Software Requirements Specification (SRS)
Blackboard Notification System Revamp

Team: 3
Authors: Cam Benassi, Caleb Bergen, Tony Choma, Leonard Nguyen, Alvin Tran
Customer: Blackboard
Instructor: Dr. James Daly

1 Introduction

The Blackboard Notification System revamp project is detailed in this document. This document is divided into multiple sections and subsections which will provide insight for the project. These subsections include the introduction, the overall description, specific requirements modeling requirements, prototype, references, and contacts.

1.1 Purpose

This document serves as a detailed guide for Universities that show interest to explain what the new Blackboard Notification System is, important terminology to be used, the requirements set for development, and how to run the web app. The system will be a component of the Learning Management System intended for students and faculty.

1.2 Scope

The product that is to be produced is a revamp to the Blackboard Notification System. Currently, Blackboard's notification system doesn't have a pleasant UI and lacks a concise way to filter notifications. There are numerous repeat filters that leave the user confused on what to select. The purpose of the new Blackboard Notification System is to make reading notifications a much smoother and less confusing experience for all users. The notification system will allow users to view new notifications and easily filter them. The student user will be able to mark notifications as read and clear notifications off of the page. Only the professor should be able to create notifications. Notification forms will be provided to the professor for them to fill out.

1.3 Definitions, acronyms, and abbreviations

- Learning Management System: a system that provides a platform for students and faculty to manage assignments and messages between one another
- LMS: Learning Management System
- Students: users that use the Student section of an LMS
- Faculty: users that use the Teacher version of Blackboard. Teachers, educators, and professors can be used interchangeably to describe the faculty
- Blackboard: an LMS that the University of Massachusetts Lowell uses
- ReactJS: front-end JavaScript library for building user interfaces based on UI components
- TailwindCSS: open-source utility-first CSS framework
- NodeJS: server environment for website backend
- ExpressJS: backend web app framework that is to be used with NodeJS

1.4 Organization

This document contains some detailed blocks outlining each use case utilized in the use case diagram. Each block contains a set of information about the use case, its relation to both the product, and its role in the use case diagram.

Section 1, acts as a guide for the reader to better understand the entirety of the app and this document. The purpose of the app is outlined, as well as the intended audience, students and faculty. The scope describes the different softwares used throughout the development of the application. The important definitions, acronyms, and abbreviations are also used for the sake of making this document understandable. The organization describes everything else involved.

Section 2, the Overall Description, includes the main details of the application. It provides an overview of the section's information, the product perspective, its functionality, characteristics, constraints, dependencies, and how the requirements are divided. The product perspective illustrates its part of the overall system it is intended for. The product functions section outlines the main functionalities of the product. It also provides diagrams that depict the high level functions that the user would generally see. The user characteristics section describes what we expect of our users. This in turn helps outline the constraints of both the program and the user. Assumptions and Dependencies states the assumptions about hardware and software that can be made about the product and lists the different dependencies used. This all segways into the division of requirements in the next few sections.

Section 3 details the Specific Requirements for everything pertaining to the product's software, functionality, design, and user experience. The software section lists what's used for the frontend and backend development as well as what's used for version control. The design and functionality sections describe how the look of the UI goes hand in hand with how the product works. This also outlines each function necessary for a proper notification system. The user experience requirements explain how the user should experience the product.

Section 4 is the Modeling Requirements, which displays all the different diagrams that describe the functionality of the app. The use case diagram outlines the goals enacted between actors and actions performed. The class diagram depicts the properties of each potential classification of components and associates them in a hierarchy. The sequence diagrams reviews a life cycle of two different scenarios in the program. The state diagram is a display of certain events or states depending on the product's conditions at any given time.

Section 5 is a display of our prototype. This will include a brief explanation of the prototype's functionality as well as some screenshots of examples and instructions for how to run it. The prototype itself is also available through the web.

Section 6 includes references used for the project. This includes documents, sources, and an entry to our website.

Section 7 provides a point of contact regarding the product.

2 Overall Description

Here our product is covered and its purpose. This includes what functionality is planned to be available at launch, the intended audience for our product, and any limitations. The overall goal of the project is to increase accessibility of the Blackboard notifications system. Our product attempts to achieve this by increasing the usability and readability of the notification page on Blackboard. Finally, this section contains information on additional features that could be implemented in the future after the product has launched.

2.1 Product Perspective

Blackboard's notification system is in need of a revamp. The system is confusing, includes many unclear and cluttered design elements, and generally presents too much information to users at once. Our team intends to design a product that addresses these issues with a redesign of how notifications are displayed to the user. This product will be a modification of a small part of the Blackboard LMS as a whole. Our product aims to give the user more control over what notifications they see on screen and to enhance the GUI design for information clarity. As opposed to the current system, which only allows users to filter by class, our design will implement three new filtering options: filter by class, which allows users to view notifications for their individual classes; filter by date which sorts notifications by date; and filter by priority, which sorts notifications by their priority level. Two additional features are planned for implementation as well, a view unread only button which shows users their unread notifications and a search feature to allow users to find specific notifications through a text search.

