# Software Requirement and Design Specifications

***[Food ordering and Delivery system]***

***Version: [1.0]***

|  |  |
| --- | --- |
| *Course Code* | CS 3004 |
| *Instructor* | Sir Abdul Rehman |
| *Project Team* | Hasnain Somani (19k-0204)  Ashmal Anis (19k-0305)  Abdul Samad (19k-1396) |
| *Submission Date* | 21/Dec/2021 |

## [Instructions]

###### No section of template should be deleted. You can write ‘Not applicable’ if a section is not applicable to your project. But all sections must exist in the final document.

* *All comments/examples mentioned in square brackets ([]) are in the template for explanation purposes and must be replaced / removed in final document.*

###### This’ Instruction’ section should also be removed in final document.

*Table of Contents*

1. [INTRODUCTION 5](#_bookmark0)
   1. [Purpose of Document 5](#_bookmark1)
   2. [Intended Audience 5](#_bookmark2)
2. [OVERALL SYSTEM DESCRIPTION 6](#_bookmark3)
   1. [Project Background 6](#_bookmark4)
   2. [Project Scope 6](#_bookmark5)
   3. [Not In Scope 6](#_bookmark6)
   4. [Project Objectives 6](#_bookmark7)
   5. [Stakeholders 6](#_bookmark8)
   6. [Operating Environment 6](#_bookmark9)
   7. [System Constraints 6](#_bookmark10)
   8. [Assumptions & Dependencies 6](#_bookmark11)
3. [EXTERNAL INTERFACE REQUIREMENTS 7](#_bookmark12)
   1. [Hardware Interfaces 7](#_bookmark13)
   2. [Software Interfaces 7](#_bookmark14)
   3. [Communications Interfaces 7](#_bookmark15)
4. [FUNCTIONAL REQUIREMENTS 8](#_bookmark16)
   1. [FUNCTIONAL HIERARCHY 8](#_bookmark17)
   2. [Use Cases 8](#_bookmark18)
      1. [[Title of use case] 8](#_bookmark19)
5. [NON-FUNCTIONAL REQUIREMENTS 9](#_bookmark20)
   1. [Performance Requirements 9](#_bookmark21)
   2. [Safety Requirements 9](#_bookmark22)
   3. [Security Requirements 9](#_bookmark23)
   4. [User Documentation 9](#_bookmark24)

[SDS 10](#_bookmark25)

1. [SYSTEM ARCHITECTURE 11](#_bookmark26)
   1. [SYSTEM LEVEL ARCHITECTURE 11](#_bookmark27)
   2. [SOFTWARE ARCHITECTURE 11](#_bookmark28)
2. [DESIGN STRATEGY 12](#_bookmark29)
3. [DETAILED SYSTEM DESIGN 13](#_bookmark30)
   1. [DATABASE DESIGN 13](#_bookmark31)
4. [APPLICATION DESIGN 15](#_bookmark32)
5. [REFERENCES 15](#_bookmark33)
6. [APPENDICES 17](#_bookmark34)

## Introduction

##### Purpose of Document

##### The purpose of this document is to provide a detailed view of the project we have been working on.

##### Intended Audience

##### The Course Instructors or any other person who is connected to this field or finds this project relevant to his needs.

[**Definition of Terms, Acronyms and Abbreviations**

[This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly. ]

|  |  |
| --- | --- |
| ***Term*** | ***Description*** |
| *ASP* | *Active Server Pages* |
| *DD* | *Design Specification* |
| UCD | USE CASE DIAGRAM |
| URL | UNIFORM RESOURCE LOCATOR |
|  |  |
|  |  |
|  |  |
|  |  |

##### Document Convention

Font style: Arial.

Font size: 10

Font decoration: none.

For headings: (As given)

Font style: Arial.

Font size: 12

Font decoration: Italic, and Bold.

## Overall System Description

##### Project Background

##### The idea behind this project is to allow people order food items from multiple restaurants at the same time, in the same order, as well as it will allow customers to pre-book their orders for future (future scheduling). In addition to benefitting the customers, it makes employment of riders easier, as they can apply for jobs without having to visit the office.

##### Project Scope

##### The system has multiple User interfaces:

##### First for the admin, who is the authorized person. Admin can view reviews, as well as register new riders and restaurants.

##### Second interface is the main interface the project focuses on: the customer’s interface. Here, a customer is allowed to apply for premium membership, select the food category to order food from, after which he gets multiple restaurants to pick from, with their ratings mentioned. After choosing the restaurants, the customer can add any item from the given menu into his cart. Apart from basic shopping, the project allows customers to track their orders too, providing complete information about the rider. The customer is also allowed to schedule their orders for future.

##### Thirdly, it has an interface for restaurants and riders to register themselves.

##### Not In Scope

##### Not Applicable.

##### Project Objectives

##### This project will finally give people to order from multiple restaurants, and get all of it delivered by the same rider. In addition, customers can select how to customize all of their products, which has not been a feature earlier.

##### It will also help people to work with us easily. Either it’s a restaurant or a rider, they can apply to work with us through our website, or contact us on our given social media platforms.

##### These are the major objectives that were aimed and achieved by this project. Apart from that, the project helped us learn various strategies for solving the same problem, with a completely new environment (Flask).

##### Stakeholders

##### The admin, who can perform various operations that no one else is authorized to do.

##### The user who can login and use the system anytime once he has signed up on the website. The user can choose any category, any restaurant, and any items from the given list. In addition, he / she can also avail extended discounts using premium membership. The user can also track their order, once the order is placed.

##### Developer who developed the project, for further releases and new versions (Both front end, and back end).

##### Internal database engineer who manages the database working.

##### Software quality assurance engineer, who is responsible to test the environment for bugs, and errors.

##### Operating Environment

***Hardware platform:*** It needs pretty basic hardware requirements. Any average system will do the work. The hardware we used to run this is mentioned below:

* Core i5 6th Generation
* 8gb ram (hardly uses 1 or 2gb(s) of ram for all the softwares we used)
* Less than 200mb of disk space for project, and related data.

