Happie: Project Final Deliverable

the meal planning application using a microservice technology

FLORIS VOSSEBELD, University of Twente, Netherlands
EVAN FRANCISZOK, University of Twente, Netherlands

Introduction

Welcome to our detailed report on the ongoing development of our Intelligent Recipe and Meal Planning Application. This document provides an in-depth look into how we've built our application using microservices architecture.

Project Description

Our project revolves around creating a smart meal planning tool. The goal is to make it easier for users to plan meals, manage their pantry, and create optimized shopping lists that minimize the cost of the groceries. We also give the users the possibility of assigning preferences and diet restrictions. We realised this by using a microservice architecture. Services included: user profile management, recipe suggestions, inventory tracking, and shopping list optimization.

Report Structure

This report is organized into several main chapters, each focusing on a specific aspect of our project:

- Motivation: We explain why we have made a meal planning application and what we aimed to achieve.
- Business Process: Here, we explain the main functions that our application performs and how they benefit potential users.
- Architecture: This chapter breaks down how we designed our application using microservices architecture.
- Design Decisions: We talk about the technology and architecture choices we made and why we made them.
- Validation: We explain how we tested our application to make sure it works well.
- Relevant Information: Finally, we cover any other important details about our project.

Table of contents

Motivation

Our motivation for this project has come forward as we both are students that like to sport a lot and don't want to spend a lot of time having to think about what, how and with what ingredients we need to cook. Added to this, one of us does have a food restriction (vegetarian) that makes finding good recipes harder. As both of us are students that don't have a large budget to live with. We would like to minimize the cost of the meals that we will have to make, certainly as we live with other people and we will need to make meals for multiple people.

As normally meal planning can be a time-consuming task that requires a lot of comparisons, searching in books or on the internet. Especially when you want to cook a new meal with some dietary restrictions. Because there is no fun in cooking and eating the same meal over and over again. This resulted in our idea to make a meal planning application that will help with these problems and streamline the experience of cooking.

Our goal is to develop an Intelligent Recipe and Meal Planning Application that addresses these pain points and helps users to make informed decisions that align with their preferences, dietary constraints, while being possible with their lifestyle.

Our objectives include:

- A personalized page containing all the features that will help with streamlining the meal planning and cooking experience. Based on their dietary preferences, diet restrictions, allergies and cost of the meals.
- Tracking of the users Inventory and generating shopping lists. This enables users to efficiently manage their pantry inventory and supermarket trips.
- These features should be implemented in a flexible and scalable designed application that allows ease of integration with external services, future improvements and expansions.

With these objectives we will try to create an application that will solve our problems with meal planning, while being a good submission for the SOA course.

Business process

Our meal planning application supports several functionalities aimed at the simplification of meal planning and the thereby required activities. These processes are designed to streamline the meal planning experience for users and facilitates decision-making by providing suggestions.

- *Recipe Suggestions*:
  One of our primary business processes will be the recipe suggestion. This service will use AI, magic algorithms or other advanced methods to generate recipes based on the user's dietary restriction, allergies and preferences. This will help decision making by only providing possible recipes for the

user. It has to be said that for this submission we used a database with some test recipes that will be cross referenced with the user preference to generate our suggestions.

- *Inventory Management*:
  Also implemented in our application is inventory management. This will allow users to track their pantry items, quantities and the expiration dates. Currently can this information only be used as insight and for the meal planning and indirectly by the shopping list services. In the future this could be extended for other uses. For example, usage patterns.

- *Shopping List Optimization*:
  This business process enables users to automatically retrieve (optimized) shopping lists based on their meal planning and their pantry inventory. This will also show what supermarket to go to in order to get the best price for the groceries, taking promotions into account. This business process will ensure that the user only purchases items that the user needs and avoid unnecessary purchases.

- *Meal Planning*:
  The meal planning connects all the other business processes together and makes the application complete. The meal planning puts the chosen suggestions together and presents it to the users. The user is able to change some specific suggestions if they are not wanted.

These business processes together help achieve the overarching goal of our Meal Planning Application: to simplify meal planning, enhance user satisfaction, and streamline the whole process.

Architecture

Our Meal Planning Application is designed with a microservices architecture, this ensures scalability, flexibility, and modularity. because we use this architecture we are able to integrate various services, each responsible for specific functionalities. In this section, we will go over the architecture of our solution, describing the services defined and how they collaborate with each other.

