Foodie keepers

*Group name: FC Foodie keepers

Yeol Yang
Information System
Hanyang Univ.
Nanjing, China
yeolyang77@gmail.com

Daphnée Correia
Electrical eng.
Hanyang Univ.
Paris, France
daphnee.correia@edu.devinci.fr

Edouard Maurice
Information System
Hanyang Univ.
Paris, France
edouard.maurice@edu.devinci.fr

Pacôme Manceaux
Electronical eng.
Hanyang Univ.
Paris, France
pacome.manceaux@gmail.com

***Abstract*—As our society is getting more concerned about inequalities, we are suggesting a way to end bad alimentation. Because of bad financial situation, a lot of people have to skip meals or use cheap junk food. To answer this issue, our project aims on giving cheap but better food to these people. Furthermore, our solution is ecological and avoid the food to go to waste. Our project is an app that links the restaurant's owners with customers with financial issues. Our app will inform its users when a restaurant nearby has food leftovers.**

***Index Terms*—Food, Online service, Application**

I. Introduction

A. Motivation

The unfinished meals after a grand party or activity, the food that cannot be sold in restaurants and snack stalls every day, or the goods to be eliminated after the supermarket's daily inventory are mainly still edible and intact food, but have to be discarded as garbage.

Accorded to the United Nations, one third of the world's food is discarded, and the main cause of waste is expired or poorly sold goods from large supermarkets all over the world. Discarded food is certainly a loss, but for businesses, it is the choice to maximize benefits to quickly empty inventory and quickly replace foods.

In this world of hyper-consumption, we wanted to give a slight inflection, a small deviation just above the minimum to perhaps participate in a change of our ways of thinking. Let's use this opportunity of premature waste in our nearest shops to introduce a new system of social sharing.

B. Problem statement (client's needs)

We believe that customers need to develop humanized services. Therefore, we decided to develop a software that can reduce food waste and is used in daily life.

Food waste is a major problem worldwide. In the US alone, up to 40 percent of food goes uneaten—meanwhile one in six households didn't have enough money for food last year.

Therefore, to meet the client's requirement and solve the social issue, we figured out our software.

C. Research on any related software

- Olio

Olio is a mobile app for food-sharing, aiming to reduce food waste. It does this by connecting those with surplus food to those who need or wish to consume such food. The food must be edible; it can be raw or cooked, sealed or open.

- Too good to go

Too Good To Go is a mobile application that connects customers to restaurants and stores that have unsold food surplus. The application covers major European cities, and in October 2020, started operations in North America.

- Phenix

One third of what we produce is produced... for nothing. But this isn't an inevitability:at Phenix,we are convinced that all waste can find a second life,as long as we use a bit of imagination

- No food waste

No Food Waste is a movement turned NGO started by Padmanaban Gopalan and his friends Dinesh manickam and Sudhakar Mohan to get rid of the problem of hunger. The team of No Food Waste scouts for marriage halls, institutions and homes that might have excess food.

TABLE I
ROLE ASSIGNMENTS

Role	Name	Task description and etc.
User	Yeol Yang	Assumes which specific services would be popular and needed in the user's point of view. Also searches for the background of the actual services.
Customer	Edouard Maurice	Predicts which requirements could be needed to raise purchasing desire in the customer's point of view. Also when the software development is done, checks if the requirements are sufficient or not.
Software Developer	Daphnée Correia	Draws out a list of software features to satisfy the customer's requirements and works on the actual software development. Tries best to reflect the customer's and user's needs.
Development Manager	Pacôme Manceaux	Totally manages the project schedule and checks the deadline of each role. Helps other roles to communicate with each other smoothly and evaluates the software features.

II. Requirements

1) Creating an account

When first opening the application, the user has to fill some information. First, he will have to enter some basic information such as his name, age and email address. Then he will have to choose whether he is in the buyer side or the seller side. Basically, seller side includes restaurant's owners, supermarket/hypermarket 's owners…and buyer side includes people interested in buying the offered products.

On the buyer side, the user will have to fill more information. He will have to give his address and also choose his favorite dishes among given categories. On the buyer side, the app will require the restaurant's address and the type of food served

2) Type of purchase

When opening the app, the buyer will have two choices: meals prepared in restaurant or groceries shopping.

In the first category, the buyer can buy leftover meals prepared in restaurant, coffee shops…at a cheaper price. And in the second one, the buyer will be allowed to buy basic aliments like eggs, milk, yoghurt…with close expiration dates always at a cheaper price.

3) Customer choice

If the buyer chooses the « meals prepared in restaurants » category, different options will be offered to him to help him make his choice. He can choose the restaurant in which he wants to buy food leftovers regarding the restaurant's location, the type of food he wants to eat or just let the app make suggestions based on the preferences he has filled earlier. To have more accurate suggestions, the buyer can also set price range. Lastly, a kind of surprise choice will also be available for those who want to try something new. With that option, the app will choose a restaurant for the buyer. On the « groceries shopping » category, the user will be allowed to choose by type of product and location of the shop.