***Operating System:*** Windows 10 any variant (Recommended)

Network Environment: Should have a decent internet connection.

***Applications:***

* Visual Studio Code
* XAMPP
* MySql Server

##### System Constraints

##### Software constraints:

This system doesn’t require any software constraints so far.

##### Hardware constraints:

It needs pretty basic hardware requirements. Any average system will do the work. The hardware we used to run this is mentioned below:

* Core i5 8th Generation
* 16gb ram (hardly uses 1 or 2gb(s) of ram for all the softwares we used)
* Less than 200mb of disk space for project and its dependencies

##### Cultural constraints:

##### The person should know how to read, and understand basic English language.

##### Legal constraints:

##### Not applicable.

##### Environmental constraints:

* There are no environmental constraints. The system can be used anywhere anytime.
* The project is developed for people of all ages and ethnicities

##### User constraints:

##### No specific age constraints. Users of all ages can use the webpage.

##### Assumptions & Dependencies

##### Assumptions:

##### The same restaurant cannot offer multiple categories.

##### A user’s cart items are lost if he logs out without checking out.

## External Interface Requirements

[This section is intended to specify any requirements that ensure that the new system will connect properly to external components. Place a context diagram showing the external interfaces at a high level of abstraction.]

##### Hardware Interfaces

##### The system is perfectly supported by desktop computers. It cannot be supported on mobiles currently as it hasn’t been deployed to a domain.

##### The type of data we are taking from the user are login credentials, delivery information, and we’re saving the information for future orders.

Processor: Intel

Installed Memory: 2 GB or Higher

Speed: 1.40 GHz or Higher

##### Operating System: 32/64-bit Operating System

##### Software Interfaces

**Operating System:** Windows 10

**Database:** MySQL Workbench

**Webserver**: XAMPP

**Web Technologies**: HTMLL/ CSS/ JS/ FLASK

**Ide and tools:** Visual Studio Code

##### Communications Interfaces

##### User email for signup should be valid.

##### Credit / debit card credentials input must be valid.

##### No encryption usage, or data transfer or synchronization issues.

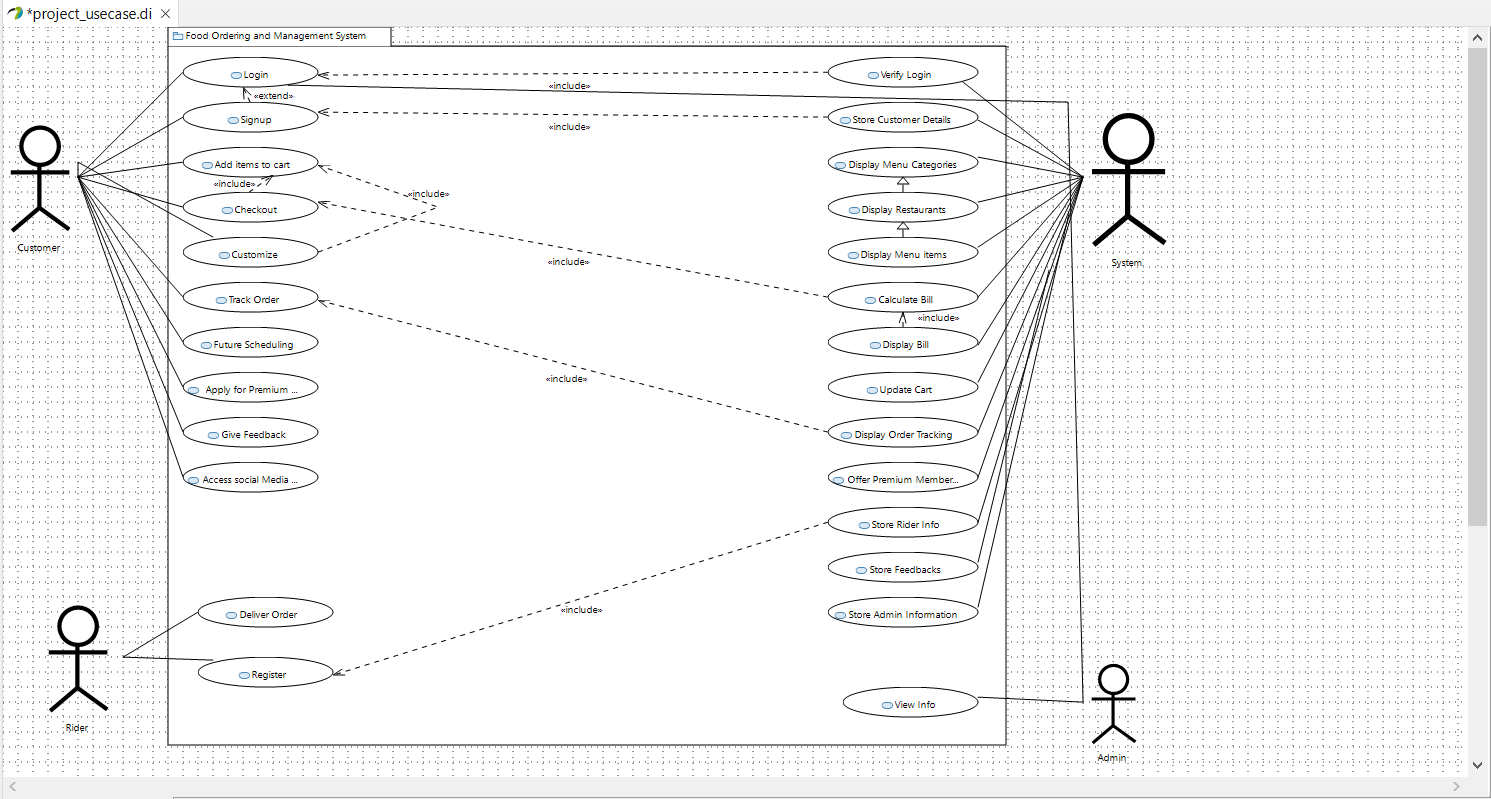
## Functional Requirements

##### Functional Hierarchy

[This section will give a big picture of overall system functionality. The main modules/features of system and their sub-functions will be described here in the form of a functional hierarchy so that, before getting into the use case, audience could grab the idea of overall system functions.]

