Group 3 project documentation

08.06.2024

Project team:

Abbie Kennard-Jones

Elisa McGarry

Emma McGuire

Sahra Abdirahman

Shafia Kashim

Victoria Plum

Introduction

Aim

Our project aims to create an innovative application that addresses a specific problem within a chosen domain. Working collaboratively, we will deliver a solution that demonstrates technical skills, creativity, problem solving and user-centric design principles.

Objectives

1. **Problem identification and analysis:**

Identify real world problems or inefficiencies within a chosen field that could be effectively addressed through the development of an application.

2. **User-centric design and functionality:**

Utilise user-centric design principles to ensure that the application is intuitive, user-friendly and meets the preferences of its target audience.

3. **Technical proficiency:**

Demonstrate technical proficiency in the app development, utilising appropriate tools to implement the desired features and functionality.

4. **Project management and collaboration:**

Establish effective project management processes, including task allocation, progress tracking, and regular communication.

5. **Quality assurance and testing:**

- Implement thorough quality assurance measures throughout the development lifecycle to ensure reliability of the application.
- Perform user testing and incorporate feedback to enhance the application according to user insights.

Roadmap of the report

The Thirty Plant Kitchen app was developed to address multiple user needs by promoting sustainable eating habits, simplifying meal planning, and encouraging healthier diets. The app achieves these goals by helping users identify seasonal ingredients and providing recipes that incorporate 30 different plants per week.

Key features

- **Sustainable meal planning:** Users can search for recipes based on seasonal and locally-sourced ingredients, reducing their carbon footprint.
- **Recipe generator:** The app includes a recipe generator that provides meal ideas based on selected seasonal produce.
- **Health tracking:** A plant counter tracks users' intake of different plants, encouraging a diverse and balanced diet.
- **User account management:** Features for sign-up, login, and saved recipes create personalised user experiences.

Technical specifications

- **Backend:** Developed using Flask with MySQL for data management and Bcrypt for user authentication.
- **Frontend:** Built with React and Redux, employing React Router for navigation and Material-UI for styling.
- **APIs:** Utilised an internal database for seasonal plants and the Edamam recipe API for recipe searches.
- **Testing:** Conducted component, integration and end-to-end testing using Jest and React Testing Library to ensure functionality and reliability.

Development approach: The project was managed using an agile methodology, facilitated by Jira for task management and Slack for communication. The team held regular meetings and used pair programming to address challenges efficiently. Wireframes were created using MoqUps, and Canva was used for establishing a consistent design aesthetic.

Outcomes

- **Enhanced skills:** The project consolidated technical skills in app development, database management, API integration, and UI/UX design.
- **Teamwork:** Strengthened teamwork, communication and project planning skills.
- **User-centric design:** Iterative design and testing ensures the app meets user needs and provides an intuitive experience.

Challenges and solutions: The primary challenge was the lack of reliable seasonal plants APIs. The team addressed this by creating a custom SQL database to store seasonal plant data, complemented by an external recipe API for diverse recipe options.

Future enhancements

- **Automated plant counter integration:** Automatically update the plant counter based on data from the recipes cooked by users.
- **Accessibility testing:** Ensure the app is accessible to users with disabilities.
- **Cross-platform testing:** Enhance compatibility across different devices and browsers.

Background

Our project delivers a unique recipe-finding service aimed at promoting the consumption of seasonal plants. Thirty Plant Kitchen is an innovative app designed to help users identify which ingredients are in season and plan meals around sustainable gardening. The app encourages users to eat 30 different plants per week through delicious recipes.

Why we designed this app

We developed the app to address several issues that people face in their daily lives:

1. **Reducing environmental impact:** Our food choices carry a significant carbon footprint and eating in a way that minimises this is crucial for sustainability.
2. **Finding meal inspiration:** People struggle with finding new, interesting meal ideas.
3. **Promoting healthier eating habits:** Making healthier food choices that support long-term health and well-being is a priority for many individuals.