New notification cards will be implemented as well with more padding between information, with clearly marked notifications boxes which will improve readability and information clarity. We will also give the user the option to mark a notification as read, and view the notification directly from the class page via a hyperlink.

The user will be able to interact with this product on their computer or a phone capable of running the Blackboard website. This product utilizes the React and Tailwind JS libraries for our interface design and scripting. Any user who wishes to use our

changes must have a device capable of running a modern web browser, as well an internet connection for the use of the Blackboard platform.

2.2 Product Functions

The major functionalities of our team's revamped notification system are as follows:

Notification Filtering:

- Filter by class: sort the user's current notifications by the associated classes
- Filter by date: sort the user's current notifications by date
- Filter by priority: sorts the user's current notifications by priority level
- Filter by category: sorts the user's current notifications by category
- View unread only: exclusively displays the user's unread notifications
- Search: allows the user to search through their notifications via a text search

Notification card:

- Mark as read: marks a notification as being read
- View button: takes the user directly to the assignment, class, or announcement when clicked

Educator functionality:

- Add notification: allows the professor of a class to send out a notification to the users

2.3 User Characteristics

The expected users of our revamped notification system are university students who use Blackboard for their classes, and educators who push notifications out to their students. Our aim is to improve accessibility by improving information clarity for highschoolers up to college graduate students. The only required skills on the student side will be using Blackboard, general internet and computer literacy. For the professors using our product, the required skills needed are no different than what is already required in the current system. Educators will need to be able to add notifications to their classes.

2.4 Constraints

Our product is constrained by the already existing limitations of the Blackboard LMS. The whole LMS as it stands already has its own student and teacher modes with its own implementation. The LMS may also have their own set of data that we may need to adjust to in the case that we do integrate our system with the official Blackboard SRS. Blackboard even has a case where it'll log the user off after a certain amount of time. These constraints somewhat limit both the users and developers from handling our software freely.

2.5 Assumptions and Dependencies

It is assumed that the hardware being used for our product meets the requirements for a modern web browser such as Chrome or Firefox. In addition, the availability of an internet connection of any speed is required as well. Our product also assumes the user has knowledge of how to use the software required for Blackboard such as a web browser. Additionally the user should know how to use the current Blackboard LMS.

2.6 Apportioning of Requirements

As for the initial release, our product plans to revamp the existing Blackboard notifications page with improved UI and notification filtering capabilities. Features that would be considered in future implementations are as follows:

- The ability to notify the user of a new notification from anywhere on the website with a pop up
- Additional notification sorting functionality
- A quick view notification summary from outside of the notifications page
- Mobile push notification support

3 Specific Requirements

1. Software Requirements
    - 1.1. The frontend portion of the software should be built using a combination of an open source Javascript web framework and a CSS framework.
        - 1.1.1. The Javascript framework should be utilized for building interactive user interfaces and components such as the home page UI, notification filters, and notification cards
        - 1.1.2. The CSS framework should be used to style our application with more ease by allowing the use of prebuilt CSS instead of building new CSS from scratch. This cuts production time and is easy to use
    - 1.2. The backend portion of our software should be built using a combination of an open source runtime environment and web application framework.
        - 1.2.1. The runtime environment should be utilized for server-side programming by handling client requests and events.
        - 1.2.2. The web application framework will be used for server routing by controlling what webpage endpoint to display.
    - 1.3. The software should be hosted on a cloud application platform.
    - 1.4. Version control software should be utilized.
    - 1.5. A repository will be utilized to host and assist in codebase sharing.
2. Functionality Requirements
    - 2.1. Notifications will be displayed on the main page.
        - 2.1.1. Each notification will have basic notification data consisting of the course name where the notification is coming from, the notification header, the type of notification, the notification message, and the date it was uploaded.
            - 2.1.1.1. Types of notification are general announcement, important announcement, assignment, and project.
            - 2.1.1.2. Notification pop-ups will also adjust to different priority levels and types
        - 2.1.2. By default, unread notifications will be presented in descending order by date.
    - 2.2. There will be a breakdown of the count of notifications by course at the top of the page.
    - 2.3. Unread notifications can be changed to ‘Read’ status when the user acknowledges the notification
    - 2.4. Notifications may be filtered by course, type, specific dates, priority, unread, and read.
    - 2.5. There will be distinguishable styling for different types of notifications.

