**Foodle**

Software Design Specification

20.05.2022

**Name - ID**

Bekir Nazmi Görkem - 150118017

Berk Kırtay - 150118043

Burak Çağlayan - 150118027

Erkam Karaca - 150118021

Rasim Sadıkoğlu - 150118009

Prepared for

CSE3044 Software Engineering Term Project

Table of Contents

[1. Introduction 3](#_Toc104060055)

[1.1. Purpose 3](#_Toc104060056)

[1.2. Statement of scope 3](#_Toc104060057)

[1.3. Software context 3](#_Toc104060058)

[1.4. Definitions & Acronyms and Abbreviations 3](#_Toc104060059)

[2. Design Consideration 4](#_Toc104060060)

[2.1. Design Assumptions and Dependencies 4](#_Toc104060061)

[2.2. General Constraints 4](#_Toc104060062)

[2.3. System Environment 4](#_Toc104060063)

[2.4. Development Methods 5](#_Toc104060064)

[3. Architectural and component-level design 5](#_Toc104060065)

[3.1. System Structure 5](#_Toc104060066)

[3.1.1. Architecture diagram 5](#_Toc104060067)

[3.2. Description for Components 6](#_Toc104060068)

[3.2.1. Description For Database 6](#_Toc104060069)

[3.2.1.1. E-R Diagram 6](#_Toc104060070)

[3.2.1.2. Relational Model 6](#_Toc104060071)

[3.2.2. Description for Backend 7](#_Toc104060072)

[3.2.3. Description for Frontend 7](#_Toc104060073)

[3.3. Dynamic Behavior for Components 8](#_Toc104060074)

[3.3.1. Interaction Diagrams 9](#_Toc104060075)

[4. Restrictions, limitations, and constraints 12](#_Toc104060076)

[5. Conclusion 12](#_Toc104060077)

# Introduction

This section provides an overview of the entire requirement document. This document describes all data, functional and behavioral requirements for software.

## Purpose

The purpose of this document is to gather up all the features that comes into mind and give a detailed description of the requirements needed for this software. It will also explain which technologies should use together, what will be the interactions between user and app. This will help the software engineer to create the program in her/his mind, what to use and not to use during the creation phase and what to pay attention to while doing it.

## Statement of scope

The purpose of “Foodle” is to create an online environment that receives feedback from users and filters them accordingly. Foodle will be designed to help our users with their restaurant choices. There are several apps like Foodle, but they do not provide enough information. For instance, they use faked photos of food, outdated menu pricing. Thanks to the information provided by GPS and other users, our users will be able to easily access the places in the appropriate location and information about those places with the filters they choose. Then, like any other user, they can rate the restaurant by sharing their opinions about that place, photos of the food, prices, and other things.

## Software context

There is several software that are similar to our product. However, none of them are as free as ours. How this works is thanks to how much free space we provide to our users. Foodle's goal is not to advertise venues, our main goal is for users to easily get ideas from others and communicate with them. In this way, they will get the information they need before they go out.

## Definitions & Acronyms and Abbreviations

USER - Actor who uses application

FEEDBACK – Comments made by users

GPS - The abbreviation of Global Positioning System

UML – Unified modelling language

ADMOB - AdMob is a mobile advertising subsidiary

2FA – Two factor authentication

SDK – Software development kit

OTP – One time password

RAM – Random access memory

# Design Consideration

There are a lot of problems that we have to consider and solve before designing our app such as responsiveness of the application (views on different devices according to their screen sizes), RAM usage, effective using of our database, selecting a convenient color palette, to consider state management, a true object-oriented design supported with diagrams such as UML, flow diagrams and writing clean code.

## Design Assumptions and Dependencies

Our app has some dependencies such as RAM usage, storage usage (both local and database), battery usage. And there a few things that may go wrong such as being frozen of the app, not responding database or inefficient back-end systems, pixel overflow from the screen due to screen sizes of the different devices, having a user interface that does not responding users’ requirements and being hard to use of that user interface.

* Our app will be working on mobile devices (smart phones).
* Our app will run on Android and preferably on iOS soon.
* User should not be waiting more than 0.8 second on any interaction, UI must be easy to use so that user can use easily, widget layers must be correctly placed.
* Color palette may change in the future, new login and sign-up methods can be added in the future and we can integrate advertisements to our app with ADMob or we can make a section that contains sponsored restaurants, and we can make deals with restaurants, and we can provide some discount coupons to our users.

