CAMPUSLINK
DESIGN DOCUMENT

Team Members:
Hareum Lee, Xavier Dextra, Sparkle Biswas, Divay Gupta, Joseph Jackson, Rudis Sprenne

Index

Purpose

Now that students do much of their work online and on computers, a quality learning software is essential to help them succeed. Having clear locations to submit assignments, take quizzes, and review lecture material is crucial for not only a student's success, but to not cause any additional stress that may come from not being able to find what one is looking for. Many learning platforms have confusing layouts, and often times there is not a single location for all of a student's assignments.

We are building CampusLink, a learning management system for creating, hosting, and editing online learning resources. Existing systems often have non-intuitive interfaces that can make it inconvenient for professors to perform routine tasks such as entering student grades, for students creating grade predictions, and for students to find assignments. For example, in other learning platforms, finding quizzes and even assignments as a student is not always apparent and can be a hassle. We aim to make this process smoother with an easy-to-use interface, and by consolidating all of a student's assignments into one tab for easy access.

Functional Requirements

1. User account
   As a user,
   a. I want to be brought to the login page upon visiting the website.
   b. I want to be able to register for a CampusLink account.
   c. I want to be able to login to my account.
   d. I want to be able to logout from my account.
   e. I want to be able to reset my password.
   f. I want to be able to view the account settings page.
   g. I want to be able to edit my information on the account settings page.
   h. I want to be able to access the website from multiple devices and have the pages scaled properly for each device.
   i. I want to be able to navigate the website using a navigation bar.
   j. I would like to easily navigate to features common to all courses, such as the calendar.
   k. I want to be able to access and search a frequently asked questions page.
   l. I want to be able to post on a discussion board.
   m. I want to be able to respond to posts on a discussion board.

2. Admin
   As a user,

   a. I want to have exclusive access to an admin page that displays all registered accounts and requests for instructor accounts.
   b. I want to be able to approve or deny professor account requests

3. Instructor (teacher)

   As a user,

   a. I want to be able to login and have the page be in an instructor view.
   b. I want to be able to create or add courses I am teaching.
   c. I want to be able to modify existing courses.
   d. I want to be able to add students to an existing course.
   e. I want to be able to see the list of students registered for the course.
   f. I want to be able to post lecture slides or notes to the course.
   g. I want to be able to upload videos to the course (such as lectures or other media).
   h. I want to be able to embed videos to the course content (such as lectures or other media) using a URL.
   i. I want to be able to create quizzes for the course, such as multi-choice or short answer quizzes.
   j. I want to be able to create assignment submissions.
   k. I want to be able to set due dates for assignment and quiz submissions.
   l. I want to be able to set the number of submissions for assignments.
   m. I want to be able to set the duration for quizzes.
   n. I want to be able to set the number of attempts for quizzes.
   o. I want to be able to see how many students submitted assignments.
   p. I want to be able to enter grades for the students in each course.
   q. I want to be able to compute and display a class average for each assignment and an overall grade.
   r. I want to be able to update an announcements and calendar page.
   s. I want to be able to view how many times a student has viewed the course page/content (if time allows).

4. Student

   As a user,

   a. I want to view my courses.
   b. I want to receive update emails on any discussion posts I am a part of or have interacted with.
   c. I want to be able to view past or submitted quizzes after the due date.
   d. I want to be able to view my grades.
   e. I want to be able to calculate my “what if” grade for all courses
   f. I want to be able to view course content
   g. I want to be able to take quizzes.

h. I want to be able to submit files.
i. I want to be able to enter temporary weights for different graded items.
j. I want to be able to login and have my course be in student view
k. I want to be able to bookmark assignments and lecture notes in a special bookmark tab
l. I want to be able to view the calendar page which shows due dates.
m. I want to be able to see the names and emails of other students in the course.
n. I want to be able to get an email update when a course announcement page is updated.

Non-Functional Requirements

1. Client Requirements:
   As a developer,
   a. I want the web application to be accessible on desktop browsers
   b. I want the web application to be accessible on mobile web browsers.

2. Architecture Requirements:
   As a developer,
   a. I want the frontend to be built with React JS.
   b. I want the backend to built using Firebase.
   c. I want the NoSQL database to be built using Firestore.

3. Hosting and Deployment Requirements:
   As a developer,
   a. I want the deployment to be performed using Vercel.
   b. I want the code to be deployed directly from GitHub repository.

4. Security Requirements:
   As a developer,
   a. I want the passwords to be secured using an irreversible hashing algorithm (bcrypt, SHA-256, or SHA-512).
   b. I want strict password creation policies to be enforced.
   c. I want user sessions to be managed with secure JWT tokens.
   d. I want access to sensitive information to be controlled with access permissions for each user type.

5. Performance Requirements:
   As a developer,
   a. I want the website to be up 24/7 unless under maintenance.

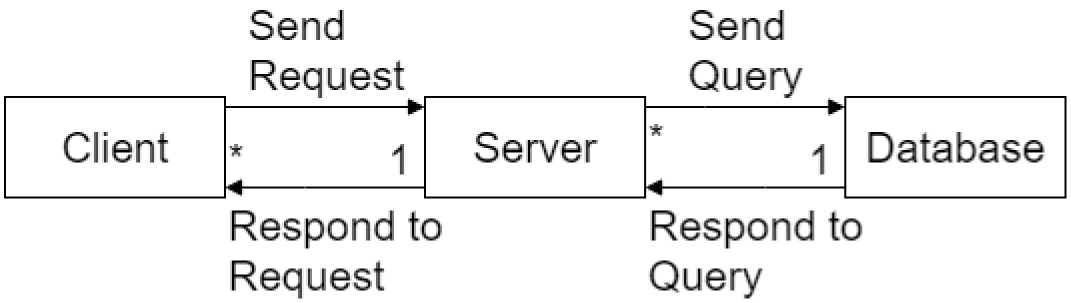
b. I want to minimize the amount of requests between the web application's client and server to increase responsiveness.
c. I want the website to be able host up to 100 users (which can be upgraded with a paid subscription) and be scalable without API modifications.
d. I want both the client and server to gracefully handle errors and the failure of other components that they interact with.

