Cook of the Wild

ENSE 374 - Team Q

Mikayla Peterson
Sebastian Lopez Melendrez
Kristina Langgard

Table of Contents

Introduction

Our goal with our project was to answer a need we felt as players of Legend of Zelda: Breath of the Wild. In the game, there is a function that allows you to cook meals to give yourself buffs to traverse the map, handle difficult weather, or survive better in combat. They give you hints about how to cook these meals, but there is no official record, either in-game or in supporting documents published by the developers. Fan-made resources for cooking are plentiful but often hard to navigate, not clear on how to make buffs, or you simply can't search or filter for what you want.

We wanted to make the experience better and more unified by amassing a database of the game's recipes, and then allowing you to filter by categories and search for recipes you may know of already, or make it easier to learn what you could make. Our goal was to provide all the relevant information in one place, with a more user-friendly experience.

Our project scope is a cookbook for our fellow Zelda players. It will display a collection of recipes up front, with more detailed views upon inspection, which will outline how to make them. Our core idea and targeted minimum viable product is being able to collectively view these recipes.

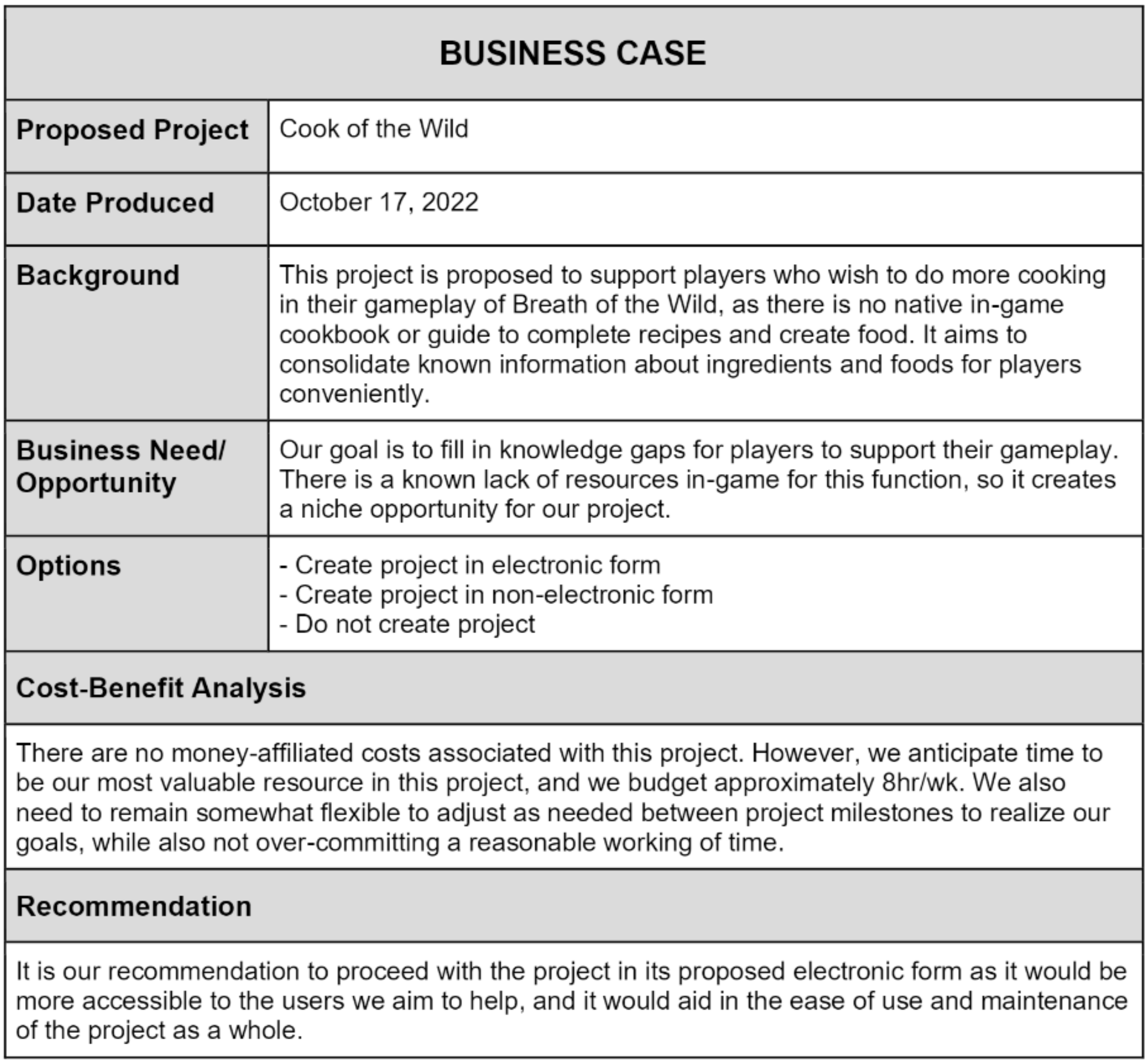
Our target demographic is other players who are experienced enough to know what the in-game items are already, so further goals also outlined more detailed information to help guide the new player experience. We also backlogged add-ons to include accounts to allow favouriting recipes, as our core experience is just to provide information in a simple and straightforward manner for hungry players.