## General Constraints

* Some permissions of Android operating system
* RAM usage
* Local storage usage
* Battery usage
* Database storage usage
* 2FA verification
* Security requirements of user data
* Performance requirements
* Network quality

## System Environment

**Front-end:**

* Flutter SDK
* Android Studio
* Android SDK
* Android Emulator
* VSCode
* GetX package (to manage state management issues)
* Git Version Control System
* GitHub

**Back-end:**

* VSCode
* Node.js
* Git Version Control System
* GitHub
* Postman API
* Typescript

**Database:**

* PostgreSQL
* RedisInsight
* PgAdmin

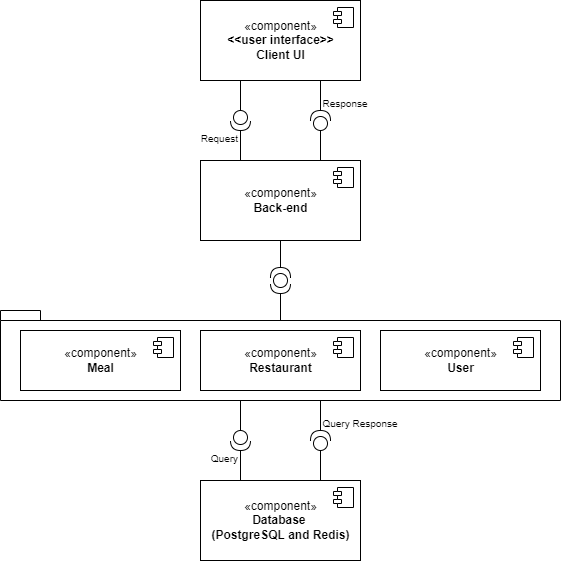
## Development Methods

We have used the waterfall software design model to implement this project, and we have written our code faithfully to the object-oriented design theory and Clean Code principles (such as we have separated frequently used UI elements as separate widgets and we placed them into components folder of every single page style). Sometimes we did pair-programming in order to accelerate our group work.

