Software Requirements Specification (SRS)
Discussion Board

Team: Group 5

Authors: Alex Scheufele, Tyler Lordan, Aidan Scribner, Bryan Tait
Customer: Educational Institutions
Instructor: James Daly

1 Introduction

This software requirements specifications(SRS) document outlines and documents the necessary details for the development of the proposed message board overhaul for Blackboard.

In this document, there will be detailed subsections on the following:

1.1 Purpose, 1.2 Scope, 1.3 Abbreviations, 1.4 Organization

2 Overall Description, 2.1 Product Perspective, 2.2 Product Functions, 2.3 User Characteristics, 2.4 Constraints, 2.5 Dependencies, 2.6 Apportioning Requirements

3 Specific Requirements

4 Modeling Requirements, 4.1 Use-Case Diagram, 4.2 Class Diagram, 4.3 sequence Diagram, 4.4 State Diagram

5 Prototype

6 References

7 Contract

1.1 Purpose

The purpose of this document is to establish a list of basic requirements between clients, users, stakeholders, and developers of our proposed message board. This document is for all the aforementioned to review the project parameters and hereby follow the established outline for the creation, documentation and deployment of the product.

1.2 Scope

The software product being developed is a message board for Blackboard. Blackboard is a Learning Management System(LMS)* used to access and manage academic course content by both student and instructor alike. BlackBoard currently has a discussion board as a feature, however it lacks in both its form and function.

The benefits of the message board is for students to have the ability to network amongst each other. Students are allowed to create topics to lead discussions, and enable student collaboration. There will be several interactive forum features, students can rate posts/threads* that they found helpful and instructors can endorse answers. This allows greater inactivity amongst students, and their respective instructors.

The objective is to eliminate the need for 3rd party websites like Piazza for student and instructor collaboration. This will allow students to seek help from each other for subjects covered in class and assignments. By removing the need for 3rd party sites the instructor can easily moderate student discourse and endorse questions or answers. All of this will occur within the BlackBoard LMS ecosystem via a web server.

The message board will enable students and instructors to create posts which can be sorted by instructor created tags. Students can rate posts allowing the most pressing questions to be shown at the top. The instructor can also endorse an answer or question for it to be featured, so all accessing that thread will notice it.

*See 1.3 for definitions of terminology

1.3 Definitions, acronyms, and abbreviations

Comment: A reply to a thread.

Discussion Thread: A topic that a student or instructor intends to discuss. Has a starting message to start the thread.

Dark mode: Dark colors for the background and light colors for the foreground i.e. white text on black background.

Endorse: A property that an instructor can give to a post. A post with this property will be given a higher priority than any post without it when a user views the discussion board.

HTTP: Hypertext Transfer Protocol, is the protocol used for communication over a computer network.

HTML: Hyper Text Markup Language, the standard markup language for documents to be displayed in a browser.

JSON: JavaScript Object Notation, is a human-readable data interchange format, to store and transmit object data.

LMS: Learning management system that hosts course content.

Netvotes: A counter that displays the total sum of user votes. Votes against or downvotes decrement the sum, while upvotes or votes in favor increment the sum.

Rating: A numerical property of posts, quantifying user approval or disapproval.

Post: A way to describe threads or comments.

Tag: A label describing a thread's content that is displayed next to a thread. Tags are used in sorting threads.

Reply: Another way of saying a Comment on a thread.

Forum: A feature in the current LMS we will replace. A top level container for threads. Forums have a title describing the content of contained threads.

1.4 Organization

The remainder of this document will include in depth descriptions of the products structure, integration into the Blackboard LMS, user expectations, constraints, and assumptions/dependencies. Use case, class, sequence, and state diagrams are shown in section 4. Section 4 provides an overview of the development process of the product and how it was modeled. The prototype is described near the end of the document in section 5. An example scenario of the prototype's potential usage is shown and described in detail. Any references are cited at the end of the document.

2 Overall Description

Section 2 will cover the context of the message board software and where it would be added in Blackboard. This section will also describe implementation details on how it will perform its functions. The expected user, constraints, assumptions and apportioning of requirements will also be listed.

2.1 Product Perspective

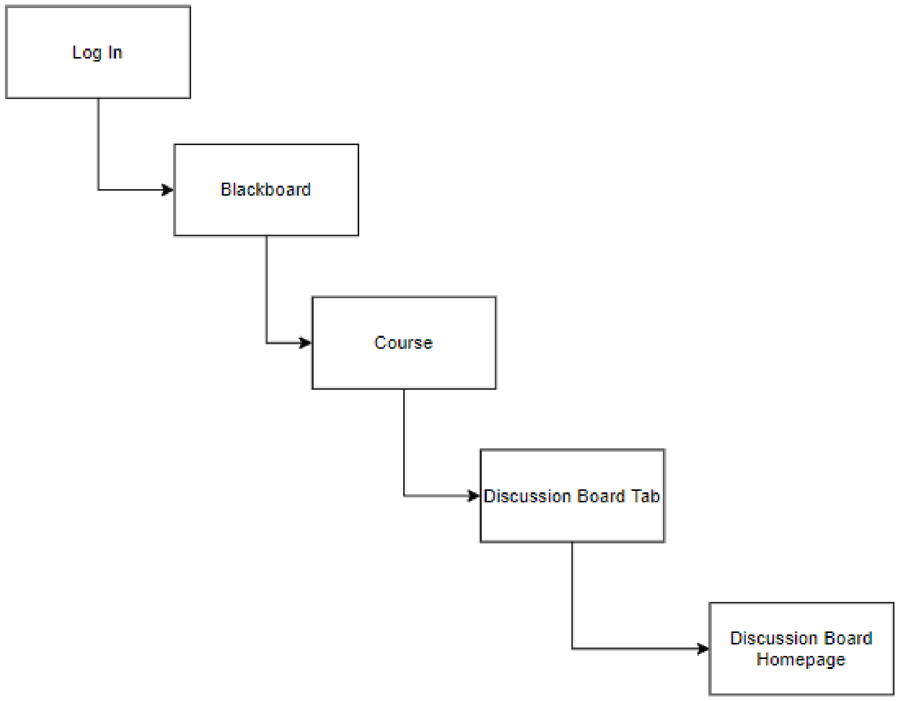
Each course in the LMS has a discussion board. The discussion board is a hub for communication between students and instructors. BlackBoard's current discussion board implementation suffers from poor organization and an unnatural layout. In specific, threads are only grouped by "Forum" with no ability to sort the forum list. Furthermore, the discussion board lacks the ability for students and instructors to provide feedback to a post without cluttering the thread with additional comments.

In developing a tool for students and instructors we faced several constraints. The most pressing is the narrow time frame for development. Another major constraint is using a flat JSON file for storage rather than a database. Lastly the team is constrained by library limitations and tools utilized in the creation of the product.

Our discussion board prototype will be accessible through any web browser supporting modern Javascript and HTML. Users will interact with the discussion board through the user interface using their computers display, keyboard, and mouse. The user will interface with the software through fillable text fields, buttons, and dropdown menus. The prototype will communicate with the user's web browser through the HTTP protocol.

