Fullstack 1 Group 1
Project Report

Group Members:

Sophia John (SJ), Beth Sirak (BS), Sarah Kelly (SK), Georgia Edwards (GE), Grace Holland (GH), Anne O'Leary (AOL).

INTRODUCTION:

"Global Delights" is a web app that supports competent cooks to explore global flavours. The website's landing page contains a map that is split into the different regions of the world, it also displays country names when hovered over. Below the map, we have some buttons that a user can click on to find out more about the cuisines from that region. When the map is clicked, it will take the user to a search bar for recipes e.g. country name, region or ingredient.

This report first contains an outline of the project background and the specifications and design of the project. Technical details and responsibilities of team members are then covered in the Implementation & Execution and Testing & Evaluation sections.
Finally, we use the conclusion section to evaluate the overall outcome of our project and our key learning points.

BACKGROUND:

The project originated from the idea that the visual prompt of a world map could be used to encourage people to consider cooking recipes from regions they haven't previously considered. A click on a region button will display a summary of the cuisine of that region. The user can then search for recipes from that region and have a shortlist displayed. Clicking on a desired recipe will display a list of ingredients and a link to a website where the chosen recipe can be viewed in full.

SPECIFICATIONS AND DESIGN:

Technical Requirements:

- □ Create a home page featuring an interactive and clickable map.
- □ Implement map functionality that allows users to click on regions for further information
- □ Design a smooth user experience that directs users to the next page upon clicking a region.
- □ Integrate an API to retrieve up to 10 recipes based on user input
- □ Implement a search functionality allowing users to input ingredients for recipe search.
- □ Display recipe search results in a user-friendly format with images and labels.
- □ Provide a detailed recipe page for each recipe, displaying its information and link to instructions.
- □ Test different functions

Non-technical Requirements:

- □ Visually appealing and intuitive user interface (UI) using agreed theme
- □ Simple navigation
- □ Consistent styling and branding elements throughout the application
- □ User-friendly layout for displaying recipe details, ingredients, and instructions.
- □ Document the project and progress for reference and timelines.

Design and Architecture

The original wireframe with the expected user experience is outlined below:

Home Page

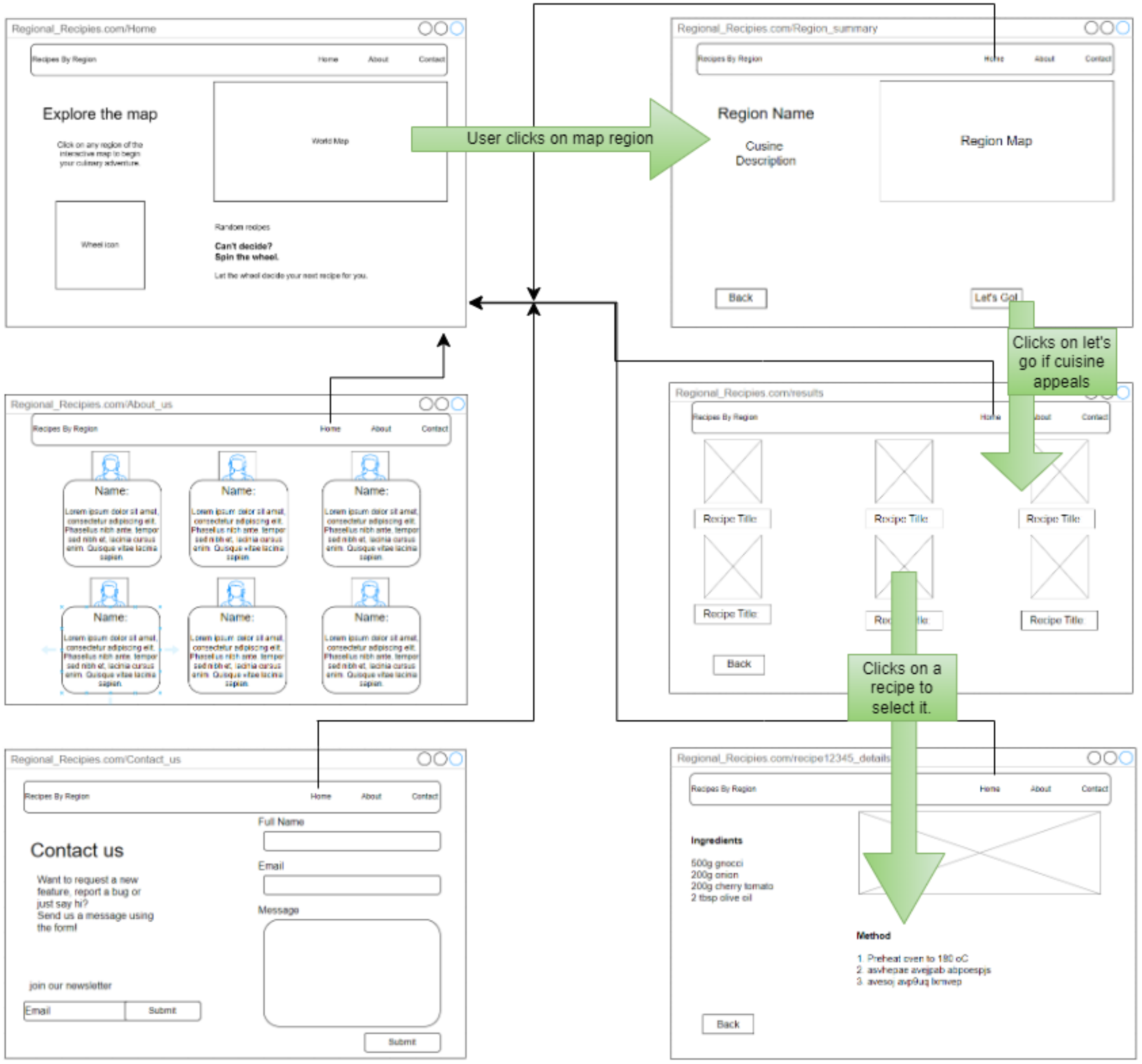
As shown by black arrows user can navigate to this page by clicking "Home" in the nav bar on any page.

