Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Bia Blasta

Project Engineering

Project Proposal

Clíodhna Ní Niaidh

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Atlantic Technological University

2022/2023

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Atlantic Technological University Galway.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

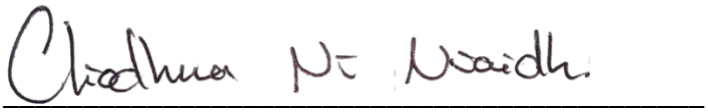


Table of Contents

1 Summary 5

2 Poster 6

3 Introduction 7

4 Technology 8

4.1 React 8

4.2 Next.js 8

4.3 Express.js 8

4.4 Node.js 8

4.5 Mongoose 8

4.6 MongoDB / Mongo Atlas 9

4.7 Amazon S3 9

4.8 Auth 0 9

5 Project Architecture 10

6 Project Plan 11

7 Project Code 12

7.1 Frontend 12

7.1.1 handleChange and handleSubmit function 12

7.1.2 useEffect Hook 14

7.1.3 handleFavouriteClick function 15

7.2 Backend 16

7.2.1 GET Endpoints 16

7.2.2 POST Endpoint 18

7.2.3 PUT Endpoint 18

7.2.4 DELETE Endpoint 19
7.3 Mongo Atlas 20
7.4 Auth0 20
8 Ethics and Sustainability 21
9 Conclusion 22
10 Frontend Results 23
11 References 26

1 Summary

Bia Blasta is a MERN Stack project that is designed to provide users with customized recipe recommendations based on ingredients they have available to them. The ingredient selected by the user is filtered through the recipe database, resulting in a customised list of matching recipes for the user to try. This is a unique solution to help reduce food waste and promote sustainable cooking practices.

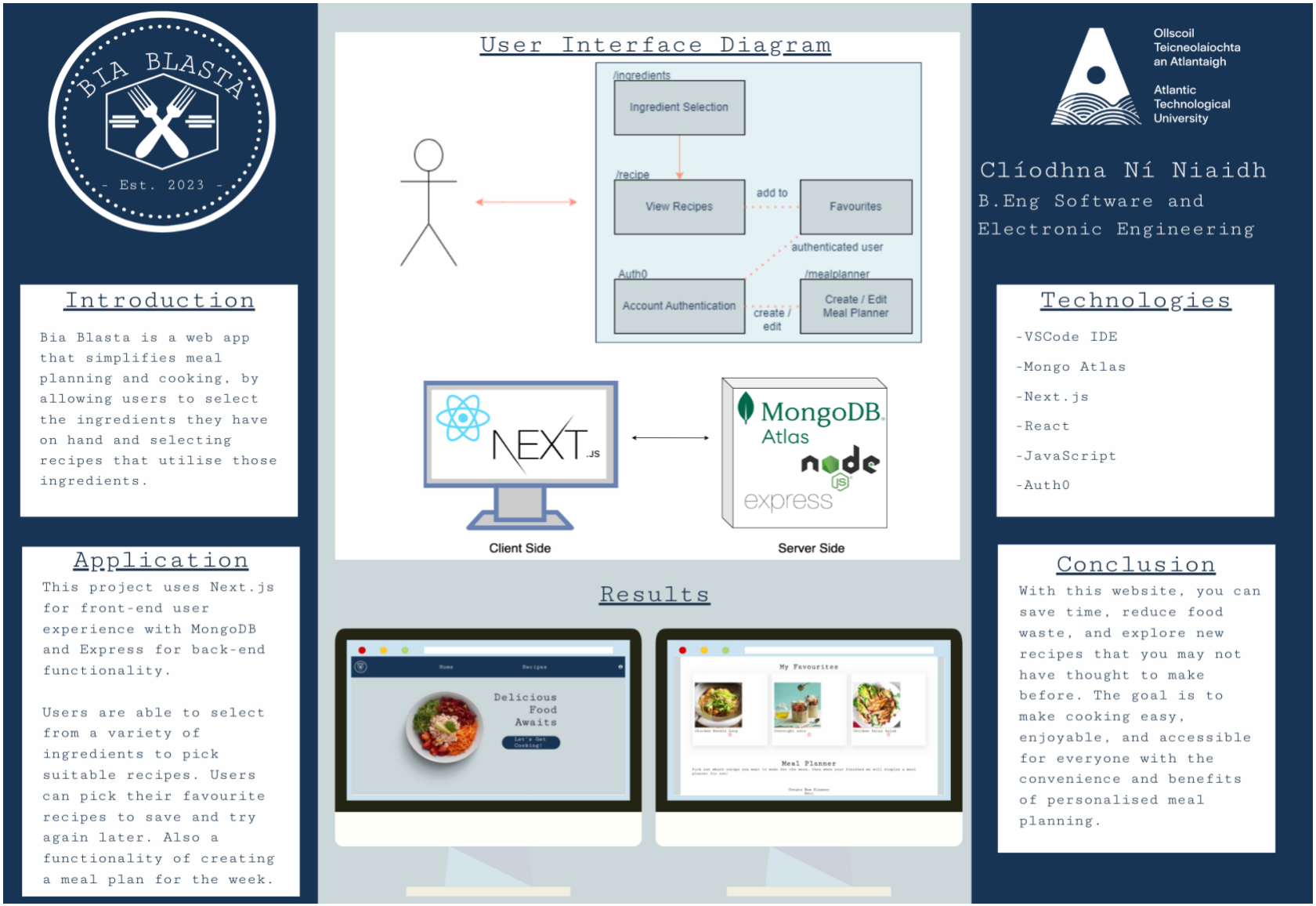
The project is built on a MERN Stack, which utilises MongoDB, Express and Node.js for server-side implementation, and React/Next.js for the client-side. Auth0 was utilised to provide authentication services of users. These software tools are used throughout the project to ensure smooth functioning. This project implements the skills and knowledge learned throughout the 4 years of this course.

Key feature of the project includes a robust search function which displays customised recipes to the user based on the specific ingredients selected. Another includes the ability to favourite recipes to the users personal account and create a meal planner for the week from all recipes available in the database.

When the user accesses the selection page, they are presented with a list of ingredients where they can select what they have available. Once the user makes their selection the information is saved as a query parameter and is submitted to the backend server. The server filters through the database and identifies all relevant recipes that contain those ingredients. The results are return to the frontend and displayed to the user.

The frontend of this project was developed using React framework with Next.js, while the backend uses Express and Node.js. The server is hosted locally while the database is stored on Mongo Atlas.

