Smart Shopper

MID-TERM REPORT

Mustafa Noori
Graduation
Fontys University of Applied Sciences
October 24 2023
Student number: 3014630
Venlo, Limburg

Date:	22/209/2023
Version:	0.1
State:	Done
Author:	Mustafa Noori
Total Words:	7580
Words Intro-Conclusion:	5680

Version history

Version	Date	Author(s)	Changes	State
0.1	24/10/2023	Mustafa Noori	Analysis phase, Design phase, Research	Done

Distribution

Version	Date	Receivers
0.1	24/10/2023	Sander Bruinsma
		Jelle Baede
		Denis Gorianin

Table of Contents

Table of Table

Table of Figures

STATEMENT OF AUTHENTICITY

Issued by the FHTenL Examination Board, September 2017

I, the undersigned, hereby certify that I have compiled and written this document and the underlying work / pieces of work without assistance from anyone except the specifically assigned academic supervisor. This work is solely my own, and I am solely responsible for the content, organization, and making of this document and the underlying work / pieces of work.

I hereby acknowledge that I have read the instructions for preparation and submission of documents/pieces of work provided by my course / my academic institution, and I understand that this document and the underlying pieces of work will not be accepted for evaluation or for the award of academic credits if it is determined that they have not been prepared in compliance with those instructions and this statement of authenticity.

I further certify that I did not commit plagiarism, did neither take over nor paraphrase (digital or printed, translated or original) material (e.g., ideas, data, pieces of text, figures, diagrams, tables, recordings, videos, code, ...) produced by others without correct and complete citation and correct and complete reference of the source(s). I understand that this document and the underlying work / pieces of work will not be accepted for evaluation or for the award of academic credits if it is determined that they embody plagiarism.

Name (in capital letters):	MSUTAFA NOORI
Student number:	3014630
Place / Date:	Venlo, 10-10-2023
Signature:	MN

Abstract

This midterm report for the Smart Shoppe project offers a comprehensive update on the development of a ground-breaking mobile app created to streamline the shopping process for customers (expats) looking for the greatest deals (cheap) at nearby supermarkets. Insights into the advancements made in research, technology stack selection, and application development are provided in this paper. The application's essential features, such as real-time pricing comparison, grocery list management, and location-based store information, are highlighted. Underscoring the dedication to providing a useful solution that enables consumers to make informed shopping selections while reducing their food costs, challenges encountered, and upcoming milestones are also mentioned.

1 Introduction

1.1 Company: CGI

CGI is a global IT and business consulting services company founded in 1976. With about 90,000 professionals across 400 locations[1], they provide comprehensive, scalable consulting services across 21 industries. In CGI Eindhoven, where I'll be working, teams of 200 professionals focus on innovative technology solutions to address practical challenges in various sectors[1]. CGI's expertise covers finance, government, healthcare, energy, and telecommunications[1].

1.2 Context

There have been considerable changes to the purchasing environment recently. Ease has arrived at customers' doorsteps thanks to the growth of technology for online purchasing. But at the same time, several problems have been faced despite this ease. To accomplish their shopping, consumers are now looking for more efficient and economical methods.

1.3 Background

The Smart Shopper project has been considered to address the complex challenges of expatriates' grocery shopping experience. Today's expatriates face several issues while shopping for groceries, such as unpredictable price variations, limited product information, complex decision-making, time-consuming processes, and missed opportunities for savings.

Recognizing these challenges, CGI has undertaken the Smart Shopper project with the aim of simplifying expats' grocery shopping experience. The project's primary goal is to provide expatriates with a user-friendly mobile application that enables them to create grocery lists, receive real-time price information, discover current offers, and optimize their shopping across various supermarkets.

[1] (*Company overview*. n.d.)

2 Project Overview

The Smart Shopper assignment aims to transform the grocery shopping experience by developing a mobile application.

2.1 Project Objective

The main objective of the Smart Shopper project is to design and develop an intuitive mobile app that will allow expatriates to create, manage, and optimize their grocery lists efficiently. The application will have a store finder feature to find nearby supermarkets and view their details.

To define the SMART objectives of the Smart Shopper project.

- Specific: Develop an app that allows users to create and compare grocery lists across various supermarkets.
- Measurable: Achieve a 20% reduction in users' monthly grocery expenses.
- Achievable: Complete the development within the given timeline.
- Relevant: Provide a user-friendly interface with 2 languages.
- Time-bound: Complete the project within the given timeframe.

2.2 Approach & Phasing

Objective: To detail the approach and phases planned for the project.

2.2.1 Approach

The project will explore the availability of supermarket APIs and select the technology stack. Define user requirements, acceptance criteria, and deliverables. Create project architecture, user interface, and database structure.

2.2.2 Phasing

The project phases are setting up project infrastructure and team. Investigate supermarket APIs and select the technology stack. Define user requirements and project scope.

Name(s)	Role
Johan Peters	Project Supervisor
Jelle Baede	CGI Coach
Denis Gorianin	Stakeholder
Mustafa Noori	Developer

Table 1: Project Team

2.3 Scope

To explain the project's scope, including its various components and limitations. The scope of the Smart Shopper project covers the following areas in Table 2: Scope:

Scope areas	Explain scope
Business process / portfolio	This includes improving consumers' grocery shopping experience and managing different data collection of products from supermarkets.
Geographical	Availability for expatriates in the Netherlands
Organizational	Collaboration with supermarkets and gathering real-time pricing data.
Application	It outlines the features like to create a shopping list, look at products, and compare prices.
Information	It relates to the data processed like product pricing, discounts, promotions, user-generated grocery lists, and user profiles.
Infrastructure	This involves applying CGI's computer and network infrastructure to ensure efficient development and deployment of the mobile app.

Table 2: Scope

2.4 Out of Scope

To give a clear picture and prevent unrealistic expectations, it is important to specify what is outside of the scope:

- **Local Stores:** Repositories for information that aren't available to the public.
- **Store Operations:** The project is not in charge of overseeing the actual layout or running of the store.
- **Payment:** Although the app offers price information, it does not permit in-app purchases.
- **Delivery Services:** No service options are offered by the app.
- **Expansion:** The app's geographic reach is restricted to a certain area, and it does not function abroad.
- **Distribution:** The app will not be made available through open channels like app stores.
- A multilingual app with many languages: (n > 2).

2.5 Project Deliverables

The project has been divided into the following milestones which represent the important aspects of this project.

Phase	Milestones	Planned date	Major Deliverables	Comments
Project Preparation	M1: Project Initiation & Team Formation	01-09-23	Project Plan, Team Formation	Initiate the project and establish the team.
Design	M2: Requirements Analysis & System Design	01-10-23	Requirements Documentation, System Design	Define project scope, design the system.
Realization	M3: Application Development & Testing	15-10-23	Developed Application, Testing Reports	Build and thoroughly test the application.

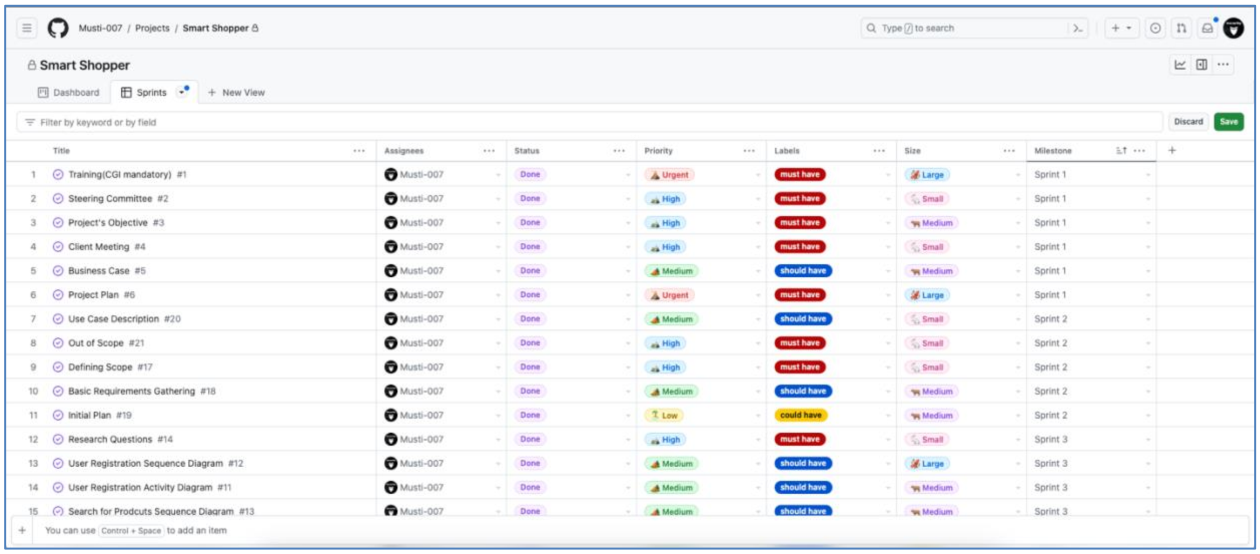
Final Preparation	M4: Infrastructure Setup & Data Integration	01-11-23	Infrastructure Ready, Data Integrated	Prepare technical environment and data.
Go Live and Support	M5: Application Deployment & User Support	01-12-23	Application Deployed, User Support in Place	Launch the application and support users.
End of Project	M6: Project Evaluation & Closure	15-01-24	Project Evaluation Report, Assessment, Closure Activities	Assess project outcomes, close the project.

Table 3: Project Deliverables

2.6 Project Management

Project management is needed to ensure the effective planning, performance, and completion of all project tasks and objectives. In the developing process of the Smart Shopper application, an agile approach is used by conducting a daily stand-up and a two-week sprint. This will allow to break down the project into smaller chunks, which will be more manageable and easier to focus tasks which are called use cases.

Each sprint's initial tasks will be placed according to their complexity and impact on the project using GitHub labels as you can see below in Figure 1. This made it easier to focus on activities that involved using the must have, should have, and could have criteria of the MoSCoW principle. A sprint retrospective will be done at the conclusion of each sprint to assess results and identify potential areas for improvement. This includes analyzing what went well throughout the sprint and identifying any difficulties or problems that surfaced.