# Architectural and component-level design

## System Structure

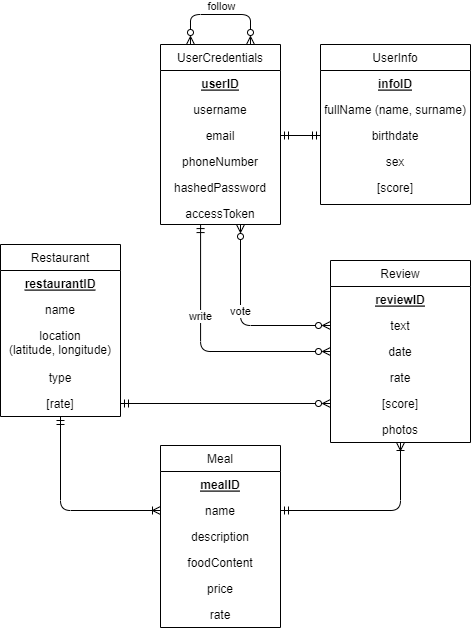
### Architecture diagram



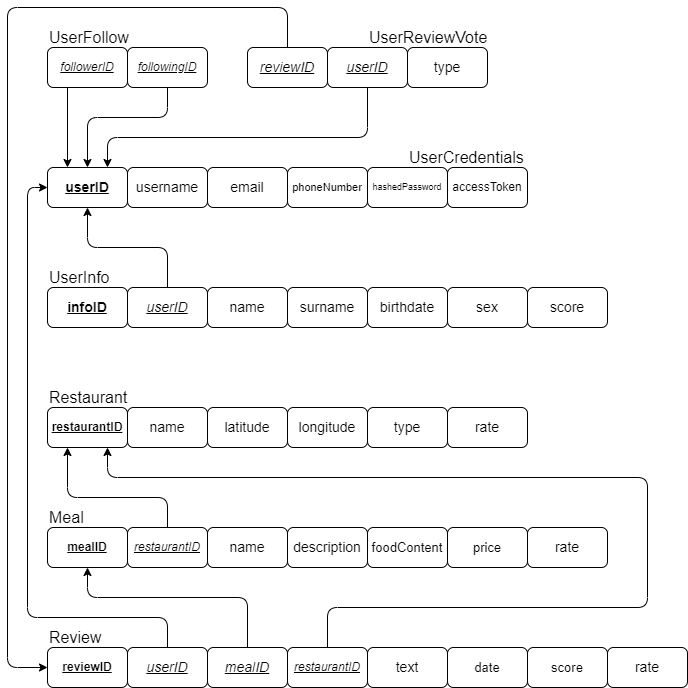
## Description for Components

### Description For Database

### E-R Diagram



### Relational Model



### Description for Backend

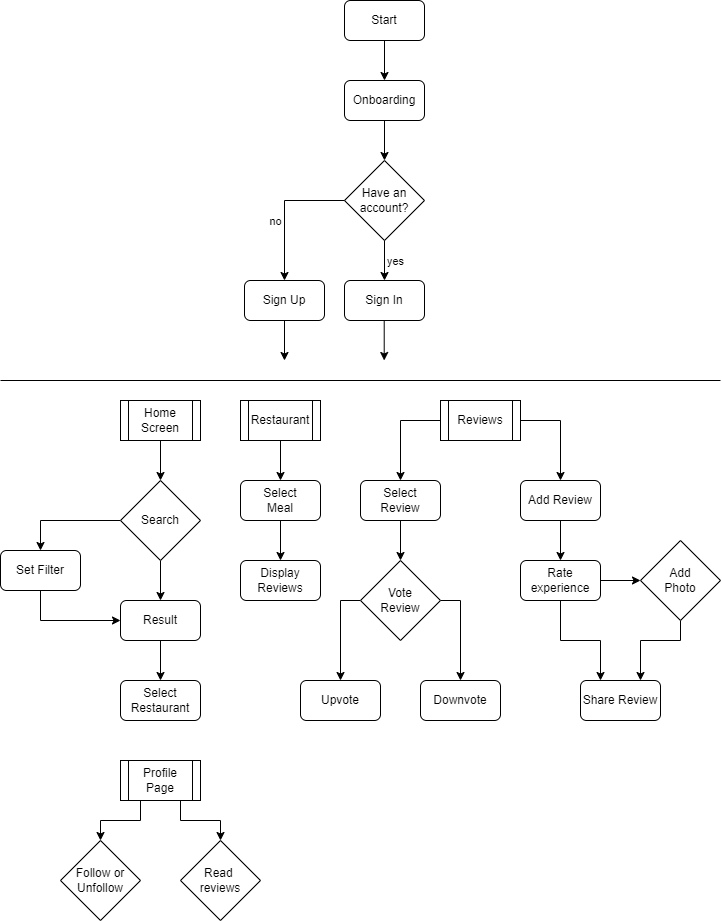
* + - 1. UML – Class Diagram

Diagram

Description automatically generated

### Description for Frontend

* + - 1. User Flow Diagram



## Dynamic Behavior for Components

Our app has three system components, and they have three main dynamic behaviors. All the three components must interact with each other to keep the functionalities of the requirements working.

* User Authorization:

This section is responsible for all authorization and authentication transactions. Our general approach for user authorization uses 2fa and user sessions. If a user enters correct credentials and authentication code, then the user will be able to use the functionalities of the program. Responsibilities of the system components and the interaction processes (i.e., register, login, and logout) are shown in the diagram.

* User Profile Interaction:

This is for all user-social processes such as, searching for other users, following, or unfollowing people or showing a user profile in the app. This aspect is significant for user interactions in our app.

* User Restaurant Interaction:

This section handles restaurant processes. It can provide users customized restaurants info based on their filters and location info on their smartphones. Users can select a restaurant, view its meals and scores as well as the reviews by other users. A user can also write a review for a meal at a restaurant he has gone to. Furthermore, users can vote reviews and those votes will give an overall score for the restaurants and their meals. With this way, unrelated or dishonest reviews will not be seen as trustable and will be shown rarely.

### Interaction Diagrams

* User Authorization Diagram:

Diagram

Description automatically generated

* User Profile Interaction Diagram:

A picture containing diagram

Description automatically generated

* User Restaurant Interaction Diagram:

A picture containing engineering drawing

Description automatically generated

# Restrictions, limitations, and constraints

Software implementation style is very crucial to us since our team members should be able to understand and successfully contribute to the project. So, coding style, formatting, continuous code refactoring is the main issue for this section.

Software design should be consistent with the requirements and reflect what we want to accomplish. We should design our system components in a realistic way. So that we will not have issues with implementation afterwards. For example, we should implement database server based on how we will implement back-end server features. If user data is to be indexed for faster retrieval, we should do it so.

Software execution and testing can show how much stress our app can stand, or it can show us if our app can function under minimal system (hardware) requirements. We should make sure that our app works and function properly under certain conditions.

# Conclusion

This document will ensure that our app will be implemented accordingly and will work and function as this document specifies. This document consists of planning, design constraints and limitations, designed system components and their behaviors, interactions between components and how those interactions work as they should function according to our apps requirements.

We have also stated our working style as Waterfall software development method, and we have stated our collaborative work as pair programming. We have supported our document with:

* Interaction Diagrams
* Flow Diagrams
* UML Diagrams
* User Class Diagrams
* E-R Diagrams
* Architecture Diagrams

Also, we have mentioned our tools that helps us on development and testing process’. We have mentioned design assumptions and dependencies above at section 2 and we have mentioned some future implementations that may be implemented according to our progress.