



Team 10 - Design Document

Jacob Brabec, Shane DeWael, Matthew Ess, Jay Hankins, Ankit Patanaik, Kedar Vaidya

Purpose

Students and faculty members alike are required to use different platforms and services to manage, consume, and curate content for courses. Our project aims to unify the features of different platforms, creating an intuitive service that centralizes quizzes, homework, attendance, grades, and relevant course information via a well-designed web application.

Our functional requirements include the following items:

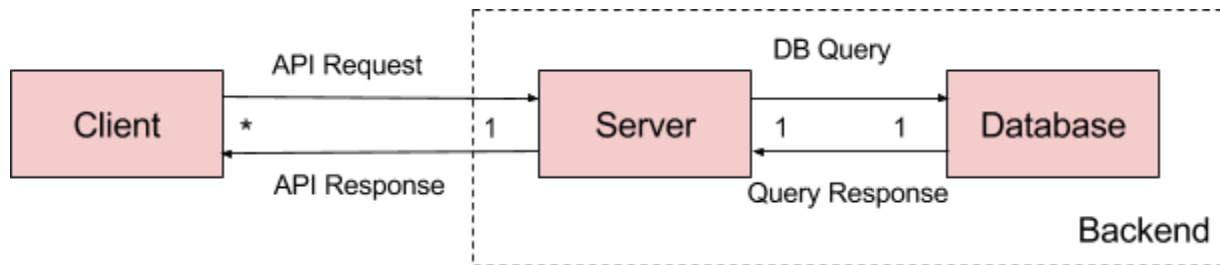
1. Creating a Class
 - a. Any registered user can create their own class. This user will become an administrator for that class.
 - b. The administrative user will create sections for their class. Each class must have at least 1 section, but can have as many as needed. The user is automatically an administrator for all of these sections.
 - c. This user can invite other users to be instructors for any of these sections.
2. Announcements
 - a. Instructors may create announcements for class sections which they are in charge of.
 - b. Announcements will be shown to users on their class home page
3. Assignments
 - a. Instructors may create Assignments for any section which they are in charge of. The instructor can write a description for the assignment, set a due date, and optionally require a file submission.
4. Grades
 - a. Instructors may assign a grade for each student for each assignment within their sections.
5. Quizzes
 - a. Instructors may create a quizzes for any section they own. Instructors can create multiple choice questions for these quizzes, and select which answer is the question's correct answer.
 - b. Students may take the quizzes any time before the deadline. Completing the quiz will automatically add the score to the gradebook.
6. Grade Categories
 - a. Instructors can create grade categories for class sections which they control. Each category can be associated with a grade weighting.
 - b. Instructors may categorize assignments they create into into grade category.
7. Conversation Threads
 - a. Students and instructors will be able to create a conversation thread in any course they are part of.
 - b. Students and instructors can respond to any thread by making their own posts.
 - c. Students and instructors can reply to posts so that the posts are nested.
 - d. Students and instructors can upvote any post once.
8. Instant Polls
 - a. Instructors may create an instant poll for any section they own. Each instant poll can have a short description and a correct answer (A, B, C, D, or E).
 - b. Instructors may start an instant poll at any time, and then close it when finished.
 - c. Students may respond to the poll during the time the poll is open.

- d. Instructors may view the poll responses for all students within their section.
- e. Instructors may reopen old polls if they choose to.

Design Outline

High Level Overview

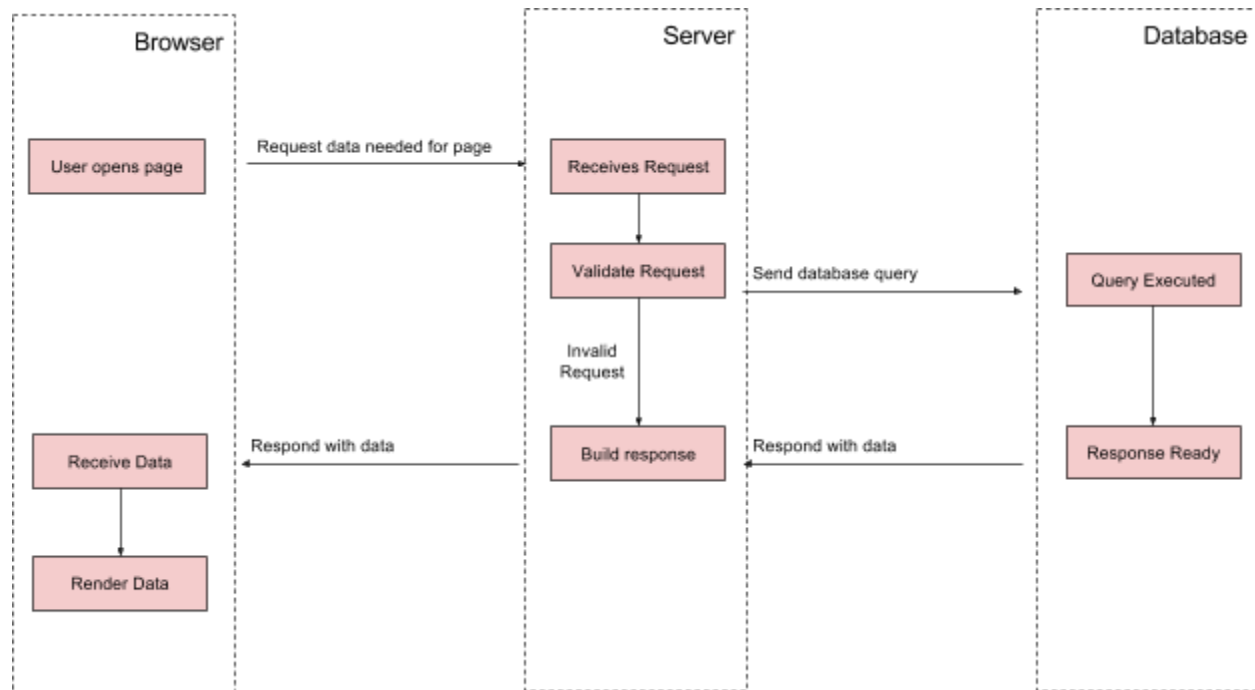
Our project will use the client-server model. The server will respond to the client requests, and the client will parse and render the response data using AngularJS. The figure below demonstrates a high-level overview of the system.