What makes Thirty Plant Kitchen stand out

Our app excels in many areas that set it apart. Firstly, our app promotes environmental sustainability by encouraging users to select recipes that use seasonal, UK-grown plants, reducing their carbon footprint. For example, consuming locally produced strawberries in season has a much lower carbon footprint compared to those imported from South Africa. By focusing on seasonal produce, our app helps users make sustainable food choices [1].

Another stand out feature is the ease of meal planning. Deciding what to eat can be daunting, but our app simplifies this process by including a recipe generator that suggests delicious recipes, using seasonal produce. By connecting with a recipe API, Thirty Plant Kitchen provides users with fresh, interesting meal ideas without the need for extensive searches through cookbooks or online resources. This makes meal planning simpler and more enjoyable, encouraging users to explore new culinary experiences.

Health and wellbeing are central to our app's design. Thirty Plant Kitchen promotes the habit of eating 30 different plants a week, a practice that has been shown to positively affect gut bacteria and overall health. Our app offers recipes to help users achieve this goal and includes a counter to track plant intake progress, supporting a diverse and balanced diet. This feature makes it easier for users to maintain a healthy diet and encourages them to incorporate a variety of plants into their meals, promoting long-term health [2].

Specifications and design

Design and architecture

Our React application is structured with a clear separation of backend and frontend.

In the frontend 'src' folder we store everything for the main app user interface. This is organised into folders for assets, components, pages, styles, tests and Redux.

Redux is the main state management library and contains two slices:

- **customerSlice**: Manages user information such as customerId, firstName, emailAddress, password and isSignedIn status
- **counterSlice**: Manages the plant counter state.

The app uses react-router-dom for routing:

- **Router** uses BrowserRouter to handle client-side routing
- **Routes** defines routes for pages such as Home, SignUp, MyAccount, Recipes, About
- The **useNavigate** hook is used on app buttons to redirect the user internally.

Styling is handled with a combination of:

- **CSS modules** for component-specific styling
- **Styled-components** for writing CSS in Javascript
- **Material-UI (MUI)** for using pre-built React components with consistent styling.

We used React hooks and states to create a dynamic user experience. Specifically, useState hook is used to manage component states, such as tracking saved recipes and loading status, and the useSelector hook to access global state values from the Redux store, like the user's sign-in status and customer ID. This combination allows us to dynamically update the UI based on user actions and real-time data, ensuring a responsive and personalised experience. For example, in the Recipelist component, we dynamically display recipes and save states using these hooks, and conditionally render elements based on the user's authentication status and interactions with the app.

We use props to design components that are reusable and configurable, for example on our About Us page to pass data to TeamCard components for team bios.

The backend folder contains everything the app requires to integrate with APIs including db_utils.py for database connection, config.py for SQL credentials, API endpoints and HTTP requests, requirements.txt file containing the relevant backend packages and libraries. We used an external Edamam recipe API for dynamic recipe data. We built an SQL database to store seasonal plant information, customer data and saved recipes.

We use the Fetch API to make HTTP requests and get resources across the network asynchronously. Fetch requests support Cross-Origin Resource Sharing, allowing us to fetch resources from different origins (i.e. the SQL database and recipe API).

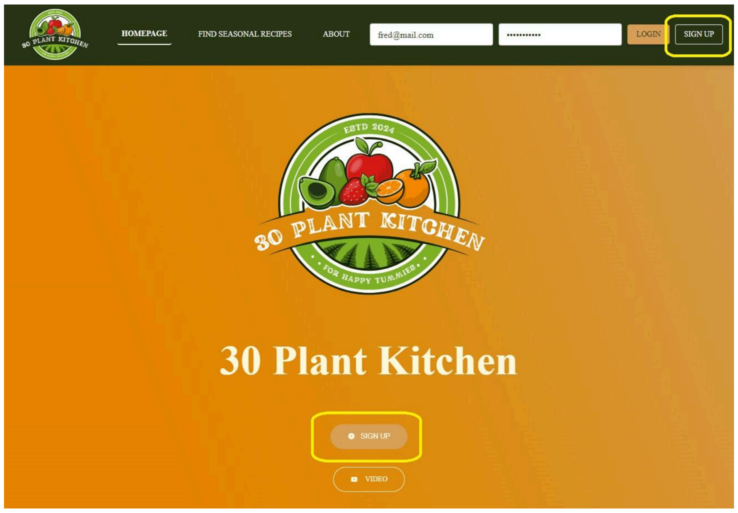
The app manages user authentication using the Bcrypt library on the backend to encrypt passwords, and Redux state to manage user authentication status.