4) Payment and pick up

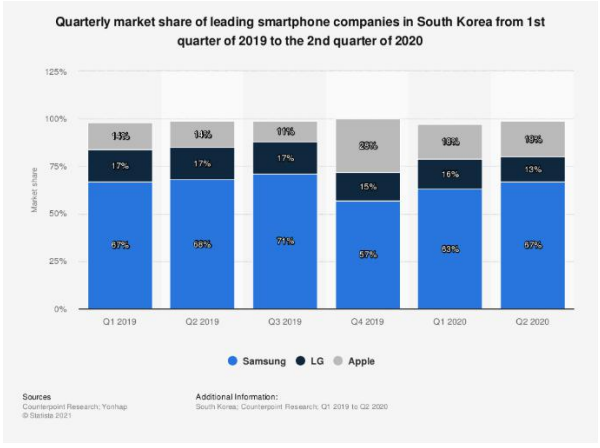
To be as convenient as possible, the app will let the buyer choose whether he wants to do online payment or pay directly by cash at the restaurant, so our app will need to be linked with an online payment device. The app will also include a schedule device to set pick up time. Lastly, two options will be available for the buyer: asking the restaurant to pack the meal, that will be a paying option, or letting the customer bring his own container, which is more sustainable.

III. Developmepment Envrionment

A Choice of software development platform

1) Platform used

The platform we will choose to work on will be a mobile app. Indeed, our project is based on fast services for daily use, so it needs to be usable at any time to be very convenient for the user. As we all carry our smartphone with us at any time of the day, a mobile app is the most appropriate platform for our service to provide fast and easy-to-use services. Plus, this app needs to be cross-platform. Since our service is based on a collaboration between professionals (restaurant owners, supermarket managers…) and individuals, the more users there are, the more diversity there will be and therefore the more attractive our service will be. We will focus on Android and iOS operating systems because the leading smartphones' companies in South Korea are Samsung (Android), LG (Android) and iPhone (iOS), according to this Statista's analysis. But for now, as our programming skills and experience in software engineering are at a development stage, we will only focus on one of the OS previously mentioned, which is Android.



Source : Statista

2) Programming language

- C sharp

Concerning the programming language, we will use C#, and more particularly the open-source platform Xamarin. We want to work with C# because all the team members have knowledge and experience on coding with this language. In addition, C# is a very convenient language when it comes to develop mobile applications because it is object oriented, so error detection is simpler, and it has a large community (it is ranked on top 10 most used programming languages among developers), which means that finding support or answers for questions is not as hard as it might be with a lesser language. Finally, this large community also ensure the continued existence and use of the language.

3) Cost estimation

Hardware	Computer x4 (Programming purpose)	4,300,000 (Microsoft Surface Pro 7)
	Mobile Phone (Testing purpose)	895,400 (Samsung Galaxy S21 5G)
		1,106,899 (iPhone 12 Pro)
Software	Back-end server (Firebase)	0 (Free Trial)
	Proto.io	0(Free Trial)
	GitHub	0
	Visual Studio	0
Human resources	Designer	4,060,000
	TOTAL	10,362,299

- Why the Microsoft Surface Pro 7:

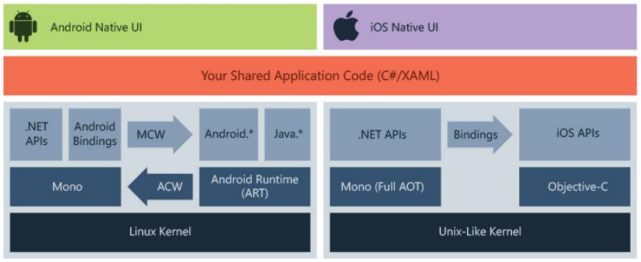
This laptop is powered by the Intel Core i5-1035G4, which is a 10th Gen processor, and an integrated GPU. There are more powerful GPUs in the market but as we are not developing a video game it doesn't really matter and it should be more than enough. This machine offers 8GB RAM and 128GB SSD-based storage, so compiling android development files and programs will take less time compared to a laptop with HDD-based storage. 8GB RAM is good enough to read data fast. Like most of the laptops made by Microsoft, this one does not support storage or memory expansion, but the development of our mobile application will not need us to have extra storage. It is also provided with Windows 10 OS, which makes things easier when it comes to using Visual Studio. This is a lightweight laptop that comes with a detachable keyboard and trackpad, it is just 1.1 pounds of weight. Though it is a full-fledged laptop, it is as light as a tablet. It means that we can easily develop new apps from coffee shops or other establishments and the laptop can be carried in a pouch. Lastly, the laptop offers a battery life of up to 10.5 hours on a single charge with support for fast charging, which is very convenient as we are used to meet in cafes for team meetings.

- Why the Samsung Galaxy S21 & iPhone 12 Pro:

To make our choice, we just searched for the most frequently bought smartphones in Korea using Android and iOS operating systems. As we plan to make our app available under this two OS, it is important that we can test how our app runs under two different operating systems to make sure that the app can be successfully downloaded, executed, and that it can interact with the supporting back-end content infrastructure.

