**Purpose**

The results of the requirements elicitation and the analysis activities are documented in the Requirements Analysis Document (RAD). This document completely describes the system in terms of functional and nonfunctional requirements and serves as a contractual basis between the client and the developers.

**Audience**

The audience for the RAD includes the client, the end users, the project manager, and the developers.

**Table of Contents**

1. Introduction 3

1.1 Purpose of the system 3

1.2 Scope of the system 3

1.3 Objectives and success criteria of the project 3

1.4 Definitions, acronyms, and abbreviations 4

1.5 References 4

1.6 Overview 4

2. Current system 4

3. Proposed system 4

3.1 Overview 4

3.2 Functional requirements 5

3.3 Nonfunctional requirements 5

3.3.1 Usability 6

3.3.2 Reliability 6

3.3.3 Performance 6

3.3.4 Supportability 6

3.3.5 Implementation Requirements 6

3.3.6 Interface Requirements 6

3.3.7 Packaging Requirements 6

3.4 System models 6

3.4.1 Scenarios 7

3.4.2 Use case model 7

3.4.3 Object model 7

3.4.4 Dynamic model 7

3.4.5 User interface 7

4. Glossary 8

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Author | Date | Changes |
|  | Ludwig Gröber | 08.07.2022 | Everything |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

*The purpose of the Introduction is to provide a brief overview of the function of the system and the reasons for its development, its scope, and references to the development context. The introduction also includes the objectives and success criteria of the project.*

## Purpose of the system

The purpose of the system is to give restaurant visitors the possibility to find, filter and reserve their table in a restaurant in advance, so that they have a fixed place when they arrive. The specifics of the system are discussed in the following Requirements Analysis Document.

## Scope of the system

The user gets a map where he can filter for restaurants by different criteria. The system shows a list of restaurants where a visitor can reserve tables. The user can choose a restaurant and reserve a table on a specified timeslot for a specified number of people. The application communicates with the server system and conforms to the usability requirements. The scope of the project besides that, is to support a multitude of users searching and finding open tables in a local area around their city.

## Objectives and success criteria of the project

The Objectives and Success criteria of the project are:

1. The user can search for restaurants on a list and on a map that displays up to 50 restaurants.
2. In a more detailed view, the user can see pictures, ratings and comments of the restaurant as well as opening times and a link to the website.
3. The user can filter the results by the restaurant type, the prize category, by distance around a certain location, by the average rating and by free time slots for reservations for specified dates and number of visitors.
4. In a different screen, a user can see the times when he can reserve a table in the chosen restaurant. After clicking on the time, the user sees an overview of all tables in the restaurant. He can choose the exact table of the free ones in the overview and thus reserve the table for the specified number of visitors.
5. When the user reserves a table, an event in the local calendar is created for the reservation.
6. A user is reminded about a reservation one day before the actual date of the reservation and must confirm it until latest 12 hours before the actual date. If the user does not confirm, his reservation is cancelled automatically.
7. A user can cancel his reservation at any time up to two twelve hours before the actual date of the reservation. After the confirmation, the user cannot cancel the reservation anymore.
8. The system should be intuitive to use, and the user interface should be easy to understand. Simple interactions should be completed in less than three clicks. Complex interactions should be completed in less than six clicks.
9. The design of the system should conform to the typical usability guidelines such as Nielsen’s usability heuristics.
10. A server subsystem with a couple of services must be used in the system.

The constraints of the project are that the version control system must be git and the Source Code Documentation must be in HTML format and the server system must use the Spring Boot framework. Other than that, the system should run on all desktop operation systems (Windows, macOS, Unix), either as Java or as browser-based application which communicates with the Spring Boot server application.

If all previously stated requirements are met, the project is a success.

## Definitions, acronyms, and abbreviations

None.

## References

Please find attached to the project a Use-Case-Diagram, a Class-Diagram as well as two Scenarios and the corresponding Communication Diagrams. Nielsen’s usability heuristics.

