University of Regina | Go far, together.

ENSE 374 – Software Engineering Management

Cookify

Hashir Owais - 200483044

Nathan Okoh 200492890

Simranjit Gahra – 200484408

Table of Contents

List of Figures

List of Tables

1 Introduction

The proposed project, "Cookify," aims to create a web-based application that allows users to input a list of ingredients they have and receive personalized recipe suggestions. This application is tailored towards young adults, university students, and single professionals looking for meal inspiration, improving cooking skills, and reducing food waste. The goal is to make cooking more accessible, save money, and cater to dietary restrictions.

1.1.1 Background and Need

Busy lifestyles often result in challenges to cook at home, and food waste is a significant problem globally. Around 30% of all food produced is wasted, with a notable contribution from households. The lack of easy access to recipes based on available ingredients also contributes to this issue. Cookify seeks to fill this gap by providing users with tailored recipes, promoting home cooking, reducing food waste, and offering cost-saving benefits.

1.1.2 Project Requirements and Stakeholder Context

The project will develop a platform that uses both an external API for dynamic recipe data and a local database for user preferences, saved recipes, and dietary restrictions. The decision to create a web-based app leveraging both these elements balances real-time access to recipe data and user personalization needs. Stakeholders have recommended focusing on scalability, user-friendliness, cost-effectiveness, and the ability to expand features over time, such as integrating additional APIs.

For a detailed breakdown of requirements, please refer to the "Project Requirements Document."

1.1.3 Overview of the Next Sections

The upcoming sections will outline the design considerations, technical architecture, development process, and an analysis of options evaluated before recommending the final approach. This will provide insight into how the hybrid solution with an external API and local database offers a balance of efficiency, personalization, and future scalability without detailing specific results or implementation details here.

2 Design Problem

2.1 Problem Definition

link to the Business Case.

2.2 Project Charter

link to the Project Charter.

3 Solution

To create a functional and user-friendly recipe-finding web-application, we considered multiple design solutions that would help users search for recipes based on available ingredients. Each solution was evaluated based on its ability to meet project requirements, including ingredient matching, ease of use, data reliability, and scalability. After brainstorming, we compared the potential solutions to determine which best aligns with our objectives and constraints. Below is a summary of each solution we explored, along with our reasoning for the final choice.

3.1 Solution 1: Static Webpage with Basic Keyword Matching

Our first solution was a straightforward static webpage where users could input ingredients, and the application would return recipes based on keyword matching. For example, if a user input “cheese,” the application might suggest a recipe like grilled cheese based solely on the keyword match.

Advantages of this approach:

- Simplicity: This approach is easy to implement and requires minimal resources, making it ideal for a quick, low-maintenance solution.
- Low Cost: With no need for a backend database or API integration, this solution would be cost-effective and would not require extensive infrastructure.
- Quick Deployment: A static webpage could be developed and deployed quickly, providing a fast turnaround for basic functionality.

Dis-advantages of this approach:

- Limited Ingredient Matching Accuracy: Without a structured database or algorithm, this solution would likely produce inaccurate results. For example, suggesting “grilled cheese” based on the keyword “cheese” alone would overlook essential ingredients like bread.
- Lack of User Personalization: This solution would not support any user-specific features like saved recipes, dietary preferences, or previous searches.
- Minimal Scalability: Expanding this solution to include additional features or a broader recipe database would be challenging and require significant rework.

While this solution had some advantages in terms of simplicity and low cost (in terms of time), its limitations in ingredient matching accuracy and lack of personalization led us to explore more dynamic options.

3.2 Solution 2: Database and External API Integration

To address the limitations of Solution 1, we developed a more advanced approach that involved using both a local database and an external API. This design would allow the app to provide more accurate recipe suggestions by matching ingredients through the external API, while a local database would support user-specific features.

Advantages of this approach:

- Improved Ingredient Matching: The external API enables more precise recipe suggestions by accessing a wider range of recipes and filtering based on actual ingredient lists.
- Enhanced Personalization: The addition of a local database allows users to save recipes, set dietary restrictions, and view past searches, making the app more interactive and tailored to individual needs.
- Scalability: This approach provides a foundation for future updates, such as adding more recipes, refining the matching algorithm, or integrating additional APIs as needed.