Cook of the Wild
An Interactive Cookbook for
THE LEGEND OF ZELDA BREATH OF THE WILD

Image 1: GitHub Banner

Documentation

Our official documentation, which is all available for viewing on our GitHub, outlines our project scope and intention, though it is mostly kept simple on purpose as that is part of what we are selling—it does admittedly fit well for the time constraints we experienced in class.



BUSINESS CASE	
Proposed Project	Cook of the Wild
Date Produced	October 17, 2022
Background	This project is proposed to support players who wish to do more cooking in their gameplay of Breath of the Wild, as there is no native in-game cookbook or guide to complete recipes and create food. It aims to consolidate known information about ingredients and foods for players conveniently.
Business Need/ Opportunity	Our goal is to fill in knowledge gaps for players to support their gameplay. There is a known lack of resources in-game for this function, so it creates a niche opportunity for our project.
Options	- Create project in electronic form - Create project in non-electronic form - Do not create project
Cost-Benefit Analysis	
There are no money-affiliated costs associated with this project. However, we anticipate time to be our most valuable resource in this project, and we budget approximately 8hr/wk. We also need to remain somewhat flexible to adjust as needed between project milestones to realize our goals, while also not over-committing a reasonable working of time.	
Recommendation	
It is our recommendation to proceed with the project in its proposed electronic form as it would be more accessible to the users we aim to help, and it would aid in the ease of use and maintenance of the project as a whole.	

As outlined in our documentation, Mikayla was our project manager, and actively filled most roles where we needed her most. Sebastian was our front-end designer and developer. Kristina was our back-end developer and database architect.

System Requirements

As our project is web-based, there should be no limitations to who could potentially view it. Currently, of course, the project is only local, but as we've been able to test ourselves on our different computers with different web browsers, it works uniformly. This is consistent with our goal of approachability and utility to serve the need we outlined with the scope of our project; we hope anyone, theoretically, would be able to make use of our cookbook to enhance their gameplay.

Front-end

The website's core display functionality is based on many concepts we practised in the lab, including HTML, CSS, Javascript, Node, and then connecting to our database with MongoDB. The design was meant to be kept sleek and not too busy, but concise enough to have all the information required for the player. Additionally, the colours were meant to be a reference to the game itself, with cyan-like blue tones and white highlights that are similar to Link's tunic in-game.

The front page by default will load all recipes from the database in no particular order, with images and names associated with each recipe card to make navigation easier. Recipes are organised by category, and the user may scroll through all recipes in the database this way. The sidebar then allows the user to filter by these categories to limit what is displayed on the front page. The user may also select multiple categories to filter by.

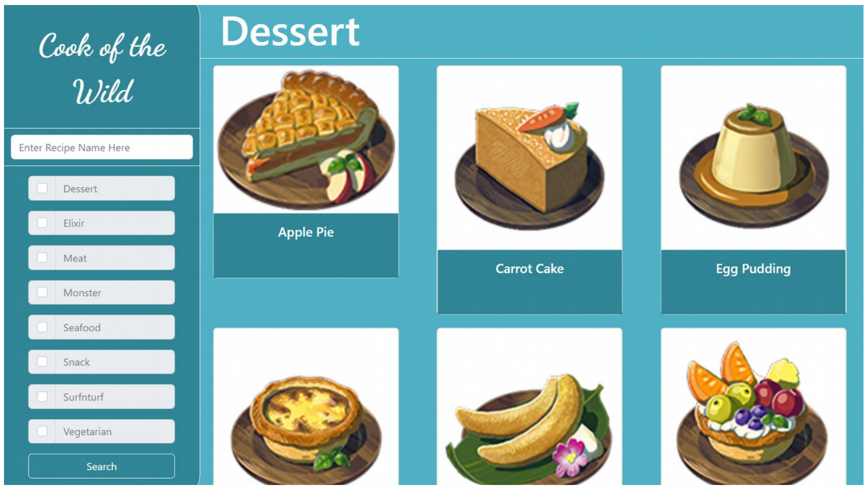


Image 2: Recipes Page (no filter)

