Interactive Learning Tool for Large Size Lectures

DESIGN DOCUMENT

Team: 40
Client: Md Maruf Ahamed
Advisers: Md Maruf Ahamed

Team Members/Roles:
Backend: Tyler Miller, Alex Swenson, Guan Lin
Frontend: Adam Walters, Jaden Ciesielski, Brandon Burt
Team Email: sdmay23-40@iastate.edu
Team Website: https://sdmay23-40.sd.ece.iastate.edu

Revised: 12/1/2022

Executive Summary

Development Standards & Practices Used

IEEE 1016-2009: Software Design Description
IEEE 1028: Software Review
ISO/IEC/IEEE 26515:2018: Developing information for users in an agile environment
IEEE 9274.1.1- JavaScript Object Notation (JSON) Data Model Format and Representational State Transfer (RESTful) Web Service for Learner Experience Data Tracking and Access
IEEE 7002- Standard for Data Privacy Process

Summary of Requirements

- Design a simple, easy to use interactive learning web application for large scale classrooms.
- Allow professors to post polls, create and participate in discussions during and outside of lectures, view student engagement statistics and control permissions of students and TA's.
- Allow TA's to participate in discussions, view student engagement statistics and perhaps have the ability to create polls/discussions according to professor permissions.
- Allow students to ask questions, participate in polls and engage in discussions anonymously or by name.
- All discussions and results of polls will be recorded to enable users to go back to previous posts

Applicable Courses from Iowa State University Curriculum

ENGL 150/250 - Discussions, weekly assignments, large classes

MATH 165/166 - Problem solving questions, weekly assignments, large classes

CHEM 167 - Large class discussions

New Skills/Knowledge acquired that was not taught in courses

- Node development
- Infrastructure
- Express.js
- Docker
- Websockets
- Webpack

Table of Contents

1 Team

1.1 Team Members

Brandon Burt

Jaden Ciesielski

Adam Walters

Alex Swenson

Tyler Miller

Guan Lin

1.2 Required Skill Sets for Your Project

Web Development in a React Environment.

UI/UX skills for the frontend design.

Express JS for the backend.

SQL for the database.

1.3 Skill Sets covered by the Team

Web Development in a React Environment. (All of us)

UI/UX skills for the frontend design. (Adam, Jaden, Brandon)

Express JS for the backend (Tyler, Alex, Guan)

SQL for the database. (Tyler, Alex, Guan)

1.4 Project Management Style Adopted by the team

Agile Development

1.5 Initial Project Management Roles

Adam - Frontend Lead

Jaden - Frontend Developer

Brandon - Frontend Developer

Tyler - Backend Lead

Guan - Backend Developer

Alex - Backend Developer

2 Introduction

2.1 Problem Statement

There is a long standing problem in large lectures for both students and professors. The problem is a lack of effective communication and interaction between the students and the professor. Many if not all of us have been a part of lectures where the professor will lecture for 50 minutes, and occasionally stop and ask if there are any questions, but for the most part the professor is met with silence. Another problem is some lecture halls are so large it is just too hard to hear a conversation between the back of the room and the front of the room where the professor is. This problem occurs in all large size lecture halls across campus and most of us are familiar with these kinds of situations. This problem is important because student professor communication is very important. Not only does communication allow the professor to know what students are struggling with, it also allows for meaningful discussions and clarification on important topics. Many students feel as though they can't ask questions for fear of being perceived as dumb or stupid which is why they stay silent. To solve these problems we will be developing an interactive learning tool to bridge the gap between students and professors. Our application will have chat rooms for different courses led by a professor. In these chat rooms the students will be able to DM each other, raise their hand, anonymously ask questions, and answer in class polls created by the professor. By allowing students to raise their hand it will give the professor a notification that there is a question which is much easier than scanning a lecture hall of 300 for a physical hand. To combat the problem of shy students, the anonymous chat feature will allow students to anonymously ask questions and allow them to feel more comfortable in the classroom. The polling feature will allow the professor to engage with the students more and periodically check if the students are understanding the day's lecture. The chat rooms will be open 24/7 and students will be able to communicate during class as well as outside of class. This application aims to bridge the gap in communication for large size classes.

2.2 Intended Users and Uses

Persona 1: StudentA

- *Characteristics:*
  - Doesn't feel confident raising their hand in classes with over 25 people
- *Personality & emotions:*
  - Shy
  - Quiet
  - Tired
- *Motivations:*
  - Wants questions to be heard in class
  - Wants to participate in class discussions
  - Wants to feel involved

Persona 2: StudentB

- *Characteristics:*
  - Possibly missed a class (or more)
  - Possibly wasn't paying attention in class
- *Personality & emotions:*
  - Busy
  - Overwhelmed
- *Motivations:*
  - Wants to review what was covered in class
  - Wants to gauge the understanding of other students (based on questions)
  - Wants to get caught up in the class

Persona 3: ProfessorA

- *Characteristics:*
  - Professor teaching a large classroom
  - Cannot hear people from the back of the classroom
- *Personality & emotions:*
  - Intelligent
  - Knowledgeable
- *Motivations:*
  - Wants to have a record of students who participate
  - Wants to encourage every student to ask questions
  - Wants to feel inclusive

Persona 4: ProfessorB

- *Characteristics:*
    - Professor teaching a large classroom
    - A new professor or new to teaching large classrooms
- *Personality & emotions:*
    - Intelligent
    - Overwhelmed
    - Nervous
- *Motivations:*
    - Wants to know how well the students are understanding their lectures
    - Wants to get feedback of their teaching from students
    - Wants to improve their teaching techniques with the use of polls/questions

Persona 5: TA

- *Characteristics:*
    - Assistant teaching large classes
    - Has little to no assistant teaching experience
- *Personality & emotions:*
    - Nervous
    - Overwhelmed
- *Motivations:*
    - Wants to help students that are struggling
    - Wants to know which students need more help/ explanations
    - Wants to feel helpful
    - Wants to take student participation into account when grading

2.3 Requirements & Constraints

Functional Requirements:

- Have the ability to run on different browsers and OS
- Code should be testable
- Code should be well documented
- Students are able to perform a hand-raise after entering in a brief question/comment
- Professors are notified whenever a hand-raise occurs during lecture
- Professors are able to clear a hand-raise after it is dealt with
- Students are able to post questions with file attachments under a course
- Professors are able to post polls under a course
- Professors are able to post announcements with attachments under a course
- All users are able to reply to post discussions, if open

- Students are able to respond to polls
- The system displays poll results after a poll closes
- Professors and TAs are able to view detailed poll participation after a poll closes
- Students are able to perform all actions anonymously to other students
- Professors are able to reveal the identity of anonymous students
- All users are able to view every existing post under a course
- Professors and TAs are able to delete posts and replies
- All users are able to view course participation statistics, including the most frequent posters
- All users are able to change their password

Resource Requirements:

- An internet-accessible server with access to data storage
- Enough server capacity to able to handle up to 500 concurrent users (Iowa State's largest lecture hall having a capacity of 431) (Constraint)

Aesthetic Requirements:

- Should look like a simple tool to use (complexity is hidden to users unless enabled)
    - Adequate spacing between posts
    - Only a small number of options to interact with a post
- Options and information should only be visible if contextually applicable and significant (e.g. posts' discussions should not be visible when looking at a list of all posts)
- Minimal, simplistic, clean interfaces

User Experiential Requirements:

- Design should be intuitive and easy for users to use
- Should give the impression of having all functionality in one central location, but without an overwhelming number of functions
- No downtime/long loading time between posting and seeing the post
    - Meaning quick updates to avoid users wondering if their post posted correctly

Economic/Market Requirements:

- Our market is comprised of students and educators
- Needs to be a low to zero cost application; free to use and cheap to host
- Should provide a better educational experience in the classroom for our market

Environmental Requirements:

- The application should make for a safer learning environment
- The application will add a layer of privacy to classroom interaction promoting people to reach out with questions.

UI Requirements:

- UI should have a very simple flow, easy to navigate and understand for all users.
- UI should have a modernized design to improve the overall user experience

2.4 Engineering Standards

IEEE 1016: Software design description
We will use this standard when planning out our project. We will make data driven decisions to help create the best product. We will create diagrams for our architecture to help give visuals of our projects setup

IEEE 1028: Software Review
We will perform regular code reviews to ensure the quality of our application. This standard talks about having personnel, users, customers, and other interested parties review the code as well as the product to ensure quality which we will do

ISO/IEC/IEEE 26515:2018: Developing information for users in an agile environment
We will use agile during the development of our application. We chose agile so that we can work on many tasks of the project in parallel and have constant communication with our stakeholders allowing us to make changes as needed

IEEE 9274.1.1- JavaScript Object Notation (JSON) Data Model Format and Representational State Transfer (RESTful) Web Service for Learner Experience Data Tracking and Access
We will use JSON notation for our communications between frontend and backend. We chose this because both of our frameworks are JS based which makes interacting with JSON very easy

IEEE 7002- Standard for Data Privacy Process
We will follow this standard in order to safeguard the answers that students provide. We chose this because, even though their information isn't overly sensitive, we still need to be cautious and safeguard their anonymity as much as possible

3 Project Plan

3.1 Project Management/Tracking Procedures

Agile Development, since this is a web development project it makes sense to develop in sprints and split the workload up into stories.