6. Appearance Requirements:
   <u>As a developer,</u>
   a. I want the web application to be aesthetically pleasing and easy to use.

Design Outline

High Level Overview

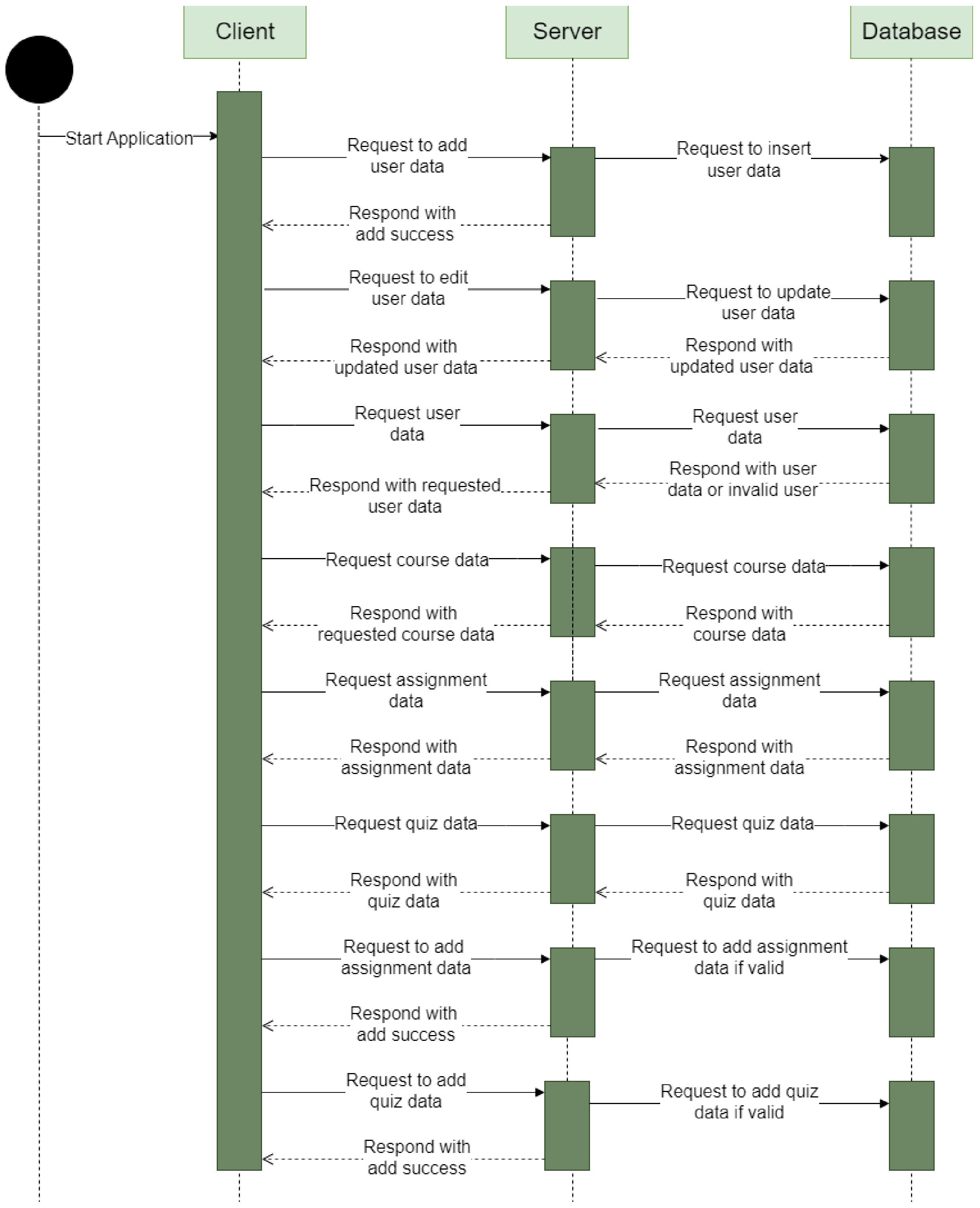
This project will be a web application that allows students to interact with classes and complete assignments, take quizzes, view lecture material, and use discussion boards. Teachers will be able to create assignments and quizzes, post lecture material, edit student grades, and interact with students through discussion boards. The application will use client-server architecture, in which one server will concurrently handle multiple clients.



1. Client
    a. Client will provide a user interface to interact with the program.
    b. Client will send requests to the server based on user input.
    c. Client will receive data from the server and display it on the user interface.
2. Server
    a. Server will receive requests from the client.
    b. Server will query the database to receive, add, or modify data.
    c. Server will generate a response to the client request and send the requested data.
3. Database
    a. A non-relational database will store data for the program, such as user information, assignments, quizzes, and grades.
    b. Database will respond to server queries and send requested data back to server

Sequence of Events Overview

The sequence diagram shows how the client, server, and database interact. The sequence begins with the user opening the web application. As the user logs in or creates an account, the client sends a request to the server. The server handles the request, and sends a query to the database. The server receives information back from the database, and responds by allowing the client to log in. Once the client is logged in, they send requests for other actions, such as updating user data, viewing course information, and viewing and completing quizzes and assignments. The server handles these requests and sends queries to the database to add or obtain the necessary data. The database responds with the requested data, and the server returns the data back to the client.