4) Information of your development environment

Regarding Xamarin platform, it will allow us to reach a lot of our prerequisites. Xamarin will make us able to develop a cross-platform app because it contains reusable code, 90% of which can be recycled for the development of apps on various platforms. Xamarin applications can be written on PC or Mac and compile into native application packages, such as an .apk file on Android, or an .ipa file on iOS. The platform IDE enables such C# coding that the result is a native look and feel of the mobile app. As Xamarin is built on top of .NET, it also automatically handles tasks such as memory allocation, garbage collection and interoperability with underlying platforms.



Explanation on how cross platform development works with Xamarin

- Microsoft Visual Studio

All of this will allow us to save time, be more efficient and offer a comfortable user experience. Since we plan to use Xamarin, we will also need to use Visual Studio as a code editor.

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. The most basic edition of Visual Studio, the Community edition, is available free of charge. This is the edition we will

use for the development of our project.

- Firebase (Amazon Web Service)

Firebase is a Backend-as-a-Service (BaaS) proposed by Google. BaaS is a cloud computing service model using which the web app and mobile app developers can connect their applications with backend cloud storage and APIs rendered by the backend applications, which is exactly what we want to do. The cloud-hosted NoSQL database is a real-time database that helps store and synchronize the data between the clients and it is offer through Firebase. This will allow us to avoid the step of linking a separate database with our server and thus will greatly facilitate things. Plus, the fact that it is a real time database allow to make real time updates.

Firebase also provide an easy login process with the Firebase Authentication. It has been proven to be very secure and easy to deploy, which is perfect for beginners like us.

Firebase is also equipped with AdMob, which is an in-app advertising facility that helps the app owner to underline the monetizing policies for his business. It is an interesting part because our app is a free app, based on collaboration between users and even if money is exchanged through our app, we don't take commissions on it. In-app adds are the best way to monetize this kind of application.

Moreover, Firebase Cloud Messaging offers a way to send notifications and messages to targeted audiences for free across all devices and platforms. As we will need to send notifications to store owners and customers when an order has been made, getting it for free is really interesting regarding the huge number of notifications we will have to send.

Lastly, the platform doesn't charge for most of its services and requires choosing a pricing plan only after reaching a certain amount of database memory. So, we will be able to use it for free.

- Proto.io

Proto.io is an industry-leading web-based prototyping platform, suitable for UX designers, entrepreneurs, product managers, marketers or even students like us.

This tool makes prototyping fast and easy, with a lot of different features and all-ready designed templates that helps to save a huge amount of time when designing our app. The platform helps build, share, present, and test interactive prototypes for any screen size: mobile, tablet, web, TV, smartwatch…

Regarding our project, the interesting part for us will be to use the mobile screen simulator. The share features of the platform are also very useful. You can add members to a project and give them roles: tester, designer, owner…which will allow all the team members to work together on the same project in real-time.

Finally, like Firebase, Proto.io is available on free trial.

- Github

GitHub is a website for developers and programmers to collaboratively work on code. The primary benefit of GitHub is its version control system, which allows for seamless collaboration without compromising the integrity of the original project.

B Software in use

- Google Maps API

The restaurants proposing meals on our platform will be presented on a map. A user will be able to access the offers in a specific area. Thus, it appears convenient using the Google Map API service to implement this functionality. The Xamarin framework facilitates this task by offering a NuGet package for map management, in addition for being cross-platform (Android and Apple).

- Chayxana

Chayxana is an open project for a mobile restaurant application on GitHub offering similar services to ours based on the Xamarin framework and coded in C#. We are thinking of taking some parts of the code and adapting it to our project.

C Task distribution

Name	Task
Yeol Yang	UI-restaurant information page, Documentation
Edouard Maurice	UI-login page, Documentation
Daphnée Correia	Documentation, Prototype Designing with Proto.io
Pacôme Manceaux	UI-list of restaurants page, Documentation