2 Poster


BIA BLASTA
- Est. 2023 -
Introduction
Bia Blasta is a web app that simplifies meal planning and cooking, by allowing users to select the ingredients they have on hand and selecting recipes that utilise those ingredients.
Application
This project uses Next.js for front-end user experience with MongoDB and Express for back-end functionality.
Users are able to select from a variety of ingredients to pick suitable recipes. Users can pick their favourite recipes to save and try again later. Also a functionality of creating a meal plan for the week.
User Interface Diagram
/ingredients
Ingredient Selection
/recipe
View Recipes
add to
Favourites
authenticated user
Auth0
Account Authentication
/mealplanner
create / edit
Create / Edit Meal Planner
NEXT.JS
MongoDB. Atlas
node
express
Client Side
Server Side
Results
Delicious Food Awaits
My Favourites
Meal Planner
Ollscoil Teicneolaíochta an Atlantaigh
Atlantic Technological University
Clíodhna Ní Niaidh
B.Eng Software and Electronic Engineering
Technologies
-VSCode IDE
-Mongo Atlas
-Next.js
-React
-JavaScript
-Auth0
Conclusion
With this website, you can save time, reduce food waste, and explore new recipes that you may not have thought to make before. The goal is to make cooking easy, enjoyable, and accessible for everyone with the convenience and benefits of personalised meal planning.

3 Introduction

The goal of this project was to develop a webapp that allows users to use up food they have laying around their kitchen, resulting in less food waste. The average Irish household puts €60 worth of food to waste each month, which equivalents to €700 every year[1]. Food waste generates up to 10% of global greenhouse gas emissions a year, with households responsible for generating 61% of food waste in the food industry[2].

In developing the project, promoting sustainable cooking practices also took an insight. By encouraging users to plan their meals for the week ahead, and use the meal planner function on the web app, this will help reduce the amount of food that goes to landfill, and contribute to greenhouse gas emissions.

With this project, households would be able to significantly reduce their food waste, which will improve the greenhouse gas emissions, promote sustainable cooking practices and help save money. By using up food items and minimising waste, indiviviuals can contribute to the larger goal of building a sustainable future.

4 Technology

In planning this project, it was essential to pick the correct technologies to work together, to ensure a smooth workflow.

4.1 React

React is a JavaScript library which is used for building user-interfaces (UI) for web applications[3]. React uses component-based architecture which mean it has reusable pieces of code called components. These components represent the UI and can be combined with other components to make complex UI's. React also allows developers to use JSX which allows HTML-like code to be written within JavaScript.

4.2 Next.js

Next.js is an open-source web development framework which uses React-based web application for server-side rendering. [4] A feature of Next.js is its server-side rendering. This is when the HTML code for a page is rendered on the server rather than the browser, which allows for a faster load-page and better search engine optimization (SEO).

4.3 Express.js

Express.js is a Node.js web application framework which provides a wide range of features for web development. Express is used to create RESTful APIs with Node.js and handle routing and respond to request from the client.[5]

4.4 Node.js

Node.js is an asynchronous JavaScript runtime, and is designed to build scalable network connections.[6] Node.js lets developers use JavaScript to write server-side code. Using Node.js in my project was a great choice as it allowed me to use JavaScript in both backend and frontend code.

4.5 Mongoose

Mongoose is a MongoDB object model for Node.js. It provides a schema-based solution to model application data. With this I was able to easily define and validate layouts of objects being stored in the database[7]

4.6 MongoDB / Mongo Atlas

MongoDB is a document-based database that uses No-SQL database programming. MongoDB is a popular choice when it comes to modern programming.[8]

4.7 Amazon S3

Amazon S3 is a storage service provided by Amazon Web Services (AWS). S3 is used to store and retrieve data. S3 is used in this project to store the images used on the web app, such as the logo and the images of the recipes.[9]

4.8 Auth 0

Auth 0 is a cloud-based identity authentication application for developers. The application allows developers to add security features to their applications, such as single page logins, password-less logins and multi-factor authentication.[10]

5 Project Architecture

The architecture diagram (Fig 5.1) represents the connection between the client-side and the server-side.

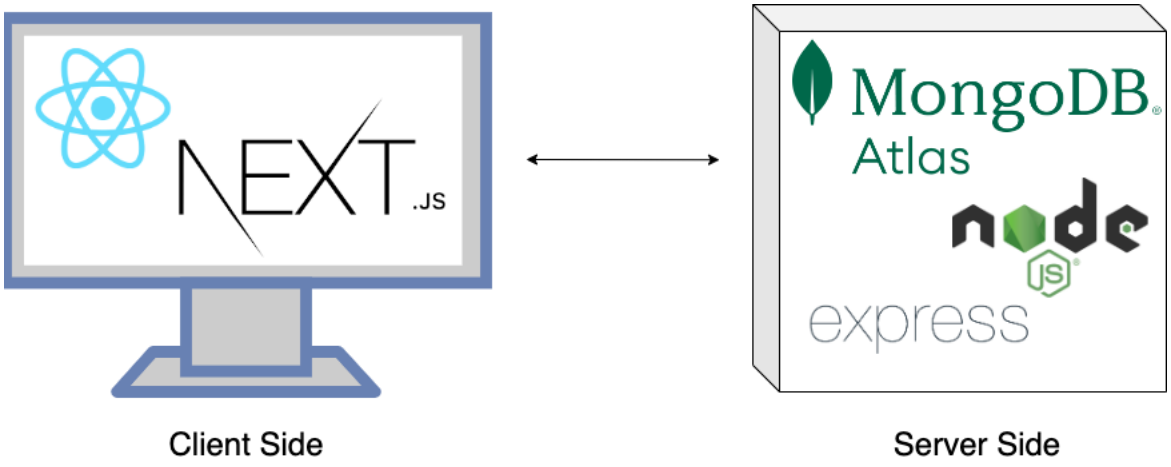


Figure 5.1 Architecture Diagram

The user diagram (Fig 5.2) represents how the user interacts with the webapp.