1. Client
 - a. Client sends AJAX requests to the server API
 - b. Client receives an AJAX response from the server
 - c. Response data is interpreted and rendered to user using AngularJS
2. Server
 - a. Receives and handles all client requests
 - b. Validates requests before processing
 - c. Queries the database and generates appropriate response to send to client
3. Database
 - a. Database stores all system information, such as users, classes, assignments, etc...

Flow of Events

The diagram below shows a typical flow of events. The sequence begins with a user opening the web app through their browser. If the client requires data from the server in order to complete the page, the client will make an AJAX request to the server. The server will first validate the request to make sure the requesting user has the required permissions for such a request. If the request is valid, the server will query the database and then generate a response object to return to the server. If the request is invalid, no query will be made, and an error response will be sent instead. From there, the client will parse and render the data onto the page for the user to view.



Design Issues

Issue: What type of architecture should we use?

- **Option 1:** Client-Server Architecture
- Option 2: Unified Architecture

We decided to go with the Client-Server architecture. Separating the application into a front and back end will make it easier to divide the work for our team, and we will be able to make changes to our back end without needing to worry about compatibility issues with the front end. If we were to use a unified architecture, the front end team would be able to work only as quickly as the back end team works. With the client-server architecture, the two teams can work asynchronously, because the front end can be “hooked” into the back end once the back end components have been completed.

Issue: What back end framework should we use?

- Option 1: Python (Django)
- **Option 2:** PHP (Laravel)
- Option 3: NodeJS

We considered using Django for our project, but since none of our team is familiar with it we decided the time investment to learn the framework would not be worth it. We also considered using NodeJS, but realized that NodeJS is still actively changing, and could lead to compatibility issues in the future. We finally settled on using Laravel because of the maturity of PHP and huge amount of resources and support communities online.

Issue: What database software should we use?

- **Option 1:** MySQL
- Option 2: Postgres
- Option 3: SQLite

After carefully comparing the three most popular relational database systems, we decided to go with MySQL. MySQL is a database software that is popular and widely supported by various server operating systems. Postgres is very similar to MySQL in terms of performance and usability, but we are more familiar with MySQL and tools associated with MySQL database management. SQLite is more suitable for embedded applications, and is typically not accessible over a network, which is critical to our web app.

Issue: How should we handle user permissions?

- Option 1: Create a different class for Users, TAs, and Administrators
- **Option 2:** Use a single user class and implement a roles/permissions system

We decided to create a roles/permissions system which we could apply to a generic user class. We realized the first option would be problematic because a single user could be a TA in one class while being a student in another. In order to address this, we would need separate entries in the database for each user in each class. A roles system will allow us to grant users specific permissions but only in specific classes and sections.

Issue: How should we authenticate API requests?

- Option 1: Send username/password on each API request
- **Option 2:** Use JSON Web Tokens for stateless authentication
- Option 3: Authenticate with an OAuth provider such as Facebook or Google
- Option 4: Authenticate with Purdue CAS (Central Authentication Service)

We decided not to authenticate with Purdue CAS because we envision our system being used by any educational institution. A drawback of using a social network OAuth provider is that many people do not want their personal lives to be mixed with their professional lives, so connecting the two could cause friction with our users. We also decided against sending a username and password combination for every API request, because this is an inherently insecure authentication method. JSON Web Tokens, or JWT, quickly became the ideal choice for our application. JWT are compact (little network impact), self-contained (doesn't require repeated database queries for authentication), and secure (signed via HMAC secret code).

Issue: What front end framework should we use?

- **Option 1:** Angular
- Option 2: React
- Option 3: Backbone

These three services provide a relatively similar feature set, but we opted to use Angular instead of its alternatives. The main reason is that several members of our team have extensive experience with Angular, and we know that it will provide the features we need.

Issue: What type of design scheme should we use?

- Option 1: Adaptive
- **Option 2:** Responsive

We want the user experience for Mango to be consistent across all devices. We considered creating a separate version of our web app for smaller screen sizes (adaptive), but decided that creating a single web app with a responsive layout would be ideal. This will allow us to maintain one responsive front end instead of two separate front end applications.

Issue: How should users navigate through their class material?

- Option 1: Allow switching class through a central page
- **Option 2:** Unified sidebar navigation

We opted to have an always-present sidebar that allows users to navigate to any class at any time. We don't want it to take more than a single click to switch between classes. We could have allowed people to switch their classes from a central page, but this would have required a click to return to that page and another to switch classes.

Issue: How should users receive notifications about important class updates?

- **Option 1:** Site based badge/popup notifications
- **Option 2:** Email notifications
- Option 3: SMS alerts

We feel that SMS notifications should be used for urgent notifications, but our service does not have any features which require the user's immediate attention. We decided to offer site-based notifications and optional email notifications. Email notifications are off by default so users don't get annoyed.