Client
Server
Database
Start Application
Request to add user data
Request to insert user data
Respond with add success
Request to edit user data
Request to update user data
Respond with updated user data
Respond with updated user data
Request user data
Request user data
Respond with requested user data
Respond with user data or invalid user
Request course data
Request course data
Respond with requested course data
Respond with course data
Request assignment data
Request assignment data
Respond with assignment data
Respond with assignment data
Request quiz data
Request quiz data
Respond with quiz data
Respond with quiz data
Request to add assignment data
Request to add assignment data if valid
Respond with add success
Request to add quiz data
Request to add quiz data if valid
Respond with add success

Design Issues

Functional Issues

1. What information is needed to create a CampusLink account?
    - Option 1: Full name, username and password
    - Option 2: Full name, university email id and password
    - Option 3: Full name, any email id and password

   Choice: Option 2

   All students and professors are given university email ids. It makes sense to create accounts using the university email ids so that professors can add students to their registered courses.

2. How does an account get approved as an Instructor account?
    - Option 1: Approval is provided by an admin account
    - Option 2: Accounts created as a Professor automatically get approval

   Choice: Option 1

   Getting approved by an admin account makes more sense here because they can verify the credentials provided by the user based on whether the provided email ID is valid (belonging to a professor). With the second option, any one could register as a professor without verification, which is not an ideal situation.

3. Should CampusLink have restricted/private discussion boards?
    - Option 1: Professors can start a discussion board and set it as private, where students cannot see other posts until they have posted
    - Option 2: Professors can start discussion boards without any option of setting it as private.

   Choice: Option 1

   Option 1 is better since professors use this feature extensively on other platforms like Brightspace. It prevents students from plagiarizing information from others' discussion posts and it's a nice feature to have in general.

4. Can students use the 'what-if' grade feature without the professor entering the grade breakdown?
    - Option 1: Students cannot use the 'what-if' grade feature without the instructor entering the grade breakdown
    - Option 2: Students can use the 'what-if' grade feature even without the instructor entering the grade breakdown. Students can manually enter the breakdown, which will get overwritten once the instructor enters the breakdown

   Choice: Option 2

We went with option 2, since we think students be able to approximate their grades at any time. Instructors sometimes don't even use the designated learning platforms, such as Brightspace, to grade students, again preventing students from using the 'what-if' grade system. Therefore students can manually enter their grade breakdown.

Non - Functional Issues

1) What kind of software are we building?
    - Option 1: Mobile native application
    - Option 2: Desktop native application
    - Option 3: Web application

   Choice: Option 3
   Option 3 makes more sense for an online learning platform. It is much more accessible to be able to submit assignments and take quizzes on a web browser anywhere than being restricted to a particular platform. Building a native mobile application will require iOS and Android implementations, which means working with something like React Native or Flutter, but that also restricts the userbase to mobile users and thus we would also require a desktop implementation, which will all be very infeasible due to time constraints.

2) What frontend language/framework should we use?
    - Option 1: React JS
    - Option 2: Angular
    - Option 3: Laravel (PHP)

   Choice: Option 1
   Option 1 is the better choice for us as a team since at least one of us is familiar with this library, and because Laravel and Angular have a higher learning curve, being full-fledged frameworks. React is also very flexible in its implementation for its components and how they interact when compared to the other two, which makes it the better choice for implementation purposes.

3) What backend language/framework should we use?
    - Option 1: Node JS
    - Option 2: Firebase

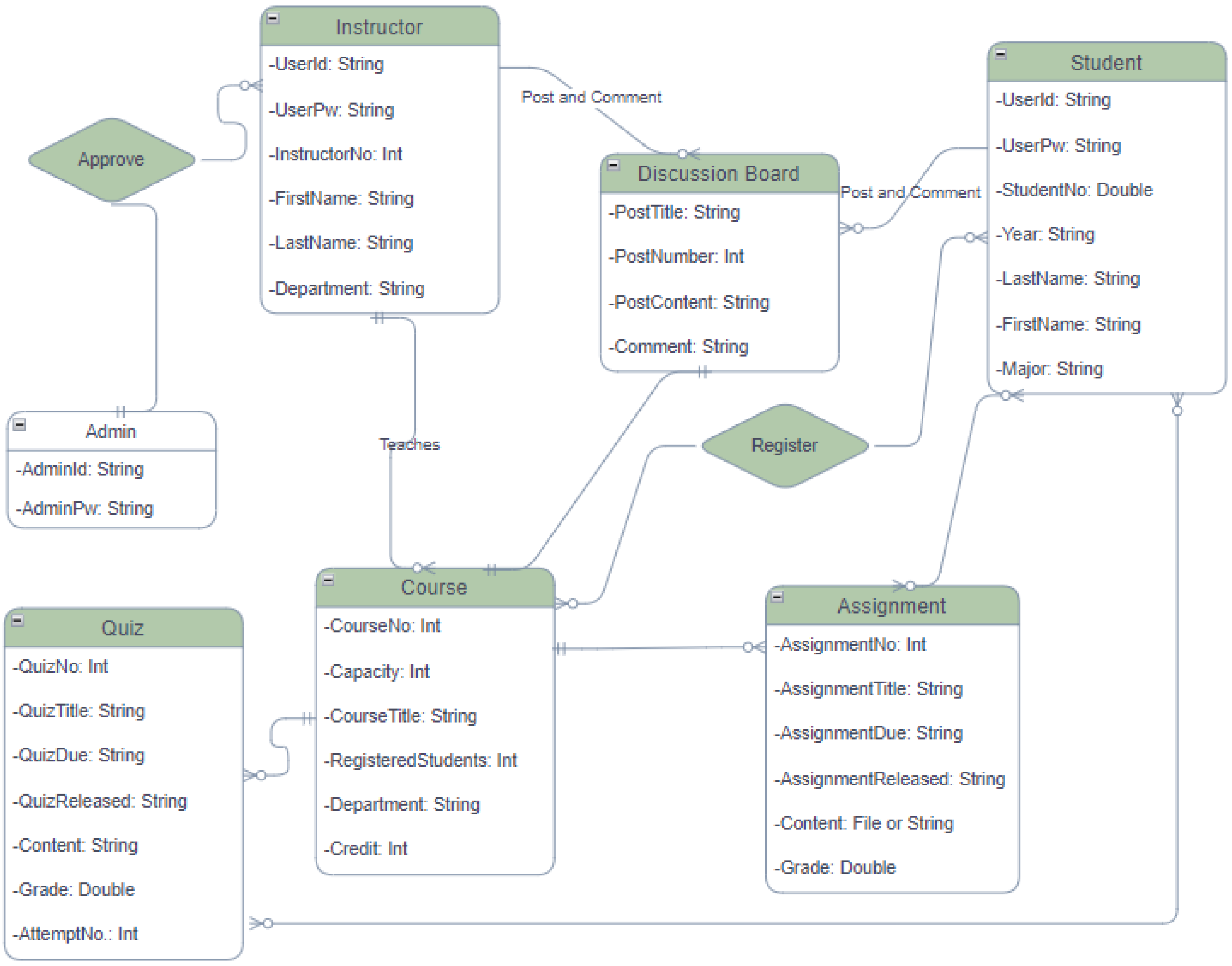
   Choice: Option 2
   Option 2 makes sense for our project as hosting the information on the cloud will make our application more scalable and more feasible in its implementation. Using Node JS will add to the complexity since we will also have to use another database, while Firebase comes with its own database (Firestore) which reduces the need for the integration of another service such as MongoDB, which will consume more time that can be spent working on web app functionality .

