**CuisineShare – Recipe sharing website**

**Team members:**

Eric Craaybeek

Huan Ngo

**Detailed Project Description:**

CuisineShare is a recipe sharing website. With this website users will be able to login and add recipes or browse other recipes that other users have uploaded. There will be support for pictures and recipe favorites as well.

We will be developing the project using the MERN stack. The database will be implemented using MongoDB, front end will be implemented primarily using react and react bootstrap, backend will be implemented using node.js.

**Front-End:**

The front end is developed using react and react bootstrap primarily. There are 7 pages currently planned: BrowseRecipes, CreateRecipe, Login, Profile, RecipeInfo, Register and Welcome.

The Welcome page has already been developed as it is very simple. It just includes a picture background with a handful of buttons that take you to some of the other pages as well as a nav-bar. The nav-bar is standard and based one whether you’re logged in. When not logged in it shows links to: home, all recipes and login. The page itself just has a button for add recipes or view recipes. Add recipes will bring you to the create recipe screen if logged in, or the login screen if not logged in. The view recipes page just links to the all recipes page.

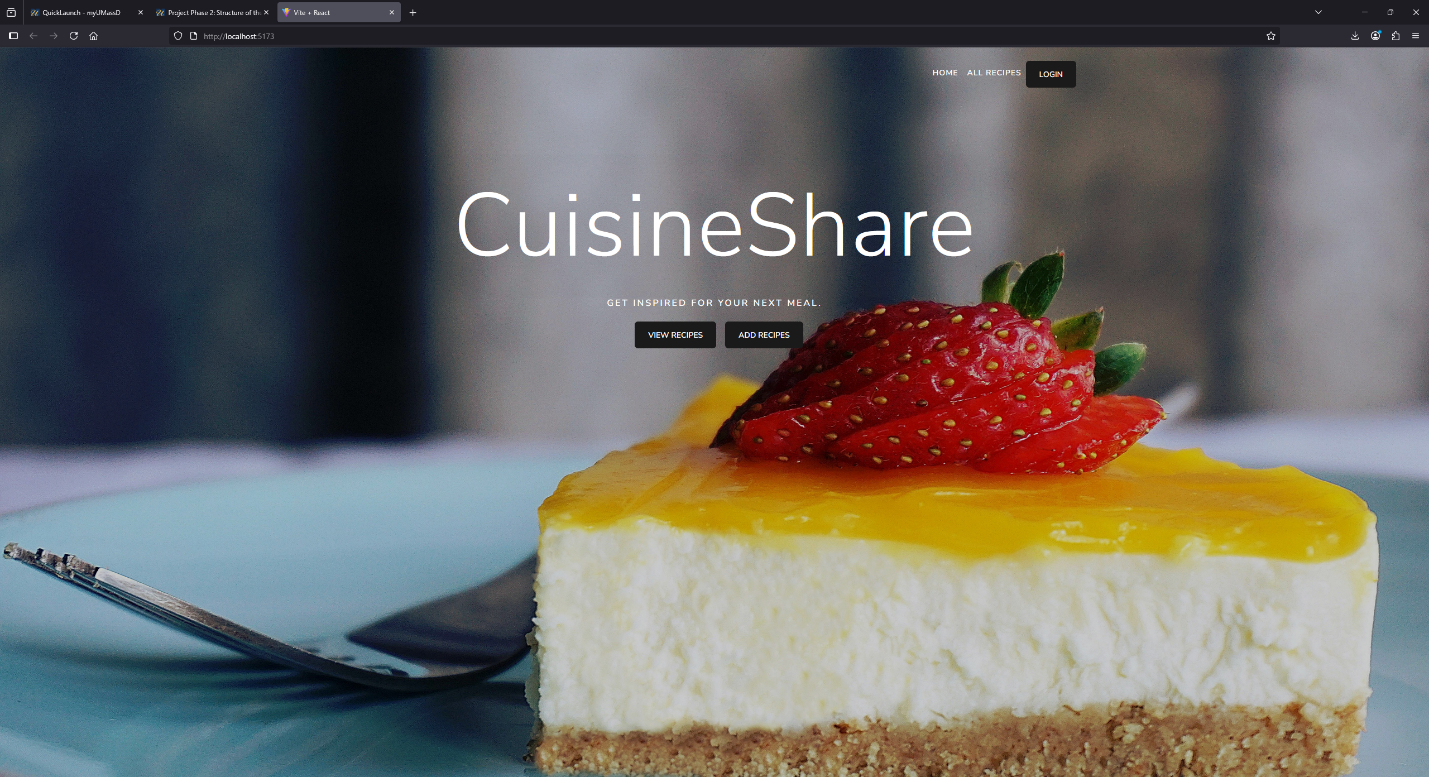


Figure 1 – Welcome page

The login page has had some development completed. It uses bootstrap to place login forms and buttons in the center of the screen. It will also have the same standard nav-bar as every other page.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 2 – Login page