Below is a diagram showing how our product fits into the blackboard LMS. Normally, the user would log in through blackboard, then access their course, and from their course they are able to access the discussion board. For the purposes of our prototype, the user can log in directly to the discussion board.

Figure 2.1.1 System Diagram

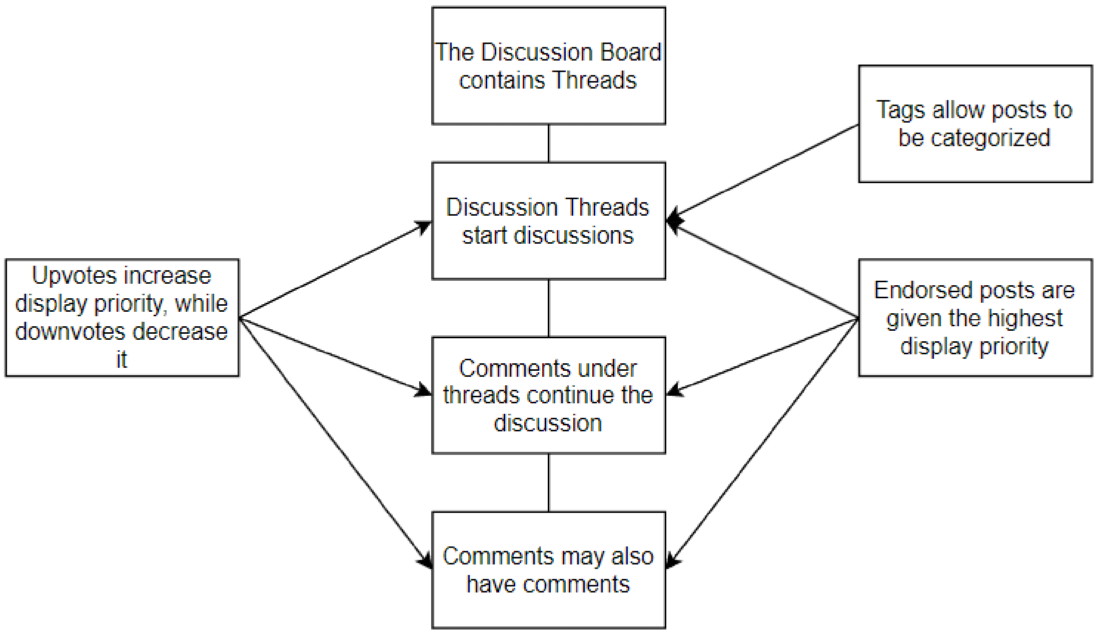


2.2 Product Functions

Users will be able to access the discussion board of their class. On the discussion board, users can create new threads that contain user-specified text. The threads will have a title section and a body section. When creating a thread, Users are prompted to input text for both the title and the body, they can also apply a tag that is selected from a list of predefined tags. After inputting valid text and applying a tag, the user may confirm the creation of the new thread (they may also cancel the creation of the thread at any point).

After the thread is created, it will be visible on the discussion board, any user may interact with the thread by either upvoting, downvoting, or replying to it. If a user chooses to reply to the thread, they will be prompted for input and after they've entered it, they may confirm to create a new comment on the thread (They can also cancel the creation, similarly to threads). Upvoting will increase the rating of the thread, while downvoting will decrease it. Comments under threads may also receive upvotes, downvotes, or replies. Users may edit or delete posts that they've created. Instructors may also endorse posts. Instructors have the ability to edit or delete posts that have been created by any user.

Figure 2.2.1 Goal Diagram


The Discussion Board contains Threads
Tags allow posts to be categorized
Discussion Threads start discussions
Upvotes increase display priority, while downvotes decrease it
Endorsed posts are given the highest display priority
Comments under threads continue the discussion
Comments may also have comments

2.3 User Characteristics

The expected user is an academic student or instructor that is proficient in navigating software interfaces and using computers. That means the user is able to use their mouse to navigate a website, and their keyboard to fill data fields. Users are expected to create posts that are relevant to the material on the discussion boards for that course.

2.4 Constraints

In developing a tool for students and instructors, the user interface must facilitate ease of use. This is the first interface constraint we face. Interacting with the discussion board must be intuitive for all users. From a technical viewpoint, an additional constraint is determined by the library we use to develop the front-end. Our ability to create a natural user experience will be constrained by this decision. The available storage of our web server will constrain the amount of data we can store, potentially limiting the amount of posts on the site. Storing data will be constrained to a JSON file storing the trees representing thread and reply structure. User operations are constrained to interacting with threads including but not limited to: creating posts, deleting posts, upvoting posts, downvoting posts, endorsing posts, replying to posts. Site operation is constrained by the necessity to present the user with an accurate, up to date, representation of the post data stored server side in the discussion board JSON file. A limitation of JSON is the inefficiency that arises when one single file becomes very large. Reading from and writing to the JSON file becomes timely, and issues of mutual exclusion can arise if many users are attempting operations concurrently. To address this a database may replace the JSON approach in a future release. Other restrictions will be forum/message board features that may need to be cut to fit in the development window.

2.5 Assumptions and Dependencies

It is assumed that the user is running a browser that is capable of running HTML5 and modern Javascript. In order to run the browser, it is also assumed that the user is on a computer that is capable of running the browser, and that they have an internet connection. In order to view and interact with the website, an internet connection, keyboard, mouse, and monitor are required.

2.6 Apportioning of Requirements

Three features will not be included initially, but are considered for future releases. Spelling and grammar correction for post content is the first. This feature was determined to be outside the scope of necessity and will not be included in the first release. Additionally the use of a database for content storage will be considered for future versions. The constraints of a JSON approach described in sections 2.1 and 2.4 make a good case for switching to a database approach. However, for the purposes of the initial release, JSON will suffice. Input sanitization is an important feature to filter profanity and inappropriate content. Though an important feature, it falls beyond the scope of necessity for the initial release and may be included in future releases.

3 Specific Requirements

1. Users can create discussion board messages.
   - 1.1. View posts on the discussion board.
   - 1.2. Post comments on discussion threads.
   - 1.3. Post Text on discussion threads.
   - 1.4. Original poster tag is displayed on comments made by the thread creator.
2. Posts are ranked based on user interactions.
   - 2.1. Users can upvote or downvote posts.
   - 2.2. Instructors can endorse posts.
3. Users can manage threads
   - 3.1. Instructors can moderate posts.
     - 3.1.1. Instructors can edit any post.