Musti-007 / Projects / Smart Shopper
Type / to search
Smart Shopper
Dashboard
Sprints
New View
Filter by keyword or by field
Discard
Save
You can use Control + Space to add an item

	Title	Assignees	Status	Priority	Labels	Size	Milestone
1	Training(CGI mandatory) #1	Musti-007	Done	Urgent	must have	Large	Sprint 1
2	Steering Committee #2	Musti-007	Done	High	must have	Small	Sprint 1
3	Project's Objective #3	Musti-007	Done	High	must have	Medium	Sprint 1
4	Client Meeting #4	Musti-007	Done	High	must have	Small	Sprint 1
5	Business Case #5	Musti-007	Done	Medium	should have	Medium	Sprint 1
6	Project Plan #6	Musti-007	Done	Urgent	must have	Large	Sprint 1
7	Use Case Description #20	Musti-007	Done	Medium	should have	Small	Sprint 2
8	Out of Scope #21	Musti-007	Done	High	must have	Small	Sprint 2
9	Defining Scope #17	Musti-007	Done	High	must have	Small	Sprint 2
10	Basic Requirements Gathering #18	Musti-007	Done	Medium	should have	Medium	Sprint 2
11	Initial Plan #19	Musti-007	Done	Low	could have	Medium	Sprint 2
12	Research Questions #14	Musti-007	Done	High	must have	Small	Sprint 3
13	User Registration Sequence Diagram #12	Musti-007	Done	Medium	should have	Large	Sprint 3
14	User Registration Activity Diagram #11	Musti-007	Done	Medium	should have	Medium	Sprint 3
15	Search for Prodcuts Sequence Diagram #13	Musti-007	Done	Medium	should have	Medium	Sprint 3

Figure 1: GitHub Project Board

At the end of each sprint, progress updates will also be sent to the project steering committee (PSC). In these updates, the tasks performed during the sprint will be briefed, problems or issues that emerge will be discussed, and an update on general progress will be provided, highlighting significant milestones and pending obstacles or deadlines. Therefore, the key points below will be considered:

- A detailed project plan and a clear and well-defined project scope are developed.
- Recognizing and mitigating risks during the project's timeline will be done.
- Quality assurance will be integrated into each phase of the project.
- Effective communication within the team and with stakeholders is established.
- A change management process will be in place to handle any changes to project scope, objectives, or requirements.
- A closure phase at the end will ensure that all project objectives have been accomplished.

3 Analysis

In the analysis phase, gathering and documenting what the app needs to do has been done properly, both in terms of basic functions and other important stuff. It was carefully figured out how expatriates will use the app in different situations. Detailed charts and tables were made to help visualize the big picture.

3.1 Functional Requirements

Functional requirements are a declaration of a structure and its mechanisms based on the requirements that a designer determines for a software or device functional behavior as expected for a wanted input.

3.1.1 Use Case Diagram

The use case diagram is the dynamic assortment of all steps and approaches of all user groups in one diagram (PlantUML, n.d.). It prohibits sets of actions, services, and functions of a system by using actors and use cases. Figure 2 depicts the use case diagram that shows use cases that are related to the Smart Shopper.

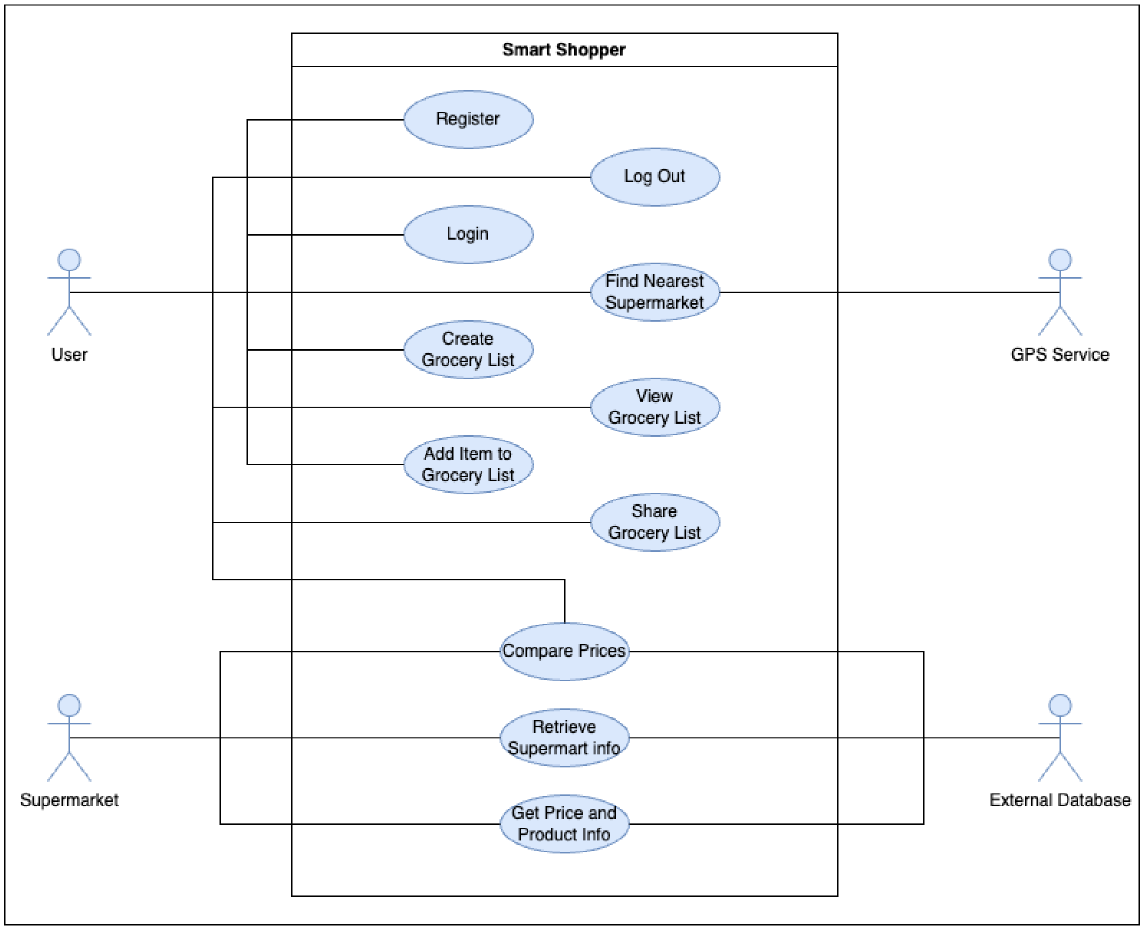


Figure 2: Use Case Diagram

3.1.2 Use Case Description

A Use Case Description is a scenario-written outline of stages of collaboration between an actor (a member of a user group) and the system (*What Is Use Case Diagram?* n.d.).

3.1.2.1 Example of Search for Products

ID	11
Name	Search for Products
Description	This use case allows users to search for specific grocery products within the application.
Actor	Shopper
Precondition	User is logged in.
Scenario	1 User opens the Smart Shopper app. 2 User enters a product name or keyword in the search bar. 3 The application displays a list of products matching the search query. 4 User selects a product from the list to view more details.
Exceptions	If no results match the search query, the application displays a message indicating that no products were found.
Extensions	None
Postconditions	User can view product details or proceed to add the product to their shopping list.

Table 4: Search for Products

3.1.2.2 Example of User Registration

ID	3
Name	User Registration
Description	This use case describes the process by which a new user registers for an account in the Smart Shopper application to access its features.
Actor	New User
Precondition	The user has downloaded and installed the Smart Shopper application. The user has not yet registered for an account.
Scenario	1 The user opens the Smart Shopper application. 2 The application presents the user with a registration screen. 3 The user enters their personal information, including their name, email address, and password. 4 The user confirms their password to ensure accuracy. 5 The user submits the registration form. 6 The application validates the user's inputs.

	7 If the inputs are valid, the application creates a new user account. 8 The application logs the user into their new account. 9 The user is redirected to the application's main screen.
Exceptions	**User Already Registered:** If the user's email address is already registered in the system, the application displays an error message, indicating that the email is already in use. **Invalid Inputs:** The application shows an error notice and asks the user to update any inaccurate or incomplete data that they enter.
Extensions	None
Postconditions	The user's registration information is saved in the system. The user is logged into their new account.

Table 5: User Registration

3.2 Activity Diagrams

Activity diagrams were created for the Smart Shopper project to illustrate how each use case's events progressed. Every activity diagram consists of a series of activities that represent the tasks completed in the use case and transitions that illustrate the transfer of control between activities. A flowchart and an activity diagram are similar in that they show the steps of a process and their timing.

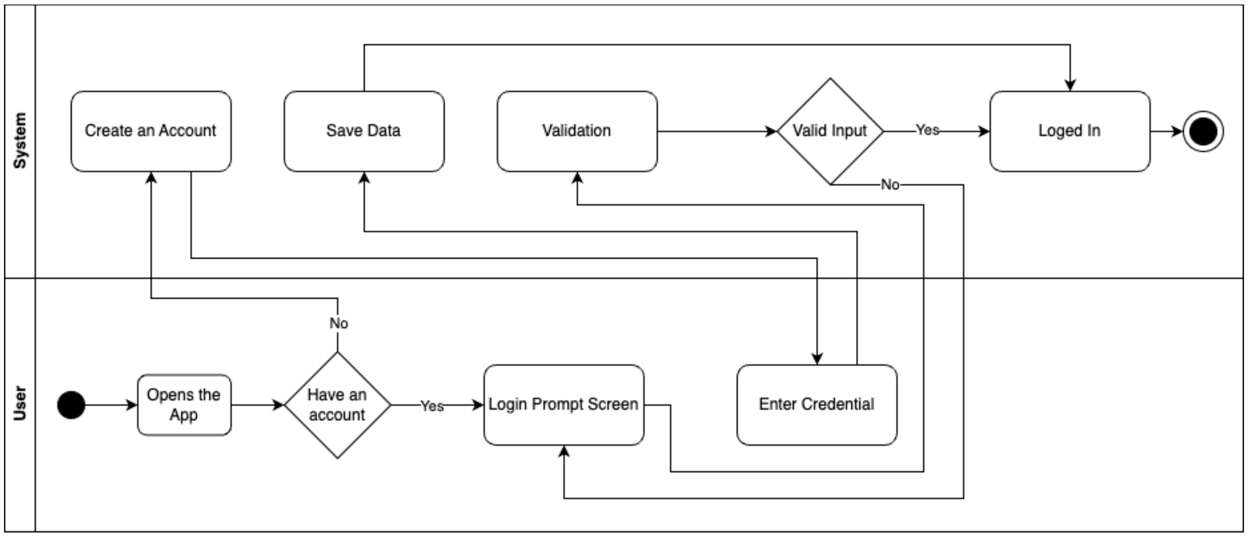


Figure 3: User Registration Activity Diagram

3.3 Sequence Diagrams

To model the interactions between the application components, sequence diagrams were used. A sequence diagram, for instance, was developed to show the progression of events that take place when a person registers. In Figure 4 shows how the controller, who then issued a query to the database, received a register request from the user interface component. The controller then transmitted the objects it had been queried from the database back to the user interface, where the user could see them.

