REMAS

System Design Document

0.1

27.12.2020

Bilal Ado

Sıla Topal

Batuhan Akçay

Mehmet Keremhan Yılmaz1

Prepared for

SE3101 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc433996772)

[1.1. Purpose of the System 1](#_Toc433996773)

[1.2. Design Goals 1](#_Toc433996774)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc433996775)

[1.4. References 1](#_Toc433996776)

[2. Current Software Architecture 1](#_Toc433996777)

[3. Proposed Software Architecture 1](#_Toc433996778)

[3.1. Overview 1](#_Toc433996779)

[3.2. System Decomposition 1](#_Toc433996780)

[3.3. Hardware Software Mapping 2](#_Toc433996781)

[3.4. Persistent Data Management 2](#_Toc433996782)

[3.5. Access Control and Security 2](#_Toc433996783)

[3.6. Global Software Control 2](#_Toc433996784)

[3.7. Boundary Conditions 2](#_Toc433996785)

[4. Subsystem Services 2](#_Toc433996786)

[5. References 2](#_Toc433996787)

SYSTEM DESIGN DOCUMENT [1]

# Introduction

## Purpose of the System

The purpose of this system is to provide an all-in-one system that will handle restaurant operations such as taking, processing and delivering orders in the restaurant. User should be able to record orders from the customers , pass them to the chef for processing, where the chef updates the progress of the order, and the waiter can pick up when it is ready. Additional features that Admins can do include viewing income, updating menu and table capacity. Additional features that Waiters can do include creating reservation, creating receipts and updating table status. Depending upon the role/authorization, each user will have access to certain features.

## Design Goals

**Performance**: In terms of response time, our Restaurant Management System (REMAS) should be able to handle all requests from users such as logging in and updating information, immediately. The ideal case will be the system can accomplishes up to 100 tasks in a fixed period of time. Since it is a web application, not much memory will be required for the system to run.

**Dependability**: REMAS will have great robustness as it will be able to survive invalid user input by displaying notification or warning messages to users when their input is invalid. REMAS will also be reliable as it will only deliver related results as requested by users. The availability of this system will be 24/7 to allow access at any time. The system will be safe and will not endanger human lives.

**Cost**: Our web application doesn’t have any cost, it’s free.

**Maintenance**: REMAS will be extensible to allow the addition of new functionalities to the system besides the current features. The system will also be easy to modify as long as the developer(s) have the knowledge of Java, Spring, etc. In terms of portability, our system should be able to run on computer with the presence of internet connection. Readability for this system will be maintained with the addition of comments in system code. Ultimately, REMAS should implement all functional requirements listed in the RAD to facilitate traceability of requirements.

**End User Criteria:** To increase the user’s utility and usability to our system, REMAS will display a user interface that is friendly, reactive, and scalable.

## Definitions, Acronyms, and Abbreviations

REMAS: Restaurant Management System

RAD: Requirement Analysis Document

SDD: System Design Document

HTML: Hyper Text Markup Language

CSS: Cascading Style Sheets

URL: Uniform Resource Locator

## References

From existing software:

Eat app restaurant management system.

Flip dish.

Square for restaurants.

MarketMan.

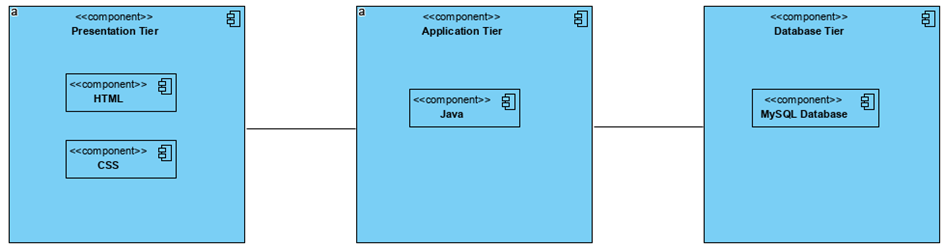
# Current Software Architecture

The general current system architecture is organized such that the views and behaviors are simple and straight forward, After login with proper credentials the user is presented with a summarized menu which provides at a glance view of all important information. All interaction as snappy to make daily operations as smooth as possible, and for this reason clutter is avoided as much as possible while trying to provide as much required information as possible.

# Proposed Software Architecture

In this section subsystems of REMAS application will be shown. The main system of REMAS consists of many subsystems. These subsystems are important for the main system to work properly