## Overview

The reason for the system development is to get the grade bonus in EIST, where we could form self-assigned teams to do a project. The functional reason is to give a customer the opportunity to find and book tables in a restaurant in advance via a web-browser application.

# Current system

*This section describes the current state of affairs. If the new system will replace an existing system, this section describes the functionality and the problems of the current system.*

Currently, there is no such system available and therefore there is no functionality provided by previous systems.

# Proposed system

*The third section documents the requirements elicitation and the analysis model of the new system.*

## Overview

*The overview presents a functional overview of the system.*

The system should allow the user to search for restaurants on a list and on a map. The user can see pictures, ratings and comments of the restaurant as well as opening times and a link to the website. He can filter the results to fit his needs. A user can see the times when he can reserve a table in the chosen restaurant and the user sees an overview of all tables in the restaurant. He can choose a table to reserve. When a user reserves a table, an event in the calendar is created. A user is reminded about a reservation one day before the actual date of the reservation and must confirm it or his reservation is cancelled. A user can cancel his reservation at any time up to two twelve hours before the actual date of the reservation. After the confirmation, the user cannot cancel the reservation anymore.

## Functional requirements

*Functional requirements describe the high-level functionality of the system. This section list all functional requirements and additionally presents the dependencies between them.*

FR1: Search for restaurants: The user can search for restaurants on a list and on a map that displays up to 50 restaurants. This functional requirement depends on the location the user is located in and therefore the map is adjusted accordingly.

FR2: See restaurants details: The user can see pictures, ratings and comments of the restaurant as well as opening times and a link to the website. This requirements depends on the information provided by the database if certain information is missing, it cannot be displayed.

FR3: Filter search results: He can filter the results by the restaurant type, the prize category, by distance around a certain location, by the average rating and by free time slots for reservations for specified dates and number of visitors. This also depends on the availability of such information, if the restaurant does not have a rating, this an not be filtered for.

FR4: Reserve table: A user can see the times when he can reserve a table in the chosen restaurant. After clicking on the time, the user sees an overview of all tables in the restaurant. He can choose the exact table the free one in the overview and thus reserve the table for the specified number of visitors. A user can of course only reserve a table, if it is still available. If there a no available restaurants from FR3 or FR2, the it cannot be reserved.

FR5: Save calendar event: When the user reserves a table, an event in the local calendar is created for the reservation. This FR depends on the completion of FR4.

FR6: Confirm reservation: A user is reminded about a reservation one day before the actual date of the reservation and must confirm it until latest 12 hours before the actual date. If the user does not confirm, his reservation is cancelled automatically. This depends on the completion of FR5.

FR7: Cancel reservation: A user can cancel his reservation at any time up to two twelve hours before the actual date of the reservation. After the confirmation (see FR5), the user cannot cancel the reservation anymore. This FR depends on FR5 and FR6, if they have been completed or not, respectively.

## Nonfunctional requirements

*Nonfunctional requirements describe user-level requirements that are not directly related to functionality. This includes usability, reliability, performance, supportability, implementation, interface, operational, packaging, and legal requirements. The section list all these non-functional requirements and additionally presents the dependencies between them.*

### Usability

NFR1: Usability: The system should be intuitive to use, and the user interface should be easy to understand. Simple interactions should be completed in less than three clicks. Complex interactions should be completed in less than six clicks. Simple interactions are view based interactions, while complex interactions are deeper functionality such as making a reservation.

NFR2: Conformance to guidelines: The design of the system should conform to the typical usability guidelines such as Nielsen’s usability heuristics.

### Reliability

There are no specific reliability requirements other than in Nielsen’s usability heuristics in NFR2.

### Performance

There is no performance requirement specified, other that the system should operate in a reasonably fast time. This is included in Nielsen’s usability heuristics in NFR2.

### Supportability

A supportability requirement is given by the constraint that the system should run on all desktop operation systems (Windows, macOS, Unix), either as Java or as browser-based application which communicates with the Spring Boot server application.

