PurdueHub - Project Design Document - Team 31

Aditya Gupta, Leonardo Delgado, Hoang Nguyen, Qasim Ali

Purpose:

As students of Purdue University, we have encountered many frustrations with current Purdue systems as they seem dated. While they are valuable tools we use daily in our academic lives, some feel clunky, unresponsive, and not befitting of a major university. We are aiming to include as many existing Purdue existing services along with a platform for Purdue students to speak with each other, serving as a central hub for all Purdue information students can use. With our project being maintained by fellow students, we can easily engage in discussion about our application and further refine it with student feedback.

There are services like Discord and Edstem, but they don't involve existing Purdue services within it, rather simply as a discussion app for other students to communicate on it. We want student discussion, but more in a formal sense while also providing useful Purdue tools to serve as the go-to for students to use.

Functional Requirements:

A) User Account:

As a user,

1) I would like to be able to register for a PurdueHub account
2) I would like to be able to register for a PurdueHub account using Google?
3) I would like to be able to register for a PurdueHub account using Facebook?
4) I would like to be able to register for a PurdueHub account using email?

5) I would like to be able to register for a PurdueHub account using an official Purdue email?
6) I would like to be able to manage my PurdueHub after logging in.
7) I would like to be able to reset my password in case of forgetting it
8) I want to be able to delete my account from PurdueHub with action confirmation.

B) User Profile:

As a user,

a) I would like to be able to create my own username, profile picture, and bio
b) I would like to be able to modify my own username, profile picture, and bio
c) I would like to choose my own profile picture from my pc documents (web browser + mobile)
d) I would like to be able to turn on and off notifications for calendar events and other pieces of information such as clubs I am interested or general information like the IR career fair
e) If my account gets banned or deleted, I would like to get an email notification or a notification when I try to sign in.

C) Messaging:

As a user,

a) I would like to be able to send and receive messages from others (assuming they are not blocked)
b) I would like to be able to block and unblock other students
c) I would like a user-friendly search that displays closely related usernames to my search request field (if time allows).

d) I would like to be able to delete chats or messages from the direct messages chatting system (if time allows).

e) I would like to be able to view a list of people I am following, a list of people I am followed by, and a list of people I've blocked.

f) Search for users to add

g) ????

D) Creating Calendar

As a user,

1) I would like to be able to view a basic calendar, with no other information added yet
2) I would like to be able to toggle on official Purdue academic dates
3) I would like to be able to set up specific events on my calendar using the day, month, and time period the said event occurs
4) I would like to be able to color code my events to my liking to avoid monotonous layouts of information
5) As a student, I want to connect my calendar and upload it from other calendar services such Google Calendar and Outlook calendar.

E) Miscellaneous

As a user,

1) I would like to be able to see general upcoming Purdue clubs through the use of BoilerLink
2) I would like to be able to search for specific Purdue clubs through a search bar
3) I would like to RSVP for an event

4) I would like to be able to view a map of Purdue using Google Maps and Purdue's maps
5) I would like to view recommended walking times between different buildings along campus.
6) I would like to view the weather forecasted for that day in correspondence to my classes.
7) I would like to view detailed class information, including course descriptions, prerequisites.
8) I would like to be able to submit my own review of a class through a professor review system
9) I would like to see an overall rating for each class and instructor based on student reviews
10) I would like to be able to favorite classes and clubs to a personalized list for quick access.
11) I would like to be able to view upcoming events on my homepage that I am interested in.
12) I would like to be able to see mutual friends and club interests with other users.
13) I would like to be able to report other students for inappropriate behavior.
14) I want to be introduced to a basic tutorial of the general layout of the website, introducing tabs/information that are important to understand.
15) I want to be able to receive little tidbits of advice in the top corner of the application for important features that many other students miss out on - can silence these little notifications.

