

R-Booking

# Requirements Specification and Analysis

Version 1.0

19.10.2018

Mert Pekduzgun

Abdulwahab A. Sani

Alper Cevik

Burak Sonmez

Prepared for

SE301 Software Engineering



## 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 .....	4
1.4.Definitions, Acronyms, and Abbreviations.....	4
1.5.Overview .....	5
2. Current System .....	6
3. Proposed System .....	6
3.1.Overview .....	7
3.2.Functional Requirements .....	8
3.3.Nonfunctional Requirements .....	9
Usability- We made our system very usable and to measure this, making a reservation only takes five clicks. And this is the main functionality of our system. So just about anyone with basic knowledge of technology can use our system. ....	9
Reliability- Our system is expected to run 100% of the time so that our users can be able to reserve at any time. ....	9
Performance- System performance needs to be optimal, this includes search results times, loading restaurant pages etc.. So our users wont need to wait too long for system to load. We also want to support at least 300 parallel users. ....	9
Supportability- System needs to be modified easily without difficulties. ....	9
Implementation- System shall be developed on Android App Studio. This means the programming language will be mainly based on Java. Our web service will be supported by Spring MVC Framework. Our database will be managed via MySQLWorkbench.....	9
Interface- Our interface will be an android application interface. ....	9
Packaging- No packaging requirements. ....	9
Legal- The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and no infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software. ....	9
3.4.System Models.....	9
Scenarios .....	10
Use case model .....	10

Object model.....	10
Dynamic model.....	10
User interface—navigational paths and screen mock-ups .....	11
3.5.Project Schedule.....	11
4. Glossary.....	11
5. References.....	11

## **REQUIREMENTS ANALYSIS DOCUMENT[1]**

### **1. Introduction**

The purpose of this system is to provide users with an easy way of being able to reserve space in a restaurant right from their mobile phone so that they are assured of being able to get a place at any restaurant of their choice.

#### **1.1.Purpose of the System**

The driving reason behind developing this system is to give restaurant goers assurance of having space/table when they go out to eat. More times than not, people go to a restaurant usually in a group and unfortunately there is no space to accommodate them all as such, our system will let them know for sure whether they will get a table or not which also saves their time and energy. It also gives the users valuable information about the restaurant and also importantly- choices. There is an expansive list that the user can choose from and also see the ratings.

#### **1.2.Scope of the System**

Our system will allow the users to reserve a place at a restaurant of their choosing prior to their visit to the restaurant. The user first needs to sign up, then after logging in they can view the variety of restaurants available/open and select the number of people that will be coming. This service is very versatile and as such it can be used by a people ranging from government officials down to students at a college.

Our system will have three users each with a different level of authority. They are the admin, the manager and the customer. The admin is the one with the most authority and he is responsible for adding new restaurants (or disabling them) from the system.

He can also add new managers and customers to the system and just as well disable their accounts. Maintaining the system is also his responsibility.

The manager is the one who will send a request to the admin to add a new restaurant and can also send a disable restaurant request. The manager can update restaurant info such as location (if the restaurant moved elsewhere), restaurant name, available capacity etc..

Finally, the customer is the one with least authority but has the most functionalities. The customer can make a reservation and select the number of people that they want to reserve for. In addition to making the reservation, a customer can of course have a change of mind and cancel their reservation. The customer can also leave an optional and rating which will be useful in giving other users insight about the restaurant's quality. They can also update their information such as email address, favourite restaurant etc. The customer will be able to search for a restaurant and view it which is part of the process for making their reservation.

### **1.3.Objectives and Success Criteria of the Project**

Our objective is to give our users the peace of mind and assurance of knowing they are guaranteed a place at the restaurant they want to visit. By doing this we also want to save their time so they don't go to a restaurant and its full and they have to choose another place again which could be full again. This can get frustrating. We believe that our system can help assist with this problem. We shall gauge the success of the project by:

- positive reviews
- number of users/downloads
- improving people's time efficiency
- system popularity
- improved planning for large groups of people

### **1.4.Definitions, Acronyms, and Abbreviations**

- RAD: Requirement Analysis Document.
- Admin: System administrator who manages users and managers and also maintains and updates the system.

- System: the software that is to be developed.
- Disable: freezes activity for the account or restaurant that is requesting to be disabled. Action cannot be undone.

## **1.5.Overview**

This RAD shall compare the current system that is being used which shall be replaced with the to-be developed system. It shall look at the upsides and benefits of replacing the old system. It also contains the system models which include functional and non-functional requirements which shall be described in detail and explained, there is also a use case description and scenarios that were created. Also included is the Gantt Chart which a schedule for when the project is expected to be fully developed along with milestones. We included a glossary as well so as to define words that may seem foreign to those outside of the project team.

The organisational structure of the RAD is designed from the ground up so, we describe the things included from the basic descriptions of our system to the more complex system models and functionalities of the system. So we started with Introduction where we specify and familiarise the reader with what the project is all about and what it will do. The purpose of the system and the objectives of the system and what it is expected to achieve. Furthermore, we give an overview of the project which tells about what this RAD contains.

