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

Tabl

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	Low	Medium	High
User Personalization	Minimal	Moderate	High
Data Scalability	Low	Medium	High
Ease of Implementation	High	Moderate	Moderate
Real-Time Access	No	Yes	Yes

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

The Cookify application integrates several critical components, each designed to fulfill a specific role in delivering a seamless user experience:

External API: Fetches real-time recipe data from an extensive online repository, ensuring dynamic and accurate ingredient-to-recipe matching.

Local Database (MongoDB): Stores user-specific information such as saved recipes, dietary preferences, and frequently accessed data, enabling a personalized experience.

Front-End (HTML, CSS, JavaScript): Provides an intuitive and user-friendly interface for inputting ingredients and displaying recipe results, ensuring smooth navigation.

Back-End (Express.js): Manages communication between the front end, database, and API, facilitating efficient data transfer and processing.

Authentication System (Passport.js): Secures user accounts and personal data, ensuring privacy and safe access to personalized features.

Hosting Platform: Deploys the application, allowing access across multiple devices with consistent performance.

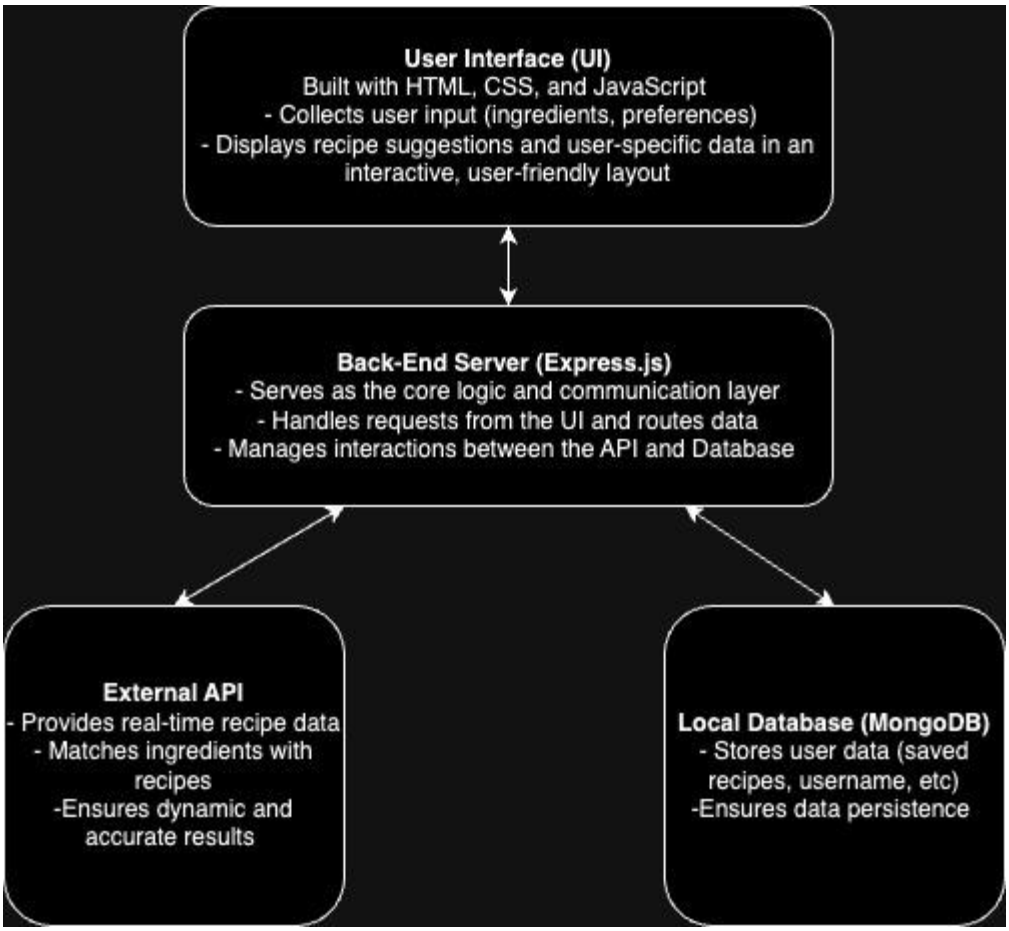

Figure 1 – Block Diagram

3.3.2

3.3.3 Features

The Cookify application includes the following features, designed to enhance functionality and usability:

Table 2: Features and Descriptions

Features	Description
Real-Time Recipe Suggestions	Utilizes an external API to provide accurate recipes.
User Personalization	Saves dietary preferences, frequently accessed recipes, and other settings.