About us:

User can navigate to this page by clicking "About" in the nav bar

Contact us:

Possible stretch target. User can navigate to this page via "Contact" in nav bar



Regional_Recipies.com/Home
Recipes By Region
Home
About
Contact
Explore the map
Click on any region of the interactive map to begin your culinary adventure.
World Map
Wheel icon
Random recipes
Can't decide?
Spin the wheel.
Let the wheel decide your next recipe for you.
User clicks on map region
Regional_Recipies.com/Region_summary
Region Name
Cusine Description
Region Map
Back
Let's Go!
Clicks on let's go if cuisine appeals
Regional_Recipies.com/About_us
Name:
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus nibh ante, tempor sed nibh et, lacinia cursus enim. Quisque vitae lacinia sapien
Regional_Recipies.com/results
Recipe Title
Back
Clicks on a recipe to select it.
Regional_Recipies.com/Contact_us
Contact us
Want to request a new feature, report a bug or just say hi? Send us a message using the form!
Full Name
Email
Message
Submit
join our newsletter
Email
Submit
Regional_Recipies.com/recipe12345_details
Ingredients
500g gnocci
200g onion
200g cherry tomato
2 tbsp olive oil
Method
1. Preheat oven to 180 oC
2. asvhepae avejpab abpoespjs
3. avesoj avp9uq bmvep
Back

