Bilkent University
Department of Computer Engineering

CS 491 - Senior Design Project I

Foodie

High-Level Design Report

Utku Gökçen 21703746

İlhan Koç 21603429

Emre Erciyas 21600991

Yiğit Erkal 21601521

Supervisor: Fazlı Can

Jury Members: Shervin Arashloo and Hamdi Dibeklioglu

Innovation Expert: Murat Ergun

December 24, 2021

1 Introduction

Food waste has become a major problem globally in recent years. According to the research conducted by the United Nations Environment Program, more than 7.7 million tons of food is wasted in Turkey every year. In addition, according to the 2021 United Nations Food Waste Index Report, 93 kilograms of food per person is thrown away every year in Turkey. This means that 17 percent of ready-to-eat food in retail outlets, homes and restaurants globally goes straight to waste[1].

Before researching the solutions to this problem, it is necessary to talk about the main causes of the problem. The main reasons for food waste in mass consumption areas are the inability of restaurants, hotels or cafeterias to adjust the size of the meals offered to their customers, buying more food than they need, not having enough storage space for the food they have and inability to track the expiration dates of the food. Restaurants, hotels or cafeterias should avoid these situations that cause food waste and the leftover food should be re-evaluated in the restaurants and hotels, if it cannot be re-evaluated, it should be donated to the people, if it cannot be donated, it should be considered as animal feed, if it cannot be used as animal feed, it should be directed to composting and ultimately sent to landfill as a last resort[2]. Food waste in businesses might occur because of many reasons; however, a study done in the US has shown that:

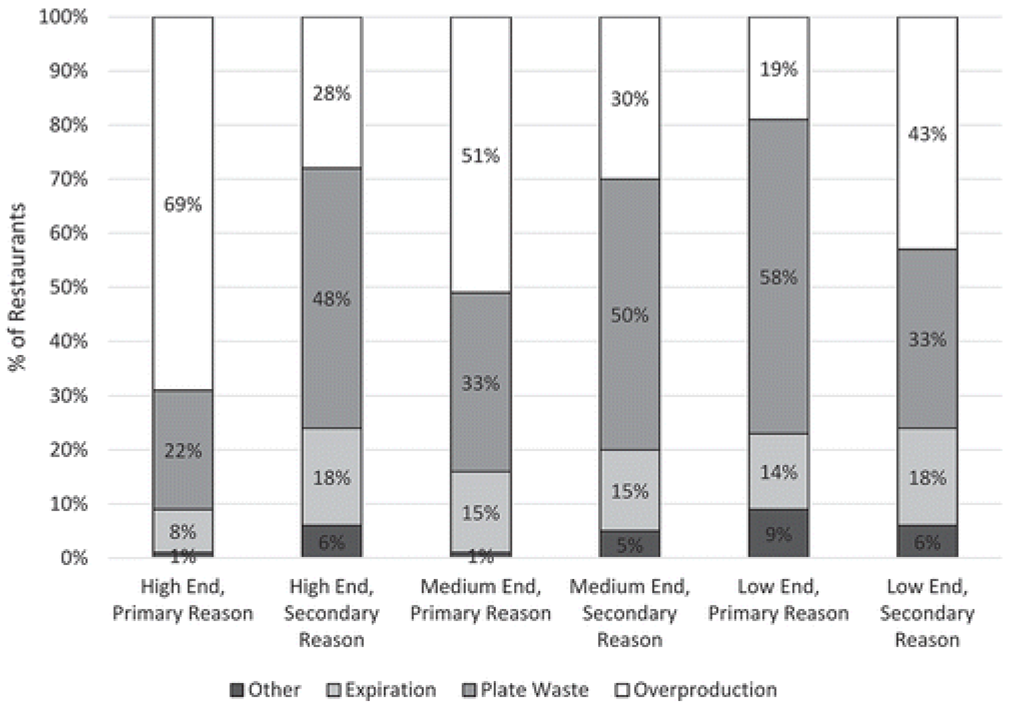

Figure 1: Food waste stated by restaurants in the US

The purpose of our senior design project is to create a platform where restaurants, hotels or cafeterias can sell leftover food at lower prices to people or animal farms when they cannot re-evaluate the leftover food. In high end and medium end restaurants, the main cause of the food waste is overproduction which is a perfect fit for Foodie [3]. In this way, leftover meals will not be wasted and both customers and businesses will be profitable by adopting a win-win approach.