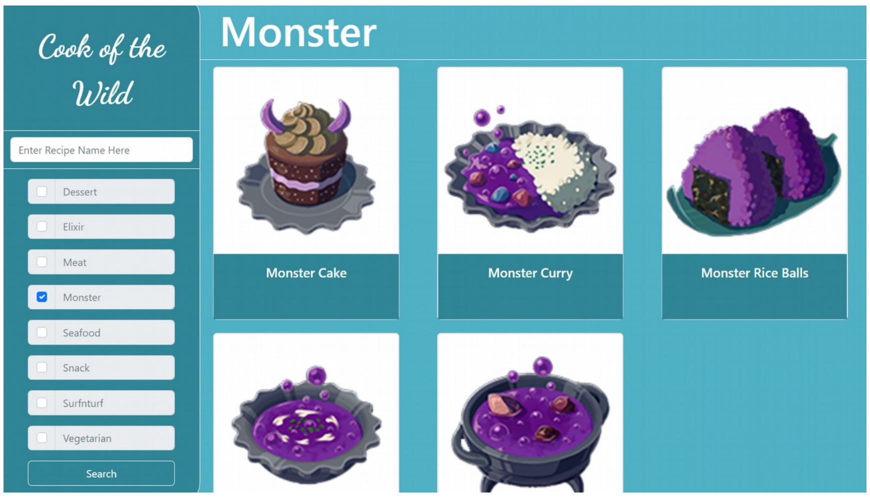


Image 3: Recipes page (filtered by "Monster")

One of our primary concerns for the filter category was that it should be easy to navigate and use. It was also important that the options provided would build and work together instead of against each other. As such, both the search and filtering options would ideally be linked together. As of the current build, the search function looks up the exact recipe name in the database, but we hope to implement a more user-friendly version by having it look up recipes similar to whatever is searched. To further help user usability, we have also made all searches case-insensitive. While the current search and filter options work separately, in future updates to the project we would like for these to be able to work together. For example, were the user to look up a basic word like “cake” and toggle the monster filter option, the “Monster Cake” recipe should show up.

Finally, clicking on any recipe card routes you to the recipe detail page, which accesses the recipe information the user is searching for. Images and descriptions are just aesthetic here. Everything is generated dynamically based on the user’s choice, so information is pulled from the database via MongoDB as the user browses.

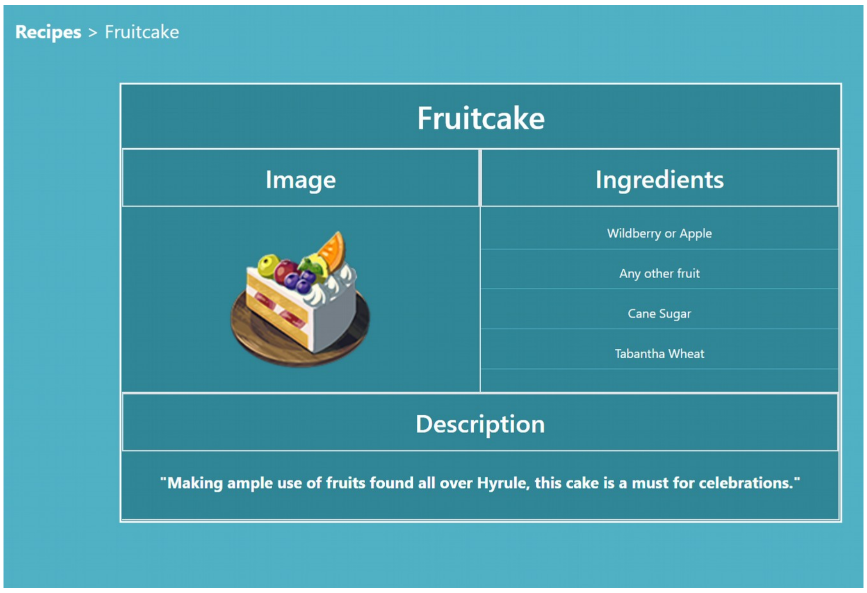
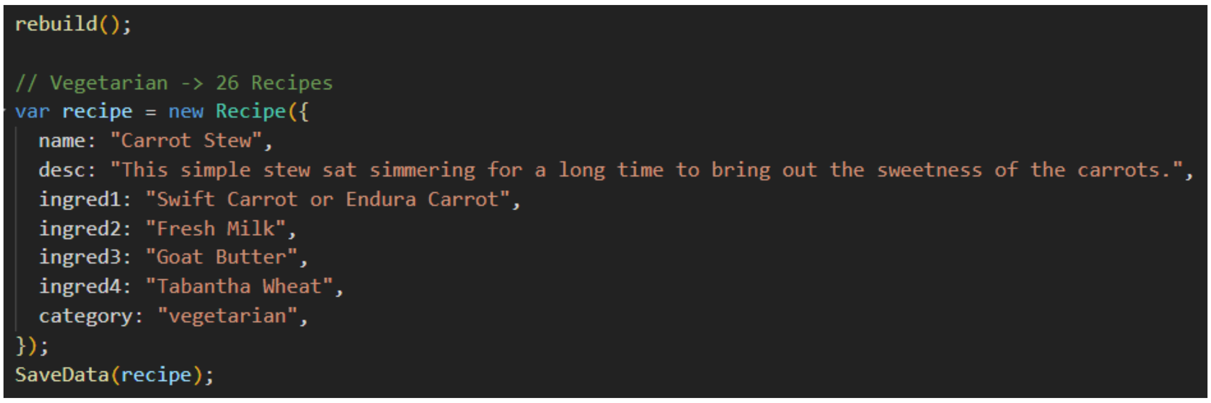