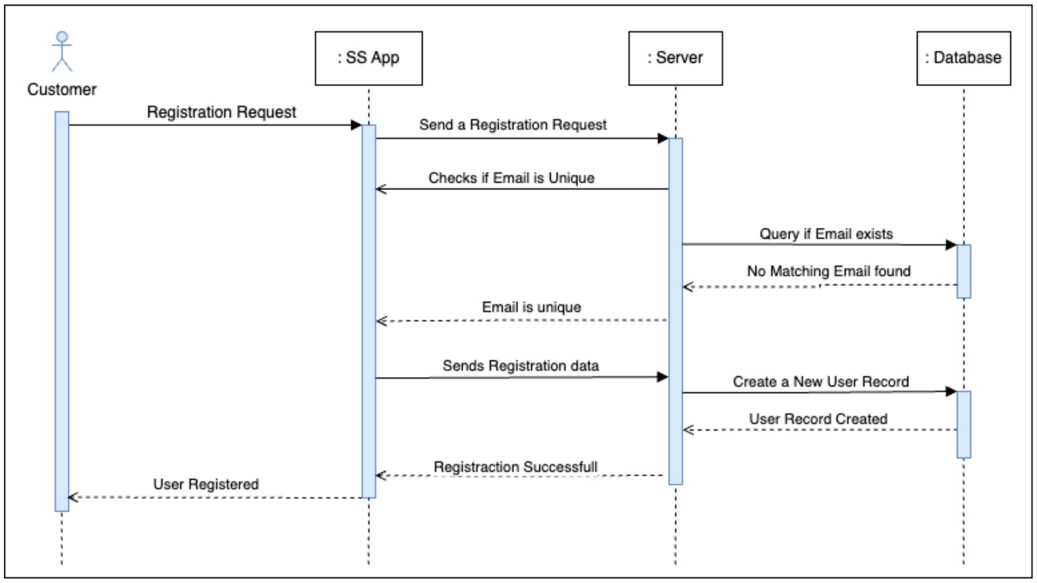


Figure 4: User Registration Sequence Diagram

3.4 Data Requirements

Data requirements are about the data used and produced throughout the procedure of achieving a goal by users.

3.4.1 Data Dictionary

A unified source of metadata or an all-term glossary for an application domain during the analysis phase. The main purpose of creating it is to understand the terms of analyzing it before starting to design data modeling. The following Table 6 has been used as its user data.
User Data:

Data:	**Type:**	**Description:**
User ID	Numeric	A unique identifier for each registered user.
Username	Alphanumeric	The chosen username of the user.
Email	String (Email Format)	The email address associated with the user's account.
Password	String (Hashed)	The user's password for account authentication.

Table 6: User Data Dictionary

3.4.2 Data Constraints

Data constraint explains DO and DON’T in the value assortment of attributes and entities while analyzing, constraints are verbally explained. The following Table 7 has been demonstrated as its example.

Data:	Description:	CONSTRAINS TYPE
User ID	A unique identifier for each registered user.	Numeric, auto-increment, unique.
Username	The chosen username of the user.	Alphanumeric, 4-20 characters, unique.
Email	The email address associated with the user's account.	Valid email format, unique.
Password	The user's password for account authentication.	At least 8 characters, a mix of upper and lower case letters, numbers, and special characters.
Grocery List ID	A unique identifier for each user's grocery list.	Numeric, auto-increment, unique.
Product Name	The name of a grocery product.	Alphanumeric, 2-50 characters.
Product ID	A unique identifier for each grocery product.	Numeric, auto-increment, unique.
Product Price	The price of a grocery product in euros.	Numeric, two decimal places.
Supermarket Name	The name of a supermarket or store.	Alphanumeric, 2-100 characters.
Supermarket Location	The physical location (address) of a supermarket.	Alphanumeric, up to 255 characters.
User Location	The user's current geographical location.	Latitude and longitude coordinates.
Product Image	An image representing the grocery product for a user-friendly UI.	Image file format (e.g., JPG, PNG).

Table 7: Data Constraints

3.5 Functional Requirements:

ID	Title	Priority	Description
FR-1	User Registration	High	• Users can create accounts with their personal information. • Registration requires a valid email address and password. • User profiles store preferences and shopping history.
FR-2	Product Search	High	• Users can search for grocery products by name or category. • The system provides real-time search suggestions. • Results include product names, prices, and store information.
FR-3	Price Comparison	High	• Users can compare prices of the same product across different supermarkets. • Price comparisons are displayed clearly for each product. • The system identifies the cheapest option and highlights it.
FR-4	Creating and Managing Lists	Medium	• Users can create, edit, and delete shopping lists. • Lists can be customized with product quantities. • Users can set budget limits for their lists.
FR-5	Store Locator	Medium	• Users can find nearby supermarkets on a map. • The system provides directions to selected stores. • Store details include address, hours, and contact information.
FR-6	Sharing Shopping Lists	Medium	• Users can share their created shopping lists with others. • Shared lists can be accessed and edited collaboratively by recipients.

Table 8: Functional Requirements

3.6 Non-Functional Requirements:

ID	Title	Priority	Description
NFR-1	Performance	High	• The application must provide real-time price comparisons, ensuring fast response times. • It should handle concurrent user requests efficiently, even during peak hours.
NFR-2	Usability	High	• The user interface must be intuitive and user-friendly. • Navigation should be straightforward, even for first-time users. • Search suggestions and filters should enhance the user experience.
NFR-3	Data Accuracy	High	• Price and product information must be accurate and up to date. • Store details and locations should be precise
NFR-4	Security	Medium	• User data, including personal information and lists, must be securely stored. • Secure login and data encryption are necessary.
NFR-5	Scalability	Medium	• The system should be scalable to accommodate future growth in users and data. • It should support additional supermarkets and products.
NFR-6	Compatibility	Medium	• The mobile app should be compatible with a range of Android and iOS devices. • Web compatibility should cover major browsers.
NFR-7	Availability	Low	• The application should be available 24/7, with minimal downtime for maintenance.
NFR-8	Accessibility	Low	• The app should comply with accessibility standards, ensuring usability for users with disabilities.
NFR-9	User Engagement	Low	• Features like user reviews and ratings aim to enhance user engagement and trust.

Table 9: Non-Functional Requirements

4 Design

The design phase is a plan or an overview of an object for its construction or implementing a process or an activity for all its stages. There are important steps to achieve a goal for better design and constraints, may consider visual, practical, socio-political, or financial considerations, and might combine certain settings.

4.1 Functional Design

Functional design is used as a model to simplify hardware and software device designs like 3D modeling and software models. Each modular also assures the functionality assigned to the part to decrease the side effect of its responsibility to the minimum level as it can (*Functional Design Specification*, 2018).

4.1.1 Window Design

Window design is the process of the appearance of a web page and selecting right color scheme, proper layout with appropriate font with correct text size. Every window has its own font, color, layout, and design, but in general, all pages should have a similar graphic design. The following Figure 5 demonstrates as example.

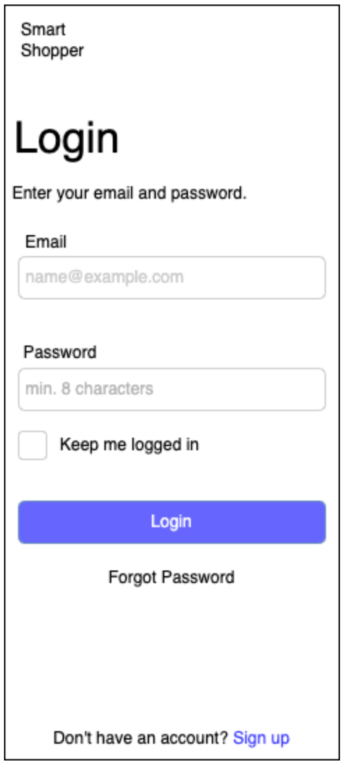


Figure 5: Login Wireframe

4.1.2 Navigation Diagram

Navigation diagram main purpose is to demonstrate the navigation on a home page to the other pages as shown in the Figure 6, whereas pages are shown as individual window or page, with the page name above. Arrows have been used to point out the navigation from one page to another and shows navigation between all pages.

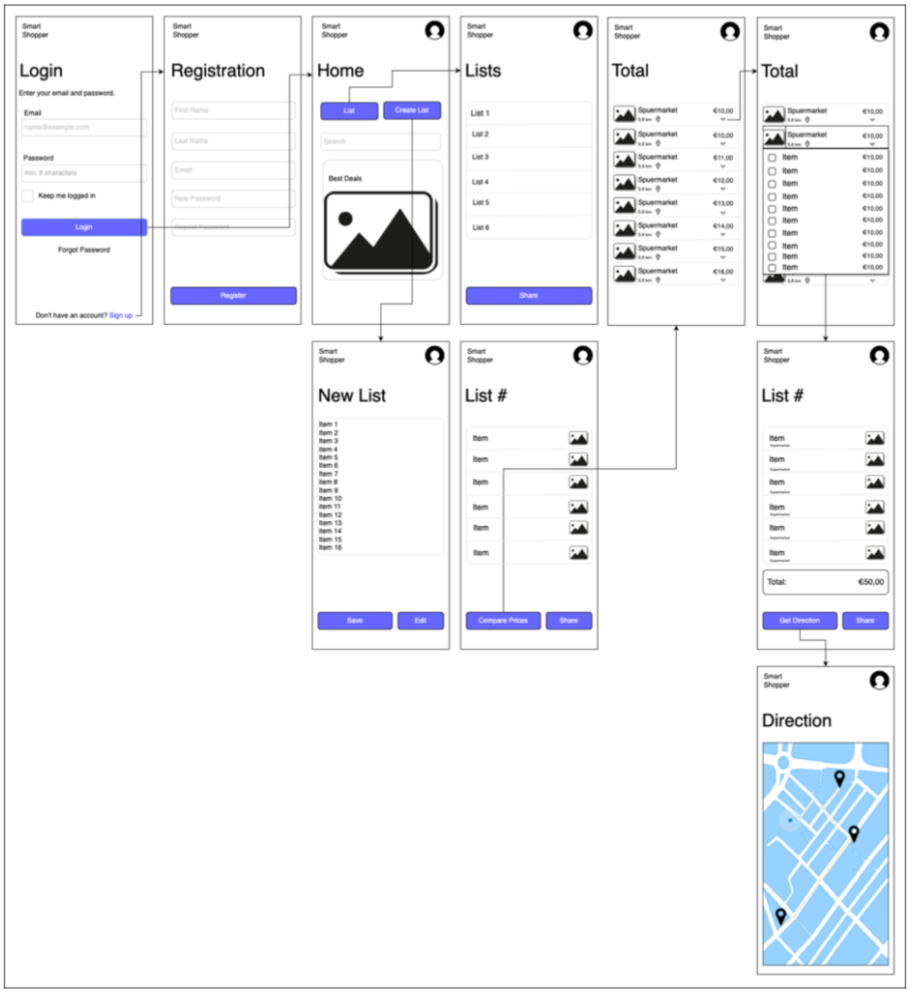


Figure 6: Navigation Diagram

4.2 Visual Design

Visual design is the use of imager font, color, layout, design, typography, and shape forms to increase and improve the usability of experiences[2]. Fields like UI design and graphic design have grown out instantly[2].