## Overview

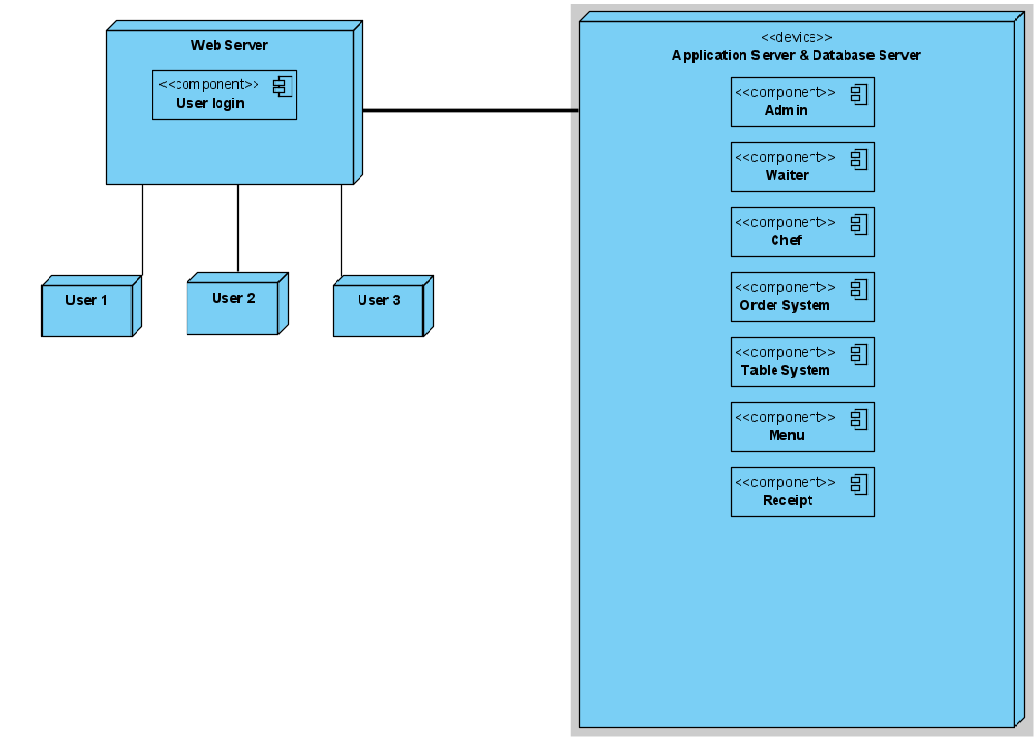
REMAS is a system using 3 tier architecture. The first tier represents the interface that users see. This tier will be done with HTML and CSS. The purpose of this tier is to allow users to interact with the system. The application tier will receive and process what the user does through subsystems. The last tier will store data using MySQL.

## System Decomposition

## Hardware Software Mapping

Component Diagram

## Persistent Data Management

****For persistent data management, a relational database created by MySQL will be used as  
the database management system (DBMS) of the REMAS. This database is used to store all data related to the REMAS, including login data, reservation data, income data, order status data, tables data, receipt data, menu data. REMAS shall implement some authentication mechanism for access to the database.

Deployment Diagram

## Access Control and Security

To provide the security all passwords must be a minimum of 8 characters in total and must consist of both upper and lower case letters along with numbers and symbols like !,@,# and so on.

|  |  |
| --- | --- |
| Actor | Functions |
| Users | * Login/Logout |
| Admin | * Update Users info * View receipts * View income * Change menu * Change capacity of tables * Add table |
| Employees | * View Orders * Change the status of the orders |
| Waiter | * Make reservation * Enter orders * Cancel orders * Change orders * Update tables status * Create receipts by table numbers |
| Chef |  |

## Global Software Control

REMAS should make sure many users can interact at a time and should not affect data consistency. System originally designed to used by one chef account, one waiter account and an admin account. Because of everything in the system can be controlled/changed by one person and the others can’t interact with it. These numbers can be increased for waiters. Increase in numbers will not cause a problem because of how restaurants work.

## Boundary Conditions

**Start up:** When the URL of the application is visited it loads and requests the user login, after that it provided, it authenticates and presents the homepage to the user.

**Shutdown:** The user logs out of the system and is taken back to the login page where they can then shut down their browser.

**Error behavior:** Whenever an exception is caught, the system displays a generic error message and requests a refresh of the current page the user is viewing.

# Subsystem Services

**LOGINPAGE**

In order to use the system users first needs to have access to the system. In this subsystem user enters their username and password. It must be checked if the login information exists in the database. If the information exists in database user direct to page related the information they entered. If the information does not exists or doesn’t match users will get a warning.

Subsystem services:

* Admin
* Waiter
* Chef

**ADMINSTOREFRONT**

This subsystem can be accessed only by admins and it is basically admin panel. From this subsystem admins can access all of their functions. Does not provides any service.

**STOREFRONT**

This subsystem can be accessed by chefs and waiters. This subsystem is basically an interface lets the employees can use their functions. Does not provides any service.

**MENU**

To access this subsystem users must be using either the admin storefront or storefront. From admin storefront admins can access the create, update and delete functions which can be use on dishes. From storefront employees can reach the information saved to database.

Subsystem services:

* SQL Communication

**TABLESYSTEM**

Table system can be reached from order system or admin storefront. This subsystem provides table information (table number) to order system from database. This systems information can be changed by admins so it provides a SQL communication to admin storefront.

Subsystem services:

* SQL Communication
* Table Number

**ORDERSYSTEM**

Order system can be accessed by employees. It’s a subsystem allows employee create, update and cancel orders and from this orders employee can create receipts.

Subsystem services:

* Receipt
* Order

**RECEIPT**

Receipt system allows creating and viewing receipt. Created receipt will be saved to database. Receipts can be view by admins in admin storefront.

Subsystem services:

* SQL communication

# References

1. <https://ase.in.tum.de/lehrstuhl_1/images/projects/tramp/sdd.pdf>
2. <https://creately.com/diagram/example/j9w8sozo/restaurant%20food%20ordering%20system>
3. <http://univera-ng.blogspot.com/2010/04/uml-ve-modelleme-bolum-10-component-ve.html>