# PRJ566 – Summer 2020

# PRJ566 – Team No: 5

# Name of Project:  Innovative food delivery service

# Project Leader: Junjie Zhang

**Last updated: 2020-06-07**

**Team Members:**

**1. Gia Dung Tran**

**2. Jaehyun Joung**

**3. Junjie Zhang**

**4. Thanh Trung Nguyen**

# TABLE OF CONTENTS

1. **Introduction/Overview - Document Information**
   1. **Document Authors**
   2. **Revision History**
   3. **Document Conventions**
   4. **Document Purpose**
   5. **Intended Audience**
   6. **Acronyms & References**
   7. **Group Agreement**
2. **Project Overview**
   1. **Project Proposal**
   2. **Project Detailed Scope & Functionality**
   3. **Stakeholders and Users**
   4. **Business Opportunity**
   5. **Risks**
   6. **Constraints**
   7. **Operating Environment**
   8. **Operational, Performance & Security Requirements**
3. **Process & Data Modeling**
   1. **Business Rules**
   2. **UML/DFD Modeling & Data Modeling**
   3. **Interface Mock-ups**
4. **Database**
   1. **Scripts to create, populate, delete tables**
      1. **Scripts to create tables**
      2. **Scripts to populate tables**
      3. **Scripts to delete tables**
   2. **Data Dictionary**
5. **Implementation Schedule**
   1. Work Breakdown Structure
   2. Implementation Schedule for PRJ666 - using Microsoft Project
6. **Measurable deliverables**
7. **Acceptance Criteria**
8. **Client / Faculty Sign-off**

# 1 - Introduction/Overview - Document Information

## 1.1 Document Authors

**Gia Dung Tran Jaehyun Joung Junjie Zhang Thanh Trung Nguyen**

## 1.2 Revision History

|  |  |
| --- | --- |
| Week 03 | Completed 1.1, 1.3, 1.4,1.5,1.7, 2.1  On going 1.2 1.6 |
| Week 04 | Completed 1.2 1.6 |
| Week 05 | Completed 2.2  On going 2.3 3.1 3.2 3.3 |
| Week 06 | Break week |
| Week 07 | Completed 2.3 3.1 3.2 3.3 |
| Week 08 | Rewrite 1.5 |
| Week 09 | Completed 4.1 4.1.1 4.1.2 4.1.3 |
| Week 10 | Completed 5.1 |
| Week 11 | Completed 5.2 |
| Week 12 | Rewrite 3.3 |
| Week 13 | Rewrite 3.1 |
| Final | Completed 1.6 |

## 1.3 Document Conventions

Any text highlighted in red indicates an exception or error

Anything highlighted in blue is in-progress

Any text highlighted in yellow is an important point.

Any text highlighted in green was recently added.

Any text *italicized* represents definitions.

Any text with ~~strike-through~~ is deleted.

## 1.4 Document Purpose

**This document provides the information of our project which is called “project name”. “Project name” is consisting of mobile and web. The document will work the application’s data and manual. This will include stake holders, resources and references.**

## 1.5 Intended Audience

**People who live alone**

**The project will work food making with delivery. Almost people who have family usually make a food during each mealtime. However, solo living people sometimes do not want to make a food for mealtime. Those people are the project’s potential customer because the project includes food making and delivery. Thus, intended audience is people who live alone.**

## 1.6 Acronyms & References

|  |  |
| --- | --- |
| **Delivery** | the action of delivering letters, packages, or ordered goods. |
| **Uber eats** | The one kind of food delivery to start the project motive |
| **Food Ingredients** | substance that is added to a food to achieve a desired effect. |

## 1.7 Group Agreement

**TEAM AGREEMENT**

**Team #: 5**

**Project Title: Innovative food delivery service**

**Project Time Frame:**

2020-05-26~ 2020-12-11

**Team Members:**

Gia Dung Tran

Jaehyun Joung

Junjie Zhang

Thanh Trung Nguyen

**Team Leadership:**

Junjie Zhang

**Team Functions:**

* *We will share information through MS Teams, OneDrive, WhatsApp, e-mail and meetings.*
* *At least two members Attend team online meetings on a regular basis and participate in discussions*
* *We will Complete assigned tasks, progress reports, and other documents needed*
* *The team will review the work is done and information to be added to the final document on Wednesday from 9:30 PM to Sunday afternoon (05:00 PM)*
* *The team leader will upload the document on Sunday before 11:30 PM*

**Team Meetings: 10:00 pm Wednesday every week**

**10:00 pm Saturday every week**

If our member misses meeting, we will have small meeting to share meeting information at 05:00 PM Sunday.

