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 started implementation of websockets to ensure effective roundtrip communication between the frontend and backend.

4.3.2 Detailed Design and Visual

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 with hostname sdmay23-40.ece.iastate.edu. Our database is running within a docker container deployed on the server. Our node application is an Express server running on port 6000 and the server uses pm2 for load balancing. Below is a diagram of our backend:

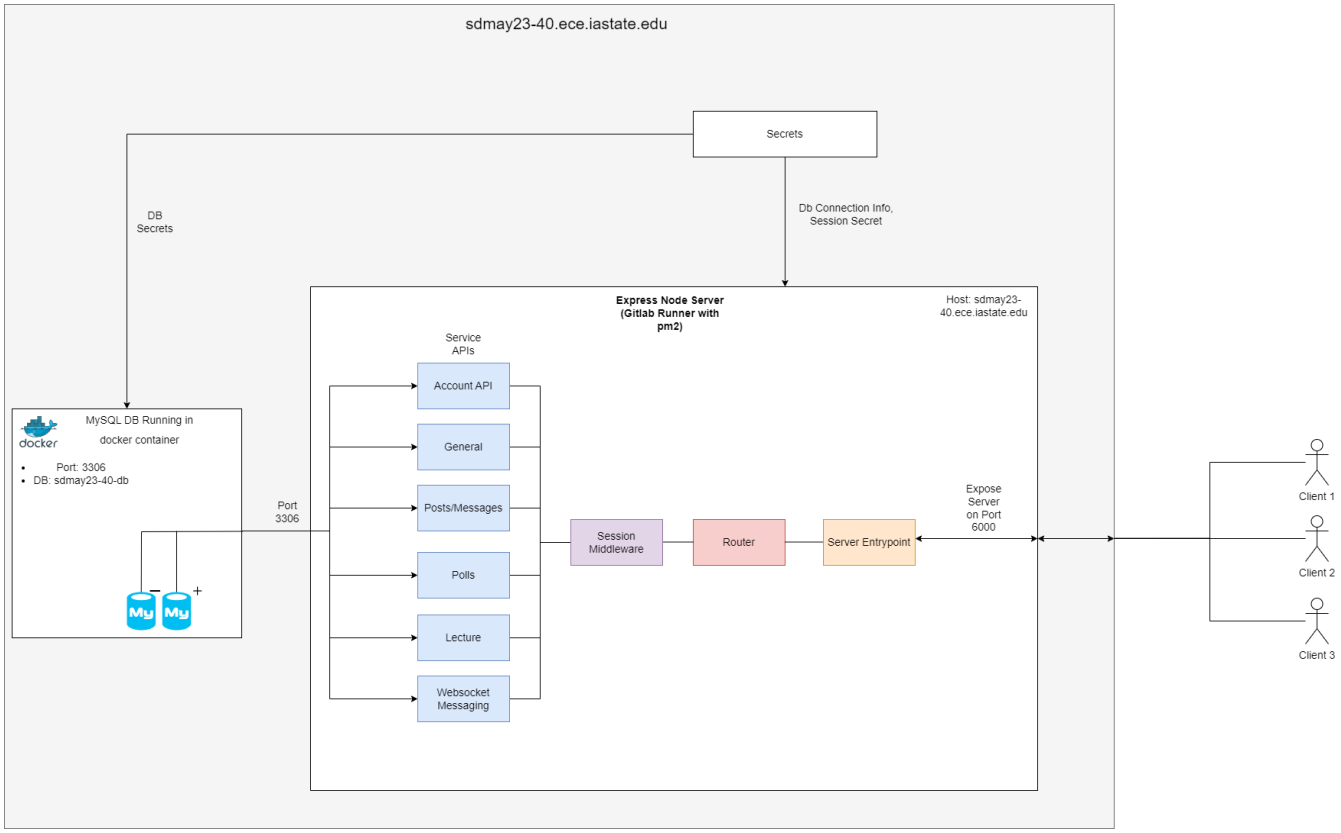

Figure 1: Backend Block Diagram

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.

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 6000. 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.