The overall architecture of our app promotes modularity and maintainability.

Key features of the app

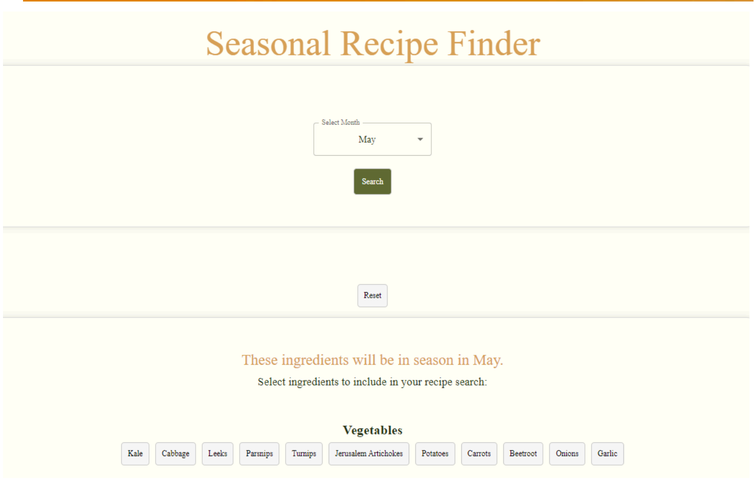
Sign up

- From the homepage or navbar, click 'Sign up'.
- On submitting the form, the user is directed to the seasonal recipes search page.



Search seasonal plants & recipes

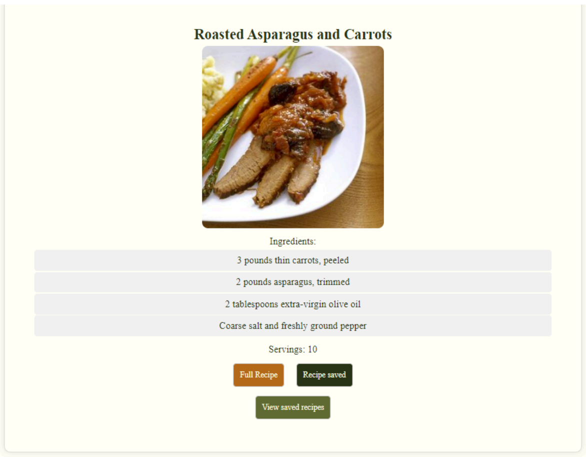
- Choose a month from the drop-down menu and search seasonal plants.
- Select which plants to include in the recipe search, then click 'Generate recipes'.



These ingredients will be in season in May.

Select ingredients to include in your recipe search:

- Click ‘Full recipe’ for the instructions.
- Click to save recipes to My account.
- If a user saves a recipe but is not signed in, they’re prompted to register and are redirected to the sign-up page.
- If a logged-in user saves a recipe, the button text changes to ‘Recipe saved’.
- An additional button is rendered to ‘View saved recipes’ in My account.
- A user cannot save the same recipe twice.


