



République Tunisienne
الجمهورية التونسية
Ministère de l'Enseignement Supérieur et de la
Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
Direction générale des études technologiques
Institut Supérieur des Etudes Technologiques de CHARGUIA



End of studies project report

Submitted for the purpose of obtaining :

National Bachelor's Degree in Information Technology

Specialization : Information Systems Development

Design, Implementation and Deployment of «ShopFlow» : A Platform for Optimized Food and Groceries Management

Elaborated by :

M. MZOUGHJI Amine

M. ZAIRI Omar

Supervised by :

M. AJOUA Abdelghafour (ISET)

M. ZEDDINI Khaled (Zconsuling)

Host Company :

Z Consulting

Academic Year : 2023/2024

Dedication

“

We Dedicate this work to:

OUR Families, for their their support and their devotion.

OUR Friends, for always being there for us.

*OURSELVES For lifting each other and going through ups
and downs together,*

Thank You.

*we dedicate this project with a sense of
accomplishment, respect and pride.*

”

- Amine Mzoughi & Omar Zairi

Acknowledgements

First and foremost, we wish to express our gratitude to our teachers at Institut Supérieur des Etudes Technologiques (ISET) Charguia.

We extend our sincere thanks to our academic supervisor, Mister AJOUA Abdelghafour,

for his trust, constructive suggestions, constant commitment, and availability throughout this internship period.

We also wish to thank our company supervisor, Mister Zeddini Khaled, for his continuous support and encouragement.

Additionally, we wish to thank Madame Tayachi Ahlem for guiding us at the start of the internship.

We hope this report will reflect the entirety of our efforts and meet the expectations of all readers.

Contents

List of Figures	
List of Tables	
Acronyms	
General Introduction	1
Chapter 1. Project Overview	3
1 Presentation of the host organization	3
2 Presentation of the Business Domain	3
2.1 Evolution of GMS: From Traditional to Digital Transformation	3
3 Study of the existing	4
3.1 Description and Critique of the existing	4
3.2 Proposed solution « ShopFlow »	7
4 Methodology Choice	7
4.1 Comparison between different methodologies	8
4.2 Adapted methodology	9
Chapter 2. Project Planning	10
1 Requirements Specification	10
1.1 Identification of actors	10
1.2 Functional requirements	11
1.3 Feature-based global Use Case diagram	12
1.4 Non-Functional Requirements	13
2 The product Backlog	13
2.1 MoSCoW Prioritization	14
2.2 Definition of Done	14
2.3 ShopFlow Product Backlog	14
3 Development Environment	19
3.1 Backend Development Environment	19
3.2 Frontend Development Environment	20
3.3 Other Tools	21
4 System Architecture	24
4.1 Monolith vs. Microservice	25
4.2 Choice Justification	26
4.3 Service Decomposition by business capability pattern	27
4.4 Database-per-service pattern	27

4.5	Mobile Application Design Pattern	28
5	Applied Technologies	30
5.1	Containerization	30
5.2	Container Orchestration with Kubernetes	30
5.3	DevOps	33
5.4	Continuous integration (CI), Continuous delivery/deployment (CD)	34
5.5	Machine Learning	35
5.6	Business Intelligence	36
6	Cloud Computing	36
6.1	Reasons to use cloud-based infrastructure	37
6.2	Choice of Cloud Provider: Microsoft Azure	37
7	The release planning	38

Chapter 3. Sprint 1 : Authentication, Inventory Management & Household Collaboration 40

1	Identification of needs	40
1.1	Sprint One Backlog	40
1.2	Sprint 1 Use Case Diagram	44
1.3	User Interface Mock-ups	44
2	Design	46
2.1	Class Diagram	46
2.2	MongoDB Collection	47
2.3	Sequence Diagrams	48
3	Tests	52
4	Deployment of the first increment	52
5	Sprint Review	52
5.1	Main User Interfaces	52
6	Sprint Retrospective	53
6.1	Retrospective Table	54
6.2	Sprint Burn Down Chart	54
6.3	MicroService Inter-Communication	54
6.4	DoD Compliance Assessment	55

Chapter 4. Sprint 2 : GMS Authentication, ShoppingListManagement & RecommendationSystem 56

1	Identification of needs	56
1.1	Sprint Two Backlog	56
1.2	Identification of actors	62
1.3	User Interface Mock-ups	62
2	Inter-Communication	63
2.1	IPC Technologies	63
2.2	Overview of Broker-Based Messaging	63
2.3	RabbitMQ	63
3	Recipe Suggestions	64
3.1	Recommender system	64
3.2	Content-Based Recommender Systems	64

4	GMS Application Prototype	65
5	Design	65
5.1	Class Diagram	65
5.2	Object Sequence Diagram «Add CartItem»	68
5.3	Deployment Diagram	68
6	Tests	70
7	Deployment of the Second Increment to the Cloud	70
7.1	Setting Up the infrastructure in Azure	70
7.2	Production CI/CD Pipeline	71
8	Sprint Review	71
8.1	Main Interfaces	71
8.2	Backlog Update	75
9	Sprint Retrospective	75
9.1	Retrospective Table	75
9.2	Sprint Burn Down Chart	75
9.3	DoD Compliance Assessment	76
Chapter 5. Sprint 3 : GMS Management & Checkout Interactions		77
1	Identification of needs	77
1.1	Sprint Three Backlog	77
1.2	Use Case Diagram	82
1.3	User Interface Mock-ups	82
2	Design	83
2.1	Class Diagram	84
2.2	Sequence Diagram «Cashier Starting Session»	85
2.3	Activity Diagram	86
2.4	Package Diagram	87
3	Consumption Trends Chart	88
4	Tests	89
5	Deployment of the third increment	89
6	Sprint Review	89
6.1	Main Interfaces	89
6.2	ShopFlow Admin Interfaces	91
6.3	User Interface	92
7	Sprint Retrospective	92
7.1	Retrospective Table	92
7.2	Sprint Burn Down Chart	93
7.3	Transition to API-Gateway Level Authentication	93
7.4	DoD Compliance Assessment	94
Chapter 6. Sprint 4 : GMS Management & Checkout Interactions		95
1	Identification of needs	95
1.1	Sprint Four Backlog	95
1.2	Identification of actors	98
1.3	User Interface Mock-ups	98
2	API-Gateway Level Authentication	99

2.1	Basic Authentication	99
2.2	Token based Authentication	99
3	Product Suggestion	100
3.1	Neighborhood-based Collaborative Filtering Recommender Systems	100
4	Design	101
4.1	Class Diagram	101
4.2	Sequence Diagrams	103
5	Tests	105
6	Deployment of the fourth increment	105
6.1	System Monitoring	105
7	Sprint Review	106
7.1	Main Interfaces	106
8	Sprint Retrospective	108
8.1	Retrospective Table	108
8.2	DoD Compliance Assessment	108
	General Conclusion	109
	Bibliography and Netnography	110
	Appendices	112
	Appendix A Chapter 2 Additional Content	113
	Appendix B Sprint 1 Additional Content	114
	Appendix C Sprint 2 Additional Content	129
	Appendix D Sprint 3 Additional Content	144
	Appendix E Sprint 4 Additional Content	158

List of Figures

1.1	Amazon Fresh « Dash Cart » [4]	5
1.2	Logo BEEP	6
1.3	Main BEEP interfaces	6
2.4	static context diagram	11
2.5	Feature-based Global Use Case Diagram	13
2.6	Microservice vs Monolith [8]	25
2.7	Microservice Architecture	26
2.8	GetX Architecture	29
2.9	Mobile Project Structure	29
2.10	Illustration of a Container[13]	30
2.11	Container Orchestration Concept[14]	31
2.12	Kubernetes Architecture	33
2.13	DevOps	34
2.14	Continuous Integration, Delivery and Deployment[20]	35
2.15	ETL Process	36
3.16	Sprint 1 Use Case Diagram	44
3.17	Verification code interface mock-up	45
3.18	Scan A Product interface mock-up	45
3.19	Product Details interface mock-up	45
3.20	User Inventory interface mock-up	45
3.21	Class diagram Sprint 1	46
3.22	MongoDB Collections Schemes	47
3.23	Object sequence diagram «Sign Up» «Backend Exclusive»	48
3.24	System sequence diagram «Authentication»	49
3.25	Design sequence diagram «Scan a Product»	50
3.26	Design Sequence Diagram «Get Users Inventories»	51
3.27	Welcome Screen Light Mode	52
3.28	Home Page Light Mode	52
3.29	Edit Account Interface	52
3.30	Invite Members Interface	52
3.31	Inventory Interface	53
3.32	Scan a product interface	53
3.33	Product Details interface	53
3.34	Sprint 1 Burn Down Chart	54
4.35	Household Interface mock-up	62
4.36	Add to cart Interface mock-up	62
4.37	Recipe Suggestions Interface mock-up	62
4.38	Class Diagram Sprint 2	66

List of Figures

4.39	Recipe Recommendation Class Diagram Sprint 2	67
4.40	GMS Class Diagram Sprint 2	67
4.41	Add CartItem Object Sequence Diagram	68
4.42	Deployment Diagram Sprint 2	69
4.43	Complete Cloud Infrastructure	70
4.44	Complete Production CI/CD Pipeline	71
4.45	Entered Verification Code For Password Recovery	72
4.46	Household	72
4.47	Recipe Recommendation List	72
4.48	Create a shopping List	73
4.49	Add product to Shopping List	73
4.50	Shopping Lists Interface	73
4.51	GMS Login	73
4.52	GMS Signup	74
4.53	GMS Google Sign up/ Login in	74
4.54	Sprint 2 Burn Down Chart	76
5.55	Sprint 3 Use Case Diagram	82
5.58	Cashier Scanned Products List	82
5.56	Final Price With Qr code of a user's shopping List Mock-up	83
5.57	Inventory Product Notes Mock-up	83
5.59	Employee Status Table	83
5.60	Class Diagram Sprint 3	84
5.61	GMS Class Diagram Sprint 3	85
5.62	Sequence Diagram «Cashier Starting Session»	85
5.63	Checkout Interaction Activity Diagram	86
5.64	Package Diagram Sprint 3	87
5.65	Consumption Trends Chart	88
5.66	Approve an Employee	89
5.67	Stock List Interface	89
5.68	Promotional Stock List Interface	90
5.69	Start Checkout Session	90
5.70	Scanning User's Shopping List QR code	90
5.71	List of Products On Cashier Dashboard	91
5.72	Supermarket Lists Interface	91
5.73	Add a Supermarket Form Interface	91
5.74	Attach Note Interface	92
5.75	Final Price for Shopping List	92
5.76	Sprint 3 Burn Down Chart	93
6.77	GMS Manager dashboard mock-up	99
6.78	User's purchases mock-up	99
6.79	Ingress Authentication	100
6.80	User Based Collaborative Filtering	101
6.81	Class diagram Sprint 4	102
6.82	Design Sequence diagram «Get products recommendations»	103
6.83	Design Sequence diagram «Add a product to Inventory»	104
6.84	Object Sequence Diagram «Notifying of Inventory Product Addition» . . .	104

6.85	How AKS Prometheus and Grafana work	105
6.86	An example of our Grafana dashboard	105
6.87	an example of our Grafana Alerts	106
6.88	User Purchases Interface	106
6.89	Household consumption statistics	106
6.90	Products recommendations interface	106
6.91	GMS Dashboard Interface first section	107
6.92	GMS Dashboard Interface second section	107
A.1	A Pod template wrapped in a Deployment [17]	113
B.1	Login interface mock-up	126
B.2	Settings interface mock-up	126
B.3	Create account interface mock-up	126
B.4	Select interests interface mock-up	126
B.5	Get Inventory By ID unit Test	126
B.6	Get Product By Name unit Test	127
B.7	Complete CI/CD pipeline succeeded following a commit in the dev branch of the Household Management service	127
B.8	Complete CI/CD pipeline failed following a commit in the dev branch of the Notifications service	127
B.9	Details page for the completed CI/CD pipeline following a commit in the dev branch of the Notifications service	127
B.10	Account Settings Interface	128
B.11	Inventory Product Details	128
B.12	Text Recognition Results	128
B.13	Inventory Product Management	128
C.1	Design Sequence Diagram «Get Users Inventories»	129
C.2	Get Shopping List By ID unit Test	143
C.3	Get Employees List unit Test	143
D.1	Get Stock List unit Test	156
D.2	Get Supermarket By ID unit Test	156
D.3	Deploying of ShoppingList Management microservice	157
D.4	Deploying of Inventory Management microservice	157
E.1	Get Purchase List unit Test	166
E.2	Deploying of Supermarket Analysis microservice	167
E.3	Deploying the Product Recommendation microservice	167
E.4	CPU Usage of our Kubernetes Pods	167
E.5	Bandwidth for our Kubernetes components classified by Namespaces	168
E.6	Average Bandwidth for the profile management Pods	168

List of Tables

1.1	Comparison between different methodologies	8
2.2	ShopFlow Product Backlog	19
2.3	Backend Development Environment	20
2.4	Frontend Development Environment	21
2.5	Tools Used in Project Development	24
2.6	Monolith vs MicroService[8]	26
2.7	the Dev and Ops teams preferences	34
2.8	Comparison Between managed Kubernetes services [24]	38
3.10	Sprint 1 backlog snippet	44
3.11	Story acceptance criteria	45
3.12	Business rules of sprint 1 class diagram	47
3.13	Evaluation of Project	54
4.14	Sprint 2 backlog snippet	61
4.15	Identification of actors Sprint 2	62
4.16	conceptual goals of Content-based methods	65
4.17	Business rules of sprint 2 class diagram	67
4.18	Backlog Update	75
4.19	Evaluation of Sprint	75
5.20	Sprint 3 backlog snippet	81
5.22	Evaluation of Sprint 3	93
6.23	Sprint 4 backlog snippet	98
6.24	Identification of actors Sprint 4	98
6.25	conceptual goals of User-based collaborative filtering methods	101
6.26	Business rules of sprint 4 class diagram	102
6.27	Evaluation of Sprint 4	108
B.1	First sprint full backlog	125
C.1	Second Sprint backlog	142
D.1	Third Sprint Backlog	156
E.1	Fourth Sprint Backlog	166

Acronymes

AWS	<i>Amazon Web Services</i>
ADEME	<i>Agence de l'Environnement et de la Maîtrise de l'Énergie</i>
AKS	<i>Azure Kubernetes Services</i>
BI	<i>Business Intelligence</i>
CD	<i>Continuous delivery/deployment</i>
CI	<i>Continuous integration</i>
DoD	<i>Definition of Done</i>
EDC	<i>Expiry Date for Consumption</i>
EKS	<i>Elastic Kubernetes Service</i>
ETL	<i>Extract, Transform, Load</i>
GKE	<i>Google Kubernetes Engine</i>
GMS	<i>Grand and Medium-sized Surfaces</i>
IPC	<i>Inter Process Communication</i>
ML	<i>Machine Learning</i>
SLA	<i>Service Level Agreement</i>
UI	<i>User Interface</i>

General Introduction

According to ADEME, about 10 million tons of consumable food are wasted yearly. In household and similar waste, there are approximately 20 kg of food waste per person per year, including 7 kg of still-packaged food products. [1]

Food waste inevitably leads to money waste. In France, the cost is estimated to be in the range of 12 to 20 billion euros per year – approximately about 159 euros/person.

According to a study conducted by OpinionWay[2] in 2023, it is found that 29 % of the French population admits to discarding food due to approaching or expired EDC.

The awareness of the impact of food production on the environment, a growing ethical question, and an economic reality that has a major impact on family budgets are leading us to modify and optimize our purchasing practices. At the same time, GMS must adapt to the ongoing profound transformation of global agri-food production by: relocation of production, rising transportation costs, customer volatility, and decreasing purchasing power.

In this context, the interests of the two actors can align. Consumers will have to subtly manage their purchases and Distributors must gain a thorough understanding of the rapid qualitative and quantitative shifts in consumer behavior.

Harmonizing these two requirements might appear daunting; nevertheless, applying information technology can offer viable solutions. For the distributors (GMS), the emphasis lies in acquiring the most current data generated in proximity to the client. For the consumer, there is a potential benefit in having their personal data processed. This includes details on purchases, stock levels with expiration dates, suggested products based on dietary habits, understanding evolving preferences, considering social practices, and ultimately gaining better control over expenses and budgets.

Z-Consulting, our host company, suggested a solution: a platform that connects consumers and distributors. The platform collects real-time data on purchases and users' habits, offering personalized recommendations to reduce waste and save money, better management for a user's own products through an inventory system, and easier checkout at a supermarket. The solution will be even more effective with increased user awareness of food waste.

For distributors, it provides insights into consumer behavior, allowing for optimized

stock levels by generating valuable statistics to better understand what is trending among all users of the platform. this alignment supports a more sustainable and economical food ecosystem, addressing food waste and promoting ethical and environmental practices.

This platform will feature user-friendly interfaces designed to be accessible and easy to use for individuals of all ages. Scalability is also crucial for ensuring the platform remains up to date and well-maintained.

This report, which is separated into six chapters, details the many procedures that were taken to complete this project. Firstly, we will provide an Overview of the Project to further precise the proposed solution. Second, we will dive into the Project's Planning where we present the functional and the non-functional requirements. We will also present our adopted methodology, the platform's architecture and some concepts used for the development of this project. And lastly, we will present the four chapters « Sprint 1 », « Sprint 2 », « Sprint 3 » and « Sprint 4 » in which we will detail their functional specifications through the sprint backlog, their conception, and their implementation.

Chapter 1. Project Overview

In this chapter, we establish the project within its broader context. It is primarily dedicated to introducing *Z Consulting*, the hosting organization where we conducted our final year project. We then delve into the existing conditions, followed by our proposed solution. Lastly, we elaborate on the work methodology we employed.

1 Presentation of the host organization

Z Consulting[3] is a company based in France, standing out as a leading partner in the field of IT solutions. Beyond its role as a consulting firm, it positions itself as a key player in facilitating digital transformations, designing innovative solutions, and committing to the success of its clients. Guided by technical excellence, continuous innovation, and close collaboration, Z Consulting has solidified its reputation as a technological leader.

Z Consulting's expertise spans various strategic domains, contributing to the development of a robust and high-performance digital landscape for its clients, including :

- DevOps
- Data
- Production Engineering
- Application Development
- Web and Mobile Development [3]

2 Presentation of the Business Domain

In this section, We will delve into key concepts essential for a thorough understanding of this project.

2.1 Evolution of GMS: From Traditional to Digital Transformation

In the early days, Grand and Medium-sized Surfaces (GMS) operated much like any other traditional retail stores. They were physical locations where customers could walk

in, browse through the aisles, and purchase groceries directly off the shelves. The focus was primarily on maintaining inventory, managing cash registers, and ensuring a pleasant in-store experience for customers.

With the advent of the internet and improved data processing capabilities, GMS began to explore more sophisticated digital solutions. Online shopping platforms emerged, allowing customers to browse and purchase products from the comfort of their homes. This marked a pivotal shift from purely physical stores to a blend of online retailing.

The digital transformation did not stop there. As GMS integrated more technology into their operations, they started generating vast amounts of data. Every transaction, customer interaction, and inventory movement was recorded and stored. This data became a valuable asset, leading to the rise of data analytics and business intelligence in retail.

Data analytics enabled GMS to gain deeper insights into customer behavior, preferences, and trends. They could now personalize marketing efforts and enhance the overall customer experience. Predictive analytics allowed for better demand forecasting, reducing overstock and stockouts.

Moreover, the integration of business intelligence tools provided GMS with real-time dashboards and reports, facilitating data-driven decision-making. This improved operational efficiency and strategic planning, giving GMS a competitive edge in the market.

It is important to note that the data collected by GMS is limited to their customers and pertains solely to their shoppers. GMS does not have access to broader user preferences outside its customer base. This limitation impacts their ability to fully leverage user preference data for broader market analysis.

3 Study of the existing

In this section, we'll provide a description of the existing, critique it, and then present our proposed solution.

3.1 Description and Critique of the existing

Currently, the grocery shopping process takes several time-consuming steps. In a typical scenario, the consumer goes to a supermarket(GMS) to choose the items they need. After making the purchases, they must wait in line for the cashier to individually scan each product. Following this step, the consumer receives their receipts, completing the purchase process.

Home delivery and 'Smart Shopping' are solutions designed to save time during grocery shopping. Various companies provide these services, with one of the most prominent being.

3.1.1 Amazon Fresh

Imagine a supermarket where you can :



Figure 1.1: Amazon Fresh « Dash Cart » [4]

- **Shop and exit without needing to visit the cashier :** thanks to carts equipped with automatic scanning for your purchases.
- **Take a relaxed stroll through the aisles with your baskets and receipt:** just like in a regular store.

Amazon Fresh provides a unique experience by combining a traditional supermarket with advanced technology. Yet, like any innovation, it has its downsides :

- **Not yet available everywhere :** Amazon Fresh is still not widely established in France.
- **No personalization for users :** Amazon Fresh is a standardized service that does not take into account the individual needs and preferences of customers.
- **Lack of inventory management :** Amazon Fresh doesn't let users keep tabs on their home food supplies, which could result in buying duplicates or expired items.
- **Limited food choices :** The options may not always be the same as in a regular supermarket, sometimes requiring compromises on certain fresh or specialized products.

Meanwhile, inventory management also presents significant challenges. Once products are stored with the consumer, especially in the refrigerator or other spaces, the crucial issue of the EDC arises. Consumers can easily forget this essential information, leading to food waste when products exceed their EDC without being consumed. There are solutions on the market, such as the application:

3.1.2 BEEP

BEEP allows you to manage EDCs with your smartphone. Accessible anywhere, it provides reminders for products nearing expiration. [5] However, this service has shortcomings:

- **Paid:** BEEP requires a subscription, which may limit its accessibility.

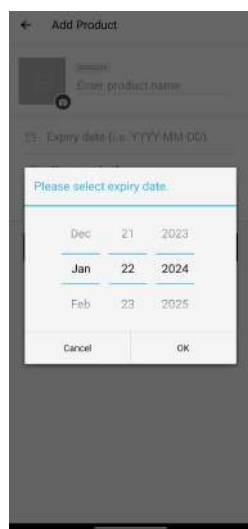


Figure 1.2: Logo BEEP

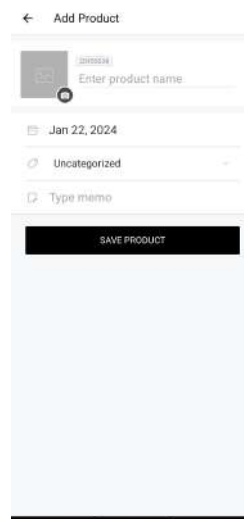
- **Manual entry:** Users need to manually enter product information, unlike automatic recognition.
- **Perfectible interface:** Some users find the design not very intuitive.
- **Restricted connection options:** No email connection option, only Facebook and Google.
- **Synchronization issues:** Some users have reported synchronization issues between different devices using BEEP.
- **No product categorization:** It is difficult to find specific products in the application as there is no categorization system or filters.



(a) Authentication page



(b) Selection of product's EDC



(c) Add a Product



(d) Products list

Figure 1.3: Main BEEP interfaces

Additionally, there remains a significant gap in the data insights available to GMS. This gap hinders their ability to fully understand and cater to their customer base effectively. Specifically:

3.1.3 Lack of Insights into Consumer Data

- **Data Limited To Their Own Customer Base:** GMS systems are often limited to analyzing data from their customers. This narrow focus can prevent them from understanding broader market trends and potential new customer segments.
- **Targeted Audience Understanding:** GMS systems often lack detailed insights into consumer preferences, behaviors, and purchasing patterns, which are crucial for tailoring offerings and marketing strategies.
- **Trend Analysis:** Without data on trending products, GMS systems may fail to stock popular items, leading to missed sales and dissatisfied customers.

Competitor Data

- **Market Positioning:** Without competitor insights, GMS systems struggle to position themselves effectively, resulting in missed opportunities to attract customers and stand out.

3.2 Proposed solution « ShopFlow »

« ShopFlow » is an innovative platform designed to enhance the grocery shopping experience and improve inventory management for both consumers and GMS.

For consumers, ShopFlow offers easy tracking of household food inventory, notifications for products nearing expiration, recipe recommendations to utilize these items, and efficient shopping list management.

For supermarkets, instead of solely focusing on insights about their existing customers, ShopFlow gathers data from all users to generate valuable insights into consumer habits, helping optimize stock levels and enhance marketing strategies.

The platform also streamlines the checkout process by generating a QR code that contains all cart information, which is then scanned at the register to quickly retrieve product data. This real-time tracking provides GMS with up-to-date inventory visibility, enabling them to maintain optimal stock levels, reduce food waste, and ensure popular items are always available.

Overall, ShopFlow creates a unique easy shopping experience for consumers while empowering GMS with the insights needed to enhance efficiency and meet consumer demands.

4 Methodology Choice

This section aims to provide a comprehensive understanding of the adopted project management approach and its alignment with the project requirements.

4.1 Comparison between different methodologies

In this section, we compare various popular methodologies.

Criteria	Scrum	2TUP	Waterfall
Project Management Style	- Agile	- Iterative	- Sequential project management
Project Requirements	- Product Backlog - Definition of done - Sprint Review - Sprint Retrospective	- Alignment with goals	- Requirements Analysis and Documentation - Complete Requirements Before Development - Limited Customer Involvement
Project Scope	- Defined in the Product Backlog - Can be changed during each sprint	- Scope adjusts based on feedback - Incremental value delivery	- Fixed
Team Size	- 3-9 members	- Small, focused teams	- Larger teams
Project Visibility and Communication	- Daily Scrum meetings	- Frequent communication - Open feedback channels	- Documents - Periodic status reports
Risk Management	- Assesses and manages risks during each sprint	- Adaptability to mitigate risks - Challenges in early risk prediction	- Comprehensive risk assessment at the beginning - Limited adjustments during development
Adaptability to requirement changes	- Changes through backlog	- Flexibility for evolving requirements	- Resisted or Implemented thorough evaluation
Project Prioritization	- Product Backlog with input from stakeholders - Adjusted at the start of each sprint	- Adjustment of focus based on user needs - Balancing discovery and delivery priorities	- Fixed prioritization at the beginning - changes are challenging during development
Performance Measurement	- Velocity - Burndown charts	- Measurement of both discovery and delivery outcomes	- Progress is measured against the project plan
Resource Management	- Self-organizing teams - roles like Scrum Master	- Balancing resources between discovery and delivery	- Clearly defined roles - Specific responsibilities

Table 1.1: Comparison between different methodologies

4.2 Adapted methodology

We chose Scrum as our project management framework due to its alignment with the evolving requirements of our business. Guided by Mr. Khaled, our Product Owner, Scrum's division of the project into feature sets enhances teamwork and collaboration. The time-boxed iterations ensure effective time management, allowing us to deliver incremental value at each sprint. The Sprint Review at the end of each iteration facilitates continuous improvement, ensuring project quality. Our team's familiarity with Scrum, gained from prior experiences, expedites our adaptation. Additionally, our commitment to regular communication through daily or bi-weekly meetings underscores our dedication to transparency and responsiveness. We also used some concepts from other methodologies including DevOps for Continuous integration/Continuous delivery(CI/CD).In summary, Scrum's flexibility, iterative nature, and proven track record make it an ideal choice for our project.

In this chapter, we introduced our host organization "Z-Consulting," followed by an overview of the business domain. We also conducted a study of the existing. Finally, we discussed the methodology chosen for this project. The next chapter will be dedicated to defining the features to be developed and planning their implementation.

Chapter 2. Project Planning

In this second chapter, our focus shifts to a comprehensive analysis of the project's requirements. We start by defining them, whether they are functional or non-functional. Next, we delve into the Product Backlog, detailing key features. The use case diagram provides an optional view of interactions, followed by the development environment and technical choices. Finally, we explore the system architecture.

1 Requirements Specification

This section's main focus will be to cover the functional analysis including actor identification, and functional and non-functional requirements. It sets the stage by analyzing project functions, identifying key players, and outlining both functional and non-functional needs.

1.1 Identification of actors

Every interactive system must ensure and facilitate interaction with its users. An actor represents an external entity that interacts with the system through its various interfaces. For this project, we identify the relevant system actors, distributed as follows :

- **Main Actors:**

1. User
2. GMS
3. ShopFlow Administrator

The static context diagram presented in Figure 2.4 illustrates a global view of the main actors involved in our applications. Here is a detailed overview of our platform :

- **ShopFlow:** This is the primary application encompassing the majority of the functionalities.
- **GMS:** This is the supermarket application, independent of our mobile app, and can be developed using any technology.

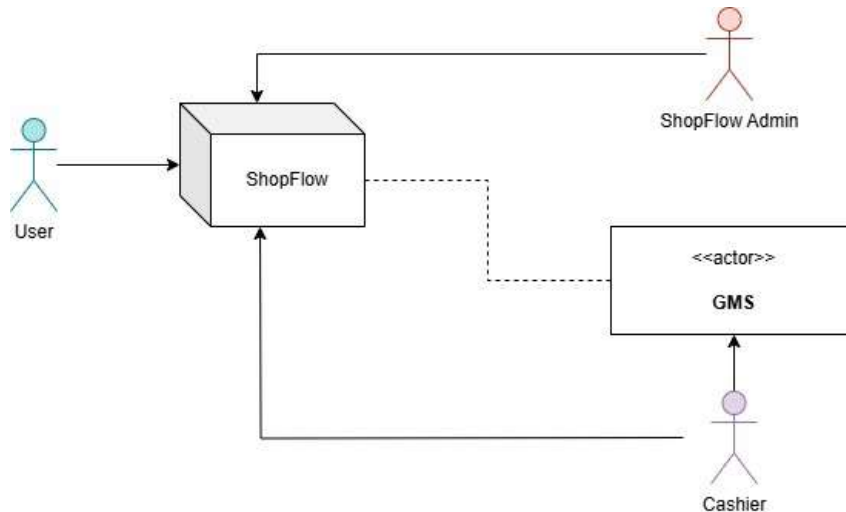


Figure 2.4: static context diagram

1.2 Functional requirements

In the following section, we will describe the functional requirements of our system.

- **Profile management**

- Consulting and editing profile information (name, address, allergies, etc).
- Adding dietary preferences (vegetarian, gluten-free, etc).

- **Product data acquisition**

- Scanning food products (QR code, barcode, character recognition, etc.).
- Manually inputting product details.

- **Inventory Management**

- Viewing the complete list of products in one's inventory.
- Sharing products with household members.
- Adding, Deleting or editing a product in one's inventaire.
- Receiving notifications for products approaching their expiration date.
- Filtering and sorting products in one's inventory (by categories, expiration date, etc.).
- Attaching notes to products in one's inventory.

- **Household Management**

- Inviting family members
- Tracking household members' purchases.
- Collaborative addition, Editing, and deletion of products.

- **Recommendation System**

- Receiving product suggestions based on dietary habits and products available in one's inventory.
- Receiving recipe suggestions based on available products in one's inventory.
- **Shopping List Management**
 - Creating shopping lists for different supermarkets.
 - Adding products to a shopping list from inventory or suggestions.
 - Marking products as purchased in a shopping list.
 - Synchronizing shopping lists with GMS.
- **History and statistics**
 - Reviewing purchase history and inventory additions.
 - Obtaining statistics on food consumption (categories, expenses, etc.).
- **Checkout Interactions**
 - Scanning the consumer's app QR code to retrieve data for all products.
 - Displaying the final price and completing the transaction.
- **Stock Tracking**
 - Updating stocks in real-time based on consumer purchases.
 - Receiving out-of-stock alerts.
 - Analyzing consumption trends to optimize stocks.
- **Statistics and analysis**
 - Accessing statistics on consumer purchasing habits.
 - Visualize consumption trends by product category.
- **Management of promotional offers**
 - Creating and managing promotional offer campaigns.
 - Targeting promotional offers based on consumer segments.
 - Tracking the effectiveness of promotional offer campaigns.
- **Supermarkets Management**
 - Managing the supermarkets and partners of the ShopFlow system.

1.3 Feature-based global Use Case diagram

Following the functional requirements section, Figure 2.5 provides a high-level overview of the system's functionality. To further illustrate how users interact with the system's features.

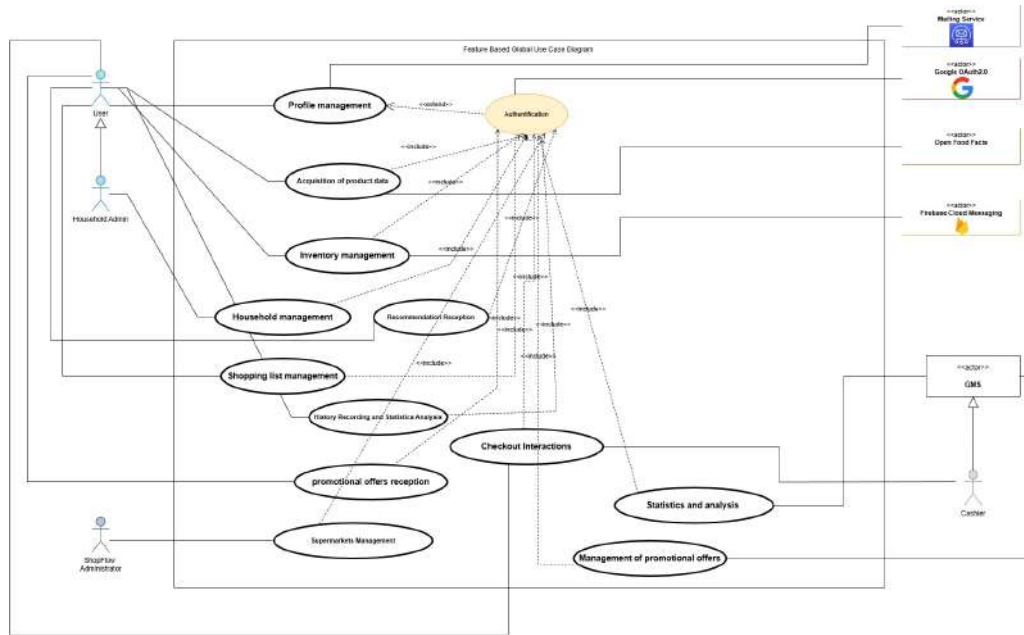


Figure 2.5: Feature-based Global Use Case Diagram

1.4 Non-Functional Requirements

Non-functional requirements are implicit demands that the system must meet. In our case, we have identified the following:

Portability : Our application must be compatible with ANDROID and IOS.

Data security: Our application must ensure security to guarantee the protection of user data.

Traceability : To generate accurate statistical data on application usage, it's essential for the solution to meticulously track each action and its source.

Performance: The response time must be minimized, ensuring non-blocking execution for time-intensive tasks to maintain a smooth user experience.

User Experience(UX): To satisfy users, this constraint is unavoidable. Therefore, the application interface must be simple, user-friendly, and easy to navigate.

Maintainability: Our project must adhere to architectural standards for easy maintenance and to ensure system scalability in case of additional requirements.

Scalability : The application's design should facilitate easy scalability to manage an increase in the number of users and products.

2 The product Backlog

In this segment, we will explain the concept of MoSCoW Prioritization, the definition of done and later present the product backlog where we will explore all the user stories and

features we need for our project.

2.1 MoSCoW Prioritization

“MoSCoW”[6], is a mnemonic for Must have, Should have, Could have, Won’t have. It is a useful method of partitioning proposed requirements into categories.

2.2 Definition of Done

The Definition of Done[7] includes all of the characteristics and standards an Increment needs to meet in order to be released. Our definition of Done includes finishing all user stories for the corresponding sprint. The following criteria must be met for the story to be considered done:

- Finishing the development.
- Completing various tests and having them accepted by the Product Owner and the developers.
- Deploying the validated increment into the cloud.

Given our limited resources, we will conduct testing and development locally. Only at the end of each sprint will we deploy the user stories to the cloud.

2.3 ShopFlow Product Backlog

Below is the product backlog of our projects, it highlights the different features and user stories.

ID is unique for the user story in question.

Business Value is the perceived worth or benefit to the business.

Priority defines the order of execution by the business value, MoSCow, and the functional dependency.