Ingredient Matching	Matches user input ingredients to a recipe database
Secure Accounts	Authentication system to ensure user's privacy and security
Scalability	Designed for integration with future features like new APIs more filtering options

3.3.4 Environmental, Societal, Safety, and Economic Considerations

Cookify incorporates several key considerations to ensure its environmental, societal, safety, and economic impacts are positive and meaningful:

Environmental Impact: the application encourages sustainable cooking practices by helping users reduce food waste. By suggesting recipes based on ingredients they already have, users are less likely to discard unused food.

Societally Impact: Cookify promotes healthy eating by offering recipes tailored to dietary preferences and restrictions, catering to diverse nutritional needs. It also empowers users to save money by reducing reliance on dining out or purchasing unnecessary groceries, making home-cooking more accessible and affordable.

Economic Impact: the project team strategically selected cost-effective tools like MongoDB, Express.js, and Passport.js to minimize expenses. Using these open-source technologies not only keeps costs low but also allows for scalable development without compromising quality.

Safety and Security: Another critical aspect of the design. Cookify prioritizes data security by implementing robust authentication systems, ensuring that users' personal information and preferences are safeguarded against unauthorized access.

3.3.5 Limitations

While Cookify offers numerous benefits, certain limitations were identified during its development:

API Dependency: The application relies on an external API for real-time recipe suggestions, making it susceptible to downtime or service changes.

Offline Access: The lack of offline functionality limits the app's usability in areas with inconsistent internet access.

Scalability Challenges: Significant database restructuring may be required as the application grows to include more features or handle larger datasets.

Language Accessibility: Currently tailored for English-speaking users, which may limit its appeal and usability for non-English speakers.

4 Team Work

4.1 Meeting 1

Meeting Agenda#1.pdf
Meeting Minutes#1.pdf

4.2 Meeting 2

Meeting Agenda#2.pdf
Meeting Minutes#2.pdf
Project Status Report 1.pdf

4.3 Meeting 3

Meeting Agenda#3.pdf

Meeting Minutes#3.pdf

Project Status Report 2.pdf

4.4 Meeting 4

Meeting Agenda#4.pdf

Meeting Minutes#4.pdf

4.5 Meeting 5

Meeting Agenda#5.pdf

Meeting Minutes#5.pdf

Project Status Report 3.pdf

4.6 Meeting 6

Meeting Agenda#6.pdf

Meeting Minutes#6.pdf

4.7 Meeting 7

Meeting Agenda#7.pdf

Meeting Minutes#7.pdf

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.

Milestone-Based Schedule

The project’s milestone-based schedule provides a breakdown of the main tasks, and deadlines. The complete schedule document is accessible here...

Milestone-Based Schedule.pdf

Gantt Chart

The Gantt chart visualizes the general timeline of the different phase of our project