        3.1.2. Instructors can remove any post.

    3.2. Users can create threads.

    3.3. Students can apply tags when creating a thread.

4. Aesthetic requirements for specifying the appearance of the message board.

    4.1. The page will employ a dark color palette for all elements.

    4.2. Users can select tags upon thread creation to categorize a thread.

    4.3. Author’s display name is shown on posts.

5. Information storage to enable data persistency across sessions

    5.1. Home page for users to login into the discussion board.

4 Modeling Requirements

<u>Figure 4.1 Use-Case Diagram</u>

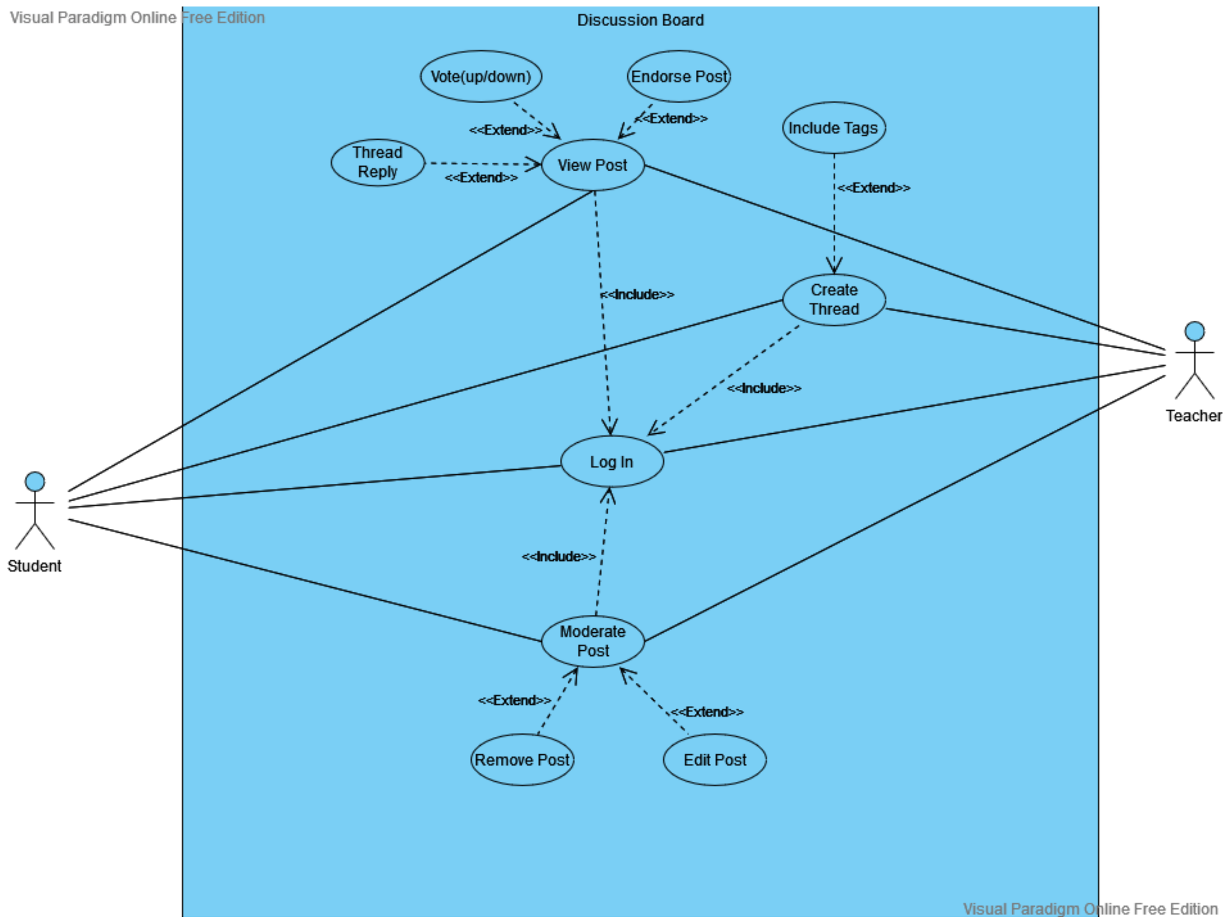


Fig 4.1 models user interaction with the system and the elements involved

Use Case Name:	Log In
Actors:	Student and Instructor
Description:	The user is prompt to login to gain access to the discussion board for the specific course
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Requirement 5.1
Uses cases:	N/A

Use Case Name:	Create Thread
Actors:	Student and Instructor
Description:	The user creates a new thread to discuss a topic related to something in the class
Type:	Primary
Includes:	Log In
Extends:	None
Cross-refs:	Requirement 3.2
Uses cases:	User must have logged into the system

Use Case Name:	Include tags
Actors:	Student and Instructor
Description:	Attaches a tag to the thread to allow for better organization of the threads
Type:	Secondary
Includes:	None
Extends:	Create Thread
Cross-refs:	Requirement 3.4
Uses cases:	User must have logged into the system and selected to create a thread

Use Case Name:	View Post
Actors:	Student and Instructor
Description:	Allows the user to view a thread and all of the comments associated with it
Type:	Primary
Includes:	Log In
Extends:	None
Cross-refs:	Requirement 1.1
Uses cases:	User must have logged into the system

Use Case Name:	Thread Reply

Actors:	Student and Instructor
Description:	A user can reply to a thread with a comment, allowing users to communicate to each other about a thread topic
Type:	Secondary
Includes:	None
Extends:	View Post
Cross-refs:	Requirement 1.2
Uses cases:	User must have logged into the system and selected to view a post

Use Case Name:	Vote(up/down)
Actors:	Student and Instructor
Description:	Allows users to vote a thread or comment in a thread up or down depending on if they found it helpful or not
Type:	Secondary
Includes:	None
Extends:	View Post
Cross-refs:	Requirement 2.1
Uses cases:	User must have logged into the system and selected to view a post

Use Case Name:	Endorse Post
Actors:	Instructor
Description:	Allows for the Instructor to endorse a post that they find is useful/helpful for the students to read
Type:	Secondary
Includes:	None
Extends:	View Post
Cross-refs:	Requirement 2.2
Uses cases:	User must have logged into system as an instructor and selected to view a post

Use Case Name:	Moderate Post
Actors:	Student and Instructor
Description:	Allows for a student to change their own posts and allows for a instructor to change any of the posts
Type:	Secondary
Includes:	Log In
Extends:	None
Cross-refs:	Requirement 3.1
Uses cases:	User must have logged into system as an instructor and selected to view a post

Use Case Name:	Remove Post
Actors:	Student and Instructor
Description:	Allows for students to remove their posts and for instructors to remove any post
Type:	Secondary
Includes:	None
Extends:	Moderate Post
Cross-refs:	Requirement 3.1.2
Uses cases:	User must have logged into system as an instructor and selected to view a post

Use Case Name:	Edit Post
Actors:	Student and Instructor
Description:	Allows for a student to edit their own post and a instructor can edit any of the posts
Type:	Secondary
Includes:	None
Extends:	Moderate Post
Cross-refs:	Requirement 3.1.1
Uses cases:	User must have logged into system as an instructor and selected to view a post