**Team Problems:**

* Members have different time schedules.
* Due to the COVID-19 pandemic, team members can only have online meetings.
* The meetings depend so much on the internet connection that sometimes the meetings cannot go well.
* Some of the team members have their part time jobs and other schedules that sometimes they cannot join the meetings on time.

**Team Commitment**

**The undersigned members agree to work together on the project until the end of the PRJ666 next Semester. They recognize that as a team and individually they are responsible for the quality of all deliverables.**

**Name Date**

|  |  |
| --- | --- |
| Gia Dung Tran | 2020-06-05 |
| Jaehyun Joung | 2020-06-05 |
| Junjie Zhang | 2020-06-05 |
| Thanh Trung Nguyen | 2020-06-05 |

ShapeShapeShapeShapeShapeShapeShapeShape

# 2 - Project Overview

## 2.1 Project Proposal

Business Type: Food delivery and Food manufacture complex.

Overview: Our business focuses on providing affordable and good quality daily meal plans for the general public. To serve 3 meals a day for our clients, we prepare and pack meals in the morning at our disposal center. Then, the delivery team will send out the food before 5 pm. At the same time, the delivery team will also collect the tableware back. And this business model will adapt the Subscription system.

Problem statement

What is problem and solution?

The problem is how to identify the client’s taste flavor. Toronto is a multicultural city. Also, some people have allergy. Also, our project’s goal is that client gets healthy food through our suggestion. The solution is that our application will show different food every day and client will choose get food or not. Also, the application should notice to nutrition fact for prevention of allergy.

Urban people usually do not have proper three meals a day because cooking a good meal sometimes is too hard to finish to a certain amount of people like businessmen, college students and people who do not have proper facilities to cook. These kinds of people usually will rely on food delivery service. However, the normal food delivery service is not economically friendly, environmentally friendly, and healthy during the pandemic because the delivery service will cost extra and increase unnecessary touch. So, a new type of delivery service is needed. The aim of this project is going to provide a more economical, environmentally friendly, and touching free delivery service.

Why this service?

Our target clients are those people who do not want to cook and cannot afford Uber every day. Our business model has huge advantages comparing with Uber. First, we control what our customers eat, this process will lower our manufacturing cost because we make 1000 the same kinds of foods for different 1000 people per day, which will cost much lower than making 1000 different foods for 1000 different people. Second, the company will control the future menu for customers either, which will also help us to cut costs because we control the menu, we control what we will, we control the food materials, then we will have more bargain power in the market. Last, this business model will use a subscription system, customers will need to pay for a week at least. This charging model will help us to maintain healthier finance.

Background: This is an Asian based company.

The need: For delivering our service properly, we need two systems such as a subscription service management system, and a delivery management system. Meanwhile, our business will need many local restaurant contractors, food suppliers, and a delivery team.

Objectives:

* Serving a good amount of various food within a month.
* Maintain a good amount of stable customer size.
* Expand the service from residents to schools and hospitals.

Is this a substantial project?

Yes, this is a substantial project. Because this project contains two critical systems which are online management systems and delivery systems, both systems are really popular now. Even if the project was not able to continue after the end of next semester. Our team can still benefit lots of experience from this project, which will help us in future employment.

## 2.2 Project Detailed Scope & Functionality

## 

**Include the Functional and Nonfunctional Requirements**

**Functional Requirements:**

1. Manage Groceries Purchasing:
   1. Purchase groceries from suppliers.
2. Manage Meal Selling:
   1. Customer subscribes for meals delivery.
   2. Customer chooses meals for each day of their subscription.
   3. Website lists ordered items from customer.
   4. Website asks customer to enter their address for delivery.
   5. Website informs customer the total cost.
   6. Website asks customer to pay.
   7. List payment methods for customer to choose.
   8. Customer chooses payment method and inputs their payment information.
   9. Website sends invoice to customer.
3. Manage Payment Subscriptions:
   1. Verify payment information and process payment.
   2. Pass through payment data to financial service.
   3. Choose type of subscriptions (2 weeks, month(s) or a year...).
4. Manage Meals delivery:
   1. Android application displays the faster route for the driver.
   2. Android application displays the food list and next available address.
   3. Verification of delivery

**Non-functional Requirements:**

1. Access for customers, employees, and management people must be carried out through a website.
2. All deliveries must be finished before 6:30pm.
3. All package must be sealed. Once, the delivery man finds any food leakage, recall the meal.

