UHub

Design Document

Index:

Purpose

Students across campus utilize multiple portals, resources, and services to access vital information regarding course work, grades, and announcements. Because of this rise in university web based resources, there has been a demand for accessibility and organization among the student population. UHub addresses this issue by offering students a unique and efficient framework; the application minimizes latencies by allowing users to access university portals, connect with other students, and organize academic life.

Non-Functional Requirements:

Architecture and Performance

Our overall design is structured to be a strong backend that is represented on a lightweight frontend. The backend will consist of everything necessary to store data (usernames and passwords) and all the framework to be able to retrieve data from all the APIs. The frontend will be a minimalistic dashboard that can be easily customized for each students needs. This will be implemented using React.js, a javascript library.

As for performance, the backend must be incredibly responsive. Logging in should quickly retrieve the essential APIs as quickly as possible.We plan to focus on the most essential APIs (Blackboard, Grades, Piazza, Email).

Security

The vast majority of information that will be passed through our service will be from APIs and separate services; they can handle the security of private information (usernames, passwords, student IDs, grades) by themselves. For the information stored, we will be using an Encrypting File System for the safety of the information.

Usability

Due to the implementation of several features and APIs, the interface and UI should be user friendly and self-explanatory for navigation and usability. We would like the front end to be split into 4 main components: the navigation bar, the sidebar, customizable sidebar, and the main body of the page that will display the different platforms and resources. If necessary, we will also create routes to new pages that carry more specific functionality with respect to the specified resource.

Hosting/Deployment

The front end will be hosted using github pages, easily accessible through a demo link on the github page. To integrate the back end with the front end, we hope to use AWS(Amazon web services) or similar.

Functional Requirements:

As a user:

1. As a user, I would like to be able to register for a UHub account.
2. As a user, I would like to be able to log on to my UHub account.
3. As a user, I would like to be able to manage my UHub account.
4. As a user, I would like to be able to reset my password to my UHub account.
5. As a user, I would like to only log on ONCE to all platforms.
6. As a user, I would like to easily navigate any platform.
7. As a user, I would like to be able to customize my homepage, changing position of elements and which elements are shown.
8. As a user, I would like to be able to "connect" with friends, that have an account on UHub. This will encourage collaboration and study groups.
9. As a user, I would like to interact with a responsive design.

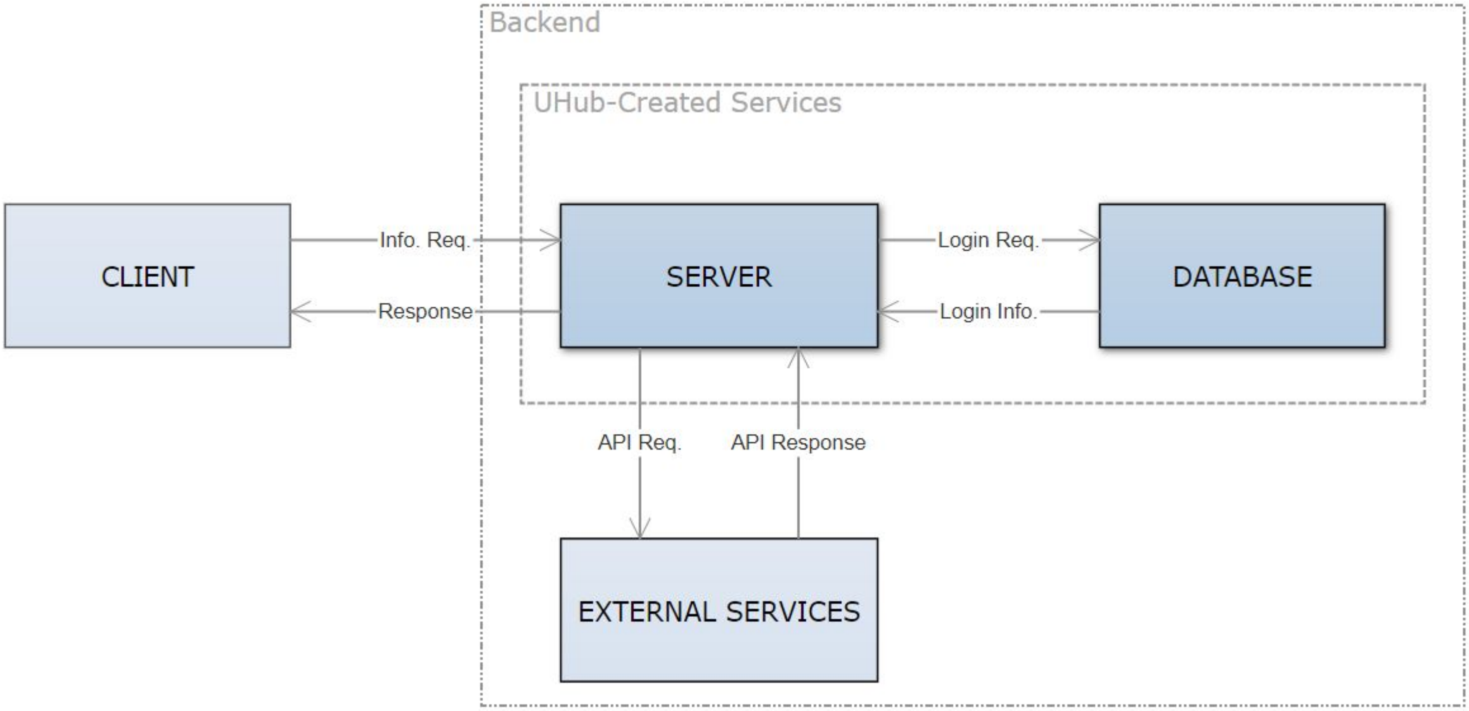
As a student:

