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

Document History

Rev.	Author	Date	Changes
1	Simon Okutan	16.07.2022	

1. 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.

1.1 Purpose of the system

Reservation-Bear is a reservation system for searching restaurants with tables available at a certain time.

1.2 Scope of the system

- System has a client and a server application
- Client is written with ReactJS (written in Type Script) and Server with SpringBoot (Kotlin)
- Server provides the logic behind the reservation process and client the UI
- [Könnte hier jemand nochmal gucken, ob dass das geforderte ist?]

1.3 Objectives and success criteria of the project

The objectives of the Reservation-Bear project are to:

- Provide a simple solution for booking tables at restaurants and manage the reservations

The project Reservation-Bear is accepted if at least the following success criteria are satisfied:

- The Reservation-Bear executable is available on July 23, 2022, and satisfies the functional requirements specified in section 3.2.
- The Reservation-Bear API should be REST-Full
- The Reservation-Bear server is programmed in the Spring-Boot framework
- The Reservation-Bear application can be compiled just with “-/ gradle boot JAR”
- All nonfunctional requirements from section 3.3 are satisfied.

1.4 Definitions, acronyms, and abbreviations

Reservation-Bear:

Name of the platform: Composed of the team name of the developers "Ice Bears" and the function of the site to be able to reserve tables.

1.5 References

GitHub Project:
https://github.com/Eistbaren

Wiki:
https://wiki.itsblue.de/collection/eistbaren-zBWxM2RXIP

Reservation-Bear:
https://reservation-bear.de/

Bumbers RAD:
https://ase.in.tum.de/lehrstuhl_1/component/content/article/43-books/264-oose-bumpers-requirementsanalysisdocument

1.6 Overview

2. 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.

The Reservation-Bear project is a "greenfield engineering" project. There is no current system to be replaced, but FRs and NFRs are specified (although not met by any previous system).

3. Proposed system

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

3.1 Overview

The overview presents a functional overview of the system.

Reservation-Bear is mainly designed to provide a quick and easy way to reserve tables in restaurants. For this purpose, it offers various restaurants, which can be filtered according to different criteria (see FRs). Also, short insights about the different restaurants can be displayed, as well as comments and ratings. After selecting a restaurant, free tables are displayed, which can then be selected. Finally, the reservation must be confirmed via email.

3.2 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.

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.

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.

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.

FR5: Save calendar event: When the user reserves a table, an event in the local calendar is created for the reservation.

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.

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.

3.3 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.

3.3.1 Usability

NFR1: 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.

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

3.3.2 Reliability

3.3.3 Performance

3.3.4 Supportability

3.3.5 Implementation Requirements

NFR3: Server system: A server subsystem with a couple of services must be used in the system

3.3.6 Interface Requirements

3.3.7 Packaging Requirements

3.3.8 Legal Requirements

3.4 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.

3.4.1 Scenarios

Formalized Scenario 1:

Name: Find Restaurant
Participants: Alice: Hungry customer
Flow of events:

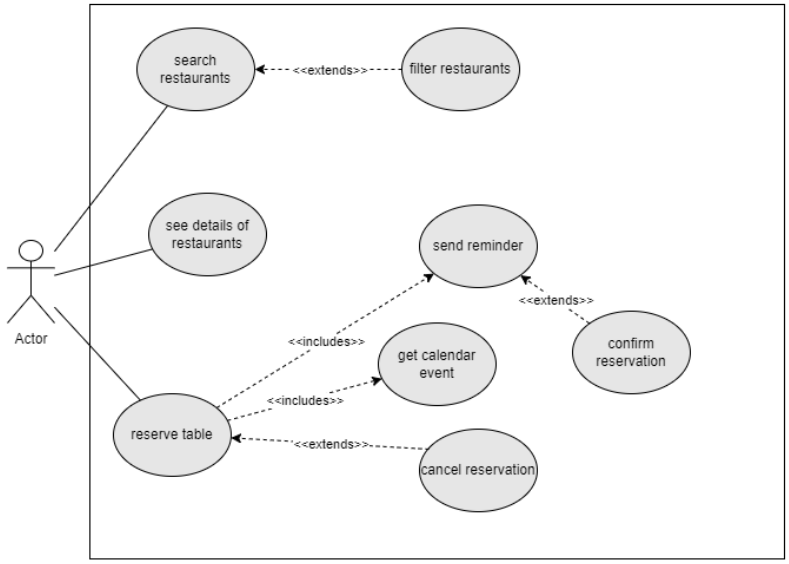
1. Opens to the website https://reservation-bear.de
2. Website displays restaurant suggestions and an option to search
3. Alice searches for an Italian restaurant
4. Website displays all restaurants which are flagged as Italian
5. Alice chooses a restaurant and clicks on it to see more details
6. The website displays Pictures, ratings and comments of the restaurant as well as opening times and a link to the website