The create recipe page and register user pages are similarly developed. They each have a handful of forms with submit buttons. Both use bootstrap for layout and input elements. The forms just each have different labels. Login, register, and create recipe each interact with an endpoint that associates with them described in the back-end section below. (/api/auth/login, /api/auth/register and /api/recipesubmit respectively.)

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3 – Register user

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5 – Add recipe page

**Back-End:**

Since we’re using the MERN stack, we’ll be using MongoDB for our data storage needs, and NodeJS with Express to interact with it. Two collections in MongoDB have been created: recipes and users. A mongoose schema has been created to work with each of these. See tables 1 and 2 for descriptions of these schema’s.

|  |  |
| --- | --- |
| **Name:** | **Type:** |
| Name | String |
| Time | Number |
| Description | String |
| Ingredients | Array (of strings) |
| Instructions | Array (of strings) |

Table 1 – recipe schema at current stage

|  |  |
| --- | --- |
| **Name:** | **Type:** |
| Username | String |
| Email | String |
| Password | String (Hashed) |

Table 2 – user schema at current stage

The users DB is interacted with via two endpoints. These are both called via “POST” messages depending on the URL used. (either “/api/auth/login” or “/api/auth/register”) Register adds a new user entry to the database. It uses bcrypt to hash the password. Login uses bcrypt for password verification. Both of these end-points have already been developed.

There are at least 3 more planned end-points: “api/recipesubmit” , “api/myrecipes” and “api/allrecipes”. Recipe submit has already been developed, it uses a second schema (Recipes) to submit a new recipe to the second collection in the MongoDB (recipes) The required fields are name, time, description, ingredients and instructions. Ingredients and instructions are both arrays of strings that will be formatted out nicely once retrieved by the front end. Time is a number value that represents the time in minutes, it can also be converted on the front-end to a more legible format. See table 3 for a list of all currently planned endpoints.

|  |  |  |  |
| --- | --- | --- | --- |
| **Endpoint URL** | **Type** | **Purpose** | **Return** |
| /api/auth/register | POST | Register a new user | Success msg |
| /api/auth/login | POST | Login using existing username/password | Success msg |
| /api/recipesubmit | POST | Add new recipe to the site | Success msg |
| /api/myrecipes | GET | Query all recipes connected to your user | All favorite and created recipes |
| /api/allrecipes | GET | Query all recipes | All recipes |
| /api/viewrecipe | GET | Query 1 recipe details | Details for 1 recipe |

Table 3 – Currently planned back-end endpoints

All endpoints are using JWT for authentication. “register” and “recipesubmit” have been created and tested with postman. The pictures below show successful submissions to the database via postman as well as the entries in Atlas MongoDB webpage corresponding to this.

Figures 6-9 depict testing on some of the endpoints as well as the successful storing of data within MongoDB within the correct collection.