We are using Gitlab in order to measure issues and progress of our project development.

3.2 Task Decomposition

- Create Figma Designs
- Create User Requirements
- Discuss Final Details with Client
- Develop API diagram
- Build Server
- Build UI
- Build API's
- Connect Front and Back ends together

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Finish Figma Designs
- Finish API Diagram
- Finish API Development
- Finish Database
- Finish UI
- Finish Connecting all UI components to Backend

3.4 Project Timeline/Schedule

Project Timeline

Week 1-4:

- Gave lightning talk
- Created mock designs
- Discussed with client

Week 5:

- Design Document: User Needs
- Initial Design for DB Schema
- Start Page Breakdown

Week 6:

- Design Document: Requirements

- Hello World Express App(Back-End)
- Initial Front-end Design(just visuals)
- Start list of APIs needed for front-end

Week 7:

- Create DB and Schema
- Finish Page Breakdown
- Design Document: Project Plan

Week 8:

- Containerize Hello World App
- Create Component Definitions
- Design Document: Proposed Design

Week 9:

- Create Containerized Deployment for Express Server
- Start work on React app
- Look into store for frontend

Week 10:

- Continue working on Express Deployment
- Continue work on React app

Week 11:

- Build Basic APIs
- Look into Express Web Sockets
- Continue work on React app

Week 12:

- Continue Building Basic APIs
- Have working Web sockets
- Have basic, working frontend application

Week 13:

- Continue Building Basic APIs
- Test connection with backend

Week 14:

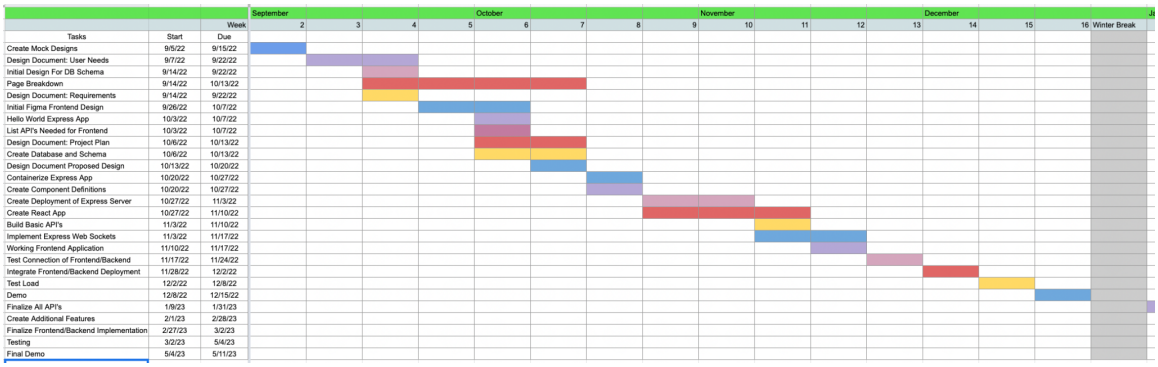
- Integrated Front-end/Backend deployment

Week 15:

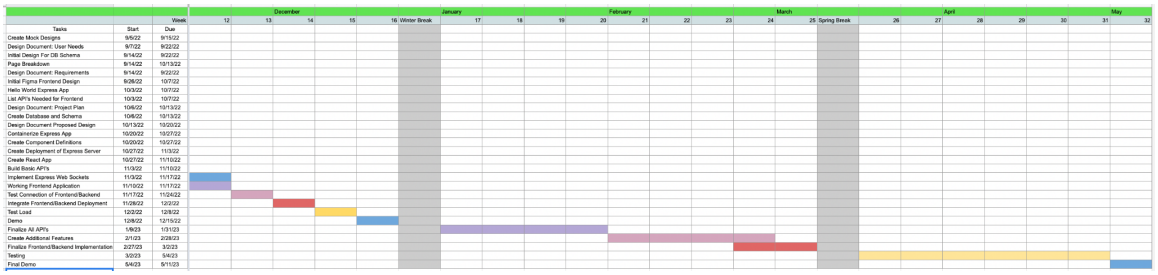
- Load testing(CPRE 491 class)

Week 16:

- End of Semester Demo
- Fix up any small bugs



Tasks	Start	Due	Week
Create Mock Designs	9/5/22	9/15/22	
Design Document: User Needs	9/7/22	9/22/22	
Initial Design For DB Schema	9/14/22	9/22/22	
Page Breakdown	9/14/22	10/13/22	
Design Document: Requirements	9/14/22	9/22/22	
Initial Figma Frontend Design	9/26/22	10/7/22	
Hello World Express App	10/3/22	10/7/22	
List API's Needed for Frontend	10/3/22	10/7/22	
Design Document: Project Plan	10/6/22	10/13/22	
Create Database and Schema	10/6/22	10/13/22	
Design Document Proposed Design	10/13/22	10/20/22	
Containerize Express App	10/20/22	10/27/22	
Create Component Definitions	10/20/22	10/27/22	
Create Deployment of Express Server	10/27/22	11/3/22	
Create React App	10/27/22	11/10/22	
Build Basic API's	11/3/22	11/10/22	
Implement Express Web Sockets	11/3/22	11/17/22	
Working Frontend Application	11/10/22	11/17/22	
Test Connection of Frontend/Backend	11/17/22	11/24/22	
Integrate Frontend/Backend Deployment	11/28/22	12/2/22	
Test Load	12/2/22	12/8/22	
Demo	12/8/22	12/15/22	
Finalize All API's	1/9/23	1/31/23	
Create Additional Features	2/1/23	2/28/23	
Finalize Frontend/Backend Implementation	2/27/23	3/2/23	
Testing	3/2/23	5/4/23	
Final Demo	5/4/23	5/11/23	



3.5 Risks And Risk Management/Mitigation

Performance target may not met when our users exceed 100	75%	We can acquire another server (or multiple servers) to help balance out the load when users exceed 100.
Malicious users get into the course that they don't belong in	60%	We could store a list of eligible users (listed by their student IDs) for each course. Then when a malicious user tries to enter/add a course it will not allow them to add it unless they are on the eligible list of users.
Course still says it is "LIVE" after the class has adjourned for the day. In this case the professor constantly forgets to adjourn/end the class	30%	

Professor continuously reveals the students name (after posting anonymously) to the class while sharing their screen.	20%	
Technical risk: There could be many more feature requests once we have our project in use. We will not be able to implement all of these requests in a short amount of time.	20%	

3.6 Personnel Effort Requirements

Personnel Effort Requirements

Task	**Hours**
Database initialization	40
Initial Docker container deployment	100
Web socket routing	50
Express.js initial setup	5
Dockerfile creation	25
REST API creation	250
Frontend page creation	350
Frontend/backend integration	50
Deployment of frontend	100
Load Testing	50

While we believe that our project will take significant amounts of time in many areas, by far the tasks that will be most time consuming are the creation of the frontend screens and APIs in the backend. This is because they will require the most logic and actual code/testing, while much of the other work will be some form of devops or a particular feature, and therefore less time consuming.

3.7 Other Resource Requirements

The server necessary to run the application on, and some sort of hosting/storage capability for the app's data once it begins to be hosted via Iowa State and used live in classrooms on campus.

4 Design

4.1 Design Context

4.1.1 Broader Context

Our project hits a lot of different areas. The table below describes a few such areas.

Area	Description	Examples
Public health, safety, and welfare	**How does your project affect the general well-being of various stakeholder groups?** Our project affects stakeholders by improving the quality of lectures for students and improving student interaction for professors. Students are able to make their voices heard and professors are able to get more feedback from students easier.	Our application allows students to ask questions during lecture and additionally the student can ask anonymously which allows all students to make their voices heard during lecture allowing for questions to be answered quickly. Professors are alerted to new questions being asked so they can answer them in a timely manner. Professors can also create polls during class to gauge the class's understanding of material as well as use the feature for attendance or other interactive learning applications. Our chat rooms will be active 24/7 which will help students to have access to help

		around the clock from their professor, TAs or other students.
Global, cultural, and social	**How well does your project reflect the values, practices, and aims of the cultural groups it affects?** Many members of the engineering profession are often quiet introverted people. These traits can make it hard to ask questions or communicate in large lectures. Our application reflects these traits by helping to create a solution to the lack of communication	Our application will allow for anonymous messaging. Since many students are often nervous to ask questions during large lectures, these students still need to make their voices heard. By allowing anonymous messaging, there is no reason for students to be nervous to speak up or ask a question they may deem as "stupid".
Environmental	**What environmental impact might your project have?** Our application will require many devices to connect to our application at a time which will inadvertently require a large amount of power especially for lectures of hundreds of people or more. This high power consumption could have a negative environmental effect.	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials Our application could increase energy usage from non-renewable resources but overall the impact of our application will be very little.
Economic	**What economic impact might your project have?** Our application could have a large beneficial financial impact on students	Our application will provide similar functionality to other apps like Top-Hat. Top hat currently requires a paid subscription. Our application could replace top-hat which could save students money from having to buy a subscription. Students are often tight on money so anything helps.