### Implementation Requirements

There is a implementation requirement specified in the additional constraints, that the server system must use the Spring Boot framework.

The other one is the NFR3: Server system: A server subsystem with a couple of services must be used in the system.

Another implementation requirement is, that the programming language java has to be used.

### Interface Requirements

NFR1 says that simple interactions should be completed in less than three clicks. Complex interactions should be completed in less than six clicks. This is the only interface requirement given.

### Packaging Requirements

Some packaging and formatting requirements are specified as additional constraints: The version control system must be git and Source Code Documentation must be in HTML format.

### Legal Requirements

There are no legal requirements or licensing problems, all parts of the project are our own original code or free to use libraries.

## System models

*The System models include scenarios, use cases, object model, and dynamic models for the system. This section should contain the complete functional specification, including mock-ups, paper-based prototypes or storyboards illustrating the user interface of the system and navigational paths representing the sequence of screens.*

### Scenarios

We developed two scenarios where the user either wants to find a restaurant he wants to go to (Scenario 2 attached to the project) or books a table at a restaurant he has already selected (Scenario 1 attached to the project). From these Scenarios we then developed two communication diagrams for the dynamic model (see below 3.4.4 dynamic model).

Besides the scenarios we developed, there was one given Scenario on which we based ours:

“Hans lives in Munich and knows that all the restaurants are really crowded over the weekend and getting a reservation spontaneously is almost impossible. He wants to visit an Italian restaurant on Saturday evening together with three friends and opens the Reservation system. He searches for Italian restaurants and sees different choices on the map. He also sees ratings as well as price categories. He chooses the Nero Pizza & Grill in Rumfordstraße 34 to see more details. He likes the pictures and reviews of the restaurant and clicks on the reservation button to see whether the restaurant has free tables on Saturday evening. The system shows a timetable for the upcoming seven days where unavailable times are greyed out. Hans chooses Saturday 17:00 and sees a layout of all the tables in the restaurant. Not yet reserved tables are selectable. Reserved tables are greyed out. Hans chooses the table next to the window for four people and confirms his choice. The system automatically creates a calendar event in Hans’s calendar. On Friday, Hans receives a reminder about the reservation and sees that he must confirm the table again, so the reservation does not get canceled. On Saturday, Hans receives a notification at 16:00. He opens the map to navigate to the restaurant.”

### Use case model

To understand the dynamics of the project better, we developed a use case model as one of our first tasks in the project, to get a better feeling for the requirements and for how we want to structure and set up the project later. You can find the use-case model attached to our project in the docs section.