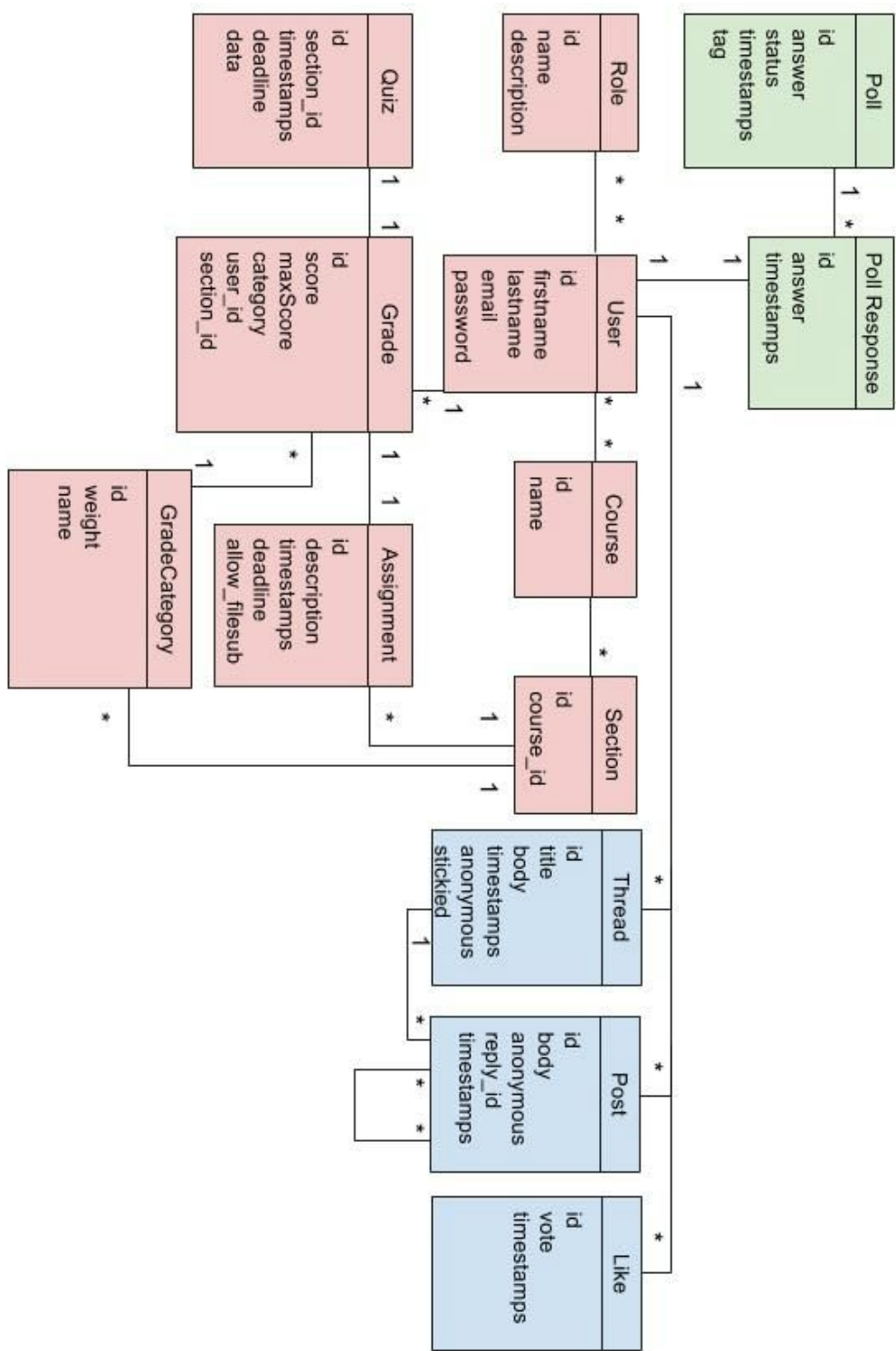
Issue: How should instructors/administrators edit their content?

- Option 1: Via an open-ended edit mode that lets admins place content anywhere they want
- **Option 2:** Via a closed-ended edit mode that automatically organizes content into pre-defined class sections

We decided on option 2 as a way to maintain a cohesive experience across all aspects of the site. An issue we noted with Blackboard is that many classes handle all of their Blackboard content differently. Some dump everything into "Course Content," others divide things up into relevant folders. We will have specific and specialized content sections, including the calendar, grade uploads, assignments, projects, announcements, etc., that are not editable by the instructors/administrators.

