Software Requirement Specification

# Introduction

## Aim of the document

The document aims to present the software system PickyEaters.

Each of the following numbered subparagraphs will point out a different characteristic of the system, starting from a general overview of the defined system, its hardware and software requirements, and its competitors – describing both the pros and cons of the system, compared to the system competitors.

Later in the document, each paragraph will present some functionalities of the system, starting with three user stories, then three functional requirements, and eventually the use case diagram.

## Overview of the defined system

PickyEater is a software system developed for all the picky eaters who struggle to find something to eat or drink when not eating at home.

The system expects three types of users:

* the base user, who will be called Pickie, interested in finding places to eat, by themselves or with their picky friends;
* the business owner user, who will be called Restaurateur, interested in promoting their business – a restaurant, pub, café, or similar;
* the system administrator, who will be called Administrator, whose job is to manage Pickies’ reports.

The Pickie will be able to find places where they can find meals that suit their diet, by declaring ingredients they don’t like and their dietary regimen (e.g.: halal, kosher, vegetarian, vegan, pregnant, etc…).  
They will be able to create Pickies groups so that both themselves and their friends’ needs are satisfied.   
Pickies will be provided with the possibility to rate dishes and report any incorrect information regarding restaurants (e.g.: wrong address, non-existent restaurant, wrong phone number, etc…) and their menus (e.g.: ingredients not declared in a dish description, ingredients declared but not used in a dish, etc…).  
Moreover, if a disliked ingredient is not listed by the system, they may suggest the Administrator to add it.

On the other hand, the Restaurateur shall register their business information into the system, providing also their menu, containing each dish and its ingredients. They are also responsible for maintaining updated menu information. The Restaurateur will be able to view statistics referring to their business and customers.

Eventually, the Administrator’s role is to be the human layer between the system and Pickies, checking and managing reports and suggestions from Pickies.

## HW e SW requirements

The system must run on a machine with a 64-bit processor.

Connection to the internet must be provided.

## Related systems, Pros and Cons

PickyEaters is a unique system as it fills a gap in the food service market that is ignored by related systems, such as Tripadvisor, JustEat, and Glovo.

It is similar to Tripadvisor as its purpose is not to order and deliver food, but to show places where to eat out.  
Yet it is also similar to JustEat and Glovo as it only concerns restaurants, pubs, cafés, and similars – not hotels, flights, etc… –, and offers an insight into a restaurant menu, showing dishes.

The potential of the idea behind PickyEaters cannot, unfortunately, be fulfilled because of the obvious lack of a corporate-like structure, funds, expertise, and time. This leads to a long list of cons that could, however, be easily mostly crossed out if only the system were developed in a proper environment; that list includes:

* Strong, prominent competitors;
* Simplistic and not captivating design of UIs;
* Simplistic functionalities;
* Underdevelopment of cybersecurity measures;
* Possibility of noncompliance with legal requirements;
* Lack of cookies technology.

As it comes to exploring the pros, it should be highlighted that:

* The system is wholly designed and developed ex novo and by only two people: the software development process follows disciplined, uniform, and recent practices;
* There is room for adding, refining, and bettering functionalities;
* In the current market, where inclusion and personalization are being more and more seen as priorities, no other similar system offers the possibility to customize and filter results at such a fine grain.