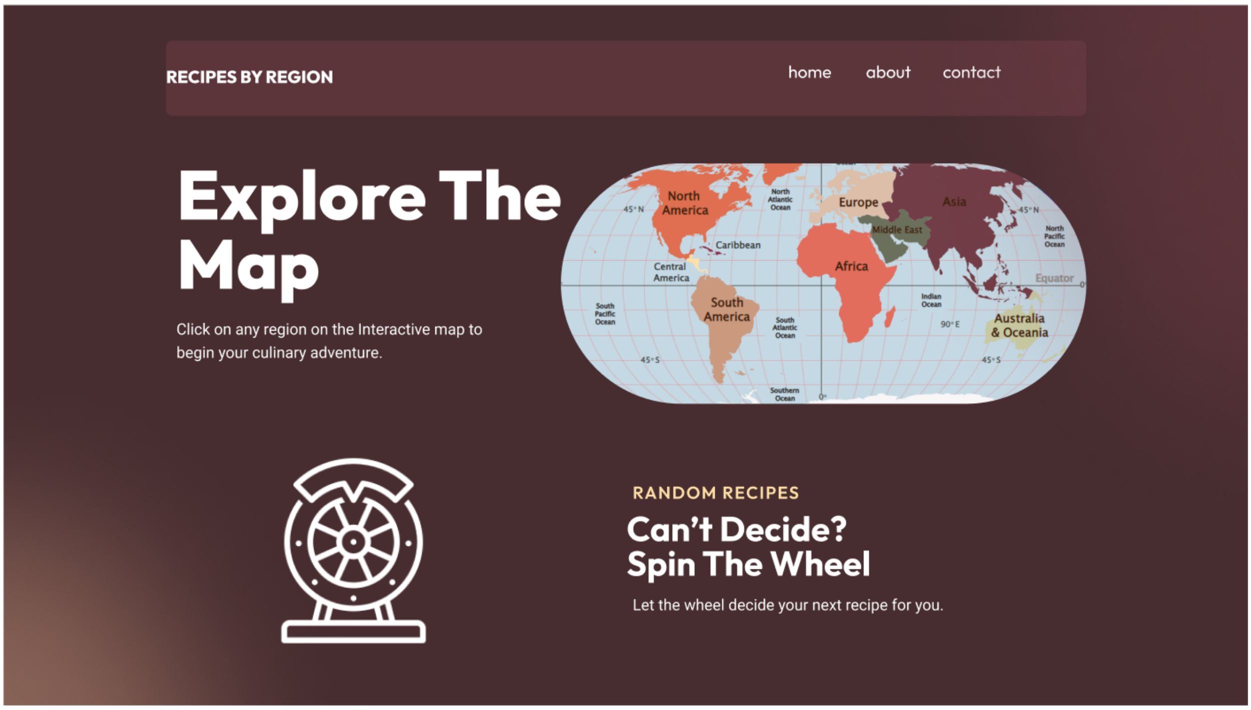
Region Summary

Recipe Results

Recipe Card

Fig 1: Wireframe produced with draw.io

A detailed design was produced using Figma:



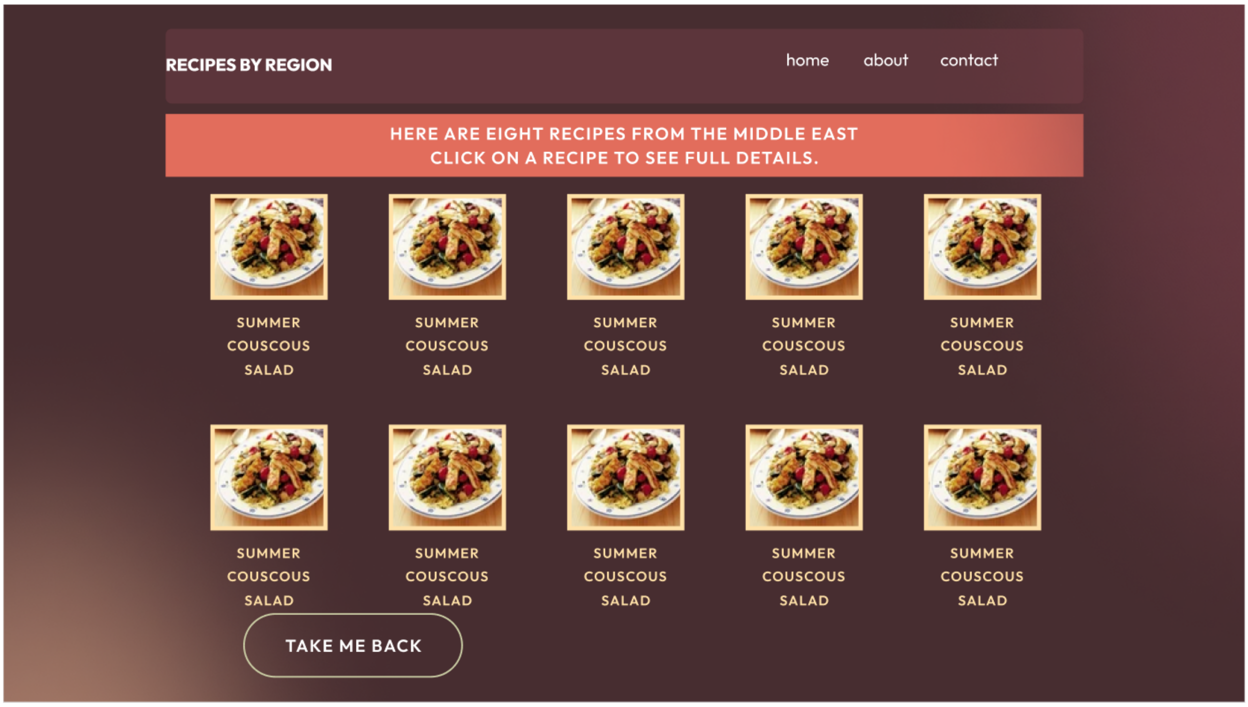
Explore The Map

Click on any region on the Interactive map to begin your culinary adventure.

RANDOM RECIPES

Can't Decide? Spin The Wheel

Let the wheel decide your next recipe for you.

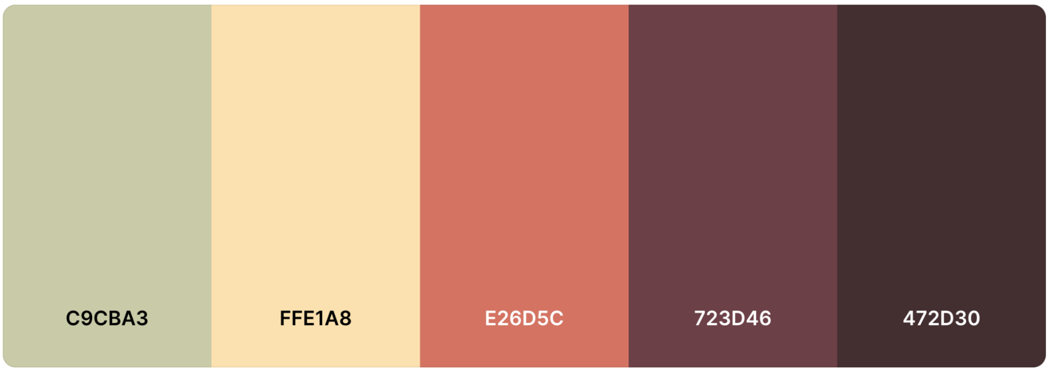


HERE ARE EIGHT RECIPES FROM THE MIDDLE EAST
CLICK ON A RECIPE TO SEE FULL DETAILS.

Fig2: Sample detailed wireframe images from Figma

Evaluation/Justification of Design:

As our target audience is quite diverse in terms of age and background we have chosen a clean modern style. We deliberately avoided any graphics or colour palettes that would appeal to any one group of people over others.
Group members reviewed possible colour palettes and selected the one shown below as the warm tone appealed:



This colour palette has the additional benefit of having good contrast between the deep aubergine (472D30) and the pale yellow (FFE1A8) and white text, making the body text easier to read. The fonts chosen were Outfit and Roboto, which offer high legibility and a “rounded” outline without the childishness associated with Comic Sans. The font size is under review and may be increased to further improve the accessibility of the website.

Our goal for the page layouts was an intuitive design allowing users to navigate the website without large quantities of text/explanation. This will included the aim of making the map regions and images and titles of recipes clickable to bring the user to the detailed recipe card. The inclusion of a navigation bar aimed to enable users to navigate the website freely.

IMPLEMENTATION AND EXECUTION:

Working Methodology and Organisation of Workload:

As a team of six with a range of shift patterns and caring responsibilities, we predominantly worked asynchronously.

To organise our work we met once or twice a week for a voice call to discuss progress/blockers and to agree on responsibilities for the following week. Outside these calls, we used the following tools to organise the team:

Tool	Use
Slack	Private group channel was the initial working area for the group. Used for all messages initially, including brief agendas for meetings, discussion of design ideas etc. Now used primarily to store relevant links (e.g. google_sheet actions diary, zoom meeting links, Figma design sheet links).
Discord	Group calls and informal/rapid response group work E.g. when GE added code to the map to link pages to regions and had her localtunnel running, she shared the link to enable another team member to check it worked for an external party.
Jira	To maintain an overview of progress on key tasks and to organise overall project workflow.