Design Details

Database Schema Mockup



Description of Database Tables

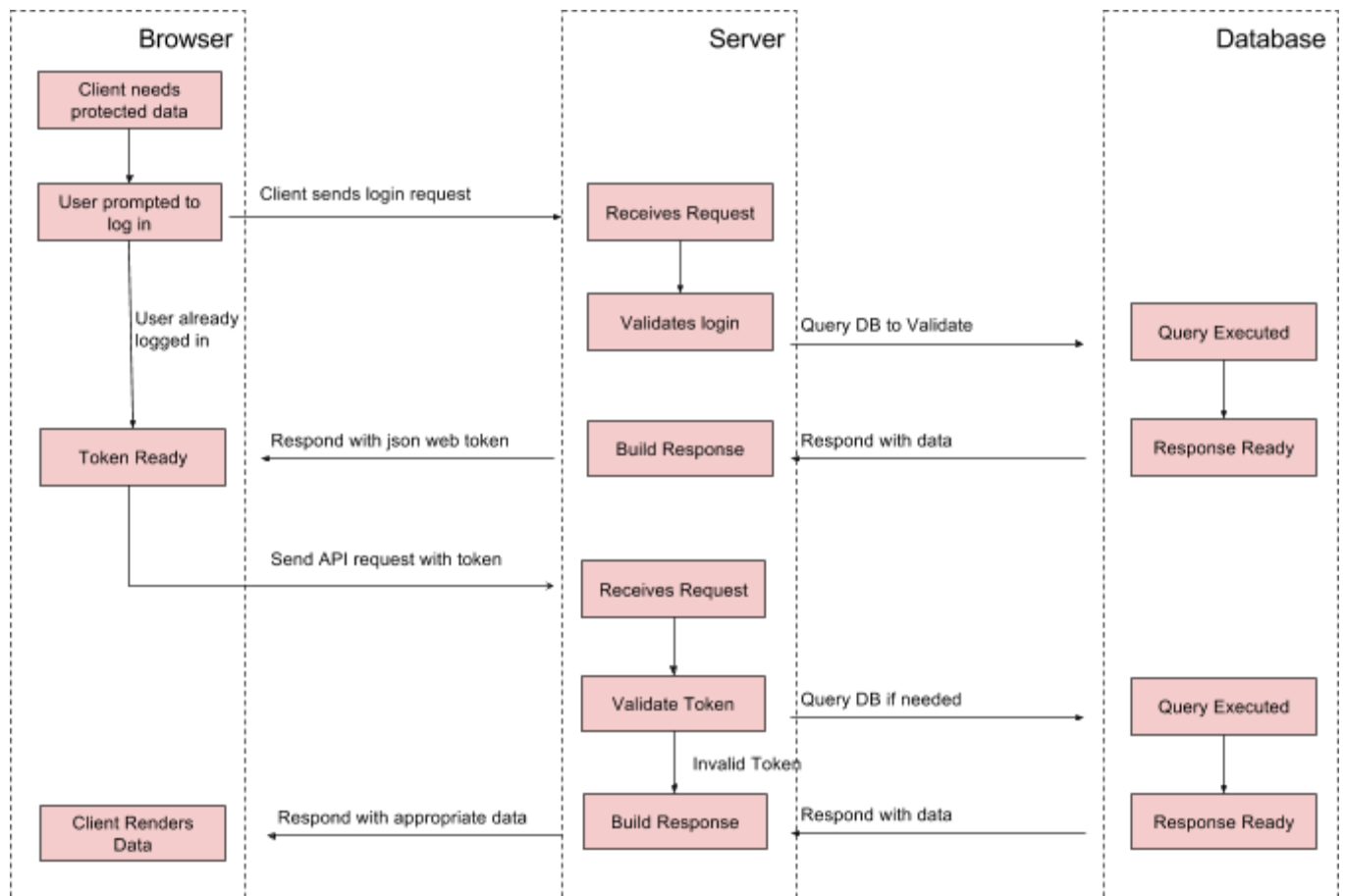
Above is a mockup of our database schema. Although things may change, we believe this design will satisfy the current functional requirements. The red boxes are for our core feature set, the green for in-class polls, and the blue for our question/answer forum.

1. User
 - a. All users are represented as part of this class, including students, instructors, and admins.
 - b. Contains information about user such as email, password, and name
2. Course
 - a. A course contains only a name.
 - b. This class exists mostly to organize various sections and provide some common information between them.
3. Section
 - a. A section points to exactly one course.
4. Announcement
 - a. Announcements represent a message that an instructor wants to send to their students. An announcement can be made to an entire course or to just one specific section.
5. Assignment
 - a. Assignments are created whenever an instructor wants to assign a grade for some item.
 - b. Assignments can also be used to receive file submissions, such as an essay, from users.
6. Grade
 - a. A grade will be associated with exactly one student. It will also be associated with either an Assignment or a Quiz. It contains the raw score a student received on that particular activity.
7. Quiz
 - a. A quiz can be created by an instructor to give to their students. The quiz is linked to exactly one course section.
 - b. The quiz class contains JSON data which represents the questions and the correct answer for each question.
8. Grade Category
 - a. An instructor may create grade categories that they can use for assignments or quizzes. Categories may be weighted (e.g. Exams are worth 90% of the grade while homework is only 10%).
 - b. Each grade category can be associated with many assignments and quizzes.
9. Role
 - a. Every user will be assigned a student role upon class signup. If a user joins a class as an administrator or TA, a new role will be assigned to the user for that specific class. This allows us to ensure permission encapsulation within each course.
10. Thread
 - a. Members of a class can create threads in their section of the Q&A platform. A thread can receive likes and responses. This feature is intended to allow rich content, such as embedded videos, pictures, and documents.
11. Post

- a. A post is a response to a thread or parent post. This allows for comments to “branch” out, which can lead to an engaging conversation without distracting from the other parent posts.
12. Like
- a. Users may like other users’ posts. Doing this creates a Like entry in the database, which associates the user who liked the post with the post itself.
13. Poll
- a. Instructors may create a poll to give students. Polls work like iClicker questions, and accept multiple choice input from students. Polls only store the correct answer, not the actual question text.
 - b. Polls are associated with one section.
 - c. Polls contain a status field to track whether they are inactive, open, or completed.
14. Poll Response
- a. If a poll is open, students may respond to it. This response is recorded as a Poll Response entry.
 - b. Each poll response is associated with one user and one poll. The response contains the multiple choice answer that the student selected.