Figure 4.2 Class Diagram

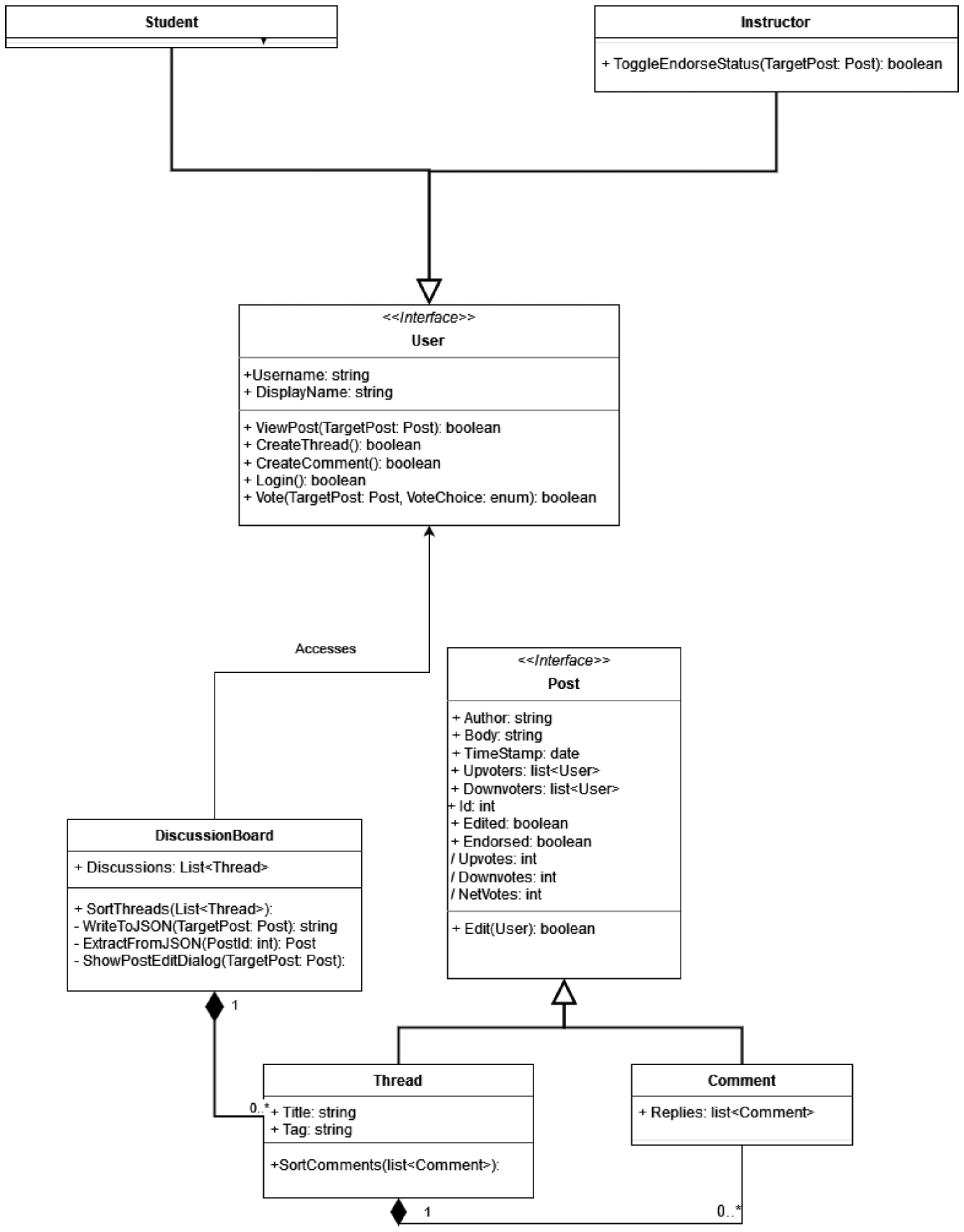


Figure 4.2 is a class diagram, showing how the classes interact, what attributes they have and methods they call in order to manage the discussion border.

Data Dictionary:

Element Name		**Description**
User		This interface represents a user of the discussion board. It defines the attributes and methods of Students and Instructors
Attributes	Username: String	This is the information for the user to use to login to the system
	DisplayName: String	This is the name that other students will see when the user make a post on the discussion board
Methods	ViewThread(Thread): boolean	This method allows for the user to view a specific thread
	Vote(TragetPost: Post): boolean	Allows for the user to upvote or downvote a post on the discussion board
	CreateComment(): boolean	This method allows for users to create a comment on a thread.
	Login(): boolean	This method allows the user to login.
	CreateThread(): boolean	This method allows for the user to create a new thread on the discussion board
Relationships		The interface is implemented by the Student and Instructors classes

Element Name	**Description**
Student	This class represents a Student user that will use the discussion board to create, view, and vote posts
Relationships	This class implements the User interfaces attributes and methods. It also

	accesses the discussion board

Element Name		**Description**
Instructor		This class represents a instructor user that will use the discussion board to create, view, vote, endorse, and moderate any posts
Methods	ToggleEndorseStatus(TargetPost: Post): boolean	Flips the current endorse status of a post.
Relationships		This class implements the User interfaces attributes and methods. It also accesses the discussion board

Element Name		**Description**
DiscussionBoard		This class represents the discussion board system and contains all of the threads organized in a certain order
Attributes	Discussions: List<Thread>	The list of all of the threads in the discussion boards
Methods	SortThreads(List<Thread>)	Allows for the discussion board to organize the list of threads in a certain way
	WriteToJSON(TargetPost: Post):string	Storing Post information in the JSON file as a string.
	ExtractFromJSON(postId: int): Post	Building the post object from the JSON file information, using an unique postId to identify it.
	ShowPostEditDialog(TargetPost: Post):	During post creation or editing show an edit dialog box to edit the text fields of the post

Relationships	Accessed by the users and it contains a list of the threads inside of it and organizes them.

Element Name		**Description**
Post		A post on the discussion board. It defines the attributes and methods for the Thread and Comment classes
Attributes	Author: String	Name of the user who made the post
	Body: String	Body text of the post
	TimeStamp: date	Time that the post was made
	Upvoters: list<User>	List of people who upvoted the post
	Downvoters: list<User>	List of people who downvoted the post
	Edited: boolean	State of whether the post has been edited after being posted
	Endorsed: boolean	State of whether the post has been endorsed or not by an instructor
	Id: int	A unique identifier for the post
	Upvotes: int	Number of upvotes the post has received
	Downvotes: int	Number of downvotes the post has received
	NetVotes: int	Absolute value of the difference between Upvotes and Downvotes.
Methods	Edit(User): boolean	Checks whether a user may edit a post, then allows editing if permissions are sufficient. Returns true if editing was successful and false otherwise.

Relationships	This interface is implemented by the Thread and comment classes

Element Name		**Description**
Thread		A thread in the discussion board, holding and organizing the comments.
Attributes	Title: String	Title of the thread that will be displayed on the discussion board to users
Methods	SortComments(list<Comment>)	Sorts the comments associated with this thread
Relationships		This class implements the post interfaces attributes and methods. It also contains multiple comments that serve as replies to the thread.

Element Name		**Description**
Comment		This class represents a reply to a thread and it holds sub-comments inside of it.
Attributes	Replies: list<Comment>	Contains the replies to this comment in the form of a list of sub comments
Relationships		This class implements the post interfaces attributes and methods. It contains subcomments that serve as replies to itself

Sequence Diagrams:

Diagram 1: User creates a thread on the discussion board

The user logs into the discussion board for their class. They create a new thread which calls the CreateThread() method in the DiscussionBoard object. This then displays a dialog box to edit the body of the thread by calling the method ShowPostEditDialog() which returns the Contents. Depending on what Contents returned as the system operates differently. If Contents is null then the system cancels out of creating a thread by returning CreateThread() as false. If Contents isn't null then the system stores the Contents in a new Thread object and returns the new set up of Thread object to the DiscussionBoard. The DiscussionBoard then converts the Thread to be stored in a json and stores it, it does this by calling the WriteToJson() method. Finally the DiscussionBoard returns true from the CreateThread() function to signify the Thread has been posted to the system. The Thread object is destroyed.

