Web Application Final Project

Shane Burk – x21110018
Moses Ademokun – x20126204

Recipe Book

Web Application Development- BSCHE2

Research

The increasing adoption of a faster lifestyle is a key factor driving the growth of the global recipe apps market. Recipe apps enhance convenience, which, in turn, saves time in preparing food. It also encourages consumers to venture outside their comfort zones regarding ingredients and preparation techniques. Vendors ensure that the recipe apps provided to the consumers have easy-to-follow recipes, cooking tips, as well as the exact amount of ingredients required to prepare the meal in a convenient way. This helps single-person households and double-income households to get healthy prepared meals.

(Technavio, 2023)

Objectives

For our final project, we have been tasked with creating a rich internet application that supports both front and backend operations. We are also required to use a database to store and retrieve data.

For our idea, we have chosen to create a recipe book web application that allows the user to store recipes on a database. In addition, to display these data on a browser.

We have implemented functionalities to enable CRUD operations, which is to Create, Read, Update and Delete items from our MongoDB database. We have documented our process in line with key web application development stages which helped to complete our working recipe app.

Further down this document, we have sectioned each file within our project, to illustrate its inner workings.

Development process and Milestones

1. Requirements review.

During our development process, we began with thinking about what type of app to be built which will employ the usage of technologies set out in the requirements. We initially floated the idea of an e-commerce clothing website and a food recipe book. However, after doing some research, we found this task of the e-commerce website to be too complicated to do, given our level of experience and time limit to complete. Hence, we decided to go with the idea of a recipe app. This allowed us to quickly gain more clarity on how we will proceed in achieving our application goals, features and technology for our food recipe app.

We then began discussions, focused studying and researching on how these requirements will be met.

Our food recipe app would allow us to store some information and run CRUD operations on it.

2. Planning & blueprints

To begin, we created the foundation of our project, which was a blueprint including flowcharts and sketches that helps determine the overall structure of our recipe web application. We did not build wireframes as we did not have enough resources for this.

3. Web application design

We decided on the final structure of our recipe app and tried a few design iterations and mockup changes go on until we were happy with the structure and styling of our recipe app.

4. Web application programming

We installed Visual studio code ide and created a visual studio code main folder, with the index.html page, app.js server file, and some partials for holding the head, nav, and footer sections.

We then created our database on MongoDB.

Technologies used:

Tools:

Visual Studio Code.

Database:

MongoDB.

Scripting methods:

HTML – Frontend scripting.

CSS – Frontend scripting.

JavaScript- Frontend scripting, to add dynamic behavior to our web application.

AJAX- Sending data from front to back end by asynchronous exchange of data between client/server side.

Node.JS – Server-side scripting.

JSON- Server-side scripting with express.

Express – Framework for front and back-end scripting of data. Also used for managing our server and routing our webpages.

MongoDB – Storing items from web page.

Mongoose – Server-side scripting used to work with our MongoDB database. This package allows us to use valuable functions for making changes on the backend.

- WebAppCA2
  - Models
    - Recipe.js

      Package: Mongoose

Recipe schema: MongoDB database schema (recipeName, ingredients, timestamp).
database export for JavaScript usage of our Recipe database.

- Public
    - CSS

      Some webpage styling, forms, HTML elements etc. Although we have used Bootstrap for the most the styling on the webpage, some divs, and html elements required more specific style properties for better layout and visuals.
    - Images

      Logo.png. This logo is intended to be placed within our nav bar.
      Favicon.ico, for the browser tab window.
- Views
    - Patrials
        - Footer.ejs

          One horizontal line.
          Line copyright sentence with symbol.
        - Header.ejs

          Metas tags, a link to style. CSS, a link to favicon icon, the title with EJS Link to title each webpage title within our app.js file.
          Link for bootstrap usage.
        - Nav.ejs

          A nav wrapper, with bootstrap styling. Container Div. which holds the logo.png image, a list of the links to the Home and Contact page. Finally, an input box and button to search for items.

404.ejs

Header.ejs patrial, Nav.ejs partial, Error 404 text message and the Footer.ejs partial. This page is used in the event of an error getting a webpage from our project.

Contact.ejs

Our contact page contains the same partials as above, with a div containing a submission of queries box. This box will send the contents to a second collection within our database, for the admins to view and respond to.