[2] (Cardello, 2023)

4.2.1 Moodboard

A Moodboard is an arrangement of images or creative composition of pictures, material pieces of text elements intended concept to present an atmosphere[2]. A Moodboard is also used to clarify equivocal, which are hard to explain ideas or generate ideas for the designer and the collation customer. As an example, below Figure 7 represents the combination of images, color, and font used together for the windows design.

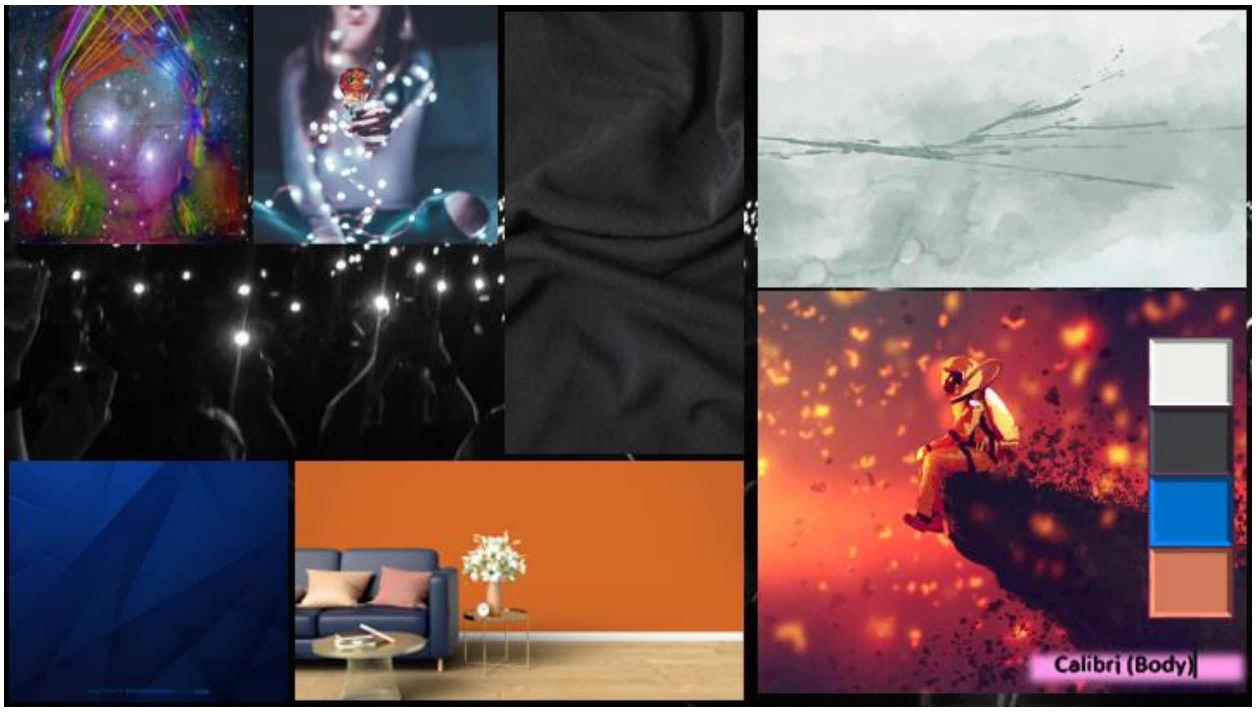


Figure 7: Moodboard

4.3 Technical Design

Technical design is a solution to the activities and its problem while developed designed has been completed. A designer co-ordinates technical design before even actual design begins (*Technical Design Stage for Building Projects*, n.d.).

4.3.1 Relational Model

The Relational Model is a way of handling the data and it structure consistency, where all data is characterized as group into relation in terms of tuples. In the below example Figure 8: Relational Model has been illustrated with its primary keys and foreign key.

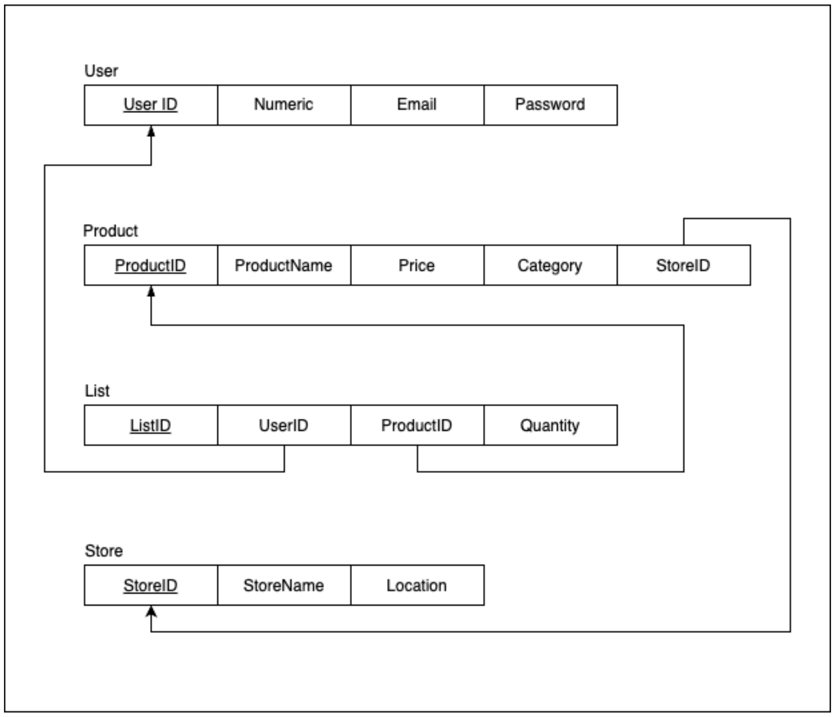


Figure 8: Relational Model

4.3.2 Database Schema

A database schema is a blueprint of a structure and design and its constraint and relation. The very core stage of creating any software or webpage. Below Table 10 depict how it were generated.

4.3.2.1 Users Table:

Key	**Description**
UserID (Primary Key):	A unique identifier for each user.
Username:	The user's chosen username.
Email:	The user's email address.
Password:	An encrypted version of the user's password.
Other user-related attributes, such as name, address, etc.	

Table 10: User Database Schema

5 API's Research

5.1 Introduction

This chapter studies several API choices and other alternative methods while assessing their applicability to the project. Based on the research findings, recommendations on the best method(s) to utilize are made after discussing the benefits and drawbacks of each strategy.

5.2 Literature Review

To start the project effectively, the available APIs and alternative data sources for Dutch grocery stores were thoroughly observed. These APIs offer a more reliable and flexible way to gather product information compared to web scraping. The primary goal was to find the best data sources for products, deals, and prices at major Dutch supermarkets. This information will be a vital part of the Smart Shopper app, offering expats accurate and up-to-date details.

In the initial project phase, we conducted extensive research on supermarket APIs to evaluate their suitability for our goals. We explored various API options and web scraping alternatives, ensuring they fit the project's needs. This chapter presents these options, discusses their pros, and cons, and offers recommendations for the best approach to use in the Smart Shopper app.

5.3 Research Question

The primary goal of the API research was to determine whether major supermarkets provide APIs or data feeds that can be integrated into the Smart Shopper mobile application. These APIs would serve as a crucial source of real-time pricing data and product information.

To what extent can integrating external grocery store data APIs into the Smart Shopper mobile application enhance its functionality and user experience, leading to an optimized grocery shopping experience for users?

This question explores how integrating external APIs, a core project element, enhances app functionality in alignment with project goals. It also explores the impact on shoppers' efficiency, a key Smart Shopper program benefit.

5.4 APIs

The below APIs are helpful in shaping the project's direction and determining the technical architecture for retrieving real-time supermarket data, a critical component of the Smart Shopper application. There are many other APIs and data sources available here are a few to consider for the Netherlands markets:

- Checkjebon API
- ShoppingScraper API
- Superscanner API
- Websrapping.Amsterdam API

In assessing the mentioned APIs, the subsequent standards were considered: Which format or formats allow for the exploration of data? Is there a broad range of products and retailers covered by the API's data? What is the price of utilizing the API?

5.4.1 Checkjebon API

Checkjebon API is designed for price comparison and product information retrieval. It allows developers to access and use data related to various products from different sources[3]. It’s an open-source project the data can be used freely[3]. The API contains the data in JSON format. It is free to use and other pricing details for API usage are available on the website. It offers different subscription tiers though[3].

The API covers a range of products, including grocery items from various supermarkets and some of them are as follow[3]: Albert Heijn, Aldi, Coop, DekaMarkt, Dirk, Hoogvliet, Jan Linders, Jumbo, Plus, SPAR, and Vomar.

Advantages	Disadvantages
• Free to use[3]. • Designed for price comparison and product information retrieval[3]. • Provides data in JSON format[3]. • Regular updates based on changes in source websites[3]. • Offers different subscription tiers. • Secured access using an API key. • Covers a range of products, including grocery items[3].	• Data accuracy can vary based on the accuracy of data from source websites[3]. • Documentation may not be as comprehensive as some other APIs. • Limited user reviews and a smaller user community[3].

Table 11: Checkjebon API

5.4.2 Shopping Scraper API

Shopping Scraper API is a web scraping service designed for retrieving product info, product images, and price information from various online retailers. It allows developers to access and integrate scraped data into their applications. The API provide the data in JSON format. The API does not specifically mention a list of NL grocery stores. It appears to focus on online retailers in general. But it contains data for below NL supermarkets: Alber Heijn, Jumbo.

Pricing details are available on the website as below. It offers a free tier with limited usage.

• Essential:	€129/mo
• Growth:	€199/mo
• Advanced:	€399/mo
• Enterprise:	€749+/mo

Advantages	Disadvantages
• Provides access to scraped product and price data from various online retailers. • Offers data in JSON format for easy integration into applications. • Regularly updates data based on changes in source websites. • Offers a free tier with limited usage, making it cost-effective for small projects.	• Accuracy of scraped data may vary depending on source websites. • Basic documentation may require additional effort for integration. • Smaller user community and limited user reviews. • Limited coverage and focus on specific types of products.

[3] (Supermarkt, n.d.)

• Secured access using an API key. • Developer support via email.	

Table 12: ShoppingScraper API

5.4.3 Superscanner API

Superscanner API is a price comparison service that allows users to compare prices of various products across different online stores. It provides developers with access to its data. The API provide the data in JSON format. Pricing details are not available on their website. While the API focuses on various products, it does include a category for grocery products of the supermarkets as follows: Albert Heijn, Jumbo, Makro, Dirk, Nettorama, PicNic, Plus, Spar, and Hoogvliet.

Advantages	**Disadvantages**
• Allows users to compare prices of various products across different online stores. • Data provided in JSON format for easy integration. • Regular updates based on data from online stores. • Offers a free tier with rate limits. • Access is secured using an API key. • Covers a range of product categories.	• Data accuracy may vary based on the availability and accuracy of data from online stores. • Basic documentation. • Limited user reviews and a relatively smaller user base. • Documentation may not be as comprehensive as some other APIs.