Complexity is a value that estimates the complexity using the Fibonacci sequence.
2 (easy), 5 (simple), 8 (quite complex), 13 (complex), 21 (extremely difficult)

MoSCoW :

- **MUST HAVE** : Non-negotiable product needs that are mandatory for the team
- **SHOULD HAVE**: Important initiatives that are not vital, but add significant value
- **COULD HAVE**: Nice to have initiatives that will have a small impact if left out
- **WILL NOT HAVE**: Initiatives that are not priority for this specific time

Feature ID	Feature	ID	User Story	Business Value	Priority	Complexity	MoSCoW
1	Profile management	1.1	As a user, I want to create an account using different signup methods	High	1	8	MUST
		1.2	As a user, I want to modify my personal information for better personalization	High	2	2	MUST
		1.3	As a user, I want to recover my password in case I forget it	Medium	2	5	SHOULD
		1.4	As a user, I want to login using different login methods	High	1	5	MUST
		1.5	As a user, I want to logout	Medium	2	2	SHOULD
2	Acquisition of product data	2.1	As a user, I want to scan products using Bar-codes	High	1	8	MUST
		2.2	As a user, I want to scan products using a QR Code	High	1	8	MUST
		2.3	As a user, I want to add products using character recognition	High	1	8	MUST
		2.4	As a user, I want to manually add a product in case the other methods malfunction	High	1	5	MUST
		3.1	As a user, I want to visualize my current inventory	High	2	2	MUST
		3.2	As a user, I want to receive a notification when a product in my inventory is soon to expire	High	2	8	MUST
		3.3	As a user, I want to attach notes to a product in my inventory to let other users in my household access it	Low	3	5	COULD

3	Inventory management	3.4	As a user, I want to add products to my inventory	High	2	2	MUST
		3.5	As a user, I want to remove products from my inventory	High	2	2	MUST
		3.6	As a user, I want to modify product information in case I made a mistake while adding a product manually	High	3	5	MUST
		3.7	As a user, I want to filter my inventory products	Medium	3	5	SHOULD
4	Household management	4.1	As a user, I want to invite other users to my household to share inventories	High	3	13	MUST
		4.2	As a user, I want to track the purchases of my household members	High	3	8	MUST
		4.3	As a user, I want to collaboratively add products with other members	High	3	5	MUST
		4.4	As a user, I want to collaboratively remove products with other members	High	3	2	MUST
		4.5	As a user, I want to collaboratively modify products with other members	Medium	3	2	MUST
		4.6	As a household admin, I want to manage the members	Medium	3	8	SHOULD
5	Recommendation System	5.1	As a user, I want to receive product suggestions based on my habits, budget, and products available in inventory	Low	4	21	SHOULD
		5.2	As a user, I want to receive recipe suggestions based on available inventory products.	High	4	21	MUST

6	Shopping list management	6.1	As a user, I want to create different shopping carts for different super-markets	Medium	5	8	MUST
		6.2	As a user, I want to add products to my shopping cart from my previous inventory	Medium	5	8	SHOULD
		6.3	As a user, I want to generate a QR-Code of my shopping list	High	5	8	MUST
		6.4	As a user, I want to add products to my shopping cart using the methods featured in Feature2	Medium	5	5	MUST
		6.5	As a user, I want to mark products in shopping cart as bought so they get added automatically to my inventory	Medium	6	8	COULD
		6.6	As a user, I want to share shopping carts with household members	Medium	5	8	SHOULD
		6.7	As a household admin, I want to track my members shopping carts	Medium	5	8	MUST
7	History and statistics	7.1	As a user, I want to consult my purchase-history	Low	6	5	MUST
		7.2	As a user, I want to consult the purchase-history of my household members	Low	6	5	MUST
		7.3	As a user, I want to obtain statistics on my food consumption (categories, expenses, etc.)	Low	6	8	COULD
		7.4	As a user, I want to consult statistics of my household members consumptions	Low	6	13	COULD
8	GMS Account Management	8.1	As an Employee, I want to create an account using different methods	Medium	7	8	MUST
		8.2	As an Employee, I want to login	Medium	7	5	MUST

		8.3	As an Employee, I want to join a GMS	Medium	7	5	MUST
		8.4	As a GMS Admin, I want to manage employees	Medium	7	5	MUST
		8.5	As a Cashier, I want to synchronize my mobile app session with the GMS system	Medium	7	8	MUST
9	Checkout Interactions	9.1	As a Cashier, I want to scan clients shopping cart's generated QR code to retrieve the products in question	High	7	8	MUST
		9.2	As a user, I want to receive the final price of all of my products	High	7	8	MUST
		9.3	As a cashier, I want to finalize the transaction	Medium	7	5	MUST
10	Stock tracking	10.1	As a Manager, I want the stock to be updated automatically after transactions	Medium	7	5	SHOULD
		10.2	As a Manager, I want to receive out of stock alerts	Medium	8	5	COULD
		10.3	As a Manager, I want to receive analytics of clients consumption trends to better optimize stock management	Low	8	13	COULD
11	Statistics and analysis	11.1	As a Manager, I want to visualize clients buying habits	Low	8	8	MUST
		11.2	As a Manager, I want to visualize the most trending products across different groups of clients	Low	8	13	MUST
		11.4	As a Manager, I want to visualize consumption trends per product category	Medium	8	8	COULD
12		12.1	As a Manager, I want to create promotional offers for certain products	Medium	9	5	SHOULD

	Management of promotional offers	12.2	As a Manager, I want to target and create promotional offers based on consumer segments.	Low	9	13	COULD
		12.3	As a Manager, I want to monitor the effectiveness of promotional offer campaigns	Low	9	8	COULD
13	Shop Flow Admin Management	13.1	As an Admin, I want to login to access the admin dashboard	Medium	8	5	SHOULD
		13.2	As an admin, I want to add a supermarket and its API paths to the shop flow system to allow users to shop there.	Medium	8	13	SHOULD
		13.3	As an Admin, I want to manage all the added GMS providers	Low	9	5	COULD


Table 2.2: ShopFlow Product Backlog

3 Development Environment

In this section, we'll explore the technologies behind our project development. We'll start by defining the tools we've employed and then delve into why we chose them.

3.1 Backend Development Envirement

the Table 2.3 showcases the environment for the backend development

Technology	Definition	Why
 Node.JS & Express	<p>Node.js serves as a powerful development environment for server-side JavaScript applications. Express, on the other hand, is a framework built for Node.js.</p>	<ul style="list-style-type: none"> - Efficient and easy to use - Express offers out-of-the-box features for creating HTTP servers and APIs - Ideal for building highly scalable and responsive microservices thanks to its asynchronous nature


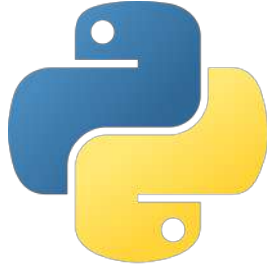

Technology	Definition	Why
	<p>MongoDB is a NoSQL database management system that stores application data in the form of JSON documents</p>	<ul style="list-style-type: none"> - Flexible structure - Ability to store data in various formats. - Efficient in handling queries. - Capability to manage large volumes of data. - Easy integration with Node.js and Express.
 <p>Python</p>	<p>Python is the most widely used open-source programming language among computer scientists. It has surged to the forefront in infrastructure management, data analysis, and software development.</p>	<ul style="list-style-type: none"> - Diverse libraries for data analysis, machine learning and AI. - Flexible and powerful libraries. - ideal for building recommendation systems and business intelligence dashboards.

Table 2.3: Backend Development Environment

3.2 Frontend Development Environment

the Table 2.4 showcases the environment for the frontend development.

Technology	Definition	Why
 <p>Flutter</p>	<p>Flutter, created by Google, is an open-source toolkit for developing natively compiled applications on mobile, web, and desktop from a single codebase.</p>	<ul style="list-style-type: none"> - Widget centric desgin. - Fast and consistent development experience across multiple platforms (IOS and Android).







Technology	Definition	Why
 <p>Angular</p>	<p>Angular is an open-source JavaScript framework, developed and maintained by Google, that provides a standard structure and additional features to simplify web and mobile application development.</p>	<ul style="list-style-type: none"> - Two-way data binding. - data-driven templates. - testing utilities.





Table 2.4: Frontend Development Environment

3.3 Other Tools

The tools utilized during our project development are summarized in Table 2.5.

Technology	Definition	Why
 <p>Gitlab</p>	<p>GitLab is a complicated collection of tools, databases, queues, and glue code holding it all together. It integrates all the software development life cycle tools.</p>	<ul style="list-style-type: none"> - Efficient code management for Kubernetes cluster. - A single tool that unifies all the steps in the Software Development Life Circle under one umbrella.[19]
 <p>Docker</p>	<p>Docker is an open platform for developing, shipping, and running applications. Docker allows you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.</p>	<ul style="list-style-type: none"> - Shipping, testing and deploying code quickly. - Reduced delay between writing code and running it in products. - Open-source and has a vast community base that supports it

Technology	Definition	Why
 Kubernetes	Kubernetes is a container orchestrator, developed by Google and then donated to Cloud Native Computing Foundation (CNCF). It is a true open-source project, with probably the highest velocity in history.[12] It is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.[8]	<ul style="list-style-type: none">- Exceptional capabilities and widespread industry adoption.- Portable, extensible, flexible.- Simple deployment and management especially of microservice-based applications- Active community
 Android Studio	Android Studio is an IDE for Android development, provides a comprehensive set of tools including debugging tools and integrated emulators.	<ul style="list-style-type: none">- Comprehensive toolset- Easy integration with flutter
 Visual Studio Code	Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications.	<ul style="list-style-type: none">- Efficient and collaborative development.- Debugging and extension capabilities.

Technology	Definition	Why
 Pycharm	PyCharm is a full-stack IDE for data science and web development that supports Python, Django, Flask, FastAPI and more.	<ul style="list-style-type: none">- Feature rich code editor.- Diverse tools.
 Microsoft Azure	Microsoft Azure is a cloud computing platform, offering a wide range of services and solutions to meet the demands of cloud computing.	<ul style="list-style-type: none">- Robust cloud infrastructure and services.- Flexible.- Built-in support for kubernetes clusters.
 Postman	Postman is an API platform for testing and using APIs.	<ul style="list-style-type: none">- Convenient for testing and validating REST APIs.- Easy to use interface.
 StarUML	StarUML is a UML modeling software that assists developers in creating diagrams to visualize the structures and behaviors of systems.	<ul style="list-style-type: none">- User-friendly interfaces.- Different tools and types of diagrams for UML.

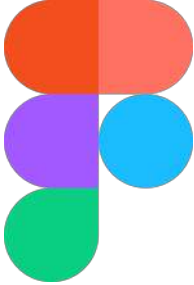


Technology	Definition	Why
 <p>Figma</p>	<p>Figma design is for people to create, share, and test designs for websites, mobile apps, and other digital products and experiences.</p>	<ul style="list-style-type: none"> - Flexible. - Multiplayer (Collaboration). - Can be used in a wide range of devices.
 <p>Overleaf Latex</p>	<p>LaTeX is a tool that lets anyone create beautifully typeset documents through writing code.</p>	<ul style="list-style-type: none"> - Collaborative work. - No downloads or installations needed. - Customization and Extensibility.
 <p>Firebase</p>	<p>Firebase is a suite of tools for hosting and developing mobile and web applications. It facilitates sending notifications and advertisements, as well as tracking errors and user interactions within the application</p>	<ul style="list-style-type: none"> • Reliable • Easy integration with mobile applications • Detailed analytics

Table 2.5: Tools Used in Project Development

4 System Architecture

To choose the most suitable architecture we first have to analyze the different architectures and align them with the project's goals and requirements.

4.1 Monolith vs. Microservice

A **monolithic** architecture is a traditional software development model that uses one code base to perform multiple business functions.

All the software components in a monolithic system are interdependent due to the data exchange mechanisms within the system. It's restrictive and time-consuming to modify monolithic architecture as small changes impact large areas of the code base.

In contrast, **microservices** are an architectural approach that composes software into small independent components or services.

Each service performs a functionality and communicates with other services through a well-defined interface.

Because they run independently, you can update, modify, deploy, or scale each service as required.[8]

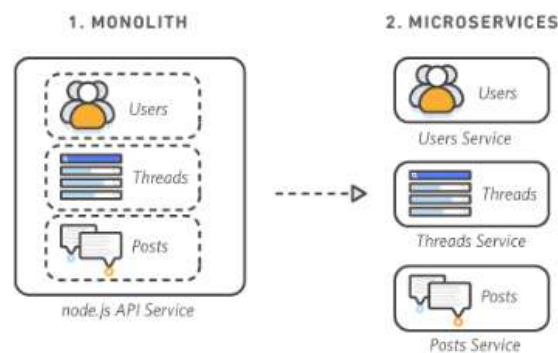


Figure 2.6: Microservice vs Monolith [8]

Criteria	Monolith	Microservice
Design	Single code base.	Independent services with autonomous functionality, that communicate with each other
Development	Requires less planning at the start but gets increasingly complex to understand and maintain.	Requires more planning and infrastructure at the start, but gets easier to manage and maintain over time.
Deployment	Entire application is deployed as a single entity.	Every microservice is an independent software entity that requires individual containerized deployment.
Debugging	Trace the code path in the same environment.	Requires advanced debugging tools to trace the data exchange between multiple microservices.
Modification	Small changes introduce greater risks as they impact the entire code base.	You can modify individual microservices without impacting the entire application.

Scale	You have to scale the entire application, even if only certain functional areas experience an increase in demand.	You can scale individual microservices as required, which saves overall scaling costs.
Investment	Low upfront investment at the cost of increased ongoing and maintenance efforts.	Additional time and cost investment to set up the required infrastructure and build team competency. However, long-term cost savings, maintenance, and adaptability.

Table 2.6: Monolith vs MicroService[8]

4.2 Choice Justification

We chose a microservices architecture for our system for the following reasons:

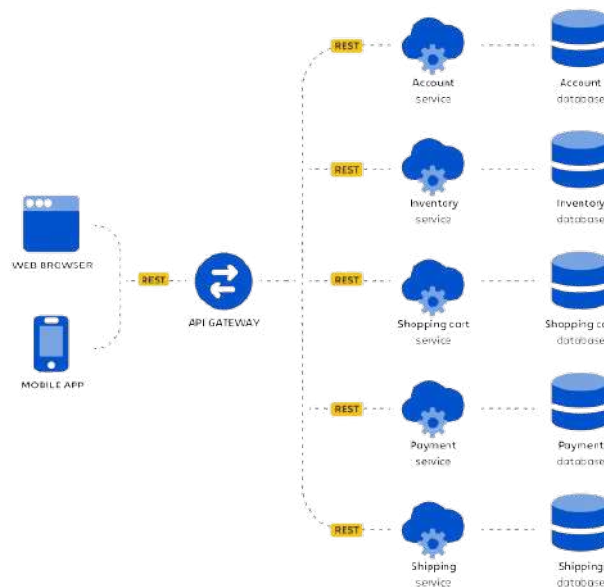


Figure 2.7: Microservice Architecture

- Technology Diversity** Our strategic adoption of a microservices architecture is primarily driven by the imperative need for flexibility in technology choices and programming languages. Our application incorporates some features requiring data analysis and machine learning capabilities, necessitating the use of diverse technologies (python for example). This inherent diversity is a challenge that a monolithic architecture restricts, whereas microservices empower us to integrate various technologies, enabling the development of specialized services tailored to each unique requirement.
- Cloud environments and DevOps** Given that our application is predominantly designed for deployment in a cloud environment. Microservices, with their inherent cloud-friendliness, emerge as the optimal architectural choice. This approach

not only aligns well with cloud environments but also allows us to harness the power of essential DevOps tools such as Docker and Kubernetes. By adopting a microservices architecture, we can effectively utilize containerization (Docker) for streamlined deployment, scalability, and resource efficiency, while Kubernetes enhances our orchestration capabilities, ensuring efficient management and scaling of microservices across our cloud infrastructure.

- **Continuous Integration/Continuous Delivery** In adopting a microservices architecture, each update to our application undergoes integration and delivery through a CI/CD pipeline. This ensures that testing, building, and deployment are performed for specific services independently. In contrast, a monolithic approach would involve building, testing, and deploying the entire application as a single unit with each code push. The microservices model, with its individualized CI/CD pipelines, provides a more efficient and agile deployment process, allowing for independent updates and faster delivery of new features.
- **Improved Resource Utilization** Microservices enable efficient resource utilization by allowing services to scale independently based on demand. This flexibility ensures that resources are allocated where they are needed most, optimizing performance and minimizing unnecessary resource allocation, which can be challenging in a monolithic architecture.

4.3 Service Decomposition by business capability pattern

One strategy for creating a microservice architecture is to decompose by business capability. A concept from business architecture modeling. a business capability is something that a business does in order to generate value. The set of capabilities for a given business depends on the kind of business. For example, the capabilities of an online store include Order management, Inventory management, Shipping, and so on. Once the business capabilities are identified, then comes the definition of a service for each capability or group of related capabilities. [9]

4.4 Database-per-service pattern

Loose coupling is the core characteristic of a microservices architecture, because each individual microservice can independently store and retrieve information from its own data store. By deploying the database-per-service pattern, we choose the most appropriate data stores which are non-relational databases for our application and business requirements. This means that microservices don't share a data layer, changes to a microservice's individual database do not impact other microservices, individual data stores cannot be directly accessed by other microservices, and persistent data is accessed only by APIs. Decoupling data stores also improves the resiliency of our overall application, and ensures that a single database can't be a single point of failure.[10]

The issue with using a database per service is that some queries need to join data that are owned by multiple services. A service's data is only accessible via its API, so we can't

use distributed queries against its database.[9]

We have decided to utilize the API Composition pattern to handle complex queries that necessitate joins across multiple databases. By referencing IDs in the documents of different collections, we can efficiently manage these relationships. When a complex query is required, we create a composed API that performs the necessary joins by invoking multiple underlying APIs.

4.5 Mobile Application Design Pattern

In this section, we will dive deeper into the design pattern of our mobile application

4.5.1 State Management

Managing the state of an application is one of the most important and necessary processes in the application lifecycle. The main task of a user interface is to represent the state. The state of an application includes everything that exists in memory while the application is running. This includes the application's resources, all the variables that the framework maintains regarding the user interface, the state of the animations, etc. From this definition, we can conclude that the model which programmatically establishes how to track changes and how to disseminate details about the states to the rest of our application is known as state management.

4.5.2 GetX as a state management solution

GetX [11] is an extra-light and powerful solution for Flutter. It combines high-performance state management, intelligent dependency injection, and route management quickly and practically. GetX operates on three core principles: productivity, performance, and organization.

- **Performance:** GetX prioritizes minimal resource consumption and high performance without using Streams or ChangeNotifier.
- **Productivity:** GetX uses a simple and minimalist syntax.
- **Organization:** GetX enables complete decoupling of View, presentation logic, business logic, dependency injection, and navigation. It eliminates the need for context for navigation or accessing controllers, resulting in cleaner, more organized code.

Figure 2.8 shows the architecture of GetX.

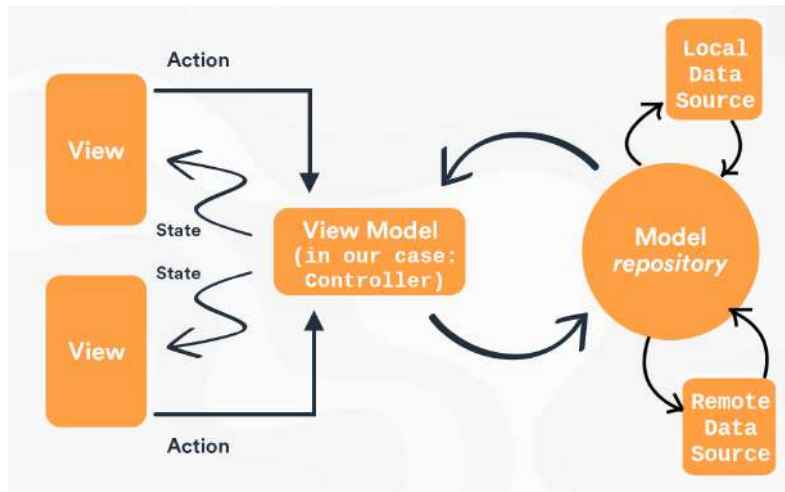


Figure 2.8: GetX Architecture

4.5.3 Mobile Project Structure

By relying on GetX and to ensure the application is easily testable and maintainable, we used a modular structure. Figure 2.9 shows the modular structure of our mobile project.

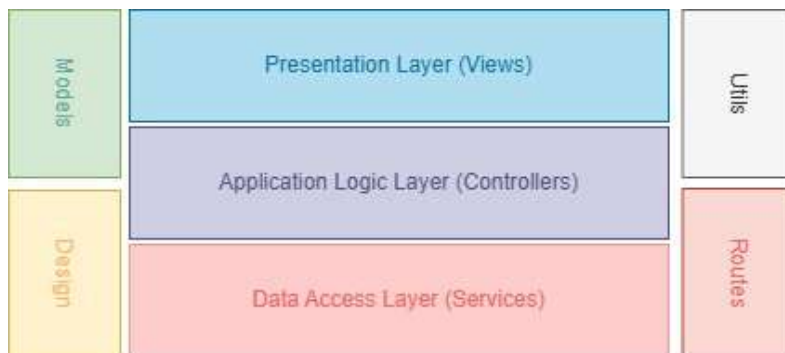


Figure 2.9: Mobile Project Structure

- The **presentation layer** is responsible for configuring and displaying the screens to the user based on the model's data.
- The **Application logic layer** layer contains all the operations necessary to ensure synchronization between the application and the server.
- The **data access** layer represents the mechanisms for reading and writing data.
- **Models** represent the structure of the data within the application. They encapsulate the various attributes and behaviors of the data entities.
- **Design** includes common UI components and themes for consistent visual representation.
- **Utils** contains utility functions for data formatting and configurations.
- The **Routes** are responsible for defining different routes for different screens and managing the navigation between the different screens of the application

5 Applied Technologies

During this section, we will dive into the various concepts that we will use in this project. These include Containerization, DevOps, Continuous integration/Continuous delivery/deployment, Kubernetes and more.

5.1 Containerization

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. [12] The advantages of containerization are as follows:

- **Speed:** Containers start almost instantly, letting us launch or expand applications in seconds.
- **Portability:** Containers work consistently across different setups, reducing issues with compatibility.
- **High density:** Many containers can run on one machine, using resources even more efficiently than VMs.
- **Scalability:** Containers make it easier to scale applications horizontally. We can add multiple identical containers to create multiple instances of the same application.
- **Rapid deployment:** Switching from a development environment to a production environment is done in one click, without worrying about configuration and resource changes.

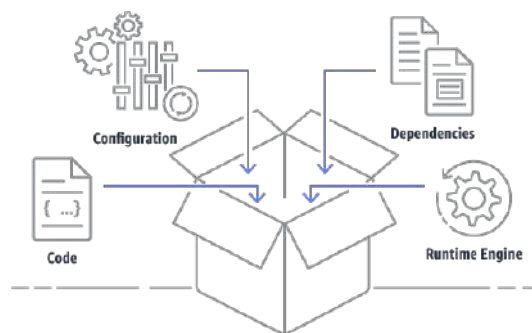


Figure 2.10: Illustration of a Container[13]

5.2 Container Orchestration with Kubernetes

As we used containers more, we needed a way to organize, scale, and watch over them. That's where container orchestrators like Kubernetes comes in.

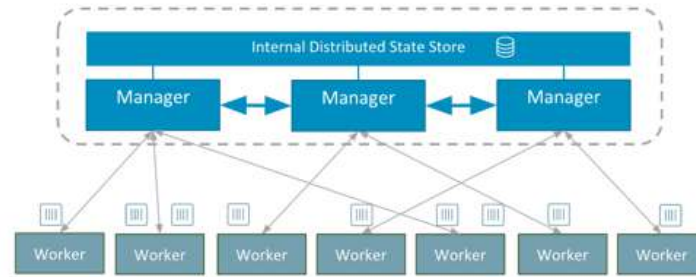


Figure 2.11: Container Orchestration Concept[14]

Functions:

- **Auto-scaling:** Orchestration systems can grow or shrink applications automatically based on demand.
- **Self-healing:** If a container or server fails, the orchestrator can restart or replace it.
- **Load balancing:** Incoming requests are spread across containers for the best performance.
- **Service discovery:** As containers move around, services can be found and communicated without manual effort.
- **Manage deployment:** Propose several deployment strategies to avoid system downtime during an update.

To achieve these goals, we employed the famous container orchestrator. Kubernetes, developed by Google and then donated to Cloud Native Computing Foundation (CNCf). It is a true open-source project, with probably the highest velocity in history.[15] It is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.[16]

5.2.1 Main components provided by Kubernetes

To manage a cluster, Kubernetes provides a range of components. In this section, we define the components used in our infrastructure. We will group the components into categories based on their functionality.

Security

- **Namespace** namespaces provide a mechanism for isolating groups of resources within a single cluster.
- **Secret** A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key.

Service: is a method for exposing a network application that is running as one or more Pods.

Here are some types of services :

- **ClusterIp** : Exposes the Service on a cluster-internal IP.
- **NodePort** : Exposes the Service on each Node's IP at a static port.
- **LoadBalancer** : Exposes the Service externally using an external load balancer.

Routing

- **Ingress:** An API object that manages external access to the services in a cluster, typically HTTP.

Deployments

You can find a snippet of an ingress controller code in the appendix, figure A.1 .

- **Pods:** the smallest deployable units of computing. It's a group of one or more containers.
- **ReplicaSet:** maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods.
- **Deployment:** provides declarative updates for Pods and ReplicaSets.

5.2.2 Kubernetes Architecture Components

Kubernetes Architecture Components. [17] A Kubernetes cluster is made of control plane nodes and worker nodes. These are Linux hosts that can be virtual machines (VM), bare metal servers in datacenters, or instances in a private or public cloud.

Control plane

A Kubernetes control plane node runs a collection of system services that make up the control plane of the cluster. Sometimes we call them Masters, Heads or Head nodes. Let's take a quick look at the different services making up the control plane. All of these services run on every control plane node.

- **The API server:** The API server is the hub of Kubernetes, handling all communication between components. Both internal system parts and external users interact through the API server. It offers a RESTful API for submitting YAML configuration files over HTTPS. These YAML files, or manifests, outline the desired state of an application.
- **The cluster store :** The cluster store is the only stateful part of the control plane, storing the entire configuration and state of the system. It's essential for every Kubernetes setup – without it, the cluster can't function.

- **The scheduler:** The scheduler monitors the API server for new tasks and assigns them to suitable, healthy worker nodes. It uses complex logic to filter out nodes that can't handle the tasks and then ranks the remaining nodes. The highest-ranked node is chosen to run the task.

Working Nodes

Worker nodes are where user applications run. At a high-level they do three things:

Watch the API server for new work assignments
Execute work assignments
Report back to the control plane (via the API server)

- **Kubelet:** The Kubelet is the main Kubernetes agent running on every worker node. When a node joins a cluster, the Kubelet is installed and registers the node's CPU, memory, and storage with the cluster. It watches the API server for new tasks, executes them, and reports back to the control plane.
- **Container runtime:** The Kubelet requires a container runtime to handle tasks like pulling images and managing containers. Initially, Kubernetes supported Docker natively, but it has since adopted the Container Runtime Interface (CRI). The CRI provides a clean, documented interface for third-party container runtimes to integrate with Kubernetes.
- **Kube-proxy:** The final component of the worker node is the kube-proxy, which manages local cluster networking on each node. It assigns a unique IP address to each node and uses iptables or IPVS rules for routing and load balancing traffic.

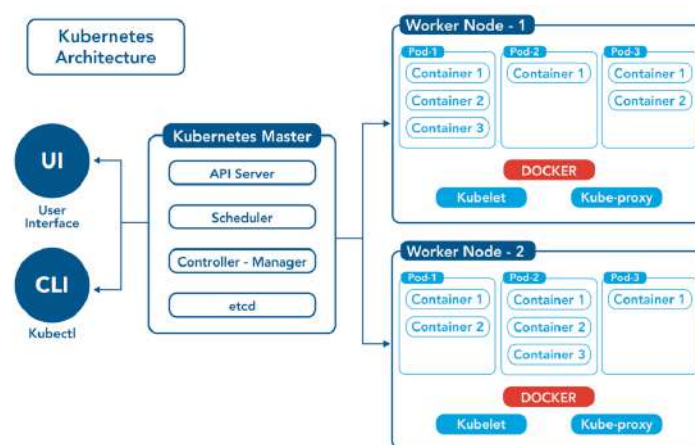


Figure 2.12: Kubernetes Architecture

5.3 DevOps

DevOps (Development plus Operations) offers process frameworks augmented with open-source tools to integrate all the phases of the application life cycle, and ensure they function as a cohesive unit. It helps to align and automate the process across the phases

of development, testing, deployment, and support. It includes best practices such as code repositories, build automation, continuous deployment, and others.[18]

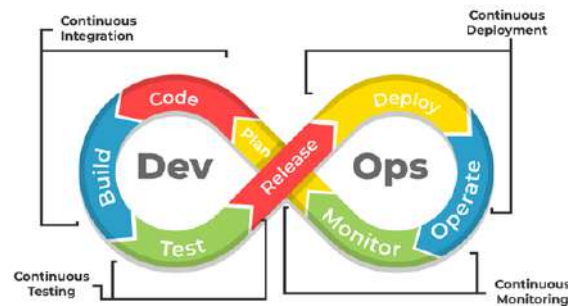


Figure 2.13: DevOps

What do we mean by **DevOps**? Despite the term being used by the software community for at least 10 years (the first session of **devopsdays**, which is now the biggest DevOps-focused conference, was held in 2009), today, there is no single, standard definition that everyone agrees on.[19]

In DevOps, the Dev and Ops teams pursue conflicting objectives:

Dev Team Seeks	Ops Team Seeks
-	Stability of production apps
Frequent deployments and updates	Manage infrastructure, not applications
Easy creation of new resources	Monitoring and control

Table 2.7: the Dev and Ops teams preferences

5.4 Continuous integration (CI), Continuous delivery/deployment (CD)

Given that a significant portion of GitLab's effectiveness arises from configuring CI/CD pipelines for diverse actions on the code, it is crucial to comprehend the essence of a pipeline. Therefore, a logical starting point for exploring this subject is to clarify the definitions of pipeline, CI, and CD.

A GitLab CI/CD pipeline is a series of steps that are performed on our files whenever we commit edits to the GitLab-hosted copy of a repository.[19] Given that each of our microservices are hosted on separate Gitlab repositories as separate projects, each project defines only one GitLab CI/CD pipeline.

5.4.1 Continuous integration

The main objective of Continuous Integration is to establish a consistent and automated method for building, packaging, and testing applications. With the implementation of

the integration process, teams are more likely to validate code changes more frequently, enhancing collaboration and software quality.

5.4.2 Continuous delivery

Continuous Delivery is a practice involving the automation of the entire software release process. The aim is to automatically prepare and track a version until it reaches production. Anyone with sufficient privileges to deploy a new version can do so at any time with just a few clicks. By eliminating almost all manual tasks, developers become more productive.

5.4.3 Continuous deployment

Continuous Deployment is an advanced stage beyond Continuous Delivery in which every change to the source code is automatically deployed to production without explicit approval from a third party. A developer's task typically concludes with the review of a pull request and its merge into the Master branch. A CI/CD service takes over by running all tests and deploying the code to production, while keeping the team informed of the results of each significant event. Continuous Deployment requires a highly developed culture of monitoring and the ability to restore quickly.

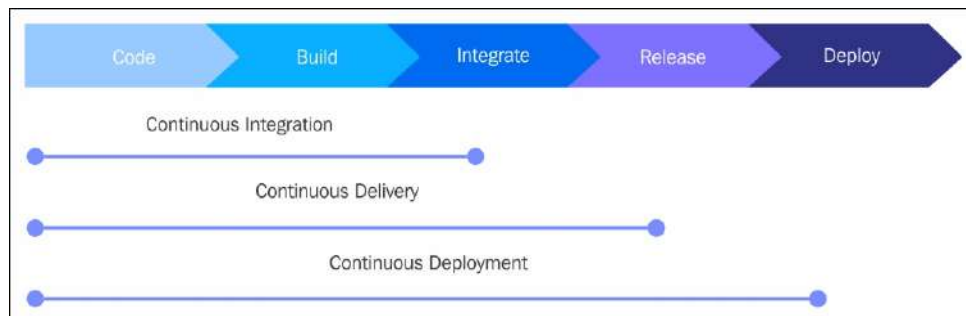


Figure 2.14: Continuous Integration, Delivery and Deployment[20]

5.5 Machine Learning

Machine learning[21] is a subfield of computer science that is concerned with building algorithms that, to be useful, rely on a collection of examples of some phenomenon. These examples can come from nature, be handcrafted by humans, or be generated by another algorithm.

Machine learning can also be defined as the process of solving a practical problem by

1. collecting a dataset, and
2. algorithmically training a statistical model based on that dataset.

Given that our project centers on recommendation systems, we will dive deeper into this concept in the upcoming chapters.

5.6 Business Intelligence

In essence, Business Intelligence (**BI**) is any activity, tool, or process used to obtain the best information to support the process of making decisions.

Whether it's calling the Psychic Hotline, using an army of consultants, or having banks of computers churning data; if it helps get a better handle on the company's current situation, and provides insight into what to do in the future, it's BI.

BI revolves around putting computing power (highly specialized software in concert with other more common technology assets) to work, to help make the best choices for your organization. This often involves querying and reporting: Querying is a request for information from a database. Reporting is presenting the information in a meaningful, understandable way. The data sources are operational databases, in use by each department to record regular transactions and store data that is frequently accessed and changed. The data sources connect into the data warehouse through a set of processes commonly referred to as **ETL**, or **Extract, Transform, and Load**. [22]

These processes do exactly what they say:

First they grab relevant data from the operational databases, then they change it into a single, unified format so all data is apples-to-apples.

After the data is clean, it's loaded into the data warehouse (in our case, we will be using a normal database).

Querying and reporting tools sit astride the data warehouse. They act as the abstraction layer between the user and the really confusing, complex code that pulls just the right data from the database and puts it on somebody's desktop in a readable format. [22]

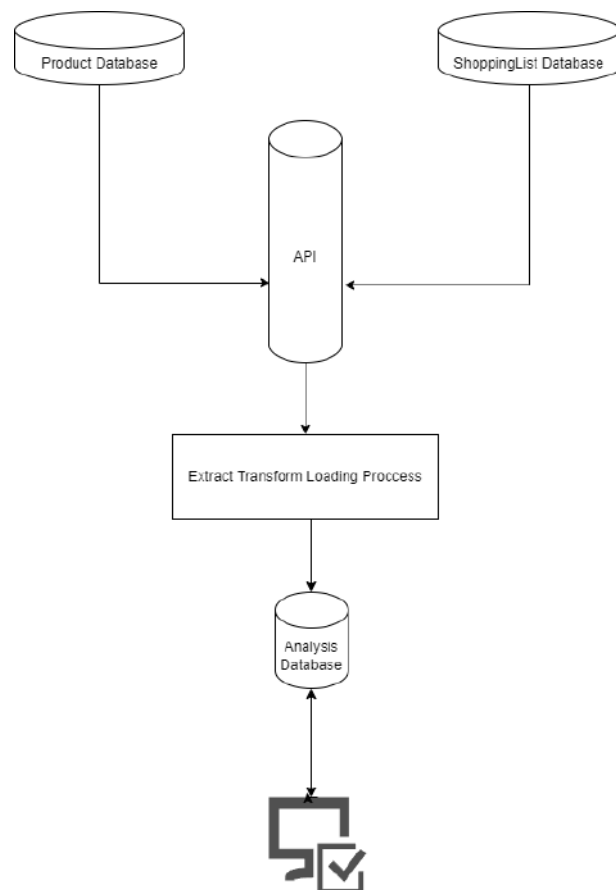


Figure 2.15: ETL Process

6 Cloud Computing

Simply put, cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the internet

(“the cloud”) to offer faster innovation, flexible resources, and economies of scale. these services, help lower operating costs, run infrastructure more efficiently, and scale as the business needs change.[23]

6.1 Reasons to use cloud-based infrastructure

Among the criteria that helped us choose a cloud-based infrastructure are:

Cost

Moving to the cloud helps companies optimize IT costs. This is because cloud computing eliminates the capital expense of buying hardware and software and setting up and running onsite datacenters—the racks of servers, the round-the-clock electricity for power and cooling, and the IT experts for managing the infrastructure. It adds up fast.

Global scale

The benefits of cloud computing services include the ability to scale elastically. In cloud speak, that means delivering the right amount of IT resources—for example, more or less computing power, storage, and bandwidth—right when they’re needed, and from the right geographic location.

Security

Many cloud providers offer a broad set of policies, technologies, and controls that strengthen security posture overall, helping protect data, apps, and infrastructure from potential threats.

Backup, recovery, and failover

Most cloud providers help to improve business continuity by offering built-in, one-click backup and recovery capabilities. Some providers also offer the ability to store backups in different geographic regions.

6.2 Choice of Cloud Provider: Microsoft Azure

Since we use Kubernetes, our choice of provider has been based on the managed Kubernetes services offered by the big 3 Cloud providers. We then did a comparison between AKS from Azure, EKS from AWS, and GKE provided by Google. We decided to group together the different features available for each service managed by the cloud provider under these sections.

Table 2.8 shows a comparison between the big 3 of cloud provider’s managed Kubernetes services

As- pect/Service	AKS	EKS	GCP
Year Re- leased	2017	2018	2014

Kubernetes Control Plane Up-grade	Manual	Manual	Automated (default) or manual
SLA	99.95%	99.99%	99.95%
Kubernetes Control Plane Price	Free	\$0.10/hour	\$0.10/hour
Network Policies	Yes: Azure Network Policies or Calico	Need to install Calico	Yes: Native via Calico
Auto Scaling	Yes	Yes	Yes
Monitoring	Azure Monitor container health feature	Kubernetes control plane monitoring connected to Cloudwatch, Container Insights Metrics for nodes	Kubernetes Engine Monitoring and integration with Prometheus

Table 2.8: Comparison Between managed Kubernetes services [24]

6.2.1 Justification of choice

Azure Kubernetes Services might be the most effective in our case since it's the most cost-effective service as we do not have to pay for the Control Plane. alongside it has some good strengths including:

- **Timely Updates:** AKS has been the fastest to provide the newer Kubernetes versions as well as minor patches.
- **Automatic Node Health Repair:** AKS provides automatic node health repair.
- **Network Policy Setup:** Azure Network Policies and Calico Network Policies can be set up automatically when a cluster is created.
- **Developer Environment:** There is a good developer environment. We can use the Kubernetes extension in Visual Studio Code to deploy to AKS. [25]

7 The release planning

This project will unfold through a series of sprints, each spanning three weeks corresponding to 105 hours or 15 working days. In every sprint, we will follow all the ceremonies, such as planning, review, and retrospective, and work on developing and testing the user stories that we have prioritized in the sprint backlog.