Roasted Asparagus and Carrots

Ingredients:

3 pounds thin carrots, peeled

2 pounds asparagus, trimmed

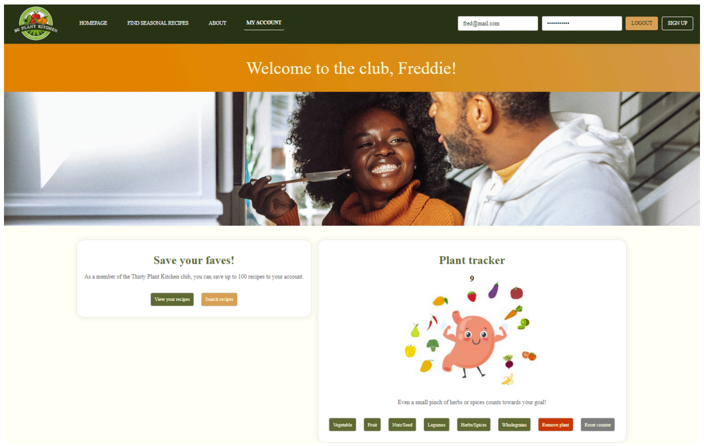
2 tablespoons extra-virgin olive oil

Coarse salt and freshly ground pepper

Servings: 10

My account

- On sign-in, a link to ‘My account’ appears in the navigation bar.
- My account displays a personal welcome message.
- User can view / hide saved recipes.
- Use the counter to track plants eaten. Each time the user adds a plant to the counter, a healthy eating fact is revealed.



Welcome to the club, Freddie!

Save your faves!

As a member of the Thirty Plant Kitchen club, you can save up to 100 recipes to your account.

Plant tracker

9

Even a small pinch of herbs or spices counts towards your goal!

Login / logout

- Enter an email address and password into login fields in the navbar.
- After signing up / logging in, the ‘Login’ button changes to ‘Logout’ in the navbar.
- When a user logs out, saved recipes and the counter are stored for the next login.

Implementation and execution

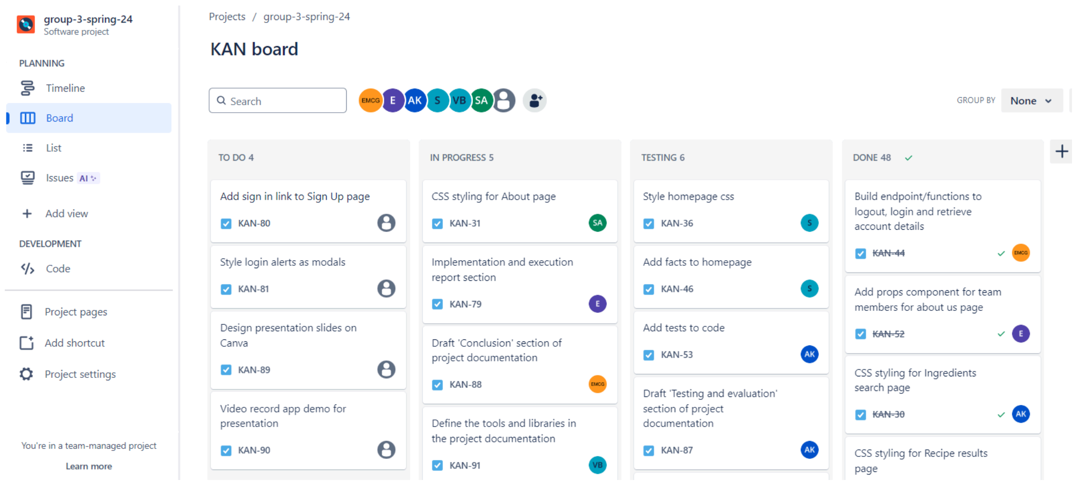
Development approach and team member roles

To kick off the project, our group came together to conduct a SWOT analysis, discussing strengths, weaknesses, opportunities and threats. During this meeting, we discussed interests, which played a pivotal role in selecting a project topic that resonated with our collective passions and expertise.

After deciding on a topic, we used MoqUps to create wireframes, outlining the desired layouts of the app screens. This visual representation established the foundation for our design process, ensuring clarity and alignment from the beginning. To maintain a consistent aesthetic across the app, we used Canva to define our brand and overall style. This attention to detail from the beginning ensured a cohesive user experience throughout.

To build on the insights gained from our SWOT analysis, we organised ourselves into frontend and backend groups, capitalising on each member's strengths and interests. Elisa, Emma and Victoria made up the backend group, while Abbie, Shafia and Sahra worked on the frontend of the application. Through collaborative discussions, project tasks were thoughtfully assigned, ensuring alignment with individual preferences and proficiencies.