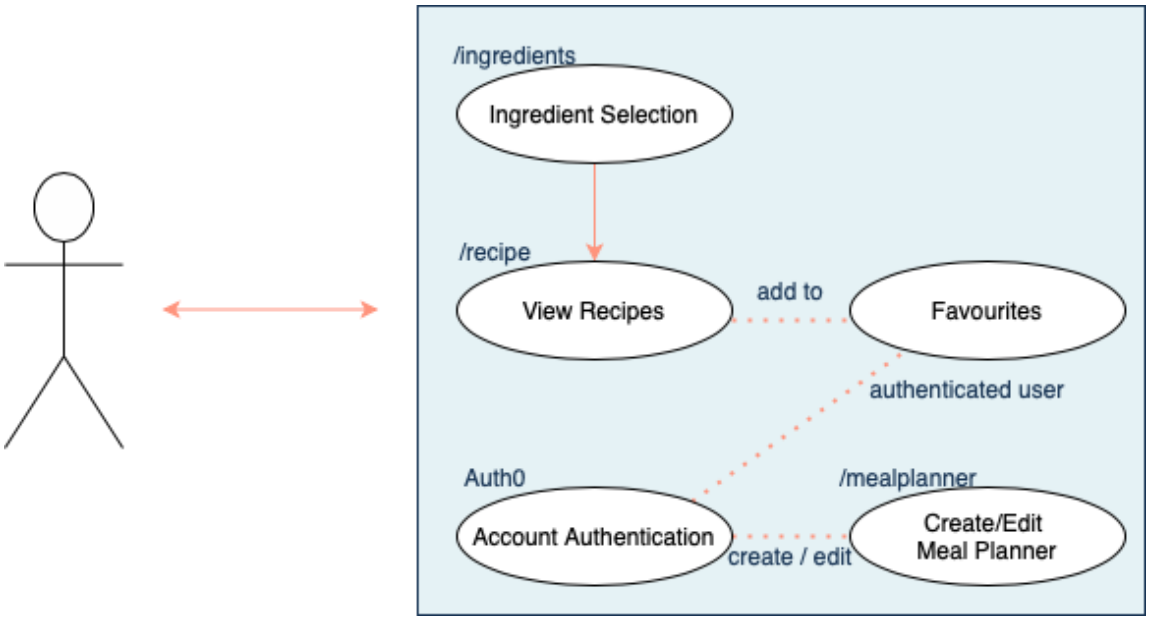


Figure 5.2 User Diagram

6 Project Plan

For project management, Microsoft Projects was used for week by week planning of the project. The timeline was updated each week to reflect on the work that was ongoing with the project. When creating a new task in the timeline a predicted date was added, in which you estimated when the work was to get done. The red lines in the timeline represent where the task went over the predicted time.

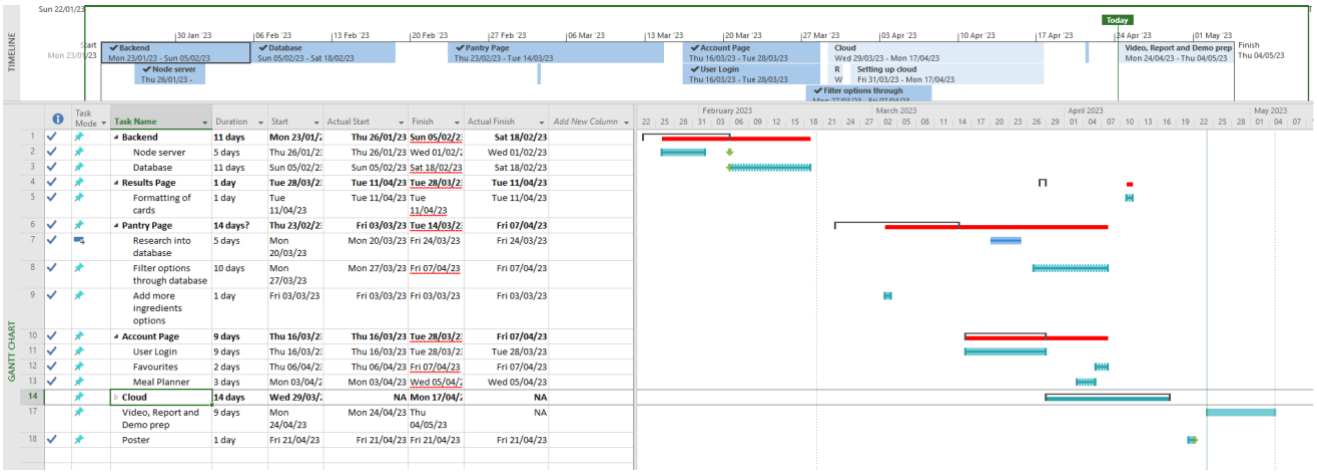


Figure 6.1 Timeline

7 Project Code

This section will be divided into two sections where the code for frontend and backend will be broken down and explained.

7.1 Frontend

The frontend code was developed using the knowledge and technologies acquired during the Full Stack module of this course and online tutorials which are referenced below.

7.1.1 handleChange and handleSubmit function

This section of web application is where the users select what ingredients they have. The ingredient selection is a form where the user selects the ingredients from the checkboxes, the form is submitted to the backend server. There are two primary functions that are responsible for creating the form.

The first function is the *handleChange()* function. This function checks whether the checkbox value is selected or unselected. There are two values *value* and *checked, value* represents the checkbox value, which is a string, while *checked* represents the current state of the checkbox, which is a Boolean value.

The if-else statement verifies the value the *checked* variable. If the checkbox is selected, the spread operator '…' is used to create a new array of the value *selectedIngredients,* with the value that was selected. The spread operator '…' allows to quickly copy all or part of an existing array into another array.[11] The new array is then set using the *setSelectedIngredients* function, this adds the new ingredients to the list of selected ingredients. The development of the handleChange code was done following this tutorial.[12]

```
const [selectedIngredients, setSelectedIngredients] = useState([]);
const router = useRouter();

const handleChange = (event) => {
  const { value, checked } = event.target;
  if (checked) {
    setSelectedIngredients([...selectedIngredients, value]);
  } else {
    setSelectedIngredients(
      selectedIngredients.filter((ingredient) => ingredient !== value)
    );
  }
};
```

Figure 7.1.1.1 handleChange Function

The second function is the *handleSubmit()* function, this submits the forms data. The *preventDefault()* is a function that prevents the form from submitting itself, this allows the *handleSubmit()* function to do the submission once the submit button is pressed. The ingredients that were selected on the form get separated by a comma using the *join()* method. The router object redirects the user to the recipe results page where they can view the customised recipe selection based on the ingredients. The ingredient selection is passed using a query parameter, *encodeURIComponent* encodes the characters to safely pass the string as a URL. The handleSubmit function and encodeURIComponent were developed from the Next.js documentation and GeeksForGeeks webpage. [13][14]

```
const handleSubmit = async (event) => {
  event.preventDefault();

  // Convert the selectedIngredients array into a comma-separated string
  const ingredientsString = selectedIngredients.join(',');

  // Redirect to the /recipes/search page with the ingredients as a query parameter
  router.push(`/recipe?ingredients=${encodeURIComponent(ingredientsString)}`);
};
```

Figure 7.1.1.2 handleSubmit Function

7.1.2 useEffect Hook

In several pages of the web application, Reacts *useEffect* hook is used to fetch results from an API endpoint. The hook allows for side effects in components, such as fetching data, directly updating DOM and the use of timers. [15] The *fetchRecipes()* function is called by *useEffect*, and uses the ingredients state variable to build a URL query parameter using *URLSearchParam()* constructer.[16] It then fetches a response from the database, which is on *localhost:3001*, to get the recipes with the query parameter. The if statement makes sure that the *fetchRecipes()* function is only returned if the ingredient variable is true.