Formalized Scenario 2:

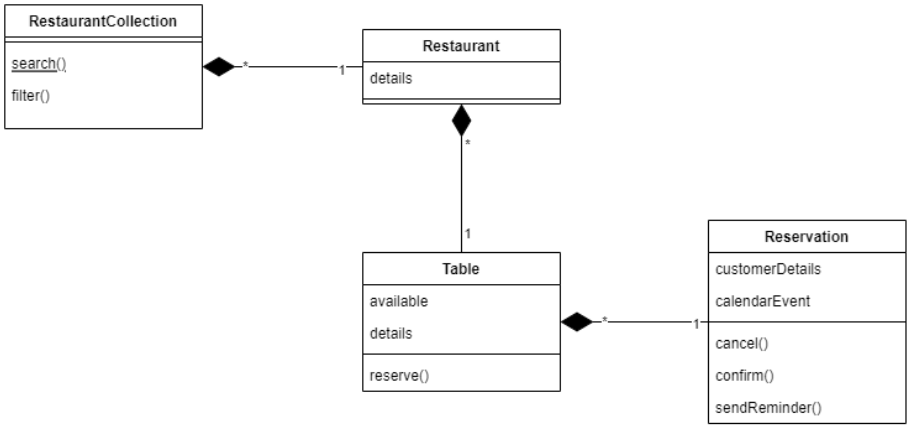
Name: make reservation
Participants: Bob: Hungry customer
Flow of Events:

1. Bob clicks on “reserve”
2. Website shows the layout of the restaurant and an input to further specify the time for the reservation
3. Bob selects a timeslot between 19:00 and 20:30
4. Website greys out all tables that are unavailable in that time slot.
5. Bob selects a table with a view towards a window.
6. Webiste show a form with a Customer Name and E-mail field.
7. Bob fills out the form and proceeds to accept the booking.

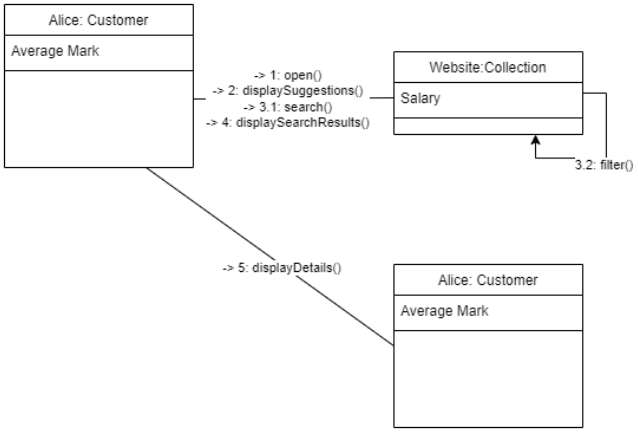
3.4.2 Use case model



3.4.3 Object model



3.4.4 Dynamic model



3.4.5 User interface

The UI of Reservation-Bear was sketched in Figma. The link to this sketch:

- https://www.figma.com/file/CfM3TQuPRtrRDGpERGXOnV/Reservation-Bear-UI-Design
- https://www.figma.com/proto/CfM3TQuPRtrRDGpERGXOnV/Reservation-Bear-UI-Design?kind=&node-id=67%3A3586&page-id=0%3A1&scaling=contain&starting-point-node-id=23%3A3276

More details can be found on our wiki page at the following link:

- https://wiki.itsblue.de/share/4f4f870a-d29e-4680-b36c-b2663dfca8bd

4. 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.

Spring-Boot-Framework: A framework is a pre-written code base that can be used to allow easy coding for complex functions. Spring Boot us a JAVA Server Framwork that helps to process REST-REST-Request: Read more: https://spring.io/projects/spring-boot

REST: "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." (Source: https://www.redhat.com/en/topics/api/what-is-a-rest-api#:~:text=A%20REST%20API%20(also%20known,by%20computer%20scientist%20Roy%20Fielding.)