Image 4: Recipe Detail Page

Back-end

The database was put together in MongoDB, and it’s set up with a script written in Javascript. Although we acknowledge in industry this step would not be necessary, we found this worked great for trading the database information between the group members since we were working on different computers.

The recipes are written and saved by the script each run; the script also dumps the database on each load to prevent duplicates. The reason we chose this approach was for simplicity; we acknowledge it is not the most efficient, but we do not need to wait for it to parse the database looking for duplicates, and edits are recognized as soon as the script is run again. The script file only needs to be included on the front-end, removing any overlap between the two, and it handles the data itself on everyone’s computer.



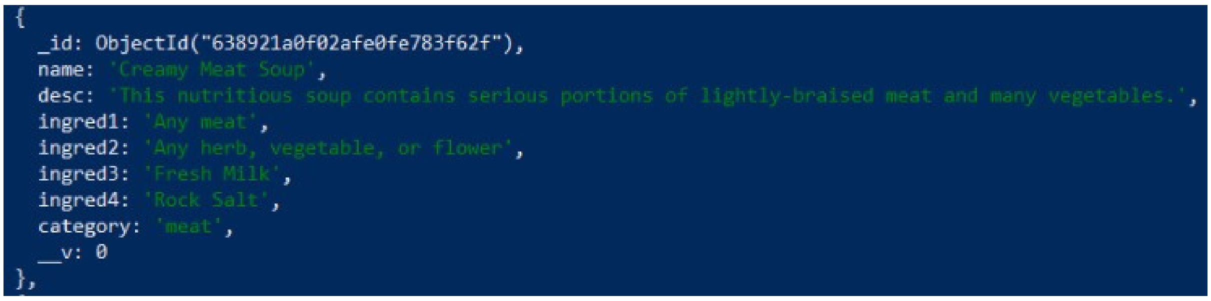
```
rebuild();

// Vegetarian -> 26 Recipes
var recipe = new Recipe({
  name: "Carrot Stew",
  desc: "This simple stew sat simmering for a long time to bring out the sweetness of the carrots.",
  ingred1: "Swift Carrot or Endura Carrot",
  ingred2: "Fresh Milk",
  ingred3: "Goat Butter",
  ingred4: "Tabantha Wheat",
  category: "vegetarian",
});
SaveData(recipe);
```

Image 5: Database File

The most work required with the database was the initial recipe setup itself. We included over a hundred recipes in our project by the end. We started slowly and entered a couple of each to test and demo, and we incrementally worked to include more recipes in each category. This was done not only for authenticity but also to demonstrate more of our filtering system and our dynamic front-end by having more examples to use. In the end, we believe it also really made the project come to life from our original vision.