## 2.3 Stakeholders and Users

**Purchase Manager**

**Delivery Manager**

**Database Manager**

**Website**

**Web admin**

**Supplier**

**Cook**

**Customer**

**Credit card operator**

**Developer**

**Delivery Manager**

## 2.4 Business Opportunity

## 2.5 Risks

|  |  |
| --- | --- |
| **Risk** | **Response** |
| The food delivery already exists | The project use subscription to show different Strategy. Customer also can get food more cheap than other delivery service |
| The cook supply needs and food taste | The project members will provide food through cooking. Before the project starts, each member survey food to potential customer. |
| Potential customers are various. | Project will start narrow target customer and will be wide later. |
| COVID-19 | It is biggest issue about the project. Delivery service man takes mask to prevent COVID-19 during delivery |
|  |  |
|  |  |

## 2.6 Constraints

**The delivery service can do at the first time.**

The producing food needs a lot of grocery.

## 2.7 Operating Environment

## 

## Students who live alone do not want to make food. Also, they use delivery service frequently. Also, they want to eat homestyle food.

## 2.8 Operational, Performance & Security Requirements

The project web needs safety. Also, each food ingredient is certificated as safety.

# Process and Data Modeling

## **3.1 Business Rules**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BR ID | Business Rule Title | Business Rule Description | Type (fact, computation, constraint, other) | References |
| BR01 | No special characters in password. | When customers register accounts, their passwords can have any special characters such as “!@#$%^&\*”. | Constraint | See the account setup policy. |
| BR02 | Only managers can access customers’ profiles. | Normal employees are not allowed to access any clients’ profiles, except the department managers. | Constraint | See the corporate management policy. |
| BR03 | No PII in recipe Email | Don’t recipe any email containing personally identifiable information (PII). | Constraint | See the corporate email policy. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BR04 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## **3.2 UML/DFD Modeling and Data Modeling**

### Use the appropriate UML Modeling diagrams and/or Data Flow diagram for your

### Project

Insert here corresponding Mockup screen(s) for each Use Case Specification

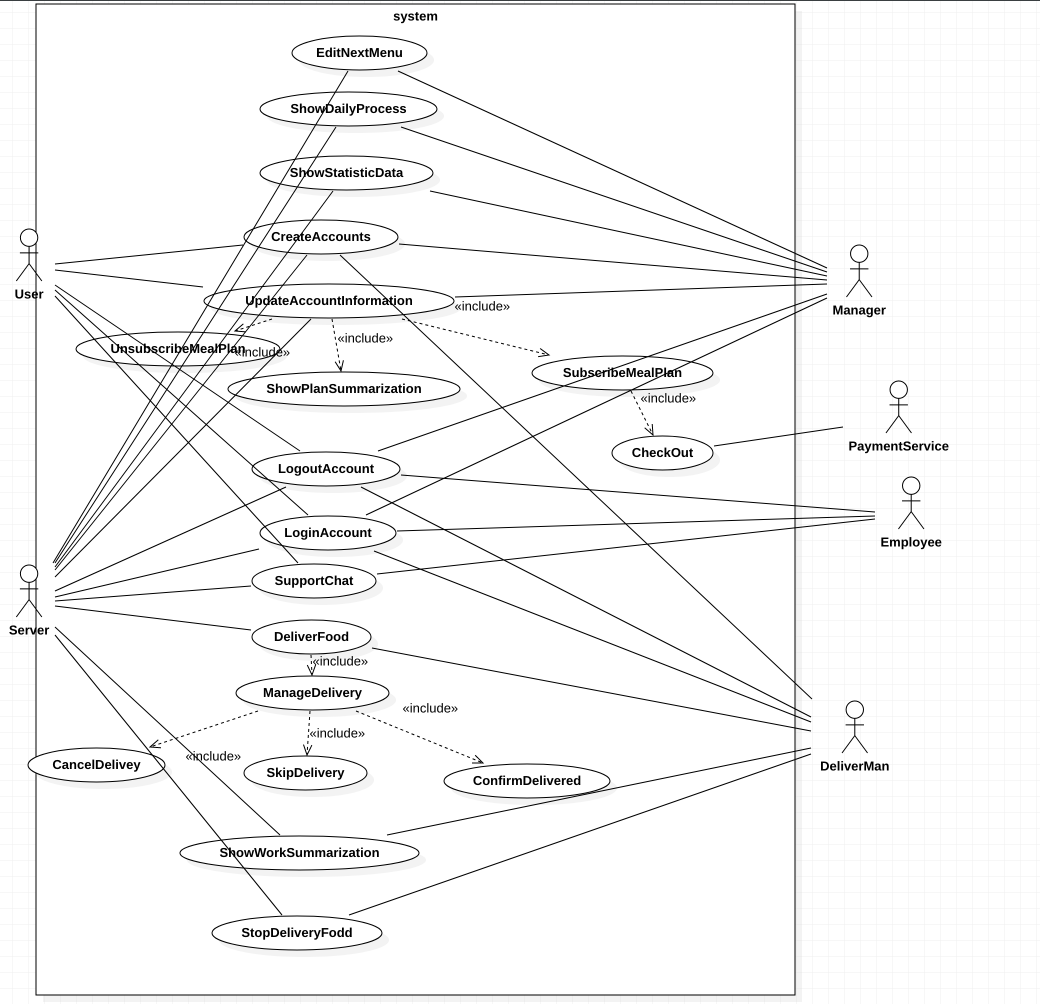
This business use case has two parts.