##### Use Cases

***4.2.1. [Food Ordering and Management system]***



|  |  |
| --- | --- |
| ***Use Case Description*** | |
| ***Use Case name:*** | Food ordering and management system. |
| ***Use Case Description:*** | |
| ***Primary actor:*** customer | ***Other actors:*** rider, system, admin |
| ***Stakeholders:*** customer, rider, admin, software developer, software designer, DBA |  |
| ***Relationships***   * ***Includes:***   view\_info includes login  store\_customer\_details includes signup  customize includes add\_items\_to\_cart  checkout includes add\_items\_to\_cart  calculate\_bill includes checkout  display\_order\_tracking includes Track\_order  store\_rider\_info includes register   * ***Extends:***   Signup extends from login | |
| ***Pre-conditions:***  customer must have an adequate internet connection, and a desktop computer to run the webpage, and he should signup before moving ahead. | |
| ***Flow of Events:***   1. user signs up. 2. user logins. 3. user selects order now. 4. System displays categories. 5. User selects one. 6. System displays restaurants. 7. User selects one. 8. System displays menu. 9. User adds items to cart. 10. User selects checkout. 11. User selects payment method. 12. If payment method is card, system asks for card info. 13. User enters info. 14. System verifies card. 15. If payment method is cash on delivery, Order summary displayed. 16. Rider delivers order. 17. Customer pays, and receives the order. | |
| ***Alternative and exceptional flows:***  1. If login user doesn’t match, the information is taken again and again.  2. if the customer doesn’t checkout, and he logs out, his cart items are lost.  3. If order is scheduled for future, it cant be tracked. | |
| ***Post-conditions:***  -user registered.  - Rider registered.  - Restaurant registered. | |

## Non-functional Requirements

##### Performance Requirements

1. Performance wise, our project is built to be very responsive and fast. The transitions between the interfaces take no time.
2. Capacity wise, our system is very storage friendly, hardly requires some Mbs of data.
3. Safety wise, the data of users, interacting with our system stays safe and can only be accessed by the system owner.
4. The software is reliable in a sense that it fulfills all the needs that it is promised to fulfill. It was tested for any sort of bugs/ issues and was fixed by the developers eventually.

##### Safety Requirements

1. We took extra care that our system must not cause any damage on the machine on which the user is running our system.
2. The only thing that the user should take care of is the entry of dummy (fake) results in the database. Dummy data must be deleted from the database.

##### Security Requirements

1. External users such as someone out of the organization must not be given access to the system’s Admin panel. Login ensures this.
2. Only the Stakeholders should have access to the system.
3. The data of the user stays safe and untouchable. So, privacy is maintained.

##### User Documentation

User manuals, will be provided alongside and they will be explained how to use the system, first they are asked to sign-in if not already and then login to continue using the system, they can select their order from menu and can customize the order.

# SDS

## System Architecture

## The architecture embodies the major static and dynamic aspects of a system. It is a view of the whole system highlighting the important characteristics and ignoring unnecessary details. In the context of our approach, architecture is primarily specified in terms of views of tier architecture which is a client-server architecture in which the presentation, the application processing and data management are logically separate processes.

## Domain Model:

## 

### *System Level Architecture*

### *COMPONENT DIAGRAM*

### 

### *DEPLOYMENT DIAGRAM*

### 

### *Software Architecture*

### *User Interface Layer:* not applicable

### *Middle tier:* not applicable

### *Data access Layer:* not applicable

## Design Strategy

## System Reuse: the webpage may be deployed on a world wide web, allowing users from all around the world to access the website.

## Future system extension or enhancement: we aim to add additional features to the webpage, such as adding chat bot using Machine learning, and deep learning, as well as add real time order tracking (on a live map).

## User interface paradigms: the program prompts the user to enter their information at multiple stages: while signing up, logging in, as well as during checkout. Moreover, the UI also displays the user information at multiple pages, such as displaying the order summary, or menu items.

## Concurrency and synchronization: only one user is allowed to use the system at a time.

## Detailed System Design

## Class Diagram:

## 

## Functions:

## Customer:

## Signup – takes customer information, to register customer as a user.

## Login - takes customer information, and verify the information from the database. If the information matches, customer information is loaded, which is used when the customer places order. It prevents asking for repeated information from the user.

## Add\_to\_cart – the customer adds the food into the cart. Food\_id is stored in the customer’s suborder.

## AddCustomization – allows the customer to customize his order.

## Checkout – allows the customer to finalize his / her cart, displays order summary, and places the order.

## Track\_order – the customer can look at the order’s status by inputting his order number. The rider details, order summary, and estimated time to deliver is shown.

## Order:

## Display – shows order summary.

## Assign\_rider – assigns an available rider to the order.

## Calculate\_amount – calculates the final amount to be payed, after the discounts applied by premium membership.

## Checkout – order finalized, and placed.

## Card\_info:

## Add – add credit / debit card information.

## Update - Update credit / debit card information.

## Verify – verifies card credentials.

## Billing:

## Generate\_bill – generates bill with payable amount, and order summary.

## Get\_cust\_info – fetches customer data from customer id to use the information in printing the bill.

## Reviews:

## Get\_review – allows customer add a review on a previously placed order, by asking for customer details, and order number.