1. As a student, I would like to view my grades.
2. As a student, I would like to view deadlines for specific courses.
3. As a student, I would like to view the courses I am enrolled in.
4. As a student, I would like to access my school email.
5. As a student, I would like to be able to ask and respond to questions related to course work.
6. As a student, I would like to have access to a messaging board.
7. As a student, I would like to be able to see if my peers are enrolled in similar courses.
8. As a student, I would like to create study groups with my peers.
9. As a student, I would like to have access to a personal to-do list.
10. As a student, I would like to have access to an interactive calendar/planner.
11. As a student, I would like to have access to Blackboard.
12. As a student, I would like to have access to Piazza.
13. As a student, I would like to have access to School Email.
14. As a student, I would like to receive notifications for course announcements and grades.
15. As a student, I would like to add events to my course calendar.
16. As a student, I would like to view general school announcements.
17. As a student, I would like to view possible transportation schedules.
18. As a student, I would like to access UHub across multiple devices.
19. As a student, I would like to have a sidebar with quick access to my resources.
20. As a student, I would like to real-time updates when platforms are updated.
21. As a student, I would like to have access to a universal feed, that displays all new "events".

Design Outline

High-Level Overview

We plan to use a *client-server model* for our project. The client is the interface where all the relevant information is displayed in a succinct and pleasant manner. Our client will be getting this data by making requests to the server which will respond by sending the data asked for by the client. The server itself will be retrieving the info through calls to various APIs (this is a fundamental process of our service). There will be multiple query calls in the server which will be fetching information from our database as well as updating it.



Components of the system -

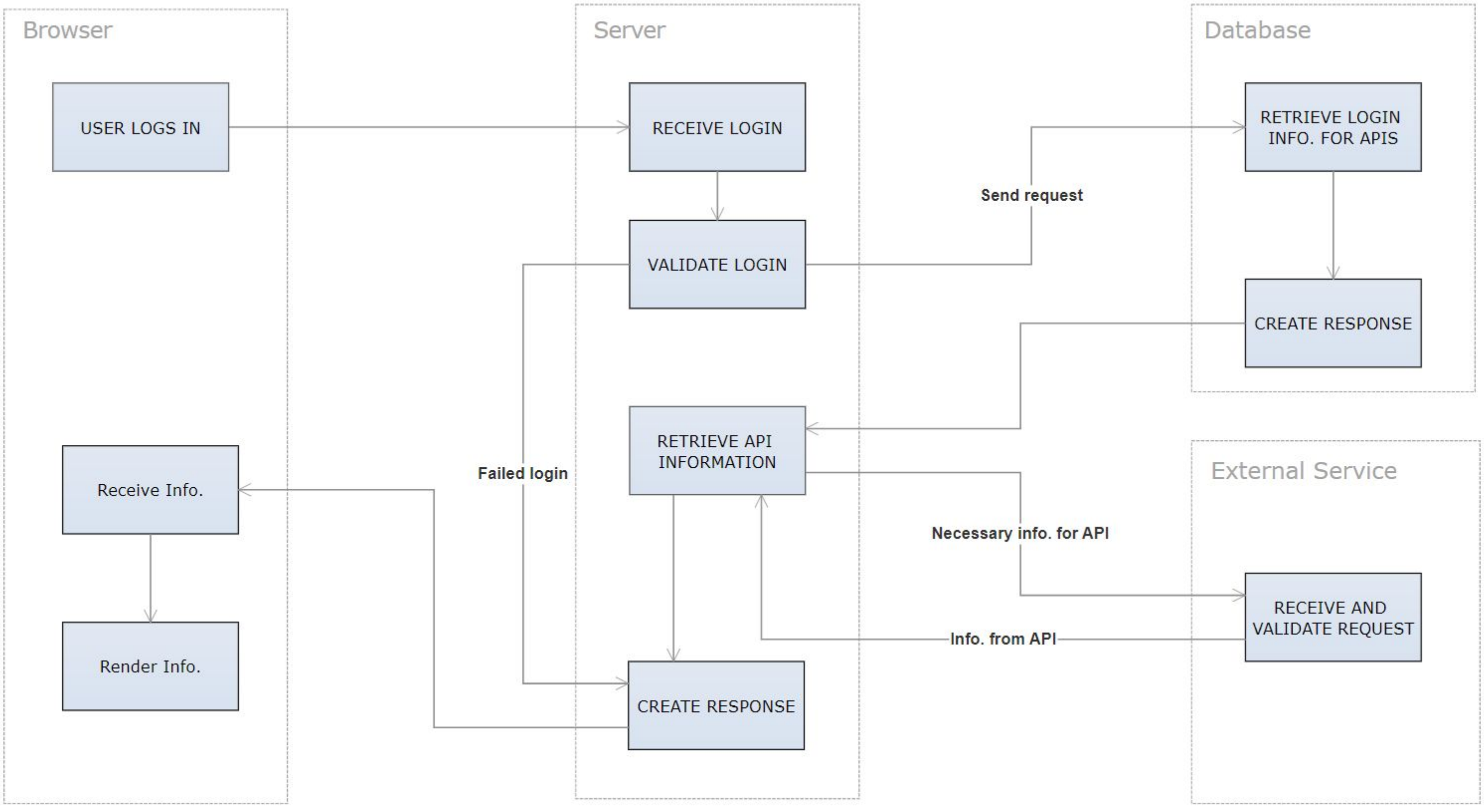
1. Client -
    a. API calls are made to request information from the server
    b. API calls will be made to send information to the server such as the usernames and passwords.
    c. Request to the Server for information will be made when necessary (for example: a request for Blackboard information when viewing a specific course page).
    d. Information is displayed in a manner which is easy to look at
2. Server -
    a. Will take user information from the client, and verify it.

   b. Using user information, will retrieve relevant information from the database: Blackboard Login Info, Piazza Login Info, Email Login Info, UI Customization Variables, Google Login Info, Basic Course Info, WebAssign Login Info, and Social Connections.
   c. Using the relevant information, will make API calls to the External Services to get the ultimate information to be shown to the User.
   d. Lastly, will format and return the information retrieved from the External Services to the Client.