In our application we implemented the following microservices:

- *Meal planning User Interface:*
  Even though we do not call it a service, it is. This service provides the user with the UI and represents the data from the other services so the user can easily view and interact with the data.
- *Meal Planning Service:*
  Facilitates meal planning by providing the best fitting personalized recipe suggestions, taking into account the current inventory availability.
- *Recipe Suggestion Service:*
  This service provides the Meal Planning Services with the meal suggestions based on the user preferences.
- *Inventory Management Service:*
  Tracks pantry items and their quantities, ensuring accurate inventory management and helps to facilitate informed meal planning and shopping list generation.

- *Shopping List Optimization:*
  Manages the generation and optimization of shopping lists based on user meal plans, pantry inventory, and current supermarket prices for the items.
- *User Profile Service:*
  Responsible for managing user profiles, including storing and retrieving user preferences, dietary restrictions, and allergies.
- *Jumbo Price Service:*
  Retrieves price information from the Jumbo supermarket, enabling cost-effective shopping list generation.
- *Albert Heijn Price Service:*
  Like the Jumbo Price Service, this service get the price information from the Albert Heijn supermarket. Getting prices of multiple supermarkets makes price comparison possible.

Architecture Explanation:

Because we have made a micro-service architecture, our solution exists out of several interconnected components. Each of these services is fulfilling a specific role in the meal planning application.

The user interface serves as the origin of all processes that will be started in the solution and as the only way for an user to interact with the application. To be able to represent the information to the user it needs to have access to data. This data is managed by multiple different other services. So the user interface has connections with some other services in order to start processes, retrieve data and update this date.

One of the services that the user interface will interact with is the meal planning service. This service will in turn act as a controller, like in an orchestration pattern. This service will orchestrate when and what the other services will have to do in order to generate the meal planning. This has to be done in order to fulfill the primary business process of the application: the meal planning. The planning is then stored in the associated database for retrieval by the user (interface).

Once there is a new meal planning available by the meal planning service, this is passed to the message queue from the shopping list optimization service. In this request all the required ingredients are sent that are still needed to complete the recipe.
Once this service has received this request to generate a shopping list it will go and retrieve the current prices of the connected supermarkets and generate a shopping list. The list will be optimized on the price. Such that the user will spend as little as possible.

There are some other services in play. These services and their respective relation with other services are represented in the figure below. In this diagram you can see that some of the services communicate synchronously and others asynchronous. Later on in this report we will clarify which ones and why we have chosen for this type of communication.