Figure 4.3 User Thread Creation Diagram

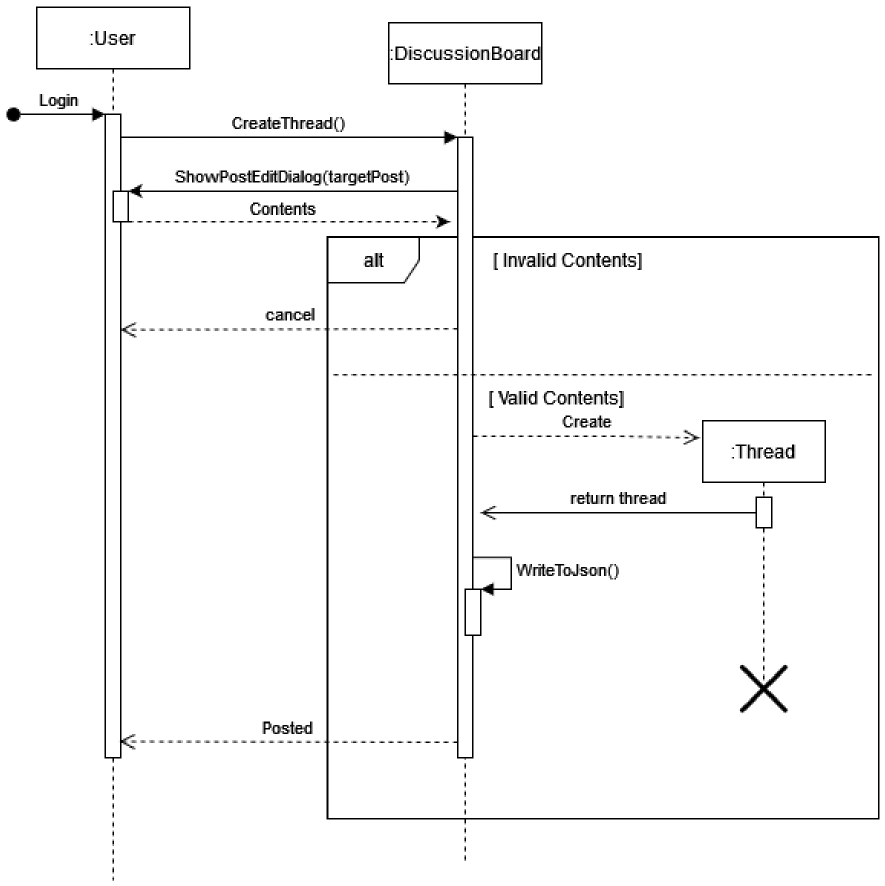


Fig 4.3 models the object interactions when a student creates a new discussion

Once the instructor has logged into the system, they click on a post to view it, calling the ViewPost() method which then returns true if the post displays to the screen. Once on the post the instructor then endorses or unendorses a post by clicking on the endorse button, this calls the ToggleEndorseStatus() method. Then the ExtractFromJSON() method is called which grabs the post to be changed from the JSON file. It then calls the FlipEndorsed() method to flip the endorsed status of the post to the opposite state and returns the state. Finally the method WriteToJson() is called to update the post in the JSON file, and true is returned back to the instructor from the ToggleEndorseStatus() method to signify the state was changed. The post object is destroyed.

Figure 4.4 Instructor Post Endorsement Diagram

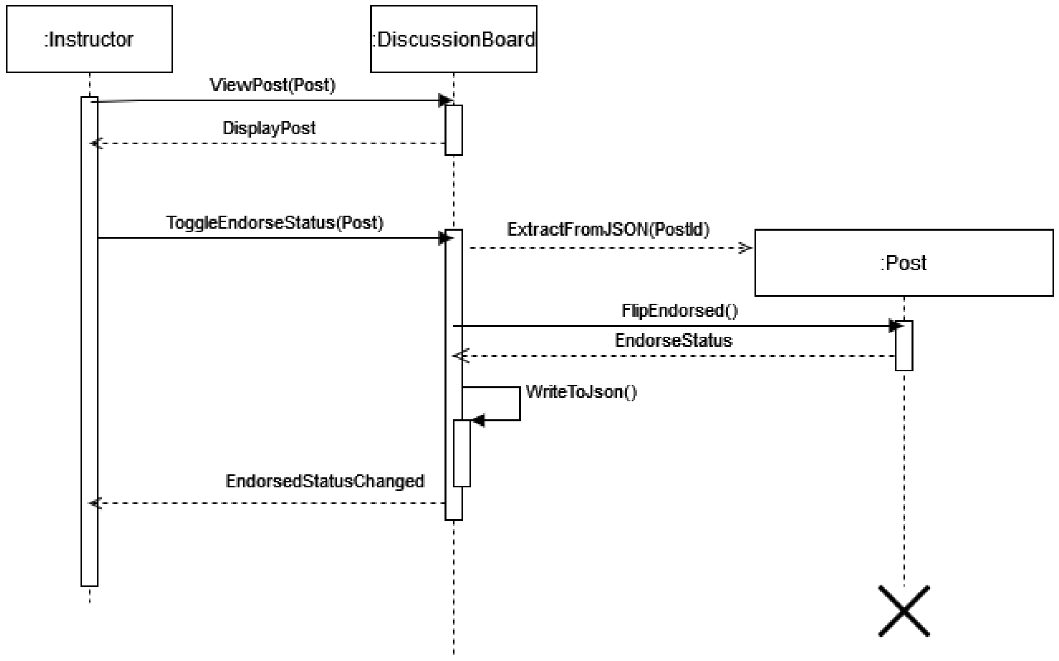


Fig 4.4 models the object interactions when an instructor endorses a given post.