3. Database -
   a. Stores all information that is not held by the APIs that we are calling: Blackboard Login Info, Piazza Login Info, Email Login Info, UI Customization Variables, Google Login Info, Basic Course Info, WebAssign Login Info, and Social Connections.
4. External Services -
   a. Blackboard, Piazza, Office 365 Email, Google Calendar, Google Notes, and WebAssign.
   b. This represents the various different APIs that we will be calling.
   c. Will be called by the Server, which is passing information from the Database.
   d. Each External Service (API) will require a different set of login information.
   e. Each External Service (API) will return a different format of output, to be parsed by the Server.

Flow of Events

The basic operation of our service will work as follows: a User will go to our website, and log in. That Login Information will be sent to the service, which will validate it through comparison with the Login Information that is stored in the Database. If it fails to validate, a response will be created by the server, which will be sent mack to the Client, to be rendered. If the login validates, the server will retrieve all the necessary information (Blackboard Login Info, Piazza Login Info, Email Login Info, UI Customization Variables, Google Login Info, Basic Course Info, WebAssign Login Info, and Social Connections) from the Database. Once it has the necessary information, the Server can retrieve all the information from the External Services (Blackboard, Piazza, Office 365 Email, Google Calendar, Google Notes, and WebAssign) using their APIs. The information from the External Services can then be parsed by the Server, and then a proper response can be formed. That response will be forwarded to the Client, to be rendered. On the next page is a diagram to help visualize this flow of information.



Design Issues

Functional Issues:

Issue: What type of design scheme should be used?

Option 1: Adaptive

***Option 2:* Responsive**

Responsive web design has become the king of all web design formats, and for a specific reason: we want our user to be able to access this application on ALL platforms. Using an adaptive web design requires more work and is, decidedly, less efficient than simply making it responsive with the use of CSS and SASS.

Issue: How interactive should the web and mobile application be?

***Option 1:* Customizable**

Option 2: Minimal Selection (Specific classes + Selecting notifications)

Option 3: Static

When it comes to the user, we want this product to feel as much of a homepage as possible. With that being said, they should be able to customize it to their own aesthetic/style so that when they open it, they feel comfortable in its usability. Within the range of plausibility, the user should be able to customize which resources are in the main page, and how they can be displayed in the form that is most efficient for them. Overall, it should create a user-friendly and playground environment.

Issue: How should users receive notifications?

Option 1: Email notifications
***Option 2:* In-Site badge notifications**
Option 3: SMS messaging

Because we want this site to be one of the users' home pages and/or go-to pages, we want them so see all their notifications within the site itself. If there is a new update, they should be able to click on an icon to view them all. They also will have the ability to filter them depending on what type of notification, and go into their settings to configure what is worthy of a notification or not. Emails can become overwhelming and repetitive, much like how many students feel about Piazza emails.

Issue: How many permissions will students have accessing these resources?

***Option 1:* Reading + Writing**
Option 2: Reading Capabilities Only
Option 3: All of the above + Editing Courses

As a user access all the resources they choose, they should be able to participate in any conversation or activity taking place. This web/mobile application is not meant to act as a portal to the site if they so choose to visit it. This will replace the need of having to visit several sites in order to answer one question or fulfill a request. If it is a simple creation of a post or grade check on blackboard, they will have access. Although, if they would like to configure settings or actions more specific to the website, they will have to go to the actual website instead, and a link will be provided to them so they can do so.

Non-Functional Issues:

Issue: What front end framework should we use?

Option 1: Angular.js
***Option 2:* React.js**
Option 3: Vue.js

In the end, we decided to go with the React library as our front end development tool. It came to be that, within the team, React is the best known tool, and Angular and Vue would take too much time for the majority of the team to learn. React.js is a very simple library to learn, known to be simpler in comparison both Angular and Vue.

Issue: What type of architecture should we use?

***Option 1:* Client-Server Architecture**

Option 2: Unified Architecture

Client-Server Architecture is optimal for the group to split up these tasks and perform efficiently in between sprints. It allows us to implement the front end and back end of this product in a timely and productive manner. Because we will now implement a back end, there will be no cautions or worries with how well React.js will be able to handle these API calls, and how well it will perform while doing so. A Unified Architecture prohibits each team going at their own pace and the formation of the separation that should be there.

Issue: What back end framework should we use?

Option 1: PHP

Option 2: Python

Option 3: Java

Option 4: NodeJS

NodeJS is the framework that our team is most familiar with, and can accomplish the most with. Also, because NodeJS is an ever evolving framework, it creates a space of more innovation and tools that we may need in the future when we progress with this.

Issue: What database software should we use?