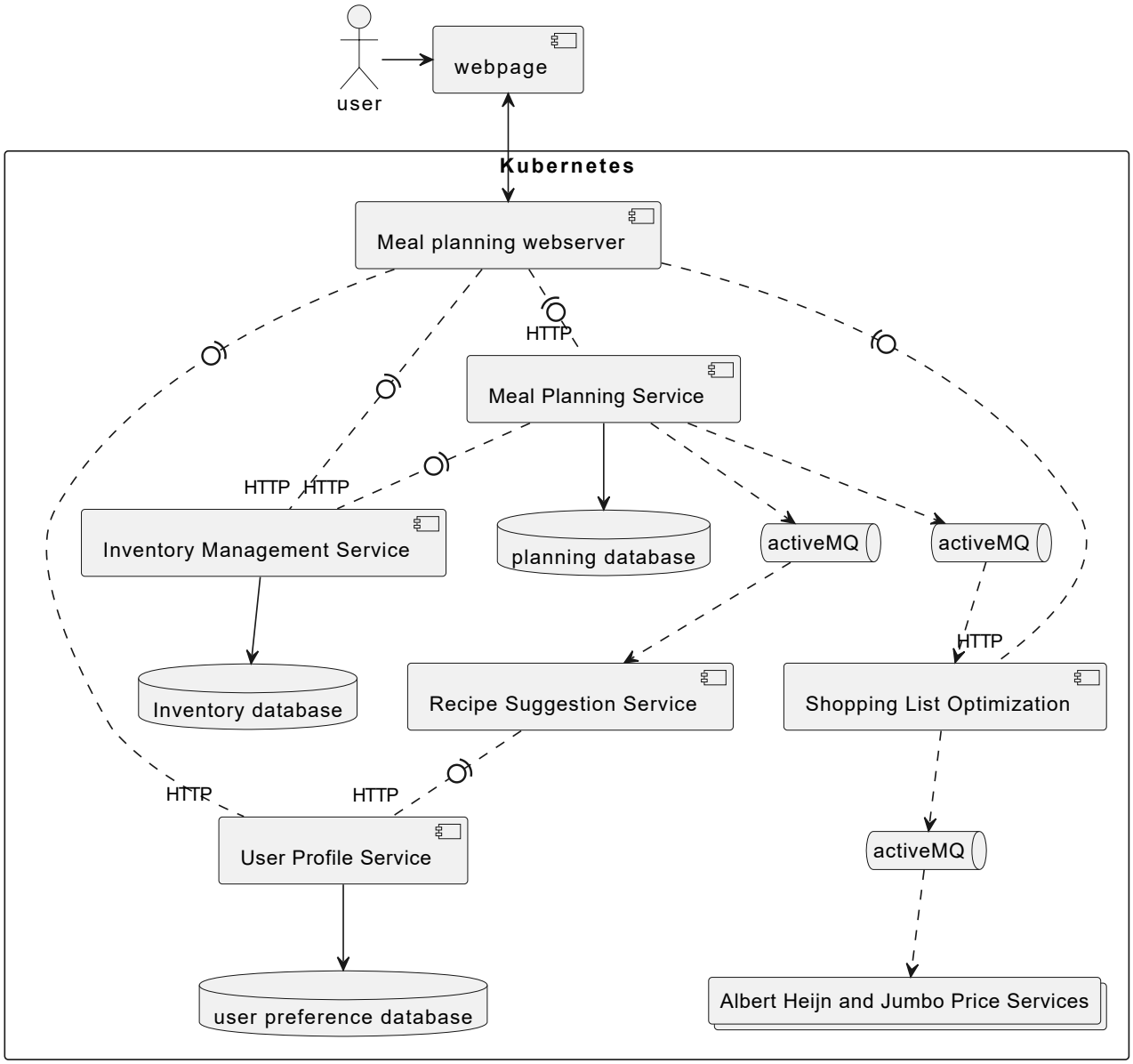

Figure 1: Component diagram

Synchronous vs. Asynchronous Services:

Synchronous Communication:

We opted for synchronous communication between all communication with the user profile and the inventory service. These two services are both services that do not have external dependencies and are only responsible for a small part of the data management of the application. We will not work with a dataset so big that this will even slow down the application to justify asynchronous communication. Most of the interactions with these services will be posting and retrieving data to be stored and doing minor alterations and transformations to this data.

The shopping list will be accessible via a message queue, but also with normal synchronous communication. We will argue why we have used a message queue later in this report. The synchronous part is only addressed by the user interface for retrieval of the shopping lists. As this is just a database query to retrieve any possibly generated list and can be done very quickly, we decided to use synchronous communication. This will also ensure efficient and responsive user interactions.

Asynchronous Communication:

In contrast, we employ asynchronous communication between the List Service and the Supermarket Service, as well as between the Planning Service and the Inventory and List services. This decision was driven by the desire to avoid blocking the application while fetching price information from third-party APIs and to streamline the meal planning process. By utilizing message queues, we can submit price inquiries in bulk and continue processing other tasks while awaiting responses, thereby enhancing the overall efficiency and responsiveness of our application.

In Contrast, we use asynchronous communication between the meal planning and the shopping list optimization service. Even though we have enabled synchronous communication between the user interface shopping list optimization service as mentioned above. When the meal service wants to communicate with the shopping list optimization service it does not require an immediate response. This is also not really possible as the shopping list service is dependent on external services that might delay a response.

The external services used by the shopping list optimization are all the price services of the supermarkets. For now there are two but this can become more in a possible future. These price services are directly communicating with API's of supermarkets and we assume these to be unreliable with regards to speed. For this reason we have also made this communication asynchronous. We can then submit multiple price inquiries and continue other processes.

The last service that uses asynchronous communication is the recipe suggestion service. We assume that our magic algorithm, AI or other method could take some time to generate the best suggestions. For now it will be very quick but in the future this could change.This could block other processes in the application. To prevent this we have made communication with this service asynchronous.

Justification of Message Queues:

We have chosen to implement message queues for all of our services that will use asynchronous communication. We have done this as all services have to perform tasks that take some time that could prevent the handling of new requests.

For instance the suggestion service could take a relatively long time to generate a new recipe. In that time there could be a new request to generate suggestions. To prevent this new request from being lost, we implemented a message queue when the meal planning communicates with the suggestion service. Then once the request is handled, the next task can be started. This is also the case for the price services. These services could potentially take more time as we cannot assume that the API of the supermarkets will respond quickly.