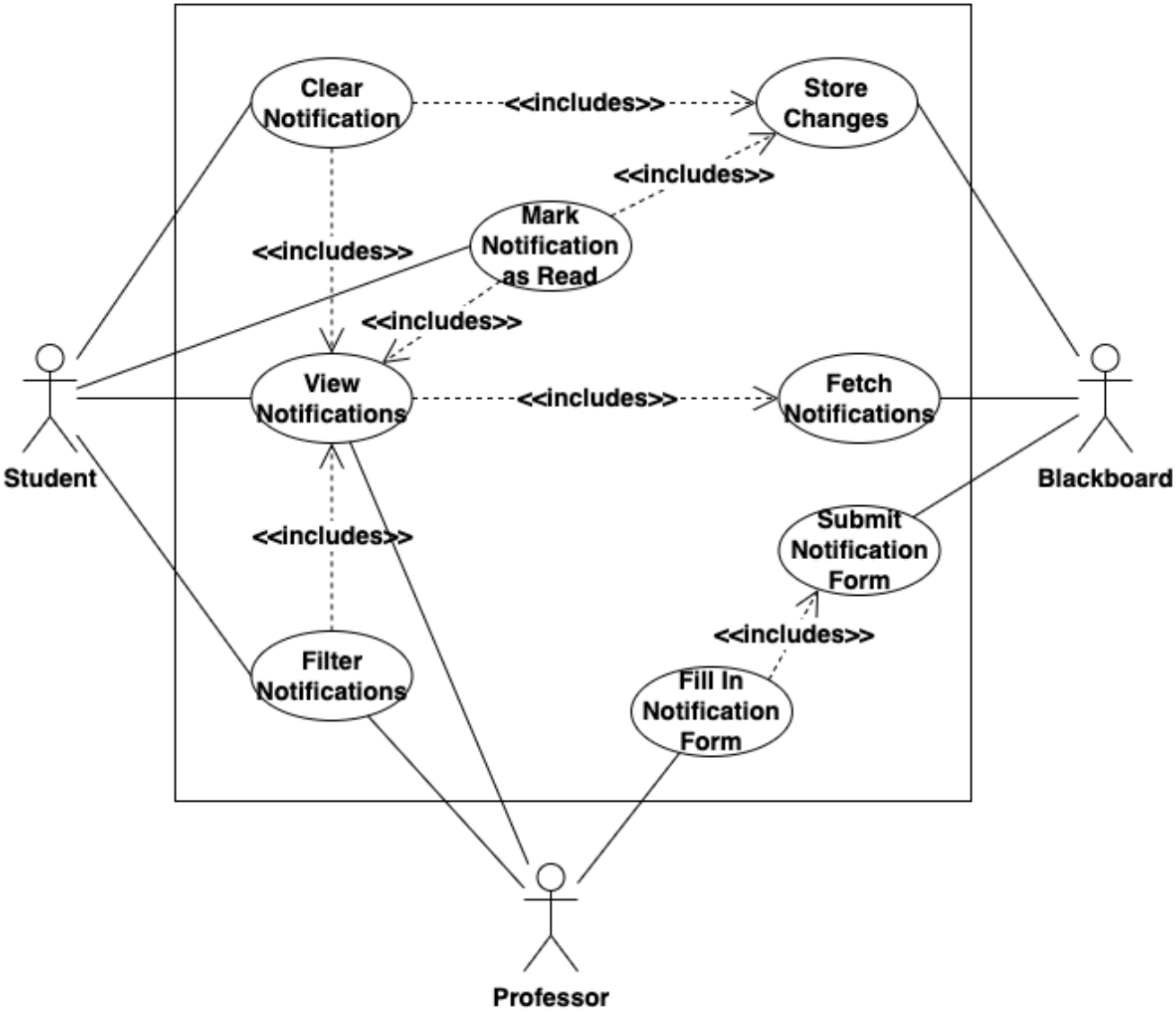
2.6. Notifications can only be viewed if the user is part of the course.
2.7. Notifications will be created by professors.
2.7.1. A fillable form will be provided to the professor so all the professor needs to do is fill in the notification information.
2.7.2. Professors will be able to attach a file.

3. User Experience Requirements
3.1. The notification system should be practical.
3.1.1. Reading notifications and using tools should be self-explanatory to the user
3.2. The notification system should be reliable with no glitches that hinder user experience and UI elements shouldn't cause lag
3.2.1. Components of the system should make sense and placements of the components should be natural as if the user expects a certain component to be in a certain location.
3.3. The notification system should be flexible.
3.3.1. Notifications will be readable and usable at any resolution size.
3.3.1.1. The webpage will scale depending on the resolution size of the device being used to view the notifications.
3.3.1.2. Users should still be able to find the notifications and understand what is happening on the page.
3.3.1.3. Users should be able to alter and interact with the notifications.
3.3.1.4. The notification popups should be 1-2 seconds short and not take up the whole screen.

4 Modeling Requirements

Use Case Diagram

Below is a use case diagram depicting the new Blackboard Notification System. There are three actors which include students, professors and Blackboard itself. Both the student and the professor are able to filter and view notifications. However, students are able to clear notifications or mark them as read while professors can't do those actions. Professors can fill out a notification form that gets submitted to Blackboard. Blackboard store changes, and send notifications to students when they view their notifications for the day. Blackboard also takes in the submitted notification form from the professor and stores it in its database.



Blackboard Notification System Use Case Diagram

Use Case Name:	View Notifications
Actors:	Student (initiator), Professor (initiator)
Description:	The home page of the Blackboard Notification System. A list of notifications will be displayed on the page.
Type:	Primary and Essential
Includes:	Fetch Notifications
Extends:	None
Cross-refs:	None
Uses cases:	The notifications must be fetched from Blackboard first.

Use Case Name:	Fetch Notifications
Actors:	Blackboard
Description:	Blackboard system will return a list of notifications back to the user when requested.
Type:	Secondary and Essential
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	To send notifications to the users view

Use Case Name:	Clear Notifications
Actors:	Student (Initiator)
Description:	Notifications will be removed from the page.
Type:	Primary
Includes:	View Notifications, Store Changes
Extends:	None
Cross-refs:	None
Uses cases:	The notifications must be able to be viewed first.