Non-Functional Requirements:

1. **Architecture and Performance**

   As a developer,

   a. I would like PurdueHub to run smoothly without students encountering bugs that will impede their enjoyment of our service

   b. I would like PurdueHub to have a reasonable response time of less than a second for our authentication to get students quickly engaged in our application.

   c. I would like PurdueHub to handle errors by providing a reasonable message to students so they can send it over for feedback or provide them insight on what went wrong

2. **Security**

   As a developer,

   a) I would like to setup a secure authentication system to allow students to register using their email and password

   b) I would like to set up alternative methods of signing up for PurdueHub apart from an official Purdue email if they do not feel comfortable for it

   c) Utilize a strong cryptographic hashing algorithm + salting for our data in storage to add complexity and discourage brute-force methods into personal information

   d) Messaging feature will use end to end encryption along with the use of TLS to protect our data from both users but also our server

3. **Usability**

   As a developer,

a. I would like to create a user interface that is not only user friendly, but also easy to navigate especially when users try to access our application on their phone
b. Implement a smooth and responsive UI, that will be intuitive to easily access their saved information regarding classes, clubs, and upcoming news they want to be notified for
c. Design our interface to allow the user to access their profile/account page from anywhere in the web application within 4 clicks.
d. We want to cater to visual impaired individuals, by including alt-text from non-text items on their page.

4. **Hosting/Development**

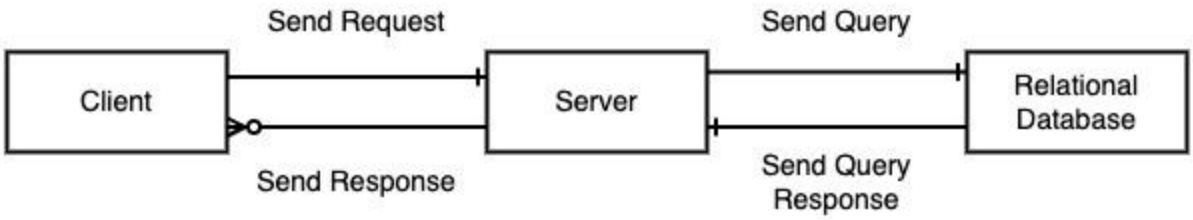
As a developer,

a) Frontend and backend will be deployed separately as two different microservices.
b) The frontend will be hosted for free on Github Pages, while the backend can be hosted, after we are far enough in development, on a cloud service provider such as AWS or Firebase.

Design Outline:

