.1: addPair(professor, course, CRN)
1.1.1: create()
: ProfessorCoursePair
1.1.2: ProfessorCoursePair is created
1.2: Pair is sent to Admin to review before adding
GRASP Justification
Operation: addPair(professor, course, CRN)
Class: PairListService
GRASP Pattern: Creator
Justification: The PairListService class contains a list of ProfessorCoursePair objects, which means it should be qualified to create one and send a request to admin to add it to the list.
Powered By Visual Paradigm Community Edition

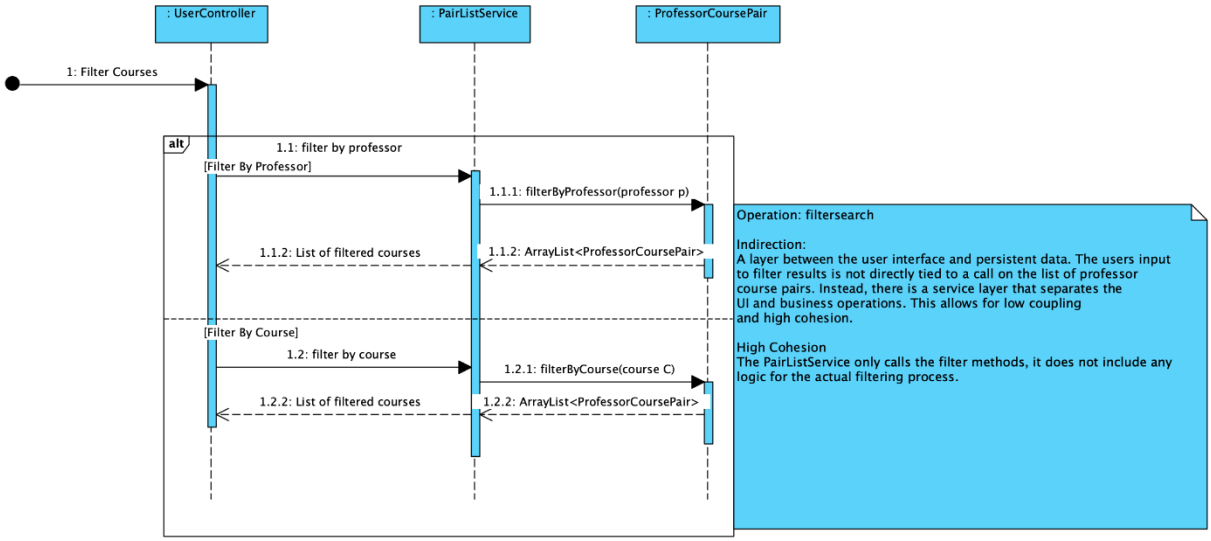

sd [Sequence Diagram – Delete Discussion Post]
DiscussionController
: DiscussionBoardService
: DiscussionPost
1: Delete Discussion Post
1.1: removePost(post, user)
1.1.1: delete()
1.1.2: Discussion post is removed from discussion board
1.2: Discussion board is updated
GRASP Justification
Operation: removePost(post, user)
Class: DiscussionBoardService
GRASP Pattern: Information Expert
Justification: The DiscussionBoardService class has a connection to a DiscussionBoard object which contains a list of DiscussionPost objects. Therefore, the DiscussionBoardService class should be adequately equipped to delete a DiscussionPost object from the list contained in the DiscussionBoard object as it has direct access.
Powered By Visual Paradigm Community Edition