JSON Web Token Authentication

We plan to use JSON web token (JWT) authentication to validate requests made to our API. A diagram of the model we intend to use can be found below.



The JWT authentication system will work as follows:

1. When the client needs to access protected data, it will prompt the user to log in
2. Once the user enters their details, the client will send a login request to the server
3. The server will validate their username and password with the database.
4. If the login was valid, a JSON web token will be generated and sent back to the client.
5. If the login was invalid, an error will be sent to the user
6. Future requests to the protected API endpoints must have a token sent with the request. If the token is valid, the request will be processed. If the token is invalid, an error will be sent instead, prompting the user to login.

Navigation Flow Map

Our navigation emphasizes ease of access. The sidebar allows users to access any course landing page from any other page of the app. The homepage gives users one click access to each of their course's 3 main modules (Content, Discussion, and Attendance) as well as their upcoming assignments. The top bar gives one click access throughout the app to the master calendar, alerts, and user profile info. Assignment details are easily accessed by priority; assignments due soon are available on the homepage and in the alerts drop down, other assignments can be reached by their entries in the Course Content pages.

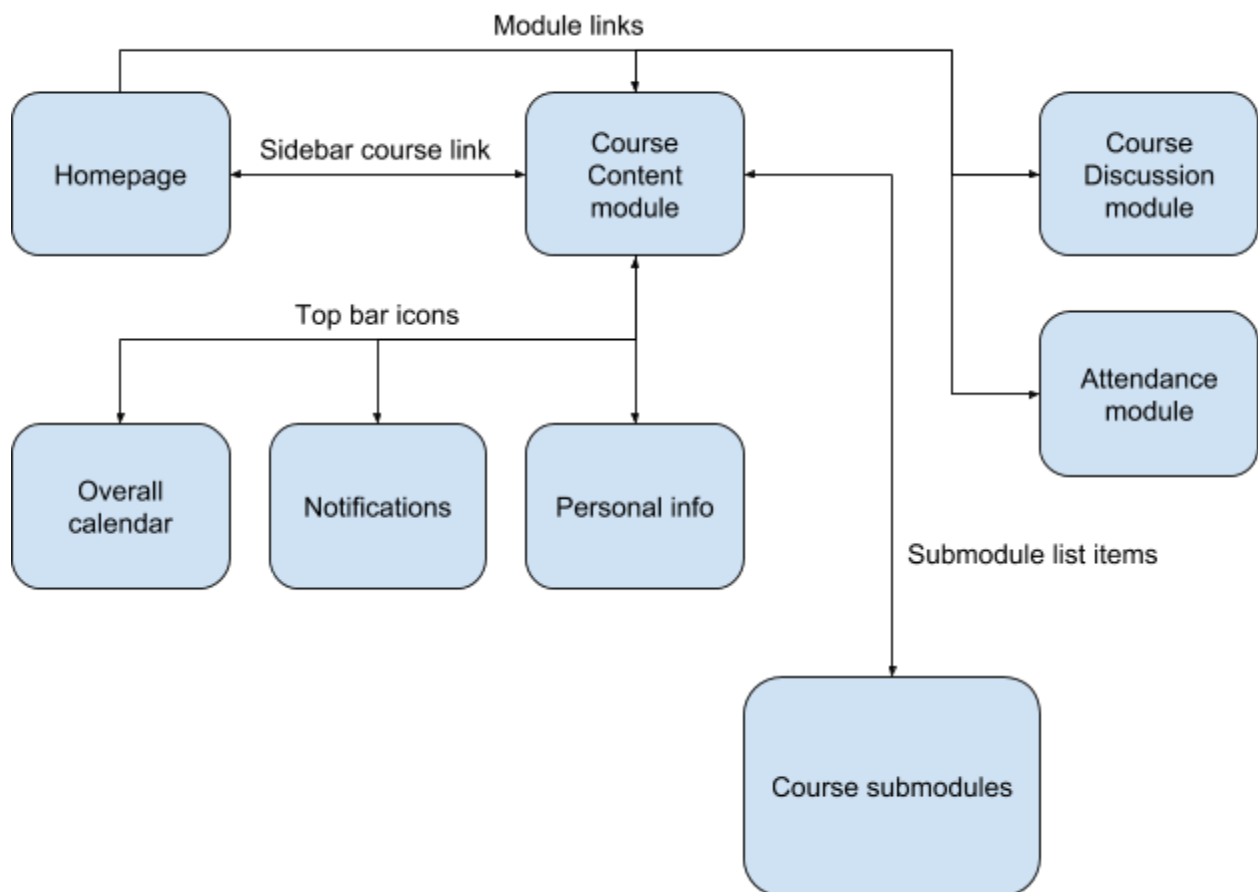


Diagram illustrating ease of access. All modules for all courses are accessible from the home page, and each Course Content submodule doesn't prevent the user from reaching other submodules or returning to the home page.