Use Case Name:	Mark Notification as Read

Actors:	Student (Initiator)
Description:	Notifications that are ‘unread’ can have their status changed to ‘read’.
Type:	Secondary
Includes:	View Notifications, Store Changes
Extends:	None
Cross-refs:	None
Uses cases:	The notifications must be able to be viewed first.

Use Case Name:	Store Changes
Actors:	Blackboard (initiator)
Description:	Changes to the notifications will be stored into Blackboard.
Type:	Secondary and Essential
Includes:	None
Extends:	None
Cross-refs:	None
Uses cases:	A change to the notifications must be made first such as clearing the notifications, or marking them as read.

Use Case Name:	Filter Notifications
Actors:	Student (Initiator), Professor (Initiator)
Description:	Notifications can be filtered by course, type, specific dates, priority, unread, and read.
Type:	Primary and Essential
Includes:	View Notifications
Extends:	None
Cross-refs:	None
Uses cases:	The notifications must be able to be viewed first.

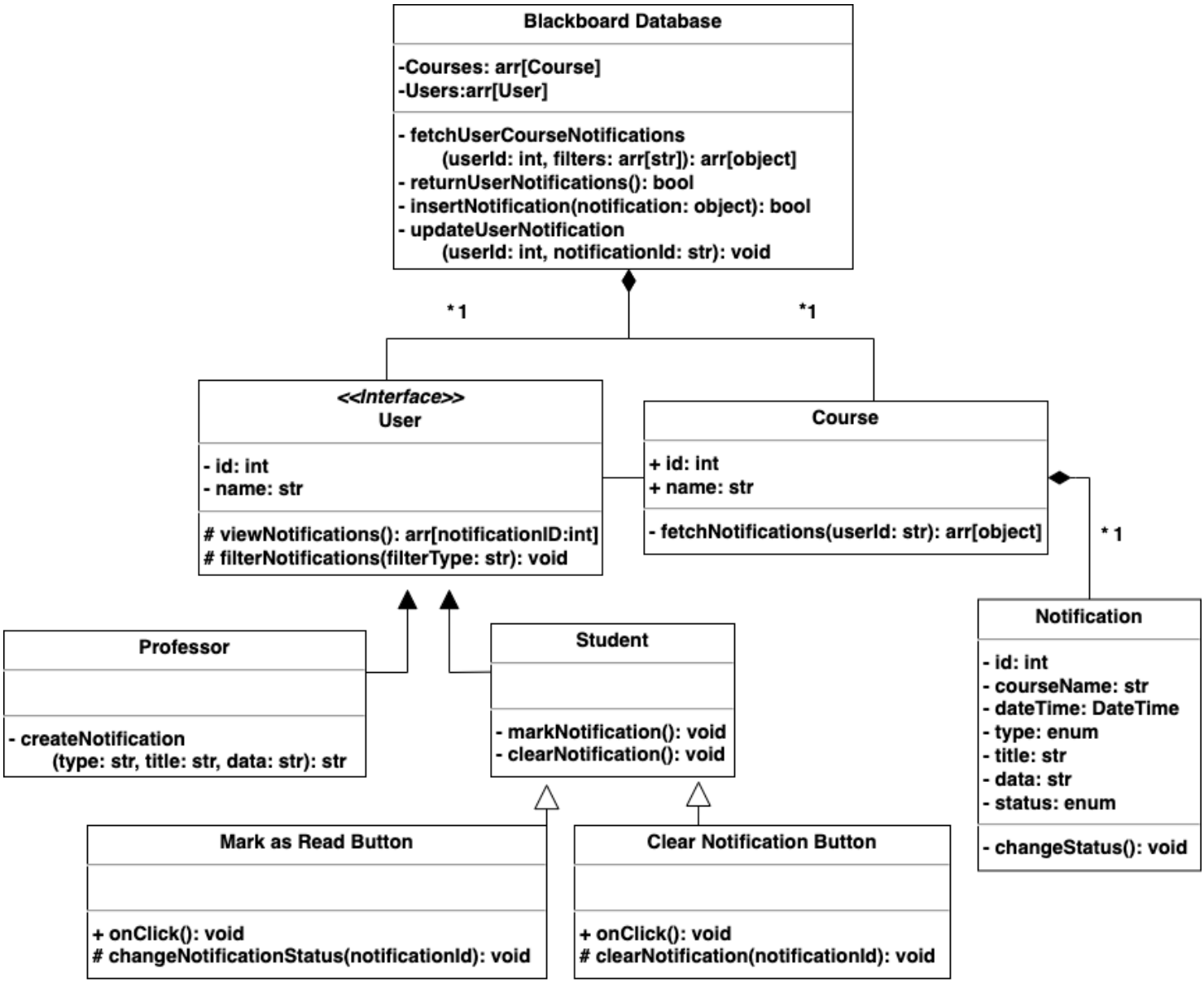
Use Case Name:	Fill in Notification Form
Actors:	Professor (Initiator)

Description:	Professor will create notifications by filling in a notification form with information.
Type:	Primary and Essential
Includes:	None
Extends:	Cancel Notification Form
Cross-refs:	None
Uses cases:	The professor can cancel the creation of the notification.