# User Stories

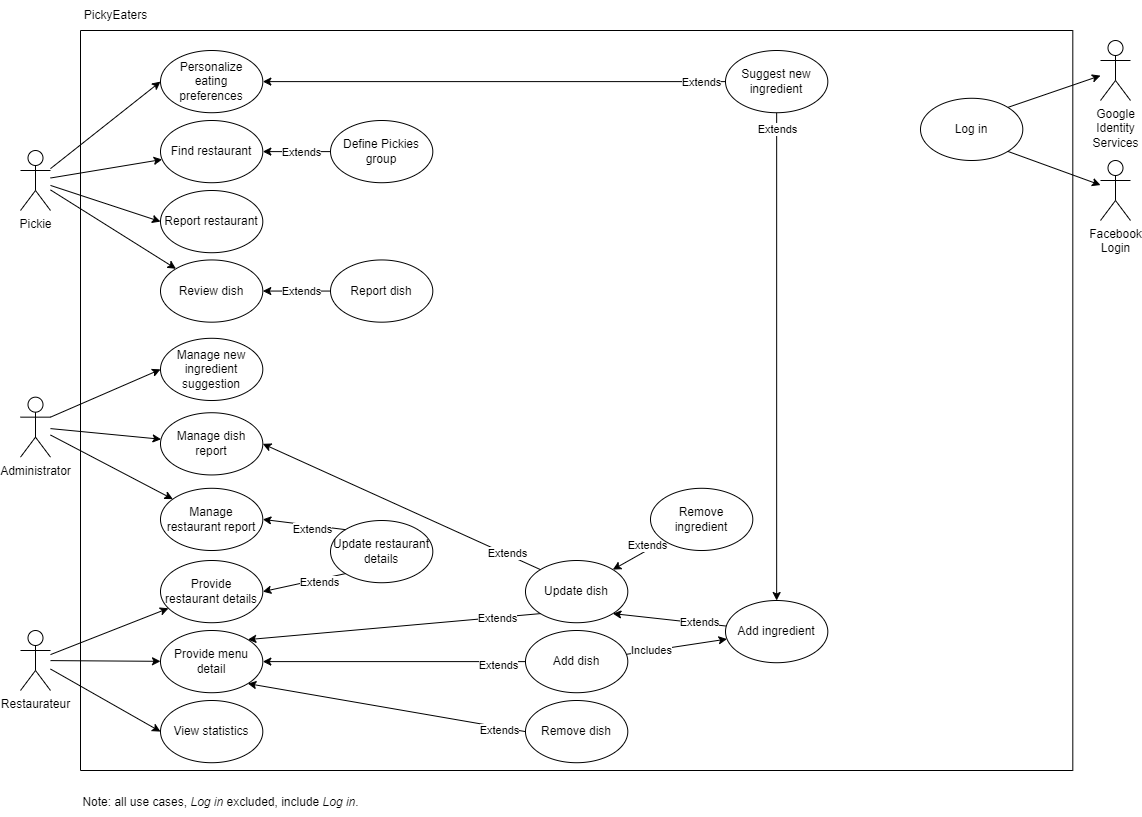
* 1. As a Pickie, I want to specify ingredients that I don’t like, so that I can find a place to eat.
  2. As a Pickie, I want to create a friend group, so that I can find a place to eat with all my friends.
  3. As an Administrator, I want to manage new ingredients suggestions, so that I can do my job.
  4. As a Restaurateur, I want to add a new dish to my menu, so that my menu is updated.
  5. As a Restaurateur, I want to remove an old dish from my menu, so that my menu is updated.
  6. As a Restaurateur, I want to view statistics regarding my business, so that I can plan strategies to increase my turnover.

# Functional Requirements

* 1. The system shall provide the Pickie with the selection of disliked ingredients.
  2. The system shall provide the Restaurateur with the selection of ingredients for a dish.
  3. The system shall provide the Restaurateur with the deletion of a dish from the menu.
  4. The system shall show the Administrator Pickies’ reports.
  5. The system shall provide the report of a dish ingredient.
  6. The system shall provide the report of a restaurant.

# Use Cases

## Overview Diagram



## Internal Steps

Name: **Add dish**

1. The Restaurateur requests to construct a schedule.
2. The system prepares a blank schedule form.
3. The system gets dish types from the database.
4. The Restaurateur selects one dish type, types the dish name into the system, and a*dds an ingredient*\* per time.
5. After each ingredient selection, the system presents the ongoing list of selected ingredients.
6. The Restaurateur indicates the schedule is complete.
7. The system saves the schedule into the database.
8. The system presents the successfulness of the schedule-saving operation.

Extensions:

1a. *Restaurateur already has a schedule*: The system brings up the current version of the Restaurateur’s schedule instead of creating new ones.   
3a. *Database does not respond*: The system notifies the Restaurateur and terminates the use case.   
4a. *More than one dish type selected*: The system removes the first dish type selection, selects only the last dish type selected by the Restaurateur and presents said selection.  
5a. *Same ingredient selected more than once*: The system notifies the Restaurateur and removes the double from the dish ingredient list.  
6a. *No ingredient selected or no dish type selected or no dish name defined or there is already a dish with the same name in the Menu*: The system notifies the Restaurateur.   
7a. *Database does not respond*: The system notifies the Restaurateur and terminates the use case.

A*dds an ingredient*\*: It refers to the use case *Add ingredient*.

Name: **Add dish**

1. The Restaurateur requests to construct a schedule.
2. The system prepares a blank schedule form.
3. The system gets dish types from the database.
4. The Restaurateur selects one dish type, types the dish name into the system, and a*dds an ingredient*\* until there is an ingredient to be added to the dish.
5. After each ingredient selection, the system presents the ongoing list of selected ingredients.
6. The Restaurateur indicates the schedule is complete.
7. The system saves the schedule into the database.
8. The system presents the successfulness of the schedule-saving operation.

Extensions:

1a. *Restaurateur already has a schedule*: The system brings up the current version of the Restaurateur’s schedule instead of creating new ones.   
3a. *Database does not respond*: The system notifies the Restaurateur and terminates the use case.   
4a. *More than one dish type selected*: The system removes the first dish type selection, selects only the last dish type selected by the Restaurateur and presents said selection.  
5a. *Same ingredient selected more than once*: The system notifies the Restaurateur and removes the double from the dish ingredient list.  
6a. *No ingredient selected or no dish type selected or no dish name defined or there is already a dish with the same name in the Menu*: The system notifies the Restaurateur.   
7a. *Database does not respond*: The system notifies the Restaurateur and terminates the use case.

A*dds an ingredient*\*: It refers to the use case *Add ingredient*.