***Option 1:* MongoDB**

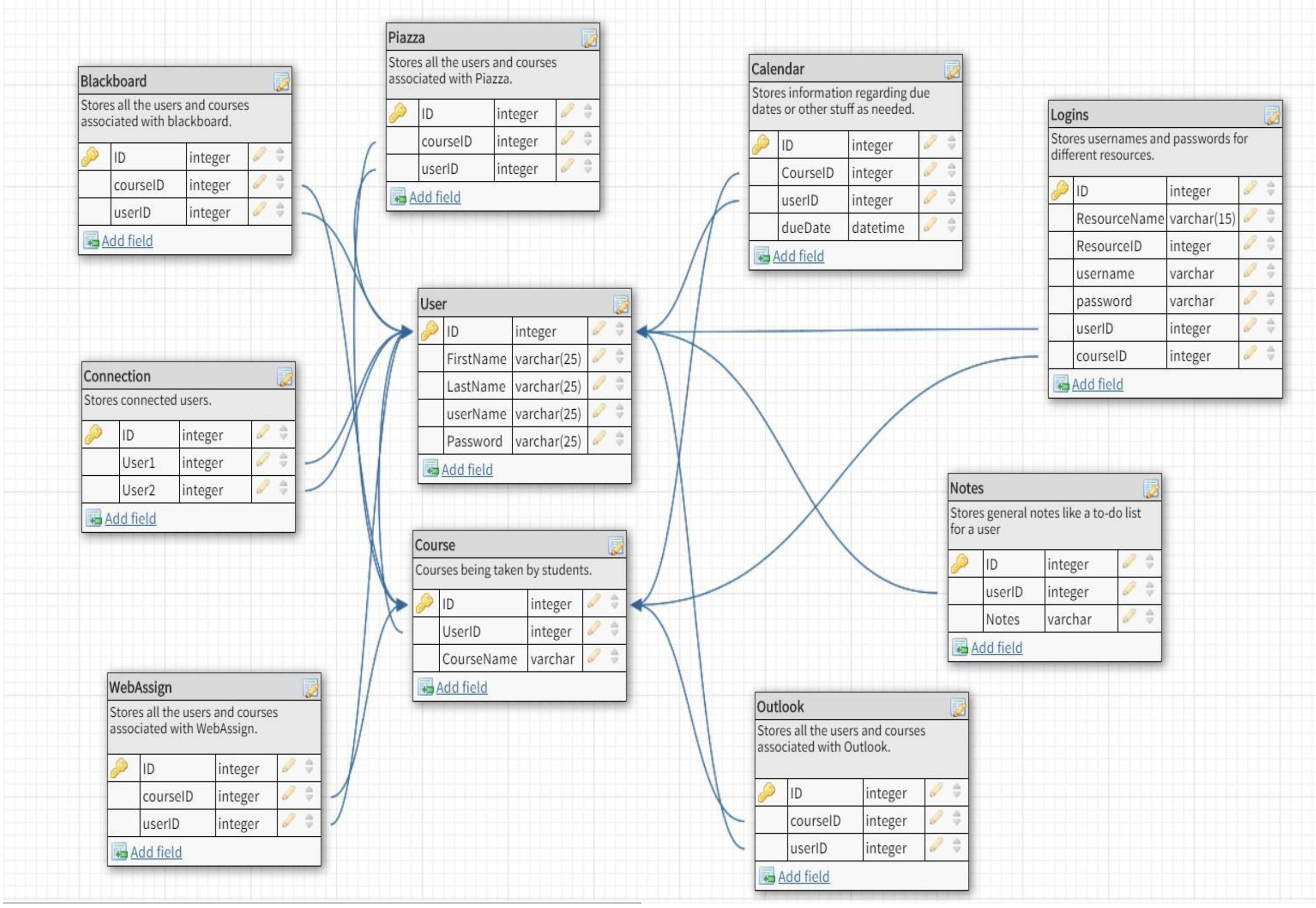
Option 2: MySQL

Option 3: SQLite

MongoD, in this case, has proven to be the most well known database software among the group.Though MySQL is the most popular and widely used, it will take an inefficient amount of time to fully grasp at the extent of the knowledge required for this project. SQLite has little use for us, and the majority of the team is not familiar with the software.

Design Details

Database Schema Mockup:



Blackboard

Stores all the users and courses associated with blackboard.

ID	integer
courseID	integer
userID	integer

Piazza

Stores all the users and courses associated with Piazza.

ID	integer
courseID	integer
userID	integer

Calendar

Stores information regarding due dates or other stuff as needed.

ID	integer
CourseID	integer
userID	integer
dueDate	datetime

Logins

Stores usernames and passwords for different resources.

ID	integer
ResourceName	varchar(15)
ResourceID	integer
username	varchar
password	varchar
userID	integer
courseID	integer

User

ID	integer
FirstName	varchar(25)
LastName	varchar(25)
userName	varchar(25)
Password	varchar(25)

Connection

Stores connected users.

ID	integer
User1	integer
User2	integer

Notes

Stores general notes like a to-do list for a user

ID	integer
userID	integer
Notes	varchar

Course

Courses being taken by students.

ID	integer
UserID	integer
CourseName	varchar

WebAssign

Stores all the users and courses associated with WebAssign.

ID	integer
courseID	integer
userID	integer

Outlook

Stores all the users and courses associated with Outlook.

ID	integer
courseID	integer
userID	integer

Description of the Database Schema:

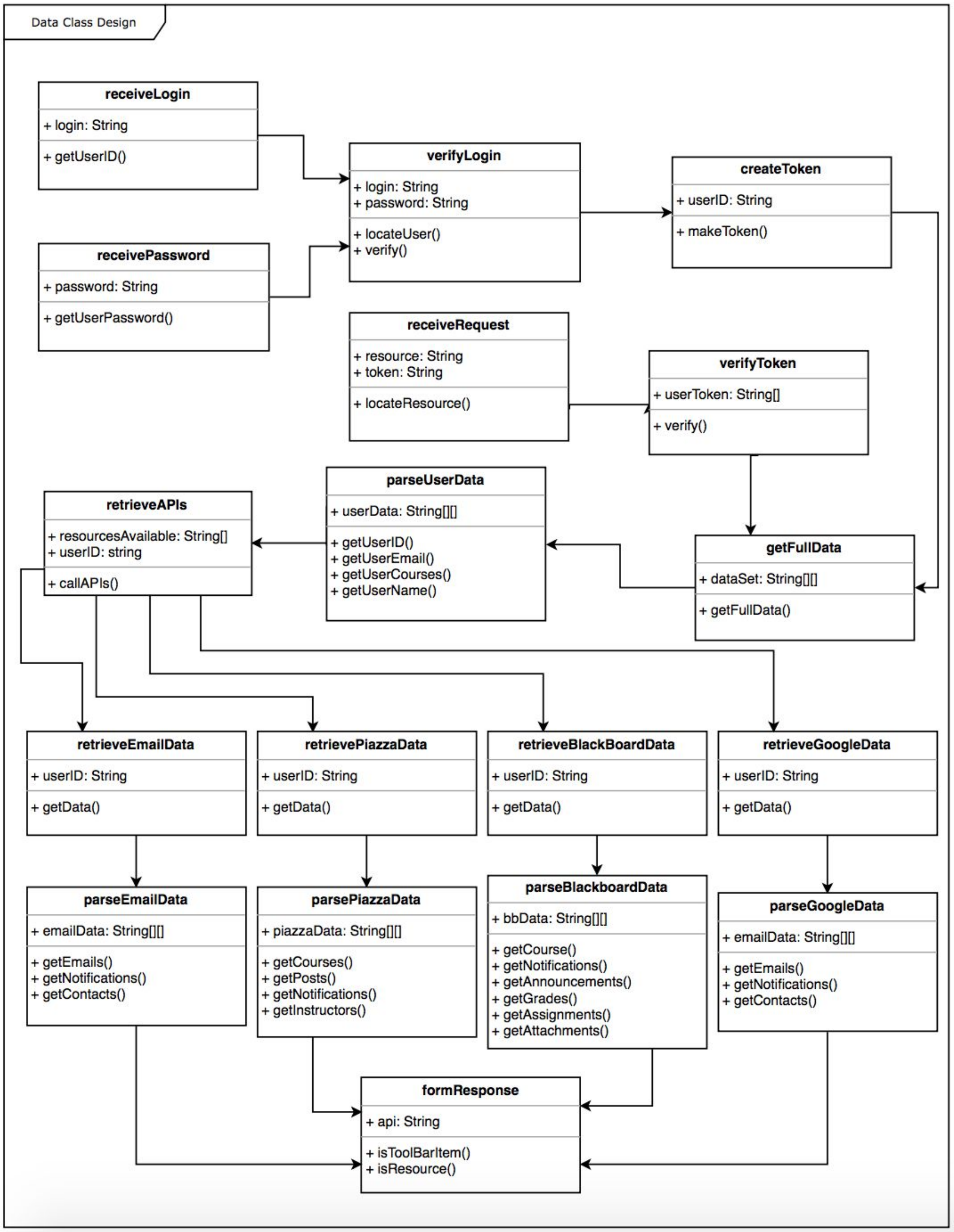
1. User -
    a. The User Table will be used to store all the important information related to the users: first and last names, their usernames that they will be using to login the website, and also the passwords associated with those usernames.
    b. This table acts as a central hub to with which all the others tables will be connected (to their respective Users).
2. Course -

   a. The Course Table is used to store all the courses the users are taking by storing the user ID's and the corresponding course name with its ID.
   b. This table will be to get the courses that one specific user is taking as it has the userID as one of its foreign keys.

3. Blackboard -
   a. This table stores all the Users and Courses related to Blackboard.
   b. We will not save the information behind the Blackboard API, as we are going to be making calls constantly to fetch information rather than storing it in our Database.
   c. Each row will have a user ID and a course ID so that if needed the course names and user info can be retrieved.
4. Piazza, WebAssign, Outlook -
   a. These tables store all the users and courses related to Piazza, WebAssign and Outlook (respectively).
   b. They function similarly to the Blackboard Table. They are named differently as they are the different major resources we are going to be using.
   c. We thought of doing a singular resource table and have a column to identify the resource type, but opted against it as it could slow down the queries if we need more specific info.
   d. If new resources come up in the future, we can make changes to the database to add new tables. But we don't predict there to be much changes as these are the core of our website.
5. Calendar -
   a. The Calendar Table stores information we need to feed the Google Calendar API to make information such as deadlines or help sessions more easily accessible.
   b. This database will be modified based on what information we choose to display on the calendar and whether Google allows for external storage of events (so that we could save them on our Database).
6. Logins -
   a. The Login Table contains usernames and passwords for all the different resources: Blackboard, Piazza, Office 365 Email, Google Calendar, Google Notes, and WebAssign for each user.
   b. This table has a property which identifies what type of resource it is and an attribute telling what User it is associated to in order to get all the login information for the different resources as needed.
7. Connection -
   a. The Connection Table stores all the Users that are connected to each other through our Social Service so that when people friend each other, we can establish a connection between them.
8. Notes -
   a. The Notes Table is a basic table which stores general information for our To-Do list that a User would keep on the website.

Data Class Diagram:

The diagram below is there to help visualize how we will be implementing the Server, to be able to handle logging in, and requesting more information once logged in. Here, we have a depiction of how our classes will be set up that the Java client will use. We also map the structure of the JSON and API routes between the four main APIs we will use, and the login sequence.



Classes that may need definition:

- receiveRequest:
  - Used when the user requests information after logging in
  - Using the Token made during login, they can go straight to getData to begin retrieving the requested information.
- getFullData:
  - This is where we would get information that pertain to the specific User from our Database: the Blackboard Login Info, Piazza Login Info, Email Login Info, UI Customization Variables, Google Login Info, Basic Course Info, WebAssign Login Info, and Social Connections.
  - The response from the Database will need to be parsed, which is why this leads to parseUserData
- receive[API]]Data:
  - All receive[API]Data will have the same function: Pass the correct information to the API, and then receive the response.
  - Each receive[API]Data will be followed by a parse[API]Data to format the response.
- formResponse:
  - Will handle formatting all of the information requested to be able to return a Response to the Client. Will have various forms of Responses for different requests.