Individual team members took responsibility for key areas (working with others to deliver) as outlined below:

Team Member	Area(s) of Responsibility/Leadership
Sophia John (SJ)	Recipe card page (jsx and css formatting etc) Research contact us options - e.g. sign up to Newsletter possibly leading to ownership of this page if feasible.
Beth Sirak (BS)	API search - find suitable free API that includes country details for recipes (NB shared with SK) Set up group GitHub repository, invite all members as collaborators. Lead on linking API to React

Sarah Kelly (SK)	API search - find suitable free API that includes country details for recipes (NB shared with BS) Setup and administration of Jira board. Development of nav bar.
Georgia Edwards (GE)	Home/landing page including the development of an interactive map
Grace Holland (GH)	Research/Design for “Spin the wheel” mechanism if the user does not want to select a region. About us page (jsx and css formatting etc).
Anne O’Leary (AOL)	Wireframe - broad outline in draw.io, refined design in Figma. Project report draft/homework document. Recipe results page (jsx and css formatting etc). Redux - store, slices.

Code Implementation

We were determined to create a solid project, so we followed some well-established best practices. Collaboration was crucial, and these practices helped us organise everything effectively. We found valuable insights in Ankur Tyagi’s Scrimba article, “7 Best Practices for Structuring a React App”, and applied them diligently. This guided us in setting up our folder structure, naming conventions, error handling, code integration, code quality checks, testing procedures, and data typing.

Our focus was on creating a seamless experience. We implemented nested routing, ensuring that users could navigate content without confusion. To manage stable values that didn’t change much we used JavaScript objects as a practical storage solution. These objects also proved handy for housing complex elements like Regular Expressions that didn’t fit neatly into JSON files.

We made good use of hooks, a toolkit provided by React for building interactive elements. Hooks facilitated responsiveness, and we combined them with shared state management to create a user-friendly recipe map website. To handle API interactions, we employed a middleware approach, a sort of mediator that efficiently managed communication with the Edamam Recipe Search API. This streamlined the process and maintained a neat code structure.

Redux implementation

Our redux folder contains our original redux folder as designed when the API calls were still in development. The original aim was to place key variables/data in a store and use slices for organisation. For example, having slices for “chosenRegion” (updated by clicking the map), “chosenRecipe” (updated by clicking the image of the desired recipe). The intention was to develop towards a slice for the recipes fetched from the API which could then be accessed by the “Recipe Results Page”. However, when the code to request data from the API was developed, it was found to be very difficult to integrate this with the existing redux structure. Due to limited time, the “Recipe Results Page” code was rewritten with a smaller scale use of state contained within that file, and using router to navigate between other pages. The redux file has been retained to give the option to return to this structure should we decide to continue developing the Global Recipes app in the future.

Test Implementation

We created test cases on our app as testing is a crucial practice in coding as it ensures the reliability, functionality, and quality of software applications. Testing allowed us to identify and rectify issues, bugs, and unexpected behaviour in our code before submitting the project and eventually deploying the app. This proactive approach helped us prevent errors, enhanced the likelihood of user satisfaction, and will help to maintain the integrity of the software. Our test cases serve as a safety net to verify that each component renders correctly, displays accurate information, and functions as intended. These test cases simulate user interactions and help catch potential errors early in the development process, reducing the likelihood of bugs in the final product. By running these tests, we can ensure that the culinary information is accurately presented to users, enhancing the overall experience and contributing to a more reliable application.

React’s component-based architecture allowed us to build and reuse elements effectively. We maintained clean code formatting using tools like Prettier. We kept component names straightforward, adhered to clear styles and organised our code systematically to ensure mutual understanding among team members. This made it easier to see the relationships between different elements, understand how they fit together, and make changes as needed. We aimed to keep components small so that they were easier to test, used meaningful component names, aimed to use destructuring where possible, use functional components instead of class components, avoided using inline styles by having separate css styling, and incorporated arrow functions.