Release							
Sprint 1		Sprint 2		Sprint 3		Sprint 4	
US 1.1	US 3.4	US 1.3	US 6.4	US 9.2	US 10.6	US 5.1	US 12.2
US 1.2	US 3.5	US 1.5	US 6.6	US 9.3	US 10.7	US 7.1	US 12.3
US 1.4	US 3.6	US 3.7	US 6.7	US 3.3	US 13.1	US 7.2	US 12.4
US 2.1	US 4.1	US 4.5	US 8.1	US 6.5	US 13.2	US 7.3	US 12.5
US 2.2	US 4.2	US 4.6	US 8.2	US 10.1		US 7.4	
US2.3	US 4.3	US 5.2	US 8.3	US 10.2		US 11.1	
US2.4	US 4.4	US 6.1	US 8.4	US 10.3		US 11.2	
US 3.1		US 6.2	US 8.5	US 10.4		US 11.4	
US 3.2		US 6.3	US9.1	US 10.5		US 12.1	
~130 hours		~120 hours		~110 hours		~135 hours	

Throughout this chapter, we've introduced both the functional and non-functional requirements, along with the product backlog and our working environment. Additionally, we've defined the architecture of our application. The next chapter will focus on unveiling our inaugural sprint.

Chapter 3. Sprint 1 : Authentication, Inventory Management & Household Collaboration

After outlining our requirements in the previous chapter, we now focus on sprint 1. First, we introduce the backlog while detailing the objective of each feature set. Second, we will introduce the main actors of this sprint followed by the user interface mock-ups. Third, we will detail the class diagram followed by a sequence diagram to better understand the workflow of our application. Additionally, we'll have a look over the interfaces that we implemented. Lastly, we will have a look over this sprint's retrospective.

1 Identification of needs

During this segment, we will present the sprint 1 backlog followed by the use case diagram and finishing with some user interface Mock-ups.

1.1 Sprint One Backlog

The core features of this sprint include the following :

- **Profile management** : This feature encompasses everything related to authentication such as the ability to **Login/Sign-up** using different methods such as Google.
- **Acquisition of product data** : This feature outlines the different methods available to the user to access information about the desired product. Users can utilize functionalities such as scanning a product using QR-codes or Bar-codes.
- **Inventory management** : This feature details the inventory logic of our application. After scanning a product, the user will be prompted to add it to the inventory ,if he wishes so, while inputting the expiration date and the desired quantity. Once added, it will be saved in his inventory along side other products previously scanned.
- **Household management** : This feature covers the idea of inviting members to a user's household by enabling them to share an inventory and monitor additions made by each member.

Sprint 1 : Authentication, Inventory Management & Household Collaboration

Feature ID	Sprint Feature	ID	User Story	Task			EST
S1_F1	Profile management	1.1	As a user, I want to create an account using different signup methods	Backend	Email-Sign up	Create the User.js model	1h
						Create the userController.js	
						Create the userRouter.js	
						Create the mongoDB database	
						Create the necessary files to link the backend with the database	
						Create the necessary middlewares	
				Google Sign up		Create the helpers for google authentication	5h
						Create and configure a google developer account	
						Create the token.js config file for JWT	
				Frontend	Sign-up Interface	Create the User.dart model	5h
						Create the userService.dart	
						Create the welcome.dart page letting you choose between signing up or logging in	
						Create the signup.dart screen	
						Create the routes.dart	
						Add the signup screen to the routes for easier navigation	
					Interactive form interface	Create the interactiveForm.dart screen	4h
						Create the interactiveFormController.dart	
						Add the interactiveForm screen to the routes for easier navigation	
				Backend	Modify user information	Create the functionality in the userController.js	1h $\frac{1}{2}$
						Add the function in the userRouter.js	

		1.2	As a user, I want to modify my personal information for better personalization	Frontend	Profile interface	Create the settings.dart screen	4h
						Add the Profile interface in the routes.dart for easier navigation	
					Edit Profile	Create the modify_profile.dart screen	4h
						Create the modifyProfile functionality in the userService.dart	
						Add the created modifyProfile functionality in the userController.dart	
S1_F2	Acquisition of product data	2.1	As a user, I want to scan products using Bar-codes	Back-end	Get Product by code	similar tasks in a previous user story	4h
				Frontend	Scan Product Interface	similar tasks in a previous user story	7h
						Create the ScanProduct.dart screen	
						Create the productDetailsScreen.dart	
						Add the ScanProduct.dart screen and the productDetailsScreen.dart to the routes for easier navigation	
S1_F3	Inventory management	3.1	As a user, I want to visualize my current inventory	Back-end	Get Inventory	similar tasks in a previous user story	7h
				Frontend	Inventory Interface	similar tasks in a previous user story	5h
						Create the inventoryDetails.dart screen	
						Add the inventoryDetails.dart screen to the routes for easier navigation	
		3.2	As a user, I want to receive a notification when a product	Backend	Notification Management	similar tasks in a previous user story	5h
						Create the necessary files to link the backend with the database and firebase	
						Create the firebase Configuration	

			in my inventory is soon to expire			Create the functionality in the notificationController.js and add it to the notificationRouter.js	
				Frontend	Notification Management	Create the firebase Configuration	3h
		3.4	As a user, I want to add products to my inventory	Backend	Add Product to inventory	Create the functionality in the inventoryController.js and add it to the inventoryRouter.js	2h
				Frontend	Add Product to Inventory Form	Create the functionality in the InventoryService.dart and add it in the inventoryController.dart	3h
Add a button that opens a form in the ProductDetails.dart							
						Create the inventoryProduct.dart screen	
S1_F4	Household management	4.1	As a user, I want to invite other users to my household to share inventories	Backend	Invite Members with Invite Code	similar tasks in a previous user story	4h
				Frontend	Invite Member Interface	similar tasks in a previous user story	5h
						Create the inviteCodeScreen.dart	
						Create the acceptInvite.dart screen	
Global Tasks				Setup the different microservices		8h	
				Setup the flutter frontend project by initializing the necessary modules		5h	
				Create the Use Case Diagram		1h	
				Create the Class Diagram		1h	
				Create three sequence diagrams		2h	
				Create Test scripts		2h	
				Build and push a Docker image for each feature		½h	
				set up a Kubernetes development environment		2h	

Sprint 1 : Authentication, Inventory Management & Household Collaboration

	create the Kubernetes configuration YAML files(deployments, services, ingress controller...)	3h
	set up a simple ci/cd pipeline to automate testing, building and deployment(locally)	3h
Total Estimation		~130 hours
For the full backlog please refer to the sprint 1 appendix, section 8.2		

Table 3.10: Sprint 1 backlog snippet

1.2 Sprint 1 Use Case Diagram

This diagram illustrate the various features planned for the first sprint into use cases.

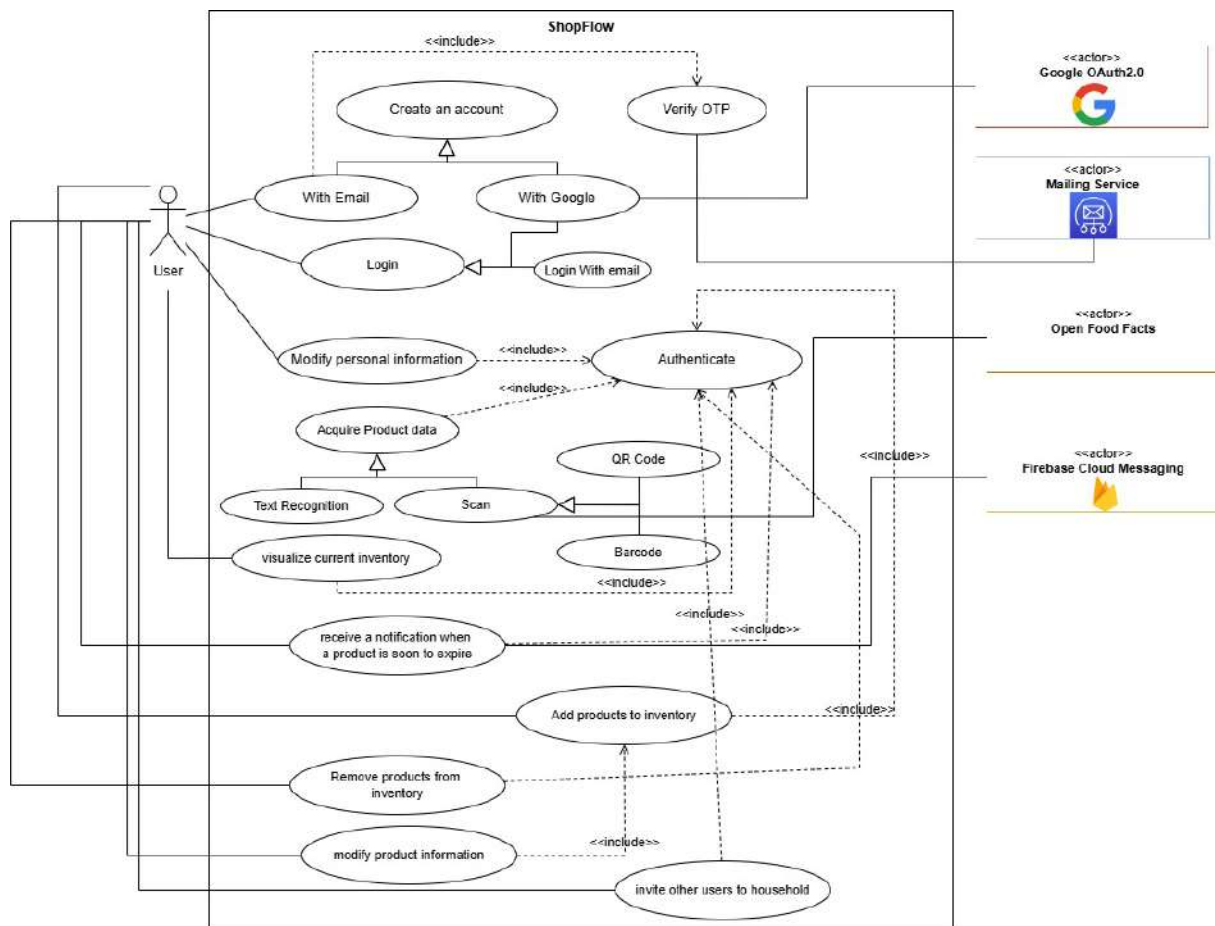


Figure 3.16: Sprint 1 Use Case Diagram

1.3 User Interface Mock-ups

These are some of the user interface mock-ups for our application:

The figure 3.17 represents the UI verification code sent to an email when signing up.

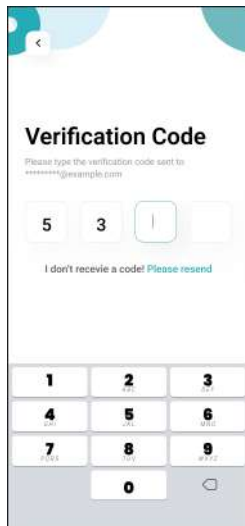


Figure 3.17: Verification code interface mock-up



Figure 3.18: Scan A Product interface mock-up

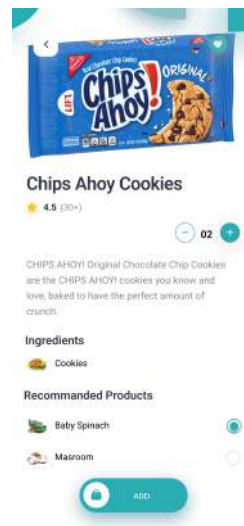


Figure 3.19: Product Details interface mock-up



Figure 3.20: User Inventory interface mock-up

The figure 3.18 represents the process of scanning a product via Bar-code, Qr Code or Text Recognition.

The figure 3.19 represents the products details interface after scanning a desired product.

The figure 3.20 represents the user's inventory, detailing the products that were added to it.

Here's a brief explanation implying an acceptance criteria analysis for one of our stories. using the Given/When/Then format.


Mockup	User Story	Acceptance Criteria
 <p>Figure 3.18</p>	<p>US2.1: As a user, I want to scan products using Bar-codes</p>	<p>Given a user is on the scan a product page,</p> <ul style="list-style-type: none"> - When the user gives camera access and scans a valid Barcode, - Then the user should get redirected to the product details page. <p>Given a user is on the scan a product page,</p> <ul style="list-style-type: none"> - When the user gives camera access but does not scan a valid barcode, - Then the system should display a warning message. <p>Given a user is on the scan a product page,</p> <ul style="list-style-type: none"> - When the user does not give camera access, - Then the system should not allow the user to scan.

Table 3.11: Story acceptance criteria

2 Design

This section is dedicated to the development of sequence and class diagrams for this sprint.

2.1 Class Diagram

This is the Class Diagram of our first sprint.

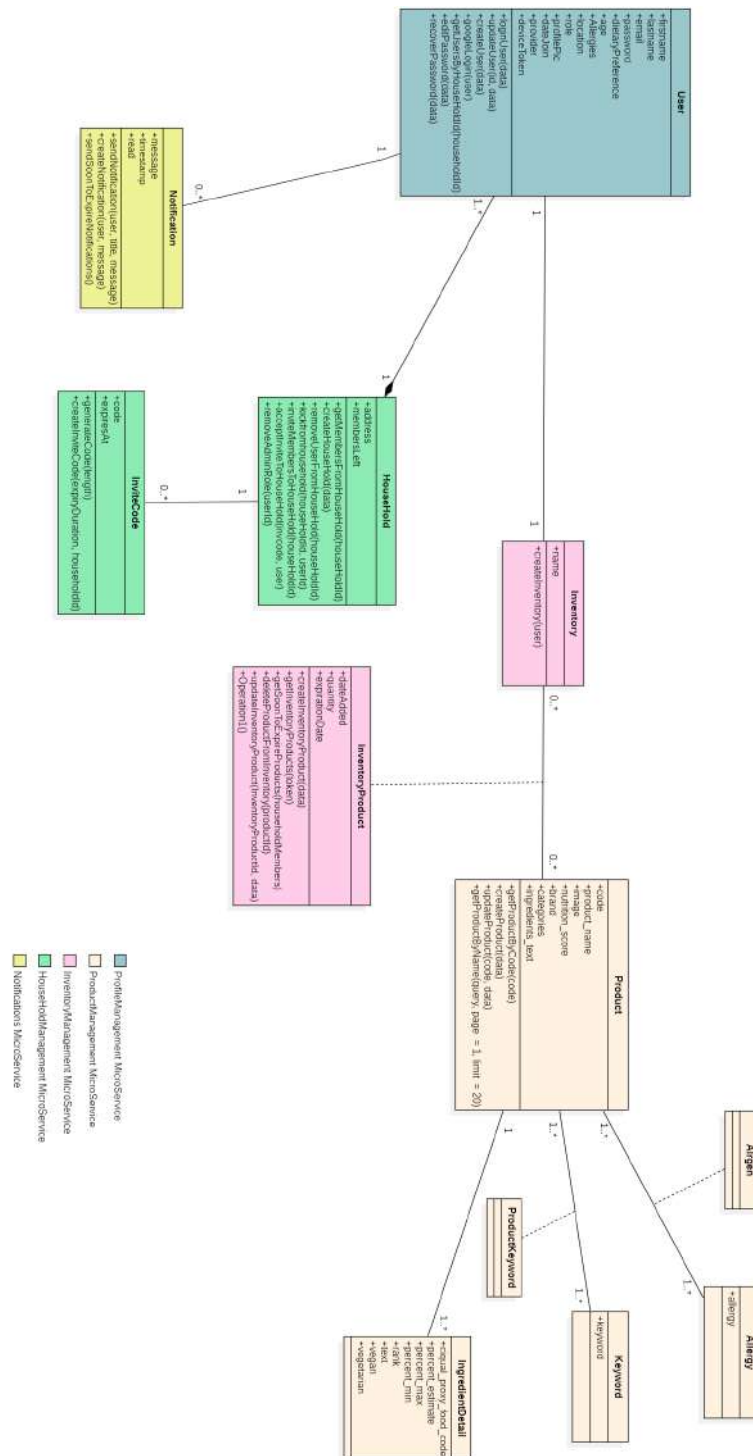


Figure 3.21: Class diagram Sprint 1

Class	Business Rule
User	- Belongs to one Household
Household	- Contains one or more users. - Would be destroyed if no Users exist anymore
User	- has one Inventory

Table 3.12: Business rules of sprint 1 class diagram

2.2 MongoDB Collection

In the figures below, we detail the design of MongoDB collections. Each collection is hosted in a separate database. The migration from the class diagram to the database design involved several changes, such as transforming relationships into embedded documents. Embedding connected data in a single document can reduce the number of read operations required to obtain data. In general, structure the schema so the application receives all of its required information in a single read operation.[26]. To improve query efficiency for our heavily queried Product class, we decided to embed the keywords and allergens, simplifying the many-to-many relationships and reducing the number of queries.



Figure 3.22: MongoDB Collections Schemes

2.3 Sequence Diagrams

In this section, we will explore various interactions of the application by employing different types of sequence diagrams.

2.3.1 Object sequence diagram «Sign Up»

This object sequence diagram presents the sign up process, focusing on the backend side and how objects interact to achieve the goal.

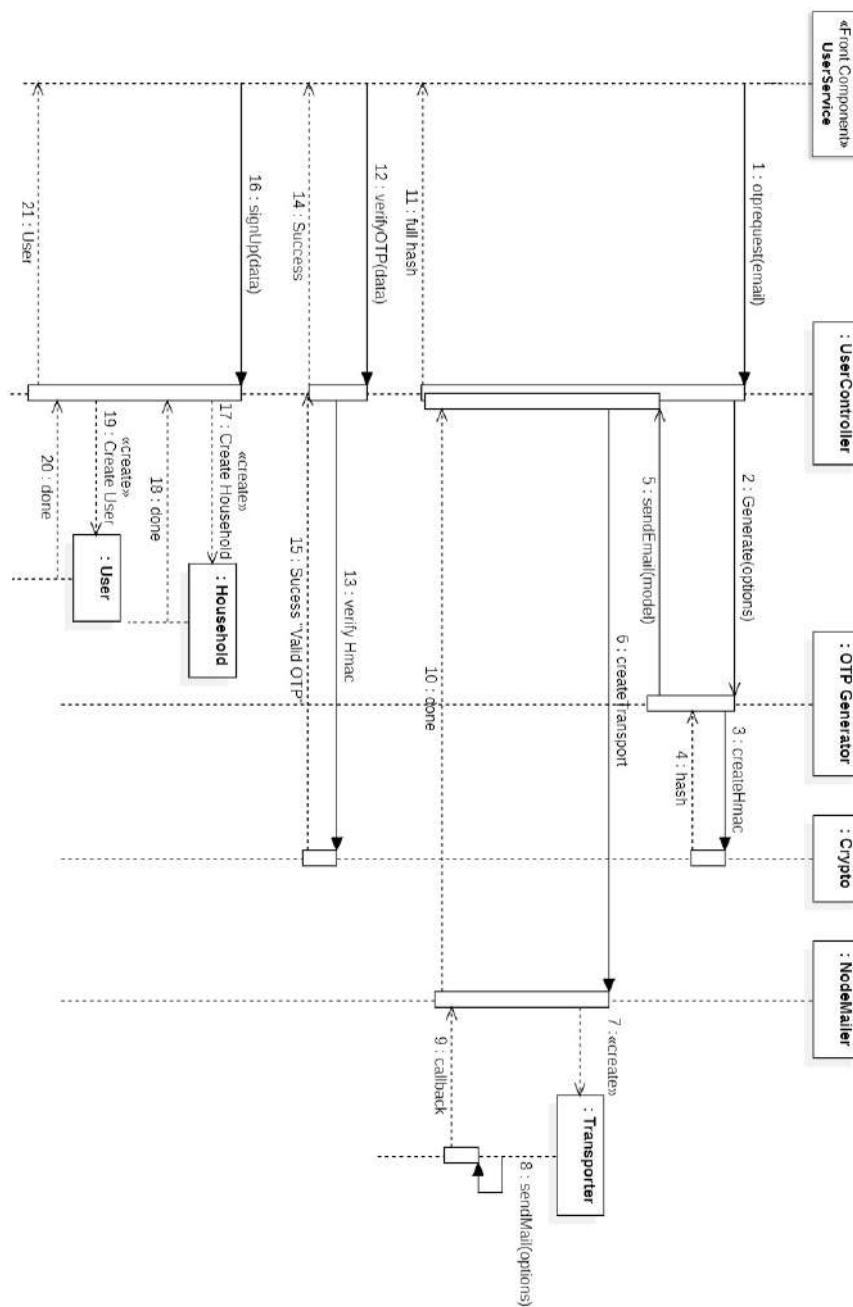


Figure 3.23: Object sequence diagram «Sign Up» «Backend Exclusive»

2.3.2 Sequence diagram «Authentication»

This Sequence diagram represents the process of logging in via Email or Google.

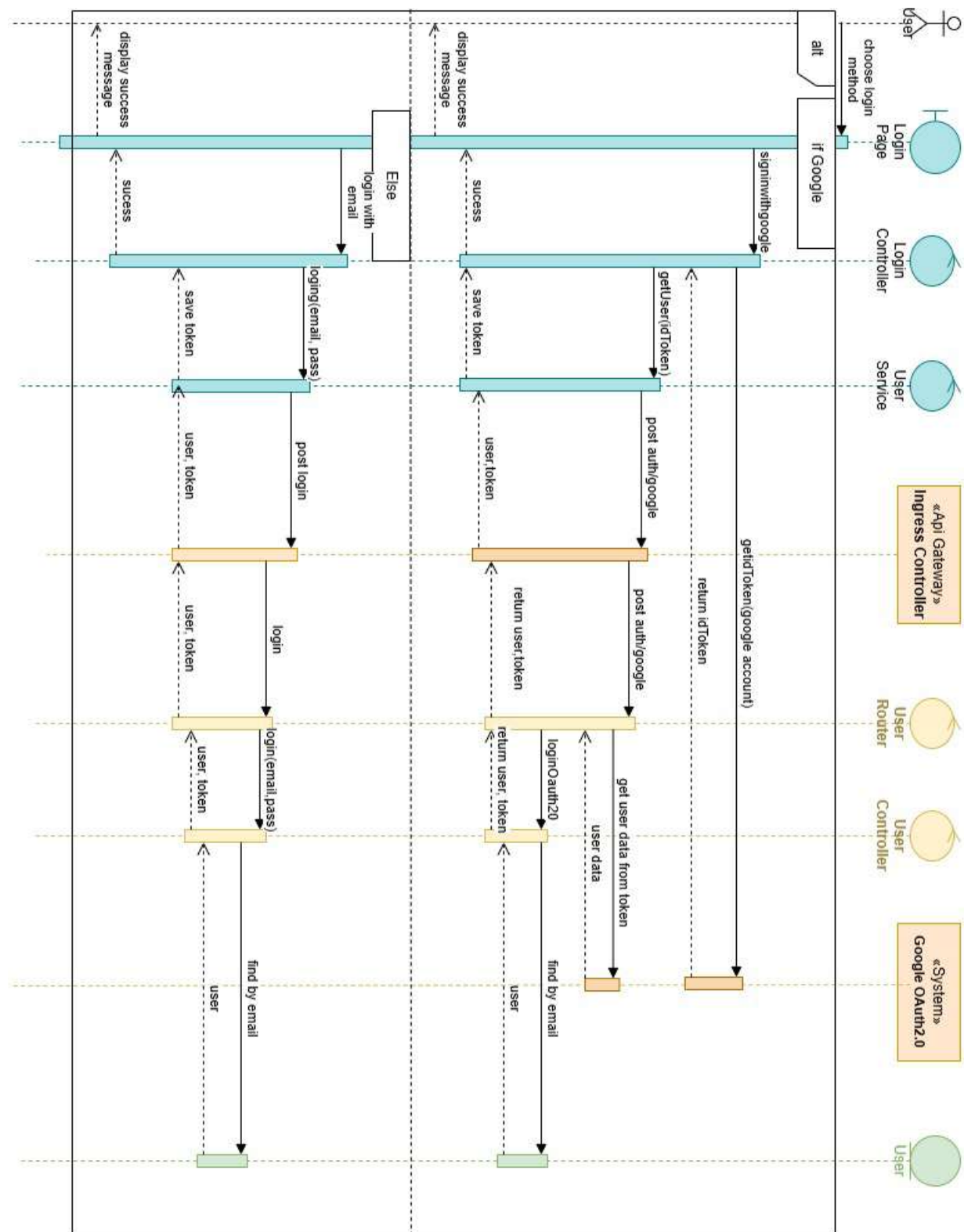


Figure 3.24: System sequence diagram «Authentication»

3 Tests

We conducted some unit tests during this sprint, we provided them in the sprint 1 appendix, section B.1.

4 Deployment of the first increment

Deployment is getting hardware and software up and running in the desired environment. It includes steps such as installation, configuration, pushing the code, testing, etc.

For more details on how we deployed our first increment please refer to the sprint 1 appendix.

5 Sprint Review

In this section, we will delve into the Sprint Review, showcasing the Product Increment delivered within the Sprint

5.1 Main User Interfaces

During the sprint review meeting, we showcased the components completed during this sprint, with the corresponding interfaces demonstrated below.



Figure 3.27: Welcome Screen Light Mode

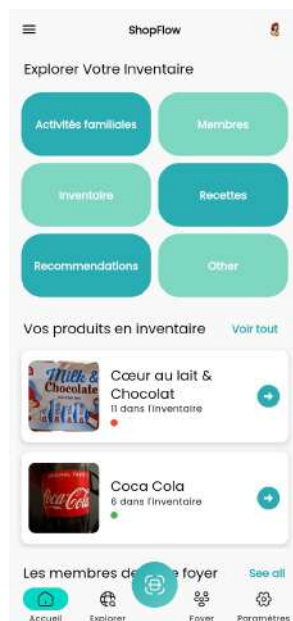


Figure 3.28: Home Page Light Mode

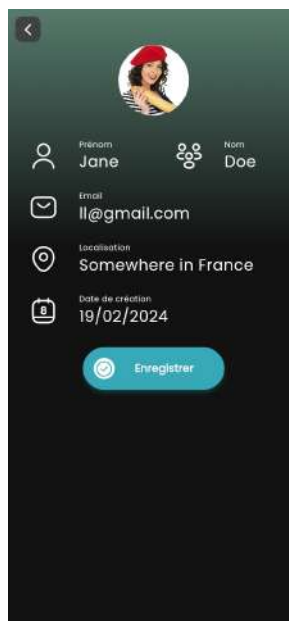


Figure 3.29: Edit Account Interface

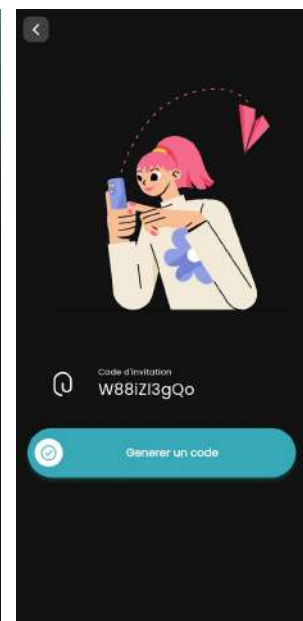


Figure 3.30: Invite Members Interface



Figure 3.31: Inventory Interface

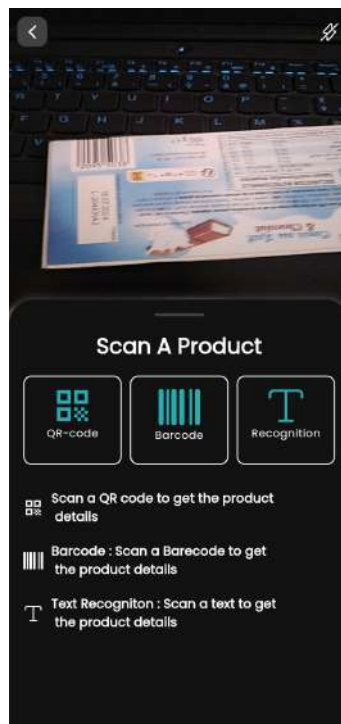


Figure 3.32: Scan a product interface



Figure 3.33: Product Details interface

The figure 3.27 represents the Welcome page interface of our application, you get the option to choose between logging in and signing up.

The figure 3.28 represents the Home page interface of our application, showing different information concerning inventory products that are nearing the expiry date, household members, the side drawer, and more.

The figure 3.29 represents the Account Details page interface of our application. Here, the user can change his informations.

The figure 3.30 represents the Invite Code interface of our application. You can generate a code that can be used later to invite other members to your household.

The figure 3.31 represents the Inventory that belongs to a user. This page holds information about the expiry date, the quantity, and the user who added that product.

The figure 3.32 represents the Scan a Product Screen. Here the user has different options: QR code, Bar-code, and Text Recognition.

The figure 3.33 represents the Details of a scanned/Searched product.

6 Sprint Retrospective

In the sprint retrospective, we will reflect on the achievements and challenges encountered during the sprint.

6.1 Retrospective Table

What went well	Challenges	Areas for improvement
-Interface Creation -Teamwork	-Kubernetes Environment configuration -Performance	-Kubernetes proficiency -Migrate to Asynchronous inter-communication

Table 3.13: Evaluation of Project

6.2 Sprint Burn Down Chart

In our project, our burn down charts are solely based on user stories development. The chart below shows our progress and remaining work throughout the sprint.

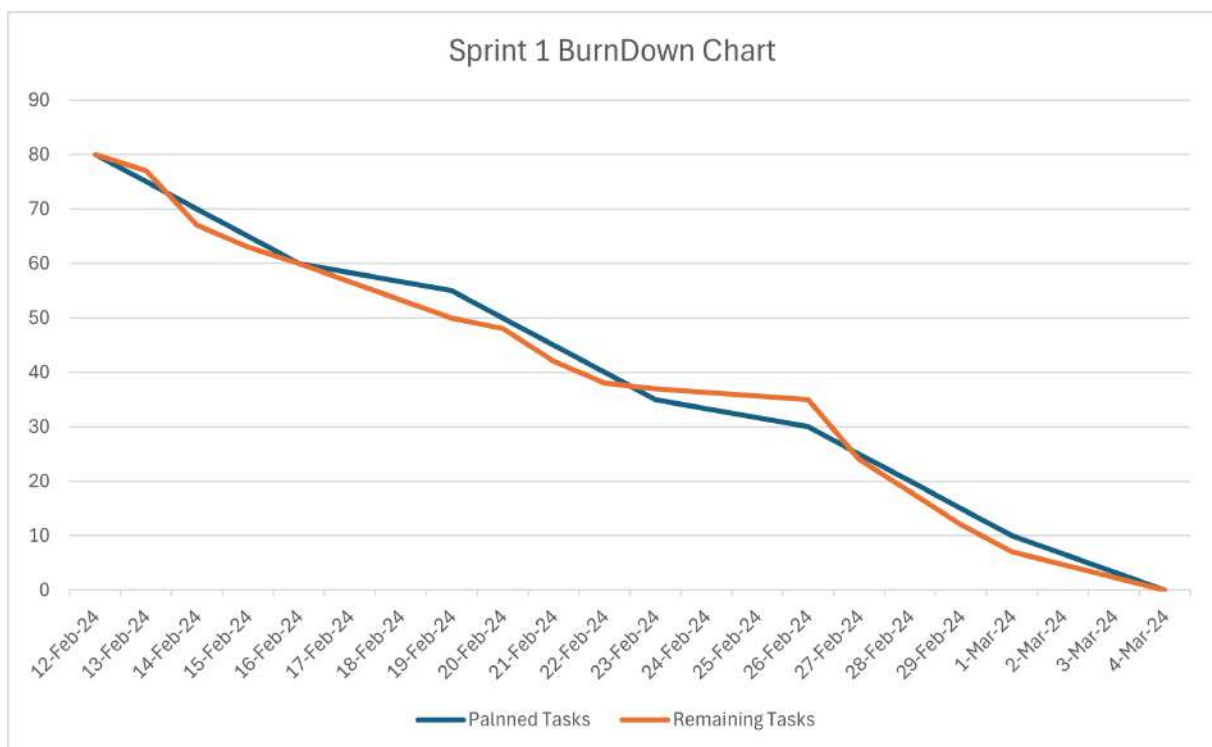


Figure 3.34: Sprint 1 Burn Down Chart

6.3 MicroService Inter-Communication

During this Sprint, We used an API-Rest based communication between the microservices. Upon further inspection and research, we found out that there are better ways to improve the performance of the inter-communication between the services by implementing a message broker. We will Dive deeper into this concept in the upcoming Sprint.

6.4 DoD Compliance Assessment

Throughout this sprint, we unfortunately haven't met all the criteria outlined in our Definition of Done. We encountered challenges in deploying this increment to the cloud. In the upcoming sprint, we will be committed to resolving this issue.

Throughout this chapter, we've detailed the user stories for Sprint 1. We then presented various mockups and established the design and different user interfaces. In the next chapter, we'll explore Sprint 2.

Chapter 4. Sprint 2 : GMS Authentication, ShoppingListManagement & RecommendationSystem

In the preceding chapter, we detailed each phase of the previous sprint's execution. Now, our attention shifts to the second sprint of our project. Maintaining consistency with the previous sprint, we'll delineate the crucial topics and phases for the most important features

1 Identification of needs

During this segment, we will present the sprint 2 backlog and main actors followed by some user interface Mock-ups.

1.1 Sprint Two Backlog

The core features of this sprint include the following :

- **Profile Management** : Continuing with the profile management feature, in the second sprint we will implement functionalities for logging out and password recovery, enhancing the authentication experience.
- **Inventory Management** : Continuing with the inventory management feature, in the second sprint we will implement functionalities for filtering user inventories, enhancing visibility for users.
- **Household Management** : Continuing with household management feature, in the second sprint we will implement functionalities for managing household members and editing inventory products.
- **Recommendation System** : This feature focuses on recipe recommendations tailored to the user's inventory. By analyzing the products available in their inventory,

the system suggests recipes that align with their ingredients. This personalized approach ensures that users receive relevant and practical recipe suggestions based on what they already have on hand.

- **Shopping List Management** : This feature delves into the shopping logic of our application. Upon scanning a product, users will have the option to add it to their shopping list, inputting details such as the expiration date, desired quantity and the desired supermarket. Once added, the item will be seamlessly added into their shopping list alongside previously added products.
- **GMS Account Management** : This feature's focus is on everything related to authentication for the GMS members, using different methods such as Google, moreover the GMS Manager can approve sign-up requests.