4) What database should we use?
    - Option 1: MongoDB
    - Option 2: Azure Cloud Database
    - Option 3: Firestore

    Choice: Option 3

    Option 3 makes the most for this project given our other architecture choices. Firestore comes packaged with Firebase, with no need for additional API keys or other functionality implementation. Being a NoSQL database, Firestore is also more flexible in helping us store and manage application data in a more fluid manner, as per the specific use case.

Design Details



Descriptions of Classes and Interaction between Classes

This ER diagram shows how classes are connected and what attributes each class is obtaining.

- Admin
    - When user request for approval to register as an instructor, Admin approves or denies.

- Instructor
    - User needs to be approved by admin to get ‘instructor access’
    - Contains identifying data about instructors’ name, instructor number, and department they belong.
    - Instructors have access to set or edit values in course, such as grades and due dates.
    - Instructors can create assignments and quizzes.

- Student
    - Contains identifying data about the student, such as name, year, major and student number.
    - Students can view courses they registered.
    - Students can view grades including ‘what if’ grades.
    - Students can view number of assignments.

- Course
    - Represents course title, capacity and number of students registered in the course.
    - Only visible to instructor and students who registered in this course.
    - Assignments and quizzes are created within the course.
    - There is discussion board for each course.

- Assignment
    - Contains identifying data such as assignment number and assignment title.

- Each assignment has released / published date and due date.
- It can be created, edited and graded by an instructor.
- Instructors can attach files to the assignment.
- Students can attach files to submit.

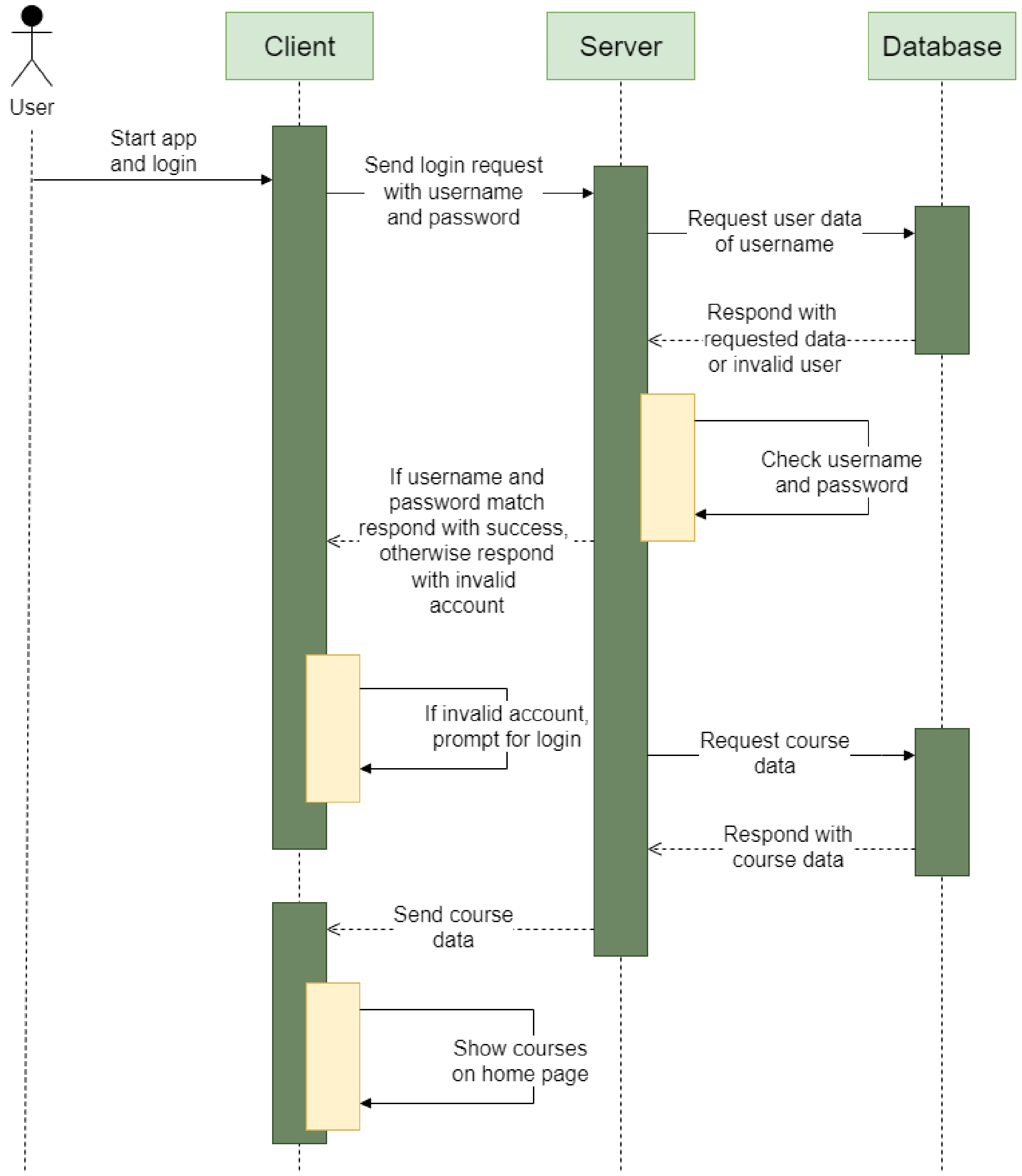
- Quiz
  - Contains identifying data such as quiz number and quiz title.
  - Each quiz has released / published date, due date and number of attempts.
  - It can be created, edited and graded by an instructor.
  - Instructor can add image, multiple choices or short answer questions.

