Restaurant Management System

Analysis and Design Document

Student: Ratoi Razvan Valeriu

**Group: 30431**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

Design and implement a client-server application used to record orders, handle payments and manage the employees’ activity in a restaurant. The system allows (i) waiters to record orders in the system and to handle payments by considering discounts depending on the loyalty of the client, (ii) kitchen employees to send notifications to waiters when the ordered food is ready, and (iii) a manager to perform CRUD operations on employees’ information and to check their activity. An order has the following attributes: table number, client ID, waiter ID, the list of ordered products (drinks, food, desert, etc.) as well as their quantity. The loyalty of a client is evaluated according to the number of times the client ordered from the restaurant as well as the cost of each order.

# Elaboration – Iteration 1.1

# Domain Model

A restaurant manager oversees the daily operations of a restaurant to ensure that it runs efficiently and profitably. This includes managing staff, ordering supplies, and ensuring customer satisfaction. The domain model for a restaurant manager includes the following entities:

1. Staff: This entity includes all employees who work at the restaurant, such as servers, cooks, and dishwashers, including the manager.
2. Order: This entity represents a customer's request for a meal or drink.
3. Reservation: This entity represents a customer's request to reserve a table at the restaurant for a specific date and time.
4. Customer: This entity represents the person who visits the restaurant to dine.
5. Payment: This entity represents the payment made by the customer for their order.

Conceptual Class Diagram:

The conceptual class diagram for a restaurant manager includes the following classes:

1. Staff

* Attributes: name, job title, salary, loyalty
* Operations: takeOrder(), prepareFood(), serveFood(), processPayment(), applyDiscount(), increaseLoyalty(), decreaseLoyalty()

1. Order

* Attributes: table number, items, total price
* Operations: addItem(), removeItem(), calculateTotal()

1. Reservation

* Attributes: table number, date, time, number of guests
* Operations: makeReservation(), cancelReservation()

1. Customer

* Attributes: name, phone number, email address, loyalty status
* Operations: placeOrder(), makePayment(), receiveDiscount()

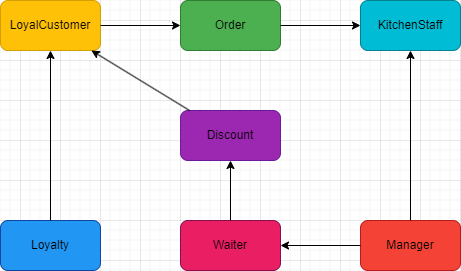
1. Payment

* Attributes: payment method, amount
* Operations: processPayment()

1. Manager (part of Staff)

* Attributes: name, job title, salary
* Operations: viewStaffActivity(), viewRevenue(), viewCustomerFeedback()

The conceptual diagram would be like the following:



# Architectural Design

## Conceptual Architecture

Having in mind the purpose of this whole project is to develop a web application and also putting emphasis on the back-end side, my first idea is to design my application by the Layered Architectural Pattern, having layers that communicate to get the data from the database to the end-user. This would imply having repositories (the layers that perform the CRUD operation on the database), services (the layers that process the data, use it to perform other operations or analyze it on a deeper level) and controllers (the layers that contain the endpoints which would be called at the front-end level, return corresponding statuses, based on input given by the user or output resulted from the deeper levels).

## Package Design

A first draft of the package diagram would look like the following:

A picture containing graphical user interface

Description automatically generated

## Component and Deployment Diagrams

Also a first draft of the Deployment Diagram would be:

Text

Description automatically generated

# Elaboration – Iteration 1.2

# *Diagram Description automatically generated with low confidence*Design Model

## Dynamic Behavior

Pay Order

*Diagram

Description automatically generated*

Take Order

## Class Design

*A screenshot of a computer screen

Description automatically generated with medium confidence*

# Data Model

*A picture containing screenshot, design

Description automatically generated*

# Unit Testing

For unit testing, I chose 3 scenarios, for each creating a mock-up of the database and repositories, being scoped the testing of the business logic: getting all orders, testing hashing the password and getting the discount of a loyal client.

# Elaboration – Iteration 2

# Architectural Design Refinement

The packages in my project consist of the following:

* Data, in which the connection to the database is established
* Command, in which I have the classes and interface needed to implement the Command DP.
* Controller, in which I have the API controllers for the orders, products and users.
* DTOs¸ in which I have some models for the credentials, users and orders.
* Facilities, in which I have the services for the JWT Token and password hashing.
* Mappings¸ in which I have the mappers needed for the models and dtos.
* Models¸ in which I have the classes corresponding to the tables in the database.
* Observer, in which I have the 2 Observer classes and the interface.
* Repositories¸ in which I have the Data Access classes.
* Services, in which I have the classes containing the business logic.
* Tests, in which I have the unit tests.

# Design Model Refinement

*A screenshot of a computer screen

Description automatically generated with medium confidence*

# Construction and Transition

# System Testing

The main testing of the application was by testing the HTTP requests with Insomnia, in which I provided the expected inputs (and some unexpected ones to check for errors) and verify that the responses are accordingly.

# Future improvements

In the future I would like to improve the menu of the application, to include images for every product and also to include promotions such as an event discount, to get the number of invited persons and choose a specific menu and to manage reservation (possibly by adding a host user).

# Bibliography

* C# fundamentals recap (https://app.pluralsight.com/library/courses/object-oriented-programming-fundamentals-csharp/table-of-contents)
* C# collection fundamentals (https://app.pluralsight.com/player?course=csharp-collections&author=simon-robinson&name=csharp-collections-fundamentals-m1-welcome&clip=0)
* Head first C# book (https://www.amazon.com/NET-Head-First-Microsoft-Programming/s?ie=UTF8&page=1&rh=n%3A764452%2Cp\_lbr\_books\_series\_browse-bin%3AHead%20First)
* Web API (https://app.pluralsight.com/library/courses/implementing-restful-aspdotnet-web-api/table-of-contents)
* C# basic authentication check with user & password and authorization using roles (for assignment 3) https://www.c-sharpcorner.com/article/basic-authentication-using-message-handlers-in-web-api/
* Message handlers in C# (https://www.infoworld.com/article/2991458/application-architecture/how-to-work-with-message-handlers-in-web-api.html)
* http://www.tutorialsteacher.com/webapi/consuming-web-api-in-dotnet-using-httpclient
* https://www.aspsnippets.com/Articles/Call-Consume-Web-API-in-Windows-Forms-Application-using-C.aspx