1.1 Purpose of the System

Foodie is designed to help restaurants not to throw away leftover foods and let people get discounted meals in an attempt to achieve more sustainability. With foodie we are aiming to decrease food waste in big cities. Many restaurants will be able to make use of excess foods in their business. Foodie will also provide its users with great opportunities to grow their businesses and will let its customers discover many new restaurants.

1.2 Design Goals

1.2.1 Usability

Usability is one of the most significant design goals of Foodie. We expect our mobile application to be used by people with different backgrounds and various technological knowledge. Therefore, Foodie should be easy to use while providing its users with sufficient functionality. To achieve this:

● After opening the application for the first time, a user manual will be displayed explaining how to use the application.

● Foodie will provide its users a simple and convenient interface that they can use easily.

● Foodie will be developed to run on different phone and tablet resolutions.

1.2.2 Security

Security is fundamental for our application since Foodie users will be allowed to use online payment through the app in addition to using their customized profiles. In order to achieve this:

● Authentication system will be used to login to the application. Therefore, users' email, phone numbers and passwords will not be visible to other users.

● Users' passwords will be stored in Firebase in an encrypted format.

● User's card information will not be stored after in-app payment.

1.3 Definitions, Acronyms, and Abbreviations

MVVM	Model–view–viewmodel (MVVM) is a software architectural pattern that facilitates the separation of the development of the graphical user interface (the *view*) – be it via a markup language or GUI code – from the development of the business logic or back-end logic (the *model*) so that the view is not dependent on any specific model platform [4].
Rest API	A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services [5].
Firebase	A database system.

1.4 Overview

In a world where wasted food in developed countries could potentially feed all the starving people, proper food management is getting more significant every day. In an attempt to create a more sustainable and fair world we created Foodie. Foodie will be a mobile application designed for people from all ages that are willing to buy leftover food for a better cost. The purpose of the creation of our mobile application is to let customers display leftover foods in nearby restaurants and cafes. The leftover foods might be a menu item or a single meal depending on the previous order. Customers will be able to use Foodie as long as they have their mobile phones with them. It is fundamental to create a simple interface for our users to be able to reach people with various backgrounds and technology knowledge.

Foodie will be supported by Android Studio which will provide our mobile application with powerful frameworks which include online payment, location services, good quality interface and models etc. For the back end, we will be using firebase as our database and rest API as our main database control tool.

2 Proposed Software Architecture

2.1 Overview

Foodie will be designed as an application that is easy to access and use. For this reason, the software architecture of the application is built by dividing the application into different layers. This layered architecture ensures that the user interacts only with the end product, and other backend processes are done on servers other than the client side. In addition, the data used in the application is stored on a different server than the client side. That is, the application is divided into different subsystems and each subsystem has different tasks. In this way, we aim to increase the compatibility of the subsystems and minimize the coupling between them.

We plan to use MVVM as the software architectural pattern of the application. The MVVM pattern separates the presentation logic (views or UI) from the application business logic. Layering the application in this way allows us to test the user interface and core business logic separately, and it becomes easier to find and fix the bugs in the user interface or in the backend.

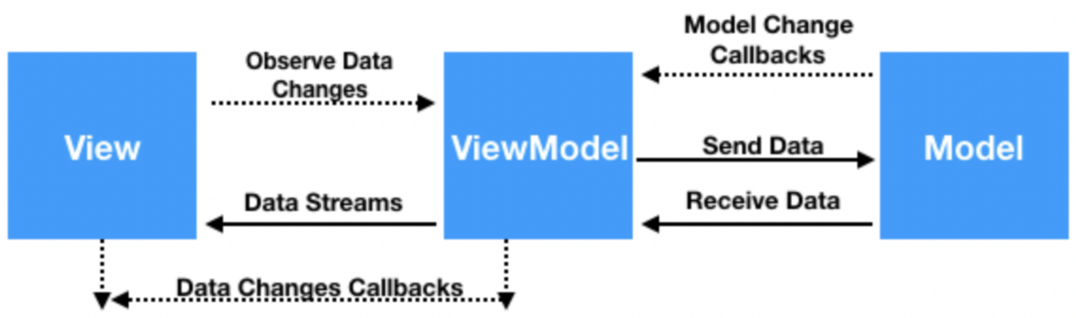

Figure 2: MVVM Software Architectural Pattern

The MVVM architectural pattern consists of three basic structures: Model, View, and ViewModel. The purpose of these structures will be explained in detail in the subsequent sections.