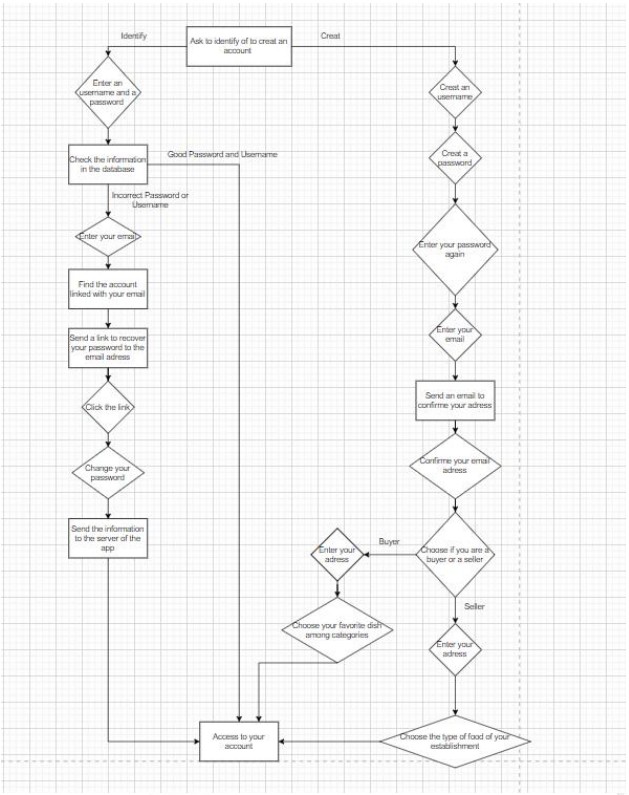
IV. Specifications

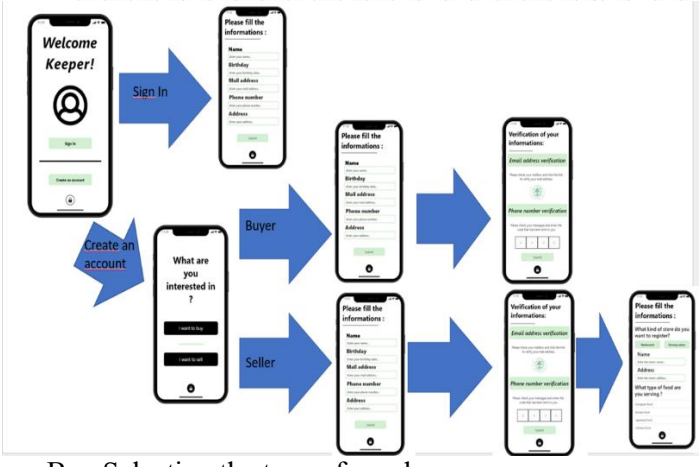
A. Creating and logging to an account

When the user opens the app, the first thing he will see will be a choice to either login to an account or sign

up. If he chooses to sign up, he will have to enter his username and password. The information will be sent to our database to check if this username is link to this password. If not, then it will ask again and if he want to recover his password. If he doesn't remember the password, then he will have to input his email address. The information will be sends to our database, who will send a link to his mail to recover his password.

If the user presses the button signs up. Then, he will have to input a username, a password and a mail address. Then two buttons will appear, asking if he is a buyer or a seller of food. If he selects buyer, he will have to input his location and select a category of food he prefers. If he chose seller, he will input also his location, and he will select the categories of food he is selling. All this information will be filled in a class then sent to the database.





B. Selecting the type of purchase

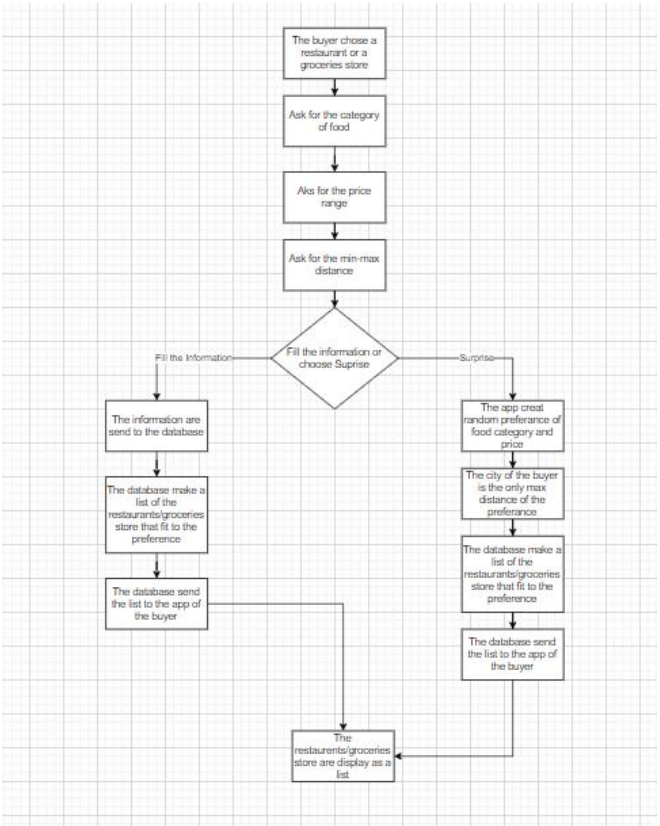
After logging to his account, the app will show to the buyer two buttons. The first one asks if he wants to buy leftovers from restaurants. The second one, suggest to buy basic food like eggs, milk and yogurt with a close expiration date from a supermarket.

C. Customer Choice

Once the buyer has selected the first button, the one to receive leftovers from restaurants, a page with multiples categories of food, a price range and the max-min distance of the restaurant. At the bottom of the page, there will be two button, one that says "Save", where the buyer saves his selection. The information will be sends to the database. Firstly, to look for restaurants that fits in the buyer preference, but also to save his preference. Once the restaurants have been found, the database will send all their information, and they will appear in a list in the app.

The other will say "No Selection". This means that the app will find randomly a restaurant. For that, the preference like the categories and the price will be chosen randomly by the app, only the address has to be the same city. The random information will be sent to the database, to find multiples restaurants fitting the categories. Then they will appear as a list to the buyer.

If the buyer has selected the other button, the one to receive food from supermarket. Then the process is almost the same as the first button. The buyer can set up a price range, choose a category and a min-max distance from him. The information is also sent to the database to find a list of groceries stores that fits the preference, and save the preference of this buyer. Then, the groceries stores that fits this category are shown as a list in the app.