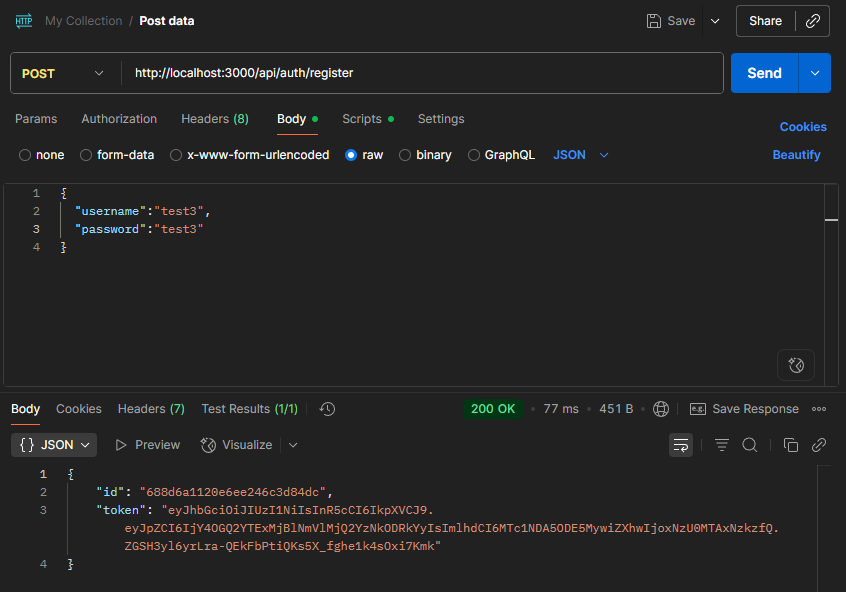


Figure 6 – Successful user registration

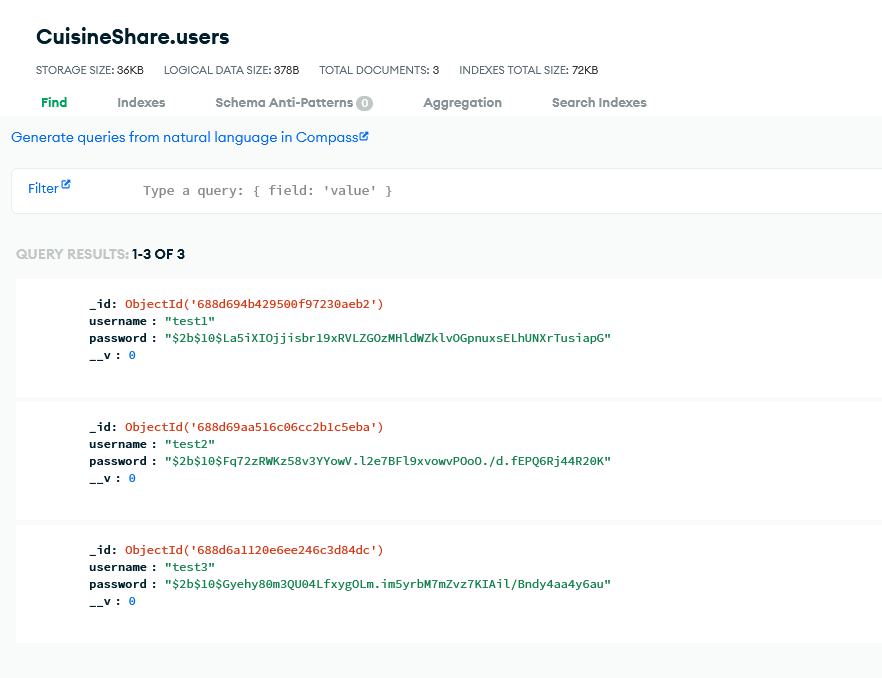


Figure 7 – Users successfully saved to CuisineShare.users on MongoDB

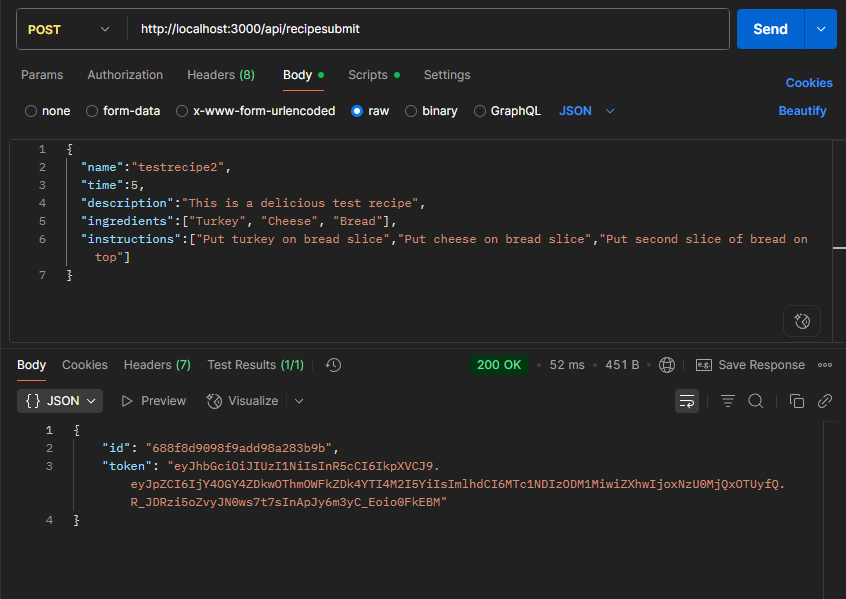


Figure 8 – Recipe submitted to MongoDB successfully

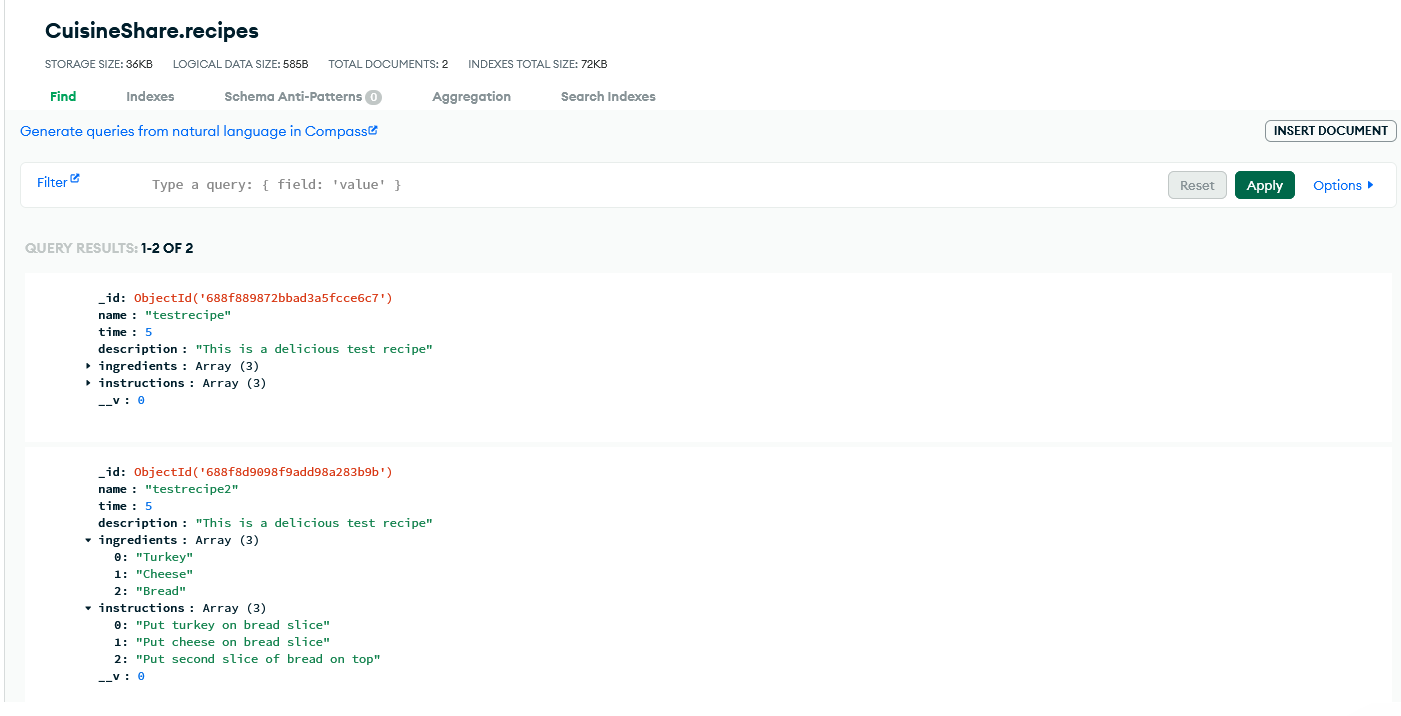


Figure 9 – Test Recipes shown in CuisineShare.recipes in MongoDB

**Diagram:**

Figure 10 shows a diagram that details the entire planned site. It uses arrows to show how buttons will navigate to other pages on the site. Endpoints are labelled by there URL where applicable (either on the page or as part of the navigation arrow). The nav-bars shown on the top will be present on every page to allow quick navigation to anywhere on the site with 1 button click. For a better view of the diagram see /submissions/pictures/CuisineShare.png (or if you have drawio installed CuisineShare.drawio). This directory is contained in the submitted code files.

A diagram of a computer program

AI-generated content may be incorrect.

Figure 10 – Full site diagram