## Rider:

## Register – allows rider to register, and apply for a job by inputting some basic credentials.

## Restaurant:

## Add – add restaurant to the website. Some basic information is required to register.

## Update – update restaurant information.

## Food ordering system:

## Add\_restaurant - add restaurant to the website. Some basic information is required to register.

## Add\_rider - allows rider to register, and apply for a job by inputting some basic credentials.

## Category:

## Add\_topping: allows customization on food.

## Food:

## Add – add food item.

## Update – update food item.

## Suborder:

## Update\_order – add items to order, as cart is updated.

## Admin:

## Login – takes login credentials from the admin, verifies it, and then allows access to the functions.

## Check\_info – checking reviews.

### *Database Design*

#### ER Diagram

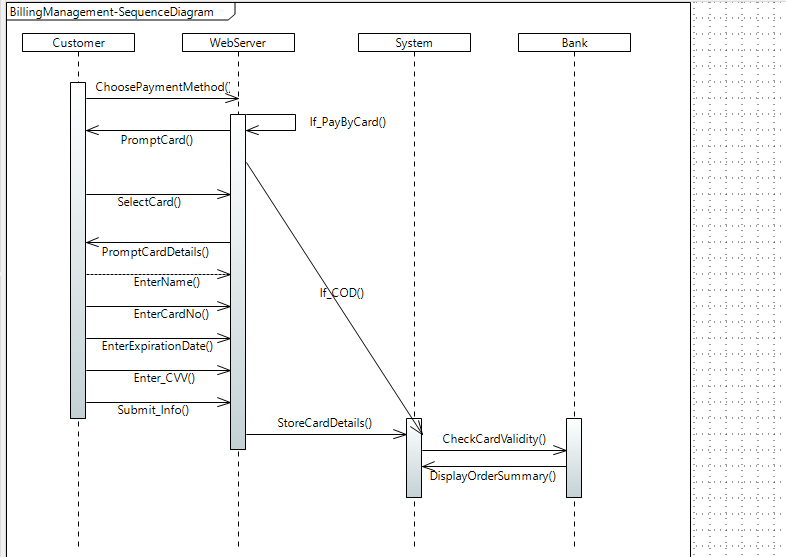
#### Data Dictionary

Not Applicable

### *Application Design*

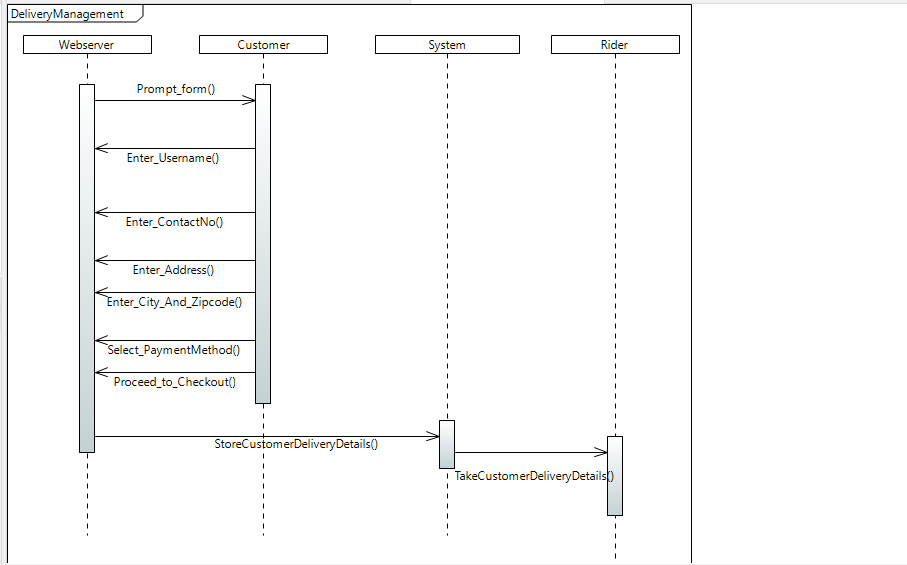
#### Sequence Diagram

* + - 1. <Sequence Diagram 1>



**Explanation:** This sequence diagram illustrates the working on Bill-Management feature. The customer chooses a payment method, which is verified by the webserver: if the payment method is by card, the customer is asked to enter his card details including name, card number, expiration date and cvv. This information is then submitted by the user, which is saved by the server. The server then checks if the card is valid or not, and displays order summary. On the other hand, if the payment method chosen was cash on delivery, the system directly displays order summary.

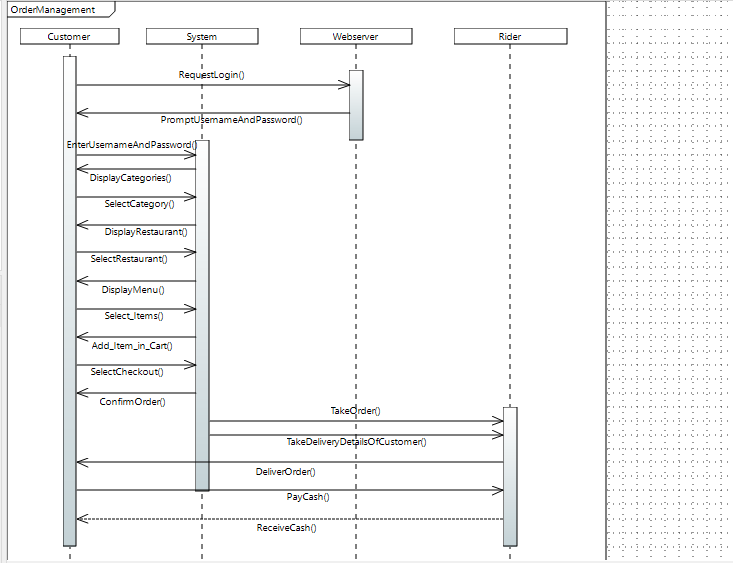
* + - 1. <Sequence Diagram 2>



Explanation: this sequence diagram is for the delivery management feature. The webserver displays the user a form, in which the user enters his details for delivery: his username, contact number, address, city, zip code, payment method, and then the user proceeds to checkout. All of the inputted details are stored in the server, and this information is then given to the rider, to get the order delivered.