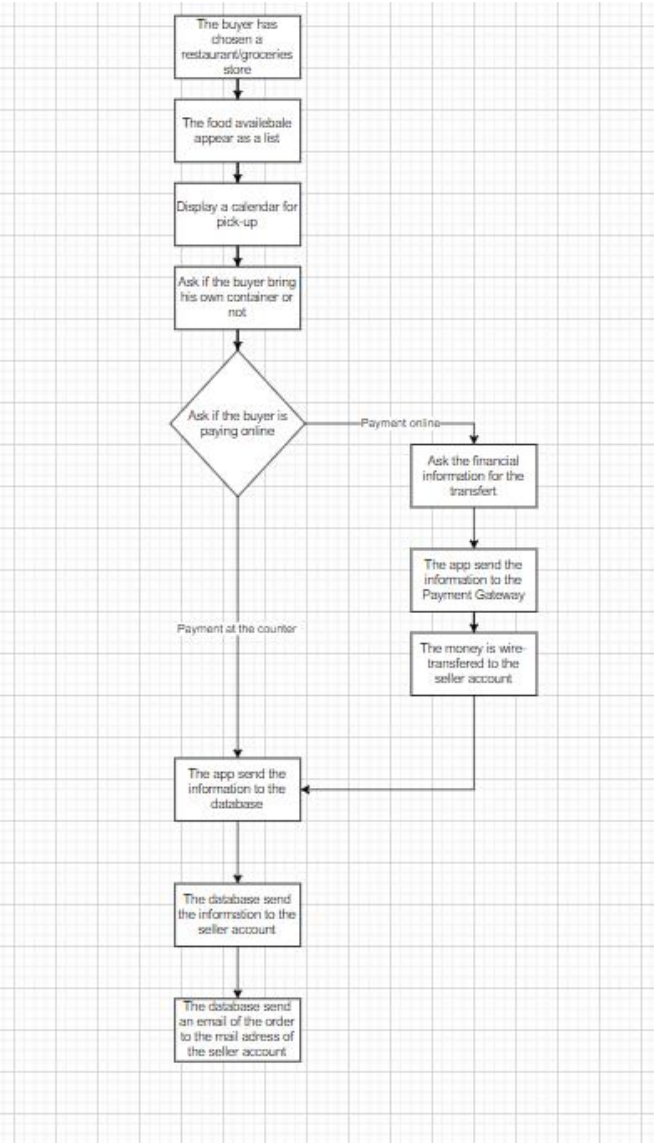


D. Payment and pickup

Once the buyer has chosen his restaurant or groceries store, the different food available will appear, with their price, as a list. The buyer will just have to click on the name of one food from the list. After, a calendar will appear, where the buyer can set the hour and the day of the delivery within the limits of the restaurant or groceries store. Then a button will appear asking if the buyer want to bring his own container for the food. If he doesn't, he can just press on a cross on the top-right of the screen. Finally, two buttons will appear, one asking to pay online, the other to pay directly at the counter.

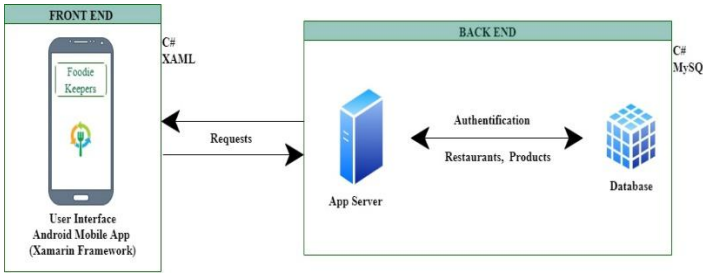
If the buyer chooses to pay at the counter, then the app will send the class that took all the information of his order to the server. Then, the server will send it to the account of the seller. A mail will also be sent to the email address of the seller account.

If he chooses otherwise, then he will have to fill his financial information for a payment online. The buyer can choose to either pay by Paypal or by credit card, by pressing some buttons. He will have to fill his financial information. Then the information will be sent to our Payment Gateway. Once the payment is complete, the order is sent to the restaurant or grocery store like in the payment at the counter option. The application will wire transfer the money to the seller.



V. Artecture design & Implementation

A. Overall architecture



The overall architecture of FoodieKeepers can be represented in two main modules that are the frontend and the backend. The frontend brings together the user experience on Android with the graphical interface and user interaction supported by the Xamarin framework. It is from here that an individual can add products for sale if he is a restaurant owner for example or make orders as a customer.

The backend is composed of a server linked to a database. When a user logs in, adds a product or places an order, a request is sent to the server. The server thus consults the database to access existing data or to add new data and sends the necessary information back to the frontend. The database contains all the user identification information, the list of restaurants, the products of each restaurant and the associated pictures.