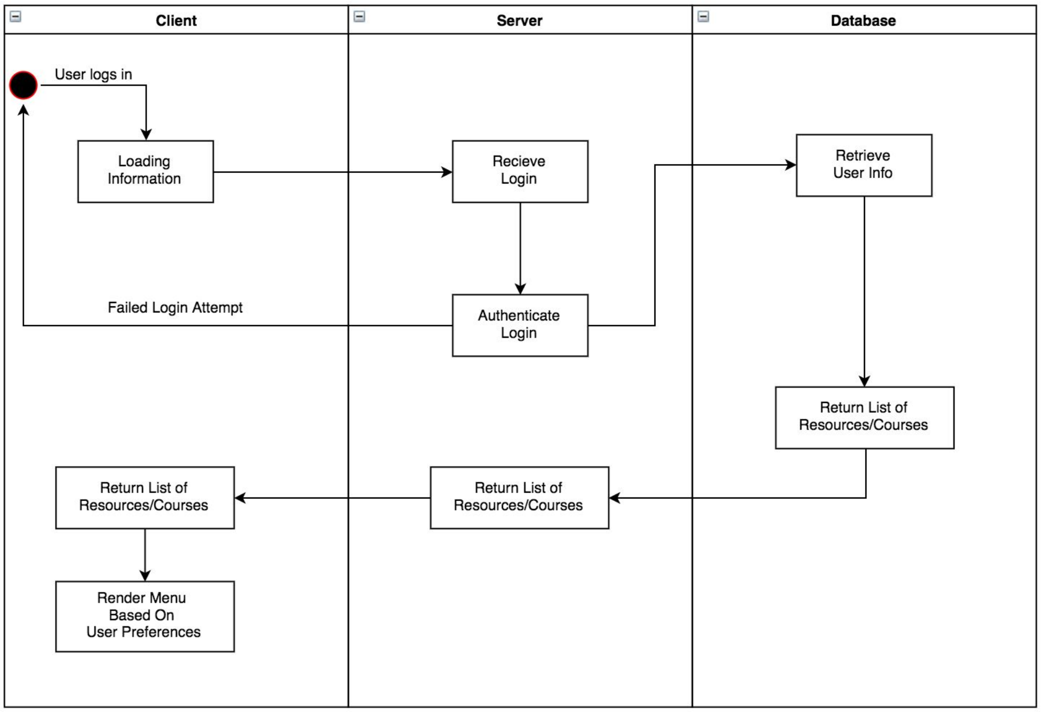
Sequence of Operation Explanation (Sequence Diagrams included below):

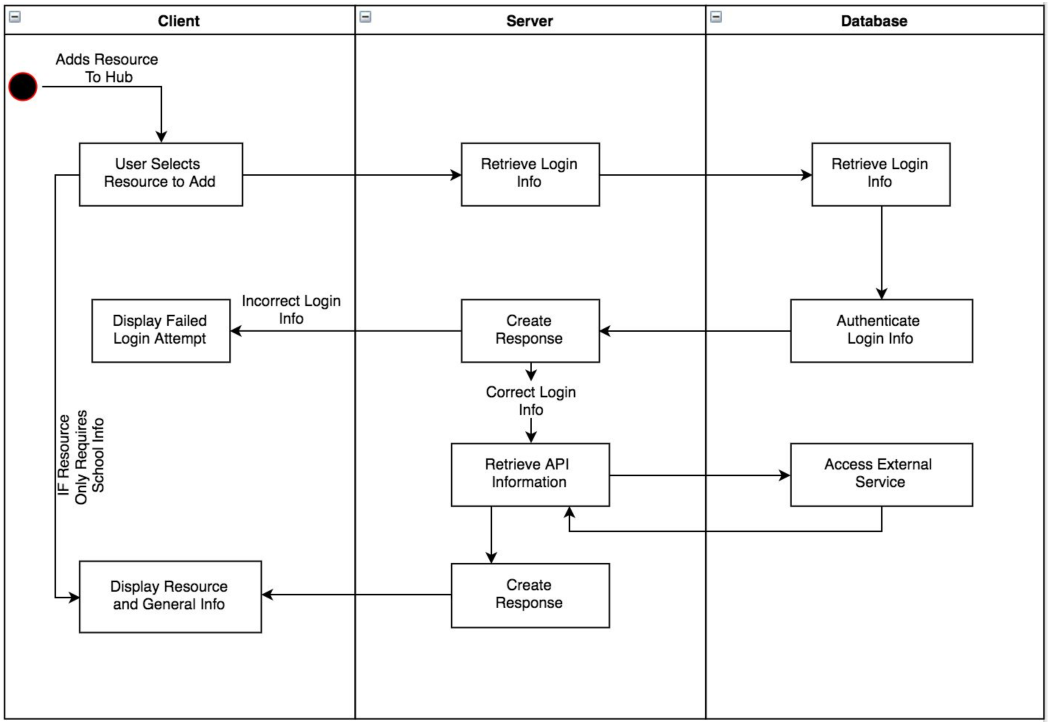
1. A User connects to our website (his device will be the Client Machine).
2. Prompted to login, the User will enter a Username and Password.
3. The Username and Password will be sent in JSON form to the Server, which will receive them.
4. The Server will parse the JSON message from the Client, and send the Username to the Database.
5. The Database will search for the information pertaining to the Username.
6. Parsing the information pertaining to the Username, the Database will return the Password that is saved (the correct password) JSON?
7. The Server will receive the correct Password, and compare it to the Password sent in from the Client.
8. If it does not match, the server will create an appropriate response to an incorrect Password, and send the response back to the Client.
9. If the Passwords match, the Server will again contact the Server with a request, this time to retrieve ALL of the data pertaining to the Username.
10. Also, the Server will send a Token to the Client, allowing it to request more information without having to login again.
11. The Database will return all of the data to the Server.

12. With this data, the Server can parse it and set the proper variables for Blackboard Login Info, Piazza Login Info, Email Login Info, UI Customization Variables, Google Login Info, Basic Course Info, WebAssign Login Info, and Social Connections.
13. The Server will now send out the required information to each External Service, through their APIs (Blackboard login info will be sent to the Blackboard API, Piazza login info sent to the Piazza API, etc.).
14. Each API will return its information to the Server.
15. The Server will parse each of the responses from the APIs, so that they're in a usable state.
16. Using this parsed information, the Server can format a proper response so that the information can be returned to the Client in JSON format.
17. The Client will receive the information, parse what is necessary, and render it.