.

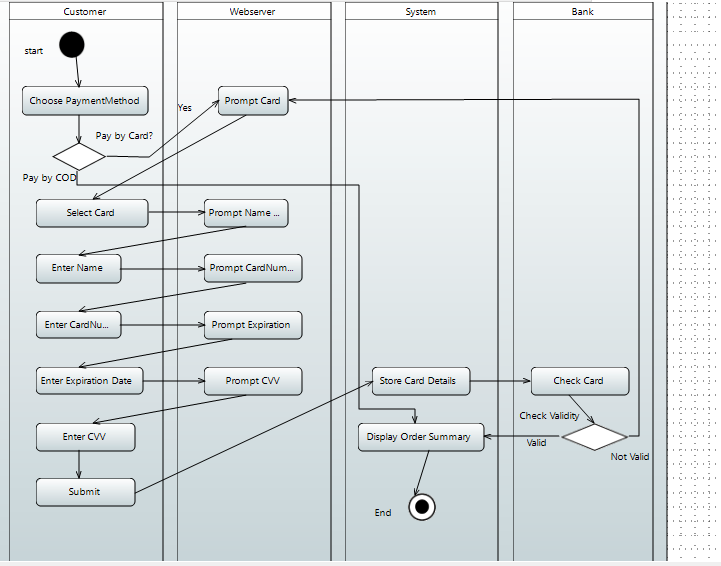
* + - 1. <Sequence Diagram n>



Explanation: this sequence diagram manages the feature for order management; the customer first requests to login, where he enters his username, and password. Once logged in, the system displays the customer categories: the customer selects a category, then , the system displays the customer restaurants available: the customer selects a restaurant, then the system displays the customer the menu offered by the selected restaurant. The customer selects an item, adds it in the cart. Once done, the customer selects checkout, and hence the system confirms the order. Order details, as well as customer details are given to the rider. The rider delivers the order, the customer pays, and the rider receives the payment.

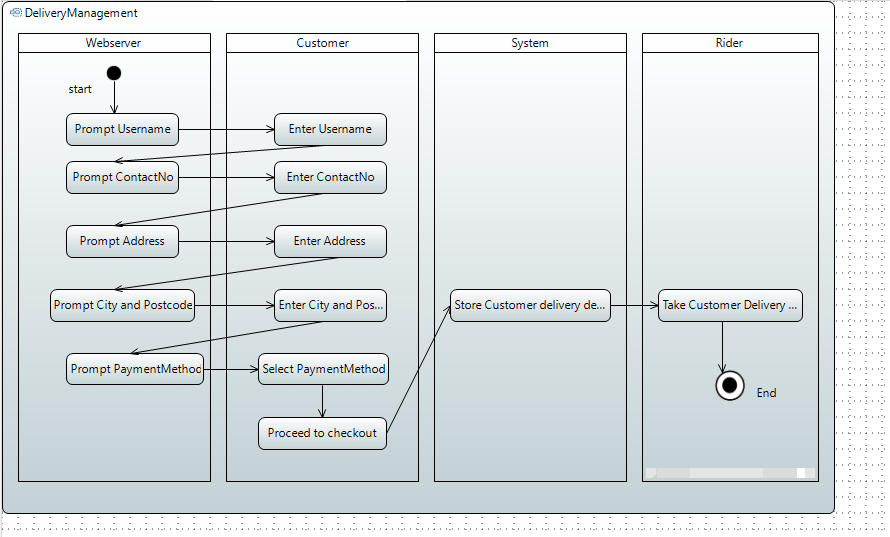
#### Activity Diagram

* + - 1. <Activity Diagram 1>



Explanation: This activity diagram illustrates the working on Bill-Management feature. The customer chooses a payment method, which is checked: if the payment method is by card, the customer is asked by the webserver to enter his card details including name, card number, expiration date and cvv. This information is then submitted by the user, which is saved by the server. The server then checks if the card is valid or not through the bank, and displays order summary. On the other hand, if the payment method chosen was cash on delivery, the system directly displays order summary.

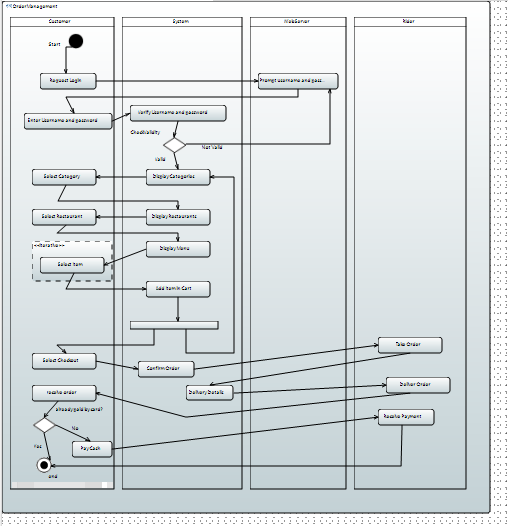
* + - 1. <Activity Diagram 2>



Explanation: this activity diagram is for the delivery management feature. The webserver displays the user a form, in which the user enters his details for delivery: his username, contact number, address, city, zip code, payment method, and then the user proceeds to checkout. All of the inputted details are stored in the server, and this information is then given to the rider, to get the order delivered.