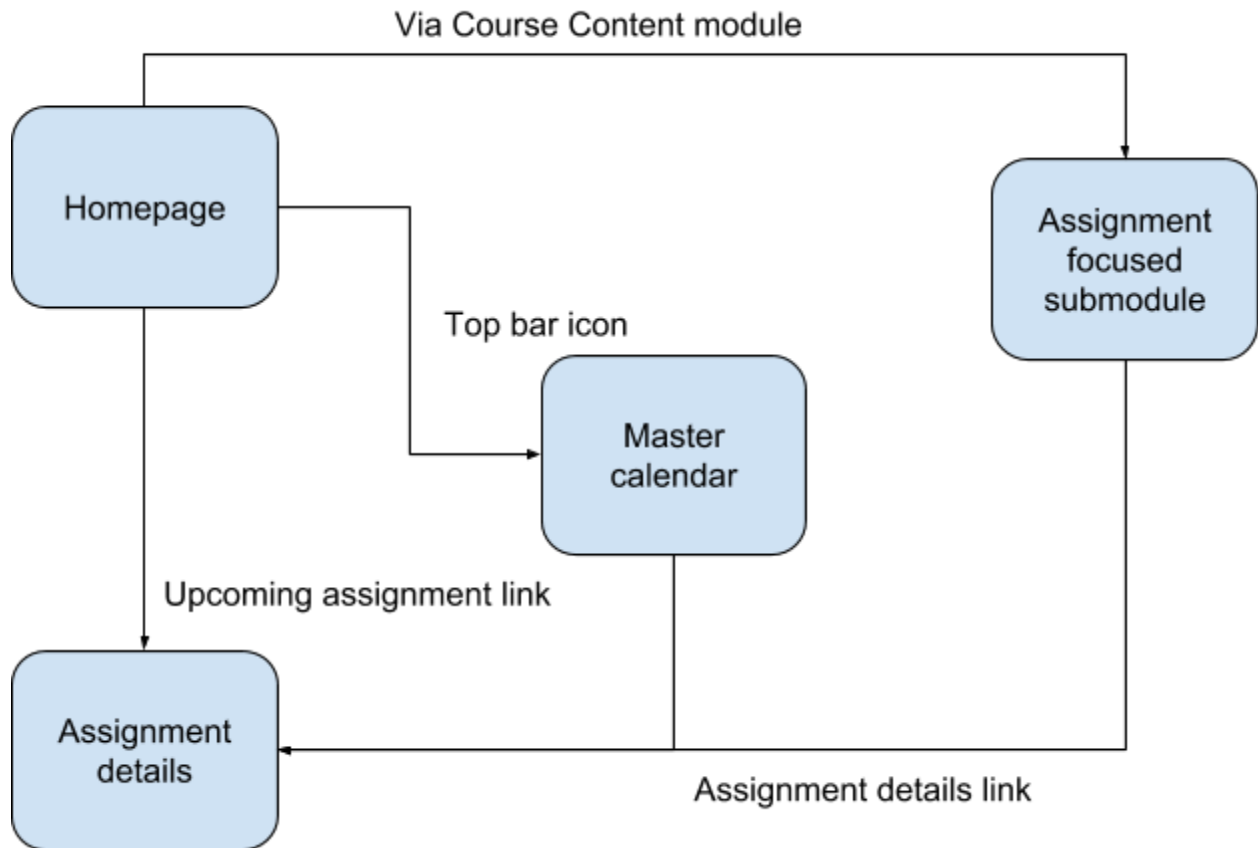
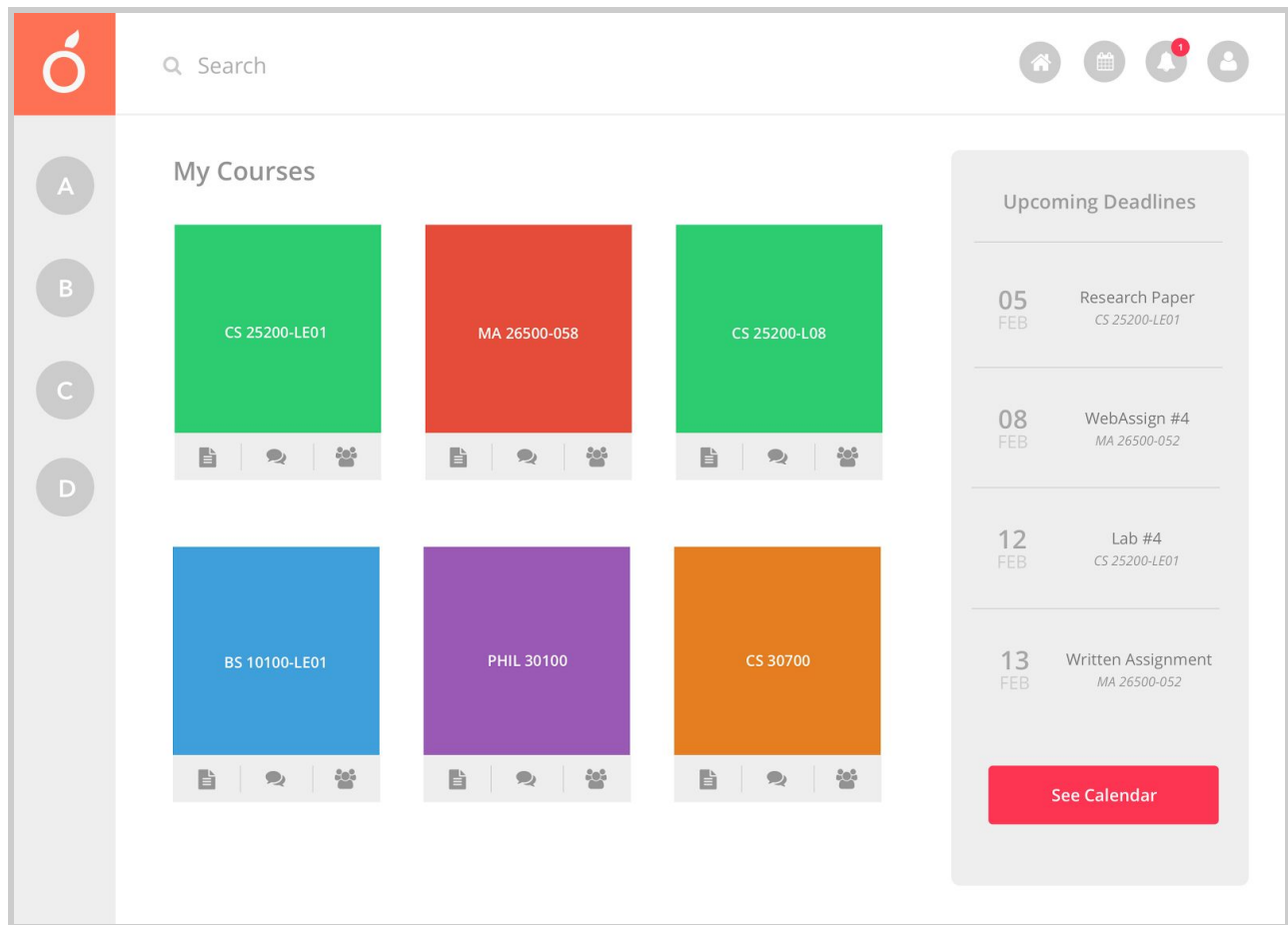