When the meal planning application sends a new list with ingredients to the shopping list optimization also with a message queue. This is done because we do not require a response for this request. And the price optimization service could potentially take a long time to finish the retrieval of all the prices. Also we do not want to lose the ingredient lists due to a HTTP timeout when this is not required in this case.

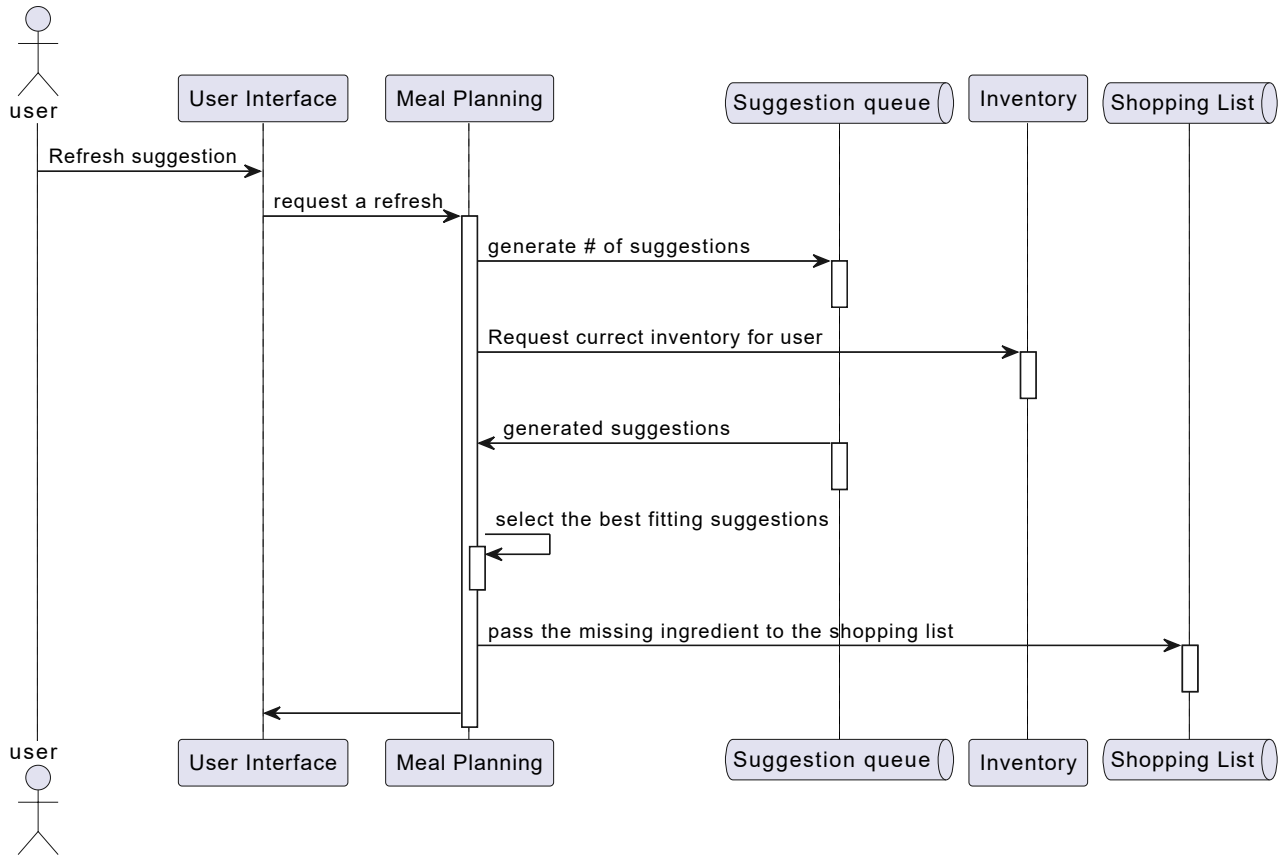

Figure 2: Sequence diagram

Design decisions

- The more concrete/specific design decisions, by explaining why you did (and didn't) use different technologies such as SOAP, REST and Message queue. Consider giving a summary table listing the services with the information about why you used (or did not use) these technologies and communication styles.

In the development of our Intelligent Recipe and Meal Planning Application, we made several concrete design decisions aimed at optimizing performance, scalability, and user experience. Each decision was carefully considered based on the specific requirements and characteristics of our application.