2.2 Subsystem Decomposition

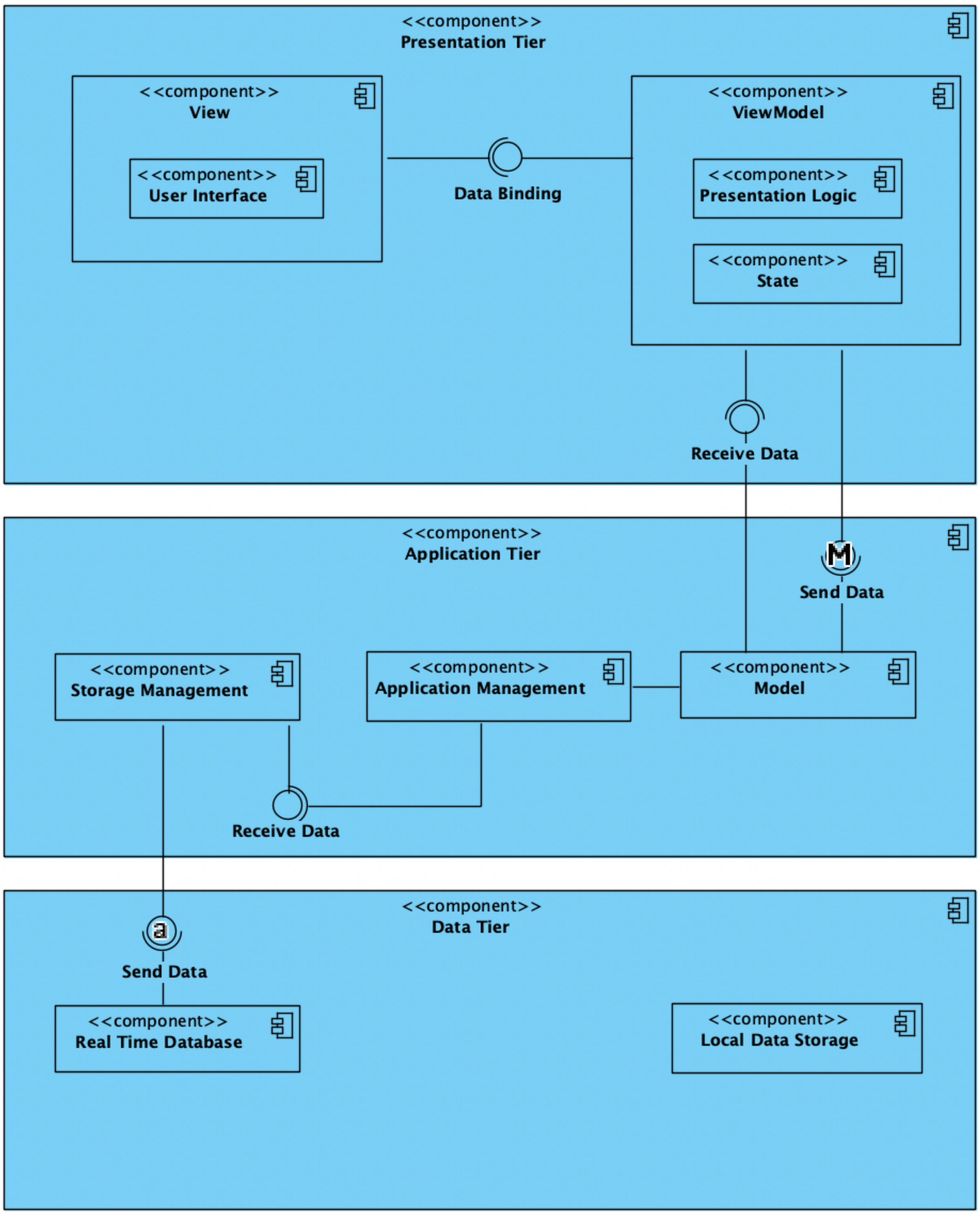

Figure 3: Subsystem Decomposition Diagram of Foodie

In accordance with the design goals of our application, we divided the system into three different subsystems called Presentation Tier, Application Tier and Data Tier by following the 3-tier architecture logic. Since Foodie will be a mobile application, this type of architectural style is suitable to decrease the coupling between subsystems. This style provides flexibility to develop and update each layer with minimum dependency to other layers.