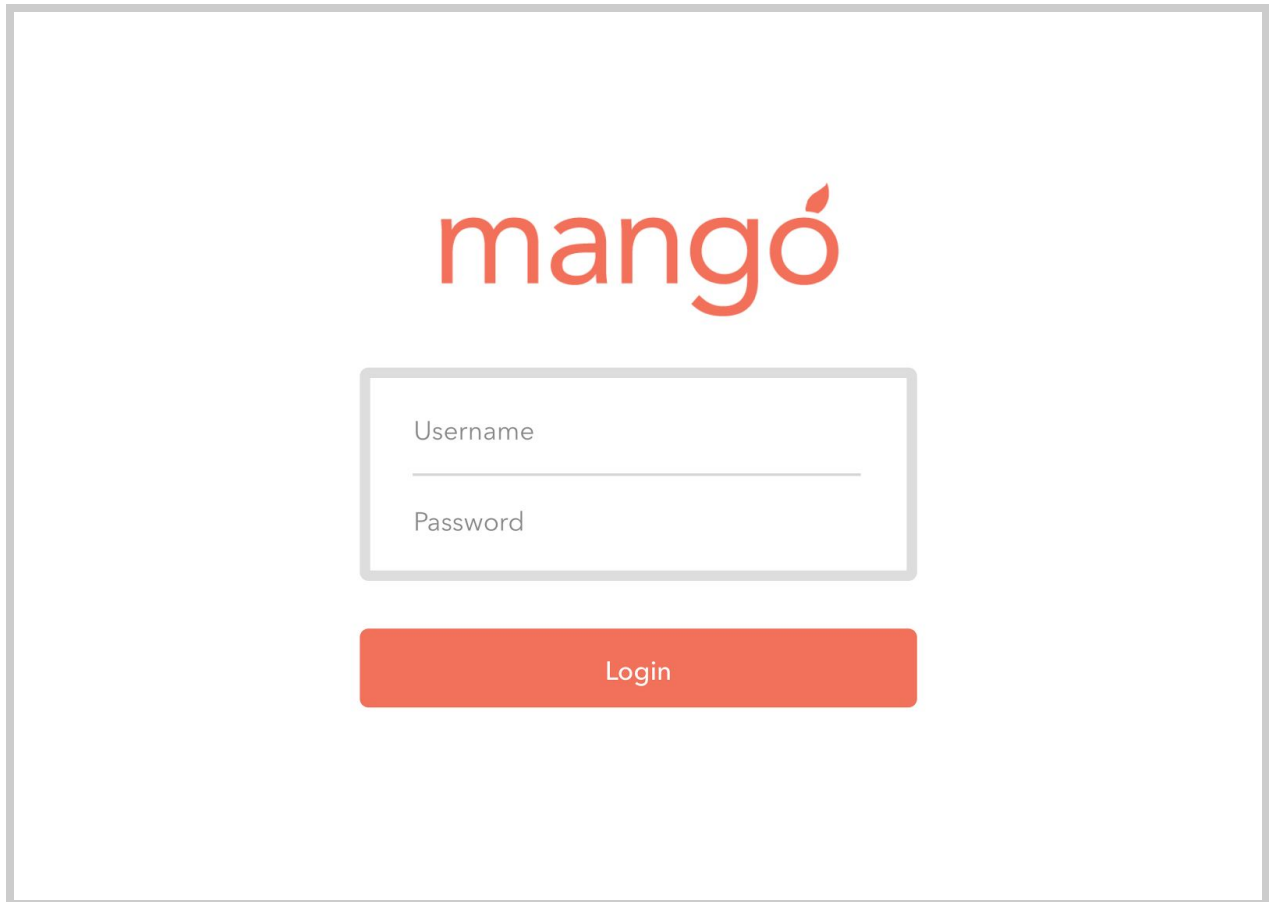
Diagram illustrating how assignment details are accessed. High priority assignments are accessible from the homepage, all others are accessible via a number of avenues (Course Content submodules, calendar).