Index.ejs

- Header.ejs partial this file contains all the html elements that are contained in the head section of our html document. Partials allow us to reuse code that is to be displayed on multiple pages, saving time and unnecessary rewriting code.
- Nav.ejs partial
- Add recipe button, onclick uses an even listener to show form used for inserting a new recipe.
- Close button to hide the add new recipe form if the user does not wish to add a new recipe.

- This form contains a POST method which will send contents to the backend for processing. These items are added to the database.
- A container div to display all current items in the MongoDB database. Express.js is utilized here, by looping through the contents of the database collection, Recipe. This collection is instantiated in the recipe.js file and exported for use. Each recipe name and ingredients are displayed within HTML elements on the index page. If there are no blogs, a sentence should display "Sorry, there are no blogs yet."

JavaScript:

- Display() gets elements by ID. Form, close and add buttons, changes their styles from none to block, and vice versa.
- Hide() gets the same elements as Display(), however does the opposite for styling to hide the form once the close button is clicked.
- An eventListener for the delete button attached to each database entry added to the main page. This listener gets the id of the specific dataset relating to the button and uses an AJAX request to delete this entry by sending it to the backend app.js file. Once this is deleted, a request to redirect user back to the page is made, to refresh the contents.

Recipe.ejs

The recipe.ejs file is what we have used to edit single recipe details. This is done from the index page, by allowing each item on the list to be a clickable link, with the data-id attached. Within the index page we have a forEach loop to display each item. With every item posted, a href link is attached to it that pulls the id for each recipe. This gives us the ability to open this on a new page, localhost:300/id.

Two divs are placed in the recipe.ejs file, one to show the item details, and one for editing the details. Two buttons are also shown here, eventlisteners are linked to some JavaScript for hiding and displaying these divs once clicked.

In the editRecipe div, we have an action ":id' and method "post". These attributes allow us to do two things, pull the id from the database and display that single one, and use the "post" http method to work within our app.js file, using app.post(). Here we can update both the database and the contents on the webpage.

App.js

Three imports at the top of the page, for using express, a backend framework for building APIs with node.js, mongoose, which allows us to work with MongoDB data modelling for Node.JS and the 'recipe' database model, in the model's folder.

Const app = express(). We have used this const to work with the express package for setting the view engine, for ejs files, started the server on port 3003, and to work with static files within the workspace.

Next, we made a variable to hold the data access key, with the username and password embedded, and a mongoose function .connect() to establish a connection with the database.

Using app.get() we have three pages for our web application, the main index, the contact page and the individual ID for each item in the database. This procedure allows us to make changes to individual database items, such as delete and update.

app.post() is used for the main index page, to allow the server to retrieve the contents of our database and send them to the index.ejs file "/".

A second app.post() function is used for the individual "/:id" pages, to display the singular contents of each item, based on the id. Two objects are created here, to update the new details and the: id.

In the index page, we used a method with AJAX to send an item to the backend for deletion with DELETE function. In app.js we use app.delete() to get this id, use findByIdAndDelete() built in function to do the heavy lifting in removing this item from the database. Finally, a redirection is requested to refresh the page, for the change to be visible.

Finally, an app.use() is made again to get the 404 page, which is to be rendered in the event of an error in retrieving any valid webpage on our project.

5. Testing & launch

We conducted various testing of the flow of all the elements on our website to ensure that all the functionalities were working well before submitting our project

Advantages of MongoDB over RDBMS for our recipe App

- **Schema less** – MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear and can also be seen in the console
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- **Ease of scale-out** – MongoDB is easy to scale. Our recipe app could be easily scaled up
- Conversion/mapping of application objects to database objects not needed.
-

Conclusion

There are different development process and technologies which can be used in web App development. Whilst HTML, CSS, JavaScript, AJAX, JSON and MongoDB were used for our recipe app, other technologies such as react JS can equally be used for a quicker less expensive web app development process. It much depends on the scope and project timeline of the web app in question when choosing technologies for its development or implementation.

Link to Video presentation

One drive

2023-08-08 Shane_Moses_ 1.avi

SharePoint

https://studentncirl-my.sharepoint.com/:v:/g/personal/x20126204_student_ncirl_ie/EXqqX449EsJChqqateOl6JEBOJgngqP06pTyW9tZBSOxdA?e=fjlPrX