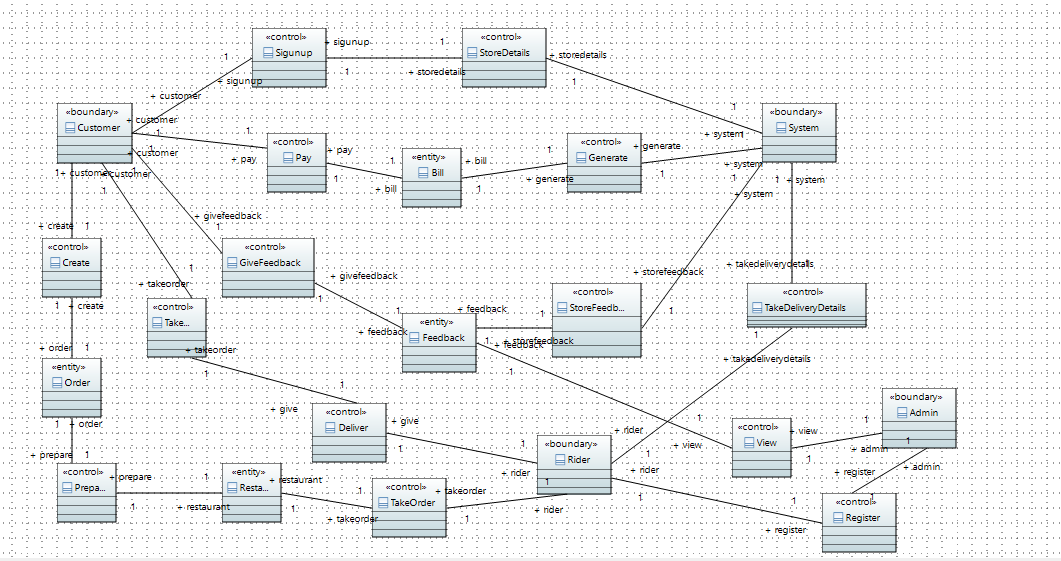
.