Dis-advantages of this approach:

- Dependency on API Availability: Relying on an external API means that recipe data is dependent on a third-party provider, which could lead to issues if the API becomes unavailable or undergoes changes.
- Increased Complexity: Integrating both a local database and an external API requires a more complex infrastructure, which may increase development time and maintenance needs.
- Higher Cost: Compared to a static webpage, this solution would be more costly to implement (in terms of time) and maintain due to the required backend infrastructure.

Despite these challenges, this solution was still promising as it addressed most of the limitations from Solution 1 and provided improved functionality. However, we sought a balanced approach that would combine real-time data access with enhanced user personalization.

3.3 Final Solution: Hybrid Web-Based Application with External API and Local Database

After weighing the pros and cons of each approach, we decided on a hybrid solution as the final design. This solution leverages both an external API for up-to-date recipe data and a local database for storing user-specific information, such as saved recipes, preferences, and dietary restrictions. This hybrid model combines the strengths of both previous solutions, offering real-time functionality with a high degree of personalization.

Advantages of this approach:

- Real-Time Access and Personalization: By using an external API for recipe retrieval, the app can provide real-time suggestions based on user-input ingredients. At the same time, the local database allows for personalization by saving user preferences, dietary restrictions, and frequently accessed recipes.
- Future Scalability: This design is built to adapt and scale as needed. Additional features, such as new APIs or advanced filtering options, can be incorporated into the system with minimal restructuring.
- Enhanced User Experience: Combining real-time data access with user-specific storage, the hybrid approach ensures a seamless, interactive experience. Users

can access fresh recipe options while enjoying a personalized app interface tailored to their preferences and history.

- Reliability and Redundancy: By storing essential user data locally, the app can maintain functionality even if the external API experiences downtime, ensuring that users can still access saved recipes and settings.

Table 1: Comparison of Solutions

Feature/Constraint	Solution 1: Static Webpage	Solution 2: Database & API Integration	Final Solution: Hybrid Web-Based App
Ingredient Matching Accuracy	Low	Medium	High
User Personalization	Minimal	Moderate	High
Data Scalability	Low	Medium	High
Ease of Implementation	High	Moderate	Moderate
Real-Time Recipe Access	No	Yes	Yes
Future Feature Integration	Limited	Moderate	High

In conclusion, the hybrid web-based application with an external API and local database provides the most comprehensive solution. This approach ensures accuracy in recipe suggestions, personalized features, and the flexibility to expand as user needs evolve, making it the best fit for the project goals.

3.3.1 Components

What components you used in the solution? What is the main purpose of using individual component? Provide a block diagram (with a numbered caption, such as Fig. 1) representing the connectivity and interaction between all the components.

3.3.2 Features

Give an account of all the features your solution has. These features may be tabulated (with a title) for improved comprehension.

3.3.3 Environmental, Societal, Safety, and Economic Considerations

Explain how your engineering design took into account environmental, societal, economic and other constraints into consideration. It may include how your design has positive contributions to the environment and society? What type of economic decisions you made? How did you make sure that the design is reliable and safe to use?

3.3.4 Limitations

Every product has some limitations, and so is the case with your design product. Highlight some of the limitations of your solution here.

4 Team Work

Since this is a group project, you must have a fair distribution of tasks among yourselves. To this end, you must hold meetings to discuss the distribution of tasks and to keep a track of the project progress.

4.1 Meeting 1

Provide Links to ‘Meeting Agenda, Meeting Minutes, Change Request, Project Status Report, Issue Log’ documents.

4.2 Meeting 2

Provide Links to ‘Meeting Agenda, Meeting Minutes, Change Request, Project Status Report, Issue Log’ documents.

4.3 Meeting 3

Provide a similar description here.

4.4 Meeting n

Provide a similar description here.

5 Project Management

Provide the link to ‘Milestone-based Schedule’ document. Use Gantt chart as well to show the progress of your work here. Mention all the tasks along with their predecessors. Provide the slack time of each task and identify the critical path.

6 Conclusion and Future Work

- A summary of what you achieved.
- Provide Link to ‘Lessons Learned Report’ document.
- While keeping the limitations of your solution, provide recommendations for future design improvements.

7 References

- Use the IEEE reference style.
- Do not put any reference if it is not cited in the text.

8 Appendix

If you want to provide an additional information, use this appendix.