Presentation tier is the subsystem at the top layer of subsystems where the user interacts with the application. Main function of this subsystem is to transform the processes in the Application tier into a form that the user can see and interact with. In the Presentation tier, the View component is responsible for providing the user interface of the application and the ViewModel component is responsible for binding the data to connect the UI elements in View to the controls in ViewModel.

Application tier is responsible for coordinating the operations performed within the application. This layer allows data to be retrieved from the Data tier, processed and then sent to the Presentation tier. In the Application tier, Application Management component is responsible for handling the requests from the user and responding back to the Presentation tier. It contains the components of the operations that can be performed within the application such as payment, ordering or chatting. Model component is responsible for creating the objects to be used in the application. It sends the data processed in the Application tier to the ViewModel component in the Presentation tier.

Data tier is responsible for storing data both in the user’s local and in the realtime database. In this layer, the data is sent to the Application tier to be processed. The main functions of the layer is to provide secure access to the database, to protect the privacy of the data and to transfer the correct data to the correct place. We are using Firebase Realtime Database for data storage and retrieval, meaning that we store the data in the realtime database and it allows us to store and sync data between users in real time. Local data storage is responsible for storing the data in the users’ devices local and we save only cache data in local data storage to enable fast loading in case the same or similar data is requested multiple times.

2.3 Hardware/Software Mapping

Foodie will be an application which is compatible with Android operating system devices. Users can interact with the servers of the application by using their Android smartphones or tablets. Users can display listed meals, buy food and manage their profiles via their Android devices.

We will plan to use the Foodie Backend Application to process the data viewed by the end user in the application. Backend application gets data such as user and stock information from Firebase Database instantly. These data are processed and made available to the end user through layered architecture of Foodie. After the data is processed, it is sent to the client side using the Rest API and is presented to the user. Foodie Backend Application also includes location, authentication and instant chat services that increases the functionality of the application. All services running on the backend server work bidirectionally. They receive data from Firebase, process it and transfer it to the client side. For this reason, Foodie Backend Application acts as a bridge between Data Server and Client.

There will be a realtime database system to keep all data related to the application. Database system will be connected to the Application Server and Application Server will perform the communication between database and client.

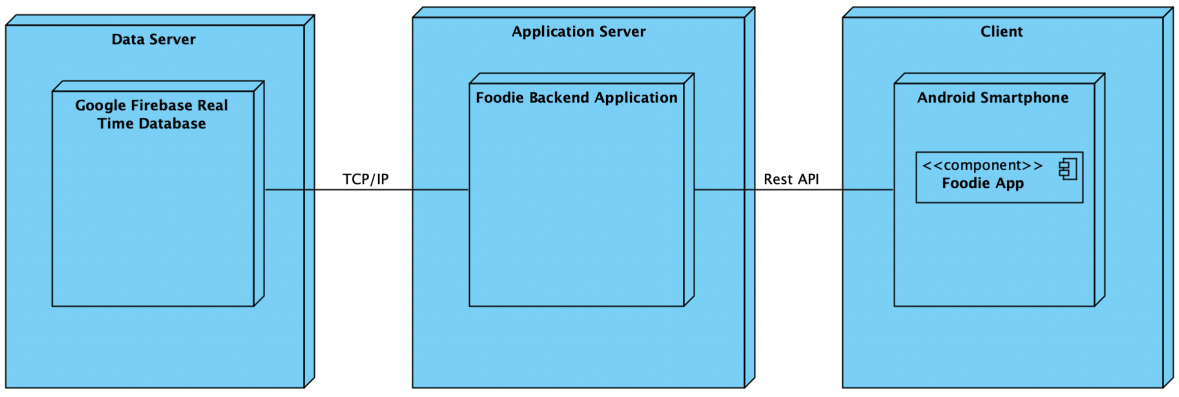

Figure 4: Deployment Diagram of Foodie

2.4 Persistent Data Management

In the application, users' data and information will be stored in Firebase Realtime Database. We will keep users’ personal information, location and other data in json format and be able to update them 24/7. In addition, private data such as stock information of businesses and shopping data of users will be stored in the Firebase Realtime Database and can be updated instantly. Since Foodie will have a realtime database, all the stock data will be updated and users will not encounter stock problems while shopping. In addition, thanks to the "Room" library we will use, the data will be stored on the users' devices' local and this data can be accessed quickly when necessary. This will increase the working efficiency of the application.