9.1.3.3. <Activity Diagram 3>

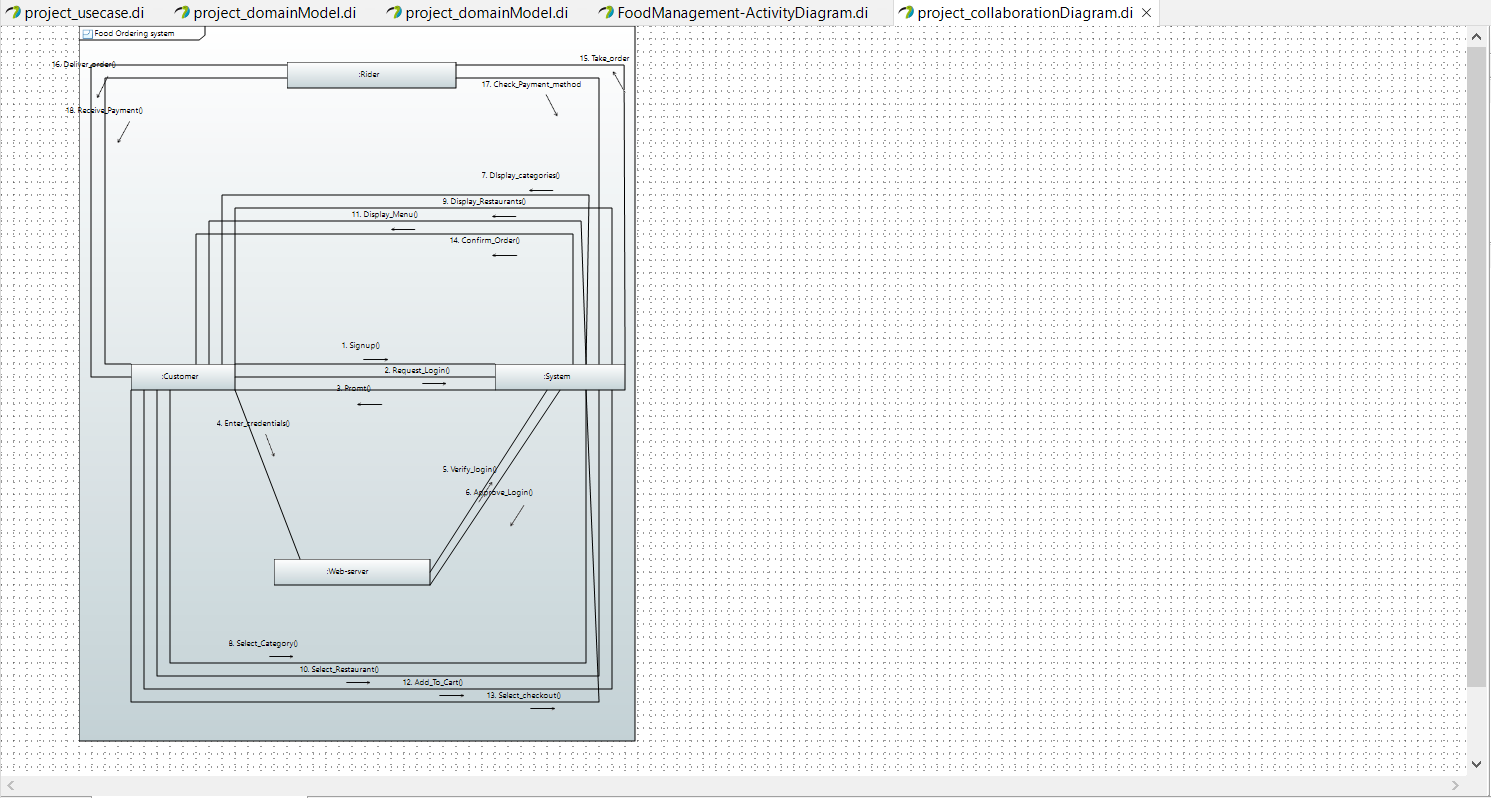
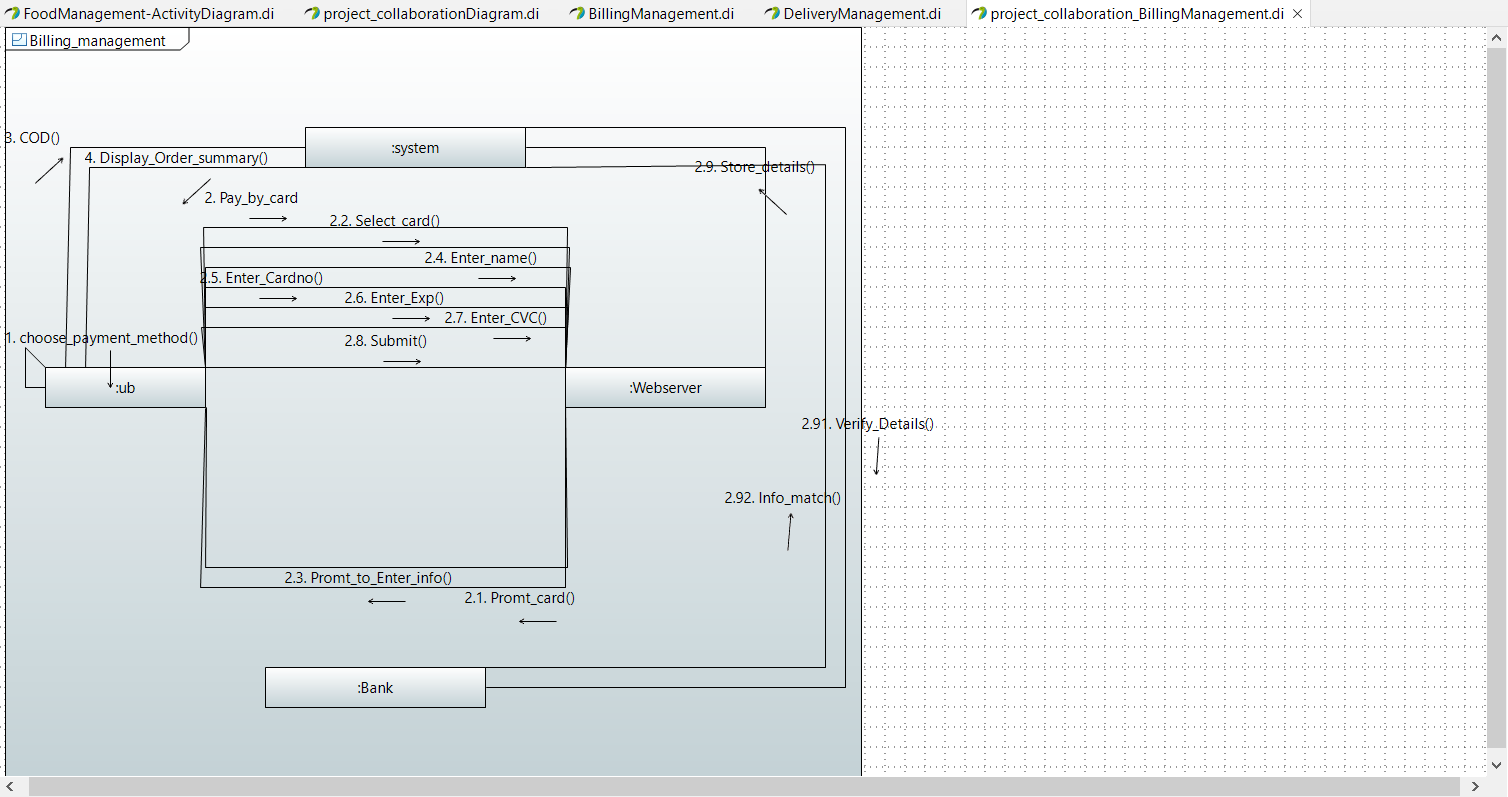
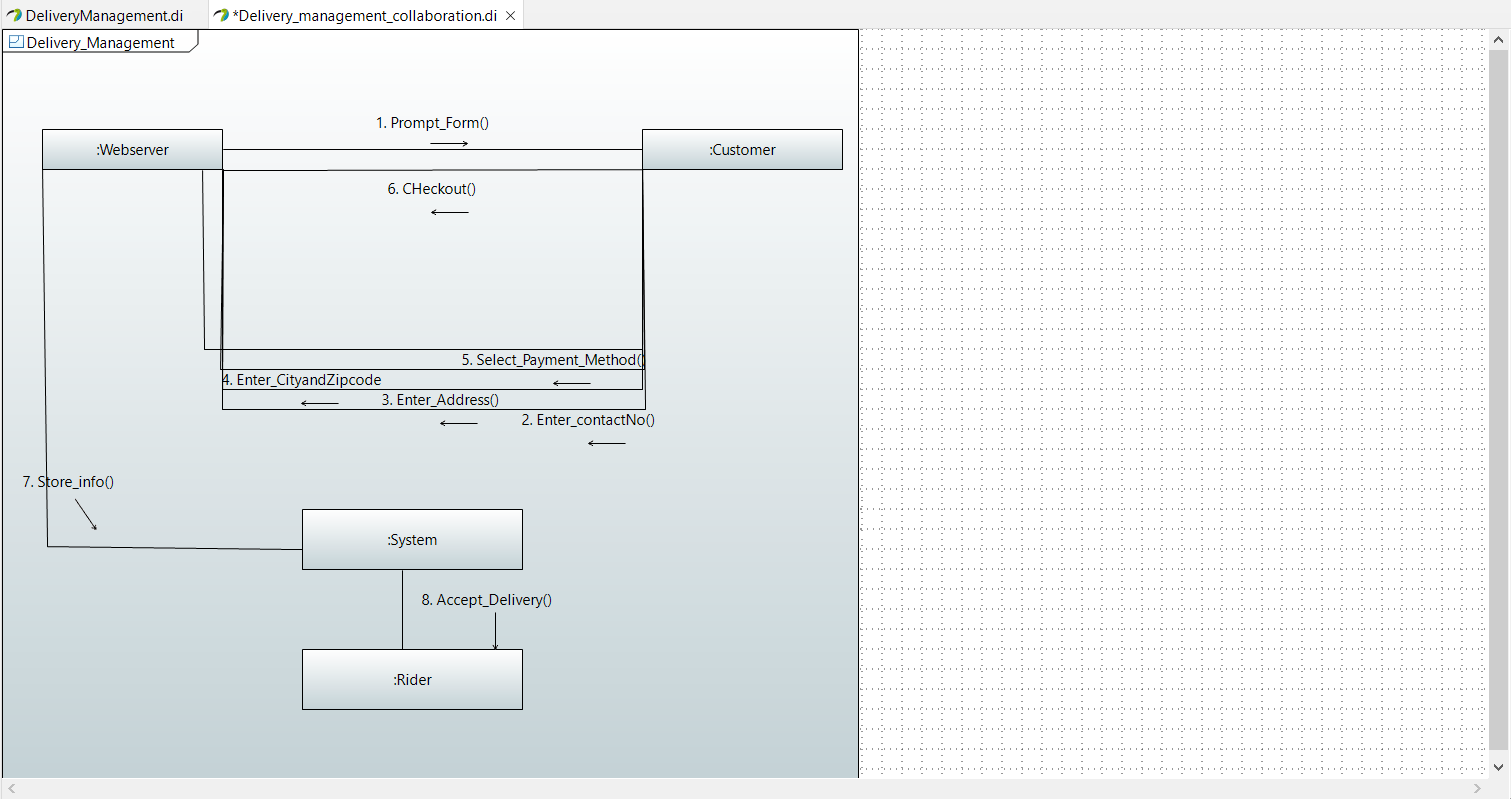


