# Introduction

This application is a website where a professor can create quizzes for his students. A quiz exists out of multiple questions with four answers each. A professor can use the quiz to test the students their knowledge while giving a lecture. He can show live results on the screen and track how his students are doing.

The description above isn’t a new concept. There are already web applications available that provide functionality as described before. One of those is Kahoot. Kahoot is a website where you can create your own quiz and let people take them real time. When someone is using Kahoot, they will display the questions on a large screen and students must answer the questions on their own device. The professor decides when the class is going to the next question and at the end there is a leaderboard with all the people that took the quiz.

Another example of a web application that provides almost the same functionality is Quizizz. Quizizz is different from Kahoot, that large screen doesn’t show the questions. The screen will show a live leaderboard. This means that the students see the questions on their own device. Therefore, the quiz pace is user player-paced. All the students take the questions at their own pace.

So why develop another quiz app? The problem with apps like Kahoot and Quizizz is that you can’t save the data. When a quiz is finished you will never be able to see the results after closing the website. Everybody can join a quiz without signing in with an account. So these apps are great when you are taking a quiz with groups that you want to test once, but when you want to test people on a regular base these apps lack the ability to track people their results.

How is this Quiz App different? First, the professor has total control of the people who are taking the quiz. If someone isn’t registered for the course, he wouldn’t be able to register himself. Second the professor can see all the actions made by students because of a built-in transaction log. The rest of the applications shows a lot of similarities with Quizizz. Students answer the questions on their own device and the professor can track the answers on a large screen. A handy feature is that the professor can show the correct answer on the screen afterwards.

Is it worth it to build it yourself? Yes, it is, because the features mentioned above aren’t possible in the free applications that are available. The advantage of your own system is that you can add more and more functionalities. While apps like Quizizz and Kahoot will stay the same. An example of future functionality can be the possibility of adding roles for student assistants. This way they can assist the professor by creating quizzes.

# User Guide

When a user starts the application, he will arrive at the login page, because logging in and registering are the only functionalities that are available when a user isn’t signed in. The first user that registers himself at the application will be given the admin role. This is important, because after that no student will be able to register until the admin registers the students that are taking the course.

## 2.1 Admin

When an admin logs in the application he will arrive at the admin overview page. On the admin overview page, he will have five functionalities to his disposal.

### 2.1.1 Managing Students

The first time the admin starts the application he needs to register students for the course. At the student managing page he can insert student identifications separated by a comma and add them to the list. He can also remove students and revoke their right to register for the course. Only students that have their identification store in this list can register for the application. However, removing a student that already registered will do nothing to their already existing account.

### 2.1.2 Quiz creator

Before the students that register can start a quiz the admin needs to create one. In the quiz creator the admin must fill in a quiz title and a quiz code. The title is just for showing and has no functional value. The quiz code however is the code that students must enter to start the quiz. Besides a title and code, a quiz needs questions. The admin can insert multiple questions and remove them if they aren’t needed anymore. He can select the correct answer with the radio buttons above the answers. When finished with the questions the admin can decide to save the quiz or remove it if he isn’t satisfied.

Another possibility is that the admin visits the quiz creator, because he wants to edit a quiz. When that is the case the quiz will be editable, and he can adjust the questions or answers. He can choose to remove the quiz as well.

### 2.1.3 Live Quiz

At the live quiz page, the admin can see the students their answers real time. By entering a quiz code, he can see the current answers that students are giving to the questions corresponding with the selected quiz. The admin can also decide to show the correct answer.

### 2.1.4 Quiz Overview

The quiz overview page shows all the quizzes that have been made by the admin. He can choose to edit them from the overview or he can choose to remove the answers that have been given by students. This way the quiz can be used again, and the data isn’t corrupted with answers of previous years.

### 2.1.5 Transaction Overview

The last functionality of the admin is the transaction overview. Here the admin can see all the actions students did during the quiz. Did they start a quiz, did they dodge a question, did they answer a question, or did they finish the quiz? To make the data more accessible there is a filter for the student identification, event, question number and session identification.

## 2.2 Student

When a student logs in the application he will arrive at the student overview page. On the student overview page, he will have only one functionality to his disposal.