Jira was used as our central project management tool, enabling us to break down tasks into actionable user stories. This approach streamlined progress tracking and allowed us to efficiently prioritise tasks, optimising our workflow and resource allocation.



KAN board

TO DO 4	IN PROGRESS 5	TESTING 6	DONE 48
Add sign in link to Sign Up page — KAN-80	CSS styling for About page — KAN-31 — SA	Style homepage css — KAN-36 — S	Build endpoint/functions to logout, login and retrieve account details — ~~KAN-44~~ — EMCG
Style login alerts as modals — KAN-81	Implementation and execution report section — KAN-79 — E	Add facts to homepage — KAN-46 — S	Add props component for team members for about us page — ~~KAN-52~~ — E
Design presentation slides on Canva — KAN-89	Draft 'Conclusion' section of project documentation — KAN-88 — EMCG	Add tests to code — KAN-53 — AK	CSS styling for Ingredients search page — ~~KAN-30~~ — AK
Video record app demo for presentation — KAN-90	Define the tools and libraries in the project documentation — KAN-91 — VB	Draft 'Testing and evaluation' section of project documentation — KAN-87 — AK	CSS styling for Recipe results page

Throughout the project, communication was crucial. We held weekly group meetings via Zoom to allow for real-time collaboration and provide progress updates. These meetings served as a platform to address any issues, often using pair programming to resolve them efficiently. Additionally, we utilised Slack to maintain communication between meetings.

Implementation process

In the planning phase of the project, our intention was to use an API to provide seasonal plant data, enabling users to make informed decisions about their cooking. However, we encountered a lack of reliable seasonal plant APIs, which presented a significant challenge to our project, prompting us to develop an alternative strategy. In response to this challenge, we developed our own database to store seasonal plants data, ensuring its integrity and flexibility to meet our app's requirements. Simultaneously, we integrated an external API for recipe searching, enriching our app's functionality with a range of culinary options. This hybrid approach of combining an internal database for seasonal plants data with an external API for recipe searching optimised our app's capabilities.

Agile development

Initially, we outlined a plan for the project's objectives. However, by employing an agile approach, supported by the use of Jira, we could seamlessly incorporate additional features such as a user log-in function, as needed. This flexibility enhanced the project's scope and adaptability, ensuring its alignment with evolving requirements.

In line with agile principles, we adopted a strategy of continuous refactoring, refining and improving our code incrementally. We initially implemented basic functions, prioritising functionality over complexity, and iteratively enhanced these functions based feedback and evolving requirements.

We also recognised the importance of collaborative code reviews in ensuring code quality and fostering knowledge sharing among team members. Regular code reviews provided valuable opportunities for peer feedback, identifying potential improvements, and maintaining coding standards consistency.

Tools and libraries

Backend:

- **Flask-CORS:** middleware to allow backend to handle requests from frontend.
- **Requests:** used to send HTTP requests to the SQL database and recipe API.
- **MySQL-connector-python:** for communication between app software and database.
- **Edamam recipe API**: publicly available recipe search API used to retrieve recipes.
- **SQL database:** built to store data on seasonal plants, customers and saved recipes.
- **Bcrypt:** password hashing library used to encrypt user passwords in the database.

Frontend:

- **React:** JavaScript library for building user interfaces. Our app is React-based.
- **Redux & Redux Toolkit:** middleware used to handle async actions and side effects.
- **React-router-dom:** used in our navbar to help the user find their way around.
- **Node Package Manager (npm):** used to install dependencies specified in our project's package.json file and to start dev server / runtime environment Node.js.

Styling:

- **MUI:** React component library that follows Material Design guidelines.
- **Boxicons** and **remixicon**: icon libraries.
- **Font-awesome:** font and icon toolkit.

Testing and evaluation

Testing strategy

The main focus of our testing strategy was component and integration testing, as well as conducting end-to-end testing.