[Sequence Diagrams placed on next page for continuity]

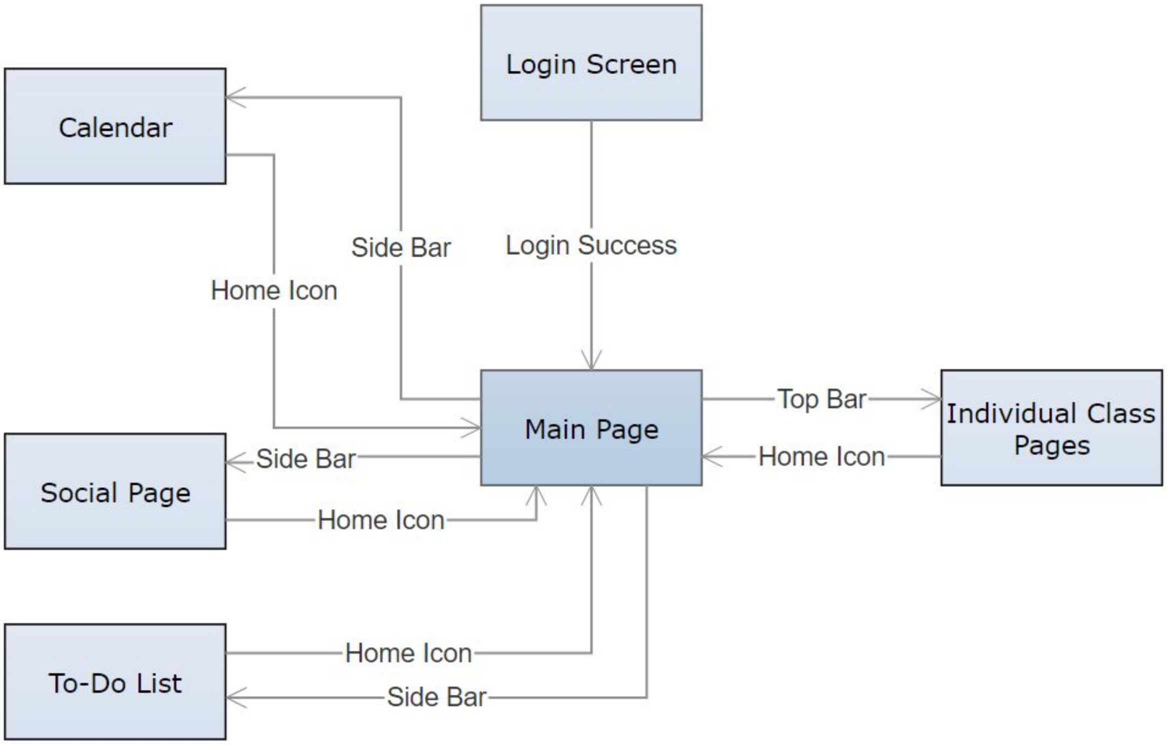
Sequence Diagrams:


Client
Server
Database
User logs in
Loading Information
Recieve Login
Retrieve User Info
Failed Login Attempt
Authenticate Login
Return List of Resources/Courses
Return List of Resources/Courses
Return List of Resources/Courses
Render Menu Based On User Preferences

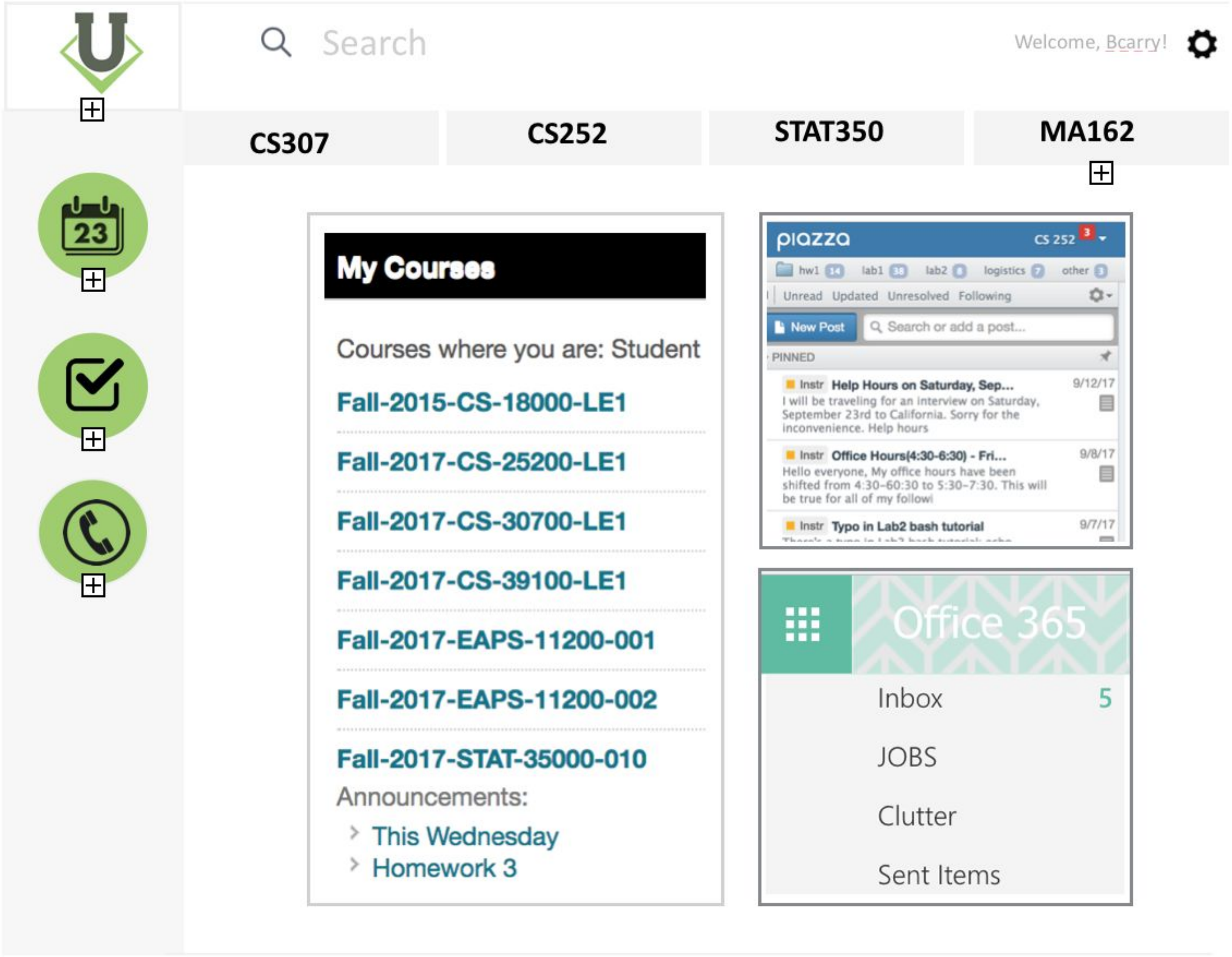


Fundamentals of Navigation:

For our web interface, we plan on implementing it in a very simple and easy-to-use manner. The Login Screen will be the entrance, which will lead the user to the Main Page. This main page will display the most information: All information pertaining to any course, sorted chronologically. Additionally, the Main Page will have linked along both the top and side bar to get to any additional utilities, like the Calendar Page, To-Do List, and Social Page, or narrow down the view to just specific courses. After using the Side or Top Bars, the view will become simpler, with the only non-informational link being back to the Home Page. On the next Page,, we have placed a Flow Diagram to best understand how we plan on allowing the Users to navigate our service.



UI Mockups and Design:



My Courses

Courses where you are: Student

Fall-2015-CS-18000-LE1

Fall-2017-CS-25200-LE1

Fall-2017-CS-30700-LE1

Fall-2017-CS-39100-LE1

Fall-2017-EAPS-11200-001

Fall-2017-EAPS-11200-002

Fall-2017-STAT-35000-010

Announcements:

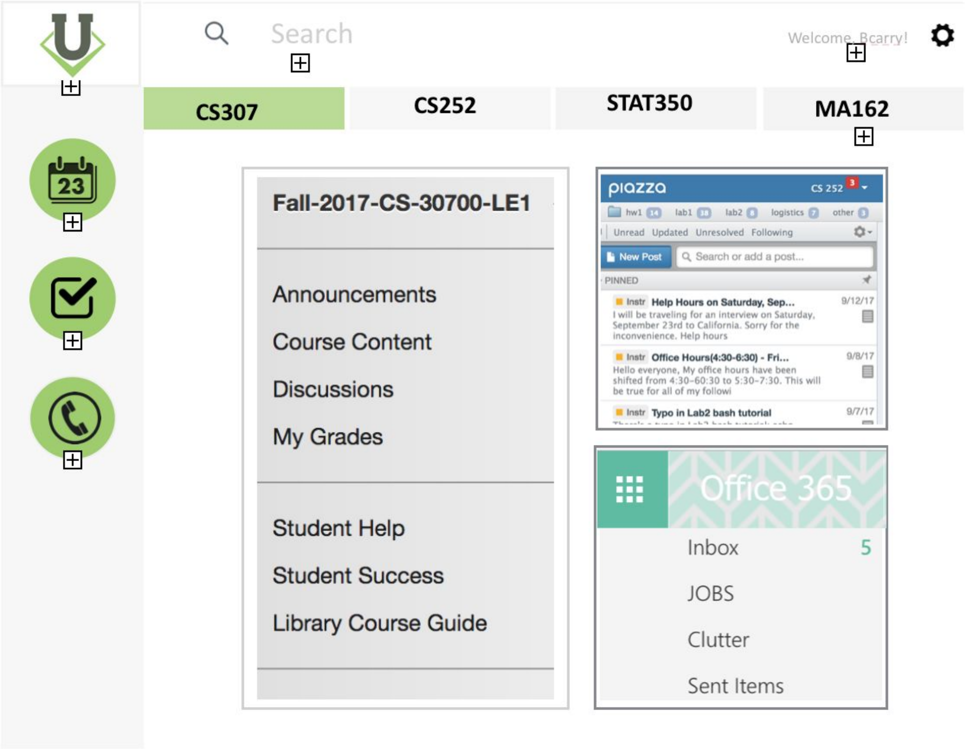
- This Wednesday
- Homework 3

Instr **Help Hours on Saturday, Sep...** 9/12/17
I will be traveling for an interview on Saturday, September 23rd to California. Sorry for the inconvenience. Help hours

Instr **Office Hours(4:30-6:30) - Fri...** 9/8/17
Hello everyone, My office hours have been shifted from 4:30-60:30 to 5:30-7:30. This will be true for all of my followi

Instr **Typo in Lab2 bash tutorial** 9/7/17

The home page will show general announcements, emails, and piazza posts(select content will be used) for all courses. On the left hand size, there will be a menu to access the calendar, to-do list, and social page. Users can search for course material through the search bar and access settings through the button on the top left.

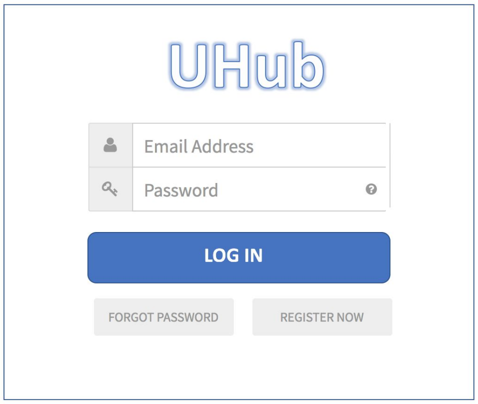


Instr Help Hours on Saturday, Sep... 9/12/17
I will be traveling for an interview on Saturday, September 23rd to California. Sorry for the inconvenience. Help hours

Instr Office Hours(4:30-6:30) - Fri... 9/8/17
Hello everyone, My office hours have been shifted from 4:30-60:30 to 5:30-7:30. This will be true for all of my followi

Instr Typo in Lab2 bash tutorial 9/7/17

After clicking on a specific course, users can see course specific material from blackboard, emails, and piazza(select content).



Log in page provides users with a single login for all platforms