Feature ID	Sprint Feature	ID	User Story	Task			EST
S2_F1	Pro-file Management	1.3	As a user, I want to recover my password in case I forget it	Back-end	Password Recovery	Create the Password Recovery functionality in the userController.js and add it in the userRouter.js	1h
				Frontend	Password Recovery Interface	Create the modifyProfile functionality in the userService.dart	2h
						add the created modifyProfile functionality in the userController.dart	
						Create the recoverPassword.dart	
S2_F4	Household Management	4.6	As a household admin, I want to manage the members	Backend	Kick Member	Create the kick member functionality in the householdController.js and add it in the householdRouter.js	1h
					Change Role	Create the change role functionality in the householdController.js and add it in the householdRouter.js	1h
				Frontend	Members Management	Create the kick member functionality in the householdService.dart and add it in the householdController.dart	3h

						Create the change role functionality in the householdServiceer.dart and add it in the householdController.dart	
						create the householdScreen.dart to display members	
						create the householdMembersActions.dart screen	
S2_F5	Recommendation System	5.2	As a user, I want to receive recipe suggestions based on available inventory products.	Backend	Recipe Suggestions	Gather a comprehensive dataset of recipes with detailed ingredient lists	10h
						Collect data on user preferences and dietary restrictions	
						capture the user's available ingredients	
						Clean and preprocess the recipe dataset, ensuring consistency in ingredient naming and formatting, and convert it into a suitable format for machine learning	
						Choose an appropriate machine learning model for recipe recommendation	
						Train the selected model on the preprocessed data	
						Evaluate the model by assessing the performance of the trained model using validation datasets	
						Integrate the trained machine learning model with the backend system	
						Develop API endpoints for handling requests related to recipe suggestions based on user inventory	
						Similar tasks from previous sprint	3h

				Frontend	Recipe Screens	Create the recipeCard.dart to list the recipes in the explore.dart screen	
						Create to recipeDetails.dart screen	
S2_F6	Shopping list management	6.1	As a user, I want to create different shopping carts for different supermarkets	Back-end	Create Shopping Cart	Similar tasks from previous sprint	5h
				Frontend	Shopping-Cart Creation Form	Similar tasks from previous sprint	5h
						Create the shoppingList.dart screen	
						Add a button that opens a form for creation of shoppingList	
		6.3	As a user, I want to generate a QR-Code of my shopping list	Frontend	Generate Qr Code	create the functionality in the shoppingListController.dart	2h
						Create the QRShoppingList.dart screen	
						Create the functionality to long hold a specific shopping cart to generate qr code	
		6.6	As a user, I want to share shopping carts with household members	Back-end	Allow Access to shopping cart for other members	Create the functionality in the ShoppingListController.js and add it in the ShoppingListRouter.js	3h
				Frontend	Allow Access to shopping cart for other members Form	create the functionality in the shoppingListServiceoller.dart and add it in the shoppingListController.dart	2h

						Create the shoppingList-Card.dart	
						Add a form that allows access for other members	
S2_F8	GMS Account Management	8.1	As an Employee, I want to create an account using different methods	Backend	Email-Sign up	Similar tasks from previous sprint	½h
					Google Sign up	Similar tasks from previous sprint	2h
				Frontend	Sign Up Form	Create the sign up form component	1h
						Create the employee service	
						Create the sign up functionality in the service and implement it in the sign up component	
						add the sign up component in the app routing module for easier navigation	
		8.4	As a Manager, I want to manage employees	Backend	Employee Management	Create the delete employee functionality in the GMSAccountController.js and add it in the GMSAccountRouter.js	3h
						Create the accept employee functionality in the GMSAccountController.js and add it in the GMSAccountRouter.js	
						Create the assign role to employee functionality in the GMSAccountController.js and add it in the GMSAccountRouter.js	
						Create the get all employees functionality in the GMSAccountController.js and add it in the GMSAccountRouter.js	
						Create the dashboard component	3h
						Create the employee component	

				Frontend	Em- ployee Man- age- ment interface	Create the delete employee functionality in the employee service and implement in the employee component	
						Create the assign role to employee functionality in the employee service and implement in the employee component	
						Create the accept employee functionality in the employee service and implement in the employee component	
Global Tasks				Configure and implement the message broker for an asynchronous inter communication between the microservices		10h	
				Setup the recommendation engine python project		2h	
				Setup the shopping list management microservice		1h	
				Setup the GMS management system		1h	
				Setup the supermarket angular frontend by initializing the necessary modules and structure of the project		5h	
				Create the Sequence Diagram		1h	
				Create the Class Diagram		2h	
				Create a Deployment Diagram		2h	
				Create Test scripts		3h	
				Build and push a Docker image for each feature		½h	
				set up a cloud production environment and deploy the increment to an external kubernetes service		5h	
				create the kubernetes configuration YAML files(deployments, services, ingress controller. . .)		1h	
				set up a simple ci/cd pipeline to automate testing, building and deployment(locally)		1h	
Total Estimation						~120 hours	
For the full backlog, please refer to the sprint 2 appendix, section B.1 (only a snippet is shown here for report length reasons).							

Table 4.14: Sprint 2 backlog snippet

1.2 Identification of actors

The main actors of the second sprint are :

Actor	Description
User	This is the main user of the mobile application
GMS Manager	This is the main user of the supermarket dashboard application

Table 4.15: Identification of actors Sprint 2

1.3 User Interface Mock-ups

These are some of the user interface mock-ups for the second sprint:

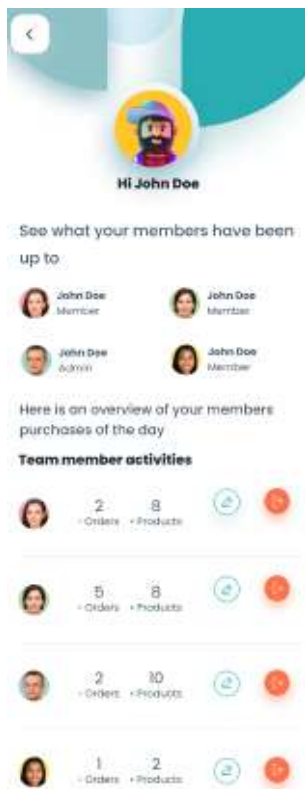


Figure 4.35: Household Interface mock-up

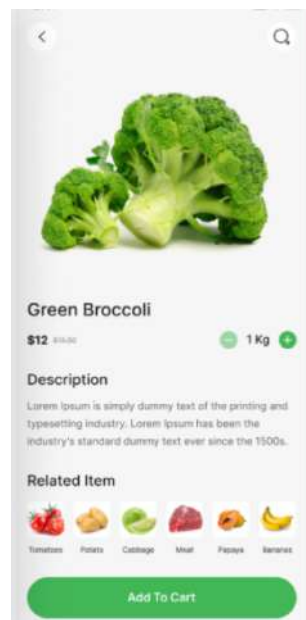


Figure 4.36: Add to cart Interface mock-up

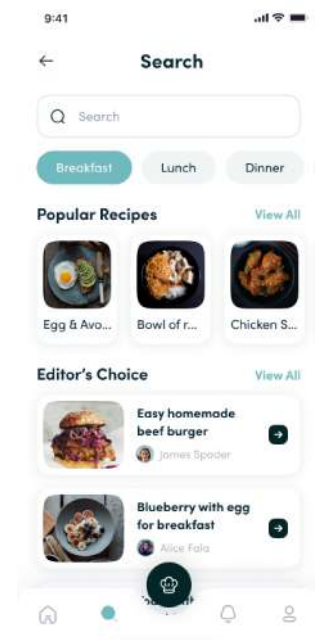


Figure 4.37: Recipe Suggestions Interface mock-up

The Figure 4.35 represents the Household interface. Here you'll find various information regarding your members, the products that they added or bought and the ability to kick them out if needed.

The Figure 4.36 represents a product Details Interface with the ability to add it to a shopping list.

The Figure 4.37 represents the explore page. Here you'll find recipe recommendations based on the user's interests.

2 Inter-Communication

As mentioned in the previous sprint retrospective, we migrated to another way of inter-communication which is via a message broker. We will present a quick overview of IPC technologies followed by a quick overview of broker-based messaging and the choice of said broker which will be RabbitMQ.

2.1 IPC Technologies

There are lots of different IPC technologies to choose from. [9] Services can use synchronous request/response-based communication mechanisms, such as HTTP based REST or gRPC.

Alternatively, they can use asynchronous, message-based communication mechanisms such as AMQP or STOMP. There are also a variety of different messages formats. Services can use human-readable, text-based formats such as JSON or XML. Alternatively, they could use a more efficient binary format such as Avro or Protocol Buffers.

We chose the AMQP as our IPC(Inter Process Communication) with the use of a message broker.

2.2 Overview of Broker-Based Messaging

A message broker is an intermediary through which all messages flow. A sender writes the message to the message broker, and it delivers it to the receiver.

An important benefit of using a message broker is that the sender doesn't need to know the network location of the consumer. Another benefit is that it buffers messages until the consumer is able to process them.

2.3 RabbitMQ

RabbitMQ [27] is a reliable and mature messaging and streaming broker, which is easy to deploy on cloud environments, on-premises, and on a local machine. It's also easy to integrate in our nodejs based microservices.

Other features include :

- **Interoperable :** RabbitMQ supports several open standard protocols, including AMQP 1.0 and MQTT 5. There are multiple client libraries available, which can be used with your programming language of choice, just pick one. No vendor lock-in!
- **Flexible :** RabbitMQ provides many options you can combine to define how messages go from the publisher to one or many consumers. Routing, filtering, streaming, federation, and so on, you name it.
- **Reliable :** With the ability to acknowledge message delivery and to replicate messages across a cluster, messages are safe with RabbitMQ.

With that said, we will include the modified **Design Sequence Diagram «Get Users Inventories»** from the **previous sprint** in the appendix of the second sprint.

3 Recipe Suggestions

In this section, we will provide a brief definition of Recommender Systems followed by a discussion about the model of the system we used during this sprint, we will discuss one more model in the upcoming chapters.

3.1 Recommender system

The topic of recommender systems[28] gained increasing importance in the nineties, as the Web became an important medium for business and e-commerce transactions. It was recognized early on that the Web provided unprecedented opportunities for personalization, which were not available in other channels. In particular, the Web provided ease in data collection and a user interface that could be employed to recommend items in a non-intrusive way.

3.2 Content-Based Recommender Systems

In content-based recommender systems[28], the descriptive attributes of items are used to make recommendations. The term “content” refers to these descriptions. In content-based methods, the rating behavior of users is combined with the content information available in the items.

In content-based methods, the item descriptions, which are labeled with ratings, are used as training data to create a user-specific classification or regression modeling problem.

For each user, the training documents correspond to the descriptions of the items they have bought or correspond with their preferences.

The class (or dependent) variable corresponds to the specified rating behavior. These training documents are used to create a classification or regression model, which is specific to the user at hand. This user-specific model is used to predict whether the corresponding individual will like an item for which their rating or buying behavior is unknown.

Content-based methods have some advantages in making recommendations for new items, when sufficient rating data are not available for that item.

The table below explains the conceptual goals of Content-based methods :

Ap- proach	Conceptual Goal	Input	Our Case
Content based	Give me recommendations based on the content (attributes) I have favored in my past ratings and actions.	User ratings + item attributes	Provide recommendations based on the products in the user's inventory and their dietary preferences by matching ingredients with the inventory products and the ingredient types(vegan, halal, etc...) in the recipes

Table 4.16: conceptual goals of Content-based methods

4 GMS Application Prototype

To test the checkout interactions between a supermarket and a user, we needed a GMS system for interaction. Since a real partner GMS is not yet provided by our Product Owner, we developed an application prototype to simulate a real supermarket experience and achieve the desired interactions.

Within this prototype, users can add desired products to their shopping lists that are mapped to the supermarket.

We developed a whole GMS system for stock management. This system contains certain features including charts for specific requirements and the ability to create promotional offers. These latter features will be more explored during the next sprints.

Our system is built to easily adapt to other supermarket setups in the future. Its design allows for smooth integration with different supermarket systems.

Given that this application is a prototype and won't transition into production, we've opted for a monolithic architecture. This choice simplifies development and testing processes, allowing us to focus on quickly building and validating the core functionalities of the system without the complexities associated with distributed systems.

5 Design

This section is dedicated to the development of class and sequence diagrams for this sprint.

5.1 Class Diagram

We will present the various Class Diagrams, including the main one and those of several microservices.

5.1.1 Main Class Diagram

This is the main Class Diagram of our application showcasing the different classes of different microservices.

Sprint 2 : GMS Authentication, ShoppingListManagement & RecommendationSystem

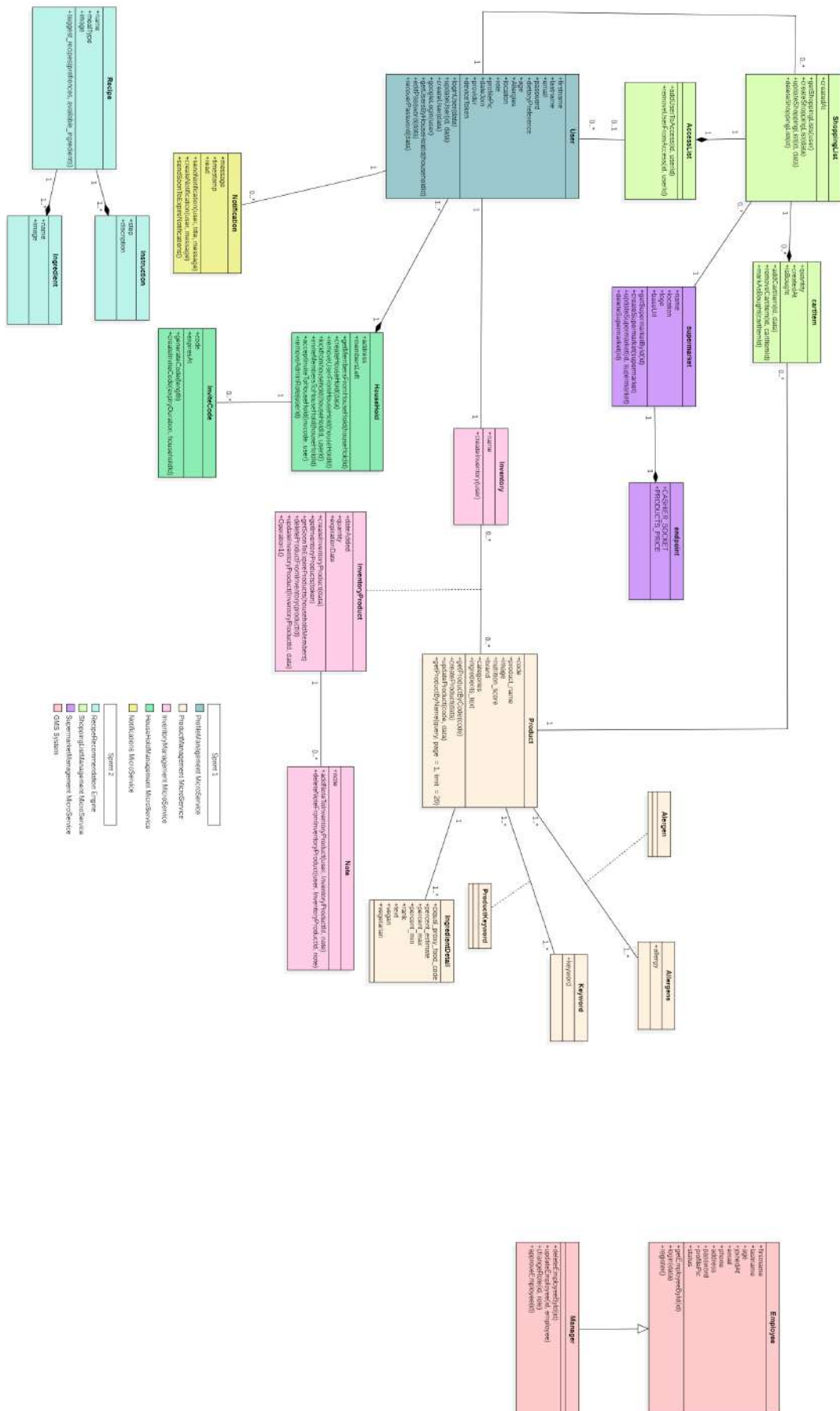


Figure 4.38: Class Diagram Sprint 2

Class	Business Rule
ShoppingList	<ul style="list-style-type: none"> - Can be owned by only one User - Can be accessed by multiple Users
Endpoint	- Would be destroyed if the Supermarket is undefined

Table 4.17: Business rules of sprint 2 class diagram

5.1.2 Recipe Recommendation Class Diagram

This is the Class Diagram of our Recipe Recommendation microservice.

We chose to separate an Ingredient class from the Product class because an ingredient represents a general category, while a product is more specific. For recommendations, using general names is more effective.

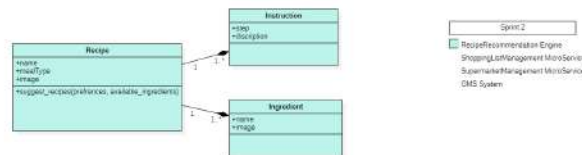


Figure 4.39: Recipe Recommendation Class Diagram Sprint 2

5.1.3 GMS Class Diagram

This is the class diagram for the GMS System

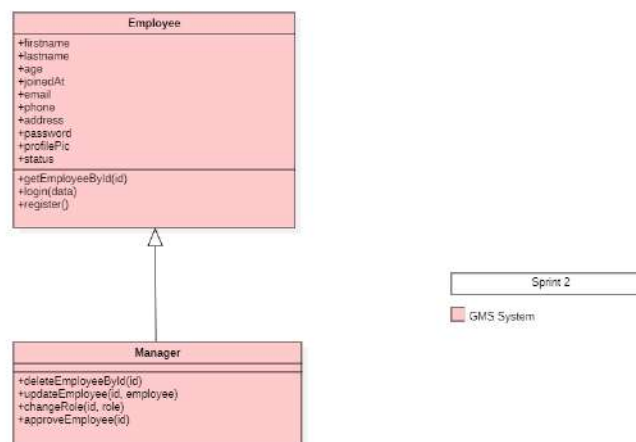


Figure 4.40: GMS Class Diagram Sprint 2

5.2 Object Sequence Diagram «Add CartItem»

This Object Sequence Diagram showcases the process of adding a product to a shopping list.

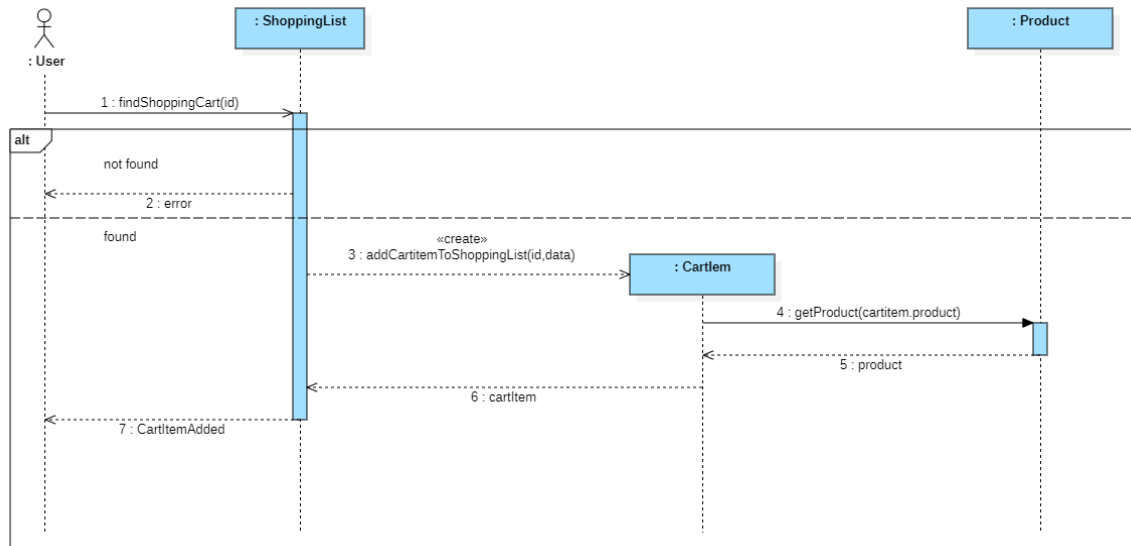


Figure 4.41: Add CartItem Object Sequence Diagram

5.3 Deployment Diagram

Below is the deployment Diagram for the second sprint. The Frontend being the mobile application "ShopFlow" and the web application "GMS Management", both of which send and receive requests to the Backend via The ingress Controller, The Microservices are registered in said controller. they communicate with each other via the Message Broker «RabbitMQ» through channels and queues. Each microservice inside the Kubernetes cluster has three pods, each with its own unique MongoDB database. Due to limited resources, we created a cluster for each database instead of grouping them all in one cluster.

Sprint 2 : GMS Authentication, ShoppingListManagement & RecommendationSystem

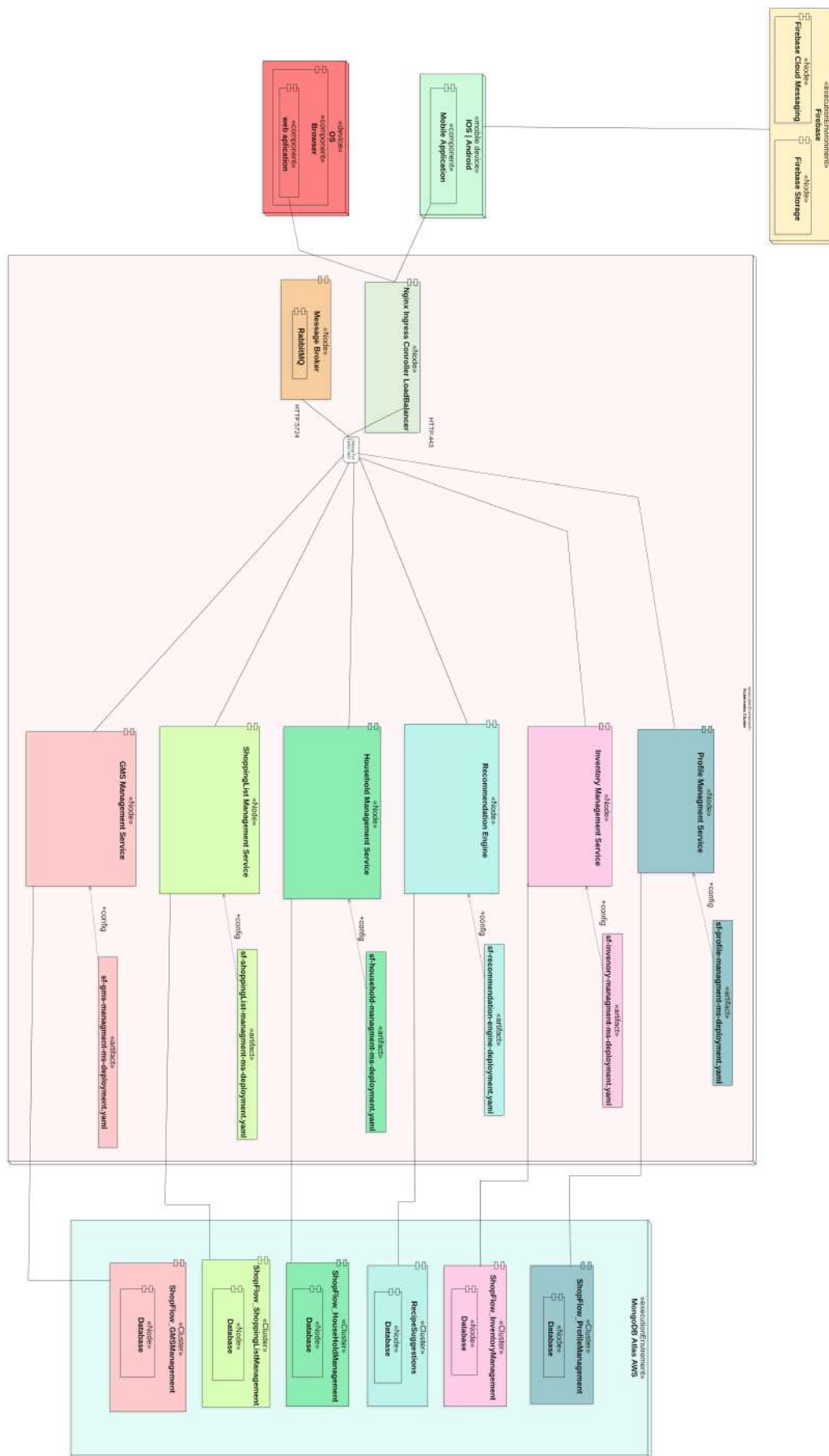


Figure 4.42: Deployment Diagram Sprint 2

6 Tests

We conducted some unit tests during this sprint, we provided them in the sprint 2 appendix, section C.1.

7 Deployment of the Second Increment to the Cloud

In this section, we will detail the process of deploying the second increment to the production cloud environment.

7.1 Setting Up the infrastructure in Azure

After selecting Azure as our preferred cloud provider and cluster manager, we will now outline the design of our final cloud infrastructure in the figure below.

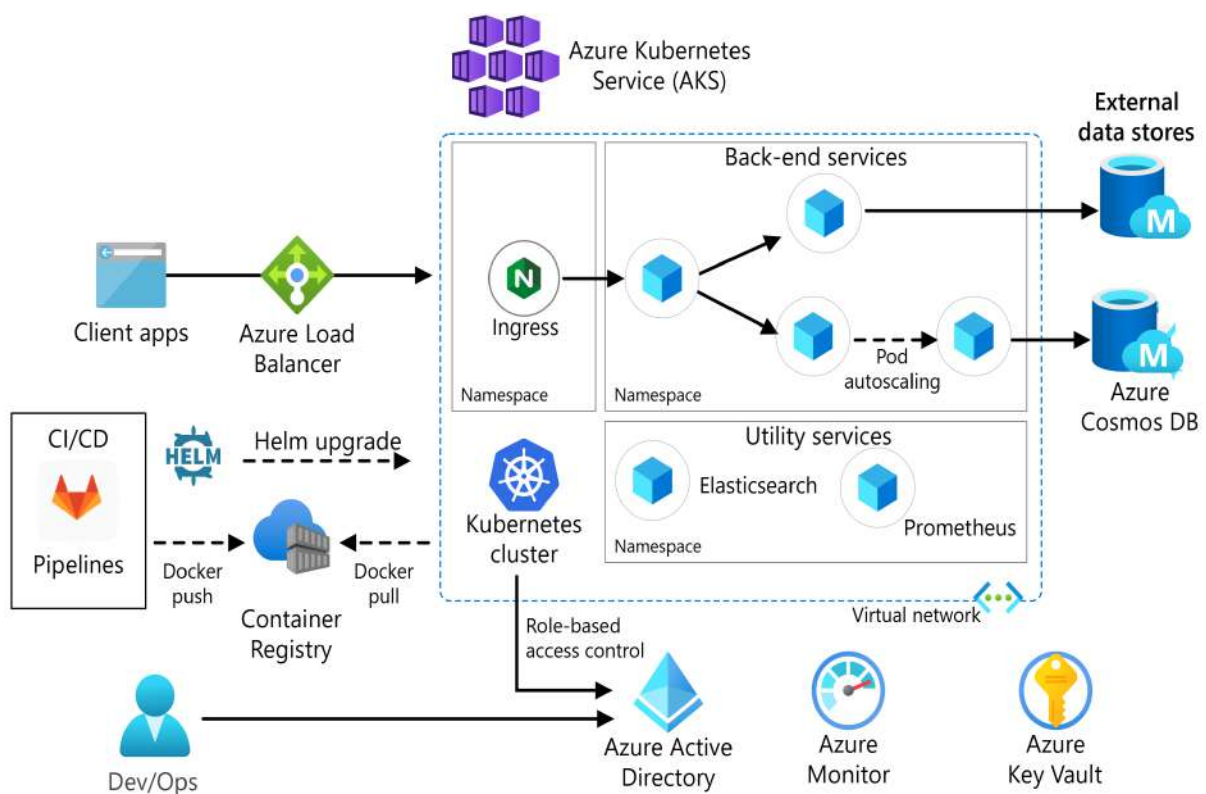


Figure 4.43: Complete Cloud Infrastructure

7.2 Production CI/CD Pipeline

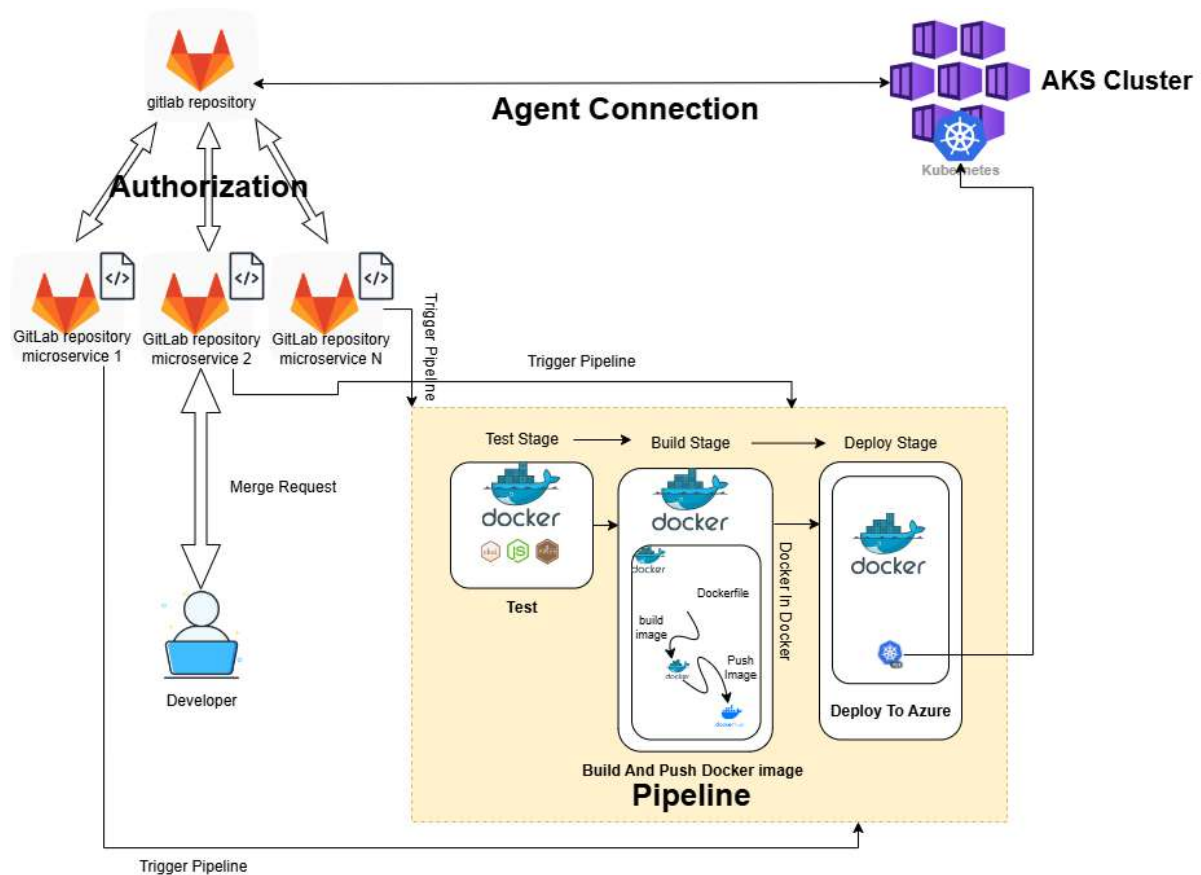


Figure 4.44: Complete Production CI/CD Pipeline

Figure 4.44 represents the CI/CD pipeline for how we handle updates in the production environment. Now, instead of just delivering changes regularly, we're moving to continuous deployment. This means whenever someone makes a change in the main branch, it automatically triggers the production process. So, users will get updates right away without any delays.

8 Sprint Review

In this section, we will delve into the Sprint Review, showcasing the Product Increment delivered within the Sprint.

8.1 Main Interfaces

In this section, we'll review the primary user interfaces for our key users in the current sprint.

8.1.1 User Interfaces

Below are some user interfaces developed during this sprint :

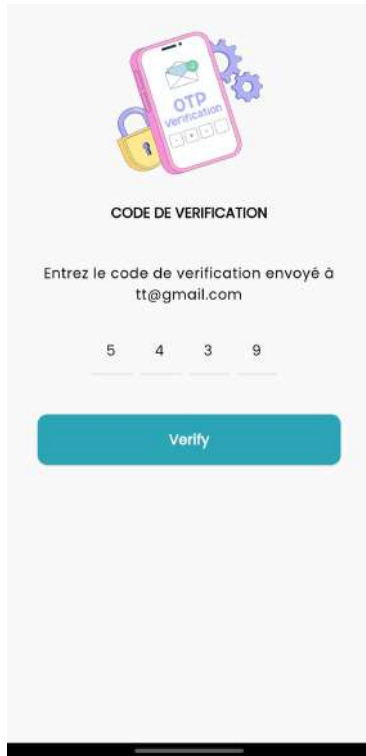


Figure 4.45: Entered Verification Code For Password Recovery

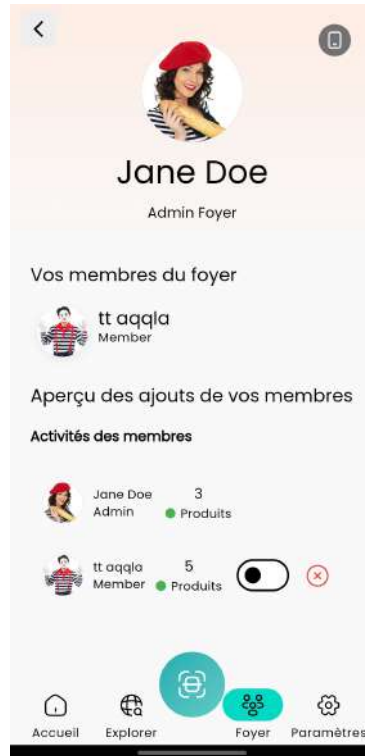


Figure 4.46: Household

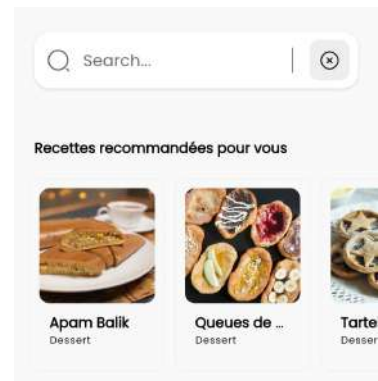


Figure 4.47: Recipe Recommendation List

The Figure 4.45 represents the interface where the user can input a received verification code to confirm the email.

The Figure 4.46 showcases the household interface, providing access to household members and options for management, including assigning admin privileges or removing members.

The Figure 4.47 showcases the Recipe Recommendation List on the Explore page, listing different recipes based on user preferences and inventory products.

The Figure 4.48 presents the interface for creating a shopping list. a user has the option to choose the desired supermarket and select household members to whom they grant access.

The Figure 4.49 showcases the interface for adding a product to a desired shopping list, a user can choose the quantity as well.

The Figure 4.50 Presents the shopping list interface. Here, the user can view other authorized users for their shopping list as well as a snippet of the added products.

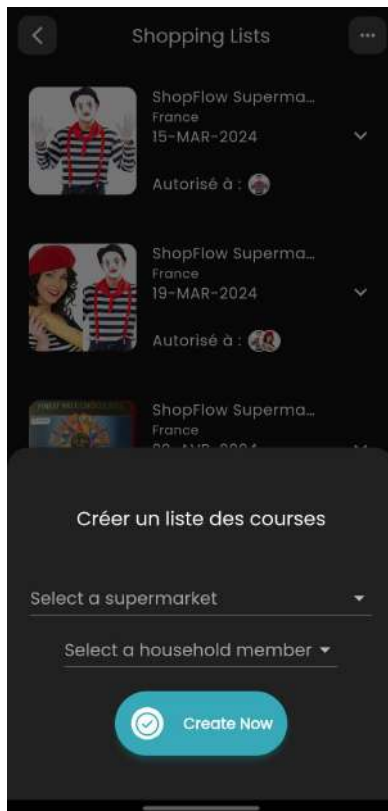


Figure 4.48: Create a shopping List

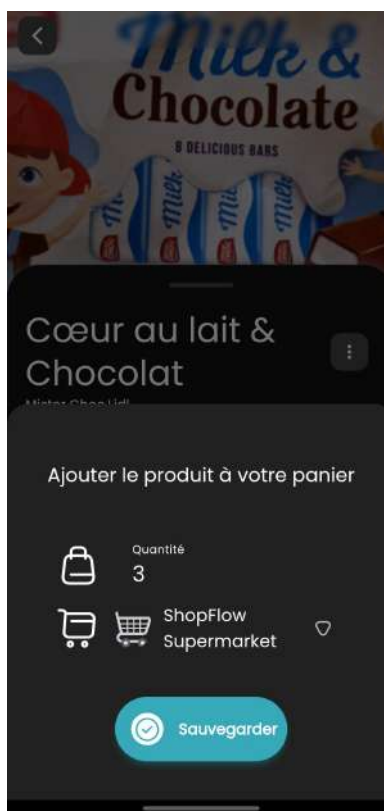


Figure 4.49: Add product to Shopping List

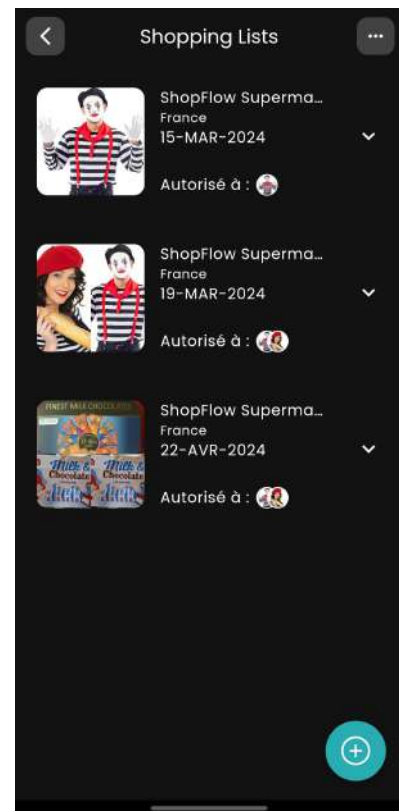


Figure 4.50: Shopping Lists Interface

8.1.2 GMS Interfaces

Here are the GMS user interfaces created in this sprint.