The second argument in the *useEffect* hook is an array that holds the dependency ingredients. This tells the hook to only re-run the code if the ingredients variable changes. Doing this ensures that the recipe data changes when a user adds new ingredients.

```
useEffect(() => {
  const fetchRecipes = async () => {
    const ingredientsParams = new URLSearchParams({
      ingredients: ingredients,
    }).toString();

    const response = await fetch(
      `http://localhost:3001/recipes/search?${ingredientsParams}`
    );
    const data = await response.json();
    setRecipes(data);
  };

  if (ingredients) {
    fetchRecipes();
  }
}, [ingredients]);
```

Figure 7.1.2.1 useEffect Hook

7.1.3 handleFavouriteClick function

The *handleFavouriteClick()* function allows for a user to add a recipe to their account. When the function is called it sends a GET request to *localhost:3001/favourite/${recipes.id}*. The endpoint is responsible for checking if the recipe is already added to the favourite collection in the database. If the recipe is already favourited, the user will get a pop-up alert message. This functionality prevents duplicates in the collection.

```
async function handleFavouriteClick() {
  // Save the recipe to favorites
  try {
    const response = await fetch(
      `http://localhost:3001/favourite/${recipes.id}`
    );

    if (!response.ok) {
      throw new Error("Failed to check if recipe is favourited");
    }

    //checks for duplicates in database
    const favouritesData = await response.json();
    const recipeId = recipes.id;

    const isRecipeInFavourites = favouritesData.some((favourite) => {
      return favourite.recipeId.id === recipeId;
    });
```

Figure 7.1.3.1 handleFavouriteClick function, duplicate check

If the recipe is not saved to the collection, then a POST request is sent to *localhost:3001/favourite*, to add the recipe to the collection. The request body sent contains the *recipeID* in JSON format. If the request is successful, the data is set in the favourites variable using *setFavorites()* function and *isFavourited* to true. The *isFavourited* variable is used to display a plus icon to the user to add a recipe to their account. If the variable is set to true, a message is displayed to the user that the recipe was successfully saved. The code for this function was developed from this tutorial. [17]

```
if (isRecipeInFavourites) {
  console.log("This recipe is already in favourites");
  alert("This recipe is already in favourites");
} else {
  console.log("This recipe is not yet in favourites");
  const postResponse = await fetch("http://localhost:3001/favourite", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ recipeId: recipes.id }),
  });

  if (!postResponse.ok) {
    throw new Error("Failed to save recipe to favourites");
  }

  const favourite = await postResponse.json();
  setFavourites(favourite);
  setIsFavourited(true);
}
} catch (error) {
  console.error(error);
}
```

Figure 7.1.3.2 handleFavouriteClick function, posting to server

7.2 Backend

The backend code was developed using the knowledge and technologies gained from the Full Stack module and Cloud Development module of this course. Research and development of the API endpoints was gotten from the Academind YouTube channel. [18]

7.2.1 GET Endpoints

In the recipe route, the API endpoints used to interact with the database. The GET routes, retrieve the recipes from the database using various methods. There is get all recipes, get by id, and get by query parameter. The *'/search'* endpoint receives the query parameter ingredients from the frontend. When the request is made to this route it extracts the ingredient query using '*req.query*'. After that the ingredients are split into an array, so each individual ingredient can be filtered. Using the mongoose Recipe schema, the route finds all recipes that contain the ingredients in the array, by using the $in operator. This searches for any matching name, from the ingredient schema, fields in the ingredient's subdocuments. If any matching recipes are found, they are added to an array called recipes and sent to the frontend via JSON object.

```
router.get('/search', async (req, res) => {
  try {
    const { ingredients } = req.query;

    // Convert the ingredients string into an array
    const ingredientsArray = ingredients.split(',');

    // Find recipes containing all specified ingredients
    const recipes = await Recipe.find({
      'ingredients.name': { $in: ingredientsArray },
    });

    res.status(200).json(recipes);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Server error" });
  }
});
```

Figure 7.2.1.1 GET query Endpoint

The *'/id'* route is responsible for fetching recipes based on IDs provided in the URL. The recipe model uses the *findOne()* method which uses the request parameter to get the ID from the URL. Once the query is successful the server responds with a JSON object with the recipe allocated to the id.

```
//get by id
router.get("/:id", async (req, res) => {
  try {
    //params cause finding by url
    const recipes = await Recipe.findOne({ id: req.params.id });
    //returns data in json format
    res.json(recipes);
  } catch (err) {
    res.send("Error " + err);
  }
});
```

Figure 7.2.1.2 GET by ID Endpoint

7.2.2 POST Endpoint

The POST endpoint handles any post requests to the '/' URL. In this case the post request is creating a new meal planner. When the POST request is made the route is expecting two properties, *day of the week*, which is a string, and the *recipeId*, which contains the recipes id that is being added to the meal planner. The route first checks if the recipe ID exists within the database, using *findById().* Once the route finds the recipe, a new *MealPlan* document is created with the inputs receive from the frontend (*day of week and recipeId*) and saved to the database.

```
// create meal plan
router.post("/", async (req, res) => {
  try {      You, now • Uncommitted changes
    const { dayOfWeek, recipeId } = req.body;

    // check if the recipes exist
    const recipes = await Recipe.findById(recipeId);
    if (!recipes) {
      return res.status(404).send("Recipe not found");
    }

    const mealPlan = new MealPlan({
      dayOfWeek,
      recipeId,
    });

    await mealPlan.save();
    res.json(mealPlan);
  } catch (err) {
    res.status(500).send(err);
    console.log(err);
  }
});
```