- Discussion Board
  - It belongs to the course.
  - Instructor and students who are registered in the course can make a post.
  - Each post contains identifying data such as post number.
  - Post has a title and its contents.
  - Instructor and students can comment on the post.
  - Post and comment can be edited or deleted only by instructor or who created them.

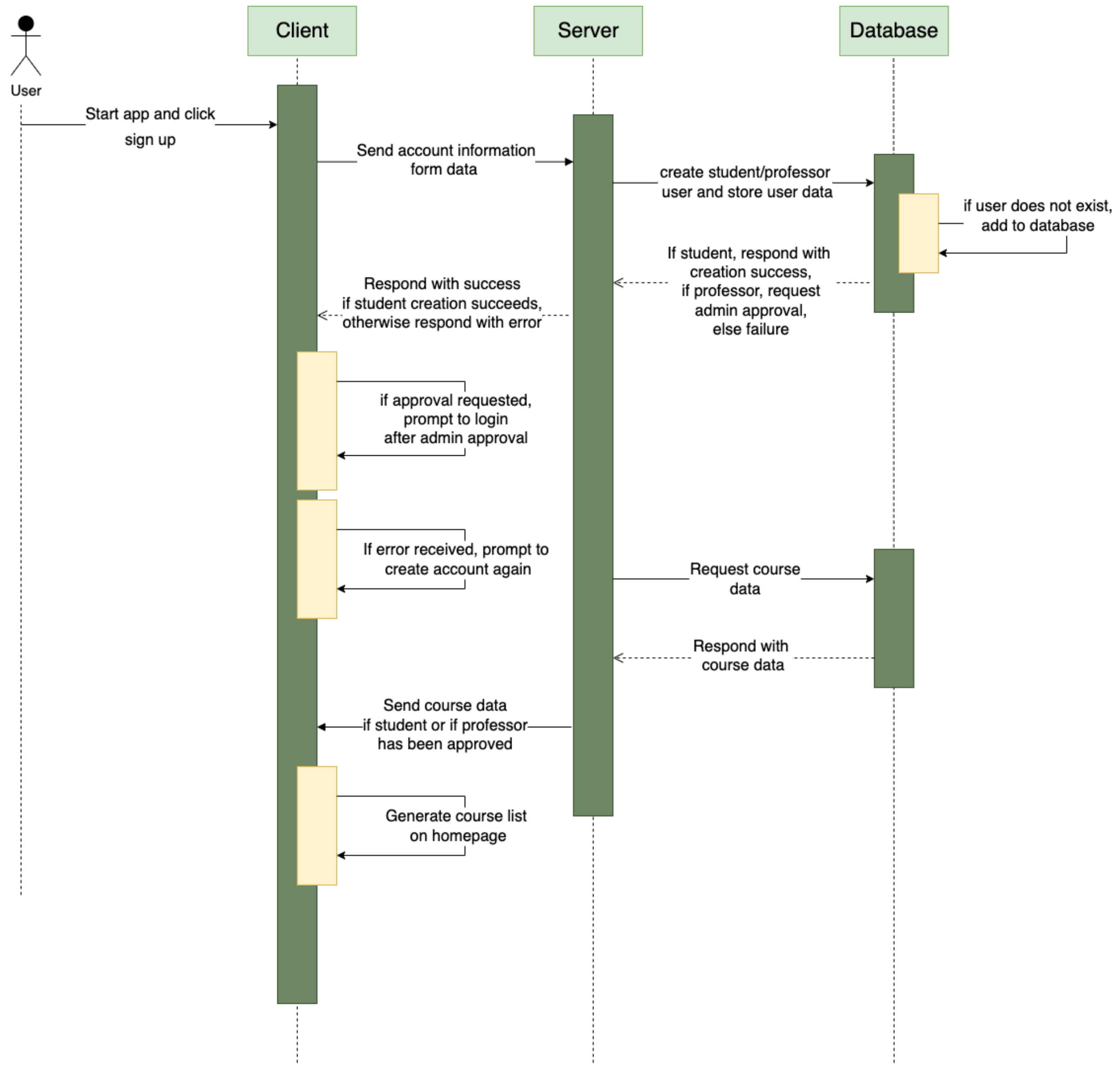
Sequence Diagrams

The following diagrams show the sequences of major events users will interact with, including loging in, creating an account, creating assignments and quizzes, and submitting assignments and quizzes. The diagrams show how requests and responses are sent in the client-server model. Users performing actions on the user interface prompts the client to send requests to the server, and the server queries the database for the requested data. The database will return the data to the server, which will then send it to the client. The client will then display the data to the user.