The first part is the client-end.

The Second part is the Driver-end.

The Last part is the manager-end.

The business use case diagram version-1.



Use case specification

* Casual Use Case Specification: UC1 EditNextMenu
* **Brief Description:**
* this use case enables the manager edits the food menu;

# Section 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The manager edits the new menu for next week**

# Preconditions

* The manager has successfully logged in the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( Manager )** | **System** | **Data Used** |
| 1 | The manager clicks on the “Edit new menu” options | The system will turn to the new page | none |
| 2 | Do nothings | The system will show an fillable form | none |
| 3 | The manager input the new menu | The system will validate the new menu | none |
| 4 | The manager presses save button | The system will save the menu data to the server | Menu data |

* **Successful Post-Conditions:**
* The system will store the data for the next week menu

# Casual Use Case Specification: UC2 ShowDailyProcess

* **Brief Description:**
* this use case enables how the manager sees the work process;

# Section 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The manager want to check the daily sales volume, and progress.**

# Preconditions

* The manager has successfully logged in the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( Manager )** | **System** | **Data Used** |
| 1 | The manager clicks on the “ShowDailyProcess” options | The system will turn to the new page | none |
| 2 | Do nothings | The system will show Pie charts | Sells Data |

* **Successful Post-Conditions:**
* The user can see the sales volume today.

# Casual Use Case Specification: UC3 ShowStatisticData

* **Brief Description:**
* this use case enables how the manager see the relevant statistic data;

# Section 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The manager wants to see the financial data**

# Preconditions

* The manager has successfully logged in the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( Manager )** | **System** | **Data Used** |
| 1 | The manager clicks on the “ShowStatisticData” options | The system will turn to the new page | none |
| 2 | Do nothings | The system will show all the financial data | Sells Data |

* **Successful Post-Conditions:**
* The user can see the financial data.

# Casual Use Case Specification: UC4 CreateAccount

* **Brief Description:**
* this use case enables how the user register an accoutn;

# Section 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The client wants to create an account**

# Preconditions

* The client visits our login page

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( user)** | **System** | **Data Used** |
| 1 | The client clicks on the “register” options | The system will turn to the new page | none |
| 2 | Do nothings | The system will show the registration page | None |
| 3 | The client inputs its personal data | The system will validate the input data | Personal data |
| 4 | Do nothings | The system will save all personal data into the database if the input data passed validation. | None |
| 5 | Do nothings | The system will turn to the user profile page. | None |

* **Successful Post-Conditions:**
* The user finishes registration.

# Casual Use Case Specification: UC5 UpdateAccountInformation

* **Brief Description:**
* this use case enables how the user updates their personal information;

# Section 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The client wants to update an account**

# Preconditions

* The client logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( user )** | **System** | **Data Used** |
| 1 | Do nothings | The system will turn to the profile page | none |
| 2 | Adjust the personal data on the profile page | The system will validate the new personal data | Personal data |
| 3 | Press the “save” button | The system will save the new personal data if the data passed the validation. | none |

* **Successful Post-Conditions:**
* The user updated his profile.

# Casual Use Case Specification: UC6 Subscribe Meal plan

* **Brief Description:**

# this use case enables how the user updates their personal information;

# Section 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The client wants to subscribe a meal plan**

# Preconditions

* The client logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( user )** | **System** | **Data Used** |
| 1 | Do nothings | The system will turn to the profile page | none |
| 2 | Press the “subscribe” button | The system will turn to the subscription page. | none |
| 3 | Choose the favorite meal plan and duration | The system will save the new meal plan request | Meal plan information |
| 4 | none | The system will return to the main page and save the process | none |

* **Successful Post-Conditions:**
* The user subscribes a meal plan.

