Restaurant Booking System

Object Design Document

1.0

21.12.2019

Berke Biten

Volkan Davşan

Ali Atahan Topal

Oğuz Kaan Yazan

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc436772639)

[1.1. Object Design Trade-offs 1](#_Toc436772640)

[1.2. Interface Documentation Guidelines 1,2](#_Toc436772641)

[1.3. Definitions, Acronyms, and Abbreviations 2](#_Toc436772642)

[1.4. References 2](#_Toc436772643)

[2. Packages 3](#_Toc436772644)

[3. Class Interfaces 3](#_Toc436772645)..7

OBJECT DESIGN DOCUMENT

# Introduction

Describes the general trade-offs made by developers (e.g., buy vs. build, memory space vs. response time), guidelines and conventions (e.g., naming conventions, boundary cases, exception handling mechanisms), and an overview of the document. Interface documentation guidelines and coding conventions are the single most important factor that can improve communication between developers during object design. These include a list of rules that developers should use when designing and naming interfaces.

## Object Design Trade-offs

**Memory Space vs. Response Time**

We used horizontal mapping for our User type models in our database. That increases our memory usage because we keep same type of data in different tables (Restaurant Owner, User, Admin). But that provides a faster operations for a specific type of user. For example searching for a restaurant in only restaurant owners data is much more efficient.

**Quality vs. Time**

The fact that we don’t have much time effected our project’s quality. Our functions are working but there is a great chance for a bug to occur in some of them because we didn’t consider all the possibilities.

**Functionality vs. Interface Design Quality**

We have so many functions to work with therefore we couldn’t pay more attention to visual design in the specified time.

## Interface Documentation Guidelines

|  |  |
| --- | --- |
| Identifier Type | Rules for Naming |
| Packages | The prefix of a unique package name is always written in capital and all lowercase. |
| Classes | Class names should be nouns, in mixed case with the first letter of each internal word capitalized. We kept our class names simple and descriptive. Nonetheless, we used whole words and avoided acronyms and abbreviations as far as possible. |
| Interfaces | Interface names should be capitalized like class names. Interface names clearly describes its common operations. |
| Methods | Our Methods are verbs, if there is two words or more in the method name, after the first word, first letter of the other words are capital. |
| Variables | All variable names start with dollar sign ($) because our code is written in PHP. Our variable names are not starting with underscore (\_) unless it is a global variable which provided by PHP itself.  When we are picking our variable names, we picked short and meaningful names in order to indicate both observers and developers of its use. |

**Exception Handling Mechanism:**

1. **Input Exception:** In order to prevent user input originated exceptions/errors such as entering invalid username or password, or entering invalid type of characters (like ‘;’) our system validate the inputs before processing them and if there is something wrong with them, shows the user an error message.
2. **No Available Capacity Exception:** When a user wants to make a reservation from a restaurant, he/she selects a time interval for the reservation. System calculates the capacity for the related restaurant in the specified interval and if there is no available capacity, an acknowledgment is being given to the user.

# Boundary Cases:

1. **Request Ticket Limit:** A user can have five open tickets at a time. If a user already has five unresponded tickets and tries to submit another one, he/she will get an acknowledgment that he/she can’t submit another one.
2. **Review Limit:** Users can only drop reviews for their past bookings.

## Definitions, Acronyms, and Abbreviations

**ODD:** Object Design Document

**PHP:** Hypertext Preprocessor

**HTML:** Hypertext Markup Language

**CSS:** Cascading Style Sheets

**Memory Space:** The amount of storage needed while solving a problem.

## References

1.Restaurant Booking System RAD Final Version.

2.Restaurant Booking System SDD Final Version.

3. Bernd Bruegge, Allen H. Dutoit, Object Oriented Software Engineering Using UML, Patterns and Java, 3rd ed., Prentice Hall.

4. SE3101 Lecture Slides.

# 2. Packages

We decomposed our subsystems into three packages which are Interface, Application Logic and Storage.

**Interface Package:** Basically, this package contains our HTML and CSS files. This package provides interface to the related user and functions. This package contains AdminInterface, GuestInterface and RegisteredUserInterface subsystems.

**Application Logic Package:** Application Logic package contains our PHP files which processes our functions in the system. Subsystems in Application Logic Package, gets data from Storage package and gets input from Interface Package. This package contains UserManagement, Search, Support, Registration, BookManagement, Review, Notification subsystems.

**Storage Package:** This package contains our database. This package is used by Application package and stores/provides information. This package contains Storage subsystem.