Table 13: Superscanner API

5.4.4 Websrapping.Amsterdam API

Websrapping.Amsterdam offers web scraping services for various types of data, including product and price information. It allows developers to access and retrieve data from websites (*Bekijk Alle Datasets – Webscraping Amsterdam*, n.d.). it provides the data in JSON format. Pricing details for API usage are available on the website, including different pricing tiers.

The API's focus is not specifically mentioned, but it appears to cover a broader range of websites rather than being limited to grocery stores and they are as follow: Albert Heijn, Jumbo, Coop, Lidl, Plus, Ekoplaza, and Dirk.

Advantages	**Disadvantages**
• Offers web scraping services for various types of data. • Provides data in JSON format. • Regular updates based on changes in source websites. • Offers different pricing tiers. • Secured access using an API key.	• Focus may not be specific to grocery stores; covers a broader range of websites. • Data accuracy may vary based on the source websites. • Limited user reviews and a smaller user community.

• Developer documentation provided.	• Documentation may require additional effort for integration.

Table 14: Websrapping.Amsterdam API

5.5 Methodology

The API research provided the following key findings:

- **Availability**: Several major supermarket chains, including Albert Heijn, Jumbo, Makro, and Dirk offer APIs or data feeds for accessing pricing and product information.
- **Documentation**: API documentation for Albert Heijn, Jumbo, Makro, and Dirk are available, providing details on endpoints, data formats, and usage guidelines.
- **Technical Feasibility:** Initial assessments indicate that integrating these APIs into the Smart Shopper mobile application is technically feasible.

The API research was conducted systematically and included the following steps:

- **Online Research:** We began by determining the data requirements for the Smart Shopper application, focusing on obtaining real-time pricing and promotion data from Dutch supermarkets.
- **API Documentation Review:** When potential APIs were identified, we meticulously reviewed their documentation to evaluate their suitability for our project's objectives.
- **Technical Feasibility:** The technical feasibility was assessed of integrating the identified APIs into the mobile application. Factors such as data accessibility, accuracy, integration complexity, cost, and support were considered.

5.6 API Result

The Checkjebon API has been found to be the most appropriate API for the project based on the evaluation in the API sections and client preferences. It offers information about the goods, costs, and special offers at the largest supermarkets in the Netherlands, and it covers a wide range of establishments and goods. The API is also rather simple to use and works nicely with the application. Utilizing the API comes at a fair price that is within the project's budget (Free).

The constraints of the other APIs considered made them less suitable for the purpose. For instance, both the Webscraping and the ShoppingScraper API only provided a very limited amount of coverage for Dutch grocery stores. Amsterdam API, which was also the most expensive API, offered data in a format that was not acceptable for the application. We also explored the possibility of web scraping if the available APIs did not meet our criteria.

5.7 Conclusion

The Checkjebon API is the preferred choice over web scraping, according to APIs research done on data-collecting techniques for the Smart Shopper application. This research’s primary goal was to determine the best approach for gathering product data from at least important significant Dutch grocery stores in order to support the Smart Shopper application. After some research, it was discovered that web scraping provides more customization and flexibility, In conclusion, considering the effort required to collect data from various sources and the time allotted for the graduation internship, the Checkjebon API was chosen over web scraping. This methodology facilitates the development of a sturdy and efficient Smart Shopper application and offers avenues for more investigation into data collection techniques and their efficacy for certain uses.

6 Implementation

This chapter discusses the present state of the Smart Shopper application's implementation, including the programming languages used, the development frameworks selected, and other technical details.

6.1 Architecture

For the Smart Shopper app, the team's familiarity with these architectures, and expectations for future growth are very important[4]. There are few of them that relate to the Smart shopper are below:

1. MVC (Model-View-Controller)[4]:
   - Model: Represents the data and business logic[4].
   - View: Displays the user interface.
   - Controller: Handles user input and updates the Model and View accordingly[4].
2. MVP (Model-View-Presenter)[4]:
   - Similar to MVC, but Presenter handles user input and mediates between Model and View[4].
3. MVVM (Model-View-ViewModel)[4]:
   - Model: Represents the data.
   - View: Displays the user interface.
   - ViewModel: Acts as an intermediary between Model and View, preparing data for the View[4].
4. VIPER (View-Interactor-Presenter-Entity-Routing)[4]:
   - A more modular architecture that separates components for better testability and scalability[4].
   - View: User interface.
   - Interactor: Business logic and data operations[4].
   - Presenter: Mediates between View and Interactor.
   - Entity: Data model.
   - Routing: Handles navigation.

MVVM (Model-View-ViewModel) architecture is a well suited for the Smart Shopper project due to its clear separation of concerns, modularity, testability, reusability, and support from a thriving development community. This architecture provides the necessary structure and flexibility for managing complex user interfaces and business logic (Michaelstonis, 2022).

6.2 Stack

6.2.1 Mobile Frontend:

6.2.1.1 React Native:

If you would like a cross-platform approach with a single codebase, React Native is a well-liked option. It allows you to build native-like mobile apps using JavaScript and React[6]. Comparing React Native with

[4] (Triare Ukraine, 2023):

other frameworks like Flutter or Xamarin, React Native stands out as the best choice for the Smart Shopper project due to its broader community support, extensive library of pre-built components, and compatibility across both iOS and Android, which optimizes development time and costs[5].

Key Takeaways:

- React Native is an open-source framework for mobile app development[6].
- It provides a native look and feel for iOS and Android platforms[6].
- It is easy for web developers to transition to mobile app development with React Native[6].
- React Native offers faster development times and reduced costs[6].
- It delivers better performance and stability than other cross-platform frameworks[6].
- It has extensive support from the community with many libraries and tools available[6].

6.2.2 Backend Development:

6.2.2.1 JavaScript:

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard[7], that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2023, 98.7% of websites use JavaScript on the client side for webpage behavior[8], often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript, in contrast to languages like C++ and Python, is well-suited for web and mobile app development due to its unique ability to run directly in web browsers[9]. This feature enhances user experience, providing a more responsive and interactive interface, making it a preferred choice for Smart Shopper[9].

6.2.2.2 Node.js

The Smart Shopper application's backend is built using Node.js, which offers a strong and effective runtime environment for handling real-time operations (Khare, 2023).

6.2.2.3 API Integration:

Express.js: For building RESTful APIs that connect the app to external data sources like Checkjebon API.

Pros:

- Wider adoption and more resources are available[10].
- Easier for developers already familiar with JavaScript[10].
- Quicker prototyping and development due to its dynamic nature[10].

[5] (Cooper, 2023)
[6] (Kumar,2023b)
[7] (ECMAScript® 2020 Language Specification)
[8] (Usage Statistics of JavaScript as Client-side Programming Language on Websites, July 2023)
[9] (Wordsmith, 2023)
[10] (Team et al., 2023)

- The company uses JavaScript[10].

Cons:

- Less strict type checking, which can lead to runtime errors[10].
- Slower[10].

6.2.3 Database:

6.2.3.1 SQLite

is a database engine written in the C programming language. It is not a standalone app; rather, it is a library that software developers embed in their apps[11]. As such, it belongs to the family of embedded databases[11]. It is the most widely deployed database engine, as it is used by several of the top web browsers, operating systems, mobile phones, and other embedded systems[11].

Many programming languages have bindings to the SQLite library. It generally follows PostgreSQL syntax but does not enforce type-checking by default[12].

- It's a Non-functional database.
- A lightweight, embedded database that's suitable for mobile app development. Both Android and iOS support SQLite[12].
- SQLite, when compared to traditional relational databases like MySQL and PostgreSQL, offers a lightweight and serverless alternative[12].
- Unlike MySQL and PostgreSQL, which require dedicated servers and complex setup, SQLite is embedded directly into the application, reducing deployment complexities. It's ideal for mobile applications like Smart Shopper, which need to efficiently manage data without a constant internet connection[12].
- However, SQLite may not be suitable for scenarios demanding high concurrency, large-scale data management, or extensive server-side processing, areas where MySQL and PostgreSQL excel. Its simplicity, offline capabilities, and ease of integration make it an excellent choice for the Smart Shopper mobile app's needs[12].

6.2.4 Version Control:

6.2.4.1 Git:

- Use Git for version control to track changes in your codebase and collaborate with the development team effectively[13]
- GitHub (Project Board)

The choice of Stack or its frameworks depends on various factors, including the project's specific requirements, team expertise, and development preferences. Each framework offers unique advantages and is suited to different use cases. The above stack is suitable based on the requirements and personal experience with the stack.

[11] (Most Widely Deployed SQL Database Estimates)
[12] (Owens, Michael, 2006)
[13] (Chacon, Scott; Straub, Ben, 2014)

7 Conclusion

To conclude during the internship, a noteworthy development on the Smart Shopper app was made. It all started with requirement-gathering phase. Here, important features are identified in what the app should do, both in terms of basic functionality and other important aspects. Fancy tools like use case diagrams to show the different parts of the app and how they all fit together are used.

Research on APIs was performed. Mainly, to identify best way of getting data from the grocery stores. After some research, using the Checkjebon API was the reasonable choice. It fits well with what the app needs to do, and it matches the tight schedule of the project.

When the actual implementation of the app started, there were some challenges in getting the search functionality and store locator feature to work seamlessly. These challenges were tackled by teaming up with colleagues and getting some advice from the client.

The Smart Shopper app implementation is the focus for the future. There are other exciting features to work on, including lists, search keywords, routing, supermarket location, and account setup. These features will help expatriates stay organized and find more cheap deals based on their preferences. Regular client meeting will be needed to make sure that the development of these features is in line with the client's needs and expectations.

8 Reference

Amazon Web Services, Inc. Retrieved April 9, 2021, from https://aws.amazon.com/amazon-mq/?amazon-mq.sort- by=item.additionalFields.postDateTime&amazon-mq.sort-order=desc

Analytics India Magazine. https://analyticsindiamag.com/9-best-kubernetes- alternatives-for-devops-engineers/

Bekijk alle Datasets – Webscraping Amsterdam. (n.d.). https://webscraping.amsterdam/data-producten/

Cardello, J. (2023, October 22). *The web design process: creating the visual design*. Webflow. https://webflow.com/blog/the-web-design-process-creating-the-visual-design?utm_source=google&utm_medium=search&utm_campaign=SS-GoogleSearch-Nonbrand-DynamicSearchAds-Global&utm_term=aud-936979375361:dsa-1537217525485___617245336969__&gclid=CjwKCAjws9ipBhB1EiwAccEi1DD-Wf9lkxyQbhit3Gt2HmEH23QilfKha1f_RWqTcE9Ou4k8HblfdRoCUSYQAvD_BwE

Chacon, Scott; Straub, Ben (2014). "Git on the Server – Setting Up the Server". Pro Git (2nd ed.). Apress. ISBN 978-1484200773.

