Step 1:

Going to MongoDB to setup the cluster, and then we connect to it using our own IP address. We will also get a link that we later need to connect to the server.

Then, we can add sample data to the database, which we can do by just loading some sample data that MongoDB already has.

Step 2:

Creating the backend for the app, by making a folder for it under the main project.

Then we create a package.json file by running npm init and installing various dependencies such as express, mongodb, etc.

Express is used for the web server. Cors is cross origin resource sharing, and you can access resources from remote hosts, and we can make the correct connections. Mongodb allows us to interact with the MongoDB database. Dotenv loads environmental variables and stores it into a file.

Nodemon makes it so we don’t have to restart the server every time we make an update to our file.

Once we run nodemon server, we’ve successfully connected to the database.

We then create a dao folder (data access object)

Now, we can load the server, and we should see a bunch of data right in the browser. We can also just test it in the browser, but a lot better to use a tool to test it, in this case, we will use Insomnia. Insomnia and postman; however, both basically do the same thing

We test out the filters through postman, and we see that name filter doesn’t actually work, because we need to set it up in MongoDB atlas. We will create a new index, with name: text.

The whole point of the app is so people can leave reviews. We need to add routes so people can make posts and edit them.

Now that we have the files, we need to test the review part by making a POST request and sending in a JSON in postman.

After this, we check if we can edit it or not using a PUT request with the objectID of the review object that’s created with a POST request.

This way of authenticating the user is not the most secure way, and finally, we try to see if we can delete the review that was made.

Now we will add two more routes to the route file, and the methods corresponding to them.

We then need to test these two routes. For the cuisines route, we need to type a GET request with cuisine after the URL.

For testing getRestaurantByID, or all the reviews associated with the restaurant, we need to make some reviews and see if it comes up or not.

We are usually not going to get an individual review, so we don’t have a route for that.

Step 3:

Now, we will make the frontend and connect them.

We will create our frontend with react, so create-react-app frontend, and we will also install bootstrap as a framework to make designing our app easier. We won’t use any custom css, so we will just use bootstrap.

We will also install react-router-dom, to route different pages.

In the source folder in frontend, the entry point for our react app is going to be App.js.

We will first create several components that allows people to route to them.

We also need to install axios to support html services and stuff.

We currently do not pagination, but should be easy to add on the frontend.

The key is that we need to keep the backend server running when we have the frontend, so this way the server is always on and the frontend can receive data.

Now, we move on to making a page for the individual restaurant. The three dots in JavaScript is called the rest operator and is used to indicate that there are a variable number of arguments.

We’ve now implemented the individual restaurant page where you can delete or edit and view the previous reviews already made. However, we can’t edit or delete your own reviews yet because you’re not logged in. So, we will now make a login page.

Step 4: Replacing Backend and Node Express with MongoDB realm

We will also host the frontend in the cloud.

In other words, based on the diagram below, we will replace the R section and the EN section of the app.

Graphical user interface, application

Description automatically generated

After replacing everything, we will get something like the following:

Graphical user interface, application

Description automatically generated

The locks indicate added security that comes with using MongoDB.

Create an application in MongDB realm called Restaurant Review.

MongoDB removed their third party services and etc., we we’re going to do the same thing but with MongoDB functions.

The authentication can also be done with Google and Facebook and it’s easy to setup in MongoDB.

We will also need to create a MongoDB webhook to connect them. Make this a GET request.

We will now improve the function to support searches, doesn’t really work anymore, and it seems like they’ve changed the dataset.