4.1.2 Prior Work/Solutions

Since our application is classified as an interactive learning application, there are two primary comparable applications that ISU Students are very familiar with, Piazza and

Tophat. Both of these applications promote classroom interaction and communication. Our application will utilize certain positive components that both tophat and piazza have and also build upon that to create a free and easy to use application for both students and staff.

Pros:

- Get live direct feedback from students/TA's/Professors
- Free to use, no third party service
- Ability to ask/respond to questions during/outside class
- 24/7 access to past lecture discussions
- Can gather statistics for professor/TA's to use for grading purposes

Cons:

- No built in lab/exam feature (tophat.com)
- No interactive textbook (tophat.com)
- No automated attendance/grading (tophat.com)
- No scheduling feature of networking events/interviews (piazza.com)
- No email feature for contacting large companies (piazza.com)
- No connection to canvas for grading (tophat.com)

4.1.3 Technical Complexity

Our project is sufficiently complex because it has both multiple components that must communicate with each other, and challenging requirements that aren't currently met by another solution. Our three main components are the frontend web application, the backend APIs and websockets, and the MySQL database that stores information for the APIs. Additional layers of complexity are added by our choice to use a Docker to deploy our backend application, as well as the fact we must implement both websockets and REST APIs that cooperate and are in sync with each other when providing information to the frontend web application. As mentioned previously, Tophat and Piazza are both somewhat similar to what we hope to build, however neither provides the full functionality that we are aiming for. Our project is unique in that it will combine the real-time polling aspect of Top Hat with the forum-like structure of Piazza to create a solution that we believe will be very valuable.

4.2 Design Exploration

4.2.1 Design Decisions

- *Design Decision:* **Frontend and Backend Frameworks**
  The underlying frameworks that the frontend and backend are built on directly influence many details of the project. The frameworks should be versatile enough

to allow for any current and future functionality that may be required of the application, but also efficient enough that development will not get blocked or throttled by the frameworks or by maneuvering them to fit our needs. To accomplish this, the frameworks should be relatively simple to use, but easily extensible to add any specific functionality.

- *Design Decision:* **Page Layouts**
  User experience and look-and-feel are both vitally important aspects of a successful software product. The user experience should be smooth and intuitive; no feature should feel obtuse or require training to use. The look-and-feel also should be comfortable and visually appealing. Incorporating both of these aspects into the project begins with careful design of the page layouts. The page layouts define what the user sees, how the user moves through the application, and how the application's features are presented.

- *Design Decision:* **API Definitions**
  In order for the frontend and backend to work together, there needs to be an agreed-upon set of interfaces the backend provides and the frontend uses. These interfaces will serve as the only form of communication between the two sides of the application, so it is important that these interfaces are established early and before major implementation work begins.

4.2.2 Ideation

For **Page Layouts**, we considered a number of different options to set this up.

1. We first discussed having the exact same layout for all of our users, just some users would have options disabled. In this case we are considering the professor having the same page layout as the student, but the student would not have the option to reveal the anonymous user's name or see the list of participants in a discussion.
2. We discussed having a very basic page that only contains basic core functions. In this case we think the users would be able to navigate through the website with ease, but we would also lose some functionality if we were to keep only the core functions.
3. We then considered having separate pages for all users. In this case, the student's view would be a completely separate page, filled with separate components, from the professor and same goes for the TA. This design option, in theory, would be much more straightforward when only considering one user per page and only considering the functionalities they would need. The issue with this option is that we are copying over a lot of the code per page.

4. We considered having a more intricate page layout, which would include the options to change it to dark mode, rearrange the classes and other functionalities that will help the user to customize the pages. The issue with this, is that we would have to add all alternative page layouts for each view and each user. This is a farther reach for us, and in turn we chose to not consider this for the first iteration of the MVP.
5. We considered implementing each page layout as a whole instead of breaking it up into components. This way it would be easy to reuse the page layout and keep it consistent throughout, but this would also mean a bunch of repeated code which will lead to bugs later on. This would also make it more difficult to make simple changes to things that could be considered components (like the top nav bar and such). So we chose against this option.

4.2.3 Decision-Making and Trade-Off

We ended up choosing option 1. We took into account many different factors, some being design based, and others over a series of constructive feedback from our project manager based on our initial evaluations. We went with this option because it ended up being the most practical choice to meet our specific project needs. This project has a lot of complexities to it, so we ended up going with a very streamlined design that will get a well rounded project out in the timeline we have by setting attainable goals to meet the requirements.

4.3 Proposed Design

4.3.1 Overview

So far our team has worked on the critical components of our project such as setting up the database server and finalized our designs for what our application is going to look like. We created many diagrams to show the flow of the application such as overall backend implementations including the database schema, and also page layouts and functionality between pages. We have also started creating the key components of the application on the frontend. We have also set up our database using MySql, implemented a CI/CD pipeline for deployment, and also started creating API's. Additionally we have implemented websockets to ensure effective roundtrip communication between the frontend and backend. We have a live version of the application deployed, and plan to have messaging working by the end of the semester

4.3.2 Detailed Design and Visual(s)

Below we have the detailed designs for the backend and frontend sides of our application.

4.3.2.1 Backend Overview

Our project's backend is composed of a MySQL database as well as a node server. We are deploying to one of Iowa State's virtual machines. Our database is running within a docker container deployed on the server. Our node application is an Express server running on port 443 and the server uses pm2 for load balancing. Below is a diagram of our backend:

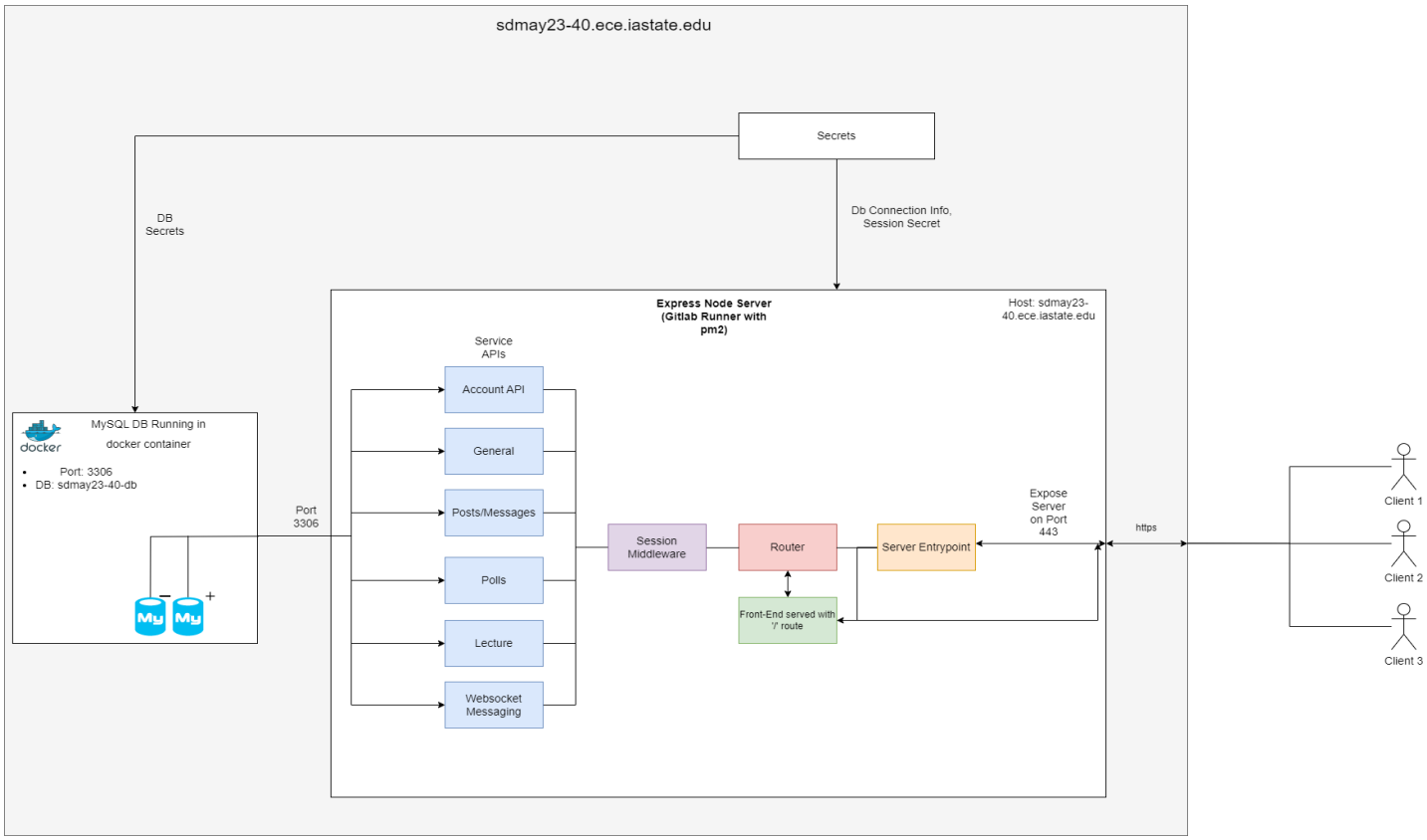


Figure 1: Backend Block Diagram

From the diagram above you can see how our application is connected within the server. We store our secrets within the server which includes DB connection information and our session key. We run our application on a gitlab runner which takes our express server, places it in a docker container running a node image, and then uses the pm2 library to run the server. We chose pm2 because it will keep the server live as well as provide us the ability to restart the server seamlessly which was vital for our deployments. Additionally we serve the front-end through express by serving the front-end index file when a user hits the https:/sdmay23-40.ece.iastate.edu route. This allows us to contain the app

completely within express and use the same deployment process for both front-end and backend

4.3.2.2 Express Application

We use the express.js node library for our backend. Express allows us to quickly create APIs, but also allows us an easy way to use websockets which are vital to the live messaging aspect of our application. On startup we load environment variables from our server for database connection information as well as for session management. After these variables have been set we have our express app load our API routes and then start listening on port 443. In order to make our APIs more secure we use sessions. In order for a user to make a request that involves a database connection, that user will first need to login. Once the user logs in they will receive a session token, linked to the user's ID that is valid for 24 hours. Every following request made to our application will go through session middleware to validate the user's token. If the validation fails the user will not be able to make requests and will need to log in. The express application is also able to serve the end-user the front-end through an api route which allows us to contain the entire project within one application.

4.3.2.3 Database

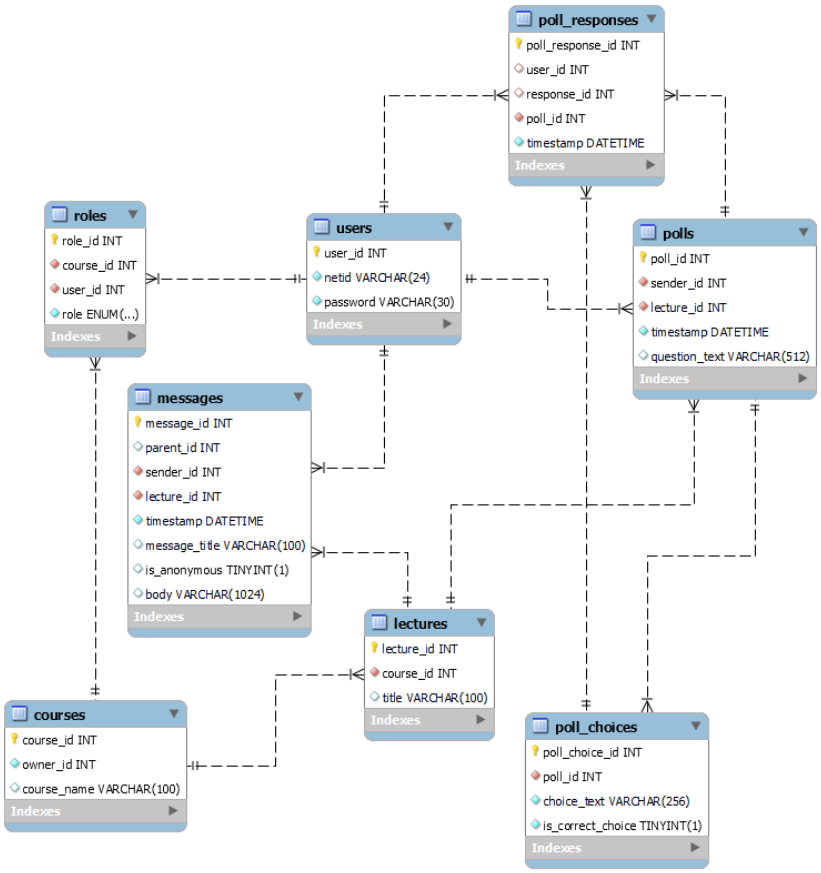


Figure 2: Database schema

We chose to use a MySQL database for this project. The schema is composed of 8 different tables which facilitate the storage of information about courses, lectures, users, users' roles, messages, and polls. A user is linked to a course and assigned a role within that course using associated entries in the roles table. Each course will have multiple lectures linked to it using the lectures table. The lectures table has associated messages and polls linked to a given lecture through the messages and polls tables. A message contains the message body, timestamp, link to the posting user, and other pertinent information. Meanwhile, a poll contains similar information, but also is pointed to by the poll_choices table which stores information about the different options offered in a poll. The poll_responses table is responsible for storing a user's selected option(s) by providing a link between the two tables.

4.3.2.4 Deployment

We have implemented a CI/CD pipeline for our application allowing us to deploy code very quickly. We utilize Gitlab runners as well as Gitlabs token storage and access token

features. When code is merged to master our pipeline kicks off and first grabs our private SSH key from gitlab which the pipeline uses to authenticate an SSH connection to our server. Once that connection has been established our gitlab runner on the server uses an authentication token to pull the latest changes from our repository. Once the latest changes have been grabbed, we restart our node server with the latest changes. This deployment pipeline covers both the frontend and backend, allowing us to only need a single pipeline and gitlab runner.

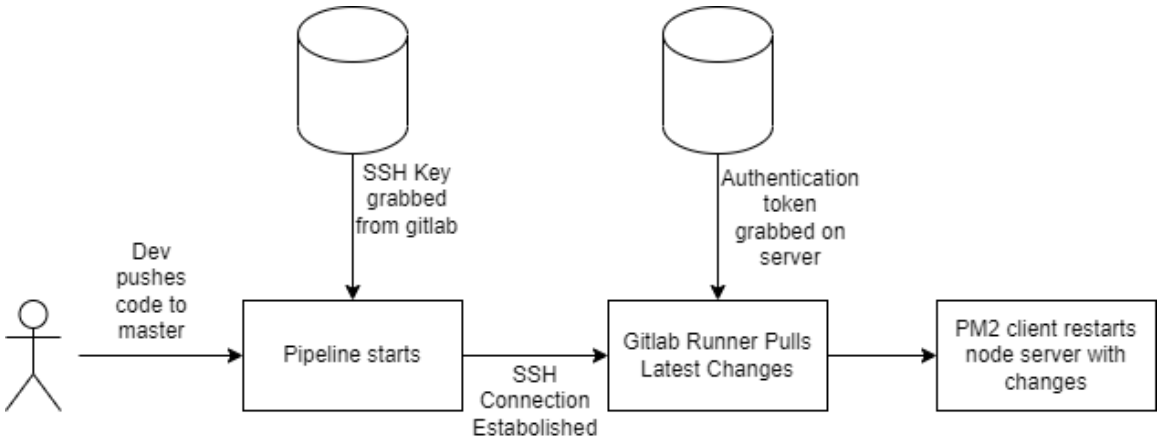


Figure 3: Pipeline flowchart

4.3.2.4 Frontend Overview

The project's frontend is a website built on the React framework, including various additional libraries (e.g. webpack, Babel) to provide common functionality and streamline the development process. The website has a persistent topbar to provide always-available navigation, and the website is organized into multiple pages that each provide a specific set of features to the user.

4.3.2.5 Frontend Design

The frontend follows the general React design approach and has a single root "app" which renders the JSX hierarchy for the website, leveraging routers through the React Router library to facilitate page changes. Pages are the next-level object, containing the layout for one single browser view. Pages are made up of HTML components, open-source library components, and custom components.