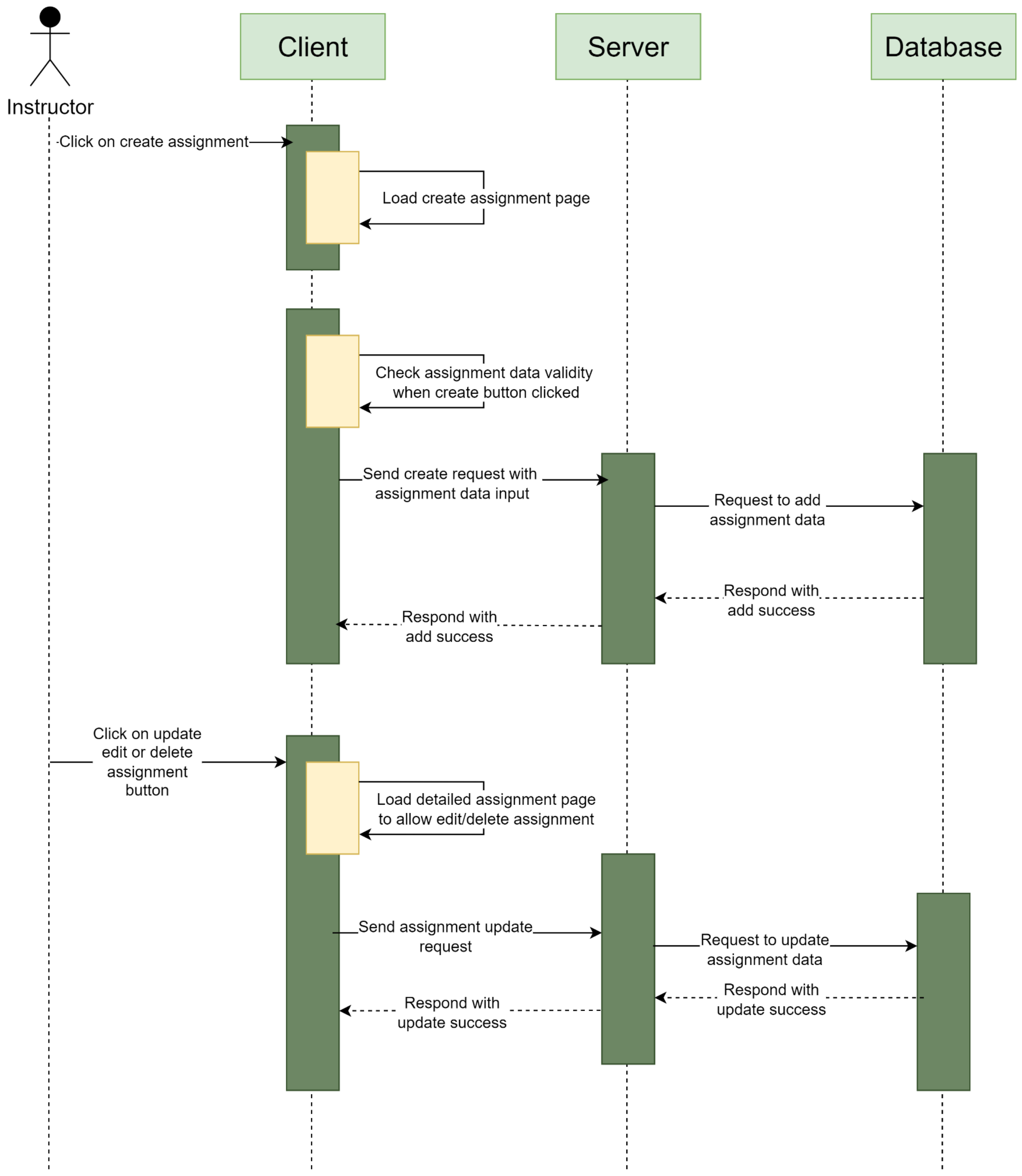
1. Sequence of events when a user logs in


User
Client
Server
Database
Start app and login
Send login request with username and password
Request user data of username
Respond with requested data or invalid user
Check username and password
If username and password match respond with success, otherwise respond with invalid account
If invalid account, prompt for login
Request course data
Respond with course data
Send course data
Show courses on home page