Figure 7.2.2.1 POST Endpoint

7.2.3 PUT Endpoint

A PUT request updates an already existing document in the database. The put request here allows the user to update an existing meal planner in the database. The route waits for the id from the URL that identifies the meal planner to be updated. The route finds the meal planner by id using *findById().* Once gotten the *dayofweek* and *recipeId* fields are updated and

overwritten by the new values specified by the user. Once saved the document is updated and saved to the database.

```
// define a route to handle updates
router.put("/:id", async (req, res) => {
  const id = req.params.id;

  // find the existing meal plan by id
  let mealPlan = await MealPlan.findById(id);

  if (!mealPlan) {
    return res.status(404).send({ error: "Meal plan not found" });
  }

  // update the fields you want to change
  mealPlan.dayOfWeek = req.body.dayOfWeek;
  mealPlan.recipeId = req.body.recipeId;

  // save the updated meal plan
  try {
    mealPlan = await mealPlan.save();
    res.send(mealPlan);
  } catch (err) {
    console.log(err);
    res.status(500).send({ error: "Failed to update meal plan" });
  }
});
```

Figure 7.2.3.1 PUT Endpoint

7.2.4 DELETE Endpoint

This DELETE endpoint is responsible for removing a document from the database. Here the DELETE request is removing a recipe from the favourites. The request finds the favourite by ID, then proceeds to delete it from the database.

```
router.delete("/:id", async (req, res) => {
  try {
    const favouriteId = req.params.id;
    const deletedFavourite = await Favourite.findOneAndDelete(favouriteId);
    if (!deletedFavourite) {
      return res.status(404).send("Favourite not found");
    }
    res.json({ message: "Favourite deleted successfully" });
  } catch (err) {
    res.status(500).send(err);
  }
});
```

Figure 7.2.4.1 DELETE Endpoint

7.3 Mongo Atlas

MongoDB is a No-SQL database program which uses JSON documents with schemas. Mongo Atlas is a cloud-based database server which is fully managed by MongoDB.[19] The Mongo Atlas gets connected to the backend server via a URI, which contains the username and password associated with the Mongo Atlas account. This URI is stored in the .env file of the backend server, as it contains sensitive information.

7.4 Auth0

Auth0 is used as a third-party authentication of the project. The Auth0 code redirects the user to a single-page login where they can sign-up or log-in. An account is set up on the Auth0 webpage where an application is set up for what type of authentication is required for the project. Research and development of implementing the Auth0 application was done from this source. [20]

8 Ethics and Sustainability

When it comes to the ethics and sustainability of food waste and consumption, they are both equally important. The distribution of food has a significant impact on the environment we live in.

When it comes to the ethics around animal welfare most people take into consideration the treatment of animals in the food industry. An example of this would be factory farming, where animals are raised in stressful conditions. There is also a large concern for the treatment of antibiotics and chemicals in animal feed, which negatively impacts human health. As a result of this people are choosing to consume less meat and opt for a vegetarian meal more often during their week. [21]

Sustainability is also an important consideration when it comes to food. Food production is a significant contributor to greenhouse gas emissions. As a result of this, there is a high interest in sustainable food systems that prioritises environmental stewardship. This emphasis reducing waste, protecting the environment and the wellbeing of citizens. [22]

9 Conclusion

In conclusion, Bia Blasta is a MERN Stack web application that aims to reduce food waste by providing users with recipe recommendations with food they have available. Throughout the project technologies were used to create the frontend and backend development. These technologies include Next.js, React, Node.js, Express, MongoDB and Auth0. This project helps promote sustainable cooking practices by allowing the user to create meal planners for the week, which contributes to a reduction in food waste and greenhouse gas emissions.

While the main functionality of this project is completed, there is still room for further development. Some of these features could include adjusting recipes to a user's dietary requirements and adding calorie tracking to the meal planner. These features were unable to be implemented to the project due to time management.

Overall, the project was a great learning experience with gaining experience using technologies such as full-stack development, improving time management and project management skills, and holding accountability to timelines set.

The completed code for both frontend and backend can be accessed from the GitHub links below.