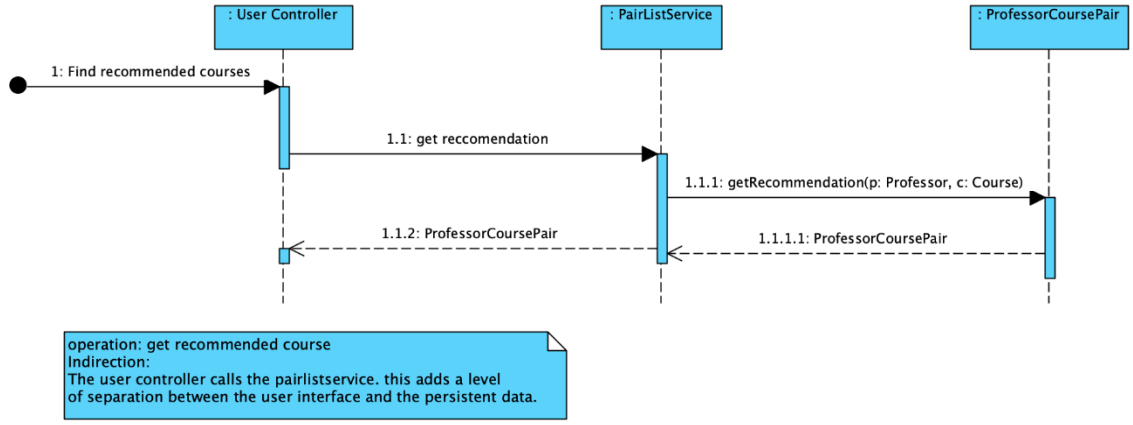


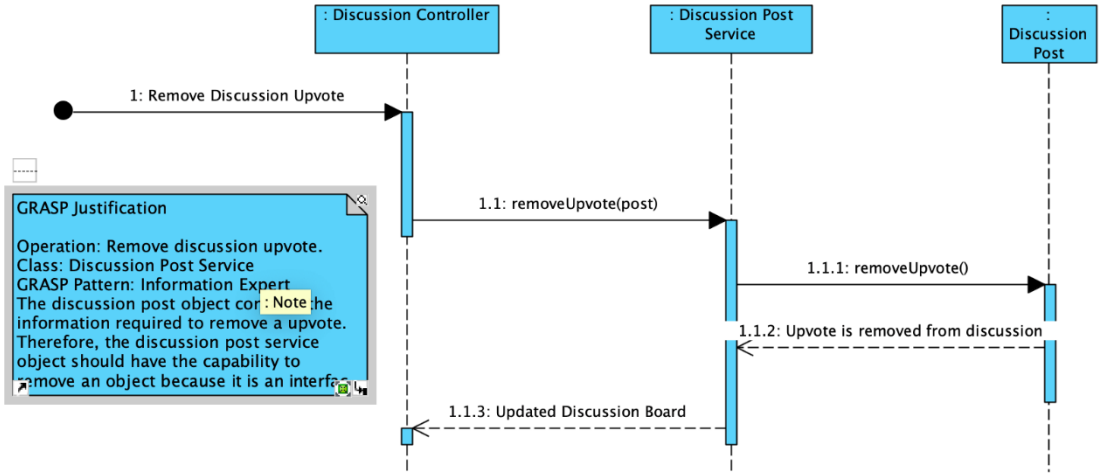


Grasp justification

Operation: upvote()
Class: DisucssionPost
Grasp Pattern: Information Expert
The DisucssionPost is in charge of keeping the information of how many upvotes it has. In order to change the number you must have an instance of DisucssionPost


: UserController
: PairListService
: ProfessorCoursePair
1: Filter Courses
alt
1.1: filter by professor
[Filter By Professor]
1.1.1: filterByProfessor(professor p)
1.1.2: List of filtered courses
1.1.2: ArrayList<ProfessorCoursePair>
[Filter By Course]
1.2: filter by course
1.2.1: filterByCourse(course C)
1.2.2: List of filtered courses
1.2.2: ArrayList<ProfessorCoursePair>
Operation: filtersearch
Indirection:
A layer between the user interface and persistent data. The users input to filter results is not directly tied to a call on the list of professor course pairs. Instead, there is a service layer that separates the UI and business operations. This allows for low coupling and high cohesion.
High Cohesion
The PairListService only calls the filter methods, it does not include any logic for the actual filtering process.


: User Controller
: PairListService
: ProfessorCoursePair
1: Find recommended courses
1.1: get reccomendation
1.1.1: getRecommendation(p: Professor, c: Course)
1.1.2: ProfessorCoursePair
1.1.1.1: ProfessorCoursePair
operation: get recommended course
Indirection:
The user controller calls the pairlistservice. this adds a level of separation between the user interface and the persistent data.



14. GRASP Justification

GRASP justifications can be found in the above diagrams.