B. Directory organization

Directory	**Content**	**Module Name**
/FoodieKeepers	Root directory, contains the _git* files and all subsequent folders.	Github Document
/FoodieKeepers/ Document	Documentation folder containing the LaTeX file and the PDF.	Document
/FoodieKeepers/ res	Images folder that contains the images we used.	Image UI interface
/FoodieKeepers/ SourceCode	Contains the main code we created for our project.	Visual Studio API

C. Module 1: Image for UI Interface

The UI interface is done with Xamarin in XAML code. We added free image found on the internet to complete our UI. We used Proto.io to design it first. Most of our image were found in this website. The user can only see this interface when lunching the app. The image and the UI are not located in the exact same repository. In Xamarin, the front-end and the back-end of the code is deeply connected. The UI calls some classes that are part of the back-end. So, this repository is more of a stock for image.

D. Module 2: API

The front-end and back-end were coded on Xamarin with Visual Studio. As said previously, they are both connected in the same project due to the environment. The code of the UI is XAML, while the back-end has been coded in C#. Theses languages have been chosen because of the environment, but also due to our knowledge of them. The back-end of the code use also functions provided by our back-end server Firebase.

Firebase provides us also a database, in plus of a back-end server. It is used for different function. It provides us a link to the database who stock all the users' profile. So, we can create more profile when a user creates an account, restaurant or customer. Also, we can provide information in the database, to help a user to sign in or to find a restaurant. We add the Firebase components we need as NuGet packages. They allow us to create specific C# function to send information to the database. We have chosen Firebase on the advice of the teacher. It is said to be easier to use for a first Software Engineering project. It provides a lot of tutorials and information on how to use its functionalities. Moreover, it is one of the most common back-end and database used for developing mobile app with Xamarin.

VI. Use Cases

A. Creating an account

account.

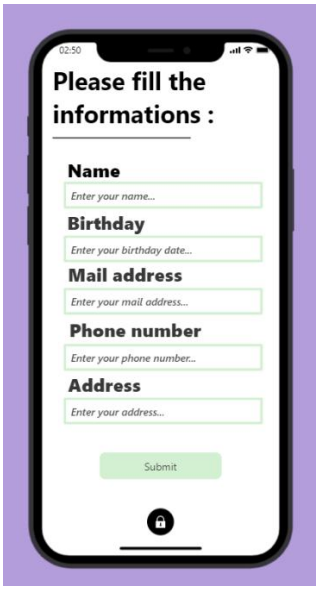
When entering the application, the loading page of the app is displayed.

When the loading is finished, the user is provided with two choices: log-in or create an account. To create an account, the user has to click on the "Create an account" button displayed on the second screen of our app.

After clicking on this button, the user will be moved to another page which will let him choose if he wants to buy or sell products on the app. Then, he has to click on one of the two boutons to confirm his choice.

1. Buyer side

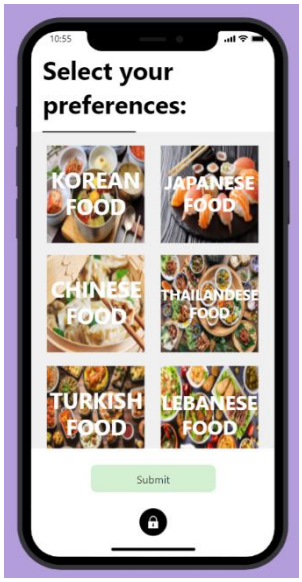
By clicking to the “I want to buy” button, the user will be led to another page which will allow him to fill his personal information. All the data entered by the user will lead to the creation of a new client profile in our database, in which the information will be stored (name, age, date of birth, address, phone number, mail address…).



After achieving this step, the user will have to verify the entered information to avoid typing errors when filling the mail address and the phone number. As this is crucial for the future use of the app, we want to make sure that such errors have been avoided. Indeed, we will need to use the phone number of the user as a link between him and the store owners after making an order if any of the two parts needs more information for example. The mail address will be used to send promotional offers, newsletters, and to recover the user’s account if he has lost his password and no longer has access to his phone number. To verify the information, we will send a mail to the mail address filled by the user with a clickable link.

After clicking on the link, the mail address will be stated as verified and then stored in the database. Same with the phone number. We will send a code by SMS, and the user will have to enter the code he has received on the allocated spaces. The entered numbers will be compared with those who were sent, and if they correspond, the phone number will be stated as verified and then registered in the database.

Finally, the user will have to fill one last step. He will have to choose his favorite kinds of food amongst the proposed selection. He can select as many kinds as he wants but he must at least select two kinds of food to complete his registration. The choice of the user will also be registered in the database to be used later. This information will help us to suggest the user various restaurants based on his preferences to offer the most personalized experience possible.