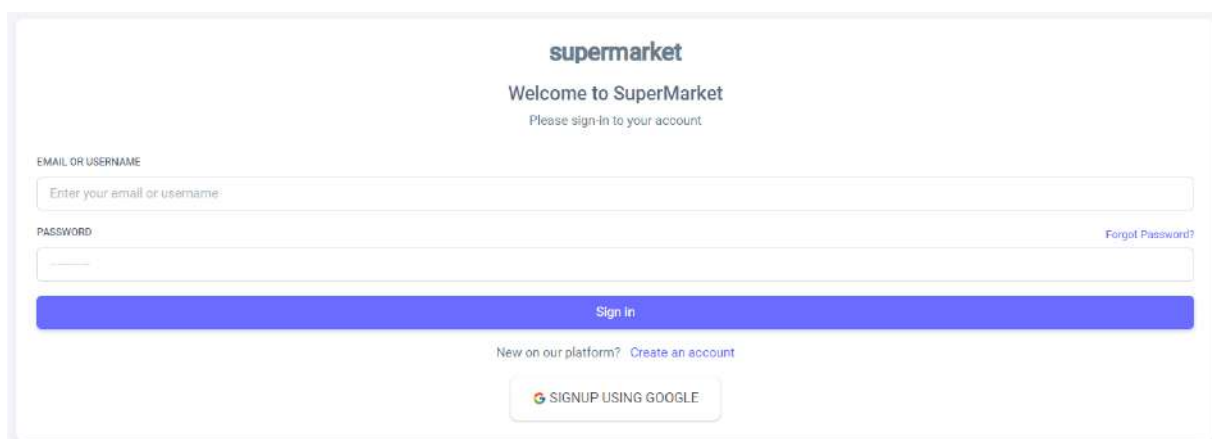
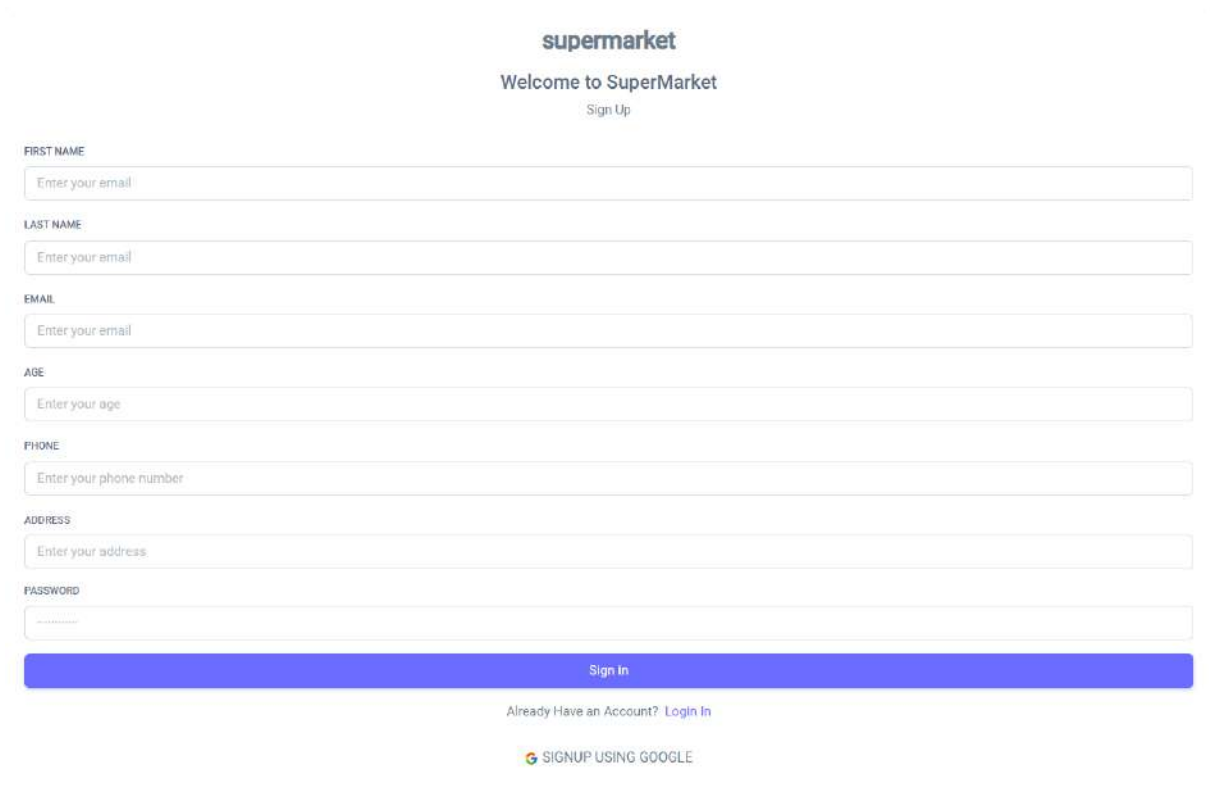


Figure 4.51: GMS Login

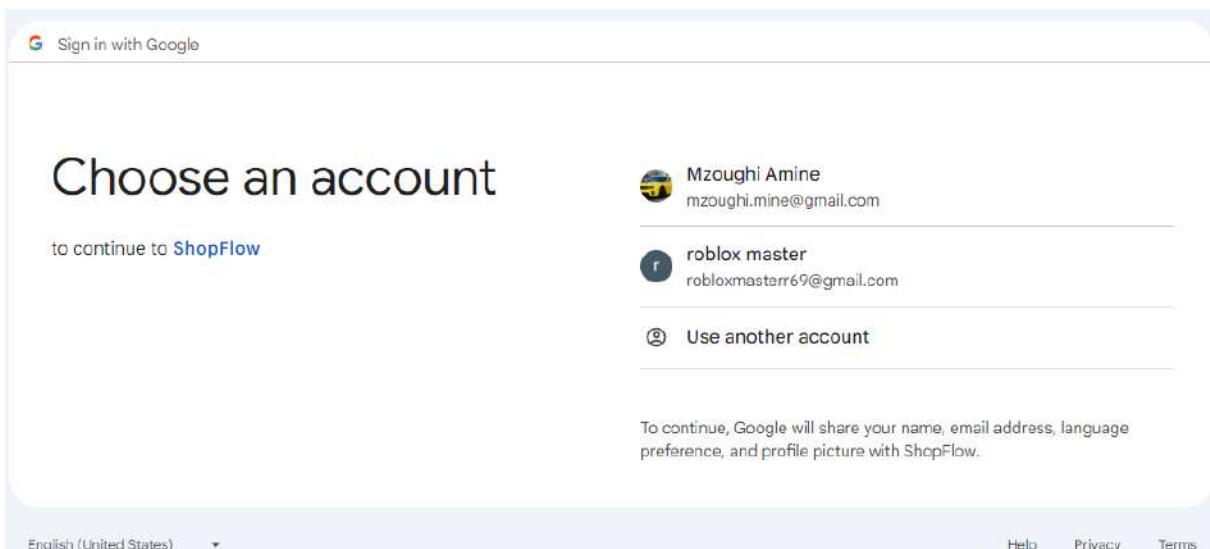
The Figure 4.51 is the GMS login Interface, where an Employee, Cashier or Manager can be authenticated



The image shows a web form for signing up to 'supermarket'. At the top, it says 'supermarket' in a bold, lowercase font, followed by 'Welcome to SuperMarket' and a 'Sign Up' link. Below this are several input fields: 'FIRST NAME' (placeholder: 'Enter your email'), 'LAST NAME' (placeholder: 'Enter your email'), 'EMAIL' (placeholder: 'Enter your email'), 'AGE' (placeholder: 'Enter your age'), 'PHONE' (placeholder: 'Enter your phone number'), 'ADDRESS' (placeholder: 'Enter your address'), and 'PASSWORD' (placeholder: 'Enter your password'). A large blue button labeled 'Sign In' is positioned below the password field. Underneath the button, there is a link 'Already Have an Account? Login In' and a 'SIGNUP USING GOOGLE' button with the Google logo.

Figure 4.52: GMS Signup

The Figure 4.52 is the GMS sign up Interface, where an Employee can sign up. Upon Signing up, the employee must wait for the manager's approval.



The image shows a Google sign-in interface. At the top, it says 'Sign in with Google'. Below this, the main heading is 'Choose an account' with the subtext 'to continue to ShopFlow'. On the right side, there are two account options: 'Mzoughi Amine' with email 'mzoughi.mine@gmail.com' and 'roblox master' with email 'robloxmasterr69@gmail.com'. Below these is a link 'Use another account'. At the bottom, there is a disclaimer: 'To continue, Google will share your name, email address, language preference, and profile picture with ShopFlow.' The footer includes 'English (United States)' with a dropdown arrow, and links for 'Help', 'Privacy', and 'Terms'.

Figure 4.53: GMS Google Sign up/ Login in

The Figure 4.53 is the GMS Google sign in Interface, where an Employee, Cashier or Manager can sign in.

8.2 Backlog Update

Unfortunately, since we didn't complete all of our user stories, the table below represents the update for the remaining stories

User Story	Action	Reason
8.3 : As an Employee,I want to join a GMS	Move to next sprint	Time
8.5 : As a Cashier, I want to synchronize my mobile app session with the GMS system	Move to next sprint	Time
9.1 As a Cashier, I want to scan clients' shopping cart's generated QR code to retrieve the products in question	Move to next sprint	Time

Table 4.18: Backlog Update

9 Sprint Retrospective

In the sprint retrospective, we will reflect on the achievements and challenges encountered during the sprint.

9.1 Retrospective Table

The table below represents our key achievements, challenges, and opportunities for improvement for this sprint.

What went well	Challenges	Areas for improvement
-Kubernetes proficiency -Teamwork	-Recommendation System Development -Socket Integration between different applications	-Time Management

Table 4.19: Evaluation of Sprint

9.2 Sprint Burn Down Chart

The chart below shows our progress and remaining work throughout the sprint.

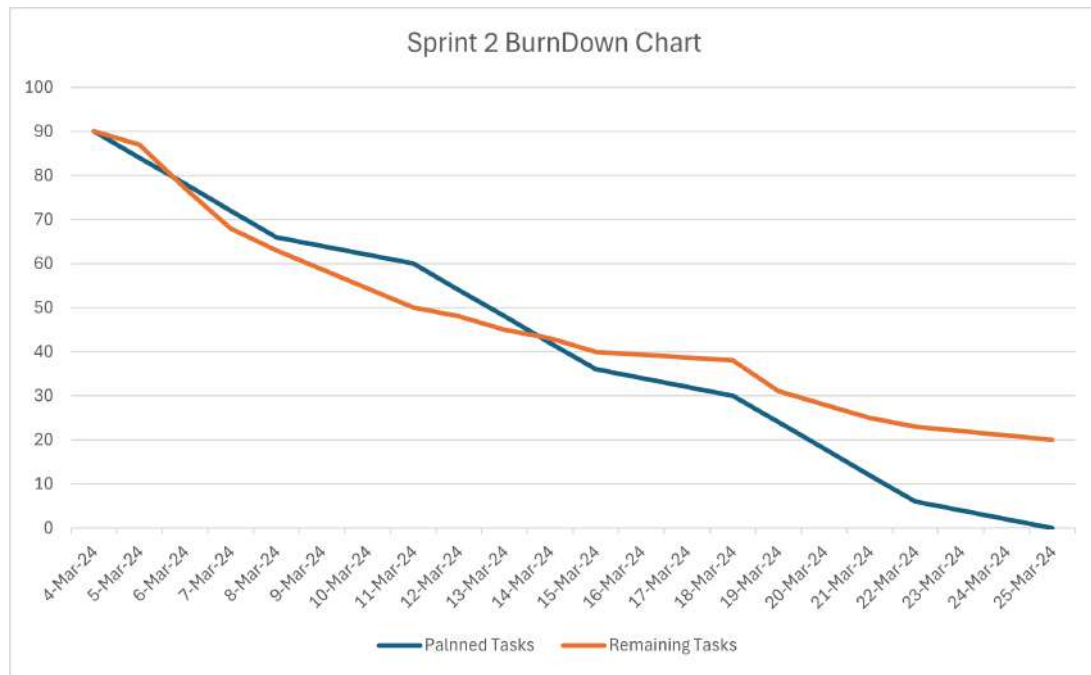


Figure 4.54: Sprint 2 Burn Down Chart

9.3 DoD Compliance Assessment

Throughout this sprint, we unfortunately haven't met all the criteria outlined in our Definition of Done. We encountered challenges in Finishing all user stories. In the upcoming sprint, we will make sure to resolve the time management issue.

Throughout this chapter, we've detailed the user stories for Sprint 2. We then presented various mockups and established the design and different user interfaces. In the next chapter, we'll explore Sprint 3.

Chapter 5. Sprint 3 : GMS Management & Checkout Interactions

In the preceding chapter, we detailed each phase of the previous sprint's execution. Now, our attention shifts to the third sprint of our project. Maintaining consistency with the previous sprint, we'll delineate the crucial topics and phases for the most important features.

1 Identification of needs

During this segment, we will present the sprint 1 backlog followed by the use case diagram and finishing with some user interface Mock-ups.

1.1 Sprint Three Backlog

The core features of this sprint include the following :

- **Checkout Interactions** : This feature's main focus is on everything related to checkout interactions between a user and a cashier. A QR code will be generated based on the user's input, which the cashier will scan to display the products on their computer. To start, the cashier initiates a session by scanning a QR code on their machine.
- **Inventory Management** : Continuing with the inventory management feature, in the third sprint we will implement functionalities for attaching notes for inventory products.
- **Shopping List Management** : This feature delves into the shopping logic of our application. Upon scanning a product, users will have the option to add it to their shopping list, inputting details such as the expiration date, desired quantity and the desired supermarket. Once added, the item will be seamlessly added into their shopping list alongside previously added products.
- **GMS Account Management** : Continuing with the GMS Account management feature, in the third sprint we will implement functionalities for the manager to assign a role to an employee.

- **Stock Tracking** : This feature focuses on stock tracking for a supermarket. Stock will be automatically updated upon a checkout interaction. Moreover, A chart will be implemented to visualize clients consumption trends to better optimize stock.
- **Shop Flow Admin Management** : This feature's focus is on everything related to authentication for the Admin of our Shopflow Mobile Application. Additionally, the admin can add supermarkets to the system, allowing users to access and shop from them.

Feature ID	Sprint Feature	ID	User Story	Task			EST
S3_F9	Check-out Interactions	9.1	As a Cashier, I want to scan clients shopping cart's generated QR code to retrieve the products in question	Backend	Cashier Carts Scanning Session	Create the carts socket listeners and emitters After establishing a connection	2h
						Add the scanning products by barcode functionality to the ProductController.js and use it in the socket listener	
				Frontend	Cashier Scanning Carts Mobile Session	Add the functionality's socket listeners and emitters to scan shopping carts in the socket service	5h
						Add the functionality to scan carts in the cashierSessionController.dart	
						Create an interface to scan carts generated QR codes	
					Cashier Scanning carts Web session	Add the corresponding listeners to the cashier service	4h
						Create a checkout component and listen to the scanning events emitted from the mobile synchronized session and show the transaction	
				Backend	Approve transaction	Similar tasks from previous sprint	5h
						Add the approve event to the cashier socket	

		9.3	As a cashier, I want to finalize the transaction		Abort Transaction	Create the Abort transaction functionality in the transactionController.js and add it in the transactionRouter.js	
				Frontend	Approve transaction	Add the functionality in the cashier service and implement it in the checkout component	
					Abort transaction	Add the functionality in the cashier service and implement it in the checkout component	
S3_F8	GMS Account Management	8.5	As a Cashier, I want to synchronize my mobile app session with the GMS system	frontend	Cashier session Socket	Create a socket server in the GMS Microservice	5h
						Save the GMS cashier socket path to the corresponding supermarket	
						Create the web socket events listeners and emitters for joining rooms and synchronizing sessions	
				backend	Cashier Session Web Interface	Create the cashier service	5h
						Create the cashier component and add a QR code dialogue	
						Add the corresponding events and listeners to the component	
					Cashier Session Mobile Interface	Similar tasks from previous sprint	5h
						Create the cashier scan QR Code screen and redirect when the connection is established	
S3_F10	Stock tracking	10.1	As a Manager, I want the stock to be updated automatically after transactions	Back-end	Remove quantity from stock	Create the functionality that automatically updates the stock upon approving a transaction in the stockController.js and add it in the stockRouter.js.	3h

		10.2	As a Manager, I want to receive out-of-stock alerts	Backend	Get Alerts	Create the StockAlert.js model	2h
						Create the functionality that creates an alert when a product's quantity reaches 0 in the stockController.js and add it in the stockRouter.js	
				Frontend	Alerts table	Create the alerts component	2h
						Add the functionality in the stock service and implement it in the alerts component	
		10.4	As a Manager, i want to receive analytics of clients consumption trends to better optimize stock management	Backend	analyzing clients' consumption trends	create Extract.py and query the database to extract relevant client-buying data	3h
						create Transform.py and Preprocess the extracted data	5h
						create Load.py and Aggregate the preprocessed data	3h
						create the mongodb chart	2h
				Frontend	visualizing clients consumption trends	integrate the chart in the dashboard	2h
13	Shop Flow Admin Management	13.2	As an admin, I want to add a supermarket and its API paths to the shop flow system to allow users to shop there	Backend	Add supermarket	Similar tasks from previous sprint	2h
				Frontend	Add supermarket details	Create the add supermarket component	3h
						Create the functionality in the shopFlowAdmin service and implement it in the add supermarket component	
						Similar tasks from previous sprint	
						Create the supermarket-Card.dart view	

					Display super- mar- kets in the mobile application	Create the supermar- ketList.dart screen Create the functionality in the supermarketSer- vice.dart and implement it in the supermarketCon- troller.dart	
Global Tasks				Setup the Supermarket management microser- vice		2h	
				Setup the ShopFlow Admin angular frontend.		1h	
				Create the Class Diagram		2h	
				Create the Activity Diagram		2h	
				Create the Package Diagram		2h	
				Create Tests scripts		1h	
				create the kubernetes configuration YAML files(deployments, services, ingress controller. . .)		1h	
				set up a ci/cd pipeline to automate testing, building and deployment(locally)		1h	
Total Estimation						~120 hours	
For the full backlog, please refer to the sprint 3 appendix, section C.1 (only a snippet is shown here for report length reasons).							

Table 5.20: Sprint 3 backlog snippet

1.2 Use Case Diagram

Below is the detailed use case diagram of this sprint, which explains further the features discussed earlier

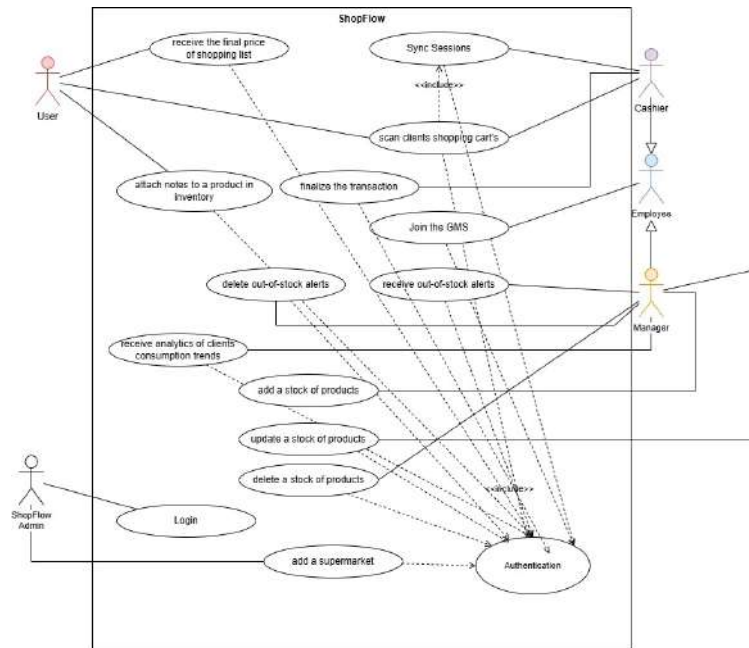


Figure 5.55: Sprint 3 Use Case Diagram

1.3 User Interface Mock-ups

These are some of the user interface mock-ups for the third sprint:

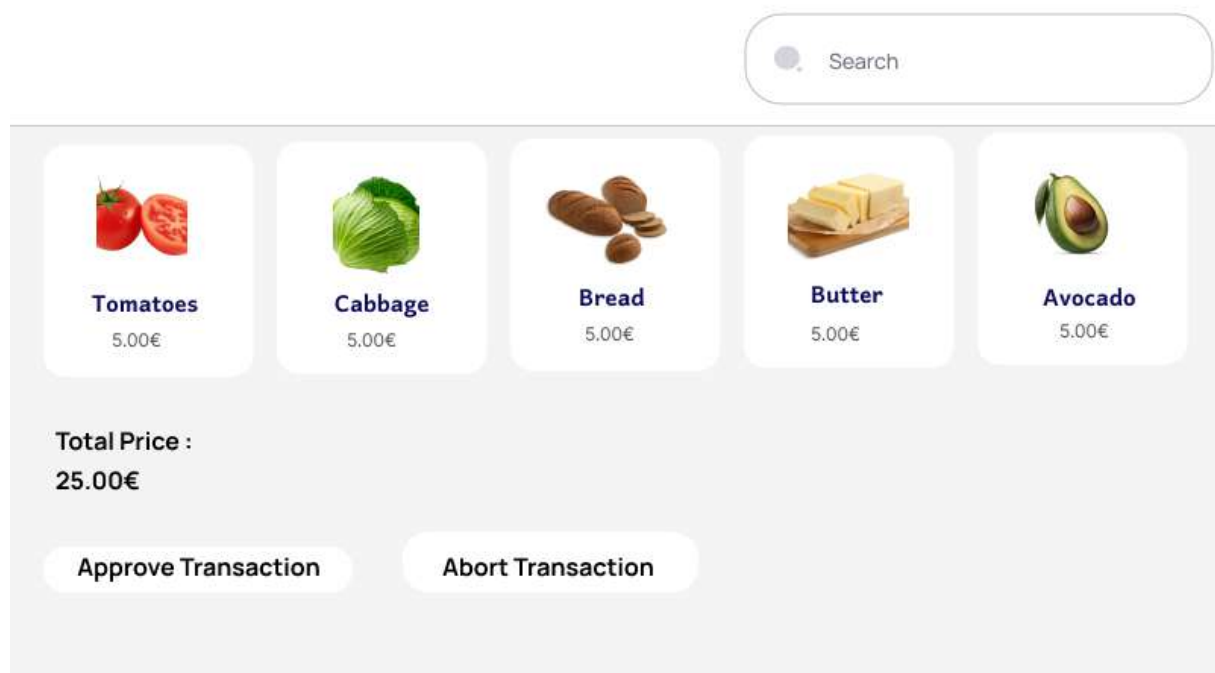


Figure 5.58: Cashier Scanned Products List



Figure 5.56: Final Price With Qr code of a user’s shopping List Mock-up

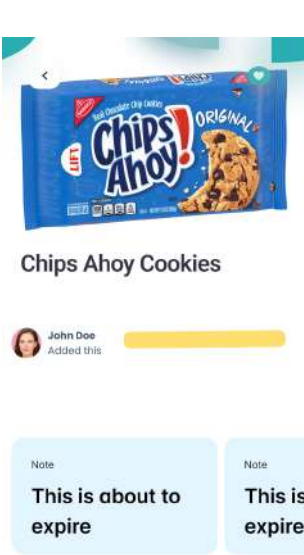


Figure 5.57: Inventory Product Notes Mock-up

Figure 5.56 illustrates the total price of all items in the shopping list, accompanied by a QR code to be scanned by the cashier.

The Figure 5.57 illustrates the user’s capability to attach notes to a selected inventory product.

The Figure 5.58 displays the products added to the cashier dashboard following the scanning of a shopping list.

A mock-up of an employee status table. At the top is a search bar with a magnifying glass icon and the text 'Search', and a settings gear icon on the right. The table has columns for 'Profile Picture', 'Name', 'Position', 'Date Joined', 'Status', and 'Actions'. It contains three rows of employee data.

Profile Picture	Name	Position	Date Joined	Status	Actions
	Balaji Nant	Cashier	2024-03-27	Approved	
	Nithya Menon	Manager	2024-03-23	Approved	
	Meera Gonzalez	Employee	2024-03-29	Pending	

Figure 5.59: Employee Status Table

The Figure 5.59 showcases both approved and pending employees.

2 Design

This section is dedicated to the development of sequence and class diagrams for this sprint.

2.1 Class Diagram

We will present the various Class Diagrams, including the main one and those of several microservices.

2.1.1 Main Class Diagram

This is the main Class Diagram of the third sprint showcasing the different classes of different microservices.

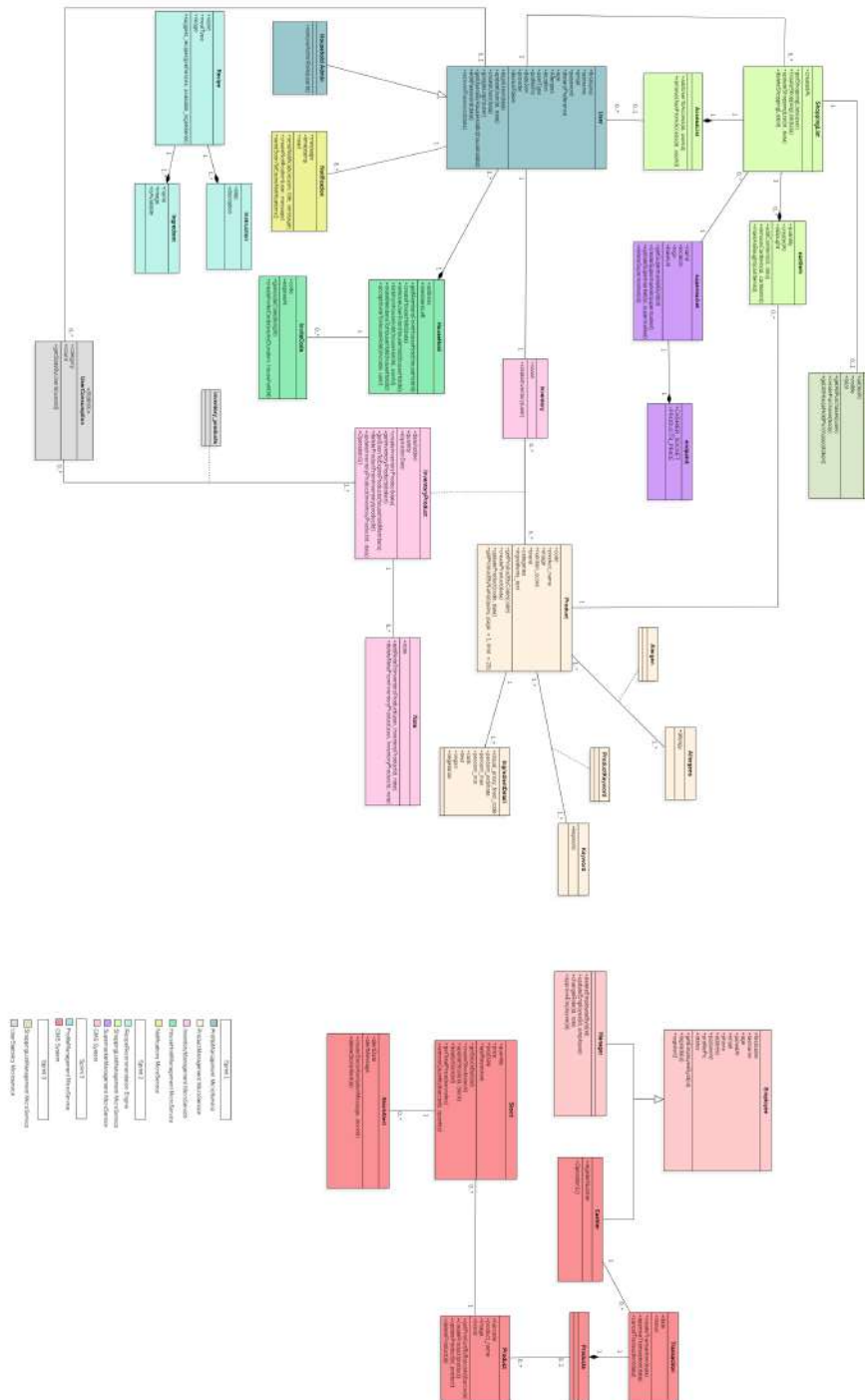


Figure 5.60: Class Diagram Sprint 3

2.1.2 GMS Class Diagram

This diagram illustrates the classes used in the separate GMS System

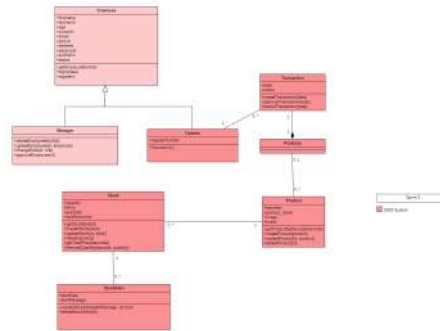


Figure 5.61: GMS Class Diagram Sprint 3

2.2 Sequence Diagram «Cashier Starting Session»

This sequence diagram illustrates the workflow initiated when the cashier scans a session QR code to start a scanning carts session.

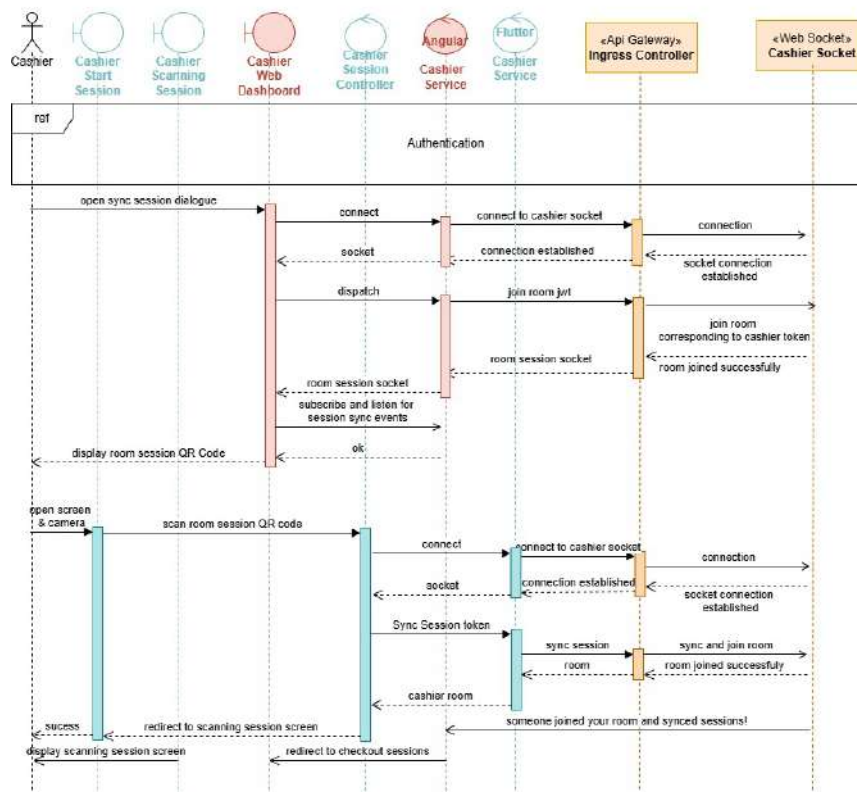


Figure 5.62: Sequence Diagram «Cashier Starting Session»

2.3 Activity Diagram

An activity can have several initial nodes, which means that several flows will start when the activity is invoked one for each initial node.[29]

The diagram shown in Figure 5.63 illustrates the interaction flow between a customer and cashier during the checkout process.

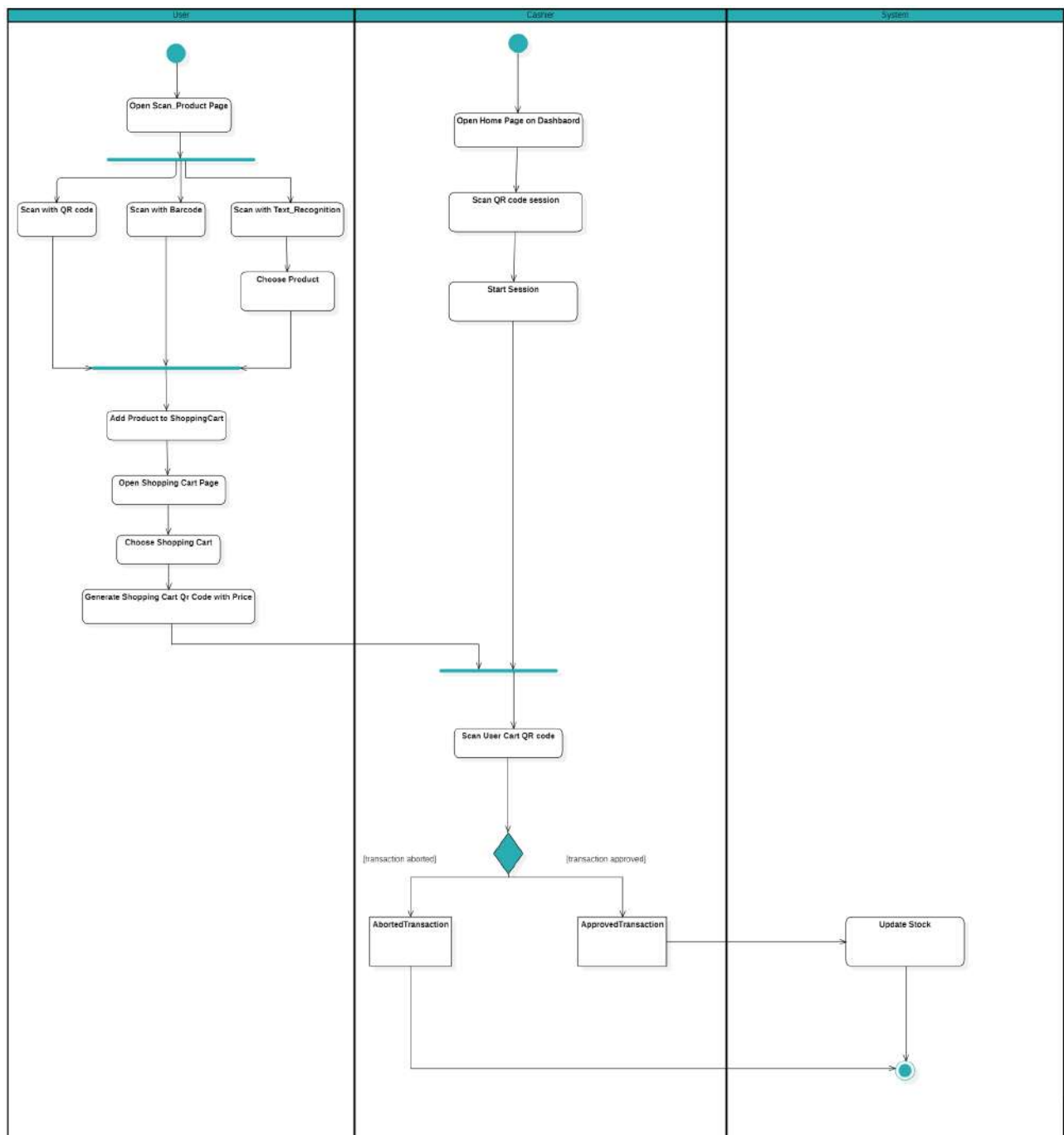


Figure 5.63: Checkout Interaction Activity Diagram

2.4 Package Diagram

This diagram illustrates the core components that make up a typical microservice within this application. While only one microservice is shown, all others share a similar structure. These components are further organized into sub-packages that can interact with each other to achieve specific functionalities.

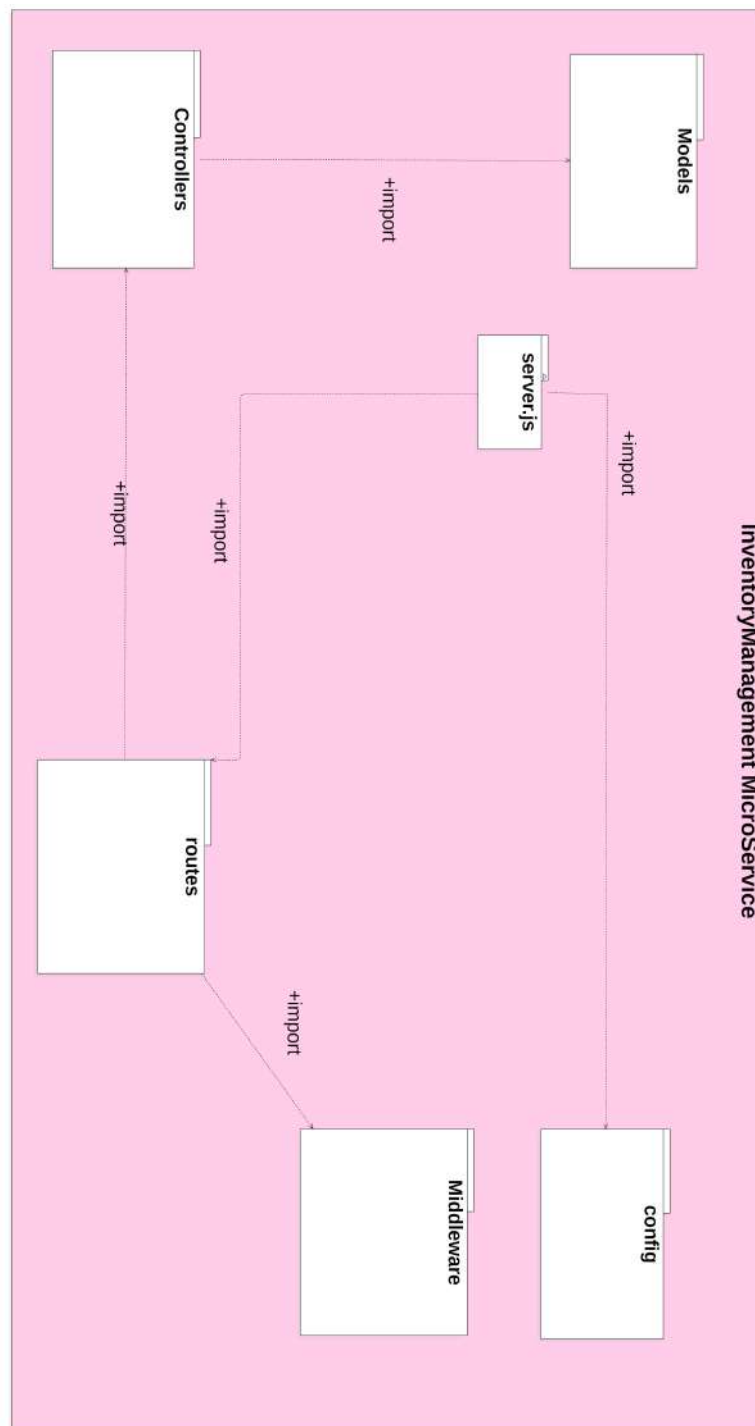


Figure 5.64: Package Diagram Sprint 3

3 Consumption Trends Chart

We implemented Business Intelligence to build this chart. It visualizes user consumption trends based on the items they add to their shopping lists per category. We first extract data from both the product management and shopping list management microservices. Then, we proceed to transform this raw data into suitable data frames which are then merged. Finally, the processed data is loaded into the database.