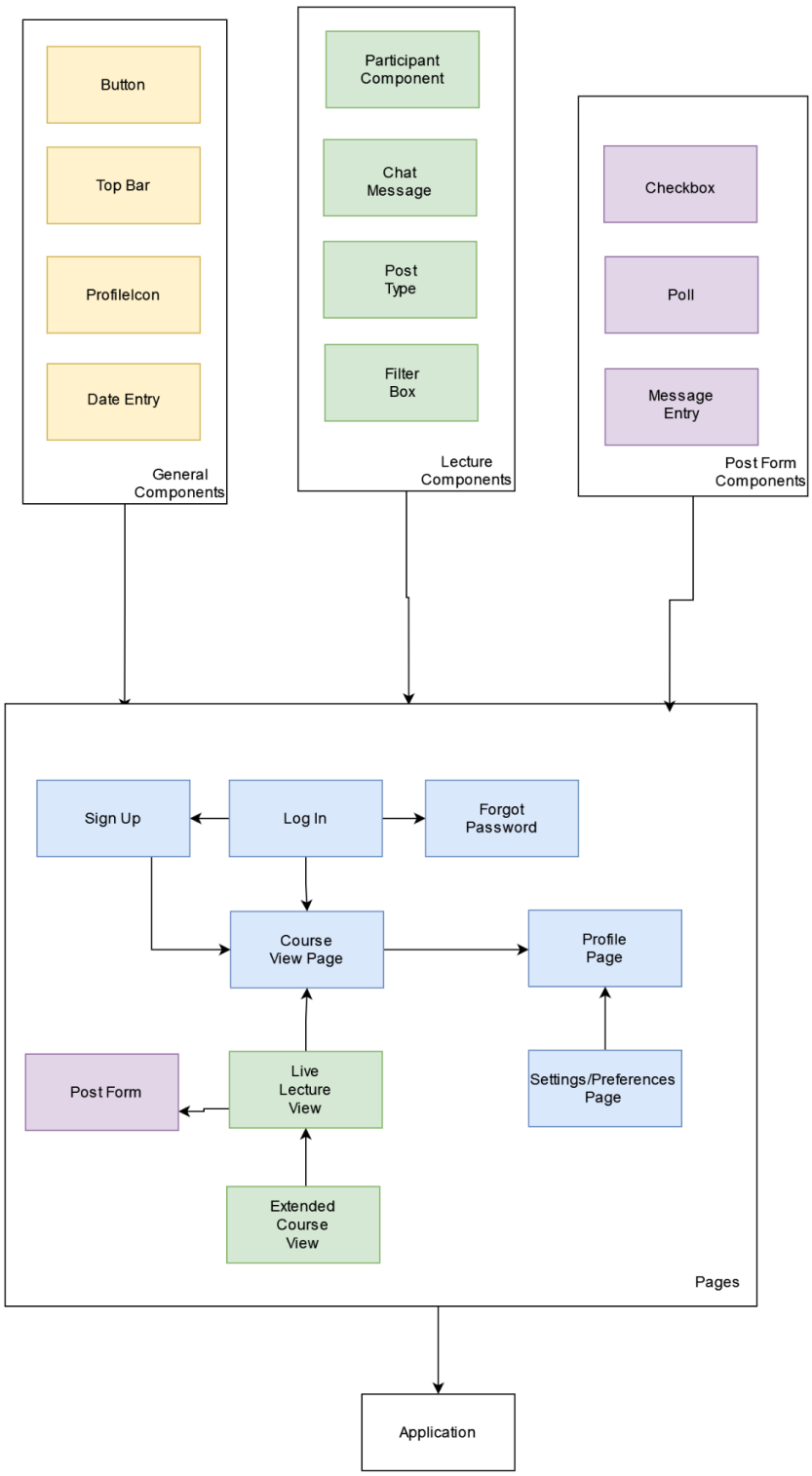


Figure 4: Frontend Breakdown

The diagram above shows this distinction between pages and components and dialogs the most important components that the pages will include (small encapsulated sub-components which will not be used directly by a page are not shown). API communication will be predominantly handled at the page-level, after which the page will layout/relayout the view.

4.3.3 Functionality

Our application is meant to be as user friendly as possible. All of the design decisions and implementations of the application should be hidden from users so that they can just worry about the intended uses of the application like answering questions.

Our application will start off with a login page with a sign up option so that the user can be assigned to three different roles: Professor, TA, Student. This will allow professors, TA's and students to login to their respective accounts. From there the professors/TA's should add the students to a course so then students will be able to join their assigned courses using some sort of generated code. From there the student should be able to view current/past discussions, questions and messages from lectures. When students are in a course they will be able to "raise" their hand, ask questions, answer questions and send regular messages to participate in the class discussion. There will also be an "anonymous" feature so that nervous or timid students may be able to ask questions comfortably.

Professors will have many more features compared to the student. They will be able to create a course, get a code for the students to join, gather statistics from polls/questions such as who is the most active when asking/answering questions, send messages, and also answer questions. The professors should be able to send out a poll during a class lecture where then students can answer it. They will also be able to view who is anonymous or not.

The TA's role will be different in a sense where the professor can assign certain permissions to the TA's. The TA's will start out with the same permissions as the student but can have almost as many permissions as the professor. The professor will enable/disable permissions for the TA's such as allowing them to post polls and other things.

Our team has created the initial UI of the application which is shown in section 4.3.2.6. This shows the design of the pages and the functionality between them.

All of the data that is exchanged during all of the interactions will be recorded on a database where we can gather and pull information from.

4.3.4 Areas of Concern and Development

Our design works well to satisfy both requirements and meet user needs. As far as the development side of things go, the Backends API / Database design set up will merge with the needs of the Front End well. We will achieve both the design and functional needs of the user through our app's features. Our final design also meets all of the requirements laid out by Maruf, our team advisor.

Some of the primary concerns will be how the system will store data, and delete data, and work under a larger scale load of a particular size classroom.

We will tackle the issue of latency and load balancing of our application by making sure nothing is front-loaded, and that everything renders as efficiently as it possibly can by staying vigilant of this potential problem in the development cycle. The data storage and deletion will be addressed by working with Maruf in order to figure out the best possible solution for this issue. Some questions that arise might include how often should data be deleted, and what level of importance should the data have that is being stored in order to preserve the most necessary items to minimize cost.

4.4 Technology Considerations

Based on our team's experience and from the previous version of this application we decided that using Express on the backend would be best suited for this type of project because it allows us to quickly create API’s and implement websockets which are essential to our application. Express also allows us to serve our front-end from the server as well which is a huge bonus. We decided to go with using a MySql database because we have previously used this and it has many benefits such as data recovery and ease of use. For the frontend we decided to use React native because of the many benefits such as allowing developers to create the UI easily and React is much easier to implement than others such as Angular.

4.5 Design Analysis

So far we have a live build of the application running on Iowa State's servers. Backend has created a plethora of APIs as well as having reliable websockets working. Front-end has completed various pages and components for the app. Our proposed design from 4.3 has proved effective and we have been able to make fast progress this semester. Earlier in the semester we went back and redid our front-end designs based on client and team member feedback, but overall we are on track for having a prototype completed by the end of the semester. We had a demo with our client earlier on the live build of the app showing the front-end, back-end integration via the login page. By the end of the semester we plan to have the live messaging portion of the application working on the deployed version of the application.

5 Testing

Our team's testing strategy is fairly lightweight. The application will not be directly responsible for calculating and/or recording student grades, nor will the application store sensitive user data beyond a name and Iowa State email address. The application simply serves an auxiliary purpose to help improve student-professor communication during lectures. Because of this, rigorous testing is not necessary for our project. However, it is still important that basic functionality is verified upon new builds.

Additionally, our testing strategy puts a greater focus on testing the backend APIs over other parts of the application. Automatic verification of the rendering of a web document is complex (and the layout is very subject to change), so it is more worthwhile if the relatively fixed APIs the frontend uses and the backend handles are given the focus of our testing efforts.

We will begin paying greater attention to testing and the execution of our testing plan after the completion of a minimum viable product to the satisfaction of our client.

5.1 Unit Testing

Frontend: We will be unit testing our most commonly used React components and pages.

To test the components and their functionality, we will use Jest and Enzyme to create test cases in order to check that the components are operating as per our requirements. For the pages, we will utilize a similar method to the components, although we will focus more on the basic layout than on specific functionality.

Backend: The units we are going to test are the database and APIs.

To test the databases, we are going to be using MySQL workbench, because it is a very popular open source database management tool for MySQL databases. With this, we can write queries, execute them, and get the results to test the database.

For APIs, we are going to be using Postman to create our tests and set them up in the CI pipeline. This will ensure that there is end-to-end functionality of the web application. We will also need to test the security of the APIs as we will need to secure specific information going over HTTP.

Tools: Jest, Enzyme, MySQL Workbench, Postman

5.2 Interface Testing

Frontend: In order to accomplish live messaging and updating during live lectures, the frontend and backend will be communicating through WebSockets. This is a very critical

interface, as it is the fundamental interface that enables the primary feature of our application. This interface will be tested through the creation of a mock frontend data consumer and a mock backend data producer on the same machine, which will then use our WebSocket infrastructure to communicate and verify successful messages. The mocking features of Jest can be used to accomplish this, as our frontend and backend are both written in JavaScript.