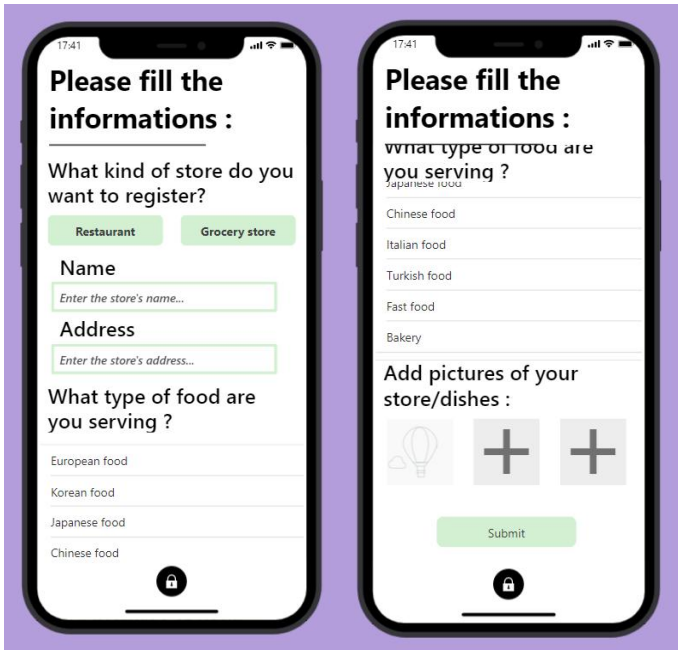


This page is scrollable so that we can propose a large selection of food to satisfy all tastes but keep an airy and functional display at the same time. When clicking on one of the categories, the button fades to inform the user about what he has already selected.

2. Seller side

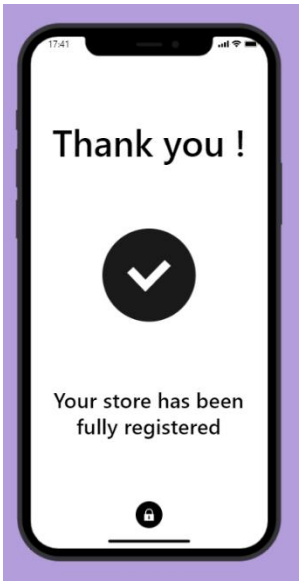
By clicking on the “I want to sell” button, the user will be led to the seller part of the application.

The first steps of the registration are similar to the buyer side registration. The user will have to fill the same information page with personal data such as name, date of birth…Then he will have to go through the verification steps. After the verification has been done, he will then have to fill his store information. Another page will appear to let him this. The user will have to give the type of store he wants to register: a restaurant or a grocery store. This is an important step since our app will separate restaurants and grocery stores when proposing goods to the buyer. Then we will need the name of the store, the type of food sold if it is a restaurant, the address and then a few pictures of the front of the building or dishes made in the restaurant for example.



As we can see, the whole page is scrollable, except the title part of the page. Plus, the list of food is also itself scrollable. To select the food sold, the user will simply have to click on the category. He is allowed to select multiple categories, but he must select at least one category. If not, an error message will be displayed when clicking on the “Submit” button. This doesn’t apply to grocery store owners. They will not have to select a category.

After filling all this, a page will be displayed to notify the user that his store has been fully registered. He will then be redirected to his main page.

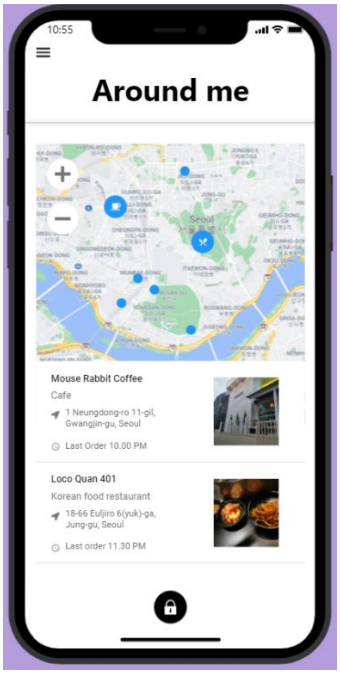


A. *Make and track an order*

After having created an account or signed up if the user already had an account, he will be redirected to the main page of the application. In this main page, different sections will be displayed. The first section is divided into several categories which will help the user choose where and what to order. First category is “Type of food”. By clicking on it, the user will be redirected to a page listing the different kinds of food available.

Then the user will click on what he wants to try and a list of restaurants serving this type of food will be displayed. The list will be sorted regarding the user’s location to be more convenient and avoid proposing restaurants located too far.

The second category is “Around me”. This category will open a map showing all the restaurants proposing leftovers around the user’s position.

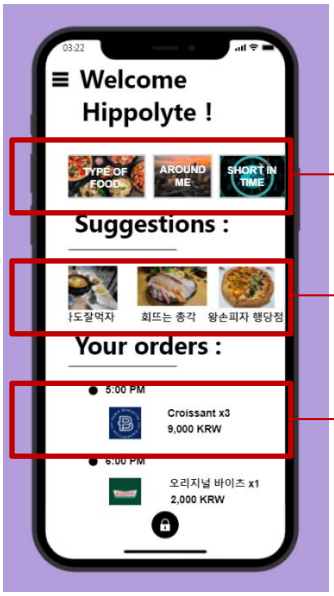


The third one is “Short in time” category. This one will show the user the restaurants with the shortest pick-up time so that the user can enjoy a meal fast if he has a sudden craving or needs to eat fast to go back to work for example.

The fourth category is “Grocery shopping”. As mentioned above, we want our app to work not only with restaurants’ owners but also with supermarket,

convenience stores…. So, clicking on this category will allow the user to see which store offers food items around him.

Last category is for those who are tired of eating the same thing and want to experience something new. The “Mystery choice” category will propose a restaurant to the user. The selection criteria will be the user’s position (it is important to not be too far, for the same reasons as mentioned above), the user’s preferences filled during the creation of his account and his former orders to avoid suggesting something he has already tried.