The chart is divided into different product categories, such as 'Snacks sucrés' and 'Produits laitiers,' each showing the frequency in which items in these categories are added to users' shopping lists (shown by the count axis). This visualization provides clear insights into user preferences and consumption patterns.

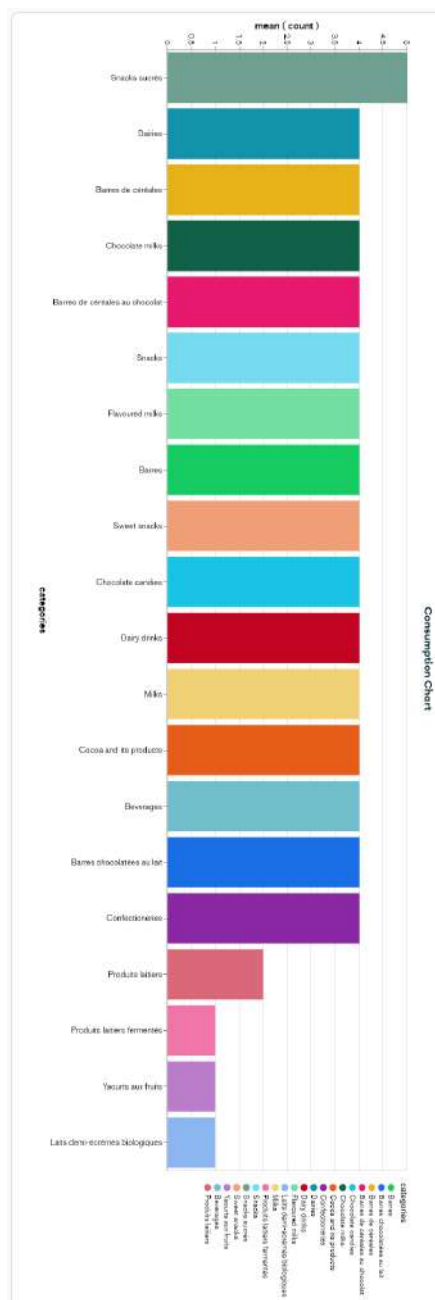


Figure 5.65: Consumption Trends Chart

4 Tests

We conducted some unit tests during this sprint, we provided them in the sprint 3 appendix, section D.1.

5 Deployment of the third increment

During the third sprint, we deployed our increment to the cloud, for more details please refer to the third sprint appendix, section D.1

6 Sprint Review

In this section, we will delve into the Sprint Review, showcasing the Product Increment delivered within the Sprint.

6.1 Main Interfaces

Below are some user interfaces developed during this sprint, including GMS interfaces and ShopFlow Admin interfaces.

6.1.1 GMS Interfaces

Here are the GMS user interfaces created in this sprint, they include the Manager's and Cashier's UIs.

GMS Manager's Interfaces





Current Employees							
#	PROFILE PICTURE	FIRST NAME	LAST NAME	DATE JOINED	POSITION	STATUS	ACTIONS
1		roblox	master	2024-03-27T02:33:51.136Z	Cashier	Approved	Modify Delete
2		Omar	Zalri	2024-03-28T12:14:07.103Z	Cashier	Approved	Modify Delete
3		test	test2	2024-04-18T13:59:44.276Z	Manager	Approved	Modify Delete
4		Yosra	Saadani	2024-04-18T16:03:54.437Z	Cashier	Pending	Modify Delete Approve

Figure 5.66: Approve an Employee

Stock List						Add Stock
PRODUCT	QUANTITY	PRICE	EXPIRATION DATE	LAST RESTOCKED	ACTIONS	
Schweineschnitzel	60	1.99	Jun 10, 2025	May 2, 2024	Delete Update Promo	

Figure 5.67: Stock List Interface

The Figure 5.66 showcases the UI providing comprehensive tools for employee management within the GMS manager's interface. Here, the manager can efficiently perform tasks such as deleting, approving, or updating employee

The Figure 5.67 shows the UI for the stock management. enabling actions like adding, deleting and updating a stock.

Stock List					
PRODUCT	QUANTITY	PRICE	EXPIRATION DATE	LAST RESTOCKED	ACTIONS
Cœur au lait & Chocolat	19	2.99 1.79	Sep 1, 2024	Apr 1, 2024	<button>Remove</button>
Viande hachée Pur bœuf	20	3.99 2.79	Oct 11, 2024	Apr 17, 2024	<button>Remove</button>
Schweineschnitzel	16	5.99 4.79	Oct 11, 2024	Apr 17, 2024	<button>Remove</button>
Safia	100	0.99 0.69	Jun 18, 2025	May 11, 2024	<button>Remove</button>

Figure 5.68: Promotional Stock List Interface

The Figure 5.68 presents the promotional stock management UI, enabling actions like adding, deleting, and updating promotional offers.

GMS Cashier's Interfaces



Figure 5.69: Start Checkout Session

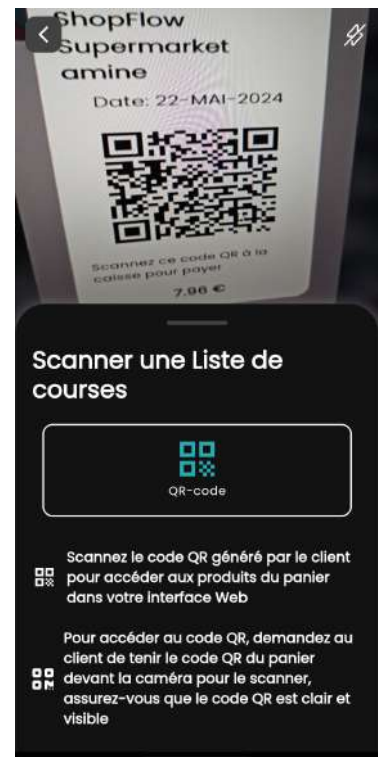


Figure 5.70: Scanning User's Shopping List QR code

The Figure 5.69 demonstrate the cashier initiating a session by scanning the room QR

code.

The Figure 5.70 depicts the cashier scanning QR codes generated by users to retrieve their products, which are then displayed on the checkout page.

The Figure 5.71 illustrates the scanned items displayed on a cashier's checkout page.



Figure 5.71: List of Products On Cashier Dashboard

6.2 ShopFlow Admin Interfaces

The Figures below represent the ShopFlow Admin Interfaces crafted during this sprint

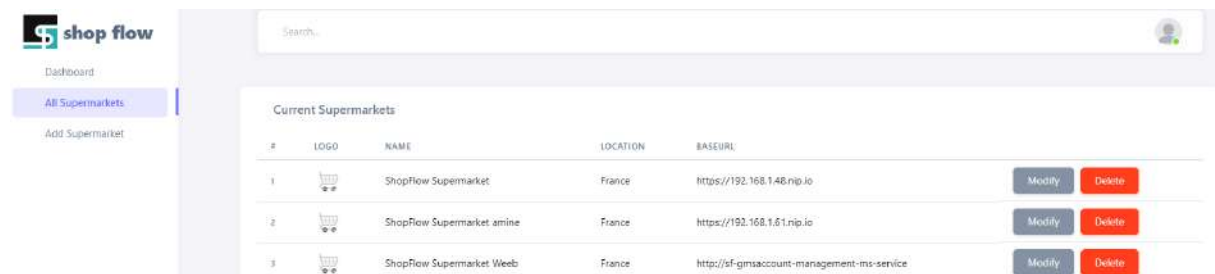


Figure 5.72: Supermarket Lists Interface

The Figure 5.72 shows the list of partner GMS for ShopFlow.

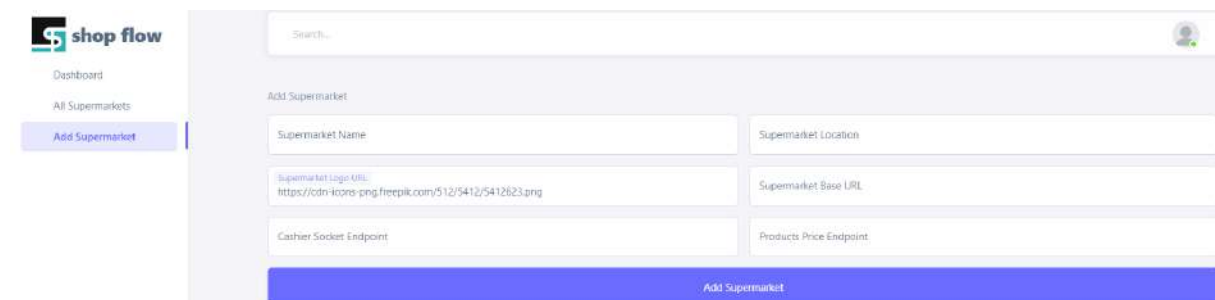


Figure 5.73: Add a Supermarket Form Interface

The Figure 5.73 shows the form for adding a partner GMS information.

6.3 User Interface

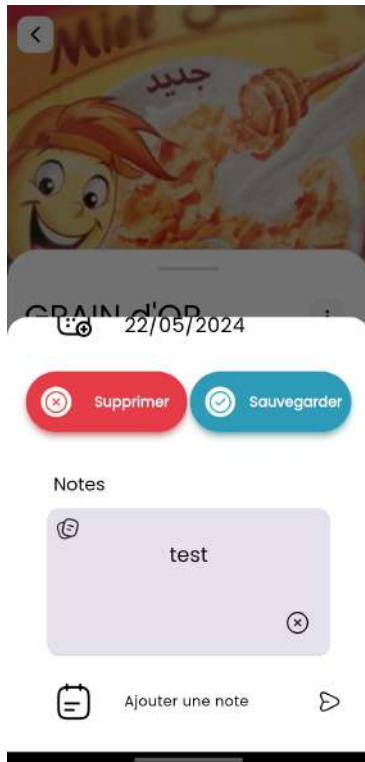


Figure 5.74: Attach Note Interface



Figure 5.75: Final Price for Shopping List

Figure 5.74 demonstrates the UI for adding notes and accessing attached notes for an inventory product.

Figure 5.75 is the UI providing access to the final price of the shopping list.

7 Sprint Retrospective

In the sprint retrospective, we'll review our achievements and challenges from the third sprint.

7.1 Retrospective Table

In this session, we will use the table below to review our key achievements, identify challenges, and discuss opportunities for improvement.

What went well	Challenges	Areas for improvement
-Interface Creation -Backend Development	- Microservice Security:Authentication vulnerabilities	-Implement More Secure Ways to authenticate requests

Table 5.22: Evaluation of Sprint 3

7.2 Sprint Burn Down Chart

The chart below shows our progress and remaining work throughout the sprint

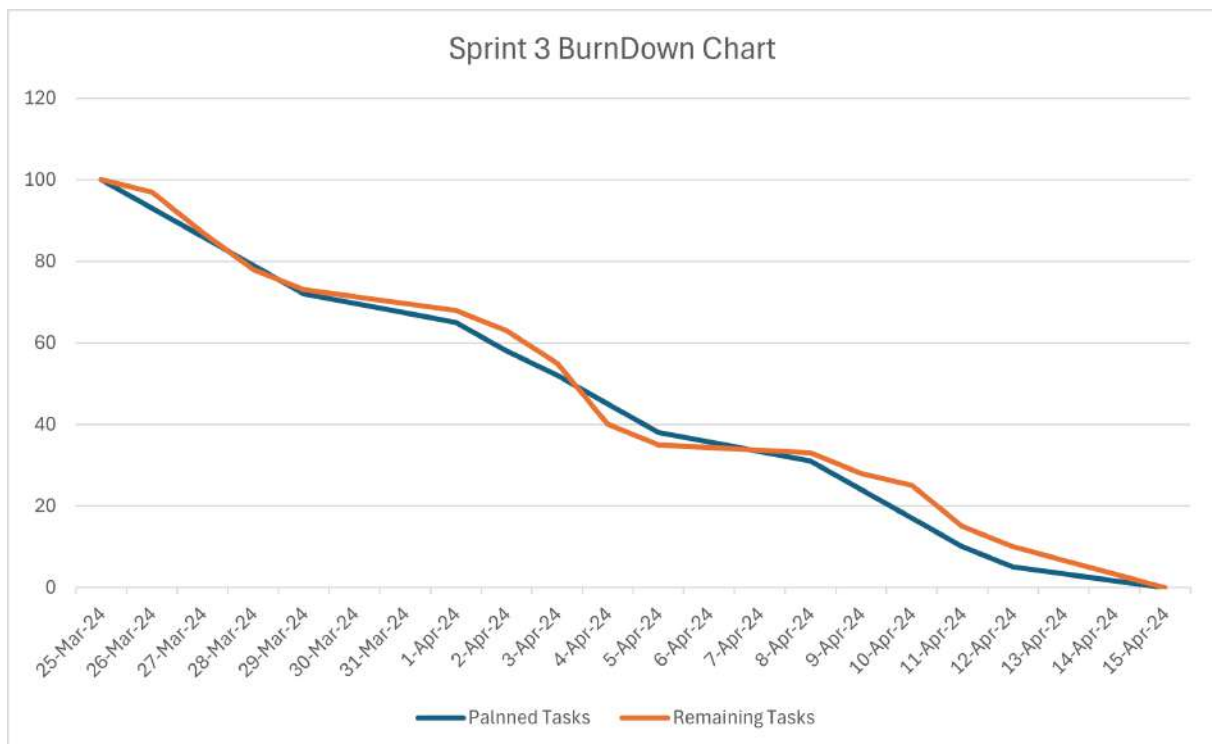


Figure 5.76: Sprint 3 Burn Down Chart

7.3 Transition to API-Gateway Level Authentication

During this sprint's development phase, we uncovered a security vulnerability: unauthorized access by third parties. Additionally, we identified redundant middleware methods across different microservices for user authentication, as well as excessive traffic on the profile management microservice.

In this upcoming sprint, we will fix these issues by integrating two levels of authentication :

- First one is introducing basic authentication functionality.
- Second one is migrating the token-based authentication to the ingress controller level.

We'll delve into these enhancements in greater detail during the upcoming sprint.

7.4 DoD Compliance Assessment

Throughout this sprint, we successfully met all the criteria outlined in our Definition of Done.

Throughout this chapter, we've detailed the user stories for Sprint 3. We then presented various mockups and established the design and different user interfaces. In the next chapter, we'll explore Sprint 4.

Chapter 6. Sprint 4 : GMS Management & Checkout Interactions

In the preceding chapter, we detailed each phase of the previous sprint's execution. Now, our attention shifts to the fourth sprint of our project. Maintaining consistency with the previous sprint, we'll delineate the crucial topics and phases for the most important features.

1 Identification of needs

During this segment, we will present the sprint 4 backlog and main actors followed by some user interface Mock-ups

1.1 Sprint Four Backlog

- **Recommendation System:** Continuing with the recommendation system feature, in the fourth sprint we will implement a product suggestion system. By analyzing the products available in users' inventory, the system suggests recipes that align with their ingredients. This personalized approach ensures that they receive relevant and practical product suggestions based on what they already have on hand and what similar users also have.
- **History and statistics :** This feature focuses on optimizing purchase tracking for individuals or households, alongside providing comprehensive statistics on food consumption patterns.
- **Statistics and analysis :** This feature delves into in-depth analysis of consumption trends, categorized by product types and user age demographics, while also incorporating sales data for a comprehensive overview.
- **Management of promotional offers :** This feature aims to optimize stock sales by integrating charts to effectively target consumer trends and assess the efficacy of marketing campaigns.

Feature ID	Sprint Feature	ID	User Story	Task			EST
S4_F5	Recommendation System	5.1	As a user, I want to receive product suggestions based on my habits	Back-end	Product Suggestions	Similar tasks from previous sprint	12h
				Frontend	Product Suggestions screen	Similar tasks from previous sprint	8h
						create ProductSuggestion.dart screen	
						Add the ProductSuggestionScreen to the routes for easier navigation	
S4_F7	History and statistics	7.1	As a user, I want to consult my purchase-history	Back-end	Purchase History	Similar tasks from previous sprint	1 1/2h
				Frontend	History Interface	Similar tasks from previous sprint	5 1/2h
						Create the PurchaseList.dart Screen	
S4_F11	Statistics and analysis	11.1	As a Manager, I want to visualize clients' consumption habits	Back-end	analyzing clients' buying consumption habits	Similar tasks from previous sprint	7h
				Frontend	visualizing clients buying consumption habits	integrate the chart in the dashboard	2h
		11.2	As a Manager, I want to visualize most trending products	Back-end	analyzing products across different groups	Similar tasks from previous sprint	6h

			across different groups of clients	Frontend	visualizing products across different groups	integrate the chart in the dashboard	2h
12	Management of promotional offers	12.1	As a Manager, I want to create promotional offers for certain products	Backend	Create Promotional Offers	Add the promotion attribute in the Stock.js model	5h
						Similar tasks from previous sprint	
				Frontend	Promotional Offers Interface	Similar tasks from previous sprint	4h
		12.3	As a Manager, I want to target and create promotional offers based on consumer segments.	Backend	Consult client segments and create promotional offers	Similar tasks from previous sprint	7h
				Frontend	Add from the chart	create the Promotion.ts Service	4h
						create the PromotionTable component	
						add the PromotionTable to the routes	
		12.5	As a user, I want to see products on promotion	Backend	Products on promotion	add the functionality of receiving promotional products	3h
				Frontend	Products on promotion interface	Similar tasks from previous sprint	5h
						create the PromoProductCard.dart component	
						integrate the PromoProductCard.dart in the explore page	

Global Tasks	Setup the product recommendation system Python project	1h
	Setup the user statistics Python project	1h
	Build and push a Docker image for each feature	1h
	Create Tests scripts	1h
	create the kubernetes configuration YAML files(deployments, services, ingress controller. . .)	1h
	create the sequence diagrams	2h
	set up a simple ci/cd pipeline to automate testing, building, and deployment(locally while developing)	1h
	set up the Kubernetes Cluster in the cloud	5h
	set up the ci/cd pipeline to deploy to our cloud Kubernetes cluster	2h
Total Estimation		~135 hours
For the full backlog, please refer to the sprint 4 appendix, section D.1 (only a snippet is shown here for report length reasons).		

Table 6.23: Sprint 4 backlog snippet

1.2 Identification of actors

The main actors of the fourth sprint are :

Actor	Description
User	This is the main user of the mobile application
GMS Manager	This is the main user of the supermarket dashboard application

Table 6.24: Identification of actors Sprint 4

1.3 User Interface Mock-ups

These are some of the user interface mock-ups for the fourth sprint The Figure 6.77 illustrates the dashboard UI design tailored for the GMS manager, enabling access to comprehensive statistics provided by the server. Figure 6.78 presents the UI design illustrating confirmed purchases for users.

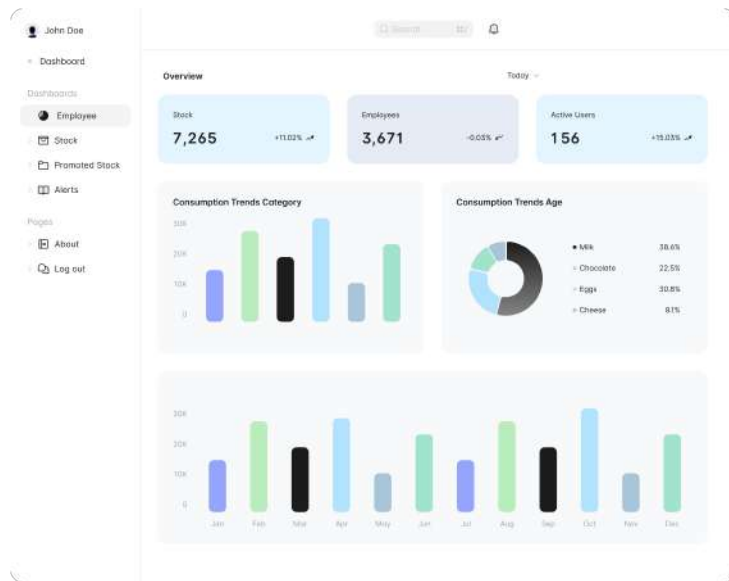


Figure 6.77: GMS Manager dashboard mock-up

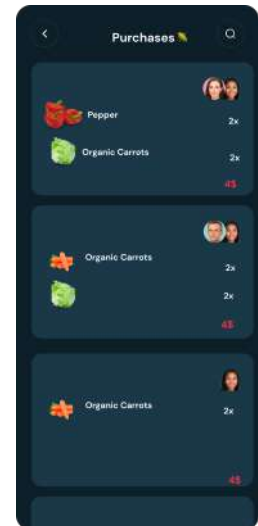


Figure 6.78: User's purchases mock-up

The Figure 6.77 illustrates the dashboard of a GMS that is accessible by a manager. Once here, they can check different statistics about users and current stock.

The Figure 6.78 showcases the user order history. It details the total amount spent and the user who made the purchase.

2 API-Gateway Level Authentication

As mentioned in the retrospective of the previous sprint, we will implement two authentication mechanisms at the Ingress Controller level.

2.1 Basic Authentication

Basic Authentication is a method for an HTTP user agent (e.g., a web browser) to provide a username and password when making a request.

When employing Basic Authentication, users include an encoded string in the Authorization header of each request they make. The string is used by the request's recipient to verify users' identity and rights to access a resource.

2.2 Token based Authentication

Instead of duplicating authentication functionality across multiple microservices, and to alleviate the burden on the profile management service, we decided to implement the functionality within the ingress controller. Now, when a user accesses our system, authentication occurs at the ingress controller level and his information gets passed through the headers of the request which eliminates the need for separate authentication requests.

The workflow is better illustrated in the figure below.

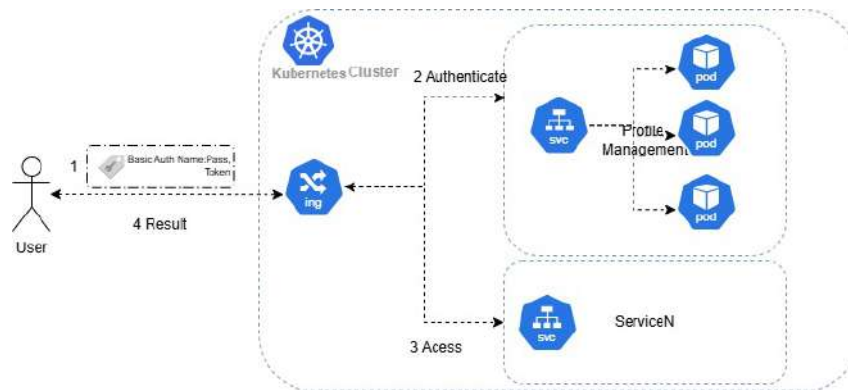


Figure 6.79: Ingress Authentication

3 Product Suggestion

As mentioned in the second sprint, in this section, we will discuss the model of the recommender system we used during this sprint.

3.1 Neighborhood-based Collaborative Filtering Recommender Systems

Neighborhood-based collaborative filtering algorithms[28], also referred to as memory-based algorithms, were among the earliest algorithms developed for collaborative filtering. These algorithms are based on the fact that similar users display similar patterns of rating behavior and similar items receive similar ratings. There are two primary types of neighborhood-based algorithms:

1. **User-based collaborative filtering:** In this case, the ratings provided by similar users to a target user A are used to make recommendations for A. The predicted ratings of A are computed as the weighted average values of these “peer group” ratings for each item
2. **Item-based collaborative filtering:** In order to make recommendations for target item B, the first step is to determine a set S of items, which are most similar to item B. Then, in order to predict the rating of any particular user A for item B, the ratings in set S, which are specified by A, are determined. The weighted average of these ratings is used to compute the predicted rating of user A for item B.

In our case, we went with **User-based collaborative filtering**.

There are two types of User-based collaborative filtering :

1. **Implicit :** implicit feedback scenarios may correspond to the use of unary rating matrices in which users express their interests by buying items.

2. **Explicit** : Explicit feedback may correspond to ratings.

In our case, we went with The **Implicit** approach. Here's a figure explaining the process :

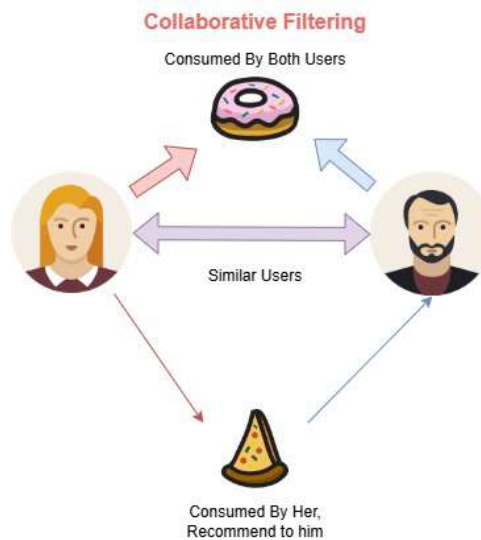


Figure 6.80: User Based Collaborative Filtering

Ap- proach	Conceptual Goal	Input	Our Case
User- based col- lab- ora- tive filter- ing	Give me recommen- dations based on a collaborative approach that leverages the ratings and actions of my peers/myself.	User ratings + com- munity ratings	Provide product recommendations by analyz- ing the user's shopping list, current inventory, and past purchases.

Table 6.25: conceptual goals of User-based collaborative filtering methods

4 Design

This section is dedicated to the development of class and sequence diagrams for this sprint.

4.1 Class Diagram

This is the Class Diagram of the fourth sprint.

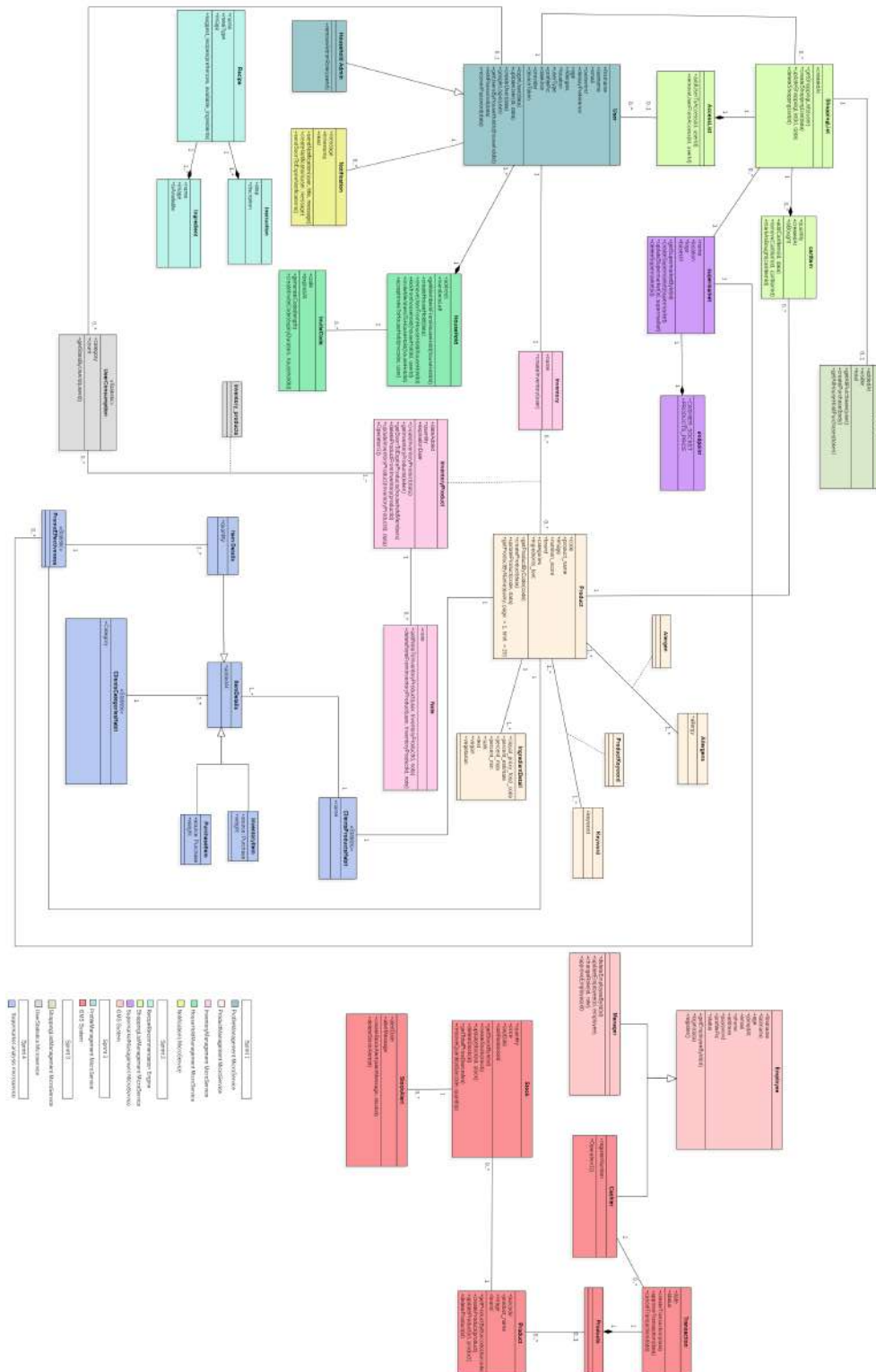


Figure 6.81: Class diagram Sprint 4

Class	Business Rule
UserConsumption	- Containers inventory products references to avoid over-counting
ItemDetail	- Could be either a PurchaseItem or an InventoryItem

Table 6.26: Business rules of sprint 4 class diagram

4.2 Sequence Diagrams

In this section, we will explore various interactions of the application by employing different types of sequence diagrams

4.2.1 Design Sequence diagram «Get products recommendations»

This design sequence diagram describes the process of getting product recommendations, illustrating communication between different components including the ML models.

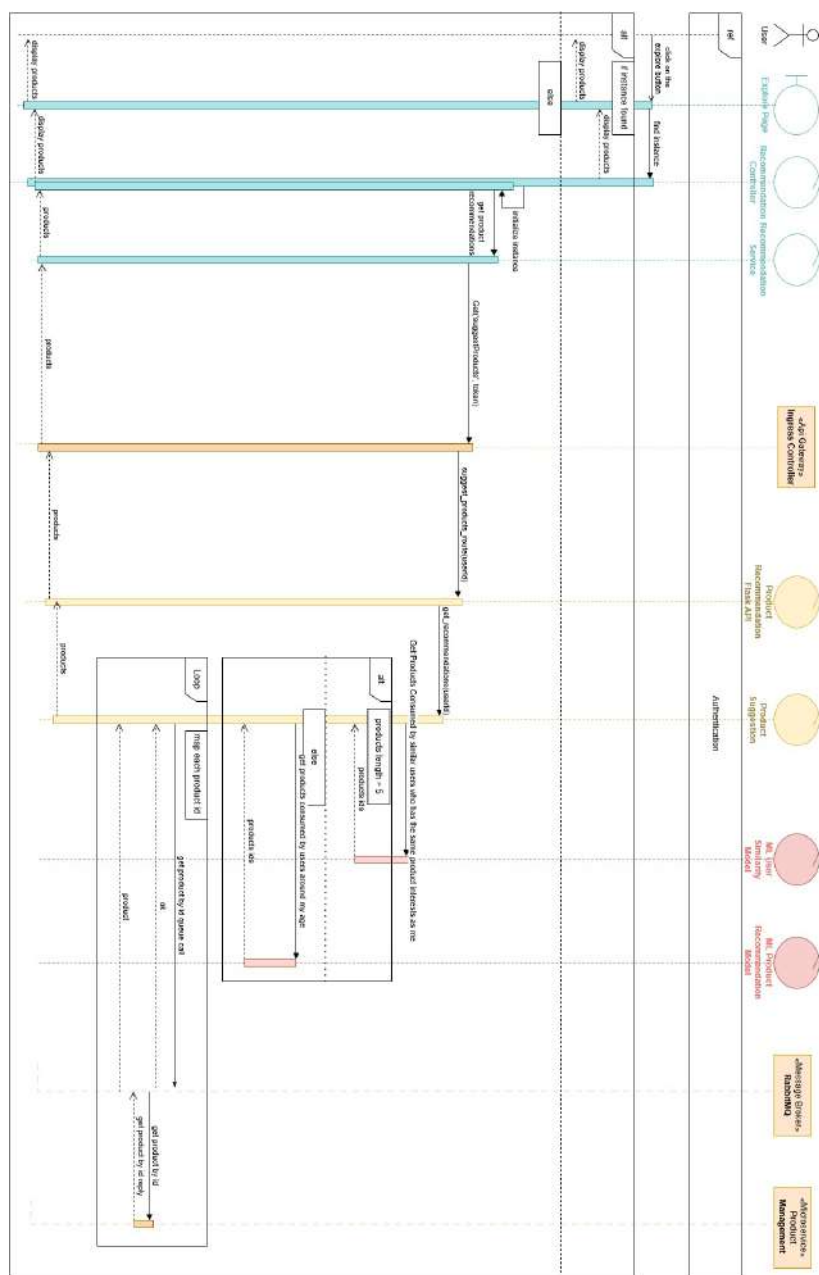


Figure 6.82: Design Sequence diagram «Get products recommendations»

4.2.2 Design Sequence diagram «Add a product to Inventory»

This design sequence diagram describes the process of adding a product to the inventory

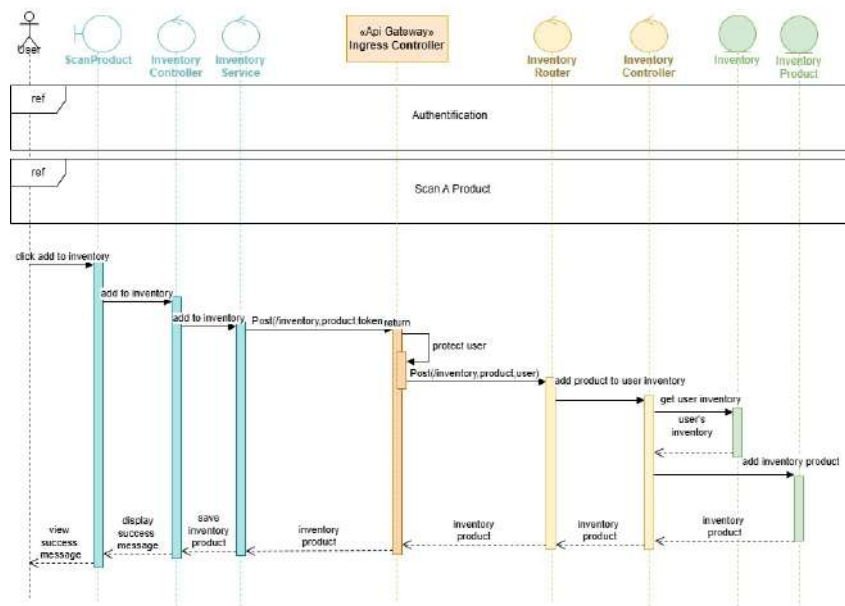


Figure 6.83: Design Sequence diagram «Add a product to Inventory»

4.2.3 Object Sequence Diagram «Notifying Services of Inventory Product Addition»

This diagram shows how the message broker informs other services when a new inventory product is added.