Company overview. (n.d.). CGI. https://www.cgi.com/en/overview

Cooper, A. (2023, October 6). Detailed Comparison Between flutter, react native & Xamarin. *ValueCoders | Unlocking the Power of Technology: Discover the Latest Insights and Trends*. https://www.valuecoders.com/blog/technology-and-apps/flutter-vs-xamarin-vs-react-native-which-cross-platform-mobile-app-development-framework-to-choose/#:~:text=Flutter%2C%20React%20Native%2C%20and%20Xamarin%20Frameworks%20are%20competent%20at%20developing,terms%20of%20seamless%20user%20experience.

ECMAScript® 2020 Language Specification. Archived from the original on 2020-05-08. Retrieved 2020-05-08.

Functional Design specification. (2018, March 27). Scott Manning. https://scottmanning.com/content/functional-design-specification/

Khare, M. (2023, October 18). *What is Node.Js and why you should use it*. Kinsta®. https://kinsta.com/knowledgebase/what-is-node-js/#:~:text=In%20a%20nutshell%2C%20Node.,and%20data%20streaming%20applications%2C%20too.

Kumar, A. (2023, September 25). Why React Native is Better than Other Similar Platforms? *YourTimeIndia*. https://www.yourteaminindia.com/blog/why-react-native-is-better-than-other-platforms#:~:text=and%20Android%20platforms.-,It%20is%20easy%20for%20web%20developers%20to%20transition%20to%20mobile,many%20libraries%20and%20tools%20available.

Michaelstonis. (2022, November 4). *Model-View-ViewModel - .NET*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

Most Widely Deployed SQL Database Estimates". SQLite.org. Retrieved May 11, 2011.

Owens, Michael (2006). "Chapter 4: SQL". In Gilmore, Jason; Thomas, Keir (eds.). The 8Definitive Guide to SQLite. D. Richard Hipp (foreword), Preston Hagar (technical reviewer). Apress. p. 133. ISBN 978-1-59059-673-9. Retrieved 30 December 2014.

PlantUML. (n.d.). *Use case Diagram syntax and features*. PlantUML.com. https://plantuml.com/use-case-diagram

Supermarkt. (n.d.). *GitHub - supermarkt/checkjebon: Dutch supermarket prices and comparison tool*. GitHub. https://github.com/supermarkt/checkjebon

Team, D., Team, D., & Team, D. (2023, May 11). *Advantages and Disadvantages of ExpressJS*. DataFlair. https://data-flair.training/blogs/expressjs-advantages-and-disadvantages/

Technical design stage for building projects. (n.d.). Designing Buildings. https://www.designingbuildings.co.uk/wiki/Technical_design_stage_for_building_projects

Triare Ukraine. (2023, June 13). *Architecture for Mobile Apps: MVC vs MVP vs MVVM vs Viper*. TRIARE. https://triare.net/insights/common-architecture-for-mobile-application/

Usage Statistics of JavaScript as Client-side Programming Language on Websites, July 2023. w3techs.com. Retrieved 2023-07-02.

What is Use Case Diagram? (n.d.). https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

Wikipedia contributors. (2023, September 15). *Users' group*. Wikipedia. https://en.wikipedia.org/wiki/Users%27_group

Wordsmith, K. (2023, January 11). Decoding the secrets of top programming languages: JavaScript, Python, Java, C#, and C++. Medium. https://medium.com/@keyboardwordsmith/decoding-the-secrets-of-top-programming-languages-javascript-python-java-c-and-c-8b184f966bb7#:~:text=JavaScript%20is%20a%20good%20choice,such%20as%20system%20programming%20and

9 Appendix

Data Dictionary

Grocery Data		
Data:	Type:	Description:
Grocery List ID	Numeric	A unique identifier for each user's grocery list.
Product Name	Alphanumeric	The name of a grocery product.
Product ID	Numeric	A unique identifier for each grocery product.
Product Price	Decimal	The price of a grocery product in euros.

Table 15: Grocery Data Dictionary

Supermarket Data		
Data:	Type:	Description:
Supermarket Name	Alphanumeric	The name of a supermarket or store.
Supermarket location	Alphanumeric	The physical location (address) of a supermarket.

Table 16: Supermarket Data Dictionary

Location Data		
Data:	Type:	Description:
Location Data	Geographic Coordinates (Latitude, Longitude)	The user's current geographical location.

Table 17: Location Data Dictionary

Product Image Data		
Data:	Type:	Description:
Product Image	File (Image)	An image representing the grocery product for a user-friendly UI.

Table 18: Product Image Data Dictionary

Database Schema

Products Table:

Key	Description
ProductID (Primary Key):	A unique identifier for each product.
ProductName:	The name of the product.
Price:	The price of the product.
Category:	The product's category (e.g., groceries, electronics, clothing).
StoreID:	A foreign key that links to the store where the product is available.
Other product-related attributes, including brand, description, etc.	

Table 19: Products Database Schema

Stores Table:

Key	Description
StoreID (Primary Key):	A unique identifier for each store.
StoreName:	The name of the store.
Location:	The location or address of the store.
Other store-related attributes, such as contact information, working hours, etc.	

Table 20: Stores Database Schema

List Table:

Key	Description
ListID (Primary Key):	A unique identifier for each wishlist.
UserID:	A foreign key that links to the user who created the wishlist.
ProductID:	A foreign key that links to the products added to the wishlist.
P:	The quantity of each product in the wishlist.
Other wishlist-related attributes, such as the date created and the wishlist name, etc.	

Table 21: List Database Schema

Relationships:

- Users have a one-to-many relationship with wishlists, allowing a user to create multiple wishlists.
- Products have a many-to-many relationship with stores, as a product can be available in multiple stores, and a store can offer multiple products.
- Users also have a one-to-many relationship with wishlists through the UserID field.

Foreign Keys:

- The UserID in the Wishlists table links each wishlist to a specific user.
- The StoreID in the Products table connects each product to a particular store.