### 2.2.1 Quiz taker

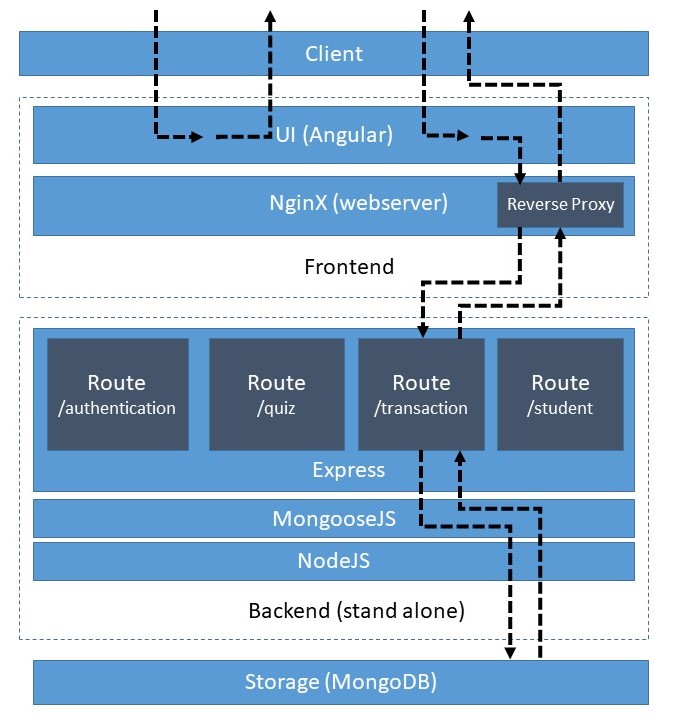
When a student starts the quiz taker he must use the quiz code provided by the admin. If the quiz code is correct, the quiz will start. When a student selects an answer, he must push the button select answer to save his answer. Otherwise when he will go to the next question the answer won’t be saved. After all the questions are saved the student can finish the quiz by submitting it. After submitting the student will receive a percentage of correct answers and he will have the possibility to view his answers and the correct answers.

# Technical description

The design of the application is based on the full stack MEAN JavaScript framework. This stack contains:

* M: MongoDB 🡪 Database
* E: Express 🡪 Rest API
* A: AngularJS 🡪 Web interface – client side
* N: Nodejs 🡪 Server-side JavaScript

The only difference in the design of the application is that AngularJS is replaced by the newest version of Angular.

The application has a three-layer design.

* Backend
* Storage
* Front-end

De front-end is developed in Angular. The front-end handles all the logic in the client. The entire front-end application is built in HTML and JavaScript and runs in its entirety in the browser. This means that the application has multiple responsibilities. It must handle navigation, client-side logic and the communication with the backend.

When the application is running in production it is being served by an NGINX web server. This web server serves the web application and handles the routing to the backend through a reverse proxy. When there is a call to ‘/api’ NGINX will redirect it to the backend application. For example, when there is a call to http://quizapp.nl/api/quiz it will redirect to http://backend/api/quiz

Backend

The backend is built on NodeJS. The backend is an abstraction layer between the client and the database. It handles the services calls from the front-end and communicates with the database. The backend it’s responsibility is handling all the data that goes to the front-end and that is coming from the front-end.

MongoDB is a NO-SQL database and that means that it has no relational functionalities. For that reason, there is a need of an object-relational mapping framework. This ORM framework is MongooseJS. MongooseJS makes it possible to define strongly typed object models and add validation to these models.

The REST API’s are defined by Express. In the backend there are multiple routers that support the front-end application in separated functionalities. When there is a web request it will first go through the server and depending on the route it will be navigated to the corresponding router. For example, /api/quiz will be redirected to the quiz router.

In the routers the data will be processed and stored in the database. The objects defined in MongooseJS will be used to store relational data in the MongoDB database. Using these objects are making sure that the data will be easy to access by queries and CRUD operations.

There are some differences between the production and local development scenario. The working of the application is the same, but the hosting and management of the different processes are different in production and local development.

In production NGINX will be used to serve the front-end, just like the reverse proxy to route the web requests to the backend. The backend will be hosted in PM2 in production. PM2 hosts the server process and make sure it will get restarted when it’s crashes. On local development the front-end is being hosted in a node process that is started by the Angular CLI. This server also has the responsibility for the reverse proxy. This configuration is in the proxy.conf.json file. The backend is hosted in a process that is started by gulp. This gulp task also watches for changes and make sure the code is recompiled every time a change is saved.

# Conclusion

I believe that this quiz application can be useful for a lot of professors that want to test their students during the lectures with short quizzes. Because of its lightweight architecture and design the application runs smoothly and can handle more than 100 students simultaneously. The Angular front-end is great for all devices and allows the use of nice components that can be integrated easily. Like the error message popups. The development on NodeJS and with the Angular CLI make it so that you can cover a lot of ground in a short time. In less than two months I’ve developed a stable application that does it work properly.

I do think that if a future developer wants to improve or expand the application he must consider the possibilities of replacing the MongodB with a SQL relational database. MongoDB offers support for quick development, but the lack of a good relational object management framework makes it hard to store data in a good and organized manner.