Frontend - https://github.com/cliodhnaniniaidh1/BiaBlasta

Backend - https://github.com/cliodhnaniniaidh1/BiaBlastaServer

10 Frontend Results

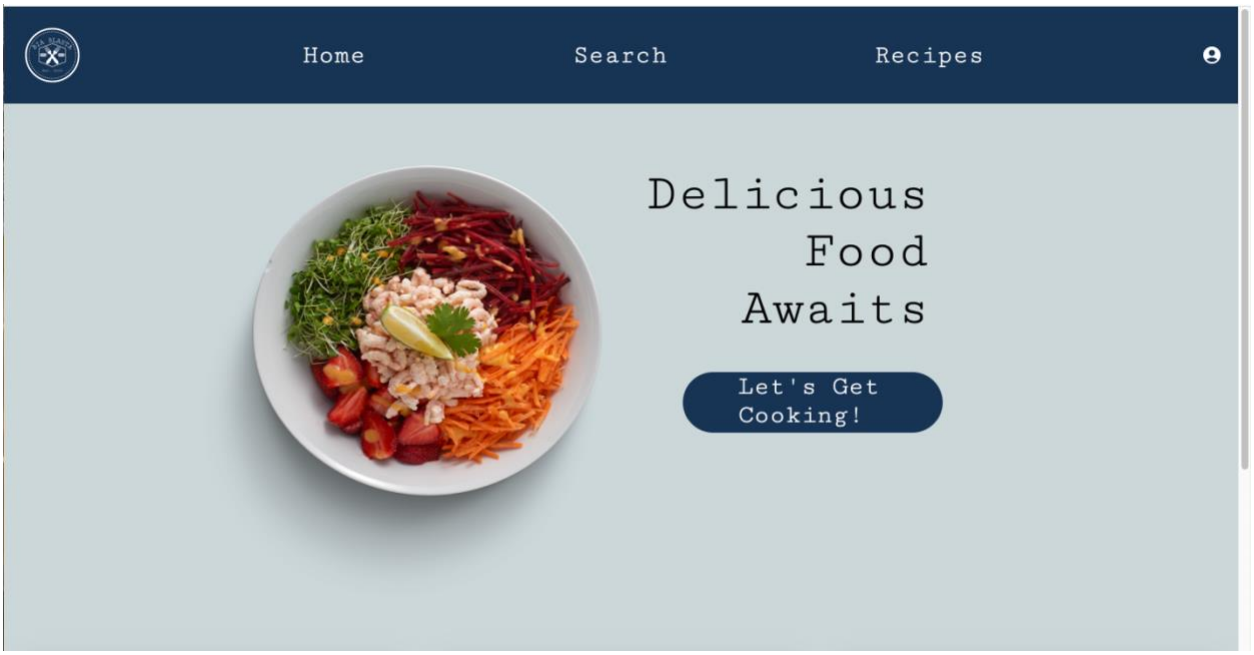


Figure 10.1 Home Page

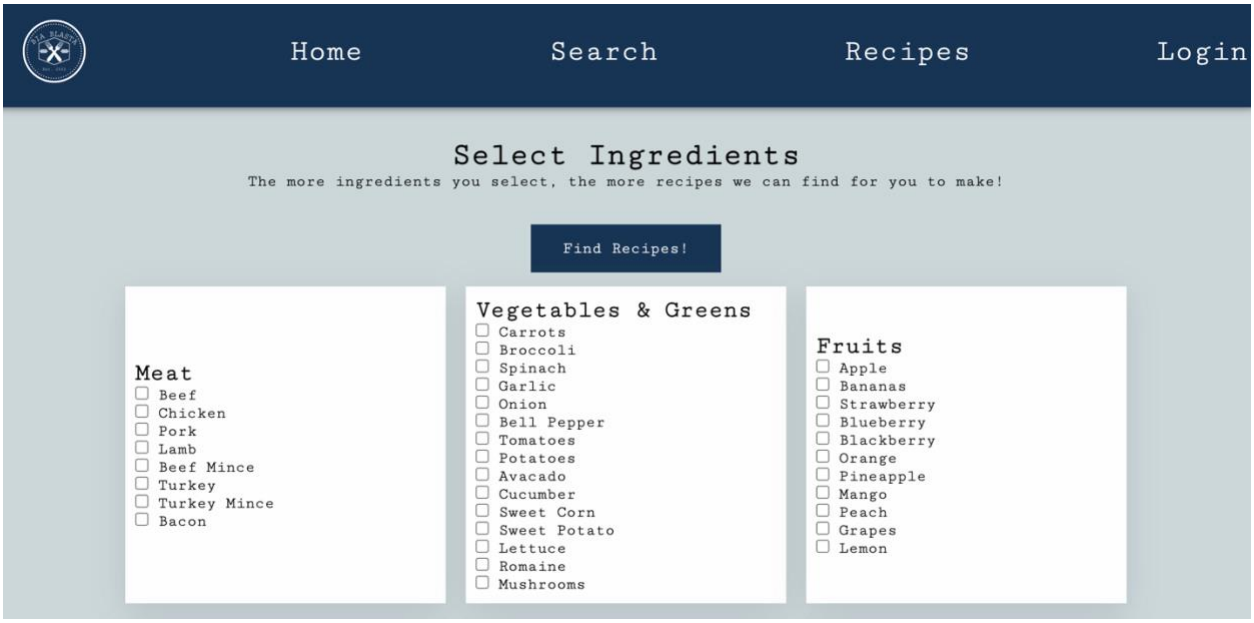


Figure 10.2 Ingredient Selection

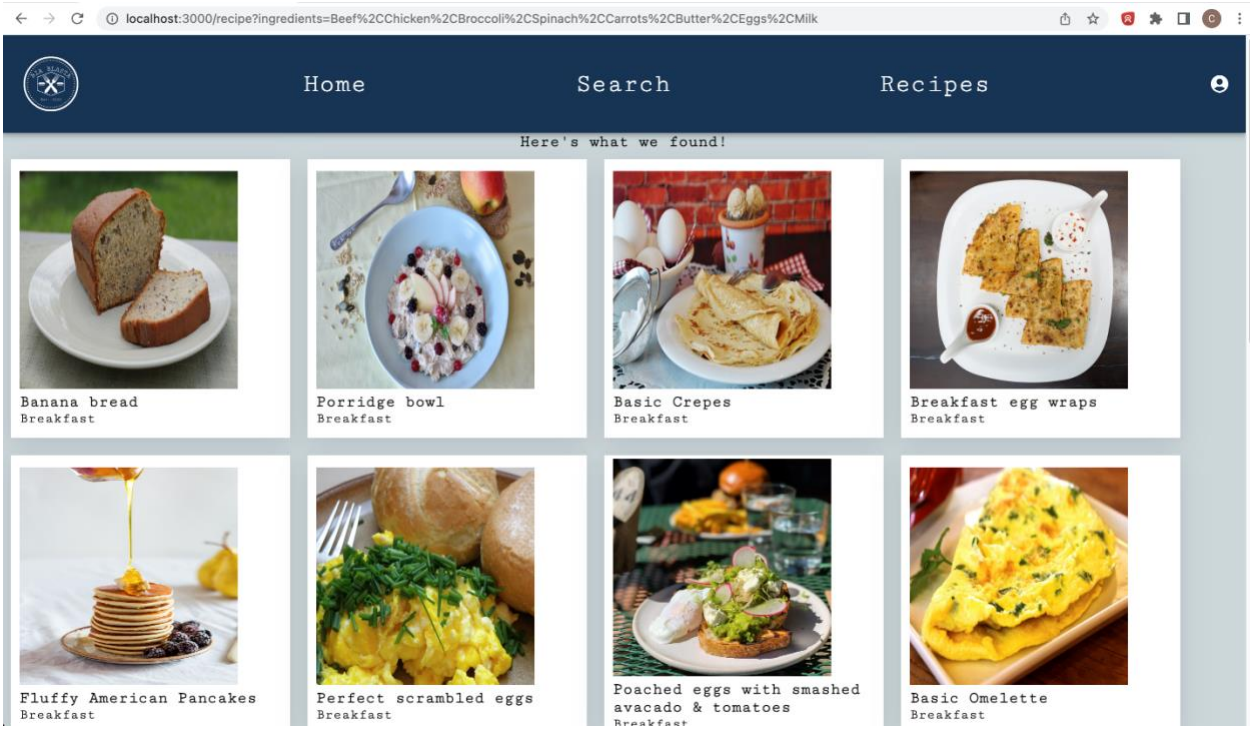


Figure 10.3 Query Results

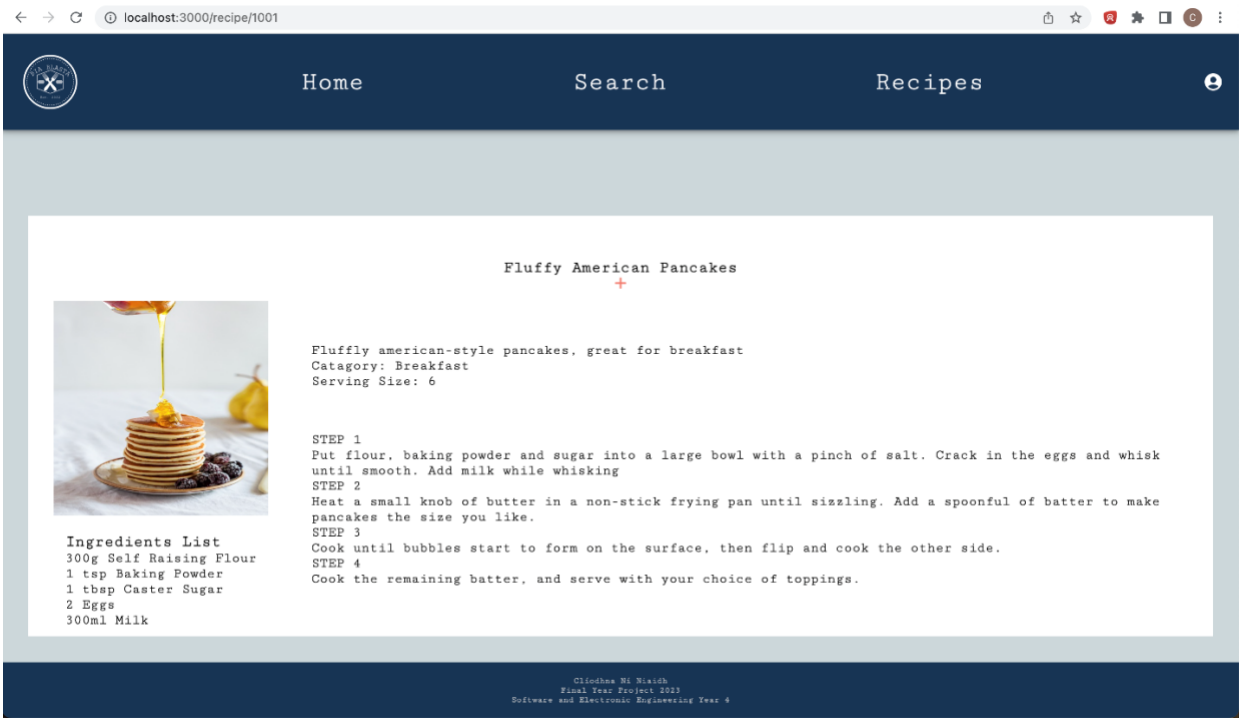


Figure 10.4 Recipe Page

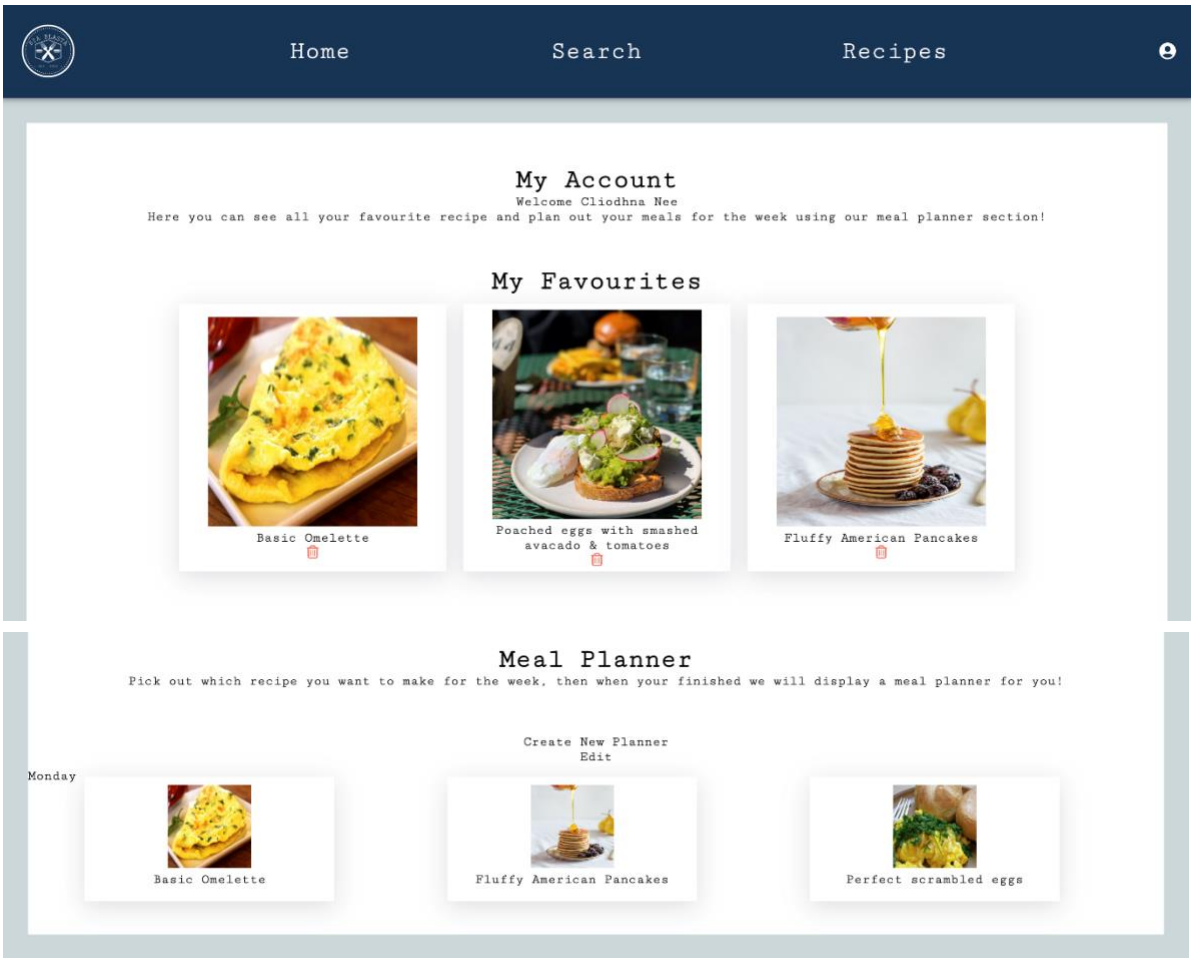


Figure 10.5 Account Page

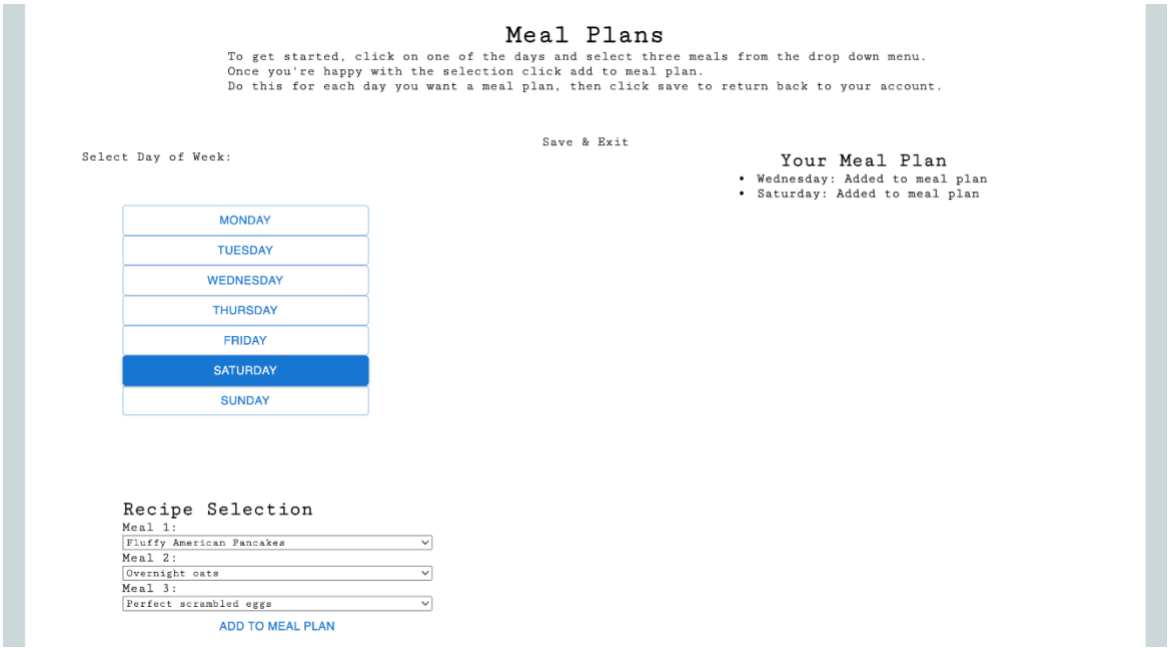


Figure 10.6 Meal Planner

11 References

[1] E. P. Agency, 'Food Waste Statistics'. https://www.epa.ie/our-services/monitoring--assessment/waste/national-waste-statistics/food/ (accessed Apr. 29, 2023).

[2] editor, 'Food Waste and Sustainable Development', *Sustainability Knowledge Group*, Mar. 31, 2021. http://3.69.203.30/food-waste-and-sustainable-development/ (accessed Apr. 29, 2023).

[3] 'Getting Started – React'. https://legacy.reactjs.org/docs/getting-started.html (accessed Apr. 29, 2023).

[4] 'Next.js', *Wikipedia*. Apr. 09, 2023. Accessed: Apr. 29, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Next.js&oldid=1148959713