Synchronous Communication

One of our design decisions was to employ synchronous communication between the Meal Planning Service and the Inventory Management Service, as well as between all services and the User Profile Service. This choice was made due to the relatively low computational overhead of these interactions, as both services are operated internally and do not require extensive processing. By utilizing simple HTTP requests, we ensure straightforward and efficient communication between these components.

Asynchronous Communication with Message Queues

In contrast, we opted for asynchronous communication between the Shopping List Optimization Service and the Supermarket Services using message queues. This decision was motivated by the need to avoid blocking our application while fetching prices from third-party APIs, which could introduce delays and hinder responsiveness. By employing message queues, we can submit price inquiries in bulk and continue processing other tasks while awaiting responses. This approach enhances the overall efficiency and responsiveness of our application.

Currently, our implementation involves each supermarket having its own message queue, with a dedicated supermarket service listening to incoming price inquiries. Upon receiving a request, the service processes it and sends back a response containing the product ID, price, and store name, facilitating seamless integration with multiple supermarkets.

Asynchronous Communication with Websockets

Additionally, we chose to implement asynchronous communication between the Meal Planning Application and the Meal Planning Service using websockets. This decision was driven by the need for real-time updates and a responsive user interface, particularly during the generation of recipe suggestions, which may involve waiting periods. By utilizing websockets, we ensure that results are delivered promptly to the client, even in scenarios where processing times vary.

Our current implementation includes an endpoint for retrieving user preferences via HTTP GET requests and a user interface for viewing and editing preferences. Similarly, a websocket connection facilitates communication between the Meal Planning Application and the Meal Planning Service, allowing for seamless interaction and real-time updates.

Summary Table of Design Decisions

Service	**Communication Style**	**Reasoning**
Meal Planning & Inventory Management	Synchronous (HTTP)	Low computational overhead; Internal operations
All Services & User Profile Service	Synchronous (HTTP)	Internal operations; Querying data from own database
Shopping List Optimization & Supermarket Services	Asynchronous (Message Queue)	Avoid blocking application; Fetch prices from third-party APIs
Meal Planning Application & Meal Planning Service	Asynchronous (Websockets)	Real-time updates; Responsive user interface; Prompt delivery of results, even during processing

These design decisions were made with careful consideration of our application's requirements and objectives, aiming to optimize performance, enhance user experience, and facilitate seamless integration of services. As our implementation progresses, we remain open to refining these decisions based on evolving needs and feedback from users and stakeholders.

Validation

- Concrete evidence that you have validated your system, with testing and usage information.

Relevant information

- Any other relevant information/knowledge that is necessary to appreciate your efforts in this project. The report should be complete and detailed enough so that the teachers don't need to look into the code to understand what has been done.

Conclusion

In conclusion, the development of our Intelligent Recipe and Meal Planning Application represents a significant step forward in addressing the challenges associated with traditional meal planning methods. Through the implementation of a microservices architecture and thoughtful design decisions, we have created a versatile and user-centric solution that offers personalized recipe suggestions, efficient inventory management, and optimized shopping lists.

Throughout the project, we remained committed to our objectives of streamlining the meal planning process, enhancing user satisfaction, and promoting healthier eating habits. By leveraging advanced technologies such as websockets, message queues, and RESTful APIs, we have achieved a high level of performance, scalability, and responsiveness in our application.

Our validation processes, including rigorous testing and user feedback, have helped ensure the reliability, usability, and effectiveness of our solution. We have successfully demonstrated the feasibility and viability of our architecture, while also considering regulatory requirements such as GDPR compliance.

Looking ahead, there are opportunities for further enhancement and refinement of our application. Future iterations may involve expanding the range of services offered, integrating additional features such as nutritional analysis or meal sharing capabilities, and optimizing performance to accommodate growing user demand.

Overall, the Intelligent Recipe and Meal Planning Application represents a significant achievement in leveraging technology to simplify and enhance the culinary experience for users. We are confident that our

solution will continue to make a positive impact in the lives of users, helping them make more informed decisions about their meals and ultimately leading to healthier, more sustainable eating habits

Acknowledgements

In this project we have used ChatGPT 3.5 and 4.0 as a supportive tool, providing structure to our paper and aiding in the identification and correction of spelling mistakes. We did also use ChatGPT for debugging and understanding errors while we were developing the application.
It is important to note that all content generated by ChatGPT underwent thorough review and validation by us. We take full responsibility for the final output.