Use Case Name:	Submit Notification Form
Actors:	Blackboard (Initiator)
Description:	The professor will submit the notification to Blackboard and it will save that notification into the system.
Type:	Primary and Essential
Includes:	Fill in Notification Form
Extends:	None
Cross-refs:	None
Uses cases:	Before submitting the notification form, the notification form must be filled out first.

Class Diagram

Below is a class diagram depicting the Blackboard Notification System revamp. It first consists of the Blackboard database that holds all of the courses and users in the current university. The database class can fetch user course notification, insert notification, return user notifications, and update user notifications. Users consist of professors or students. Students have the ability to mark a notification as 'read' or clear notification with buttons. Professors have the ability to create notifications for each course which will in turn be sent to students. Each course has its own number of notifications that include attributes shown in the diagram.



Class Diagram

A data dictionary is provided to give more explanations to the attributes, methods, and connections to the classes in the class diagram.

Element Name		**Description**
Blackboard Database		The Blackboard Database contains all of the information in our software. The database will consist of Users and Courses. The notifications will be contained in the courses.
Attributes		
	Courses:arr[Course]	Courses in database
	Users:arr[User]	Users in database
Operations		
	fetchUserCourseNotifications (userId: str, filters: arr[str]): arr[object]	The database will fetch the user's course notifications while basing it off of filters. An array of notification objects will be returned to the database.
	returnUserNotifications(): bool	The database will return the fetched course notifications back to the user. A boolean object will be returned to the database to tell if the operation was successful or not.
	insertNotification(notification : object): bool	The database will insert a new notification object to the database. A boolean object will be returned to the database to tell if the operation was successful or not.
	updateUserNotification(userI d: str, notificationId: str): void	The database will update the status of a notification for a user.
Relationships	There can only be one Blackboard database and its relationship to other classes is that the database will contain many user classes and many course classes.	
UML Extensions	None	

Element Name		**Description**
Course		The courses at the educational institution that professors teach and students register in.
Attributes		
	id: int	The course id.
	name: string	The name of the course.
Operations		
	fetchNotifications(userId: int): arr[object]	
Relationships	Has a relationship with Blackboard Database where there are many courses to the database at the educational institution. Has close relation with class, user, where the users are professors or students of the class. Every course will have their own notifications that belong to the courses.	
UML Extensions	None	

Element Name		**Description**
Clear Notification Button		This is a clickable button that will be used by the student to clear any notifications off the screen.
Attributes		
	None	
Operations		
	onClick(): void	Clickable button.
	clearNotification(notificationId): void	When clicked, the notification will be cleared off of the screen.
Relationships	Has a relationship with student class. Only the student class should	

	be able to clear notifications.
UML Extensions	None

Element Name		**Description**
Mark as Read Button		This is a clickable button that will mark a notification as read and will change its status from unread to read.
Attributes		
	None	
Operations		
	onClick(): void	Clickable button.
	changeNotificationStatus(notificationId): void	Will mark down the notification as read and change the notification's status.
Relationships	Has a relationship with student class. The student will be the one receiving the notifications and it will appear as unread to the student so only the student class should have this class.	
UML Extensions	None	

Element Name		**Description**
Notification		The announcements that will be created by the professor and which will be read by the student.
Attributes		
	id: int	Notification id.
	courseName: str	Course name the notification belongs to.
	dateTime: DateTime	The date and time the notification

		was created.
	type: enum	The type of notification such as general announcement,
	title: str	The title of the notification will be what the notification is about.
	data: str	The message and content of the notification.
	status: enum	The status of the notification. The status can be ‘unread’ or ‘read’.
Operations		
	changeStatus(): void	The status of the notification will be changed based on the user’s request if the ‘Mark as Read’ button has been clicked.
Relationships	Has a relationship with courses. These notifications belong to the courses, and from there, the notifications can be provided to the user.	
UML Extensions	None	

Element Name		**Description**
Professor		The one who teaches the course.
Attributes		
	None	
Operations		
	createNotification(type: str, title: str, data: str): str	The professor will be able to create notifications by filling in a form with information about the notification’s type, title, and the data or content of the notification.
Relationships	The professor is a subclass to the user class.	
UML Extensions	None	

Element Name		**Description**
Student		The one who learns in a course.
Attributes		
	None	
Operations		
	markNotification(): void	The student will be able to mark notifications as unread to read.
	clearNotification(): void	The student will be able to clear notifications off of the screen.
Relationships	The student is a subclass to the user class.	
UML Extensions	None	