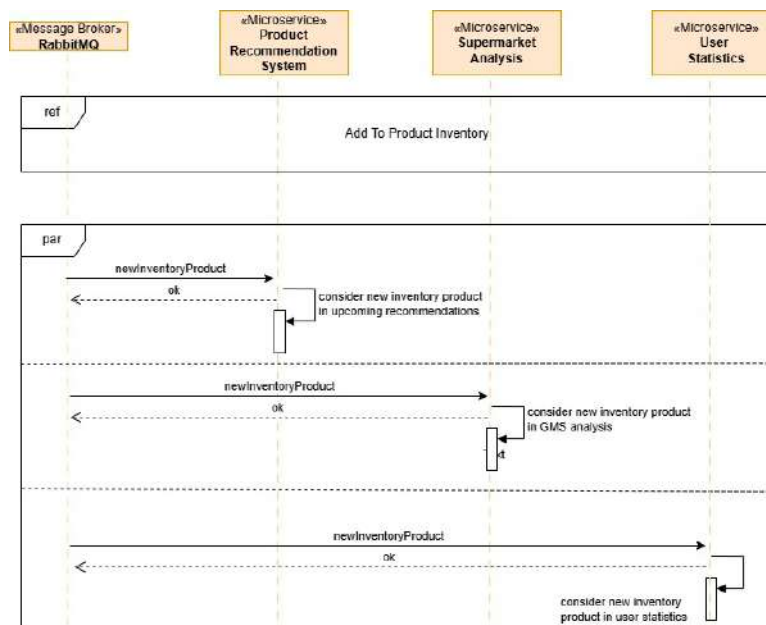


Figure 6.84: Object Sequence Diagram «Notifying of Inventory Product Addition»

6 Deployment of the fourth increment

6.1 System Monitoring

The diagram illustrates the architecture of the Data Collection Rules Association. It shows a cluster of nodes (Node 1, Node 2, ..., Node N) sending metrics to a Data Collection endpoint. The endpoint transforms metrics and sends them to Managed Prometheus, which then queries the metrics. The diagram also shows the association of Data Collection Rules with the Managed Prometheus instance.

The screenshot displays the Azure Managed Prometheus dashboard for a Kubernetes cluster. The top navigation bar includes a search bar, a 'Share' button, and a 'Last 1 hour' time range selector. The left sidebar shows a navigation menu with options like Home, Starred, Dashboards, Playlists, Snapshots, Library panels, Reporting, Explore, Alerting, Connections, and Administration. The main content area features a 'Data Source' dropdown set to 'Managed_Prometheus_defaultazuremonitorworkspace-eus2', a 'cluster' dropdown, and a 'shopflow' dropdown. Below these are six summary cards for 'Headlines': CPU Utilisation (10.3%), CPU Requests Commitr (31.4%), CPU Limits Commitmen (467%), Memory Utilisation (32.7%), Memory Requests Com (16.0%), and Memory Limits Commit (300%). A 'CPU' section contains a 'CPU Usage' chart showing a line graph of CPU usage over time, with a legend for 'default', 'gitlab-agent-azure-agent', 'ingress-nginx', and 'kube-system'. The chart shows a peak in usage around 11:10. Below the chart is a 'CPU Quota' section with a 'CPU Quota' chart.

Design, Implementation and Deployment of «ShopFlow»

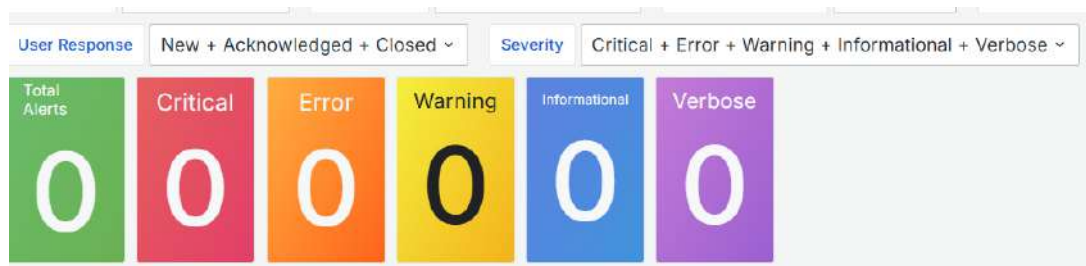


Figure 6.87: an example of our Grafana Alerts

7 Sprint Review

In this section, we will delve into the Sprint Review, showcasing the Product Increment delivered within the Sprint

7.1 Main Interfaces

In this section, we'll review the primary user interfaces for our key users in the current sprint.

7.1.1 User Interfaces

Below are some user interfaces developed during this sprint.

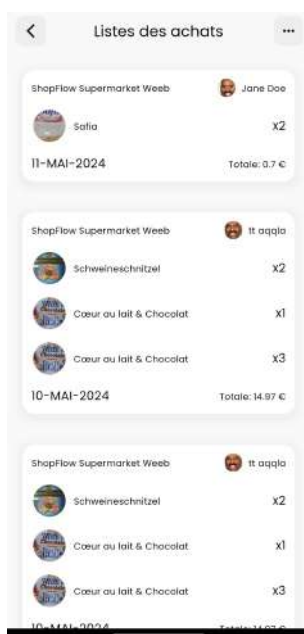


Figure 6.88: User Purchases Interface



Figure 6.89: Household consumption statistics

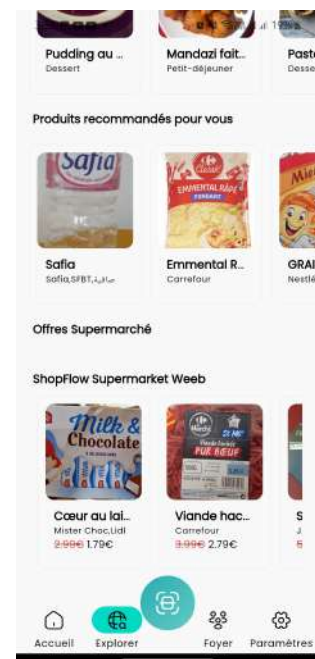


Figure 6.90: Products recommendations interface

The Figure 6.88 demonstrates the UI for the confirmed purchases of the user, here the user can find purchases made for the shopping carts they created or have access to.

The Figure 6.89 is the UI providing statistics of the household members' consumption categories.

The Figure 6.90 showcases the products recommended for the user, the products appear on the explore page.

7.1.2 GMS Manager Interfaces

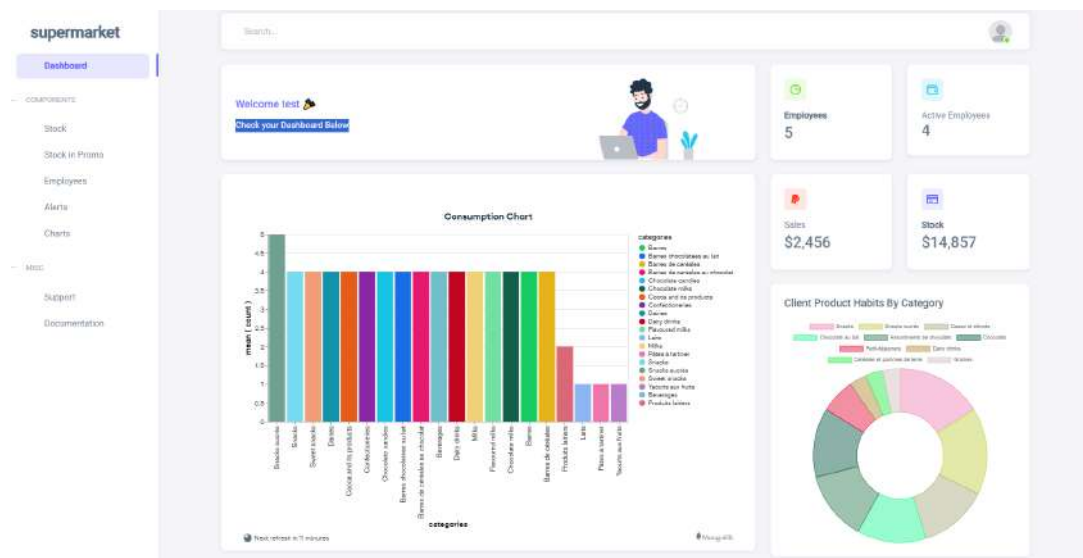


Figure 6.91: GMS Dashboard Interface first section

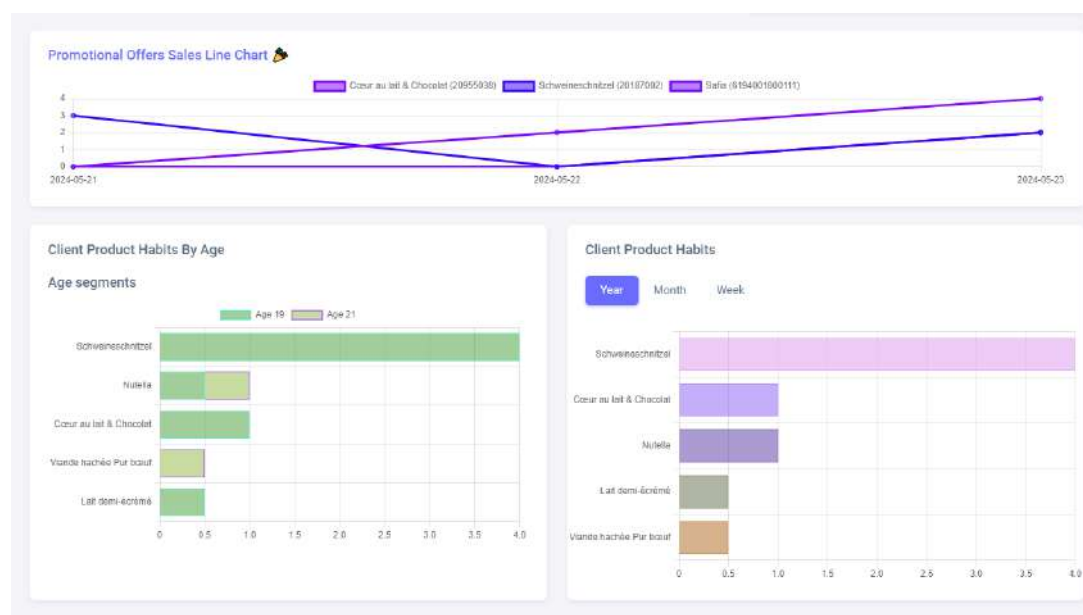


Figure 6.92: GMS Dashboard Interface second section

The Figure 6.91 demonstrates the UI for the first section of the GMS manager dashboard. The section features various statistics and charts: a bar chart displaying the most frequently purchased product categories by users, and a donut chart showcasing the most consumed categories (including products in inventories) across all users over the past year.

The Figure 6.92 presents the UI for the second section, which includes three charts: a line chart illustrating the effectiveness of promotional offers during the past week for a specific GMS, a stacked bar chart displaying the most consumed products across different age segments, and a horizontal bar chart showing the most consumed products within a specific date range.

8 Sprint Retrospective

In the sprint retrospective, we'll review our achievements and challenges from the third sprint.

8.1 Retrospective Table

In this session, we will use the table below to review our key achievements, identify challenges, and discuss opportunities for improvement.

What went well	Challenges	Areas for improvement
-Recommendation System Development -Time Management	-	-

Table 6.27: Evaluation of Sprint 4

8.2 DoD Compliance Assessment

Throughout this sprint, we successfully met all the criteria outlined in our Definition of Done.

Throughout this chapter, we've detailed the user stories for Sprint 4. We then presented various mockups and established the design and different user interfaces. We will wrap up this report with a general conclusion in the next section.

General Conclusion

«ShopFlow» has been an immensely informative and unique journey, providing us with the opportunity to explore and acquire skills across diverse areas of information technology, including DevOps, business intelligence, and machine learning.

The primary objective of the «ShopFlow» platform is to provide an optimized platform for managing food and groceries, addressing issues of food waste, and efficient inventory management. Also providing useful insights for partner GMS. Throughout the project, we conducted a thorough analysis of the current landscape, identifying significant challenges in food waste management in France and the inefficiencies in existing solutions. Our proposed solution, «ShopFlow», aimed to mitigate these problems by leveraging advanced data management and user-friendly interfaces.

The project involved several key phases, including the study of the current market, the development of functional and non-functional requirements, and the implementation of SCRUM. We designed and deployed a series of features across multiple sprints, including authentication, inventory management, household collaboration, shopping list management, data visualization, and a recommendation system.

Despite the project's successes, we identified several areas for improvement. These include deploying the mobile application in official stores, integrating more advanced machine learning algorithms for more personalized recommendations, and expanding the system's features. Future work could also explore additional user interface enhancements and broader integration with external systems for a more comprehensive food management ecosystem.

In conclusion, «ShopFlow» has demonstrated significant potential in optimizing food and grocery management. By addressing the initial problems and providing innovative solutions, we have laid a solid foundation for future developments and improvements. The application not only meets its primary objectives but also opens up new perspectives for enhancing food management practices and reducing waste on a larger scale.

Bibliography and Netnography

- [1] **ADEME**. Etude « Pertes et gaspillages alimentaires : état des lieux et leur gestion par étapes de la chaîne alimentaire » [Online; Consulted on 15-January- 2024].
- [2] **OpinionWay-Smartway** smartway.ai/fr/blog/2023/06/22/etude-francais-18-24-ans-champion-gaspillage-alimentaire-2023 [Online; Consulted on; consulté le 17-January- 2024]
- [3] **Z-CONSULTING** [Z-Consulting Official Website](#) [Online; Consulted on 22-January- 2024].
- [4] **Amazon** [The next evolution of the Dash Cart: New features and expansion to first Whole Foods Market Store](#) [Online; Consulted on 22-January- 2024].
- [5] **BEEP** [BEEP - Expiry Date Tracking](#) [Online; Consulted on 22-January- 2024].
- [6] **Steve McConnell**, **More Effective Agile: A Roadmap for Software Leaders** [Online; Consulted on 3-Feb- 2024]
- [7] **Scrum** [What is a Definition of Done?](#) [Online; Consulted on 3-Feb- 2024].
- [8] **Amazon** [Difference Between Monolithic and Microservices Architecture](#) [Online; Consulted on 5-Feb- 2024].
- [9] **Chris Richardson**, **Microservices Patterns WITH EXAMPLES IN JAVA** [Book; Consulted on 6-Feb- 2024]
- [10] **Amazon** [Enabling data persistence in microservices : Database-per-service pattern](#) [Online; Consulted on 8-Feb- 2024].
- [11] **GetX** [Get Flutter Package](#) [Online; Consulted on 10-March- 2024]
- [12] **Docker** [Use containers to Build, Share and Run your applications](#) [Online; Consulted on 2-March- 2024]
- [13] **FAUN—Developer Community** [What is a Container?](#) [Online; Consulted on 2-March- 2024].
- [14] **Azure days** [Container Orchestration with Docker Swarm, Marathon, Kubernetes](#) [Online; Consulted on 2-March- 2024]
- [15] **Viktor Farcic**, **A Practical Guide to Kubernetes** [Online; Consulted on 3-March- 2024]

- [16] **Kubernetes** [An overview of Kubernetes](#) [Online; Consulted on 3-March- 2024]
- [17] **Nigel Poulton**, **The Kubernetes Book** [Book; Consulted on 7-March- 2024]
- [18] **Sricharan Vadapalli**, **DevOps: Continuous Delivery, Integration, and Deployment with DevOps** [Book; Consulted on 5-March- 2024]
- [19] **Christopher Cowell**, **Automating DevOps with GitLab CI/CD Pipelines** [Book; Consulted on 3-March- 2024]
- [20] **TechZone Automation** [Continuous Delivery](#) [Online; Consulted on 7-March- 2024]
- [21] **Andriy Burkov**, **Machine Learning Engineering** [Book; Consulted on 10-April- 2024]
- [22] **Swain Scheps**, **Business Intelligence FOR DUMmIES** [Book; Consulted on 20-April- 2024]
- [23] **Microsoft Azure** [What Is Cloud Computing?](#) [Online; Consulted on 1-May- 2024]
- [24] **Toptal** [K8s/Kubernetes: AWS vs. GCP vs. Azure](#) [Online; Consulted on 2-May- 2024]
- [25] **Pluralsight** [AKS vs EKS vs GKE: Managed Kubernetes services compared \(pluralsight.com\)](#) [Online; Consulted on 2-May- 2024]
- [26] **MongoDB** [Model Many-to-Many Relationships with Embedded Documents](#) [Online; Consulted on 20-Feb- 2024]
- [27] **RabbitMQ** [RabbitMQ: One broker to queue them all](#) [Online; Consulted on 2-March- 2024]
- [28] **Charu C. Aggarwal**, **Recommender Systems The Textbook** [Book; Consulted on 15-April- 2024]
- [29] **IBM** [Activity Diagrams Control nodes](#) [Online; Consulted on 9-April- 2024]

Appendices

Appendix A

Chapter 2 Additional Content

Ingress Controller Example

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/use-regex: "true"
    nginx.ingress.kubernetes.io/websocket-services: "sf-gmsaccount-management-ms-service"
    service.beta.kubernetes.io/aws-load-balancer-proxy-protocol: ""
  name: my-ingress
spec:
  ingressClassName: "nginx"
  rules:
  - http:
      paths:
      - path: /api-pm
        pathType: Prefix
        backend:
          service:
            name: sf-profile-managment-ms-service
            port:
              number: 80
```

Figure A.1: A Pod template wrapped in a Deployment [17]

Appendix B

Sprint 1 Additional Content

Sprint One Backlog

Feature ID	Sprint Feature	ID	User Story	Task			EST
S1_F1	S1 Profile management	1.1	As a user, I want to create an account using different signup methods	Backend	Email-Sign up	Create the User.js model	1h
						Create the user-Controller.js	
						Create the user-Router.js	
						Create the MongoDB database	
						Create the necessary files to link the backend with the database	
						Create the necessary middlewares	
					Google Sign up	Create the helpers for google authentication	5h
						Create and configure a google developer account	

						add the interactiveForm screen to the routes for easier navigation	
		1.2		Backend	Modify user information	Create the functionality in the userController.js add the function in the userRouter.js	1h
			As a user, I want to modify my personal information for better personalization	Frontend	Profile interface	Create the settings.dart screen add the Profile interface in the routes.dart for easier navigation	4h
					Edit Profile	Create the modify_profile.dart screen Create the modifyProfile functionality in the userService.dart add the created modifyProfile functionality in the userController.dart	4h
		1.4	As a user, I want to login using different login methods	Backend	Email Login	Create the functionality in the userController.js add the functionality in the userRouter.js	1h
					Google Login	Create the helpers for google authentication Create the login functionality in the userController.js	

				Frontend	Lo- gin Interface	create the login- Controller.dart	2h
						create the login functionality in the userSer- vice.dart	
						create the login- Screen.dart	
						add the created login functional- ity in the user- Controller.dart	
						add the login- Screen to the routes for easier navigation	
S1_F2	S1 Acquisition of product data	2.1	As a user, I want to scan products using Bar-codes	Backend	Get Prod- uct by code	create the Prod- uct.js model	4h
						create the Pro- ductController.js	
						create the ProductRouter.js	
						create the mon- godb Database	
						Create the neces- sary files to link the backend with the database	
						Create the functionality in the productCon- troller.js and add it to the Produc- tRouter.js	
				Frontend	Scan Prod- uct Interface	Create the Prod- uct.dart model	7h
						Create the Prod- uctService.dart	
						Create the ProductCon- troller.dart	
						Create the Scan- Product.dart screen	

						Create the productDetailsScreen.dart	
						add the ScanProduct.dart screen and the productDetailsScreen.dart to the routes for easier navigation	
		2.2	As a user, I want to scan products using a QR Code	Back-end	Get Product by QR code	Create the functionality in the productController.js and add it to the ProductRouter.js	1h
				Frontend	Scan Product Interface	Create the functionality in the productController.js	1h
						implement the functionality into the ScanProduct.dart screen	
		2.3	As a user, I want to acquire products using character recognition	Back-end	Get Product by Name	Create the functionality in the productController.js and add it to the ProductRouter.js	3h
				Frontend	Scan Product interface	Create the functionality in the productController.js	5h
						Implement the functionality into the ScanProduct.dart screen	
					Text Recognition Result Interface	Create the TextRecognitionResults.dart screen	1h

		2.4	As a user, I want to manually search a product in case the other methods malfunction	Back-end	Search A product manually	use the Get Product by Name functionality	½h
				Front-end	Search a product manually	integrate a search bar and products results in the Explore-Screen.dart	3h
S1_F3	S1 Inventory management	3.1	As a user, I want to visualize my current inventory	Backend	Get Inventory	Create the Inventory.js model	7h
						Create the InventoryProduct.js model	
						Create the InventoryController.js	
						Create the InventoryRouter.js	
						Create the mongoDB database	
						Create the necessary files to link the backend with the database	
						Create the functionality in the inventoryController.js and add it to the inventoryRouter.js	
						Create the necessary middlewares	
				Frontend	Inventory Interface	Create the Inventory.dart model	5h
						Create the InventoryProduct.dart model	
						Create the inventoryService.js	

						Create the inventoryController.js	
						Create the inventoryDetails.dart screen	
						add the inventoryDetails.dart screen to the routes for easier navigation	
		3.2	As a user, I want to receive a notification when a product in my inventory is soon to expire	Backend	Notification Management	Create the Notification.js model	5h
						Create the notificationController.js	
						Create the notificationRouter.js	
						Create the mongoDB database	
						Create the necessary files to link the backend with the database and firebase	
						Create the firebase Configuration	
						Create the functionality in the notificationController.js and add it to the notificationRouter.js	
				Frontend	Notification Management	Create the firebase Configuration	3h
		3.4	As a user, I want to add products to my inventory	Backend	Add Product to inventory	Create the functionality in the inventoryController.js and add it to the inventoryRouter.js	2h

				Frontend	Add Product to Inventory Form	Create the functionality in the InventoryService.dart and add it in the inventoryController.dart	3h
						Add a button that opens a form in the ProductDetails.dart	
						Create the inventoryProduct.dart screen	
		3.5	As a user, I want to remove products from my inventory	Back-end	Remove Product From Inventory	Create the functionality in the inventoryController.js and add it to the inventoryRouter.js	1h
				Frontend	Remove Product From Inventory Button	Create the functionality in the InventoryService.dart and add it in the inventoryController.dart	1h
						Add a button that deletes an InventoryProduct ProductDetails.dart	
		3.6		Back-end	Modify Inventory Product	Create the functionality in the inventoryController.js and add it to the inventoryRouter.js	1h

			As a user, I want to modify product information in case I made a mistake while adding a product manually	Frontend	Modify Inventory Product Form	Create the functionality in the InventoryService.dart and add it in the inventoryController.dart Add a button that modifies an InventoryProduct in InventoryProductDetails.dart	2h
S1_F4	S1 Household management	4.1	As a user, I want to invite other users to my household to share inventories	Backend	Invite Members with Invite Code	Create HouseHold.js model	4h
						Create InviteCode.js model	
						Create the householdController.js	
						Create the householdRouter.js	
						Create the mongoDB database	
						Create the necessary files to link the backend with the database	
						Create the functionality in the householdController.js and add it to the householdRouter.js	
				Frontend	Invite Member Interface	Create the HouseHold.dart model Create the InviteCode.dart model Create the householdService.dart	5h

						Create the houseHoldController.dart	
						Create the functionality in the houseHoldService.dart and add it in the houseHoldController.dart	
						Create the inviteCodeScreen.dart	
						Create the acceptInvite.dart screen	
		4.2	As a user, I want to track the products of my household members	Backend	Get Inventory products of every household member	Create the functionality in the houseHoldController.js and add it to the houseHoldRouter.js	3h
						Create the necessary middlewares	
				Frontend	Shared Inventory Interface	Create the functionality in the houseHoldService.dart and add it in the houseHoldController.dart to get a grouped-Inventories of a household	3h
						Create the InventoryUserDetails.dart screen	
						Create the household-Screen.dart that shows members of household	

		4.3	As a user, I want to collaboratively add products with other members	Back-end	Add Product to inventory	Create the functionality in the productController.js and add it to the ProductRouter.js	2h
				Frontend	Members who added inventory product interface	Create the functionality in the houseHoldService.dart and add it in the houseHoldController.dart	2h
						Add a segment in the InventoryProductDetails.dart screen to show who added the specific product	
		4.4	As a user, I want to collaboratively remove products with other members	Back-end	Remove Product from inventory	Create the functionality in the productController.js and add it to the ProductRouter.js	1h
				Frontend	Inventory Interface	Create the functionality in the houseHoldService.dart and add it in the houseHoldController.dart	1h
						Add a button in the InventoryProductDetails.dart screen to delete a specific product	
Global Tasks				Setup the profile management microservice		2h	
				Setup the product management microservice		2h	

	Setup the household management microservice	2h
	Setup the inventory management microservice	2h
	Setup the flutter frontend project by initializing the necessary modules such as the design of the application (the common widgets, the main theme and the constants such as the colors, the text theme etc), the home page screen and some utility files.	5h
	Create the Use Case Diagram	1h
	Create the Class Diagram	1h
	Create three sequence diagrams	2h
	Create Test scripts	2h
	Build and push a Docker image for each feature	$\frac{1}{2}$ h
	set up a local kubernetes development environment	2h
	create the kubernetes configuration YAML files(deployments, services, ingress controller. . .)	3h
	set up a simple ci/cd pipeline to automate testing, building and deployment	3h
	Total Estimation	~130 hours

Table B.1: First sprint full backlog

User Interface Mock-ups

These are some of the user interface mock-ups for our application:

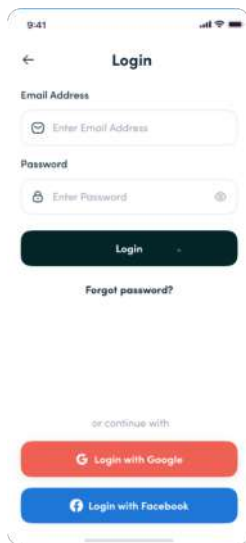


Figure B.1: Login interface mock-up

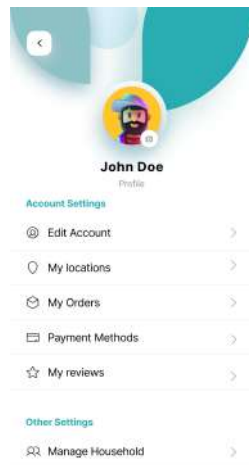


Figure B.2: Settings interface mock-up

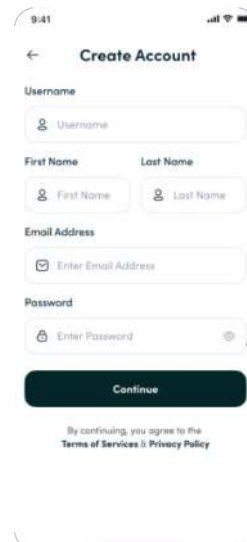


Figure B.3: Create account interface mock-up



Figure B.4: Select interests interface mock-up

The Figure B.1 illustrates the login interface.

The Figure B.2 illustrates the settings interface.

The Figure B.3 illustrates the sign up interface.

The Figure B.4 illustrates the dietary preferences interface.

Tests

This is the unit test for the Get Inventory by Id functionality. We tested it by creating a mockInventory variable to ensure the coherence and accuracy of the data received.

```
> inventorymanagement@1.0.0 test
> mocha 'tests/**/*.test.js'

DB Connected Successfully
(node:31448) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:31448) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

Get Inventory By Id
✓ Get Inventory By Id should return an inventory (698ms)

1 passing (781ms)
```

Figure B.5: Get Inventory By ID unit Test

This is the unit test for the Get Product By Name functionality. We tested it by creating a mockProduct variable to ensure the coherence and accuracy of the data received.

```
> mocha 'tests/**/*.test.js'

Channel Created!
DB Connected Successfully
(node:34772) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use 'node --trace-warnings ...' to show where the warning was created)
(node:34772) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

ProductController
  ✓ getProductByName should return products (783ms)

1 passing (786ms)
```

Figure B.6: Get Product By Name unit Test

Setting up the CI/CD pipeline

Here's a zoomed-in view of the status of the Build, Test, and Deploy stages in the “House-Hold Management” and “Notifications” Services pipeline:

Changes 1		Pipelines 1	
Status	Pipeline	Created by	Stages
Passed 00:02:43 3 days ago	kick user from household #1204346737 dev → Sba0cabe		✓ ✓ ✓

Figure B.7: Complete CI/CD pipeline succeeded following a commit in the dev branch of the Household Management service

Changes 12		Pipelines 1	
Status	Pipeline	Created by	Stages
Failed 00:01:33 1 week ago	notifications work + schedule every two days #1198953225 dev → aad33811		✓ ✓ ✗

Figure B.8: Complete CI/CD pipeline failed following a commit in the dev branch of the Notifications service

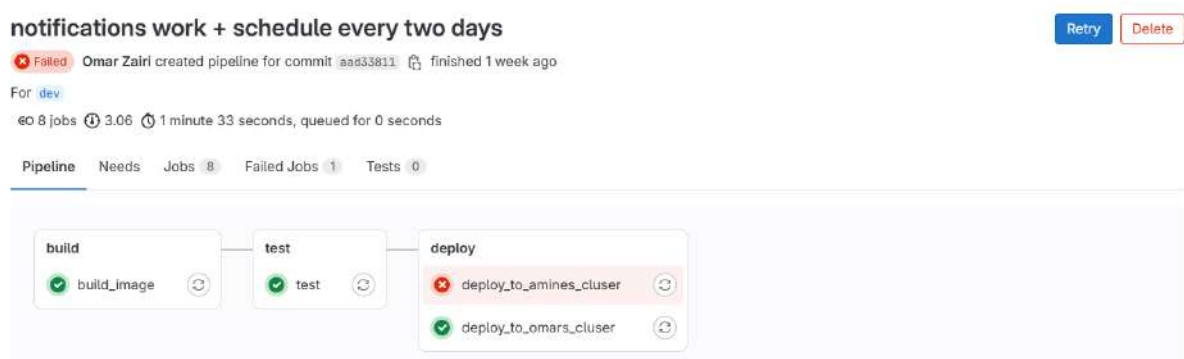


Figure B.9: Details page for the completed CI/CD pipeline following a commit in the dev branch of the Notifications service

Main User Interfaces

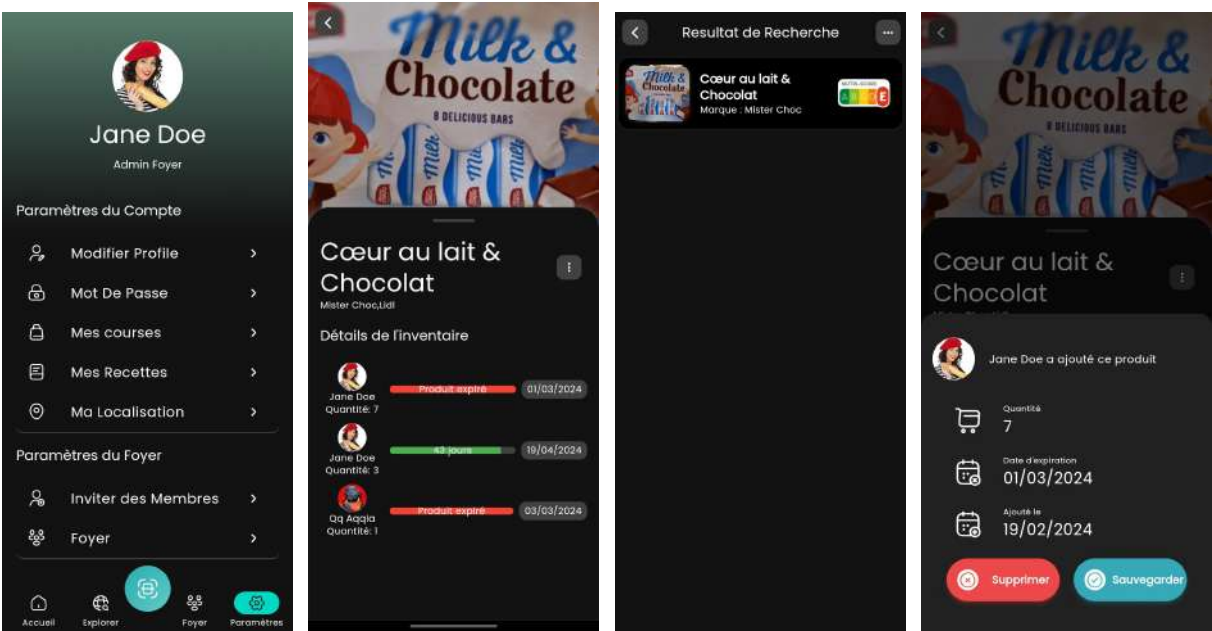


Figure B.10: Account Settings Interface Figure B.11: Inventory Product Details Figure B.12: Text Recognition Results Figure B.13: Inventory Product Management

The Figure B.10 represents the Account Settings interface providing a central hub for users to access and modify their profile details.

The Figure B.11 showcases the detailed information for a specific inventory product. This includes the product itself, the household members who added it, the quantity, and the expiration date.

The Figure B.12 shows the results of a Text Recognition Scan.

The Figure B.13 illustrates the interface for modifying information of an inventory products.

Appendix C

Sprint 2 Additional Content

This is the new Design Sequence Diagram, «Get Users Inventories» It illustrates how we integrated the message broker.

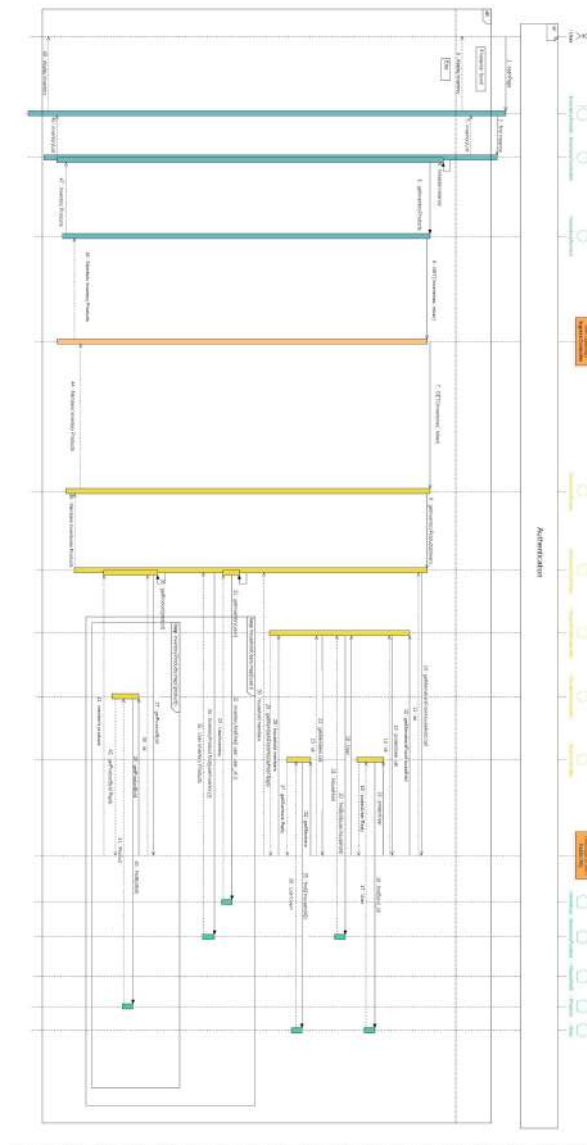


Figure C.1: Design Sequence Diagram «Get Users Inventories»

Sprint two full Backlog

Feature ID	Sprint Feature	ID	User Story	Task			EST
S2_F1	Profile Management	1.3	As a user, I want to recover my password in case I forget it	Back-end	Password Recovery	Create the Password Recovery functionality in the userController.js and add it in the userRouter.js	1h
				Frontend	Password Recovery Interface	Create the modifyProfile functionality in the userService.dart	2h
						add the created modifyProfile functionality in the userController.dart	
		1.5	As a user, I want to logout	Frontend	Log out	Create the recoverPassword.dart	1/2h
						Add an option in the drawerMenu.dart for logging out	
S2_F3	Inventory Management	3.7	As a user, I want to filter my inventory products	Frontend	Filter Inventory by Member and by name	Create the filter functionality in the inventoryController.dart	2h

S2_F4	House-hold Management	4.5	As a user, I want to collaboratively modify products with other members	Frontend	Modify own and members products	Create the modify functionality in the inventoryController.dart	2h
		4.6	As a household admin, I want to manage the members	Backend	Kick Member	Create the kick member functionality in the householdController.js and add it in the houseHoldRouter.js	1h
					Change Role	Create the change role functionality in the householdController.js and add it in the houseHoldRouter.js	1h
				Frontend	Members Management	Create the kick member functionality in the householdServiceer.dart and add it in the householdController.dart	3h
						Create the change role functionality in the householdServiceer.dart and add it in the householdController.dart	
						create the householdScreen.dart to display members	

						create the householdMembersActions.dart screen	
S2_F5	Recommendation System	5.2	As a user, I want to receive recipe suggestions based on available inventory products.	Backend	Recipe Suggestions	Gather a comprehensive dataset of recipes with detailed ingredient lists	10h
						Collect data on user preferences and dietary restrictions	
						capture the user's available ingredients	
						Clean and preprocess the recipe dataset, ensuring consistency in ingredient naming and formatting, and convert it into a suitable format for machine learning	
						Choose an appropriate machine learning model for recipe recommendation	
						Train the selected model on the preprocessed data	
						Evaluate the model by assessing the performance of the trained model using validation datasets	

						Integrate the trained machine learning model with the backend system	
						Develop API endpoints for handling requests related to recipe suggestions based on user inventory	
				Frontend	Recipe Suggestions & Recipe Details Screens	Create the recommendationService.dart	3h
						Create the recommendationController.dart	
						Create the Recipe.dart model	
						Create the Ingredient.dart model	
						Create the Instruction.dart model	
						Create the recipeCard.dart to list the recipes in the explore.dart screen	
						Create to recipeDetails.dart screen	
				Backend	Create Shopping Cart	Create the CartItem.js model	5h
						Create the ShoppingList.js model	
						Create the ShoppingListController.js	
						Create the ShoppingListRouter.js	

S2_F6	Shop- ping list management	6.1	As a user, I want to create different shopping carts for different supermarkets			Create the necessary middlewares	
						Create the mongoDB database	
						Create the necessary files to link the backend with the database	
						Create the create shopping list functionality in the ShoppingList-Controller.js and add it in the ShoppingListRouter.js	
				Frontend	Shop- ping- Cart Cre- ation Form	Create the CartItem.dart model	5h
						Create the ShoppingList.dart model	
						Create the shoppingListService.dart	
						Create the shoppingList-Controller.dart	
						Create the functionality in the shoppingList-Service.js and add it to the shoppingList-Controller.js	
						Create the shoppingList.dart screen	
						Add a button that opens a form for creation of shoppingList	

		6.2	As a user, I want to add products to my shopping cart from my previous inventory	Frontend	Add Product to shopping cart	Add a button in the inventoryProductDetail.dart that opens a form to add products in selected shopping cart	2h
		6.3	As a user, I want to generate a QR-Code of my shopping list	Frontend	Generate Qr Code	create the functionality in the shoppingListController.dart	2h
						Create the QRShoppingList.dart screen	
						Create the functionality to long hold a specific shopping cart to generate qr code	
		6.4	As a user, I want to add products to my shopping cart using the methods featured in Feature2	Backend	Add cart item to shopping cart	Create the add cart item to shopping cart functionality in the ShoppingListController.js and add it in the ShoppingListRouter.js	1h
				Frontend	Add cart item to shopping cart	Using the same methods as adding a product to inventory in the previous sprint.	1h
		6.6	As a user, I want to share shopping carts with household members	Backend	Allow Access to shopping cart for other members	Create the functionality in the ShoppingListController.js and add it in the ShoppingListRouter.js	3h

				Frontend	Allow Access to shopping cart for other members Form	create the functionality in the shoppingList-Serviceoller.dart and add it in the shoppingList-Controller.dart	2h
						Create the shoppingList-Card.dart	
						Add a form that allows access for other members	
		6.7	As a household member, I want to track other members shopping carts	Frontend	Shared Shopping Lists between members	create the functionality in the shoppingList-Controller.dart	2h
				Backend	Email-Sign up	Create the Employee.dart model	1½h
						Create the GMSAccount-Controller.js	
						Create the GMSAccountRouter.js	
						Create the mongoDB database	
						Create the necessary middlewares	
						Create the necessary files to link the backend with the database	
						Create the helpers for google authentication	2h

S2_F8	GMS Account Management	8.1	As an Employee, I want to create an account using different methods		Google Sign up	Create the token.js config file for JWT	
						Create the functionality in the GMSAccount-Controller.js and add it to the GMSAccountRouter.js	
				Frontend	Sign Up Form	Create the sign up form component	1h
						Create the employee service	
						Create the sign up functionality in the service and implement it in the sign up component	
						add the sign up component in the app routing module for easier navigation	
		8.2	As an Employee, I want to login	Backend	Email Login	Create the functionality in the GMSAccount-Controller.js and add it in the GMSAccountRouter.js	½h
					Google Login	Create the helpers for google authentication	
						Create the login functionality in the GMSAccount-Controller.js and add it in the GMSAccountRouter.js	

				Frontend	Log-in Interface	Create the login form component	½h
						Create the employee service	
						Create the login functionality in the service and implement it in the sign up component	
						add the login component in the app routing module for easier navigation	
		8.3	As an Employee, I want to join a GMS	Backend	Accept employees into GMS	Add the approve and abort functionality in the GMSAccount-Controller.js and add it in the GMSAccountRouter.js	2h
						Add the status of pending when a new sign-up request is sent	
				Frontend	Accept employees into GMS Interface	Add the functionality to the employee service	3h
						Add a table into the employee's component that shows a list of requesting employees	
				Backend	Employee Management	Create the delete employee functionality in the GMSAccount-Controller.js and add it in the GMSAccountRouter.js	3h

		8.4	As a Manager, I want to manage employees			Create the accept employee functionality in the GM-SAccountCon-troller.js and add it in the GMSAccoun-tRouter.js	
						Create the assign role to employee functionality in the GMSAccount-Controller.js and add it in the GMSAccoun-tRouter.js	
						Create the get all employees functionality in the GMSAccount-Controller.js and add it in the GMSAccoun-tRouter.js	
				Frontend	Em- ployee Man- age- ment interface	Create the dashboard component	3h
						Create the employee component	
						Create the delete employee functionality in the employee service and implement in the employee component	

						Create the assign role to employee functionality in the employee service and implement in the employee component	
						Create the accept employee functionality in the employee service and implement in the employee component	
				Backend	Cashier session Socket	Create a socket server in the GMS Microservice	5h
		8.5				Save the GMS cashier socket path to the corresponding supermarket	
						Create the web socket events listeners and emitters for joining rooms and synchronizing sessions	
				Frontend	Cashier Session Web Interface	Create the cashier service	5h
						Create the cashier component and add a QR code dialogue	
						Add the corresponding events and listeners to the component	

					Cashier Session Mobile Interface	Create the Cashier Session.dart service and add the socket events listeners and emitters	5h
						Create the cashier session controller	
						Create the cashier scan QR Code screen and redirect when the connection is established	
S2_F9	Checkout Interactions	9.1	As a Cashier, I want to scan clients' shopping cart's generated QR code to retrieve the products in question	Backend	Cashier Carts Scanning Session	Create the carts socket listeners and emitters After establishing a connection	2h
						Add the scanning products by barcode functionality to the ProductController.js and use it in the socket listener	
				Frontend	Cashier Scanning Carts Mobile Session	Add the functionality's socket listeners and emitters to scan shopping carts in the socket service	5h
						Add the functionality to scan carts in the cashierSession-Controller.dart	
						Create an interface to scan carts generated QR codes	

					Cashier Scan- ning carts Web session	Add the corre- sponding listen- ers to the cashier service	4h
						Create a check- out component and listen to the scanning events emitted from the mobile synchronized session and show the transaction	
Global Tasks				Configure and implement the mes- sage broker for an asynchronous in- ter communication between the mi- croservices			10h
				Setup the recommendation engine python project			2h
				Setup the shopping list manage- ment microservice			1h
				Setup the GMS management sys- tem			1h
				Setup the supermarket angular frontend by initializing the neces- sary modules and structure of the project			5h
				Create the Sequence Diagram			1h
				Create the Class Diagram			2h
				Create a Deployment Diagram			2h
				Create Test scripts			3h
				Build and push a Docker image for each feature			$\frac{1}{2}$ h
				create the Kubernetes configuration YAML files(deployments, services, ingress controller. . .)			1h
				set up a simple ci/cd pipeline to automate testing, building and de- ployment			1h
Total Estimation				~120 hours			

Table C.1: Second Sprint backlog

Tests

This is the unit test for the Get Shopping List By ID functionality. We tested it by creating a mockShoppingList variable to ensure the coherence and accuracy of the data received.