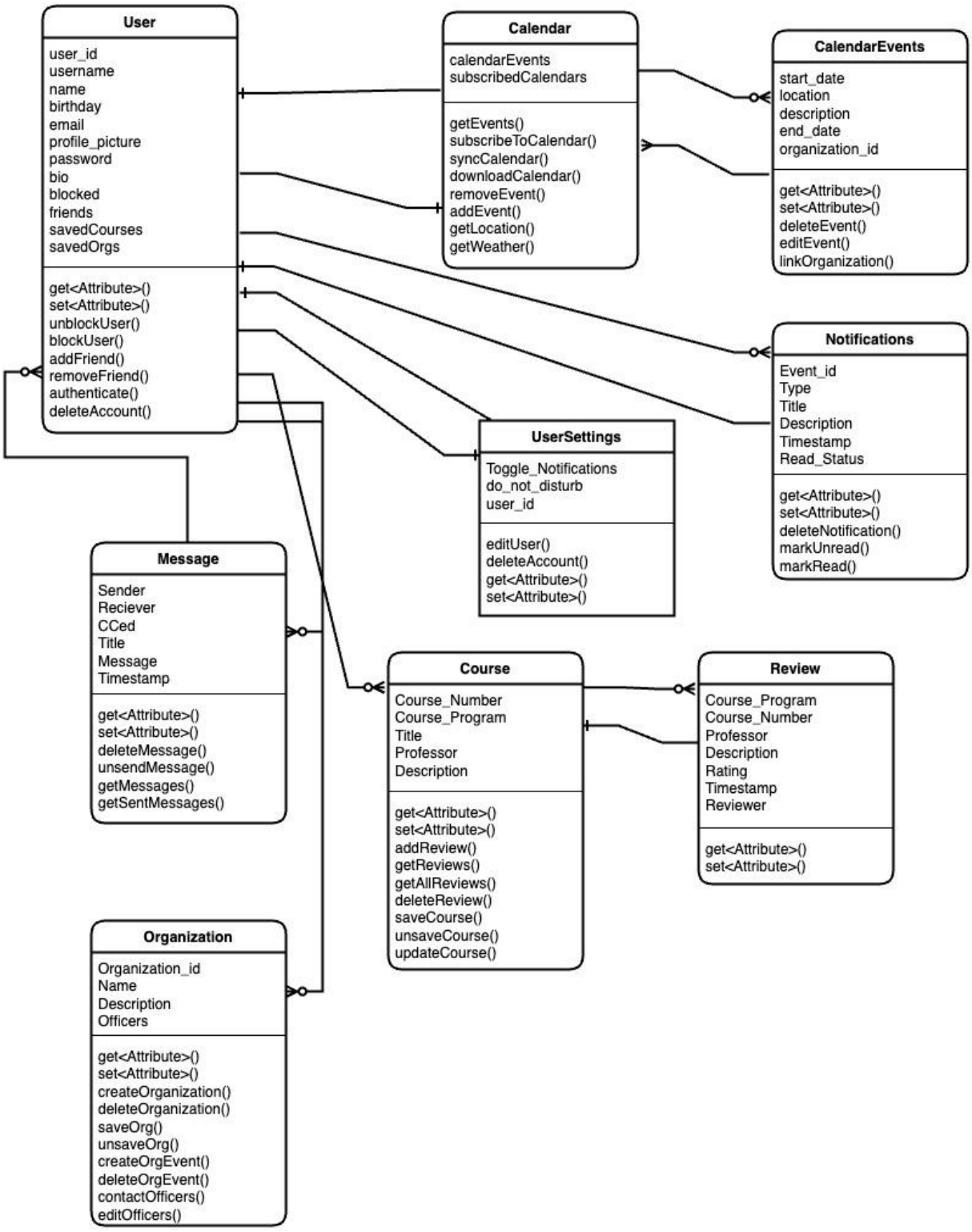
High-Level Overview

This project will be a web application that allows users to create and maintain their profile and the related information. This application will allow users to send messages and manage a calendar among other things. The application utilizes a client-server model written in Javascript that allows users to concurrently interact with the features of the application, maintaining a durable, secure information state.



1. Client
    a. Provides an interface for the user to interact with application
    b. Sends HTTP requests to the Server
    c. Displays response data from the server
2. Server
    a. Accepts HTTP requests from the Client, handles requests, sends HTTP response to Client
    b. Sends queries to relational database for information gathering and handles response
3. Relational Database
    a. Maintains durable, secure information state about user information and related entities
    b. Receives requests from Server, executes queries, sends response to Server

Design Details:



Description of classes and interactions between them.

It can be assumed that all classes have a unique identifier/id.

1. User
    a. A user object is created upon registering an account.
    b. Each user gets a unique identifying id.
    c. Each user contains the relevant sign-in information and personal information.
    d. Each user can also modify other users on their friends list, blocked list, and see relevant messages and notifications (other classes).
    e. Each user also has a personal calendar corresponding to them.
    f. Each user can also like and unlike courses, clubs, and organizations that they want to follow.
2. Message
    a. A message, similar to an email, between users, containing relevant information.
    b. A user can delete, view, and send/unsend messages.
    c. Each message will also have a unique id for identifying messages.
3. UserSettings
    a. Specific settings related to a user containing fields listed above.
    b. A user can modify these settings whenever desired and the settings are unique to a user.
4. Notifications
    a. Notifications about calendar events, friend requests, messages, etc. related to a specific user.

    b. These notifications have unique identifiers and an organization/tag identifier matching the notification to the relevant event-type or organization.
    c. A user can delete, view, and unview notifications.
5. Calendar
    a. Contains the information relating to a user's personalized calendar views. This includes calendars related to organizations at Purdue and personal calendars as well as personal events.
    b. Each user has a unique calendar and the calendar stores the ids of calendar events.
6. CalendarEvents
    a. Contains relevant information to a specific calendar event. This information can be related to multiple user calendars.
    b. Each calendar event has a unique id as well as an organization id related to the calendar event. This organization id can be empty or set to a preset to indicate that it isn’t related to an organization.
7. Course
    a. Each course contains the professor teaching the course that semester, along with a unique id of the course section (e.g. CS) and the course number.
    b. Each course can also have reviews for the course.
    c. Users can leave reviews for courses as well.
8. Reviews
    a. Each review contains the person who left the review, along with a rating and the course described in the review.
    b. Users can filter for reviews, delete their reviews, and add reviews of their own.