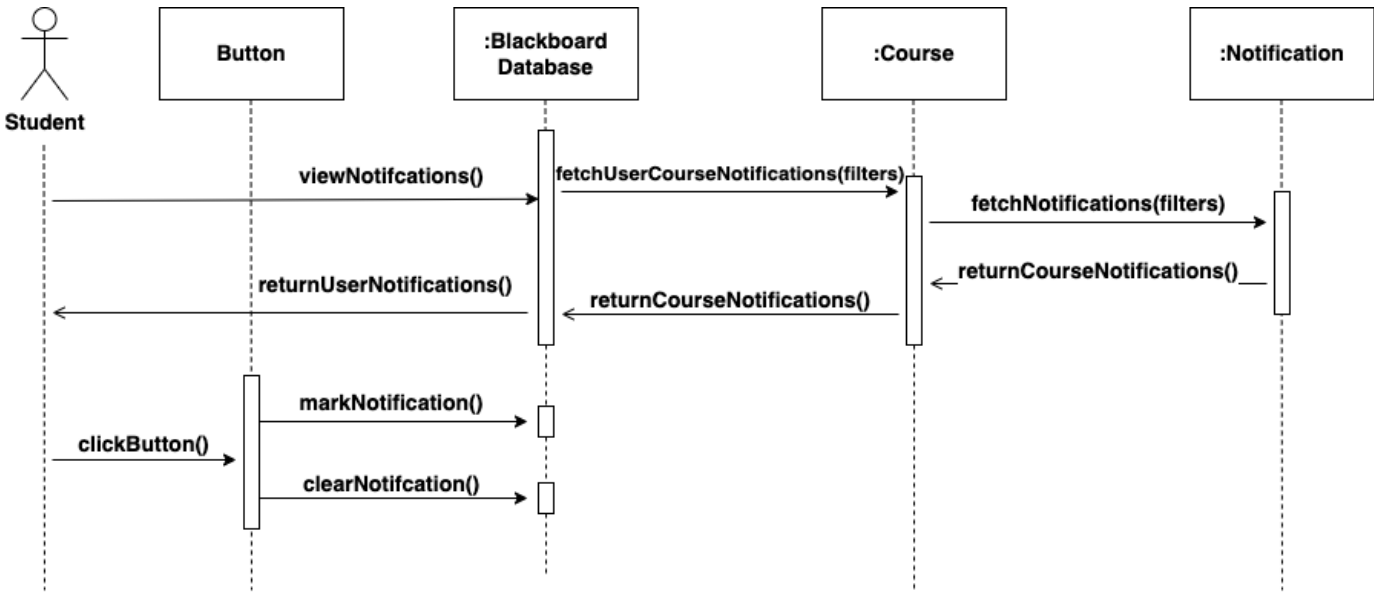
Element Name		**Description**
User		The general user of the Blackboard notification system.
Attributes		
	id: int	The user's id.
	name: str	The user's full name.
Operations		
	viewNotifications(): arr[notificationId:int]	The user will be able to view notifications. An array of notification objects will be returned to the user.
	filterNotifications(filterType: str): void	The user will be able to filter the notifications to some extent.
Relationships	Has a relationship to the Blackboard Database class where there are many users to the Blackboard Database. There is a connection from the user to the courses where many users are part of many courses.	

UML Extensions	None

Representative Scenarios of System and Sequence diagrams.

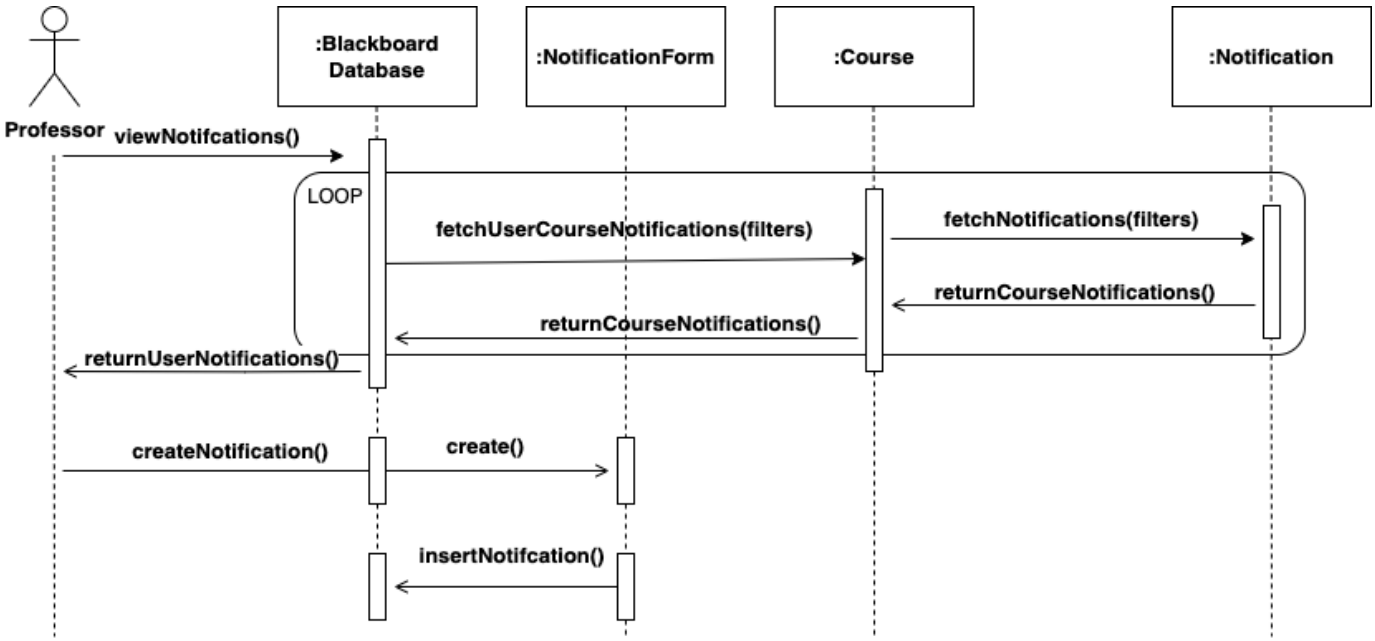
There will be two general representative scenarios of the system. One scenario will be based off of a student user, and the other scenario will be based off of a professor user. For each scenario, there will be sequence diagrams provided that will demonstrate certain parts of the scenario. **Sequence diagrams** involve roles, lifelines, activation bars, messages, and frames. Roles are instances in a system that can be from the software or be an exterior person. Lifelines are dashed lines coming below the roles that represent the duration of an instances' life. Activation bars are the duration in which the roles perform a task. Messages are displayed with arrows depicting method calls. Frames are sectioned off portions of the system where a smaller process is happening.