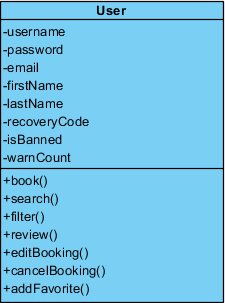
# 3. Class Interfaces

**3.1 Interface Package**

**3.1.1 User Class**

Users have username and password.They can login to our website with that informations.They have also typical personal informations like email, firstname and lastname. We use email for the communicate with the users in some specific cases. We keep the recovery code for every user that registered our system. System send it to user’s email when they forgot the password. In that way, they can reset the password and access the website again. There are also isBanned and warnCount attributes. Our admins use them for the management of the website.

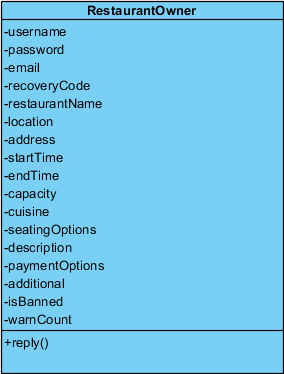
Users can book a table from a restaurant which is our main function.They can search for a restaurant and filter that search.They can also drop a review to the restaurants after the bookings. With edit and cancel booking operations they allowed to make changes or cancel their upcoming bookings.



**3.1.2 Restaurant Owner Class**

Restaurant owners have username, password, email, first name, last name and recovery code like every registered user in our system. Restaurant owners stands for a restaurants. So this class have some attributes for restaurants. These attributes are restaurant name, location, address, start time, end time, capacity, cuisine, seating options, description, payment options, and additional. All of these attributes informs the user to get an idea about the restaurant.The system use some of these attributes in search and filter operations. Also when a user try to book a table, system checks the suitability of booking with capacity and start-end time attributes.

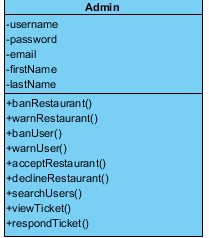
Restaurant owners have reply function. They can reply the review of their restaurants.



**3.1.3 Admin Class**

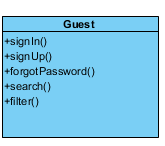
Admins have username, password, email, first name and last name attributes.

They have functions that helps to manage the website. They can ban or warn users and restaurant owners. If a registered user has more than 10 warning, that user banned from the website automatically. An admin controls the restaurant signups. If there is a restaurant sign up, admins check it first then they accept or decline the signup. Admins can search for a registered user to see their profile. Also admins respond the tickets that sent by the users or restaurant owners.



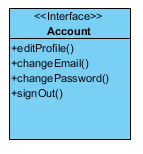
**3.1.4 Guest Class**

A guest can use sign up and restaurant sign up functions to become a registered user. They can also use forgot password function if they have registered to the website already but can’t remember their password. If they continue to use the website as a guest they can search for a restaurant and filter the search.



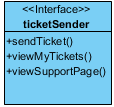
**3.1.5 Account Interface**

This interface allows the registered users (restaurant owners and users) and admins to make changes in their account and profile. They can change the informations that seen in the their profiles. For example a restaurant owner can change the images, menus, and the other features(address,seating options,payment options etc.) of their restaurants. Also they can change the password and email from the account settings. With sign out function they can exit the system and become a guest.



**3.1.6 ticketSender Interface**

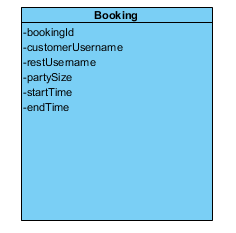
This interface allows the registered users (User and Restaurant Owner) to send a ticket, view the tickets that sent by themselves and view the support page.



**3.2 Application Logic Package**

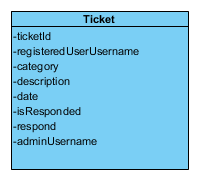
**3.2.1 Booking Class**

Like we mentioned before, user can book for a table from a restaurant and this is the our main function. We keep the datas of the bookings in booking class. A booking has booking id that identifies the booking. It contains the party size, start and end time of the booking that chosen by user. Also we should know the which user book a table from which restaurant. Therefore a booking has customer and restaurant username.



**3.2.2 Ticket Class**

A ticket is created by a registered user. It has ticket id to identify the ticket. A ticket contains category, description and date. These attributes given by user who send the ticket. An admin can respond the tickets. So we keep this respond and isResponded attributes in the ticket class. When a ticket is responded, isResponded attributes become true. Also ticket class has admin and registerdUser username.



**3.2.3 Review Class**

A review is created by a user that has a past booking in a restaurant. It has a id to identify a review. Users evaluate the restaurants with the review. They specify the quality of the restaurant and price of the foods. We keep these datas with price and star attributes. Also a review has text which contains the comment of the user. Similarly to booking it has customer and restaurant username. Restaurant owners can reply the reviews that dropped for their restaurant and it is kept with the reply attribute.