The different categories that have been discussed

Restaurants proposed in the "Suggestions" part

Summary of the order

As this part is also scrollable, all the categories can't be shown on the screenshot.

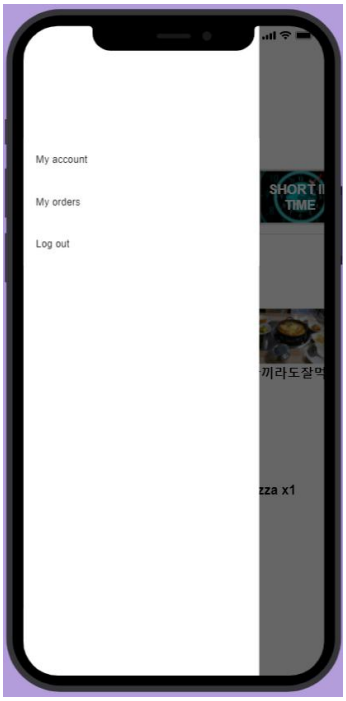
Underneath this part, there is also a section called "Suggestions". This is a section for those who want to make their choice quickly without having to take a look in the different categories, or for those that the displayed pictures have made hungry. The suggestions are made based on the user's preferences.

After having his choice made, the user will click on the restaurant's picture. This will lead him to the restaurant's page. He will then be able to see all the proposed products with their price and the restaurant address on the map. If the user clicks on the map, it will open his navigation app and then he will be able to see how far he is from the restaurant and how he can go there.

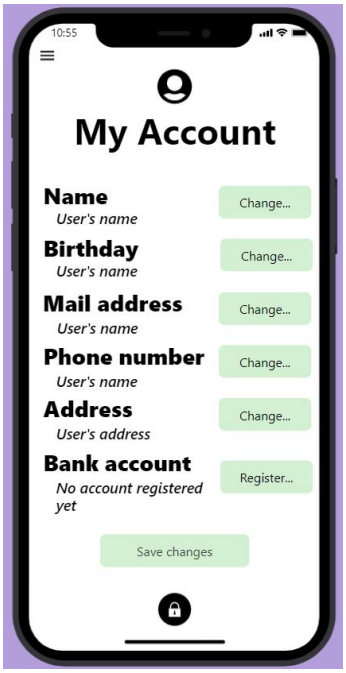
After choosing the products he wants to order by clicking on the images and adding them to the cart, he will be able to finalize his order. For this, he will choose a pick-up time in the timetable, confirm the products in the cart and choose his payment method: by cash directly in the restaurant or by card on the application. If he chooses to pay by card, he will have to fill his banking information. The user can also fill this information before making the order by registering it in his account, but we will come back to this later.

After having the order done, the user will receive a message to confirm that it has been well registered and a summary of the important information (time of pick-up, name of the restaurant, address…). Then, the order will also be displayed on the main page of the app as shown in the second text part to be sure that the user won't forget it and can easily check the information. All the three parts mentioned are scrollable too.

There is also a sliding menu on the top left of the screen. This menu has three categories: "My account", "My orders", "Log out".



The "My account" section will allow the user to modify the information he has previously filled. He will also be able to register a credit card for the payment of his future orders.



The "My orders" section will provide the user a summary of all the orders he has made, with the restaurant's name, the items purchased, the price, the date of the purchase…

Finally, the “Log out” section will allow the user to disconnect from his account. The vast majority of our screens have a sliding menu too, except the registration parts, to allow an easy and intuitive navigation between the pages.

VII. DISCUSSION

This is our first project in Software Engineering. From the beginning, we wanted to do something really useful for our first time. The idea of Foodie Keepers was quickly found. The purpose was to make a software that would fit to our competencies but would also help some people. There are only a few examples of this app in the world, and none in Korea. It took us a lot of time to understand some concepts in this course, and a lot of mistakes were made in choosing the components.

A big part of our work was the research. We had to look out for numerous tutorials and examples on the internet. For some of our weekly meetings, we had to explain what we found and suggest some components. To make sure we don’t spend too much time and efforts on the same parts. We divided the project into different tasks for each of us. Every weekly meeting was there to see where we were on each of them and decide our next step. More than often, we also added multiples meeting in one week.

Our biggest difficulty was the back-end server and the database. At first, we tried to use AWS. One of our members had a little experience on using this. It turned out that most of the app on Xamarin doesn’t use AWS. All the example of a food ordering app used mostly Microsoft Azure. We chose then to use this one. However, Azure is a very complex back-end server. It was too much time consuming to understand how it works. On the advice of the teacher, we took Firebase, a much easier back-end server, using a real time database. Our functions were created by using this back-end server.

The project was hard to achieve. Nonetheless, we learned a lot. We got a better understanding on what component fit the best for developing an application. We also learned the relationship between the different parts of a software, how the UI is connected to the back-end server who is connected to the database. Furthermore, we discovered how we should manage a team for a software engineering project. It isn’t obvious to see where we start, how to motivates the other members, and dividing the tasks to be faster.