For the student user scenario, let say that the student is already logged into Blackboard and noticed that the notifications button is highlighted. The student will click on the notifications button and will be brought to the Blackboard notification page. The student will read through all notifications that have the unread status and will mark the notifications as read. The student remembers that there was an important announcement from the week prior and decides to filter through the notifications to search for it. The student will filter the notifications using the course, and the time period the notification will be sent to find it. A new list of notifications pops up on the screen displaying the notifications that are sent from that time period and are from that class. After reading through the new list of notifications, the student was able to reread the important notification. After reading the notifications, the student continues on with using the other functions of Blackboard.



Student Sequence Diagram

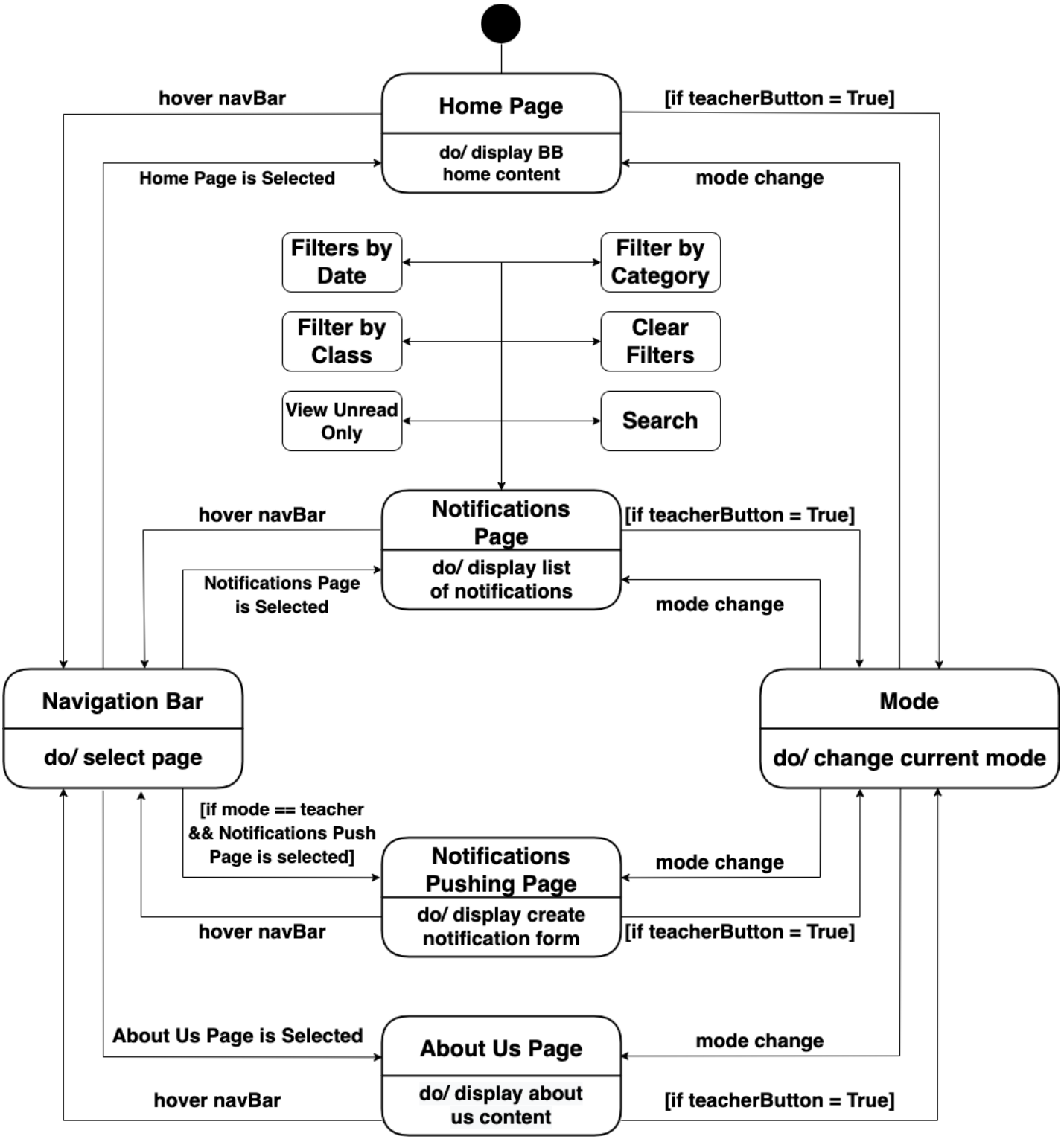
For the professor user scenario, let's say that the professor is already logged into Blackboard and wants to create a new notification. The professor will be given a switch that enables notification pushing in the dropdown menu. They will then click on the notifications button and will be brought to the Blackboard notification page. The professor will then get brought to a form. The form will consist of spots where the professor will need to fill out information for the course they want to send the notification to, the type of notification, the title of the notification, and the notification content. After the professor fills out the notification form, they submit it and the notification is sent out to everyone in the course. The professor is brought back to the Blackboard notification page and the new notification that the professor just wrote will appear as the latest notification. After creating the notification, the professor continues on with using the other functions of Blackboard.