4.3.2.3 Database

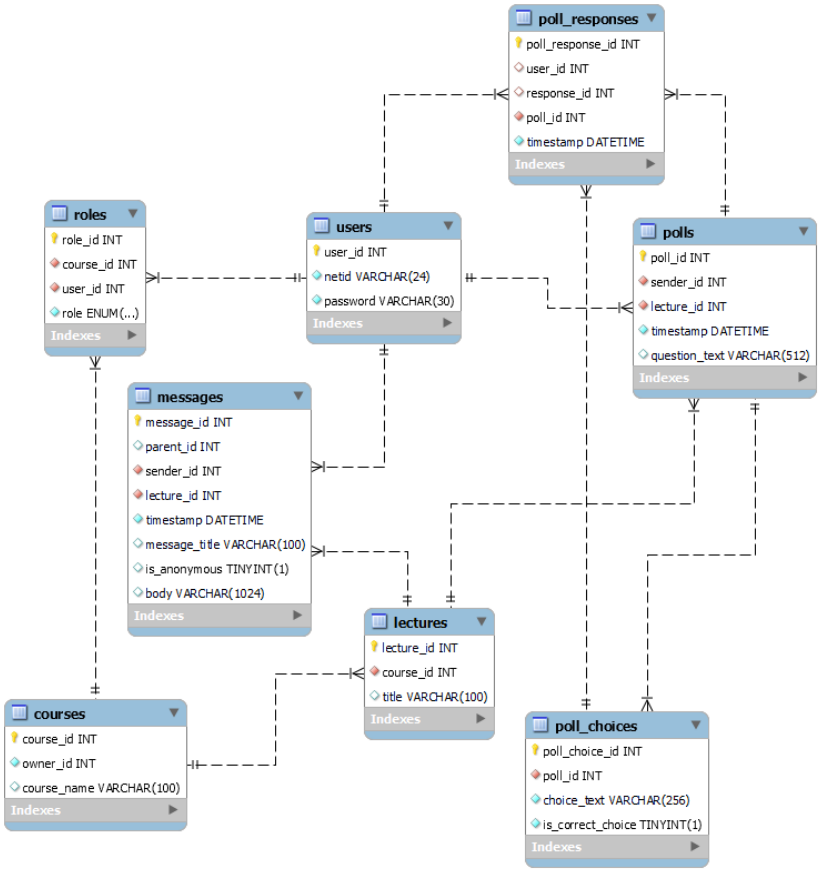

Figure 2: Database schema

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 tables is responsible for storing a user's selected option(s) by providing a link between

4.3.2.4 Backend Deployment

We have implemented a CI/CD pipeline for our backend 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.

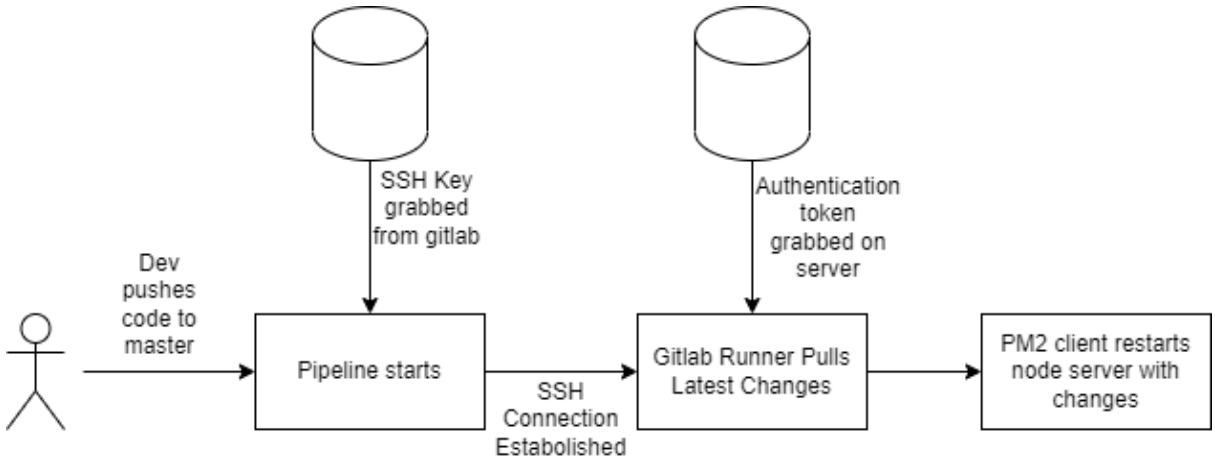

Figure 3: Pipeline flowchart

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.