We also integrated useful tools and libraries, such as react-simple-maps for mapping functionality and zoomable-group for enhanced zoom features. Additional tools like jest ensured thorough testing, axios facilitated server communication, and BrowserRouter managed proper rendering in the browser.

Implementation Process

Implementing the code was done through atomic Github commits to local branches where the code was reviewed by other members of the team before a commit to main was established. With this, we were able to catch errors that may interfere with the main branch and cause errors when the app was run. In our weekly meetings, we were also able to see what people had been doing through screen shares. Additional meetings were held between smaller groups as and when needed so sources could be shared to help others if there were areas that were more troublesome to work out.

Agile Development

As previously mentioned, our team took an Agile approach and organised our workflow using Jira. We refactored code in response to team feedback and issued pull requests to enable the team to respond to pushes to Github.

While Agile development was in many ways effective, we did face some challenges due to the flat structure of our team. In a typical agile team, roles such as Product Owner or Scrum Master ensure that there are people who are not directly involved in coding but instead maintain an overview of the project's progress and how each developer's work will integrate. As each of us was directly involved in developing it was difficult to maintain an awareness of each other's work, despite our best efforts to do so.

On several occasions, individual members learned "on the job" how to complete their components and it was only once we had each figured out how to code our components that we realised they weren't fully compatible. Even in Agile development, it would probably be beneficial to have defined inputs and outputs from each section and an agreed architecture for the project as a whole, which at the time of starting this project we did not have the expertise to define.

Implementation Challenges

We had a few situations where the team had created components using HTML and then when trying to convert to a React file, it was not possible due to the differences in the code. As a result, we had to change the original design slightly. So, with the Wheel, to make it usable on React we had to change the design slightly. As a result, the arrow went but we were able to add hyperlinks to each of the continents pages. Another example is with our interactive map. With the HTML doc, the world was split into its regions and then when clicked on it would take you to that regions page on the react app. However, when we tried to convert to React, due to an svg not being easily changed to be jsx ready, we had to use jvectormap to create a map that was interactive and group the countries into the regions, the map would connect the API search bar and then we decided to add buttons below the map that would connect to the regions pages where you could find out more information about the local cuisines. With the way the API worked, a search bar was added and used to find recipes from specific regions rather than getting the information from the region the user clicked. We all learned a lot to make the website usable, we learnt how to get the API to work and fetch the data required based on regions. All challenges and changes were brought up with the rest of the group and others would see if they could maybe help resolve the issue or find a work around together.

TESTING AND EVALUATION:

Testing our recipe map application using React and Jest was pivotal to ensure its reliability. Jest empowered us to create an array of test cases that evaluated distinct components. These tests encompassed various functions. By mimicking user actions and scenarios, we could proactively spot and tackle potential issues, guaranteeing a seamless user journey.

In a practical setting, it's important to adhere to a pragmatic testing strategy, employing the testing pyramid concept. This framework advocates a balanced distribution of tests across different tiers. At the base, unit tests verify individual components in isolation, giving a strong foundation. Moving up, integration tests examine component interactions and functional tests replicate user interactions, backing it up against user stories. At the top of the pyramid would be user acceptance tests, getting real life feedback on the application - however we did not manage to do this for our recipe application.

Whilst testing is incredibly useful, it has some limitations. No testing can be exhaustive, and practical constraints can limit thorough testing. Factors such as user perceptions and unexpected behaviour could derail automated tests. Therefore, a comprehensive strategy would blend automated and manual testing, accommodating these factors and product assurance.

CONCLUSION

In conclusion, we have managed to get the website to a stage where it is working but we have had a few roadblocks along the way. It has been a learning experience for all involved as we were all new to React and Full stack at the start. Our idea was very specific and as a result there were not many API's that actually fit our requirements. If we were to do it again we would work from a database as this would be better for our needs. Overall as a team we have worked well together but it has been a challenge trying to coordinate times when everyone is able to make meetings and sessions and get on the same page about the overall style, idea and finish of the project especially as this is a part time course. Overall, we are proud of the final product but know that there are areas in which it could be improved but with the time and experience of the team we did well to achieve what we did.