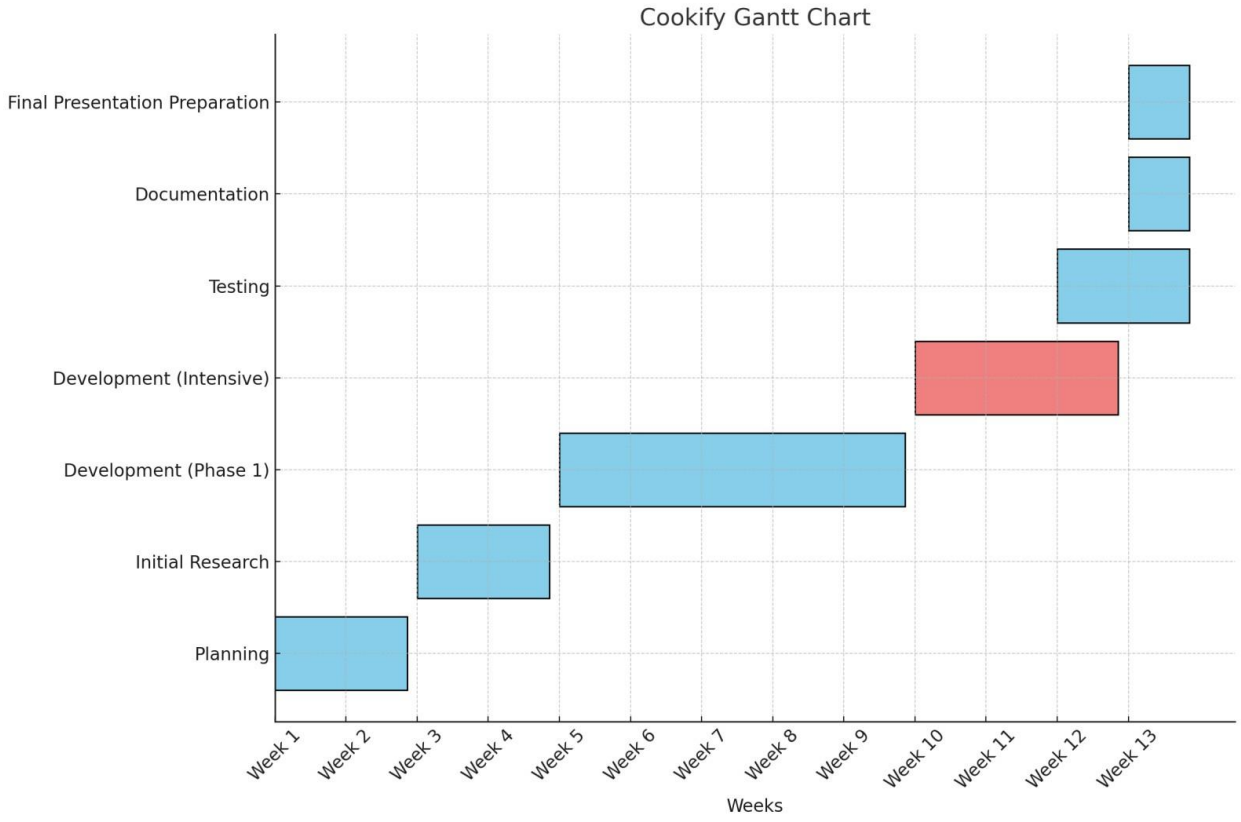

Figure 2 – Gantt Chart

Table 2: Slack Time and Critical Path Analysis

Task #	Task Name	Start Date	End Date	Slack Time	Critical Path
1	Project Initialization	September 27, 2024	October 4, 2024	0 days	Yes

2	Design Finalization	October 5, 2024	October 18, 2024	0 days	Yes
3	Front-End & Back-End Dev	October 19, 2024	November 14, 2024	0 days	Yes
4	API Integration	November 15, 2024	November 20, 2024	0 days	Yes
5	Testing and Debugging	Ongoing (starts during development)	December 1, 2024	N/A	No
6	Final Deployment	December 1, 2024	December 2, 2024	0 days	Yes

The slack time between each task was 0 days because we started each task right after the previous and followed the critical path of the project. We incrementally worked on tasks 1 to 6 to ensure we completely the project on time

6 Conclusion and Future Work

Conclusion

The Cookify project successfully delivered a functional and user-friendly recipe-generation web application. By leveraging a hybrid design approach, the application combines real-time recipe retrieval using an external API with personalized features enabled by a local database. Core functionalities such as ingredient matching, dietary restriction filtering, and recipe saving were seamlessly implemented, enhancing the user experience.

The project effectively addressed key societal and environmental challenges, promoting sustainable cooking practices by reducing food waste and encouraging home-cooked meals. Using open-source technologies like MongoDB, Express.js, and Passport.js ensured cost-efficiency while maintaining a robust and scalable design. Despite encountering challenges such as the learning curve for new technologies and minor delays in task synchronization, the team overcame these through collaboration, iterative development, and guidance from ENSE labs.

Cookify has laid a solid foundation for future enhancements, successfully demonstrating its potential as a scalable and impactful application for improving meal preparation accessibility.

Link to Lessons Learned Report

Future Work

To enhance Cookify further and address the limitations identified during development, the following improvements are proposed:

Add Recipe Images: Integrate recipe images into the interface to create a more engaging and visually appealing user experience.

Customizable Recipe Categories: Allow users to categorize recipes based on preferences such as "Quick Meals," "Budget-Friendly," or "Low Calorie" to improve usability and personalization.

Native Mobile App Development: Expand accessibility by developing a native mobile version of Cookify, enabling seamless use on smartphones and tablets.

Advanced Filtering Options: Introduce filters for recipe suggestions based on cooking time, difficulty, and dietary needs for a more tailored user experience.

These recommendations aim to refine Cookify's functionality, expand its user base, and enhance its positive societal and environmental impact. With these improvements, Cookify has the potential to become a widely used platform for accessible and sustainable meal preparation.

7 References

8 Appendix