```
> shoppinglistmanagement@1.0.0 test
> mocha 'tests/**/*.test.js'

Channel Created!
DB Connected Successfully
(node:2972) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use "node --trace-warnings ..." to show where the warning was created)
(node:2972) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

Get ShoppingList
✓ Get ShoppingLists should return a shoppinglist (478ms)

1 passing (480ms)
```

Figure C.2: Get Shopping List By ID unit Test

This is the unit test for the Get Employees List functionality. We tested it by creating a mockEmployees variable to ensure the coherence and accuracy of the data received.

```
DB Connected Successfully
(node:20968) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use "node --trace-warnings ..." to show where the warning was created)
(node:20968) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
DB Connected Successfully

Get Employees
✓ Get Employees should return a list of employees (481ms)
```

Figure C.3: Get Employees List unit Test

Appendix D

Sprint 3 Additional Content

Sprint three full Backlog

Feature ID	Sprint Feature	ID	User Story	Task			EST
S3_F9	Check-out Interactions	9.1	As a Cashier, I want to scan clients shopping cart's generated QR code to retrieve the products in question	Backend	Cashier Carts Scanning Session	Create the carts socket listeners and emitters After establishing a connection	2h
						Add the scanning products by barcode functionality to the ProductController.js and use it in the socket listener	
				Frontend	Cashier Scanning Carts Mobile Session	Add the functionality's socket listeners and emitters to scan shopping carts in the socket service	5h

						Add the functionality to scan carts in the cashierSession-Controller.dart	
						Create an interface to scan carts generated QR codes	
					Cashier Scanning carts Web session	Add the corresponding listeners to the cashier service	4h
						Create a check-out component and listen to the scanning events emitted from the mobile synchronized session and show the transaction	
		9.2	As a user, I want to receive the final price of all of my products	Backend	Total Shopping cart price	Create the Stock.js model	3h
						Create the stock-Controller.js	
						Create the stock-Router.js	
						Create the get total price functionality in the stockController.js and add it in the stockRouter.js to create its endpoint.	
						Add the endpoint in the supermarket database to get the total price from the supermarket to the frontend.	

				Frontend	Total Shopping cart price	Create the supermarketService.dart	5h
						Create the supermarketController.js	
						Create the dynamic GMSService.dart	
						Create the DynamicsupermarketController.dart	
						Create the the get total price functionality in the supermarketService.js and implement it in the DynamicSupermarketController.js	
						Create the QRShoppingScreen.dart	
		9.3	As a cashier, I want to finalize the transaction	Backend	Approve transaction	Create the Transaction.js model	5h
						Create the transactionController.js	
						Create the transactionRouter.js	
						Create the Approve transaction functionality in the transactionController.js and add it in the transactionRouter.js	
						Add the approve event to the cashier socket	

					Abort Transaction	Create the Abort transaction functionality in the transactionController.js and add it in the transactionRouter.js	
				Frontend	Approve transaction	Add the functionality in the cashier service and implement it in the checkout component	
					Abort transaction	Add the functionality in the cashier service and implement it in the checkout component	
S3_F3	Inventory management	3.3	As a user, I want to attach notes to a product in my inventory to let other users in my household access it	Backend	Attach note	Add the functionality in the inventoryController.js and implement it in the inventoryRouter.js	3h
					Delete note	Add the functionality in the inventoryController.js and implement it in the inventoryRouter.js	
				Frontend	Attach note interface	Create the functionality in the inventoryService.dart and implement it in the inventoryController.dart	2h

					Delete note Button	Add a button inside the inventoryProductDetails.dart that opens a form to attach a note to an inventory product.	
						Create the functionality in the inventoryService.dart and implement it in the inventoryController.dart	
						Add a button inside the inventoryProductDetails.dart that deletes a note from an inventory product.	
S3_F6	Shopping list management	6.5	As a user, I want to mark products in my shopping cart as bought so they get added automatically to my inventory	Back-end	Mark Product As Bought	Create the functionality in the shoppingListController.js and add it in the shoppingListRouter.js	2h
				Frontend	Mark Product As Bought	Create the functionality in the shoppingListService.dart and add it in the shoppingListController.dart	1h

						Add a button in the shoppingList-Card.dart interface to mark a product as bought	
S3_F8	GMS Account Management	8.3	As an Employee, I want to join a GMS	backend	Accept employees into GMS	Add the approve and abort functionality in the GMSAccount-Controller.js and add it in the GMSAccountRouter.js	1h
						Add the status of pending when a new sign-up request is sent	
						Add the functionality to the employee service	
						Add a table into the employee's component that shows a list of requesting employees	
						Add the approve and abort functionality in the GMSAccount-Controller.js and add it in the GMSAccountRouter.js	
						Add the status of pending when a new sign-up request is sent	
				frontend		Add the functionality to the employee service	3h

					Accept employees into GMS Interface	Add a table into the employee's component that shows a list of requesting employees	
		8.5	As a Cashier, I want to synchronize my mobile app session with the GMS system	frontend	Cashier session Socket	Create a socket server in the GMS Microservice Save the GMS cashier socket path to the corresponding supermarket	5h
				backend	Cashier Session Web Interface	Create the web socket events listeners and emitters for joining rooms and synchronizing sessions Create the cashier service Create the cashier component and add a QR code dialogue Add the corresponding events and listeners to the component	5h
					Cashier Session Mobile Interface	Create the Cashier Session.dart service and add the socket events listeners and emitters	5h

						Create the cashier session controller	
						Create the cashier scan QR Code screen and redirect when the connection is established	
S3_F10	Stock tracking	10.1	As a Manager, I want the stock to be updated automatically after transactions	Back-end	Remove quantity from stock	Create the functionality that automatically updates the stock upon approving a transaction in the stockController.js and add it in the stockRouter.js.	3h
		10.2	As a Manager, I want to receive out-of-stock alerts	Backend	Get Alerts	Create the StockAlert.js model	2h
						Create the functionality that creates an alert when a product's quantity reaches 0 in the stockController.js and add it in the stockRouter.js	
				Frontend	Alerts table	Create the alerts component	2h
						Add the functionality in the stock service and implement it in the alerts component	

		10.3	As a Manager, I want to delete out-of-stock alerts	Back-end	Clear Alert	Create the functionality that clears an alert in the stock-Controller.js add it in the stockRouter.js	1h
				Frontend	Clear Alert	Add the functionality in the stock service and implement it in the alerts component	2h
						Add a button to clear an alert	
		10.4	As a Manager, i want to receive analytics of clients consumption trends to better optimize stock management	Backend	analyzing clients' consumption trends	create Extract.py and query the database to extract relevant client-buying data	3h
						create Transform.py and Preprocess the extracted data	5h
						create Load.py and Aggregate the preprocessed data	3h
						create the mongodb chart	2h
				Front-end	visualizing clients consumption trends	integrate the chart in the dashboard	2h
		10.5	As a manager, I want to add a stock of products	Back-end	Add Stock	Create the functionality in the stockController.js and add it in the stockRouter.js	1h

				Frontend	Add Stock interface	Create a dialog form to add a product	2h
						Create the functionality in the stock service and implement it in the dialog form component	
		10.6	As a manager, I want to update a stock of products	Back-end	Update Stock	Create the functionality in the stockController.js and add it in the stockRouter.js	1h
				Frontend	Update Stock interface	Create the functionality in the stock service and implement it in the dialog form component	2h
		10.7	As a manager, I want to delete a stock of products	Back-end	Delete Stock	Create the functionality in the stockController.js and add it in the stockRouter.js	1h
				Frontend	Delete Stock Button	Create the functionality in the stock service and implement it in the stock component	2h
						Add a delete button	
13	Shop Flow Admin Management	13.1	As an Admin, I want to login to access the admin dashboard	Backend	Email Login in	Create the shopFlowAdmin.js model	1h
						Create the shopFlowAdminController.js	
						Create the shopFlowAdminRouter.js	

						Create the necessary middlewares	
						Create the functionality in the shopFlowAdminController.js and add it in the shopFlowAdminRouter.js	
		13.2	As an admin, I want to add a supermarket and its API paths to the shop flow system to allow users to shop there	Frontend	Email login Interface	Create the shopFlowAdmin service	1h
						Create the adminLogin component	
						Create the functionality in the shopFlowAdmin service and implement it in the adminLogin component	
		13.2	As an admin, I want to add a supermarket and its API paths to the shop flow system to allow users to shop there	Backend	Add supermarket	Create the Supermarket.js model	2h
						Create the supermarket-Controller.js	
						Create the supermarketRouter.js	
						Create the mongoDB database	
						Create the necessary files to link the backend with the database	
						Create the functionality in the supermarket-Controller.js and add it in the supermarketRouter.js	

				Frontend	Add super-market details	Create the add supermarket component	3h
						Create the functionality in the shopFlowAdmin service and implement it in the add supermarket component	
					Display super-mar-kets in the mobile app-lication	Create the Supermarket.dart model	3h
						Create the SupermarketService.dart	
						Create the supermarket-Controller.dart	
						Create the supermarket-Card.dart view	
						Create the supermar-ketList.dart screen	
						Create the functionality in the supermarket-Service.dart and implement it in the supermarket-Controller.dart	
					Setup the Supermarket management microservice		2h
					Setup the ShopFlow Admin angular frontend by initializing the necessary modules and structure.		1h
					Create the Class Diagram		2h
					Create the Activity Diagram		2h
Global Tasks					Create the Package Diagram		2h
					Build and push a Docker image for each feature		1h
					Create Tests scripts		1h

	create the kubernetes configuration YAML files(deployments, services, ingress controller...)	1h
	set up a simple ci/cd pipeline to automate testing, building and de- ployment	1h
Total Estimation		~110 hours

Table D.1: Third Sprint Backlog

Tests

This is the unit test for the Get Stock List functionality. We tested it by creating a mockStockList variable to ensure the coherence and accuracy of the data received.

```
DB Connected Successfully
(node:23864) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use 'node --trace-warnings ...' to show where the warning was created)
(node:23864) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

StockController
  ✓ getAll should return stocks (676ms)

1 passing (680ms)
```

Figure D.1: Get Stock List unit Test

This is the unit test for the Get Supermarket By Id functionality. We tested it by creating a mockSupermarket variable to ensure the coherence and accuracy of the data received.

```
DB Connected Successfully
(node:9728) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use 'node --trace-warnings ...' to show where the warning was created)
(node:9728) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

Get Supermarket By Id
  ✓ Get Supermarket By Id should return a supermarket (491ms)

1 passing (493ms)
```

Figure D.2: Get Supermarket By ID unit Test

Deployment of the third increment

ShopFlow / ShoppingListManagement / Jobs / #6944691991

deploy_to_azure_cluser

Passed Started 5 days ago by Omar Zairi

Search job log

```
1 Running with gitlab-runner 17.0.0-pre.88.g761ae5dd (761ae5dd)
2 on green-6.saas-linux-small-amd64.runners-manager.gitlab.com/default YKxHMyexq, system ID: s_a201ab37b78a
3 Preparing the "docker+machine" executor 00:09
4 Preparing environment 00:03
5 Getting source from Git repository 00:01
6 Executing "step_script" stage of the job script 00:03
7 Cleaning up project directory and file based variables 00:00
82 Job succeeded
```

Duration: 18 seconds
Finished: 5 days ago
Queued: 0 seconds
Timeout: 1h (from project)
Runner: #32976645 (YKxHMyexq) 6-green.saas-linux-small-amd64.runners-manager.gitlab.com/default
Commit: 02f4ea98
dep
Pipeline #1306110207 Passed
deploy

Figure D.3: Deploying of ShoppingList Management microservice

ShopFlow / InventoryManagement / Jobs / #6944694466

deploy_to_azure_cluser

Passed Started 5 days ago by Omar Zairi

Search job log

```
1 Running with gitlab-runner 17.0.0-pre.88.g761ae5dd (761ae5dd)
2 on green-2.saas-linux-small-amd64.runners-manager.gitlab.com/default ns46NMmJT, system ID: s_05d7af104313
3 Preparing the "docker+machine" executor 00:10
4 Preparing environment 00:02
5 Getting source from Git repository 00:02
6 Executing "step_script" stage of the job script 00:02
7 Cleaning up project directory and file based variables 00:01
93 Job succeeded
```

Duration: 18 seconds
Finished: 5 days ago
Queued: 0 seconds
Timeout: 1h (from project)
Runner: #12270648 (ns46NMmJT) 2-green.saas-linux-small-amd64.runners-manager.gitlab.com/default
Commit: 8a108f11
dep
Pipeline #1306110685 Passed
deploy

Figure D.4: Deploying of Inventory Management microservice

Appendix E

Sprint 4 Additional Content

Sprint four full Backlog

Feature ID	Sprint Feature	ID	User Story	Task			EST
S4_F5	Recommendation System	5.1	As a user, I want to receive product suggestions based on my habits, and the products available in my inventory	Backend	Product Suggestions	Gather a comprehensive dataset of products and client past product lists	12h
						Collect data on user preferences, habits and dietary restrictions	
						capture the user's available inventory products.	
						Clean and pre-process the recipe dataset, ensuring consistency in product naming and formatting, and convert it into a suitable format for machine learning	

						Choose an appropriate machine learning model for product recommendation.	
						Train the selected model on the preprocessed data.	
						Evaluate the model by assessing the performance of the trained model using validation datasets.	
						Integrate the trained machine learning model with the backend system	
						Develop API endpoints for handling related request	
				Frontend	Product Suggestions screen	create Product-Suggestion.dart model	8h
						create Product-SuggestionService.dart	
						create Product-SuggestionController.dart	
						create Product-Suggestion.dart screen	
						Add the ProductSuggestion-Screen to the routes for easier navigation	

S4_F7	History and statistics	7.1	As a user, I want to consult my purchase-history	Backend	Pur- chase History	Create Purchase.js model	1 1/2h
						Create PurchaseController.js	
						Create PurchaseRouter.js	
				Frontend	Pur- chase History Interface	Create Purchase.dart model	5 1/2h
						Create PurchaseService.dart	
						Create PurchaseController.dart	
						Create the PurchaseList.dart Screen	
		7.2	As a user, I want to consult the purchase-history of my household members	Back- end	House- hold pur- chase history	Add the functionality to the PurchaseController.js and implement it in the PurchaseRouter.js	3h
						Add the groupedHouseHoldPurchaseHistory functionality to the PurchaseController.dart	
				Frontend	Pur- chase History Interface	Add the Purchase History screen to the routes for easier navigation	2h
				Backend	Con- sump- tion Statistics	create Extract.py and query the database to extract relevant inventories, products and client data	7h

		7.3	As a user, I want to obtain statistics on my food consumption			create Transform.py and Preprocess the extracted data	
						create Load.py and Aggregate the preprocessed data	
						create app.py and develop an API endpoint	
				Frontend	Consumption Statistics Interface	Create the Consumption Statistic.dart model	6h
						Create the Consumption StatisticService.dart	
						Create the Consumption StatisticController.dart	
						Create the Consumption StatisticScreen.dart	
						Add the Consumption StatisticScreen to the routes for easier navigation	
		7.4	As a user, I want to consult statistics on my household members' consumptions	Frontend	Consumption Statistics Interface	Add the functionality in the Consumption StatisticController.dart	3h

S4_F11	Statistics and analysis	11.1	As a Manager, I want to visualize clients' consumption habits (most bought, added to inventories products)	Backend	analyzing clients' buying consumption habits	create <code>tract.py</code> and query the database to extract relevant client-buying data	7h
						create <code>Transform.py</code> and <code>Preprocess</code> the extracted data	
						create <code>Load.py</code> and <code>Aggregate</code> the preprocessed data	
						create <code>app.py</code> and develop an API endpoint / or <code>mongodb</code> chart	
				Front-end	visualizing clients buying consumption habits	integrate the chart in the dashboard	2h
		11.2	As a Manager, I want to visualize most trending products across different groups of clients (by age)	Backend	analyzing products across different groups	create <code>tract.py</code> and query the database to extract relevant inventories, products and client data	6h
						create <code>Transform.py</code> and <code>Preprocess</code> the extracted data	
						create <code>Load.py</code> and <code>Aggregate</code> the preprocessed data	

				Frontend	visualizing products across different groups	integrate the chart in the dashboard	2h
		11.4	As a Manager, I want to visualize consumption trends per product category	Backend	analyzing product consumption per categories	<div>create Extract.py and query the database to extract relevant consumption data</div> <div>create Transform.py and Preprocess the extracted data</div> <div>create Load.py and Aggregate the preprocessed data</div> <div>create app.py and develop an API endpoint / or mongodb chart</div>	8h
				Frontend	visualize consumption categories	integrate the chart in the dashboard	2h
				Backend	Create Promotional Offers	Add the promotion attribute in the Stock.js model	4h

12	Management of promotional offers	12.1	As a Manager, I want to create promotional offers for certain products			Create PromotionController.js	
						Create PromotionRouter.js	
				Frontend	Promotional Offers Interface	create the promotion component	7h
		12.2	As a Manager, I want to take promotional offer off a stock	Backend	Edit Promotional offers	create the Promotion service	
						create the functionality in the service and implement it in the promotion component	
				Frontend	Promotional Offers Interface	add the functionality to take a stock off promotion in the StockController.js	2h
						add the endpoint to the router	
						add the functionality to the StockService.ts	1 1/2h
						add the functionality to the StockComponent.ts	
						add a button that deletes	
		12.3	As a Manager, I want to target and create promotional offers based on consumer segments.	Backend	Consult client segments and create promotional offers	create Extract.py and query the database to extract relevant segments' consumption data	7h
						create Transform.py and Preprocess the extracted data	

						create Load.py and Aggregate the preprocessed data	
				Frontend	Add Promotional offers from the client segments	create the Promotion.ts Service	5h
						create the PromotionTable component	
						add the PromotionTable to the routes	
		12.4	As a Manager, I want to monitor the effectiveness of promotional offer campaigns	Backend	monitor sales of products during the promotional period	create Extract.py and query the database to extract relevant promotion data	8h
						create Transform.py and Preprocess the extracted data	
						create Load.py and Aggregate the preprocessed data	
				Frontend	interface for monitoring products	create the PromotionEffectivenessChart component	5h
						integrate the PromotionEffectivenessChart component in the dashboard	
		12.5	As a user, I want to see products on promotion	Backend	Products on promotion	add the functionality of receiving promotional products	3h
				Frontend	Products on promotion interface	create the PromotedProduct.dart and the SupermarkedOffers.dart models	5h

						create the PromoProduct-Controller.dart	
						create the PromoProductCard.dart component	
						integrate the PromoProductCard.dart in the explore page	
Global Tasks				Setup the product recommendation system Python project			1h
				Setup the user statistics Python project			1h
				Build and push a Docker image for each feature			1h
				Create Tests scripts			1h
				create the kubernetes configuration YAML files(deployments, services, ingress controller...)			1h
				create the sequence diagrams			2h
				set up a simple ci/cd pipeline to automate testing, building, and deployment(locally while developing)			1h
				set up the Kubernetes Cluster in the cloud			5h
				set up the ci/cd pipeline to deploy to our cloud Kubernetes cluster			2h
Total Estimation						~135 hours	

Table E.1: Fourth Sprint Backlog

Tests

```
> shoppinglistmanagement@1.0.0 test
> mocha 'tests/purchaseHistort.test.js'

Channel Created!
DB Connected Successfully
(node:12964) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:12964) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

Get Purchases
✓ Get Purchases should return a Purchases (752ms)

1 passing (754ms)
```

Figure E.1: Get Purchase List unit Test

This is the unit test for the Get Purchase List functionality. We tested it by creating a mockPurchase variable to ensure the coherence and accuracy of the data received.

Deployment of the fourth increment

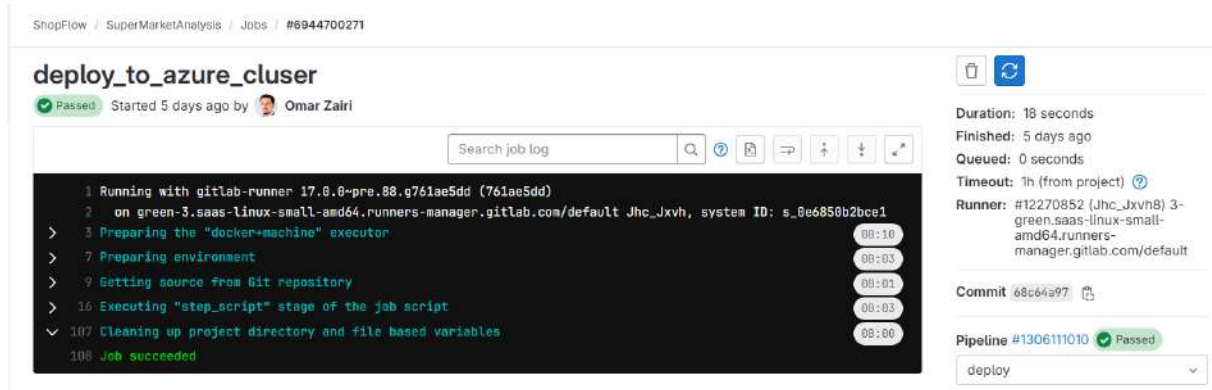


Figure E.2: Deploying of Supermarket Analysis microservice

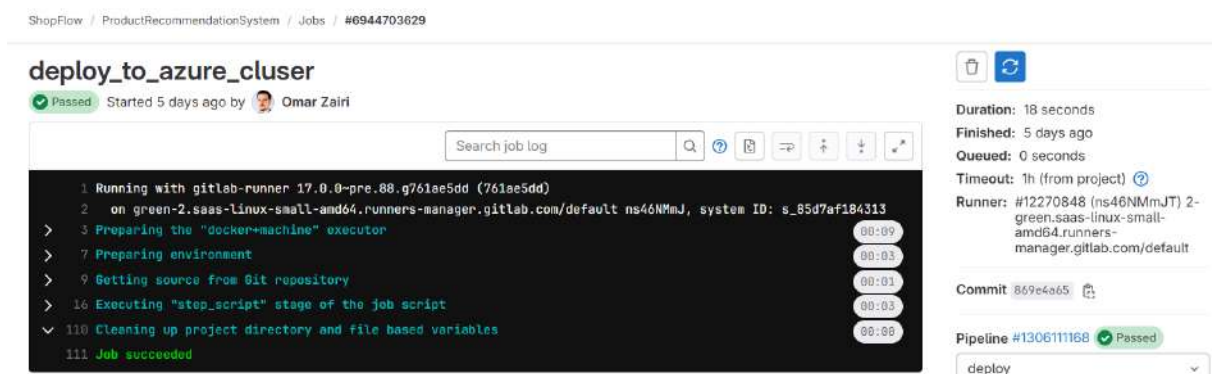


Figure E.3: Deploying the Product Recommendation microservice

System Monitoring

The figures below represent some of the metrics of our system.

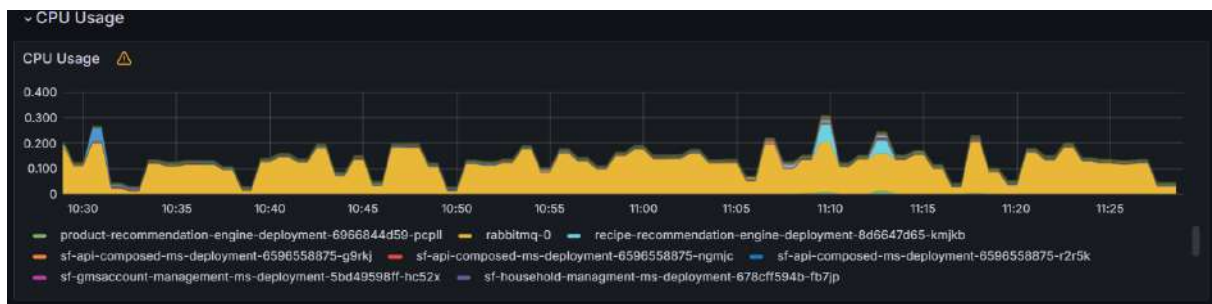


Figure E.4: CPU Usage of our Kubernetes Pods

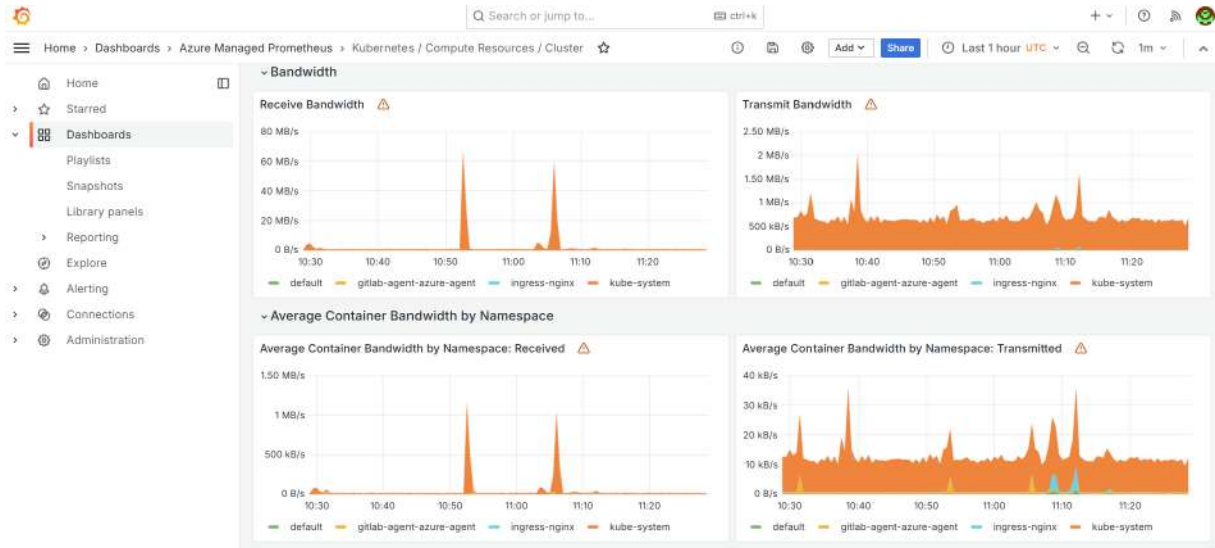


Figure E.5: Bandwidth for our Kubernetes components classified by Namespaces

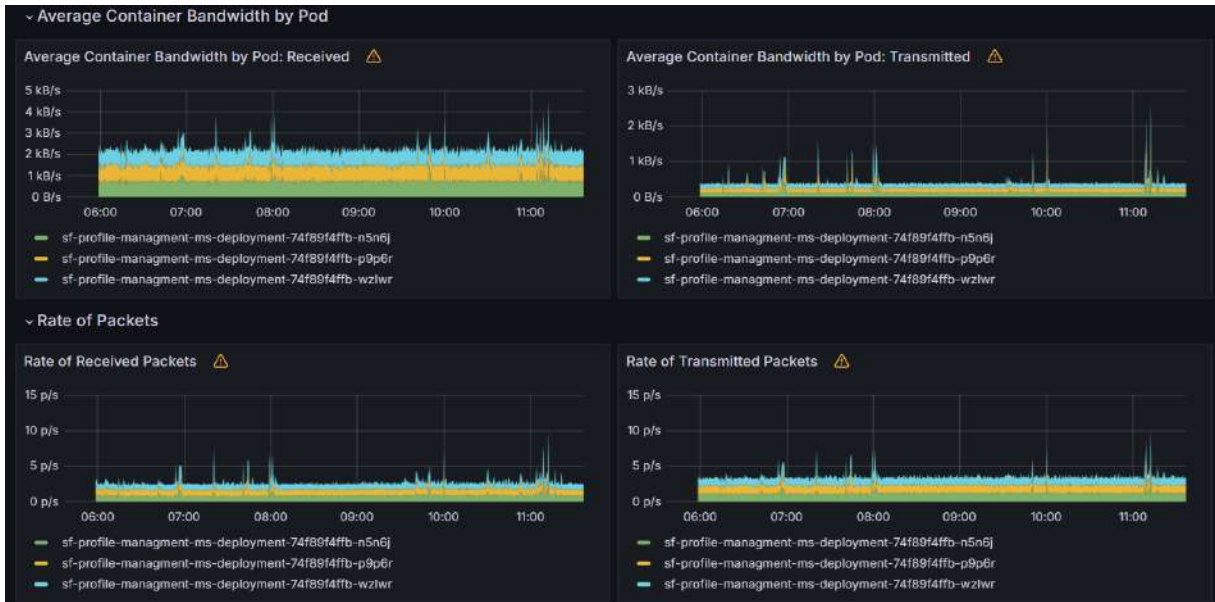


Figure E.6: Average Bandwidth for the profile management Pods

تلخيص

يقدم هذا التقرير لغرض الحصول على الإجازة الوطنية في تطوير البرمجيات. تم تنفيذ مشروع شوبفلو تحت إشراف شركة زاد كونسولتينق ويهدف إلى تحسين إدارة الأغذية للمستهلكين والمساحات الكبرى والمتوسطة. تركّز المنصة على تقليل هدر الطعام وتحسين إدارة المخزون من خلال تقنيات البيانات المتقدمة.

الكلمات المفتاحية : سكّرام، الخدمات المصغرة، التطوير العملي المستمر، تطوير الويب والجوال، فلاتر، أنجولار، نودجي إس، بايثون، تعلم الآلة، ذكاء الأعمال، كوبيورناتيس، التكامل المستمر\التسليم المستمر

Résumé

Ce rapport est soumis dans le cadre de l'obtention d'une licence nationale en Développement des Systemes d'information. Le projet, "ShopFlow," a été réalisé sous la supervision de Z Consulting et vise à améliorer la gestion des aliments et des courses pour les consommateurs et les Grandes et Moyennes Surfaces (GMS). La plateforme se concentre sur la réduction du gaspillage alimentaire et l'optimisation de la gestion des stocks grâce à des techniques avancées de données.

Mots clés : Scrum, Microservices, DevOps, Développement Web et Mobile, Flutter, Angular, Node.js, Python, Apprentissage Automatique, Intelligence d'Affaires, Kubernetes, CI/CD

Abstract

This report is submitted for the purpose of obtaining a national Bachelor's degree in Software Development. The project, "ShopFlow," was carried out under Z Consulting and aims to enhance food and grocery management for consumers and Grand and Medium-sized Surfaces (GMS). The platform focuses on reducing food waste and optimizing inventory management through advanced data techniques.

Keywords : Scrum, Microservices, DevOps, Mobile and Web Development, Flutter, Angular, Nodejs, Python Machine Learning, Business Intelligence, Kubernetes, CI/CD.