Figure 4.5 State Diagram

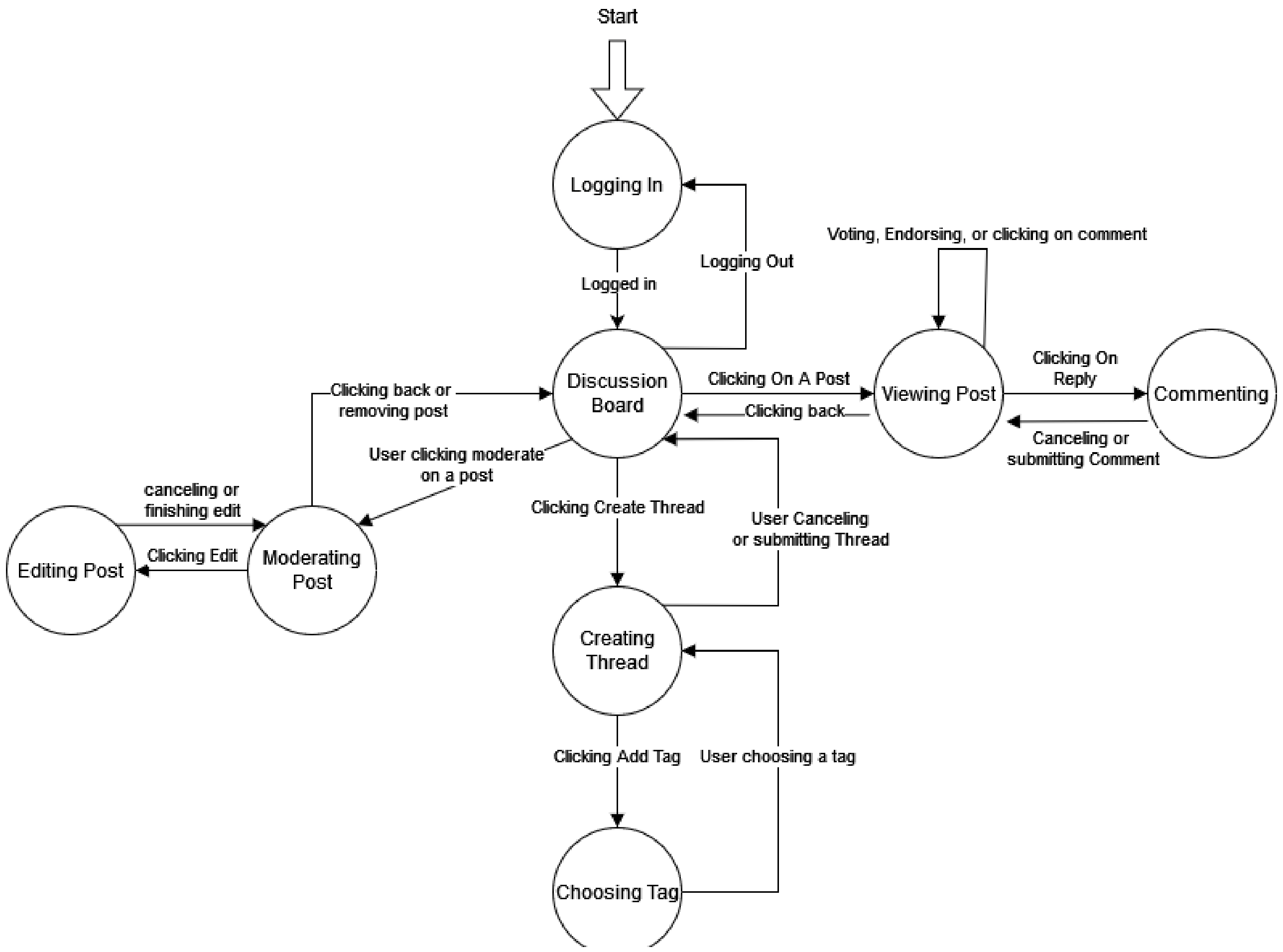


Fig 4.5 shows an overview of the workings of the discussion board and all its various states and how they change.

5 Prototype

The prototype will show a working message board. The board will include several clickable boxes, that either adjust displayed numerical values on posts or allow users to enter information in a text field for post creation. During thread creation there will be a menu to select relevant tags to categorize the thread. Users can edit their own posts after creation, while instructors can edit any post.

5.1 How to Run Prototype

System requirements:

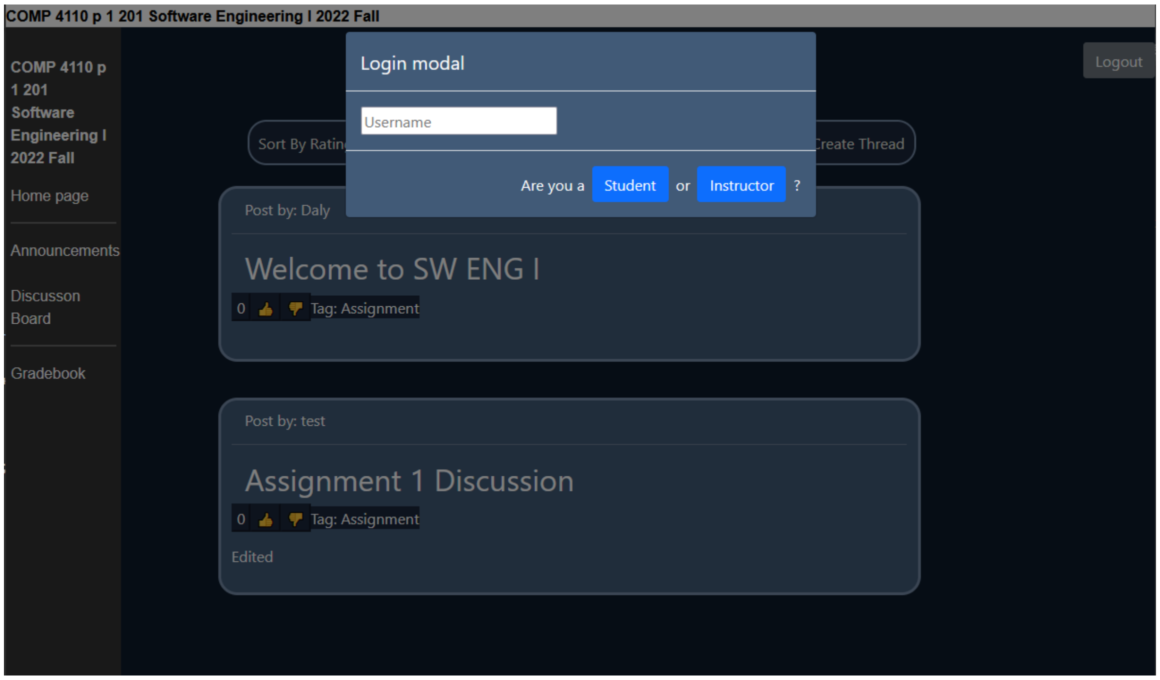
- NodeJS and npm (Node Package Manager) installed
- HTML5 and Javascript capable browser
- Keyboard and mouse

To run the prototype:

1. Clone the "Prototype-V2" branch here: https://github.com/Bryan0x05/software_eng_I_project/tree/prototypev2
2. Open the repository location in a terminal.
3. Ensure you have NodeJS installed by running the command:
   a. 'npm –version' (with two ticks - -)
4. If NodeJS is not installed, install from here and return to step 2:
   a. https://nodejs.org/en/
5. Download dependencies by running the command:
   a. 'npm i'
6. Run the server by running the command:
   a. 'npm run start'
7. Navigate to http://localhost:3000 in a web browser.
8. To stop the server, ctrl+c in the terminal you ran it from.