Recipes themselves are defined as objects with a name, description, and up to five ingredients, as well as a category based on the kind of dish. This was done not only to make the front-end more appealing and overall easier to write, but so that each characteristic of the recipes could also be used in the filter system. Recipe pictures that follow the same nomenclature were included so they could also be called dynamically.



```
{
  _id: ObjectId("638921a0f02afe0fe783f62f"),
  name: 'Creamy Meat Soup',
  desc: 'This nutritious soup contains serious portions of lightly-braised meat and many vegetables.',
  ingred1: 'Any meat',
  ingred2: 'Any herb, vegetable, or flower',
  ingred3: 'Fresh Milk',
  ingred4: 'Rock Salt',
  category: 'meat',
  __v: 0
},
```

Image 6: Recipe in the database

Buffs in foods were rather difficult to implement in this format since several different buffs could be applied to one food, depending on which ingredient you use; for example, a recipe that calls for ‘any mushroom’ could be made with a mushroom that applies a cold-resistance buff, or a heat-resistance buff, and still make the same food. We opted not to include detailed buff information in our final product and focus on the food recipes themselves. Subsequently, we did not include elixirs, although they could be considered food. There are varying degrees of effectiveness of elixir buffs which provide a level of intricacy to implement beyond the scope of the foods we already included. To account for this, perhaps we would have to include a few iterations of each elixir.

MVP1

Our original milestone was fully realised in the scope of our project and was what we could demo for our presentation. It focuses on the core feature of loading from the database and filtering or searching through to find different recipes. A user may then select a recipe, which loads a new page with the dynamic information they've selected to show them the recipe details.

MVP2

Our second milestone would bring in the member experience, including a login function to allow for favouriting recipes for future use. This was pushed back separately as it is not a core function to operate with the use of the database and was only for user experience and customer satisfaction.

Additionally, the user would likely want to be able to see the buffs that each food would give. While originally an MVP1 task, the gathering of these buffs was a more time-consuming and complex process than originally thought. As such, a decision was made to push it back to MVP2 so that the feature could be better implemented and researched. Unfortunately, we did not have time to see this completed alongside the recipes.

Finally, more in-depth ingredient information and locations were another piece of information we wished to offer players. This could have been relatively iterative to implement dynamically based on data we already had, but it would have still required significant time dedication in front-end work to account for multiple different ingredients and switch cases. This information could have worked well with the food buffs, so we decided we did not have the time to allow for the kind of overhaul required to implement this.

Project Feedback

The feedback we got during the peer review was positive overall. The critiques that we did get from Team Riker weren't actionable for this project but are things we would like to consider in the future. One piece of feedback that we will keep in mind in the future is to remember to include images or videos of what we're talking about during a vlog. Another piece of feedback was that our lofi prototype was a bit difficult to read. This could have been fixed by either using software or just being more careful when drawing the prototype by hand.

Summary & Reflection

Altogether, our team is very proud of the project we came up with, and the work we completed. We enjoyed our time, had fun making it, and were very happy to show our final product. Since we were able to accomplish all of our objectives for MVP1, we feel that we successfully completed the project. Something that we like the most about our project is that we can see ourselves actually using this website in the future while playing Breath of the Wild. The only thing we didn't really like about the project was the time limitation; we wish we had more time to implement experiences beyond MVP1. We're proud of our final product and how we worked together as a team. All members of the team did their best to support each other, communicate when they were having issues, and did their assigned tasks to the best of their ability.

Collectively we learned that to have an effective and productive team, the team members need to trust each other and be pragmatic. When assigning tasks to members, we had to trust one another to not only complete the task on time but also do it to the best of their ability. The team also did a great job being honest with themselves and the rest of the team when they were struggling to complete a task. On an individual level, Mikayla learned that she doesn't have to micromanage people for a project to succeed and that she enjoyed the knowledge-management aspects of the project. Sebastian learned that he enjoyed doing both front-end and back-end work, and he's considering becoming a full-stack developer. Kristina found that full-stack development was very satisfying, albeit a lot more work, and she also learned that she likes working with databases.

There are a lot of things we learned/experienced that we will be using going forward. The overall experience of going from project initiation to execution is valuable for not only future group projects and capstone but for working in industry too. Kanban boards were a particular thing we found useful and will use in the future. We were able to practise our soft skills, which we will be using in future projects and our jobs. After doing some reflection, we couldn't think of anything that we've experienced that we might not need to use in the future. Sure, depending on what we're doing as a career, we might not be writing the initial project management documents, or we might not be the ones drafting the architecture diagrams but we will likely need to be able to understand them. We wish we had a little more guidance on what to put in some of the project management documents and the UML diagrams; we weren't sure if we were putting enough detail or the right kind of detail in these documents.