[5] 'Express.js', *Wikipedia*. Sep. 22, 2022. Accessed: Apr. 29, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Express.js&oldid=1111721738

[6] 'About', *Node.js*. https://nodejs.org/en/about (accessed Apr. 29, 2023).

[7] 'Mongoose ODM v7.1.0'. https://mongoosejs.com/ (accessed Apr. 29, 2023).

[8] 'Why Use MongoDB And When To Use It?', *MongoDB*. https://www.mongodb.com/why-use-mongodb (accessed Apr. 29, 2023).

[9] 'Cloud Object Storage – Amazon S3 – Amazon Web Services'. https://aws.amazon.com/s3/ (accessed Apr. 29, 2023).

[10] 'Auth0 - Wiki | Golden'. https://golden.com/wiki/Auth0-4NNBR53 (accessed May 01, 2023).

[11] 'React ES6 Spread Operator'. https://www.w3schools.com/react/react_es6_spread.asp (accessed Apr. 30, 2023).

[12] 'How to use handleChange() function in react component?', *GeeksforGeeks*, Jan. 13, 2021. https://www.geeksforgeeks.org/how-to-use-handlechange-function-in-react-component/ (accessed May 03, 2023).

[13] 'Building Forms with Next.js | Next.js'. https://nextjs.org/docs/guides/building-forms (accessed May 03, 2023).

[14] 'JavaScript encodeURI(), decodeURI() and its components Functions', *GeeksforGeeks*, Oct. 15, 2018. https://www.geeksforgeeks.org/javascript-encodeuri-decodeuri-and-its-components-functions/ (accessed May 03, 2023).