Wireframes

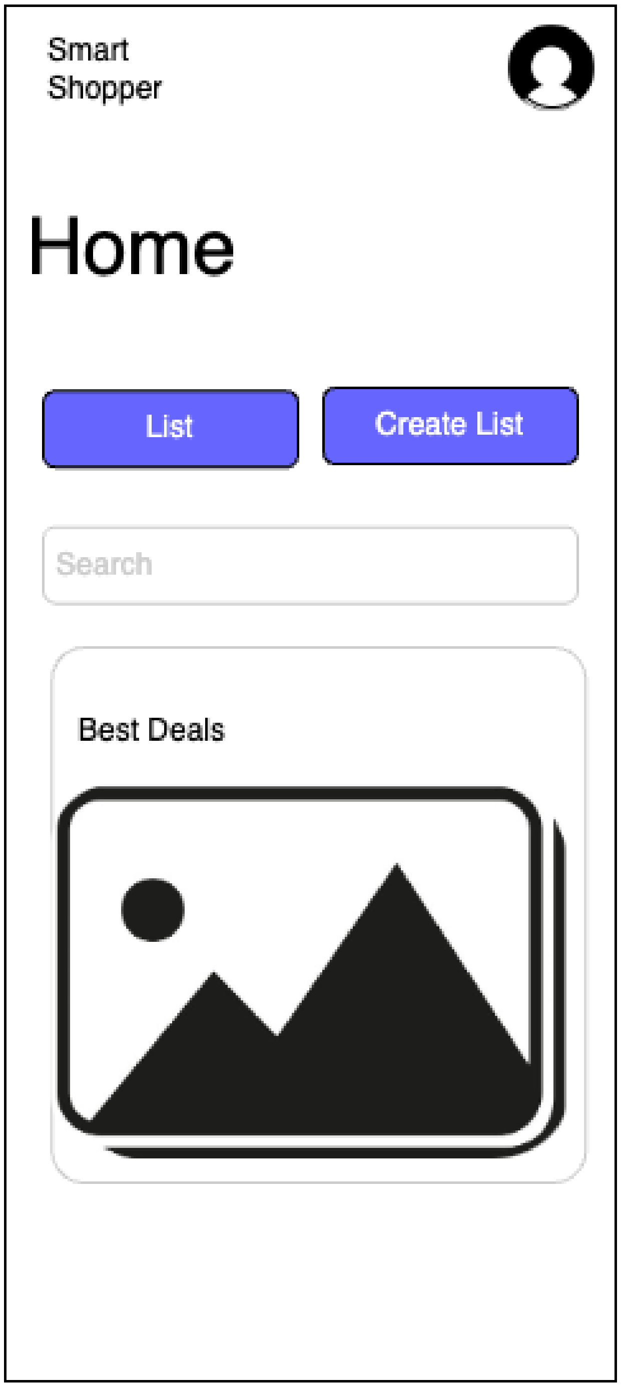


Figure 9: Home Wireframe

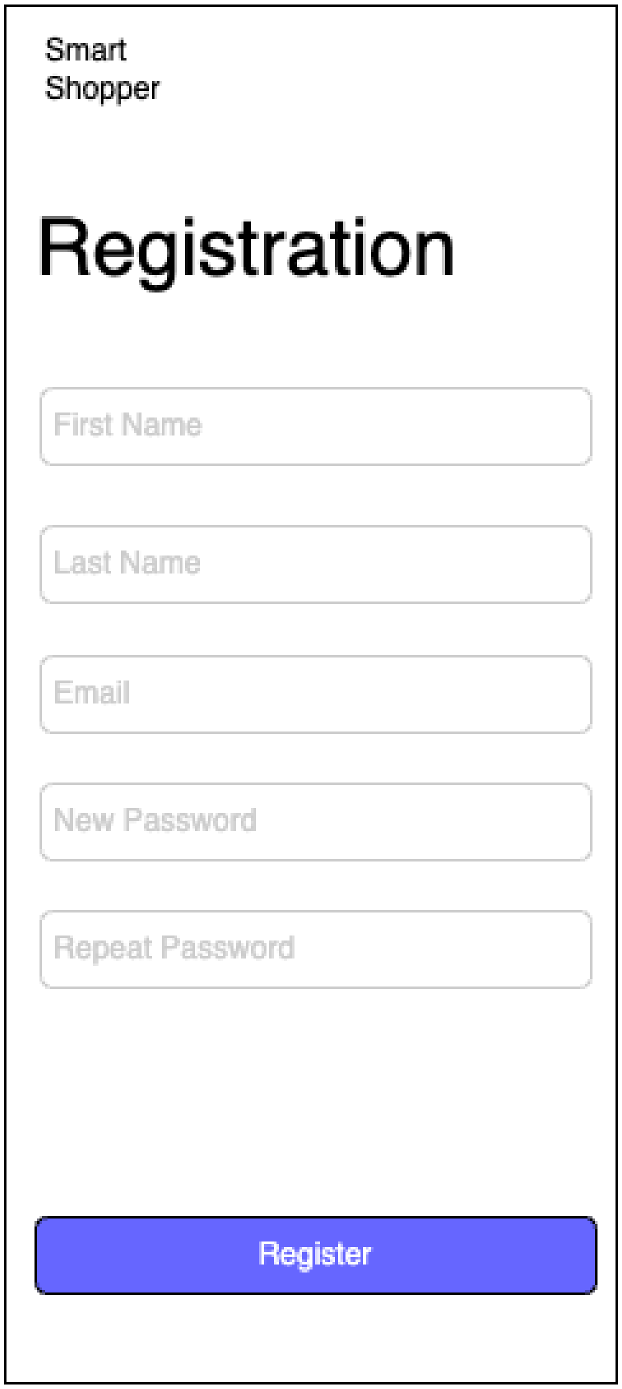


Figure 10: Register Wireframe

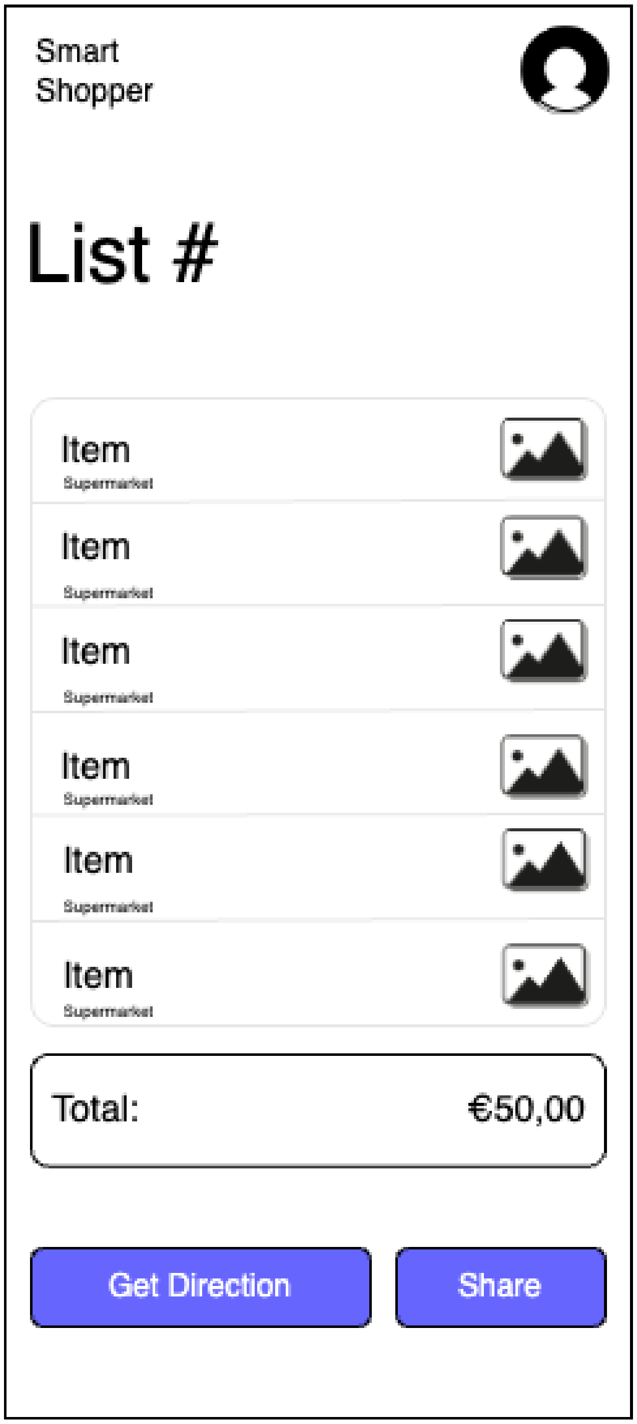


Figure 11: List Wireframe

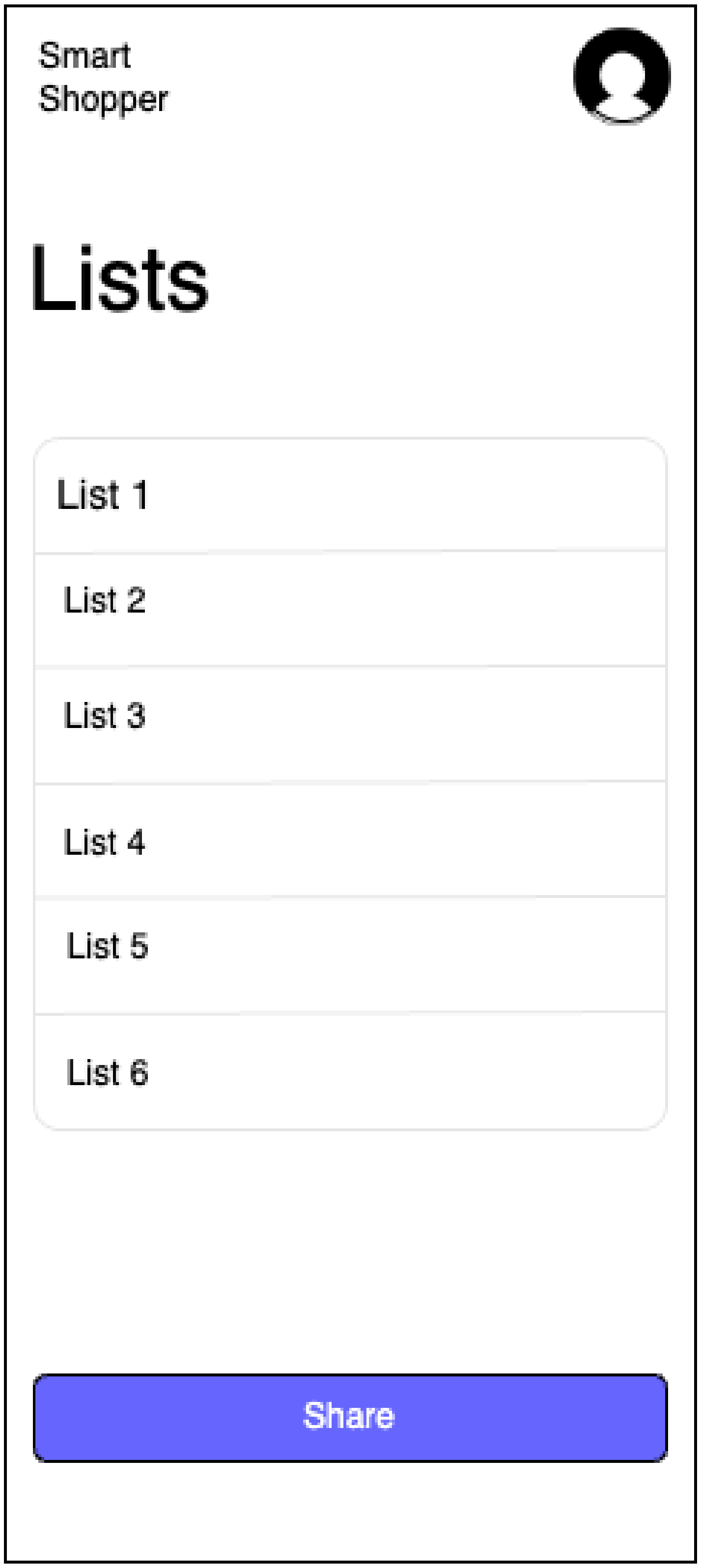


Figure 12: All Lists Wireframe

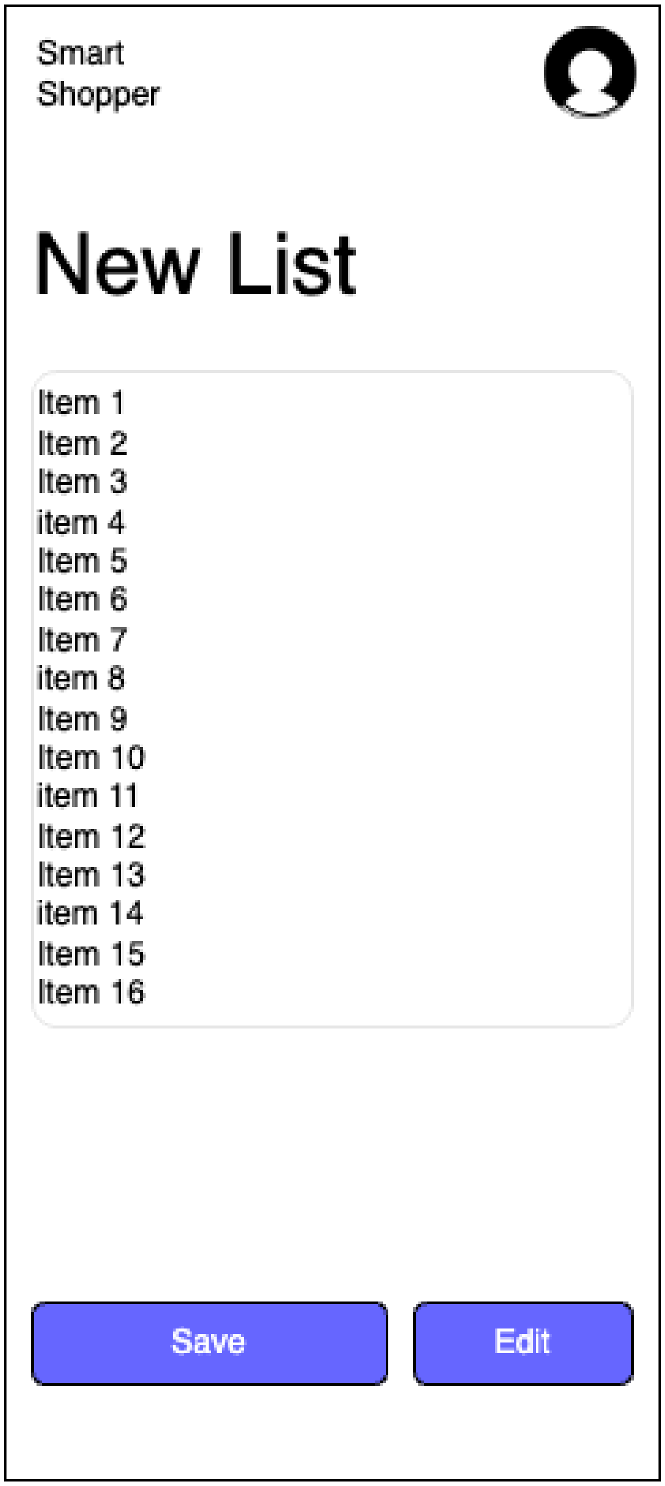


Figure 13: Create New List Wireframe

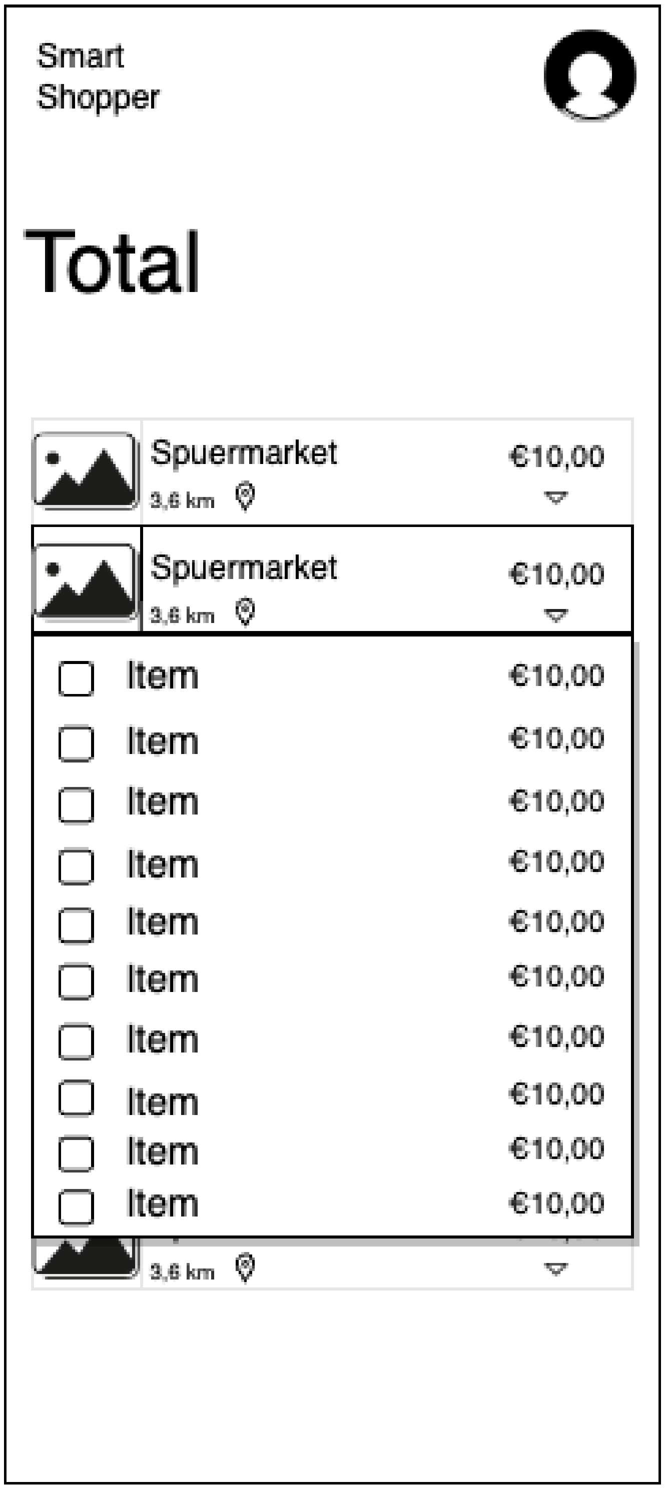


Figure 14: Supermarket Based Dropdown List

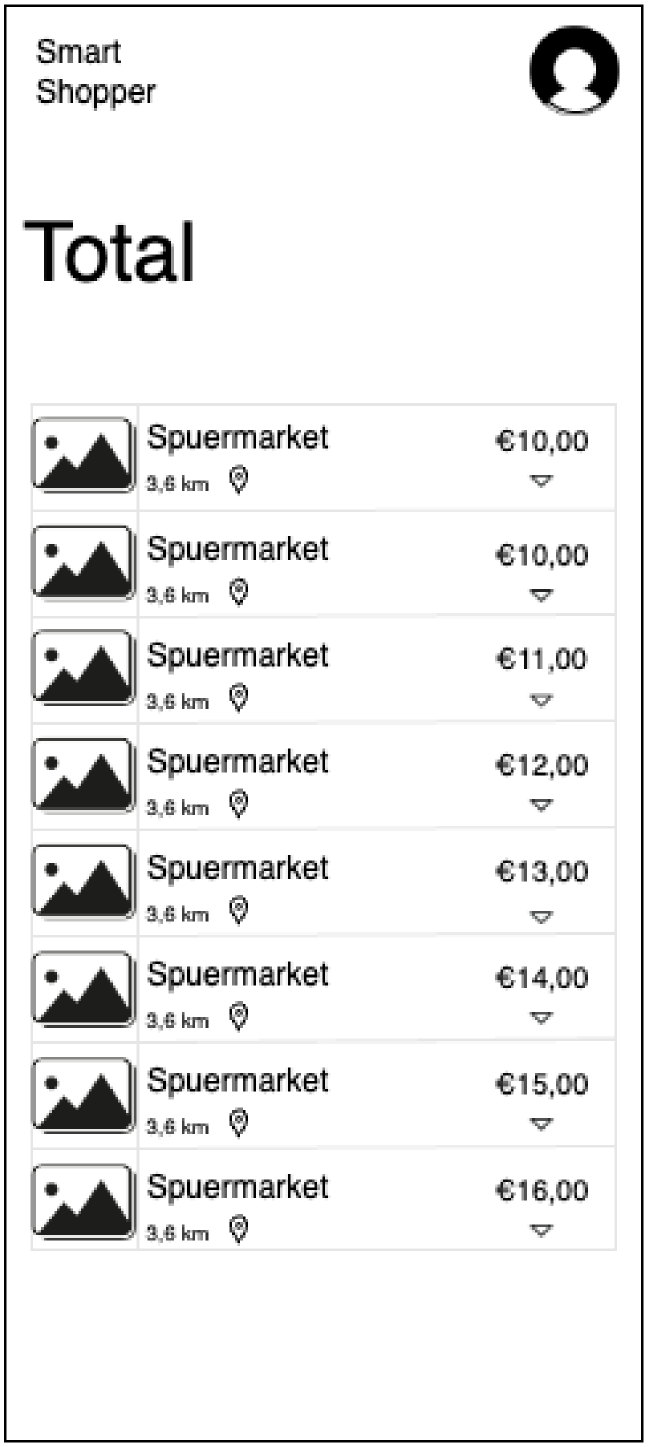


Figure 15: Same Supermarket Total

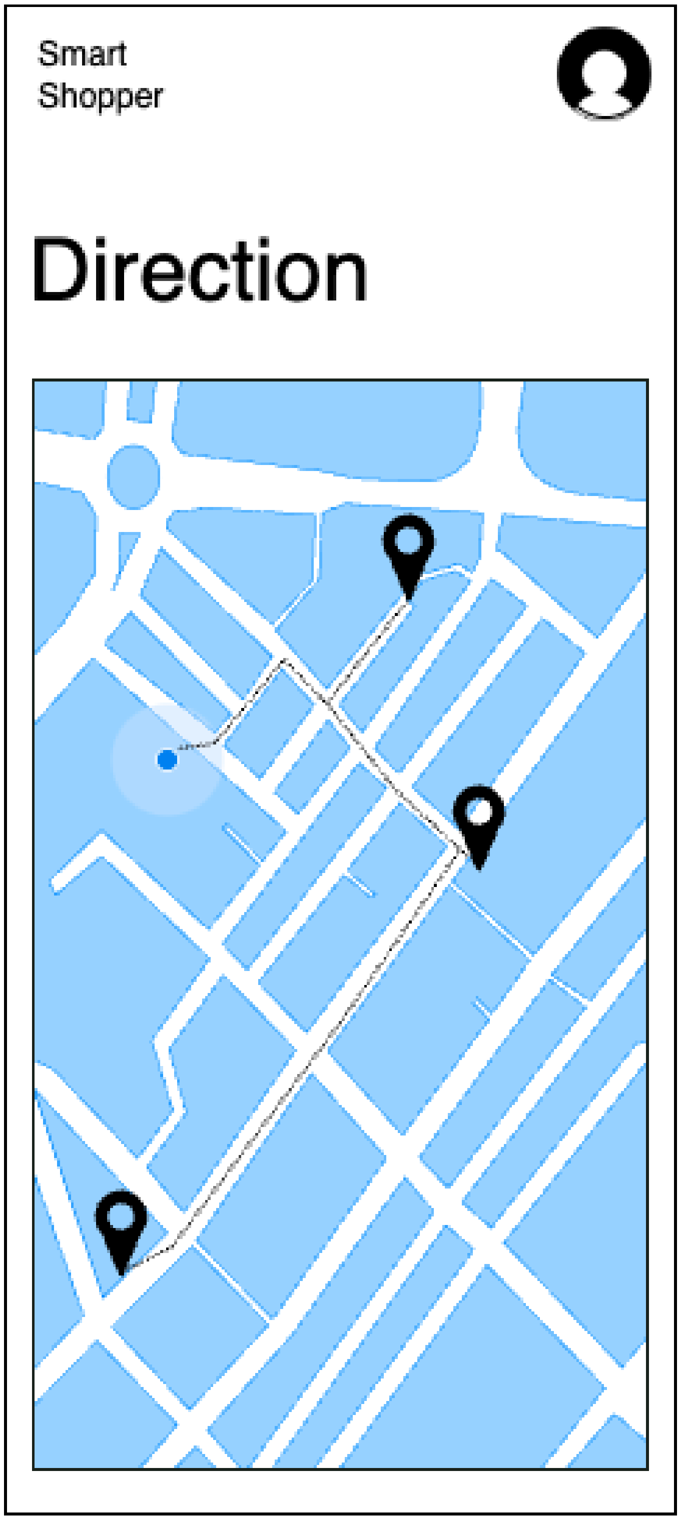


Figure 16: Stores Routing

Reflection

I learned a lot about software development and practical work throughout the Smart Shopper project, looking back on it all. My academic skills could now be used in a real-world situation thanks to this internship. Gathering project requirements, a crucial stage in the development process, was the main emphasis of the first few weeks of the internship. Defining the functional and non-functional needs throughout this phase underscored the significance of good communication and teamwork with the client. I developed my ability to evaluate client demands, pose pertinent questions, and recognize obstacles and constraints.

The project moved on to the exciting and difficult analysis and design phase after the requirements were established. After visualizing the actors and their relationships with use case diagrams, I wrote thorough use case descriptions. This argument emphasized how crucial it is to have a strong design that takes into account all of the requirements and application specifications. Selecting the best method for gathering data for the Smart Shopper app from different Dutch grocery stores was one of the biggest development problems. We conducted a thorough analysis before deciding to effectively get real-time data on items, prices, and promotions using the Checkjebon API. This option complied with the internship's duration and functional requirements for the application.

It was my duty to carry out the all the use cases throughout the implementation phase. I had difficulties integrating with the Checkjebon API and making sure that data flowed well, but I was able to overcome them by working with my teammates and consulting with clients. During the internship, I refined my abilities in efficient communication, managing time, and allocating resources. I developed my practical software development experience by utilizing a variety of approaches, including Agile development, requirement collecting, design, and execution.

To conclude my software programming abilities were significantly enhanced by the Smart Shopper project. I'm appreciative of the chance to work at CGI and gain knowledge from seasoned experts on this project. I'm excited to use the knowledge I've gained from this internship in next project phases.