Explanation: this activity diagram manages the feature for order management; the customer first requests to login, where he enters his username, and password. Once logged in, the system displays the customer categories: the customer selects a category, then the system displays the customer restaurants available: the customer selects a restaurant, then the system displays the customer the menu offered by the selected restaurant. The customer selects an item, adds it in the cart. Once done, the customer selects checkout, and hence the system confirms the order. Order details, as well as customer details are given to the rider. The rider delivers the order, the customer pays, and the rider receives the payment.

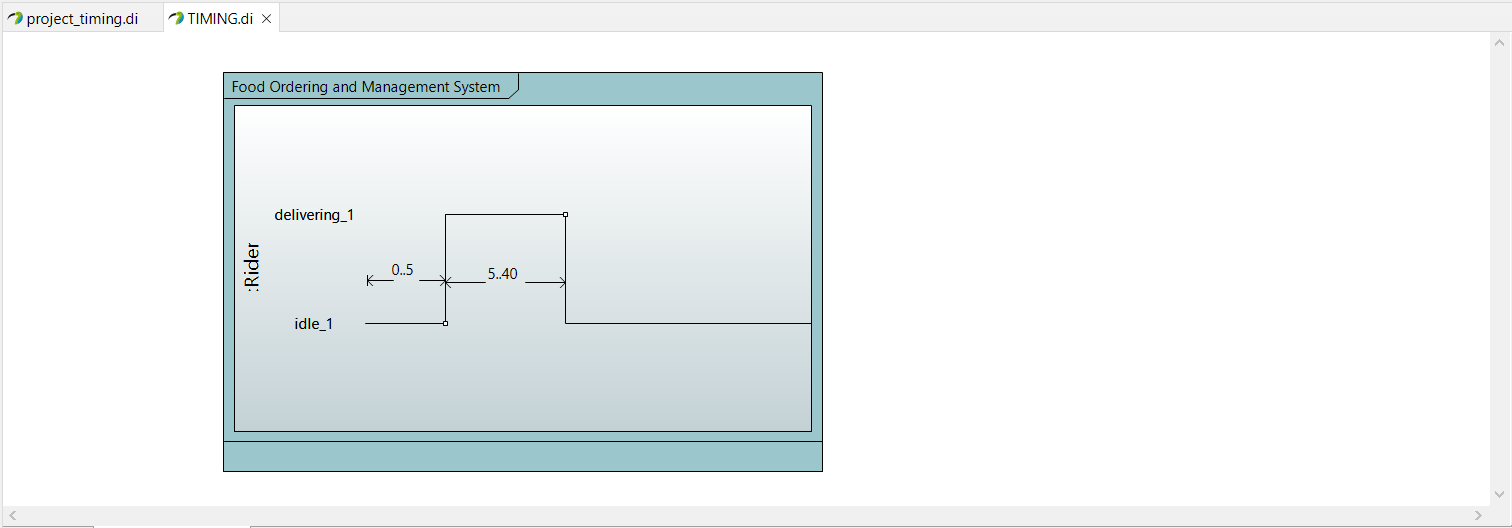
**Entity Class Diagram:**

****

**Collaboration Diagram:**

1. 
2. 
3. 

**Timing Diagram:**



## References

Ronacher, A., 2017. Flask documentation. *Retrieved August*, *15*, p.2018.

Grinberg, M., 2021. Flask-socketio documentation. *línea]. Disponible en: https://flask-socketio. readthedocs. org/en/latest/.[Último acceso: 2015]*.

https://jinja.palletsprojects.com/en/3.0.x/

https://getbootstrap.com/docs/3.3/getting-started/

https://flask.palletsprojects.com/en/2.0.x

## Appendices

Not Applicable.