Next up is the current system section where we talk about the system that is currently being used by a majority number of people and this is the system we intend to replace and ultimately improve. We mention the things that make this current system inefficient and how is it time consuming and leaves room for uncertainty such as visiting a restaurant and it is full and so people have to search for a another place.

In proposed system section we speak on how the to-be developed system shall be an improvement on the old one. We discuss the advantages and why it is the right step forward. We mentioned that the current system is time consuming and uncertain, in the proposed system, this problem is solved by allowing the customer to reserve a table/ space for a number of people. If you reserve the space, you are guaranteed a table and this saves time and makes it easier for people to choose where to go. The proposed system will also be made to be as user friendly as possible so that customers of all ages and computer literacy can use it with minimal difficulty.

In overview section, we discuss the functionalities of the system which is what the system is going to be capable of and what actions it supports.

In functional requirements section, we discussed how the different users of our system -the manager, admin and customer- interact with the system and their different levels of authority. We make mention of what the Restaurant Booking system is capable of doing and its features how the users have different but sometimes similar functions.

In non-functional requirements section, we talk about functions that are not necessarily defining the behaviour of our system but rather, they are criteria for how the system should be judged or rated. These criteria include: usability, performance, reliability or supportability, implementation, interface, legal and packaging. We shall discuss these in more detail in later sections of the RAD.

In system models section, we talk about the scenarios, use cases, object model and dynamic model of the system. We created scenarios for manager, admin and the customer as our actors. Each scenario discusses how the actor shall partake in the use case and achieve the task at hand. For the use cases we gave full descriptions including, entry conditions, exit conditions, flow of events etc. The use case is an action to be performed by the specified actor within the system.

Lastly, in the glossary section we defined terms and other words or phrases needed to fully understand the RAD with minimal fuss. It is a guide for the reader who gets lost or misunderstands something.

## **2. Current System**

The reservation system currently being used is by making a phone call to the restaurant a person desires to go and provides details to get the reservation, the correspondent then lets the caller know if the reservation is available. Alternatively the person can go to the restaurant physically and make the reservation at the reception. Using this method, if the person has no idea about the restaurant (quality, food taste, location etc) then it doesn't provide insight about the restaurant you are going to. In addition to this, because it is via network, the caller can be made to wait because of a busy line or, being even unable to reach the restaurant due to some network problems. In this case the caller is being made to wait. If the person goes physically to make the reservation, he goes without knowledge of whether there is available space or not, and he may not know the location.

## **3. Proposed System**

The system to be designed is going to be made for making online reservations for restaurants. A restaurant manager signs up and sends a request to the admin to registers

the restaurant. For the registration, the manager needs to provide information about the restaurant which include it's location, food type, menus etc.. The system admin accepts the restaurant registration request which then makes the new restaurant visible to customers who may want to reserve. Then the customer also signs up and logs into their account and from there, they browse through the list of available restaurants and view the restaurant of their choice. The rating of a restaurant helps a customer decide where they want to make the reservation. So each restaurant has a rating which is given by previous customers based on their personal experiences. Because this is done online, a customer doesn't need to call to make a reservation so this avoids the inconvenience of waiting. Another benefit of this proposed online reservation system is that the menus and type of restaurant are displayed on screen before making a decision, so it helps to make up people's minds. Restaurants have ratings, so our customers have a general idea of knowing what kind of restaurant they are walking into. The uncertainty we spoke of earlier about restaurant location won't be a problem either because restaurant managers provide the location address before registering to the system. The customer can search for the address and pinpoint the location. After a customer completes their reservation, they are assured of being able to get a place in that restaurant and it avoids the frustration of going to a restaurant and it is unfortunately full, prompting another search for another place which may be full as well. We make all this unnecessary by making it possible online where there is a variety of different restaurants.

### **3.1.Overview**

In this section, we describe the functional requirements of our system and how they are implemented and who implements them. We have three users in our system. First, let's start with the Customer. The customer has access to the most functionalities and he is the one who is able to: Make Reservation, Rate Restaurant, Edit Profile, Search Restaurant, Change Reservation, Cancel Reservation. These steps all require a login action to be performed first meaning, a customer must be registered on the system before having these functionalities available. Our customer is able to reserve a restaurant through our system and to this he logs in, he chooses the restaurant of his choice and he selects the number of guests he wants to reserve for. He is shown the available times and he chooses the one that suits him the most. After this he completes his reservation.

Next up we have the Manager. He is the one who is responsible for adding restaurants to the system but first, he needs to send a request to the admin who has to approve the add request. The manager has the following functionalities available to them: Send Add Restaurant Request, Send Disable Restaurant Request, Edit

Restaurant, Edit Profile, Show Reservation List. As with the customer, the manager needs to be registered to the system before these functionalities become available to them.