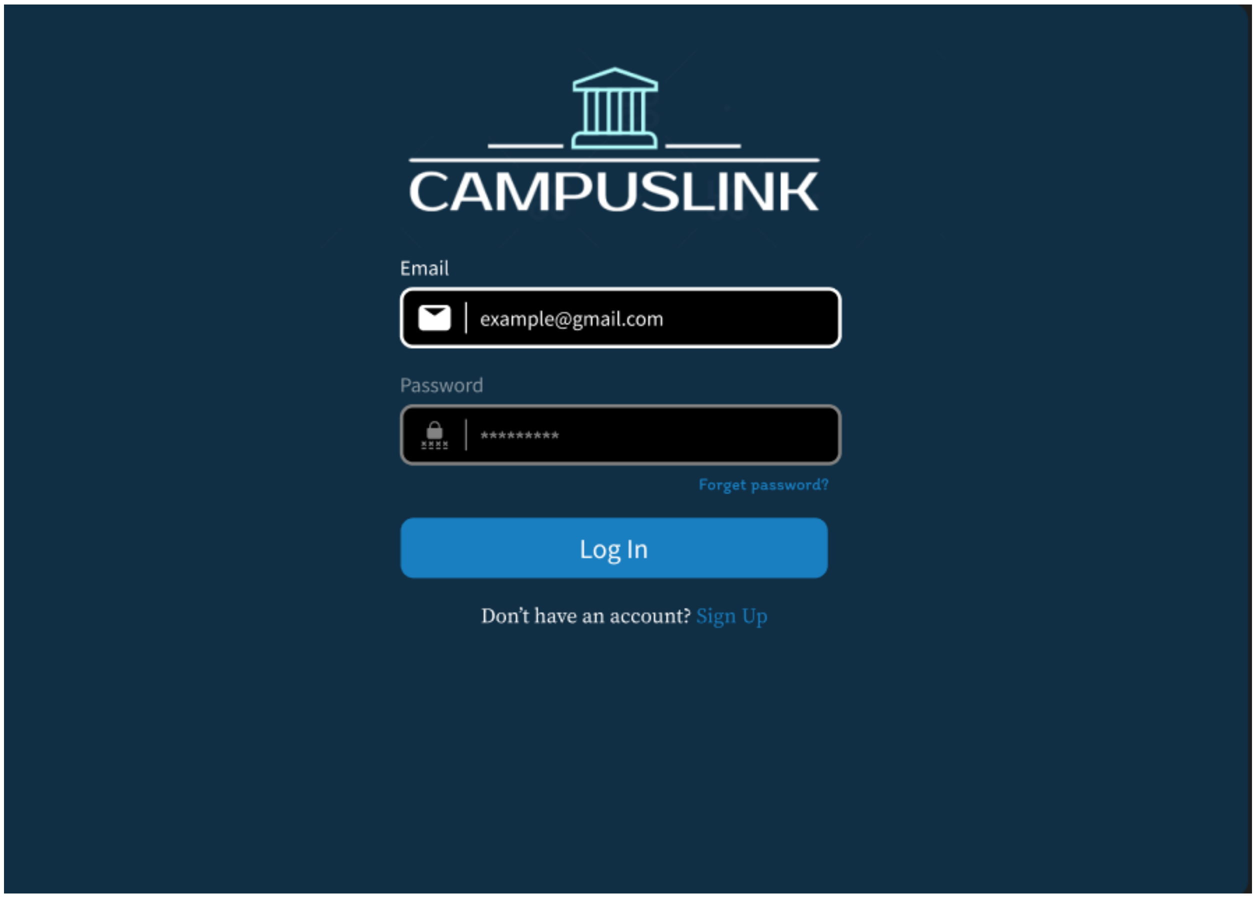
2. Sequence of events when a user creates an account


Client
Server
Database
User
Start app and click sign up
Send account information form data
create student/professor user and store user data
if user does not exist, add to database
If student, respond with creation success, if professor, request admin approval, else failure
Respond with success if student creation succeeds, otherwise respond with error
if approval requested, prompt to login after admin approval
If error received, prompt to create account again
Request course data
Respond with course data
Send course data if student or if professor has been approved
Generate course list on homepage

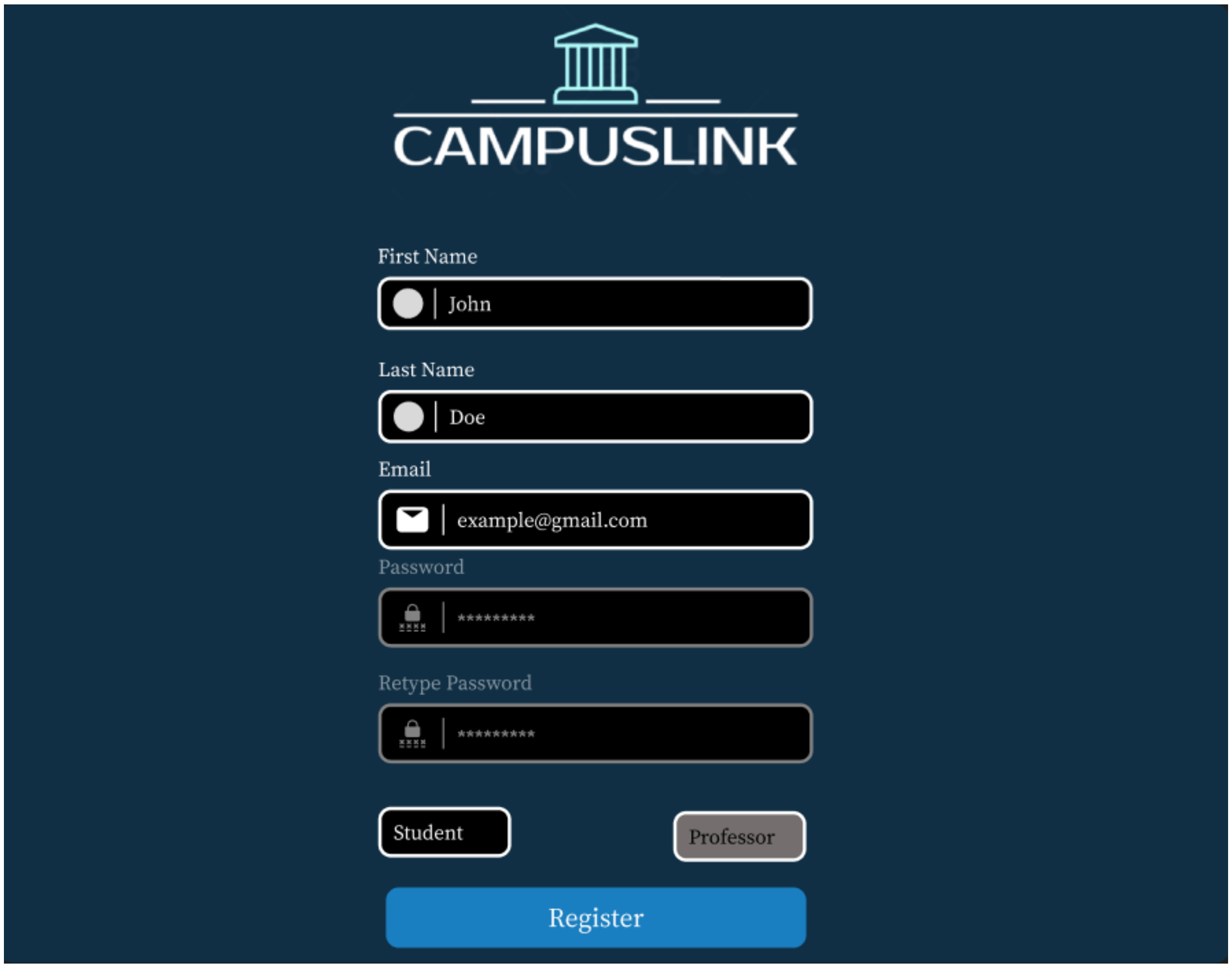
3. Sequence of events when an instructor creates an assignment