Using Jest and React testing library, we created a series of tests for each of our components. For testing components in isolation, tests are in place to check that elements are rendering as expected and functions are delivering expected results. Components are then integration tested, checking that they behave as expected in relation to each other. On components using Redux, we also created a mock Redux store to check for the correct function.

We used the React Developer Tools Chrome extension to test:

- Props and State: by selecting components in the tree, we were able to verify that correct data is passed to child components and state changes are reflected accurately.
- Redux State Management: we monitored the store state and actions to track state changes, dispatch actions, and debug any related issues.

We used console.log statements throughout our components to log state values, props, function calls, errors or unexpected behaviour. We logged data fetched from APIs, Redux store states, and component lifecycle methods to ensure that the data is as expected and properly processed.

By combining console.log statements for logging specific data and using the React Developer Tools extension we were able to effectively test and debug our React application during the development phase, ensuring its functionality and performance.

CSS was tested on a less formal basis, in so much as visually we are seeing the expected results of the CSS on their specific pages.

Functional and user testing

Throughout development we conducted recurring tests of the function of the app. We checked that each new feature implemented behaves as expected before pushing to main, and were able to refer to our own documentation to clarify what functionality is expected.

As a result of ongoing user testing we implemented several improvements to the app to make the user experience more intuitive and dynamic. For example, we added alerts if users logged in with missing or incorrect credentials. We added a modal to redirect the user to sign up if they tried to save a recipe but were not logged in. We added conditional formatting to the navbar to hide the My account page if the user was not signed in, and dynamically changed the text of the login/logout button depending on the user’s status.

System limitations

Currently, our app requires users to manually add unique plants used in their recipes to the plant counter. This can be time-consuming and may result in some plants being overlooked or incorrectly recorded. To address this limitation, we aim to enhance the app in the future by incorporating a feature to track and add any unique plants used in recipes directly to the plant counter. This improvement will ensure more accurate and comprehensive tracking of plant usage, and will provide a more efficient user experience.

We would also extend testing further to implement more automated tests, using Cypress or Selenium to aid us. We would also include accessibility testing, for colour schemes and screen-readers. Additionally, we’d implement cross-browser and cross-platform testing to ensure our app is functional and attractive in many more implementations.

Conclusion

Building the Thirty Plant Kitchen app has been an enriching experience that consolidated everything we have learned during the Code First Girls degree.

Through identifying a real-world problem and designing an intuitive, user-friendly app, we have deepened our understanding of user-centric design principles and technical proficiency. The app's development required us to use a variety of tools and technologies, from setting up a robust backend with SQL and Flask to creating a dynamic, responsive frontend with React and Redux. Each step reinforced our technical capabilities and our ability to integrate different components into a cohesive and functional product.

The project management and collaboration aspect was integral to the success of our project. Using Jira for task allocation and progress tracking, Github to share and review code, alongside Zoom meetings and Slack communication, ensured that we maintained a structured and efficient workflow.

Quality assurance and testing were integral to our development process. By implementing component, integration, and end-to-end testing with tools like Jest and React Testing Library, we ensured the reliability and performance of our app. User testing provided valuable feedback, allowing us to iterate and enhance the application, ensuring it met user expectations and delivered a smooth experience.

In conclusion, the development of the Thirty Plant Kitchen app has showcased our ability to transform an idea into a fully functional application. It highlighted our technical skills in app development, our creativity in design, and our effectiveness in teamwork and project management. This project has solidified our learning from the CFG degree and equipped us with valuable skills and experiences that will be instrumental in our future careers.

References

1. B. BBC, "The simple formula to cut your diet's carbon footprint", BBC, . [Online]. Available: The simple formula to cut your diet's carbon footprint - BBC Food. [Accessed: 06-08-2024].
2. M. Adrienne , " Experts say we need to eat 30 plants a week. This is how I fared", The Guardian, 2024. [Online]. Available: https://www.theguardian.com/wellness/2024/apr/04/30-plants-week-gut-health. [Accessed: 06-08-2024].