2.5 Access Control and Security

Our application uses payment information, business information and customer information. In our system, we give importance to privacy and personal data. If someone wants to be a customer in our application, he or she has to fill in the required information such as name, surname, address, age, location. We will store this information in our database. However, creators of the application and the creator of that customer will be the only ones who can access the personal information. For the business information, the same process will be used. Creator of the profile and admins will be the ones who can access the personal information of the business. However, the address will be accessible for the whole customers as an addition. As a last note, payment information will not be stored in the database or any other storage. Customers will fill the required credit card information for that product and payment will be done. After the completing payment, the personal payment information will be deleted permanently so that no one will reach that information.

2.6 Global Software Control

Server of our application will control and manage the all-system operations. When a customer gives an order, with the help of database operations, our server will send the required information to the business owner. Then, if a business accepts the order, data will be updated in our database and customers will be notified. Location control will have a similar process. Customers' data about location will be updated regularly in small time periods and this data will be updated in our database repeatedly. Business performs also will be updated in our database in order to control and to keep updated. For instance, when the business owner makes the change in the menu. This data will be overwritten in our database. Then, customers will be able to see the updated menu.

2.7 Boundary Conditions

In our application, we determined three main boundary conditions which are initialization, termination and failure.

Initialization

To use our application, a user needs a phone with Android OS. User can download our application from Google Play Store to his or her mobile phone. After downloading the application, stable internet connection is another need for the usage of the application. As a last thing, customers have to allow us to access their location information. According to their location, the business list will be shaped.

Termination

The user can close or exit the application whenever he or she wants. In our application, the server always works. Hence, if the user terminates the application, the changed data will be saved in our database and there will not be any big effect on the system of the application.

Failure

In our application, there are some cases about failure of the app. We listed these possibilities in order to increase the working stability of our application.

First of all, in Foodie, we use databases to collect and save the data. If failure occurs in our database, this kind of failure affects the whole application. Therefore, we decided to have a backup storage, if any problem occurs to our database.

Another case is run-time problems. If our application does not work properly in some functionality, the effect of this failure influences each user. As an example, there may be bugs in the program, collected data can be lost or this data can be written wrong. To solve this case, we will give importance to the testing period. Every functionality will be tested to minimize any kind of failure.

Lastly, our application requires a stable internet connection for the whole usage time. If the user loses internet connection, the program can not run any functionality and this event results in the application termination.

3 Subsystem Services

3.1 Presentation Tier Subsystem

Presentation Tier Subsystem handles the user interface, namely the front-end part of the application. Since we are using the MVVM architectural design pattern, we used View and ViewModel components while constructing this subsystem.

View Component

The user interface of the application is managed by the View component. This interface contains necessary widgets such as buttons, image views or text views for the user to perform operations within the application. The View component contains the formatted data that the end user sees.

ViewModel Component

The ViewModel is used to connect the UI and the Model. There is no direct interaction between View and Model. View performs related operations via ViewModel. ViewModel does not have direct access to View and data is bound to View with data binding method.

3.2 Application Tier Subsystem

Application Tier Subsystem manages the main business logic part of the application. This subsystem allows data to be retrieved from the Data Tier, processed and sent to the Presentation Tier. In addition, in this subsystem, Data Tier and Presentation Tier are updated according to the operations that the end user will perform in the application.

Model Component

Model Component is responsible for creating necessary objects and data for the user interface. These objects and data are transferred to the user interface using the ViewModel Component. After the data in the Model is transferred to the ViewModel, it is transferred to the View by using the data binding method and displayed to the end user.

Application Management Component

Application Management Component is responsible for processing the requests made by the user in the Presentation Tier. It takes requests, performs necessary actions, and updates the Model component. It also requests data from the Storage Management Component, processes this data and sends it to the Presentation Tier through Model Component.

Storage Management Component

Storage Management Component is responsible for storing the necessary data while the application is in use. For example, while the user is viewing a business page, business data is first transferred from the Data Tier to the Storage Management Component. Then, this data is transferred to the Application Management Component where the data is processed. After the data is processed, the Model Component will update itself and send the data to the ViewModel Component.

3.3 Data Tier Subsystem

The Data Tier subsystem is responsible for data management and data communication. This subsystem is secure because it can only be accessed from the Application Tier subsystem. In this way, every change that occurs in the Data Tier subsystem will have a relevant request in the Application Tier subsystem. Also, this subsystem has two different components that are responsible for storing data in different ways.