9. Organizations
    a. Each organization represents a club or organization at Purdue with a unique organization id.
    b. They contain basic information including a title/name, description, and a list of officers.
    c. Officers stored in this list can be seen by any user to be contacted.
    d. Officers can also attach an organization_id to a calendar event when they create one.

Design Issues:

Functional Issues:

1) What information is needed for creating an account on PurdueHub?
    a) Option 1: Nothing, entering as a guest student
    b) Option 2: Name, Email, Password
    c) Option 3: Name, Username, Email, Password
    d) Option 4 - Name, Username, Email, Password, Phone Number

Choice: Option 3

Justification: Since we are aiming for student activity, we wanted to implement a username system on PurdueHub to allow other students to find their friends by their unique username. It avoids issues with common names that we will have to account for, leading to a decrease in efficiency and this choice puts creativity into the students' hands. Phone numbers are not required for any of the services we provide.

2) What happens if I am experiencing messaging harassment?
    a) Option 1: Add a block button
    b) Option 2: Ban students for inappropriate behavior
    c) Option 3: Implement censoring feature
    d) Option 4: 1 + 2

Choice: 4

Justification: Banning students for not showing common courtesy is not an extreme measure and will be taken if necessary, especially in cases where repeated harassment occurs. We also implement a block button to make it easier on users' sides to remove any form of communication from students not showing decency. Censoring feature is not needed since we have the block button if someone does not appreciate the use of inappropriate language. It allows the students freedom in their communication with close friends and others whom they are comfortable with.

3) What happens if I experience a sudden disruption while performing an activity such as adding an event to my calendar
    a) Option 1: Nothing
    b) Option 2: Provide a detailed messaging for system errors and the ability to report these issues to our team

Choice: 2

Justification: Since we are aiming to deal with common student frustrations, by simply providing no information to students, we are only aiming to cause further annoyance. By

providing detailed messages for common errors, and general errors, and the ability to directly report these details to us, can students be satisfied that system errors are being dealt with promptly.

4) How should students modify their calendars?
    a) Option 1: Manual Input
    b) Option 2: Upload of a Google Calendar or other common calendar services
    c) Option 3: All the above

Choice: 3

Justification: If we are aiming to be a useful resource for students, by giving them the option to upload their respective calendar service to ours, can we retain students. If we only rely on manual input, for complicated schedules held by many students, it wastes time on their end, leading to less retention on our application. We are aiming to become integral in the lives of Purdue University students. So we want to ease the transition of the use of multiple services to hopefully only 1 as much as possible.

Non-Functional Issues:

1) What framework should be used?
    a) Option 1: React
    b) Option 2: Angular
    c) Option 3: Vue.js

Choice: 1

Justification: We decided to use React for our front end of our web application due to its popularity and most importantly, ease of use. Since we are dealing with some inexperienced developers, utilizing a framework that is not only popular, but also easy to use allows us to develop PurdueHub much more quickly. This allows us to satisfy the requirements needed from us as a team and perform at our best. Also, React's component-based architecture and virtual DOM makes it an effective framework to utilize with its flexibility.

2) What hosting service should be used?
    a) Option 1: AWS
    b) Option 2: Firebase
    c) Option 3: Heroku

Choice: A

Justification: We decided to use AWS to host our web application because of its popularity and maintainability. Additionally, AWS is well-documented, with good customer support and, as students, we have credits to apply to cover the costs. While Firebase has some free models, they are restrictive and require the application to stay within some size parameters. Heroku doesn't have any free pricing models and as such, AWS made the most sense from an affordability and maintainability standpoint.

3) What should we use for server-side programming?
    a) Option 1: Node.js (JS)
    b) Option 2: Node.js + Express.js (JS)