Instructor
Client
Server
Database
Click on create assignment
Load create assignment page
Check assignment data validity when create button clicked
Send create request with assignment data input
Request to add assignment data
Respond with add success
Respond with add success
Click on update edit or delete assignment button
Load detailed assignment page to allow edit/delete assignment
Send assignment update request
Request to update assignment data
Respond with update success
Respond with update success

UI Mockups



This is the login page. There is a button for creating an account in case the user does not have an existing account and a login button if they do. There is also a button to reset your password in case the user has forgotten it.



First Name
John

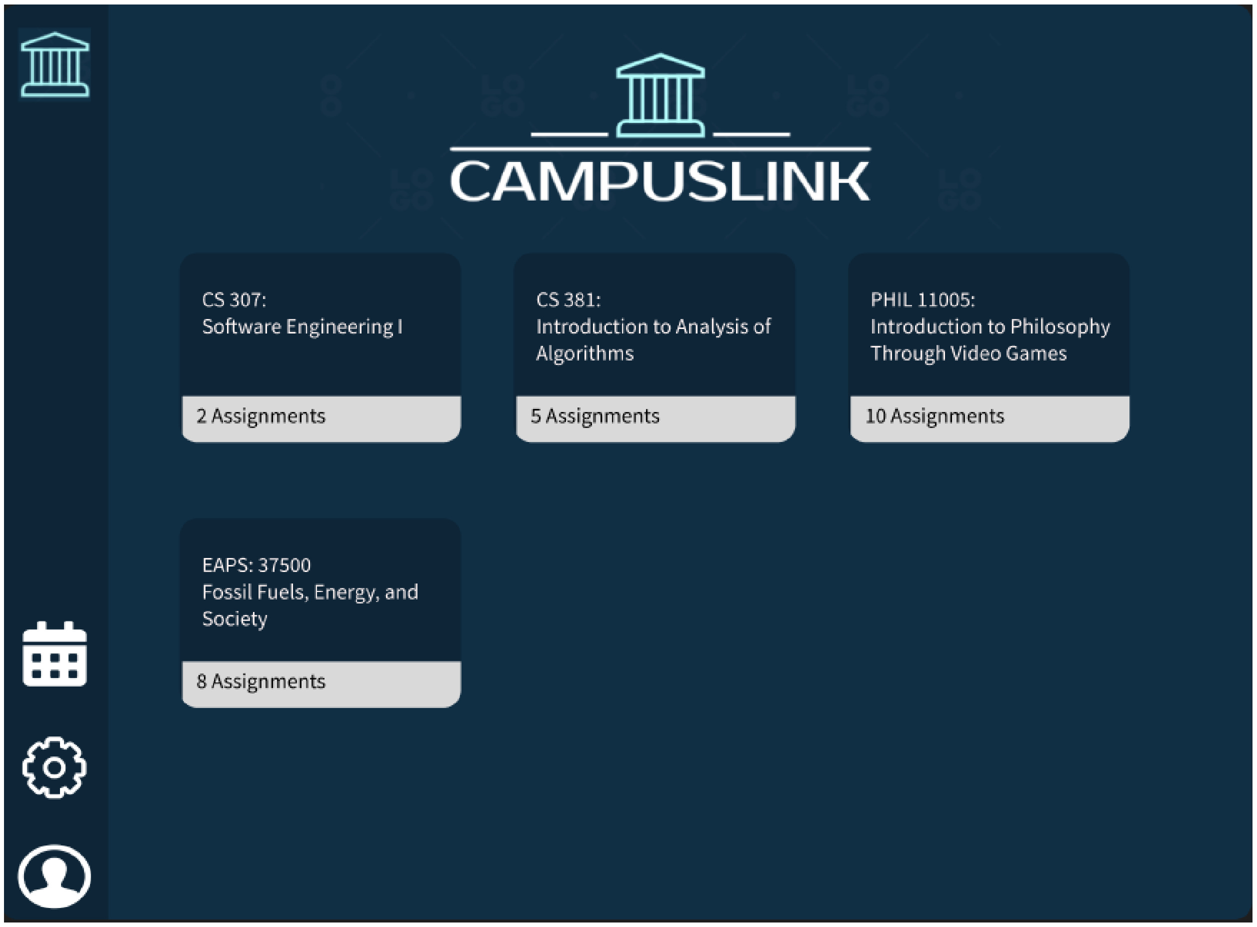
Last Name
Doe

Email
example@gmail.com

Password
****

Retype Password
****

This is the account creation page. The user must fill in all the information and select the account type (student or professor). The register button sends all the account information to the server.



This is the student homepage view, which is visible once the student is logged in. The registered courses are visible and can be clicked on to retrieve course data. The left side houses the navigation pane, where the top button takes the user to the homepage view, the calendar button displays a calendar with upcoming assignment due dates, the settings button takes the user to the settings page, and the account button takes the user to the account settings page.