Professor Sequence Diagram

State Diagram

Below is a state diagram depicting the improved Blackboard Notification System. The user starts at the home page of the website and is given a few options. They are able to change website mode from student to teacher mode. They are also able to select a page from the navigation bar. From the navigation bar, the user is able to choose to navigate to the home, notifications, about us, and notifications pushing pages. However the notifications pushing page must have the teacher mode enabled. Otherwise the notification pushing page is not shown and navigable. In the notifications page, the user is able to filter by date, category, and class. They are also able to search, view unread only, and search for notifications as well.



State Diagram for LMS

5 Prototype

The prototype shall display a sample view from the student's point of view. On the top, there is a navbar to access other areas of the webpage. In a production environment, this would have linked to Blackboard's other areas, as they have now. For the sake of this project, however, there are links to other aspects of the project. To the left, there is a sidebar, containing tools for users to sort and filter the notifications present on the screen. These options include filtering by class, date, priority, category, unread notifications, or just a general text search. The remainder of the screen is filled by notification cells. These cells all include the same basic information about a notification, title, class, category, and due date (if applicable). Each cell also contains two buttons, a 'Mark as Read' button and a 'View' button. The first one is to mark the notification as read, and the second one is to view the notification in the class's announcement page.

5.1 How to Run Prototype

Prototype V1 URL: https://swelms.herokuapp.com/prototype

This prototype can be run on any OS. The only software required to run it is a terminal window, npm, and the latest version of NodeJS. To run the prototype, first download the codebase by cloning the repository. Navigate to the root folder of the repository and run the command 'npm start --prefix client". This will run 'npm install' on both the client and server folders, installing the necessary packages through npm. The script will then run npm start on the client folder, running the application.

5.2 Sample Scenarios

Student:

Students will use this tool to stay in touch with announcements from their professors. For example, a student may log into Blackboard, then see that they have a new notification. They click on the notification page to see that there is a new, unread notification waiting for them. An example of a notification may be something along the lines of 'Exam 1 has been graded'. Seeing this, the student can click 'Mark as Read' to dismiss the notification and alert from the original homepage they came from, then go to their class page to check out their grade.

Teacher:

Teachers can use this platform to communicate and send push notifications to their students. An example of this could be if students were handed out an assignment to complete but a question had a typo on it that needed to be corrected. A teacher could send out a notification, alerting the students of the typo and how to correct it. Since students would not have to wait until the next meeting to hear about this change, they would have more time to work on the assignment.

6 References

[1] C. Benassi, C. Bergen, T. Choma, L. Nguyen, A. Tran, "Blackboard Notification System Revamp" (2022). https://swelms.herokuapp.com/

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

[3] "Documentation - tailwind CSS," *Documentation - Tailwind CSS*. [Online]. Available: https://v2.tailwindcss.com/docs.

[4] P. Nasilele, "How to create a search bar in react," *a*, 02-Aug-2022. [Online]. Available: https://plainenglish.io/blog/how-to-implement-a-search-bar-in-react-js.

[5] "Tailwind CSS search form - free examples & tutorial," *Tailwind Elements*. [Online]. Available: https://tailwind-elements.com/docs/standard/forms/search/.

[6] "W3schools how to," *W3Schools How TO - Code snippets for HTML, CSS and JavaScript*. [Online]. Available: https://www.w3schools.com/howto/default.asp.

[7] "Tailwind UI - official tailwind CSS components & templates," *Tailwind UI - Official Tailwind CSS Components & Templates*. [Online]. Available: https://tailwindui.com/.

[8] "How to do simple form validation in #Reactjs," *Learnetto*. [Online]. Available: https://learnetto.com/blog/react-form-validation.

[9] "Array.prototype.sort() - javascript: MDN," *JavaScript | MDN*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort.

[10] "Dropdowns," *Official Tailwind CSS UI Components*. [Online]. Available: https://tailwindui.com/components/application-ui/elements/dropdowns.

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.