We will also be creating a test that will both post a message/discussion to an archived lecture and then test that we can retrieve that message and it is categorized with the correct lecture and date. This will be tested using a local server as to not actually store the test messages in the database. We will be setting up these tests as we establish connections between the frontend and the backend.

Backend: For our backend, we have essentially two interfaces. The first interface is our Express application, and our second interface is a MySQL database. Our Express server communicates with our database for many of our endpoints, so testing is a high priority. To test this interface, we have created a local script that creates a local database and starts our Express server locally. The script automatically adds data to our database, and we test the Express server to database connection by calling our various endpoints. In the future, we plan to add a stage to our deployment pipeline that will check if our health endpoint works. The health endpoint will simply check if the Express server and database can communicate properly. If this test fails, the pipeline will fail.

Tools: Jest, Docker, Bash

5.3 Integration Testing

Frontend: The most critical integration path relevant to the frontend is the loading and routing between various pages. Since we are using React with a single root element, this isn't quite as straightforward, and special care will need to be taken to ensure it works well. Automated testing of this integration would come down to simply loading an example nested page (i.e. not the root page) and verify that it loaded correctly. In order to load a page like this, the routing logic would have to run successfully in sync with the page loading/rendering, so this will work as an integration test.

Besides page loading/routing, proper display of fetched backend data is the other important integration. Testing the exact rendering of the data in the DOM would be excessive, but there are other strategies we will leverage to test this integration. For example, when testing the display for the list of courses, we will test to ensure that the proper number of courses is being displayed.

Backend: The most important components that will need to have integration tests are the ability to call the APIs and the APIs' ability to fetch data from the database and return it to the calling user. A great example of this would be to add a "Health" endpoint that ensures it is able to connect to the database, grab some data, perform some sort of simple calculation on it, and then return a success or failure result. In order to test this, and other similar but possibly more simple tests (i.e. only testing the integration of the Express app and the database), our main tool will be Jest.

Tools: Jest, Docker

5.4 System Testing

The root system of our application is the operation of live lectures, and the most essential system-level interaction that needs to function correctly is the start-to-finish sending and receiving of messages from users. If that works correctly, then we can have much higher confidence in our application, as all communication during live lectures takes the form of a "message," whether that is a conventional text message or a multiple-choice poll. For unit testing, testing the frontend message producers, backend message processors, and frontend message consumers would suffice. Also, for the backend, testing our API controllers will suffice since these controllers handle all of the logic for our APIs. For integration testing, testing the successful operation of WebSockets will ensure correctness for communication. Additionally, the backend will test its integration with the frontend, server, and database by having a health check on deployments. This check will call a health endpoint that will return success if the API call succeeds in making a database connection.

Tools: Jest, Docker, Bash

5.5 Regression Testing

Frontend: Along with the use of CI/CD and standard agile development methodologies, the frontend can assure that the app is rendering visually as it should through the use of snapshots. Enzyme and Jest give us the ability to generate snapshot tests that can determine if the build is rendered correctly. This, along with the use of CI/CD, covers that our features being pushed go through without breaking existing functionality.

Backend: We are implementing CI/CD for our web application. Regression testing on the CI/CD pipeline will be used to make the newer releases of our application more efficient and consistent, as it verifies the existing functionality. We will make sure that new features or changes pass all prior unit tests on a development branch before merging to the master branch, which will be our main application. With this method, we can

ensure that the master branch will always be working and avoid spending lots of time debugging the application if an error occurs. The main features that should not break as they are critical to our application are the security of data, such as passwords, and important APIs, such as login and lecture data getters/setters.

Tools: GitLab CI/CD, Bash

5.6 Acceptance Testing

Frontend: We will show our client the functionality of each of the pages, as well as the aesthetic to ensure it matches with what he had in mind. We will also have potential student(s) or testers that will try to navigate our application to make sure it is both intuitive and something that they would enjoy using.

Backend: We will show our client the APIs and validate with him that they are providing the expected functionality in an acceptable manner.

5.7 Security Testing

Frontend: As the frontend will be communicating through HTTPS and will not be responsible for storing sensitive data (minus the session information, which will be stored in browser storage where it's isolated from other webpages), security testing is not applicable for this side of the application.

Backend: The backend involves authentication on requests to our server for security. When the user logs in, they are given a session, and this session is required to make any future API calls to the server. We test this through middleware we have added to our server. All requests go through this middleware, and the middleware checks if a user has a valid session before the user is able to make a request. We have a local startup script that uses Docker to create a local database, and then our server is also started locally. We use this local startup to test our APIs and make sure that a user can only make a request if they have a valid session.

Tools: Docker, Bash

5.8 Results

Currently, we have just exited the design and planning phase and are currently in the process of developing the core features of our application. As such, we have not performed/written any tests beyond basic health checks, but as we continue to implement our application beyond the minimum viable product, we will write tests to ensure functionality according to the requirements.

6 Implementation

6.1 Backend

Backend has made very good progress on implementation so far. We have the entire infrastructure built out this semester, so we can focus on feature development and testing for the remainder of this project. For our future implementation we plan to use jest mostly for testing and we plan to mostly rely on integration testing for the backend so that we can verify all of our APIs work. For the development aspect we will continue creating APIs for the front-end to consume by creating new routes within express to retrieve whatever data the front-end requires. We plan to have a prototype working with basic functionality by the end of the semester, and plan to have most feature development done midway through the next semester so that we can focus on final testing and client acceptance.

6.2 Frontend

Frontend has begun implementation of a few of the crucial pages. We have the login page connected to the backend for approval of login.

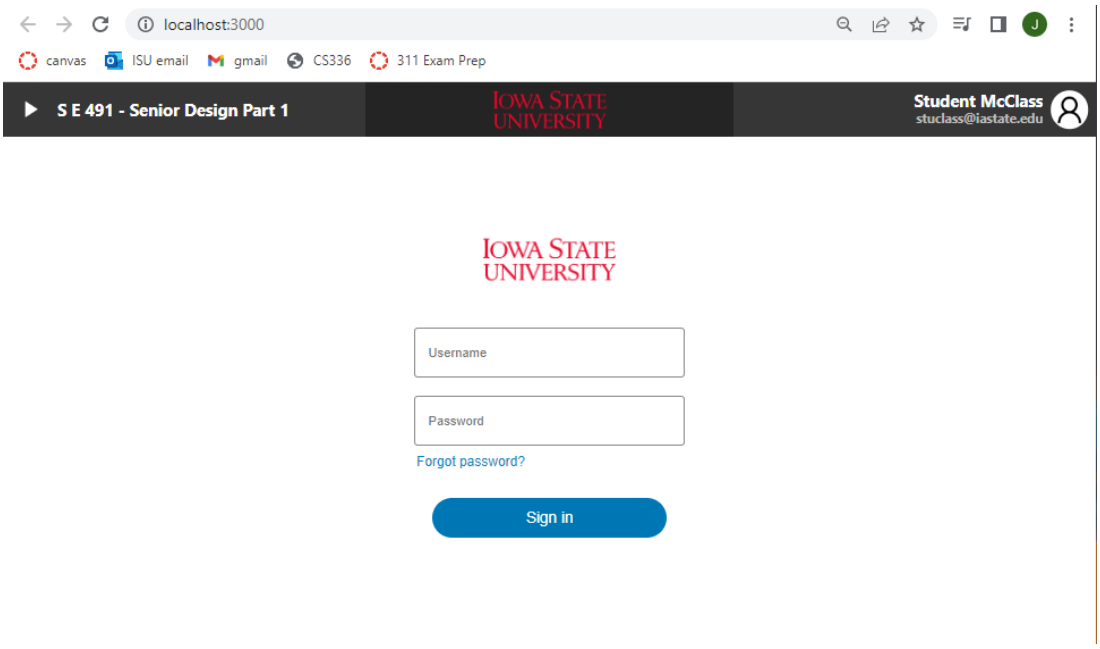


Figure 5: Login Page

We have plans to implement the course view page, as seen below in figure 6. We plan to have this page automatically update when loaded to retrieve the previous lectures as well as load which lecture is live, if any.