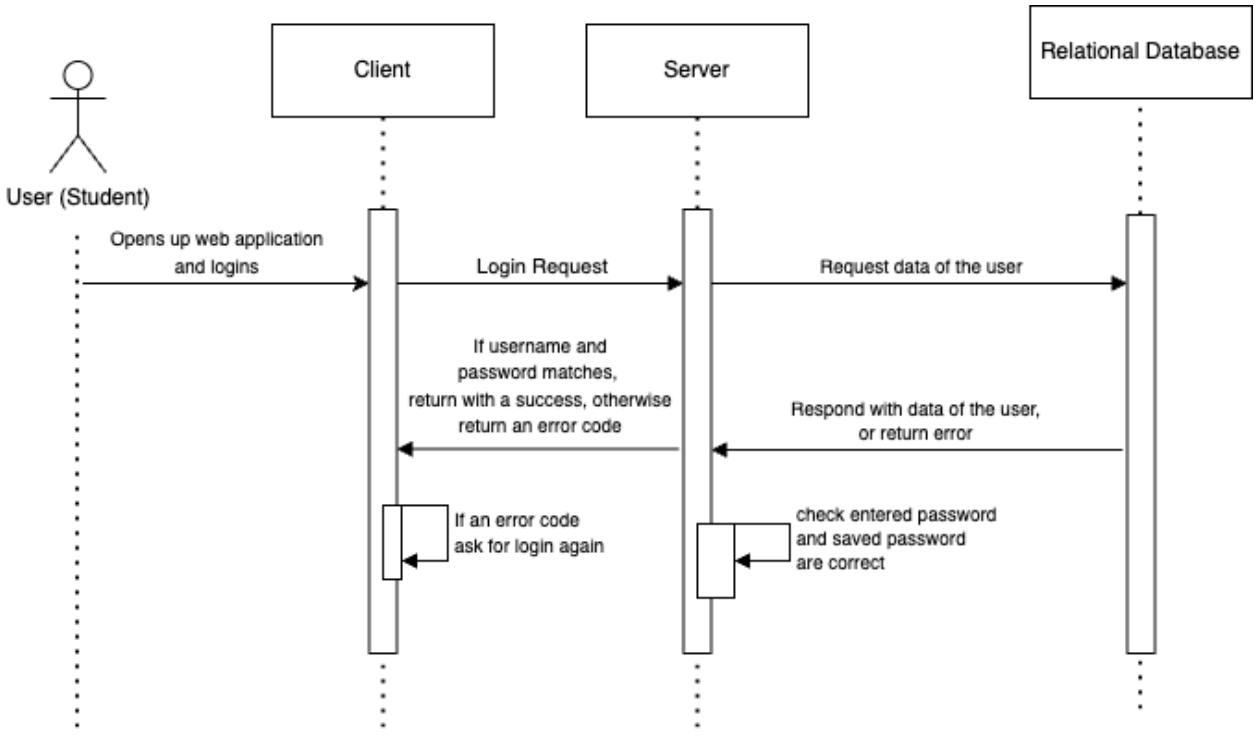
c) Option 3: Flask (Python)

Choice: 2

Justification: Node.js + Express.js combo allows our team to build our server-side application quickly. The community and amount of resources for both allows our inexperienced members to catch up quickly, allowing us to develop more efficiently. The flexibility and scalability brought on by both of these allows us to target future goals if we seek any with our application.

Sequence Diagrams

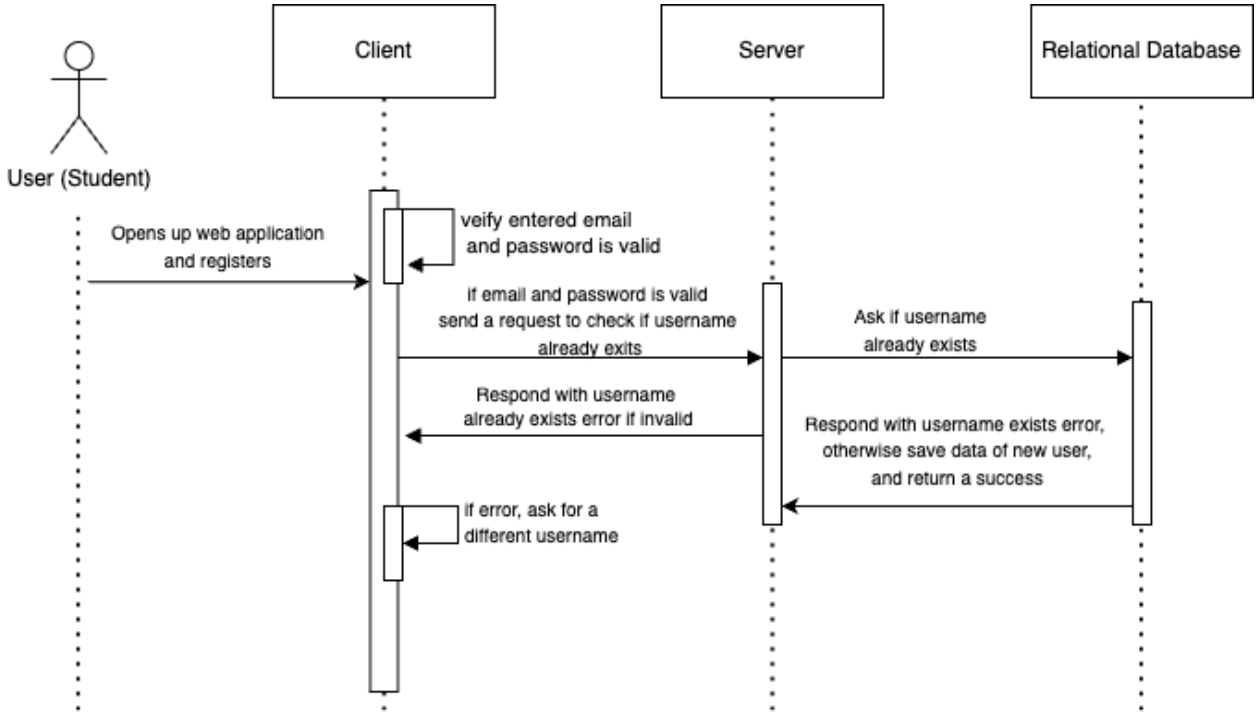
Signing into PurdueHub:



When the user opens up PurdueHub, they will be greeted with a sign in page in which they can use their username and password to sign in to our service. We will check if the entered username

and password are valid by sending a login request from the client to our server, which will retrieve the associated user data if it exists. If the username does not exist, we send back an error code and ask the user to go ahead and retype a valid username and password. If the username does exist, we go ahead and compare the password saved in the database to the entered password, if they match, proceed to the homepage of the user, otherwise return an error code for an incorrect password and ask the user to go ahead and try again.

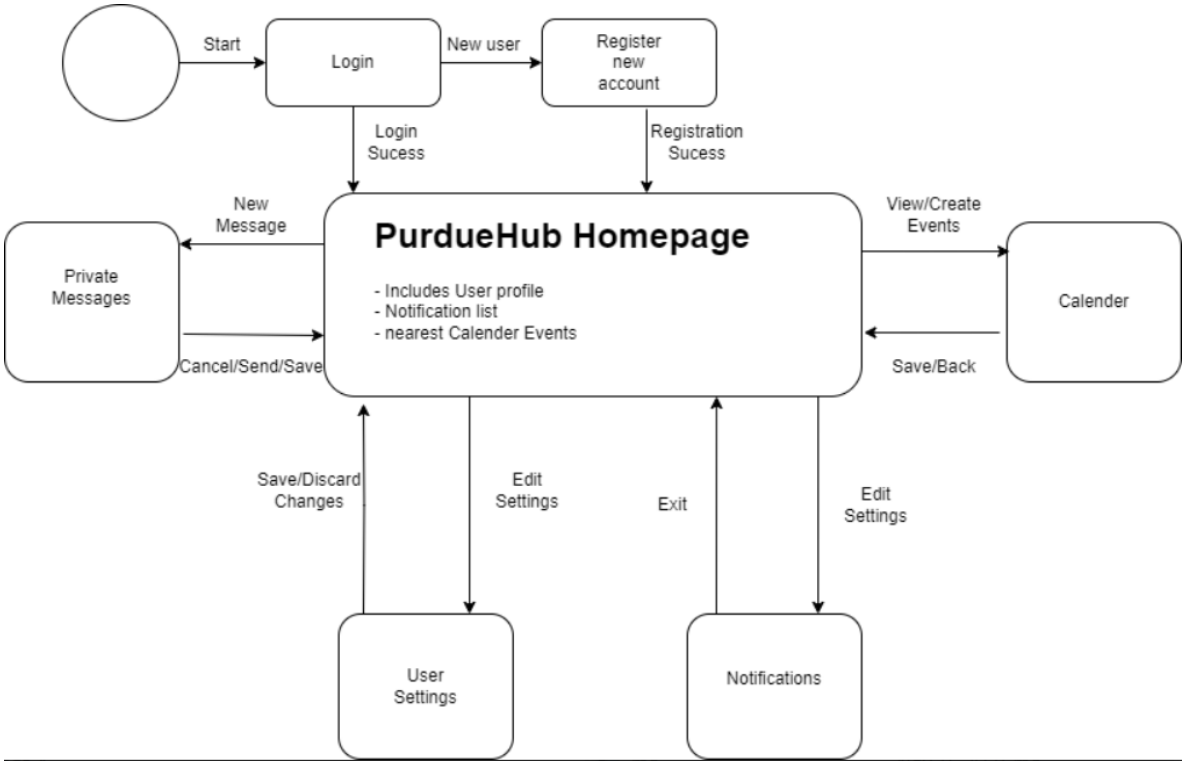
Registering for PurdueHub:



When the user opens up PurdueHub, they will be greeted with a sign in page in which they can select a register page if they do not have an existing login. They can enter an email, password,

and a unique username. We validate if the email and password follow our guidelines (does not include certain characters and or other information). If the email and password entered is valid, we send a request from the client to the server to verify if the unique username already exists within our database. If it does, we simply ask the user to enter a different username, otherwise we accept their information and they can go ahead and login to the homepage.

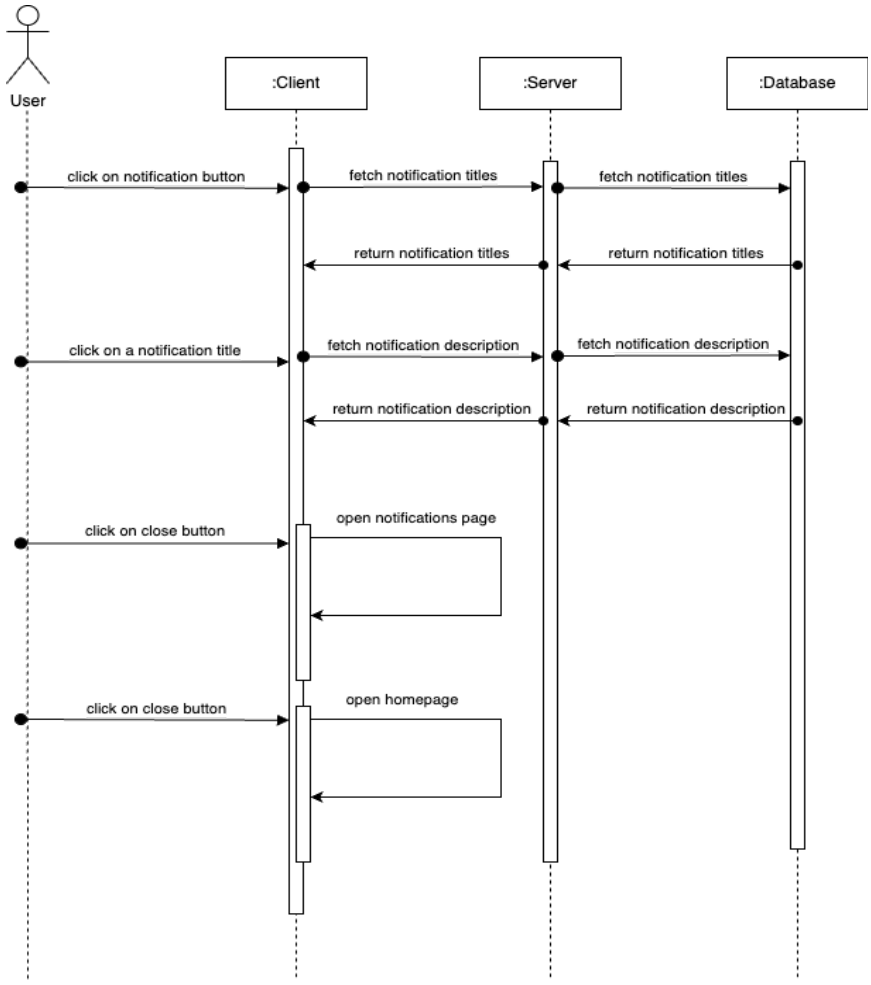
Navigation:



The user is first greeted with the Login page where they can enter their account credentials or sign up and create a new account. Once the users log in, they are greeted by the PurdueHub Homepage. On the top of the page are the users notifications and the rest of the page displays the