[15] 'React useEffect'. https://www.w3schools.com/react/react_useeffect.asp (accessed Apr. 30, 2023).

[16] *Get URL Parameters in JavaScript | URLSearchParams*, (Feb. 10, 2022). Accessed: May 01, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=CZP1iQFQjEY

[17] 'The Complete Guide to React User Authentication with Auth0', *Auth0 - Blog*. https://auth0.com/blog/complete-guide-to-react-user-authentication/ (accessed May 01, 2023).

[18] '(2) Node.js + Express - Tutorial - What is Express? And why should we use it? - YouTube'. https://www.youtube.com/watch?v=45dAt9Gz8rE&t=1s (accessed May 02, 2023).

[19] 'MongoDB Atlas Database | Multi-Cloud Database Service', *MongoDB*. https://www.mongodb.com/atlas/database (accessed May 02, 2023).

[20] 'Simple Next.js User Login Authentication | 5 Steps in 5 Minutes! | Auth0 - YouTube'. https://www.youtube.com/watch?v=jgKRnhJBfpQ (accessed May 02, 2023).

[21] 'What is food ethics? – Food Ethics Council'. https://www.foodethicscouncil.org/insights/what-is-food-ethics/# (accessed May 02, 2023).

[22] E. P. Agency, 'The Office of Environmental Sustainability'. https://www.epa.ie/who-we-are/roles--responsibilities/organisational-structure/the-office-of-environmental-sustainability/ (accessed May 02, 2023).