### Object model

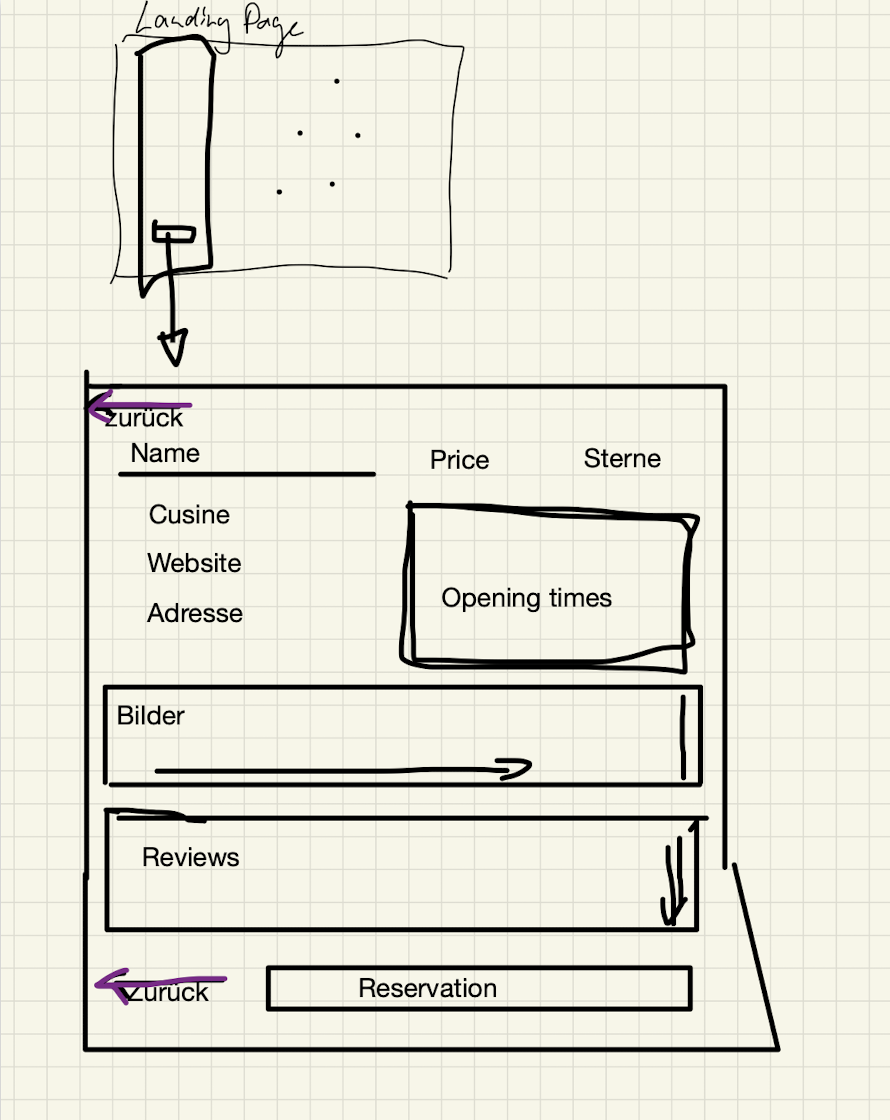
For the Object model we have created a class diagram where each class we used is specified and its functionality in terms of methods and attributes is represented. The Object model evolved throughout the project, as we thought of new functionality (such as filtering for opening times right in the beginning) and we continuously added new methods and attributes where needed.

### Dynamic model

Our dynamic model consists of two communication diagrams which were modeled according to the two scenarios we developed. In the diagrams we modeled the different messages sent between classes in the process described in the Scenarios. Please find the diagrams attached to our project.

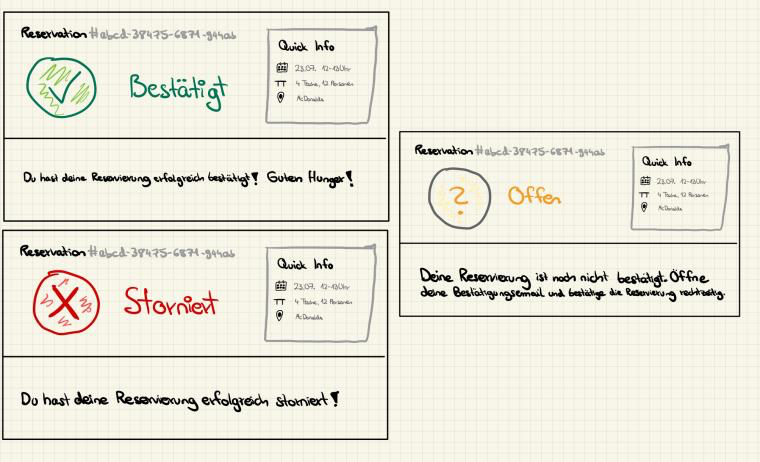
### User interface

Sometimes, to foster a common understanding of an interface, you need to sketch it. And therefore, I’m happy to present my own original sketches of the landing page and the restaurant detail view:



From the design we developed there, we built on in the next sprint based on current design compatibility.

Another sketch for the confirm and cancel interface, we built in one of the sprints:



# Glossary

*A glossary of important terms used in the project and in the system model ensures consistency in the specification and a common understanding of terms used by the client.*

All the terms in the project are commonly used so there is no need for further explanation.