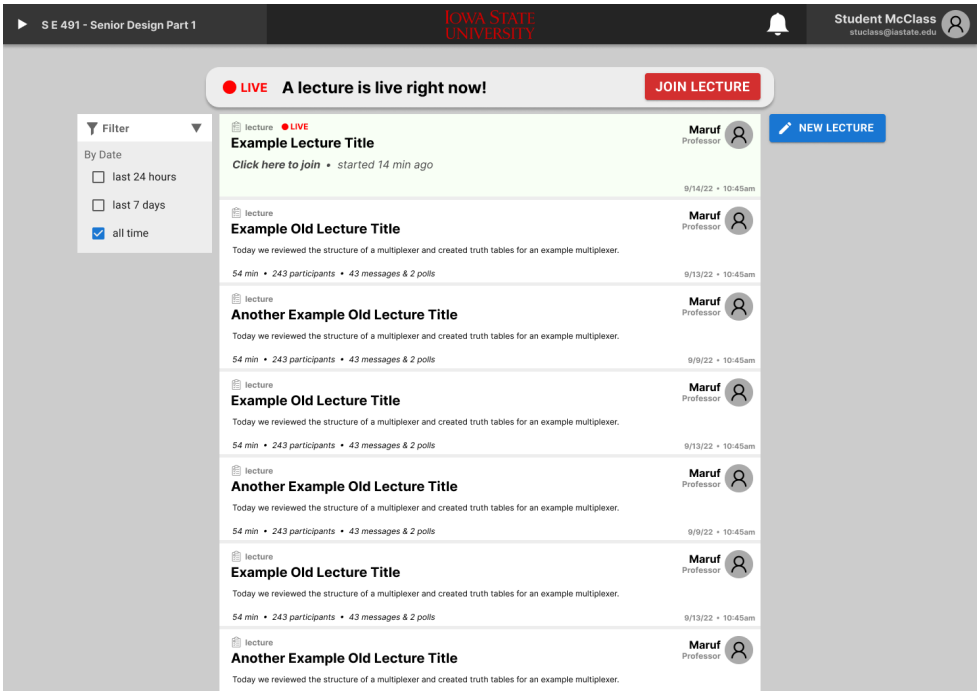


Figure 6: Course View Page

We have also implemented a static live lecture view page, as seen below in figure 7. This page will soon be connected to all other users on the live lecture feed using websockets. This way, when any user posts, all other users will have their pages updated to show the new message or poll. Simultaneously, the backend will be storing these messages in order for retrieval in the course view page.

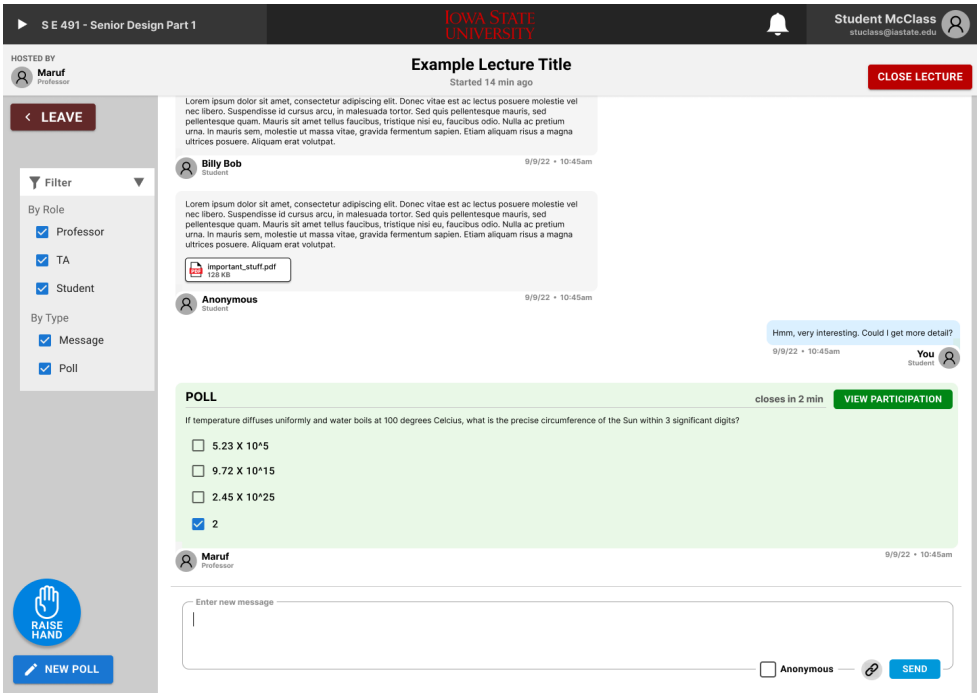


Figure 6: Live Lecture View

7 Professional Responsibility

7.1 Areas of Responsibility

Area of Responsibility	Definition	NSPE Canon	IEE Computer Society
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.	Section 3 discusses SE's should ensure their products meet the highest professional standards possible at a reasonable cost. Section 3 differs from NSPE because cost is not mentioned in the definition in column 2 and 3
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	Section 3 covers this portion. In section 3.09 discusses engineers giving realistic estimates of cost. This means engineers should do their best to charge customers reasonably for the work that they will provide. This section differs from NSPE because it also discusses ethical choices related to work being performed which NSPE does not cover in column 2 and 3
Communication Honesty	Report work truthfully, without deception, and are understandable to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts	Section 1 covers this portion. Parts 1.01, 1.04, 1.06 best cover this portion since it deals with the public. To summarize, software engineers should hold themselves accountable for their work, disclose any potential dangers to the appropriate people and avoid deception is all aspects of their work. This section is very similar to NSPE, but goes more into detail giving many subsections instead of a broad overview.
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	Section 1 also covers this portion. To summarize, software engineers should disclose any dangers to the appropriate personnel. Software engineers should also consider disabilities that would prevent certain people from benefiting from the software. An example of this would be providing a colorblind option on a game or website. This

			description differs from NPSE in column 2 and 3 because it gives specific use cases and talks about disabilities which are related to health, but should be its own section.
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	Section 2 covers this portion. Software engineers should use only client or employer property in authorized ways. Software engineers should also keep confidential information from their client or employer secret. This differs from NSPE because it also talks about only using products obtained ethically whereas NSPE is more dialed into talking only about acting faithfully to an employer.
Sustainability	Protect environment and natural resources locally and globally.	None	Section 1 covers this portion. Software engineers should only use/write software that is safe, and does not harm the environment or invade others privacy. Overall work should be completed for the greater good. This differs from NSPE which talks about protecting the environment for future generations, but does not mention software directly.
Social Responsibility	Produce products and services that benefit society and communities	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor and reputation of the profession.	Section 8 and 6 covers this portion. Software engineers should not give unfair treatment based on prejudices and obey laws governing their work. This differs from NSPE because these sections also talk about lifelong improvement in their knowledge of the different fields of software engineering.

7.2 Project Specific Professional Responsibility Areas

Area of Responsibility	**Applicability**	**Current Level of Performance**
Work Competence	Work competence is very applicable. We want to ensure that we are competent in the skills required to build this application and therefore	**Medium** We are doing a lot of good work, but there are some areas we are lacking. For example the course view design was just 4 rectangles. We decided on medium

	produce a quality result.	because there is much more good work being done than bad work, but since there is still some work that is lacking we can improve.
Financial Responsibility	Financial responsibility is not very applicable. All of the resources we are using are either open-source or already provided by the university, meaning there are no/little costs involved in this project.	**High** We have not needed to use funds for this project which puts us at a high rating. We are using open source software and university resources, allowing us to keep our costs low. We are giving our client reasonable estimates and keeping costs low.
Communication Honesty	This area is extremely important and applicable to our project. Honest and candid communication between team members, as well as with outside entities, is important to avoid miscommunications and maintain common understandings.	**Medium** It is good that we communicate at our meetings every week, but it can be hard to contact certain individuals within the team. We have also had a few meetings that were not effective and took far longer than they should have.
Health, Safety, Well-Being	Health, safety, and well-being is not very relevant to our project. There are no true safety or well-being implications included in our applications.	**High** We have done a good job identifying potential risks within our application. We added security measures to prevent unauthorized requests to our backend server, and the front-end team also has implemented a login process. We have done a good job identifying and implementing security measures which will minimize risks to our stakeholders.
Property Ownership	This category of responsibility is important to our project, especially as it pertains to user information security. We must ensure that users are only able to access their own information, or that of their students if they are a professor or TA for a given course.	**High** We meet with our client often and discuss our progression even though our client is a bit hands off. We have been transparent about the software we will be using and we are acting faithfully by adhering to our clients needs.
Sustainability	Sustainability is not applicable to our project as there are no implications related to the environment or natural	**High** Our application has a low environmental impact to begin with, so we are doing well in this aspect. For the

	resources outside of the negligible amount of electricity required to run the users' clients and the server.	little confidential information we will store we provide appropriate security. I gave us a high rating since our application is very sustainable since it only uses ISU resources.
Social Responsibility	This is an important area for our project. We are trying to help students have a more productive learning environment, and therefore benefit the academic community.	**Medium** I think most members have been taking responsibility for their work, but some have struggled. I think we can improve by everyone taking some time to improve their knowledge of the tools and software we will be using. Since I think we can improve a fair bit in these areas I gave us a Medium.