Realtime Database

This component is responsible for storing all data on the system in the Firebase Realtime Database [6]. Data is taken from this component and transmitted to the Application Tier Subsystem for processing.

Local Data Storage

This component is responsible for storing data in users' locales. We will use SharedPreferences APIs for this purpose [7].

4 Consideration of Various Factors in Engineering Design

Public Health

Restaurants may want to sell their outdated food. This situation is a risk factor for public health so we will check the comments of the restaurants regularly. If there are comments about outdated foods than usual, we will suspend these restaurants strictly.

Public Safety

Our application uses some personal information such as location information. This kind of personal information may be a problem for public safety. Hence, this information will not be shared by any third-party groups.

Public Welfare

Our application is aimed to enhance the public welfare by giving a chance to sell leftover foods with lower prices. Hence, leftover foods will not be wasted and people in need will have a chance to buy discounted foods.

Global Factors

Our application will not have any effect on global factors. Foodie will be a platform especially created for Turkey. Furthermore, we will give an option to change the language to English because of the foreign people who live in Turkey.

Cultural Factors

There will be a cultural interaction with foods. People will get a chance to buy foods from other cultures with lower prices.

Social Factors

Our mobile application aims to create social awareness by finding a way to use leftover foods. Furthermore, people in need will get a chance to buy discounted foods easily.

Environmental Factors

Leftover food is thrown away by most of the restaurants. Restaurant’s wastes may cause more damage to the environment. Our application will find a solution for this waste.

Economic Factors

Our mobile application also aims to find a solution for people in need. With the help of discounted leftover food, these people will reach the food with very optimal prices.

	Effect Level	Effect
Public Health	10	Possibility of selling outdated food
Public Safety	6	Keeping personal data confidential
Public Welfare	8	People who are in need will have a chance to get discounted food.
Global Factors	0	Global factors have no effect on our solution.
Cultural Factors	6	Possibility of eating foods from different cultures.
Social Factors	7	Our application aims to create a social awareness about leftover foods.
Environmental Factors	10	Application will be environmentally friendly and it will aim to minimize leftover foods.
Economic Factors	10	Discounted foods will be an opportunity to people who are in need

5 Teamwork Details

5.1 Contributing and functioning effectively on the team

	Utku	Emre	İlhan	Yiğit
Contributions	● Leading the team and managing issues faced during the development and design process. ● Designing the user interface of the application. ● Designed the system architecture of the application.	● System Goals and Requirement s.	● Writing report	● Prepared use case model, object-class diagram and dynamic models. ● Application research
Functionalities	● System Architecture Designer ● Team Leader ● Product Manager ● UI Designer / Developer			● Full stack Developer

5.2 Helping creating a collaborative and inclusive environment

Among the ideas that came up, only the Foodie application caught the attention of the whole group, and the idea of development became exciting within the group. Working as a group has taught each of us to work collaboratively and has shown how to take responsibility when tasks are distributed. This has taught us the advantages of working in groups. Meetings are held every two weeks to ensure that the roles assigned within the team are fulfilled by each individual. In these meetings, the plan of the coming weeks and the criticism of the past weeks are made. In addition to these, team leader Utku creates a nice and effective working environment by reminding us of urgent work to be done via Whatsapp and Discord and by holding short meetings. In the Foodie application, what has been done so far is tested after each meeting in order to preserve the general structure and bring the project to the closest shape. Meetings within the group are carried out through the Discord application, and face-to-face meetings with our supervisor Fazlı Can.

5.3 Taking lead role and sharing leadership on the team

	Utku	Emre	İlhan	Yiğit
Roles	Team Leader	System Developer / Tester	Full-Stack Developer	UI Designer / Developer

6 References

[1] Dahlberg, L. and Krug, E., 2006. Violence a global public health problem. Ciência & Saúde Coletiva, 11(2), pp.277-292.
[2] “Flutter documentation,” *Flutter*. [Online]. Available: https://flutter.dev/docs. [Accessed: 21-Nov-2020].
[3] Yevvon Yi-Chi Chang. (2021) All you can eat or all you can waste? Effects of alternate serving styles and inducements on food waste in buffet restaurants. *Current Issues in Tourism* 0:0, pages 1-18.
[4] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel
[5] https://www.redhat.com/en/topics/api/what-is-a-rest-api
[6] https://firebase.google.com/docs/database
[7] https://developer.android.com/reference/android/content/SharedPreferences