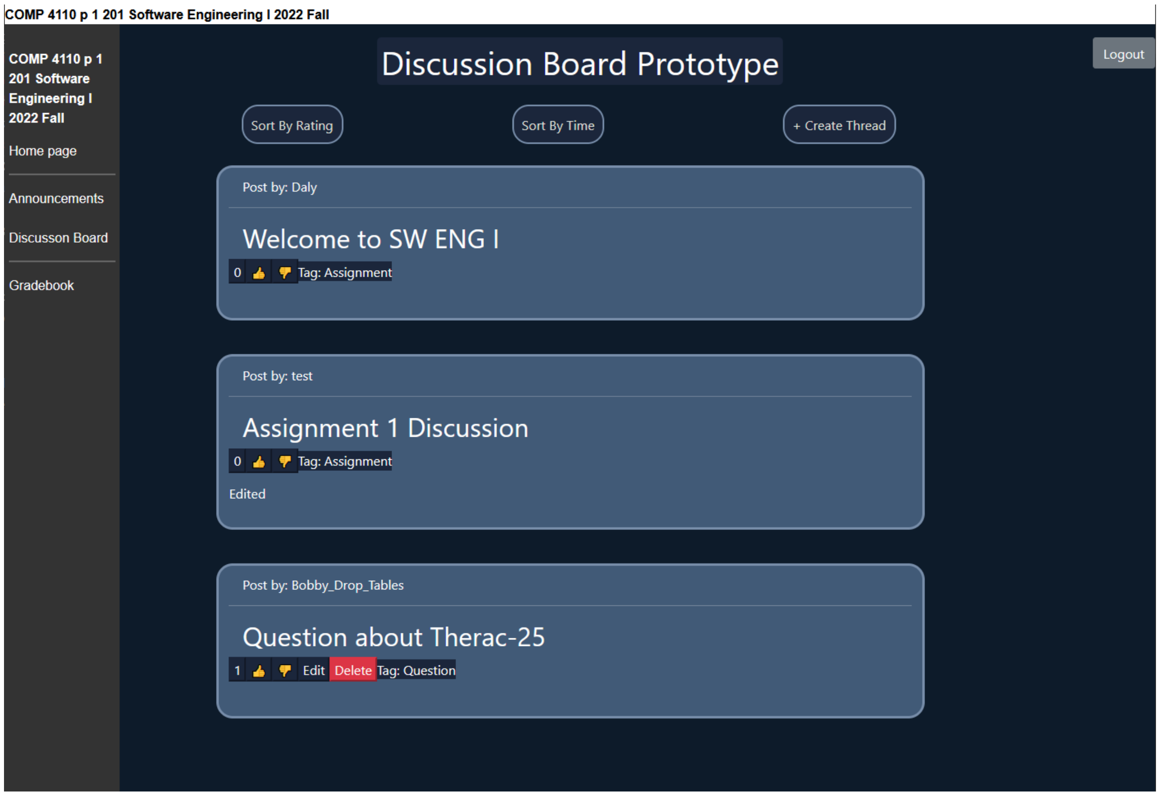
5.2 Sample Scenarios

Figure 5.2.1 Login Page Screenshot



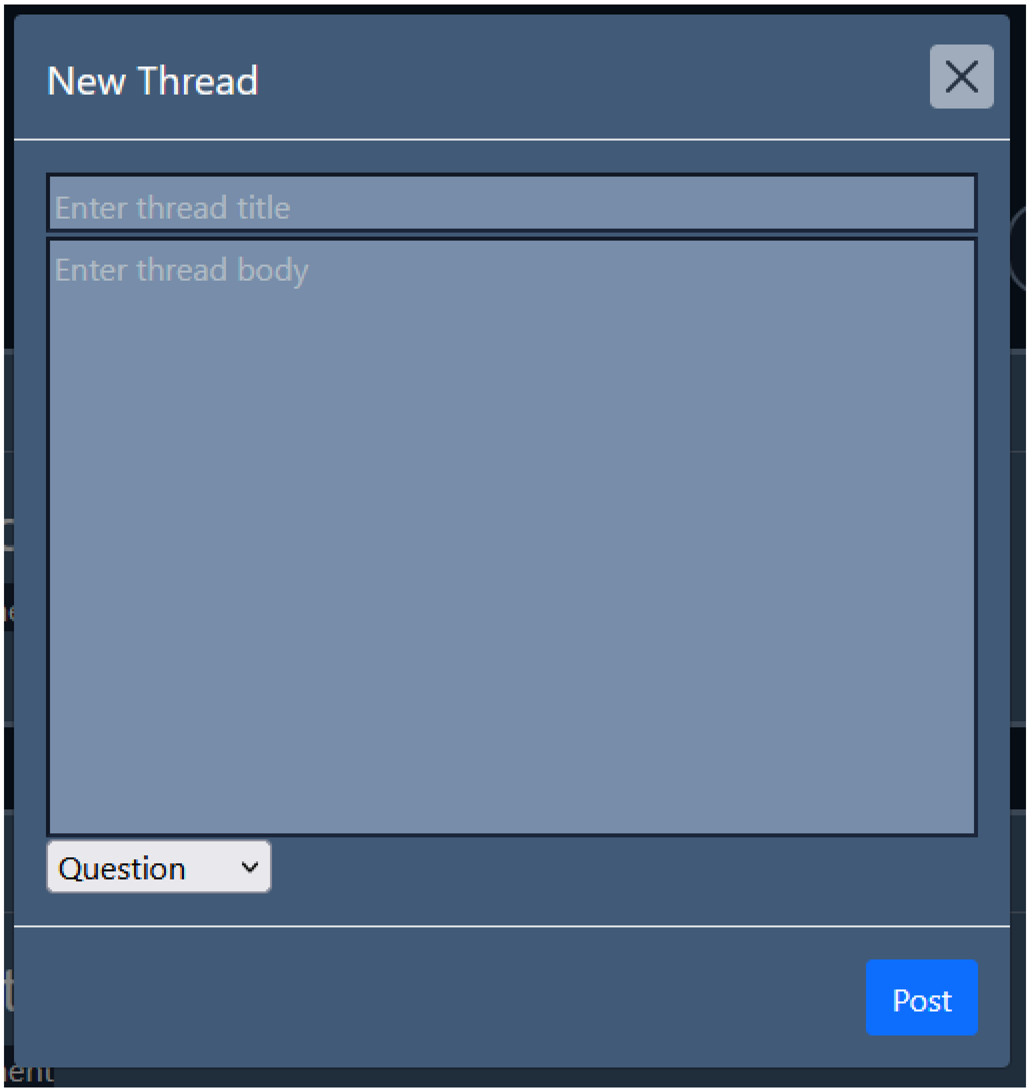
The user is prompted to login immediately upon loading the page. The user enters their name, and whether they are an instructor or student.

