<Project Name>

Analysis and Design Document

Student:

**Group:**

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

*[Present the project specification]*

# 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()

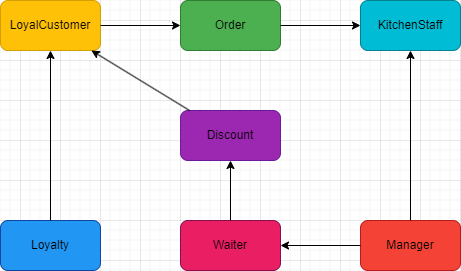
1. Payment

* 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

# Design Model

## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

# Data Model

*[Create the data model for the system.]*

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

*[Present future improvements for the system]*

# Bibliography