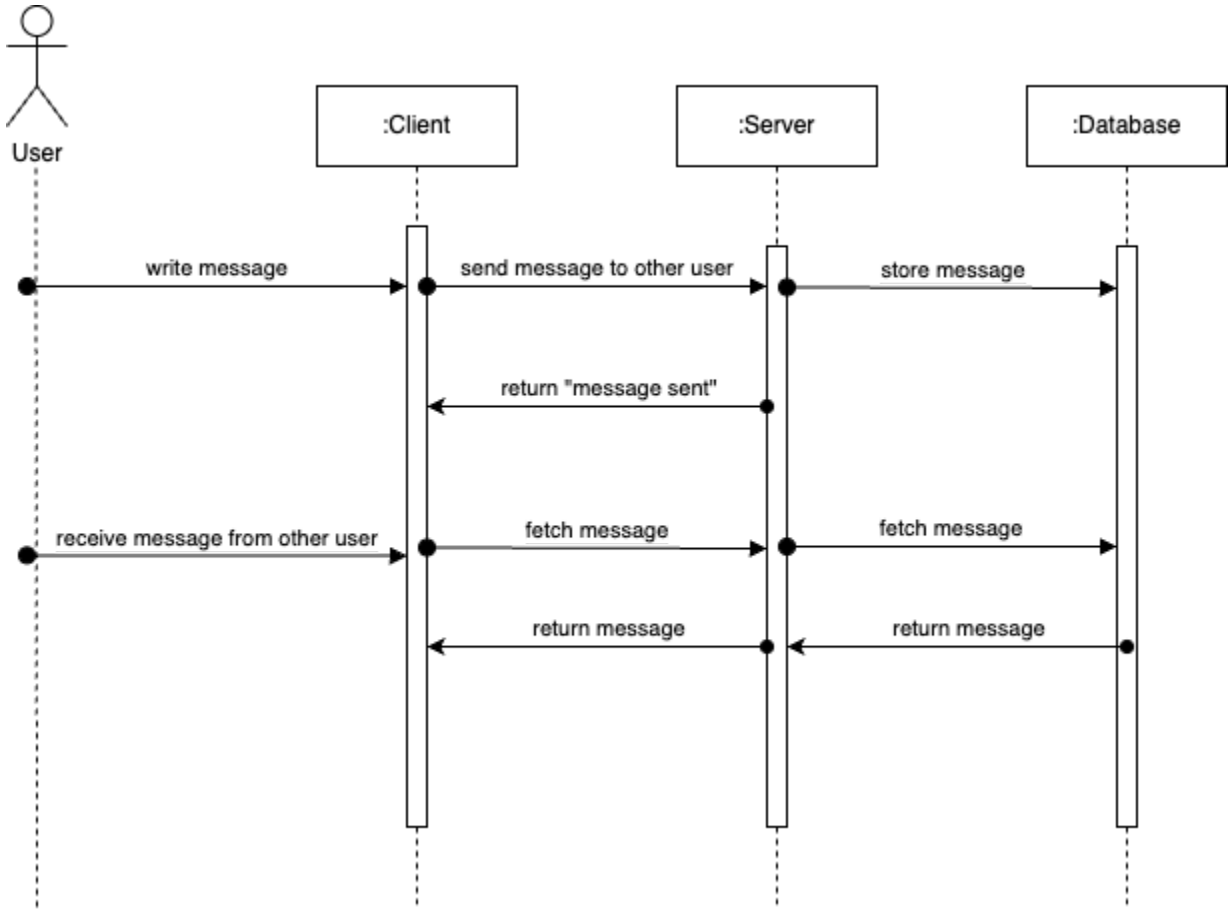
users profile. The main page with also display buttons for User settings, the Calendar, Upcoming events and Private Messages.

Viewing notifications:



Starting at the homepage, when the user wants to view their notifications, they will click the notification button which will show them a list of their notifications that have been fetched from the database. The user can click a notification title from their list to read the description of that notification which is also fetched from the database. When the user is done with that notification, they can click the close button to go back to the notifications page and when the user is done with all notifications, they can click the close button to view the homepage again.

Sending and receiving messages:



When the user writes and sends a message, the client send the message to the server and the server delivers the message to the intended user and also stores the message to the database. The

server also lets the client know that the message has been successfully sent. When the user has received a message and wants to read it, the client will fetch the message from the database and the database will return the message to the client by way of the server.