15. Test Coverage Plan

Testing Plan

Account

- Test account creation
- Test account creation with bad email (not a Baylor email)
- Test account login
- Test account logout
- Test account Reset Password

Saved Courses

- Test adding a course
- Test adding a duplicate course (should not be able to add a duplicate)
- Test removing a course
- Test finding recommended courses

Saved Professors

- Test adding a professor
- Test adding a duplicate professor (should not be able to add a duplicate)
- Test removing a professor

Reviews

- Test writing an empty review
- Test canceling a review

- Test writing a half-completed review
- Test a completed review
- Test searching for a professor
- Test searching for a course
- Test filtering for a professor
- Test filtering for a course
- Test filtering by reviews
- Test filtering by ratings
- Test filtering by workload
- Test Flagging a review
- Test canceling a flagged review

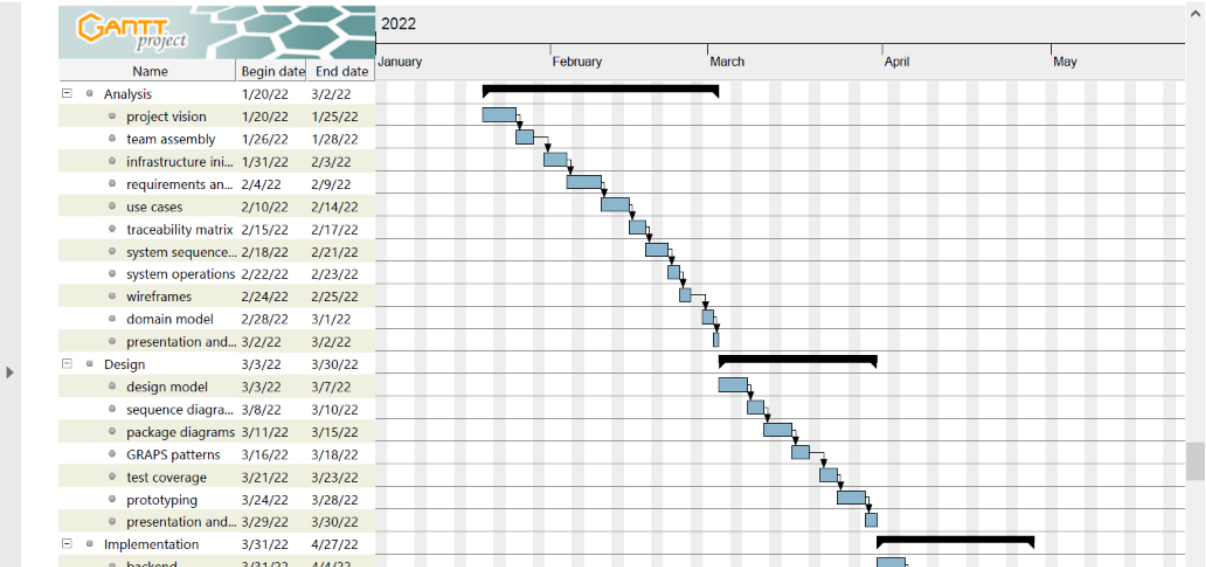
Discussion Board

- Test Posting to discussion board
- Test Posting an empty discussion
- Test canceling a post to a discussion board
- Test replying to a discussion board
- Test posting an empty reply
- Test canceling a reply to a discussion board
- Test upvoting a discussion
- Test removing upvote
- Test removing discussion post
- Test removing a discussion post you didn't write
- Test removing a discussion reply
- Test removing a discussion reply you didn't write

Other

- Test adding a new professor/course pair
- Test adding an existing professor course pair
- Test adding an existing professor as a professor course pair
- Test adding and existing course as a professor course pair

16. Updated Gantt Diagram



17. Issue Tracking

Issue tracking is done for code issues through GitHub issues. For non-code issues (compliance, design consistency), issues are tracked through Jira. Additionally, Jira issues can be linked to commits on GitHub, and vice versa.

18. Timecards/Point Distribution

Team Member	Hours Worked	Distribution of Grade
Varun Apte	28	16.66%
Ricardo Boone	28	16.66%
Warren Burrus	28	16.66%
Harm Drenth	28	16.66%
Mia Gortney	28	16.66%
Anshpreet Kaur	28	16.66%
Josh Wilson	28	16.66%