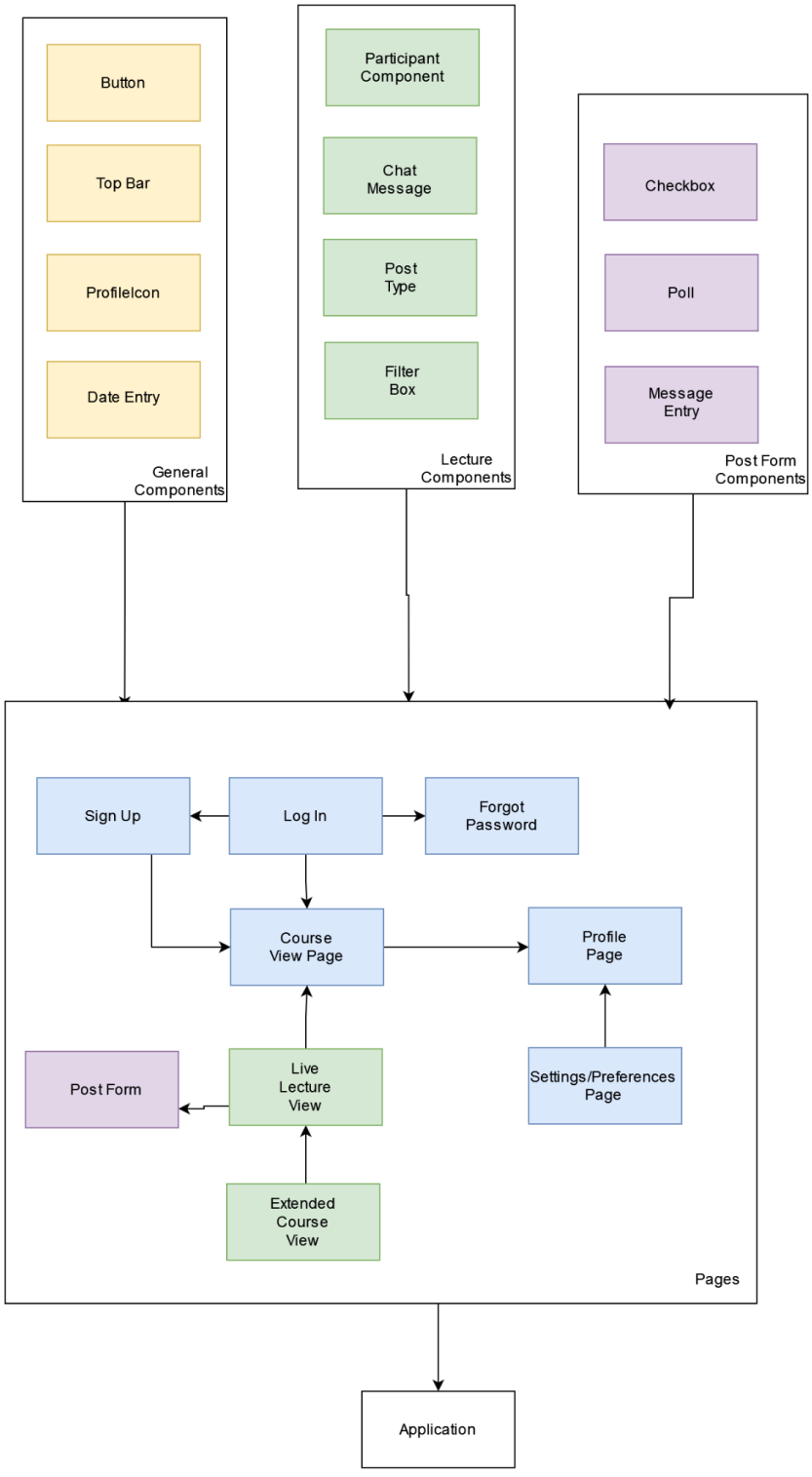

Figure 4: Frontend Breakdown

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.

4.3.2.6 Frontend Page Designs

Below are the designs for the primary pages that the frontend will be implementing.