Figure 5.2.2 Discussion Board Homepage



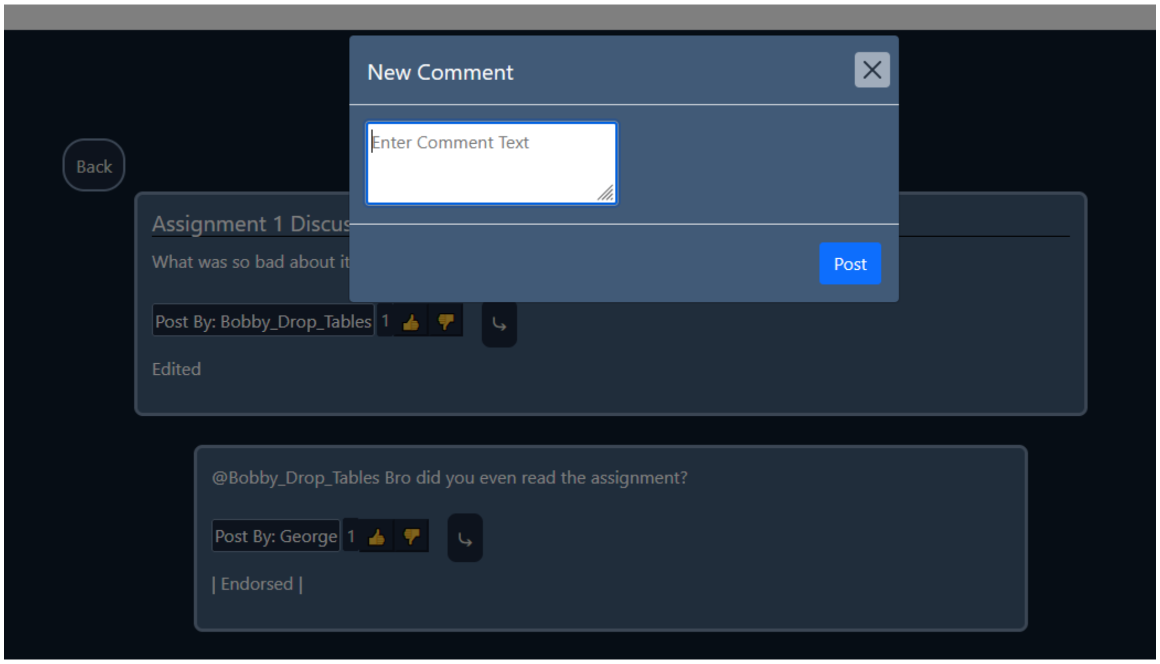
The Discussion Board Homepage displays all of the threads. Users can click "Sort By Rating" or "Sort By Time" to reorder the display of threads. Clicking on a thread will open it, showing its comments. Users can upvote, downvote, edit, or delete posts from the homepage if they have the permissions required.

Figure 5.2.3 Thread Creation Modal



After clicking the create thread button, the thread creation modal is displayed. The user inputs text defining the title, they can also select a tag. Clicking "Post" creates the new thread, adding it to the Discussion Board. If the user wants to cancel the creation, they can click the X button at the top, which brings them back to the main page.

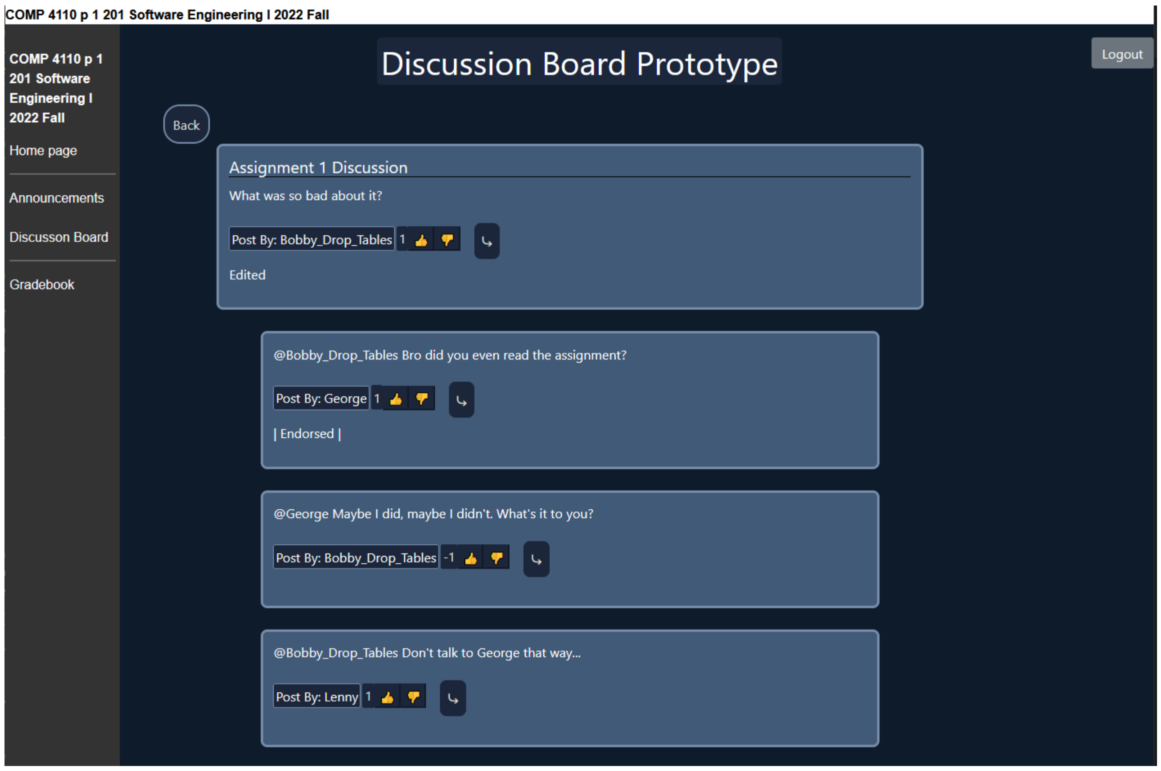
Figure 5.2.4 New Comment Modal



@Bobby_Drop_Tables Bro did you even read the assignment?

Clicking a reply button opens the ‘New Comment’ modal. The user enters their reply and clicks ‘Post’. The comment is appended to the thread.

Figure 5.2.5 Thread View



COMP 4110 p 1 201 Software Engineering I 2022 Fall

Discussion Board Prototype

Assignment 1 Discussion

What was so bad about it?

Post By: Bobby_Drop_Tables 1

Edited

@Bobby_Drop_Tables Bro did you even read the assignment?

Post By: George 1

| Endorsed |

@George Maybe I did, maybe I didn't. What's it to you?

Post By: Bobby_Drop_Tables -1

@Bobby_Drop_Tables Don't talk to George that way...

Post By: Lenny 1

Clicking on a thread displays the thread and its comments in 'Thread View',hiding all other threads. Users can reply to the thread and each other's comments. Instructors can endorse the thread or any of the comments listed. Users can upvote or downvote comments by clicking the 'thumbs up' and 'thumbs down' respectively.

6 References

[1] D. Thakore and S. Biswas, “Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks,” Proceedings of IEEE Military Communication, Atlantic City, October 2005.

[2] Scribner, Aidan, et al. “Discussion Board.” *Discussion Board Project*, 1.0, Team 5, 9 Nov. 2022, https://bryan0x05.github.io/software_eng_I_project/. Accessed 11 Nov. 2022.

[3] J. Thornton and M. Otto, “Get started with bootstrap,” *Bootstrap v5.2*. [Online]. Available: https://getbootstrap.com/docs/5.2/getting-started/introduction/. [Accessed: 16-Nov-2022].

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.