We also have the Admin who is overseer of all operations on the system which includes accepting requests and declining requests. His functionalities include: Add Restaurant, Disable Restaurant. However, before he is able to perform these actions, he obviously needs to receive requests from the manager.

### **3.2.Functional Requirements**

In our system, one of our main functionalities is making a reservation which is done by the customer. As we mentioned earlier the customer is a user who is registered to the system. To make a reservation he logs in and chooses from a list of restaurant which one he wants to go to. He selects the number of people going and then choose the available times. He confirms the reservation and he is displayed with a receipt. Our system will also support changing reservation. This happens when the customer has to change change/edit his reservation. For example, he wants to increase the number of guests to reserve for in the same restaurant, so he changes his reservation to edit this. The system also allows the customer to rate a restaurant. This will be done by going to previous reservations and choose the restaurant. There are five stars and the customer can give from one to five stars with five being the best. Customer can also search for a restaurant by name. We have included a search bar functionality and when the customer presses it, the keyboard pops up and he types in the name. Results matching the name are returned.

Our Manager is able to send requests to add their restaurant to the system. He goes to the add restaurant section and he is presented with a form. He fills in the fields which are required to register the restaurant to the system. The fields include: restaurant's name, address, phone number, features. He then sends the request which shall be delivered to the admin's inbox. Our system also allows the manager to be able to take down their restaurant from the system. To do this he has to send another request to the admin. He goes to the "show my restaurants" section and he selects one restaurant at a time (if he has multiple ones). He then presses on the send disable request which shall be delivered to the admin inbox.

The admin's main functionalities will be to add and disable restaurants on the system. To add a restaurant he opens his inbox and he has some waiting requests sent by managers, he opens it and if he chooses to accept the request, he adds the restaurant to the system's database so customers can now make reservation to this restaurant. To disable restaurant, he again opens inbox and there are some requests, he



opens it and if he chooses to accept the request he disables it from the system and customers cannot make reservations to the restaurant anymore.

### **3.3.Nonfunctional Requirements**

*Usability-* We made our system very usable and to measure this, making a reservation only takes five clicks. And this is the main functionality of our system. So just about anyone with basic knowledge of technology can use our system.

*Reliability-* Our system is expected to run 100% of the time so that our users can be able to reserve at any time.

*Performance-* System performance needs to be optimal, this includes search results times, loading restaurant pages etc.. So our users wont need to wait too long for system to load. We also want to support at least 300 parallel users.

*Supportability-* System needs to be modified easily without difficulties.

*Implementation-* System shall be developed on Android App Studio. This means the programming language will be mainly based on Java. Our web service will be supported by Spring MVC Framework. Our database will be managed via MySQLWorkbench.

*Interface-* Our interface will be an android application interface.

*Packaging-* No packaging requirements.

*Legal-* The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and no infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

### **3.4.System Models**

Describes the scenarios, use cases, object model, and dynamic models for the system. This section contains the complete functional specification, including mock-

ups illustrating the user interface of the system and navigational paths representing the sequence of screens.

/\*

For example, one of the functionalities our system is designed to support is to make a reservation. To do this, firstly the customer needs to create an account. The customer subsequently logs in after successfully creating the account. After this he browses through the list of available restaurants and views the restaurant. He then chooses the number of people to reserve for and finalises his reservation. For the object models we depict our classes using UML diagrams and show the relation between them. In the dynamic model, we showed interactions between objects within a use case using sequence diagrams and showed how state machines are used to show the behaviour of that single object.

\*/

### *Scenarios*

A scenario is an instance of a use case.

### *Use case model*

A use case is a generalization of a number of scenarios. Therefore, the number of scenarios must be equal to or greater than the number of use cases.

### *Object model*

The analysis object model, depicted with UML class diagrams, includes classes, attributes, and operations. The analysis object model is a visual dictionary of the main concepts visible to the user.

### *Dynamic model*

The dynamic model is depicted with sequence diagrams and with state machines. Sequence diagrams represent the interactions among a set of objects during a single use case. State machines represent the behavior of a single object (or a group of very tightly coupled objects). The dynamic model serves to assign responsibilities to individual classes and, in the process, to identify new classes, associations, and attributes to be added to the analysis object model.

When working with either the analysis object model or the dynamic model, it is essential to remember that these models **represent user-level concepts, not actual software classes or components.**

*User interface—navigational paths and screen mock-ups*

### **3.5.Project Schedule**

Prepare Gannt Chart, and add it to this section.

## **4. Glossary**

To establish a clear terminology, developers **identify the participating objects** for each use case. Developers should **identify, name, and describe them** unambiguously and collate them into a glossary.

## **5. References**

This subsection should:

- Provide a complete list of all documents referenced elsewhere in the RAD, or in a separate, specified document.
- Identify each document by title, report number - if applicable - date, and publishing organization.
- Specify the sources from which the references can be obtained.

The following is an example of listing a book in this section. Check the text to see how it is cross referenced (The whole document is based on [1]).

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3<sup>rd</sup> ed.

2.