UI Mockups

We designed the UI to have consistent elements across the entire site. An example of this is the sidebar, which allows users to quickly navigate between classes. Another element that is persistent is the top bar, which allows users to search across the site, access their home page, user page, calendar of assignments and events, and their notifications. The rest of the UI is clean and has consistent colors across the site to allow users to instinctively navigate through new elements and pages of the application.

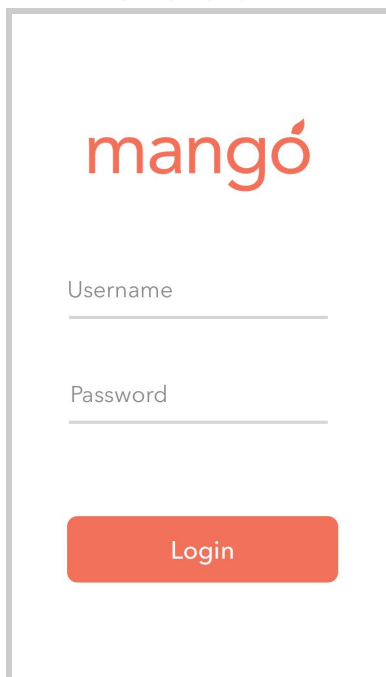


This is the landing page after you login. It includes links to all of your courses, with each course containing course content, course discussion, and course attendance. The icons on the sidebar will represent courses.




A desktop login page mockup. At the top center is the 'mangó' logo in a red, lowercase, sans-serif font, with a small red leaf icon above the 'ó'. Below the logo is a light gray rectangular box containing two input fields. The first field is labeled 'Username' and the second is labeled 'Password'. Below these fields is a solid red rectangular button with the word 'Login' in white text.

The desktop login page



A mobile login page mockup. It features the 'mangó' logo at the top. Below the logo are two input fields, one labeled 'Username' and one labeled 'Password'. At the bottom is a red rectangular button with the word 'Login' in white text.

The mobile login page




CS 25200-LE01


Course Discussion

Attendance


Course Content




Announcements




Assignments




Quizzes



Grades




Calendar







Resources

Contact your instructor



Search

Announcements


05 FEB

EAPS 12000 is a second eight week course, which means that all of the course deadlines are in the second half of the semester. However, about 50% of the course work can be done in the first half of the semester if you want to spread this course over the whole semester. EAPS 12000 is an entirely online course, which means there are no class times or locations. Part of the course blackboard site is already open so that you can see the syllabus, deadlines, and some basic information, and so that you have access to the syllabus quiz and the reading assignments and quizzes connected to the textbook. But you do not have access to the majority of the course at this time.

02 FEB

Links to the course eText and Quizzes / Reading Assignments are now available through the course blackboard site. I would encourage you to buy access to the eText and Quizzes / Reading Assignments through the course blackboard site (total cost for both is \$74.95). The reading assignments and quizzes are individual work that you can do ahead of the official opening of the course if you want to spread the course work more evenly over the semester. More details are given on the course blackboard site.

This is the announcements page, as well as the page users land on when first navigating to course content. For any class, the user can see recent announcements made by the professor.



A

B

C

D

CS 25200-LE01

Pinned Posts

Industry days poster

Please find below the industry days poster I showed at the beginning of today's class. I strongly encourage you to speak to the company representatives...

Recent Posts

Question For Files in reports_dir

Before I start to create file for the process, should I first delete all the files in reports_dir (which is refer to the wat wat)...

What Pid to monitor

I used ps -u \$USER and tried many whatever I don't really care what this text says. Pid to monitor yey lorem ipsum more lorem...

What Pid to monitor

I used ps -u \$USER and tried many whatever I don't really care what this text says. Pid to monitor yey lorem ipsum more lorem...

What Pid to monitor

I used ps -u \$USER and tried many whatever I don't really care what this text says. Pid to monitor yey lorem ipsum more lorem...

What Pid to monitor

I used ps -u \$USER and tried many whatever I don't really care what this text says. Pid to monitor yey lorem ipsum more lorem...

What Pid to monitor

I used ps -u \$USER and tried many whatever I don't really care what this text says. Pid to monitor yey lorem ipsum more lorem...

What Pid to monitor

I used ps -u \$USER and tried many whatever I don't really care what this text says. Pid to monitor yey lorem ipsum more lorem...

What Pid to monitor

I used ps -u \$USER and tried many whatever I don't really care what this text says. Pid to monitor yey lorem ipsum more lorem...

Search

hw1 21

hw2 9

lab1 31

New Post +

↑

1

↓

What Pid to monitor

Posted by **Some Student** on Jan. 11

I used ps -u \$USER and tried many PID to monitor and tried to access their stat file, but I always got permission denied, I have also done chmod +x on it.

The command that is in my bash to checkout the stat and is giving the permission denied:

echo \$(/proc/\$PID/stat)

lab1

↑

0

↓

To get the info of a process, you need to have access to it, so it has to be a process that's running in your user directory. If you run top, you'll see the list of all processes, and in each row the originator of the process. You can pick one from that list, or just run the consumer script and use that ;)

Posted by **Some Student** on Jan. 11

↑

0

↓

To get the info of a process, you need to have access to it, so it has to be a process that's running in your user directory. If you run top, you'll see the list of all processes, and in each row the originator of the process. You can pick one from that list, or just run the consumer script and use that ;)

Posted by **Some Student** on Jan. 11

This is the course discussion page. This has a list of posts on the left while allowing the user to view the content and replies on the right. This page allows for users to continue to easily switch between classes by utilizing the sidebar on the far left.