7.3 Most Applicable Professional Responsibility Area

Social responsibility is the most important area of responsibility. This is because we need to ensure that our application is giving an appropriate level of privacy to the users of our system. Examples of this are situations such as not allowing student users to see their peers' individually identifiable poll responses, or TAs and professors only being allowed to have elevated permissions to see users' information for the correct courses.

8 Closing Material

8.1 Discussion

Our team has worked on many of the critical components of the product such as having a fully responsive, working frontend and backend. Our team has worked on many things in order to achieve this goal, the main ones being setting up the server and implementing APIs and websockets to create easy communication between the users and the server or database, and the setting up of a React design on the frontend containing many pages and components. With this, our team has met many of the requirements that were established in the planning stages of the project and can verify that almost all of the requirements of the product are met with great precision. Since we are still in the development phase and don't have the complete application ready, the main critical components of the web application work and are ready for demonstration.

8.2 Conclusion

As it stands currently, in a large lecture hall that consists of up to perhaps five-hundred students, it is hard for the professor to communicate with the students. Such as it can be

difficult for the professor to hear or answer all of the questions that may be raised during discussions. To solve these types of problems, we proposed the best plan of action was to develop an interactive learning tool to bridge the gap between students and professors. Our application has chat rooms for different courses or lectures led by a professor. In these chat rooms the students will be able to message each other, raise their hand, anonymously ask questions, and answer in class polls created by the professor. These types of features on our web application help create easy communication between students and professors in large size classes.

Looking back throughout this semester we believe that there were some things that could have been improved. One improvement we could have made was to implement some core components earlier such as API's. We believe that if we could have developed the main API's earlier then the team would have a lot more time to focus on other critical components. The main correction we could have made was to plan much more diligently. Our schedule was really only planned out for this semester and we kind of shrugged off next semester and gave large overarching tasks such as testing for three months. We feel that if we planned out thoroughly for both semesters it would allow us to see which dependencies perhaps could be completed a lot earlier, thus giving the team much more slack.

As we transition from a successful design phase to finishing implementation, we are hopeful that the professors, TA's and students will find the application useful for large classrooms and enhance the way they teach and learn.

8.3 References

IEEE 1016: Software design description
We will use this standard when planning out our project. We will make data driven decisions to help create the best product. We will create diagrams for our architecture to help give visuals of our projects setup

IEEE 1028: Software Review
We will perform regular code reviews to ensure the quality of our application. This standard talks about having personnel, users, customers, and other interested parties review the code as well as the product to ensure quality which we will do

ISO/IEC/IEEE 26515:2018: Developing information for users in an agile environment

We will use agile during the development of our application. We chose agile so that we can work on many tasks of the project in parallel and have constant communication with our stakeholders allowing us to make changes as needed
IEEE 9274.1.1- JavaScript Object Notation (JSON) Data Model Format and Representational State Transfer (RESTful) Web Service for Learner Experience Data Tracking and Access
We will use JSON notation for our communications between frontend and backend. We chose this because both of our frameworks are JS based which makes interacting with JSON very easy

IEEE 7002- Standard for Data Privacy Process
We will follow this standard in order to safeguard the answers that students provide. We chose this because, even though their information isn't overly sensitive, we still need to be cautious and safeguard their anonymity as much as possible

8.4 Appendices

N/A

8.4.1 Team Contract

Team Members:

Brandon Burt

Jaden Ciesielski

Adam Walters

Alex Swenson

Tyler Miller

Guan Lin

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:

   Thursdays: 5:45 - 6:45 pm (Group Meeting)
   Backup-Time: Sunday 6 - 7pm

   Additional Front-end and Backend Meeting to be scheduled weekly once roles are assigned

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

   Discord, Snapchat, gitlab issues, email

   *Tag on discord

3. Decision-making policy (e.g., consensus, majority vote):

   Consensus

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

   Shared drive folder with a document for each meeting
   minutes will be on the document and we will follow our advisors template

   Note Taker: Rotation
   Member Minutes will be on the advisors template and we will enter minutes at the end of each meeting

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

   80% +

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Each group member will be responsible for the work assigned to them. If the member requires help they will let the group know as soon as possible. Communicate often and keep all members relevant to your tasks in the loop. Don't wait to be told to complete a task.

3. Expected level of communication with other team members:

   Keep the group informed of the work you are doing either via discord or through our weekly team meetings. Communicate often so that we don't get out of sync on the project. At the minimum weekly updates at the front-end/back-end meetings as well as the group meeting

4. Expected level of commitment to team decisions and tasks:

   Be committed to the choice the group has made, even if you don't fully agree. Put in your best effort. Expected 6 hours of work per week as per our advisors standards

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

   Backend: Tyler, Alex, Guan
   Frontend: Jaden, Adam, Brandon

   Frontend Manager: Adam
   Backend Manager: Tyler

   Meeting Coordinator: Jaden

   *Roles may change based on project needs

2. Strategies for supporting and guiding the work of all team members:

   Create Gitlab issues for stories
   Follow advisor template with contributions and plan work the following week
   Work towards milestones
   Get refined goals and timelines from advisor at (bi)weekly meetings

3. Strategies for recognizing the contributions of all team members:

   Fill out advisor template with contribution minutes and create a gitlab issue for any code related changes

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

   Tyler: Extensive Node API development with AWS Lambda and Express.js. 1.5 years experience with React.js and half a year of experience with Vue across 2 companies in Industry
   - React
   - Vue
   - Node.js/Typescript
   - Express
   - AWS
   - Python
   - MySQL and Postgres
   - Containers
   - Agile
   - Jest, Vitest, Cypress
   - Jenkins

   Jaden: Have 2+ years of experience in agile app development communicating directly with the RF engineer clients, using REST APIs and maintaining servers.
   Competencies:
   - Python
   - React
   - Node.js
   - Javascript/Typescript

- Java
- AWS
- Linux/UNIX

Adam: Have 5+ years of experience developing video games, requiring knowledge of
UI/UX, using extensive APIs, exploit protection, and of course programming a large networked project. *Competencies:*

- C/C++
- Java
- HTML/CSS
- JavaScript/TypeScript
- Node.js
- AWS
- UNIX

Guan: Have 1+ years of mobile application development on the frontend, and automated unit and integration testing.

- Python
- Java
- C#
- C/C++
- Javascript
- HTML/CSS
- SQL
- Agile

Alex: Have 2 years of experience as a full stack developer working with Node.js, React.js, various types of databases and many different other languages and frameworks. Experienced in coordinating with multiple teams across the enterprise.

- React
- Node.js
- Express
- AWS
- Python
- MySQL and Postgres
- Containers
- Agile

- Jest
- Jenkins

Brandon: Have experiences in Web Development, App Development, and IT primarily. Worked with various modern web technologies, all 3 major frameworks, (React, Angular, Vue JS). Software Testing experience using React Testing Library, Jest, and Enzyme. Primarily Frontend Experience. Worked for Cerner Corporation, Union Pacific Railroad, ISU, United States Department of Agriculture, and more recently Collins Aerospace under Raytheon Technologies.

- React, Vue JS
- HTML
- Javascript (Vanilla, ES5, ES6)
- CSS
- Agile Development
- Jest/Enzyme
- SQL

2. Strategies for encouraging and supporting contributions and ideas from all team members:

Make sure every team member contributes at least one idea per meeting. Be open to all team members' ideas. All members will help in the creation of stories at development time

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

   Communicate early if there is a problem, and don't be afraid to speak up. Bring up any issues at our weekly team meetings.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

Create a shell for the eventual application and have a well formed development plan for next semester. Create A level work

2. Strategies for planning and assigning individual and team work:

At our weekly meetings, divide work out and have each member in charge of a task or tasks. We will create stories at development time and during the planning phase we will keep a list of tasks to complete as they arise and assign tasks each week. Team work will be completed during team meetings

3. Strategies for keeping on task:

Keep a timeline for the project to give us an idea if we are ahead of schedule or behind so that we can plan accordingly. Manage time well and complete a reasonable amount of work each week. Make sure tasks are well defined so that members don't need to spend a lot of time finding requirements

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

Initially confront the individual privately and determine if the infraction needs to be brought up at a team meeting. If the issue cannot be handled at the team meeting then escalate to our TA/Professor and Advisor if needed

2. What will your team do if the infractions continue?

Bring up the issue with our TA/Professor and Advisor

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) _____Tyler Miller_____ DATE ____12/1/22_____
2) _____Brandon Burt_____ DATE ____12/1/22_____
3) _____Alex Swenson_____ DATE ____12/1/22_____
4) _____Adam Walters_____ DATE ____12/1/22_____
5) _____Guan Lin _____ DATE ____12/1/22_____

6) _____Jaden Ciesielski_____ DATE _____12/1/22_____