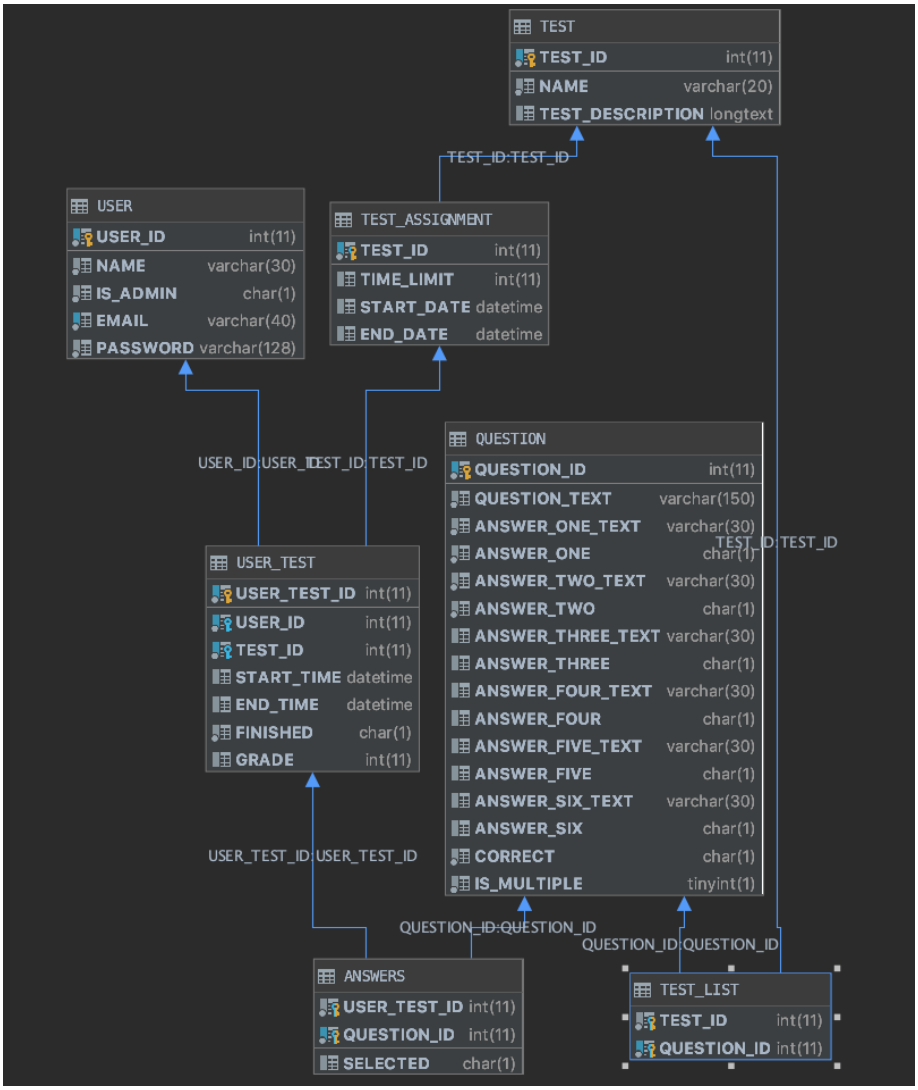
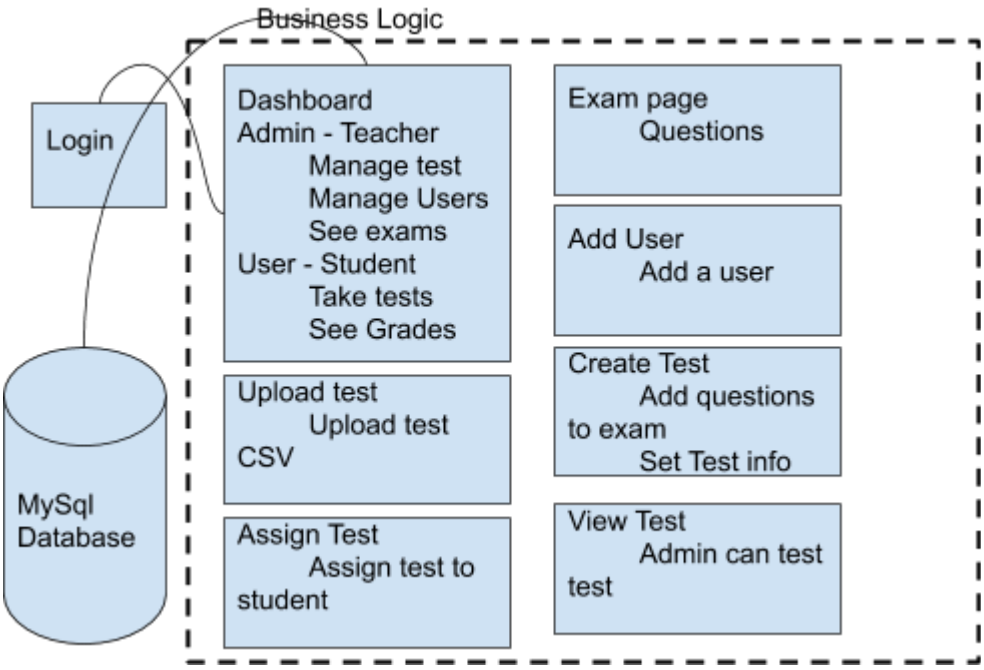
Design Documentation

Team Members: Kostin Galusha, Adam O'Leary,
Cory Marriott, Ari Chai, Sean Rowland, Robert Kats

The Stack

For the web app we decided to use React as it has massive libraries for support. This allowed use to focus more on functionality rather than deeper back end requirements. For example we the axios api for help with server to client communication, this allowed us to focus on what to communicate rather than how. We used Nodejs for a similar reason it allows as it allows to run javascript without the need for a bowser, this allows use to work in the same language for both the front end and back end. For our database we decided to implement a MySQL server because our group was familiar with it and we could work efficiently with it.





The Interface

Login:

We modeled the login page to be clean, simple, and visually pleasing. It has two text boxes for email and password for login and a login button.

Admin dashboard:

The admin page shows options for test creation, question creation, CSV uploading, and student account creation. Below we have the created tests with options to view, remove, and assign. Under that we have the question bank. The admin page is designed to be simple and easy to navigate. All of the core functionality is easily accessible.

Student dashboard:

The student dashboard shows the user information with an option to update the password. This is an important feature because the student will initially be given a password from their admin. The student also sees the exams assigned to them and relevant information such as the time and date. The bar also has a buttons to begin the exams. A conformation is presented if the exam is timed in the case that the student did not mean to start the exam.

Goals:
The goals of the project is to design a web application what allows students to take an exam that a teacher has created. The teacher is able to create student accounts, make exams and questions and import exams from a csv file. The exams may have a timer and give a grade back upon completion.

Behavior:
Quizzical is a multi-page web app. The backend runs on Nodejs to communicate with the React frontend and the database is MySql running on AWS for all the user data. Each page is a React component that renders when on the page. When going to another pages React renders the page, asking the server for user data. The backend queries the database and send it to the front end. The exams use a React question component to create a list of the questions. This allows for an auto generated list without the need to create each question individually.