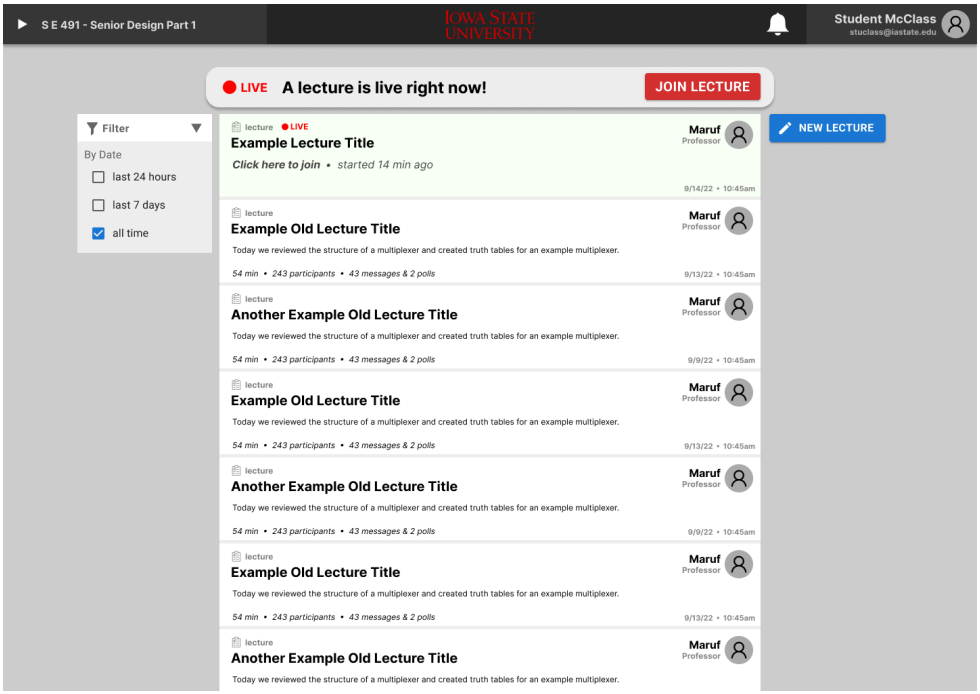

Figure 5: Course View Page

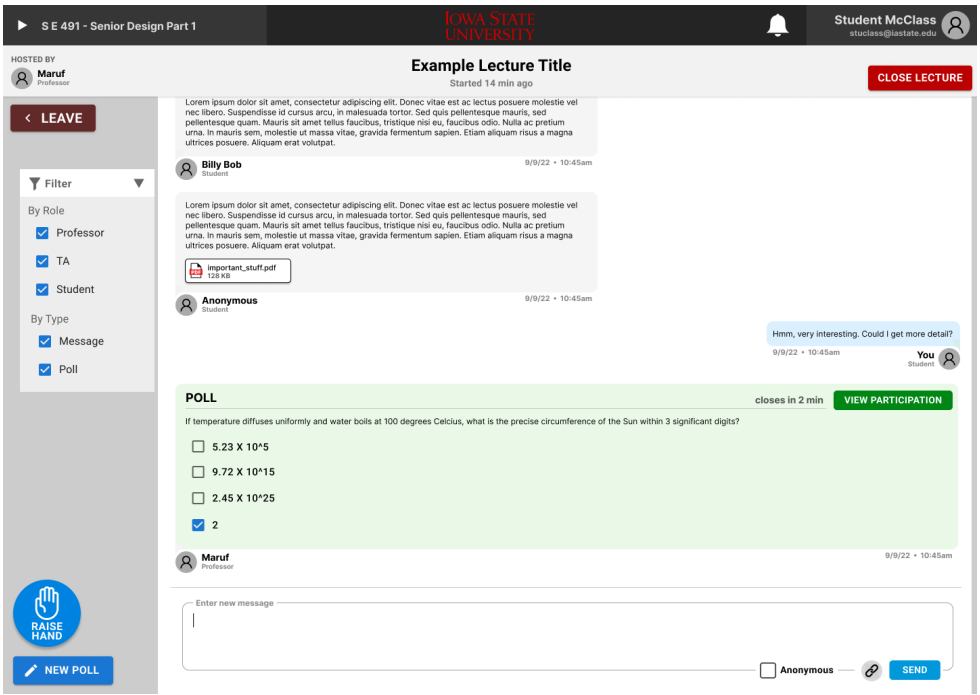

Figure 6: Live Lecture View

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. 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

We have built out an API design and started mapping variables for a database. Frontend has finalized page layouts, and worked with the Backend to assure the API design will meet the design requirements. There has been some component development thus far on the front end as well.

We went back to the drawing board on our page design, because after discussing some matters between the front and back end the redesign we are proposing will allow for a more seamless integration of the front and backend and will also meet some of the needs desired by our project advisor.

Once we meet with our advisor today we should have enough of the right information and validation to push forward in the development cycle on a much larger scale. Our plans are to have a basic functioning MVP produced soon once we have the API's, Database, and Frontend Structure all figured out and finalized. Once we achieve all of these goals this week, our goal should be completely feasible.