# Casual Use Case Specification: UC7 Unsubscribe Meal plan

* **Brief Description:**

# this use case enables how the user unsubscribe their meal plans;Section

# 

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The client wants to unsubscribe a meal plan**

# Preconditions

* The client logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( user )** | **System** | **Data Used** |
| 1 | Do nothings | The system will turn to the profile page | none |
| 2 | Press the “unsubscribe” button | The system will turn to the unsubscription page. | none |
| 3 | Choose the meal plan that you want to cancel | The system will cancel the meal plan delivery | Meal plan information |
| 4 | none | The system will return to the main page and save the process | none |

* **Successful Post-Conditions:**
* The user unsubscribes a meal plan.

# Casual Use Case Specification: UC8 ShowPlanSummarization

* **Brief Description:**

# this use case enables how the user understands their meal plans;

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The client wants to show the meal plan and subscription summarization**

# Preconditions

* The client logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( user )** | **System** | **Data Used** |
| 1 | Do nothings | The system will turn to the profile page | none |
| 2 | Press the “ShowPlanSubscription” button | The system will turn to the summarization page. | none |
| 3 | Do nothings | The system will show the plan summarization | Meal plan information |

* **Successful Post-Conditions:**
* The user saw the plan summarization

# Casual Use Case Specification: UC8 LogoutAccount

* **Brief Description:**
* this use case enables how the user logout;

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The client wants to logout**

# Preconditions

* The client logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( user )** | **System** | **Data Used** |
| 1 | Press the logout button | The system will logout | none |

* **Successful Post-Conditions:**
* The user logged out

# Casual Use Case Specification: UC10 LoginAccount

* **Brief Description:**
* this use case enables how the user logs in;

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The client wants to logout**

# Preconditions

* none

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( user )** | **System** | **Data Used** |
| 1 | Do nothings | The system will load the login page | none |
| 2 | Input its account number and password. | none | none |
| 3 | Press the login button | The system will validate the input data | Account and password |
| 4 | Do nothings | The system will turn to the profile page if the input information can pass the validation. | none |

* **Successful Post-Conditions:**
* The user logged in

# Casual Use Case Specification: UC11 SupportChat

* **Brief Description:**

# this use case enables how the user contact customer support;

# 

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The client wants to contact customer support**

# Preconditions

* The user logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( user )** | **System** | **Data Used** |
| 1 | Clicks on the support button | The system will turn to the support page | none |
| 2 | Do nothings | The system will show the support phone number | Support number |

* **Successful Post-Conditions:**
* The user can use the phone number contacting our manager

# Casual Use Case Specification: UC12 DeliverFood

* **Brief Description:**
* this use case enables how the driver begin to deliver;

# 

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The delivery man wants deliver food**

# Preconditions

* The user logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( driver )** | **System** | **Data Used** |
| 1 | Clicks on the “deliver food” button at the main page | The system will turn to the delivery | none |
| 2 | Do nothings | The system will show the delivery information | Delivery information |

* **Successful Post-Conditions:**
* The user can use the phone number contacting our manager

# Casual Use Case Specification: UC14 ManageDelivery

* **Brief Description:**
* this use case enables how the driver manages his delivery;

# 

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The delivery man wants deliver food**

# Preconditions

* The user logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( driver )** | **System** | **Data Used** |
| 1 | Clicks on the “deliver food” button at the main page | The system will turn to the delivery | none |
| 2 | Do nothings | The system will show the delivery information | Delivery information |
| 3 | Clicks on the “manageDelivery” button | The system will turn to the “ManageDelivery” page | no |

* **Successful Post-Conditions:**
* The user can use the “manageDelivery” features.

# Casual Use Case Specification: UC15 cancelDelivery

* **Brief Description:**
* this use case enables how the deliver cancels his delivery;

# 

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The delivery man wants to cancel delivering food**

# Preconditions

* The user logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( driver )** | **System** | **Data Used** |
| 1 | Clicks on the “deliver food” button at the main page | The system will turn to the delivery | none |
| 2 | Do nothings | The system will show the delivery information | Delivery information |
| 3 | Clicks on the “manageDelivery” button | The system will turn to the “ManageDelivery” page | no |
| 4 | Clicks on the cancel button to cancel the current delivery | The system will show the cancellation alart | no |
| 5 | Do nothings | The system will turn back to the “startDelivery” page | no |

* **Successful Post-Conditions:**
* The user finishes a cancellation process.

# Casual Use Case Specification: UC16 SkipDelivery

* **Brief Description:**
* this use case enables how the deliver skips his current delivery;

# 

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The delivery man wants to skip the current delivery**

# Preconditions

* The user logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( driver )** | **System** | **Data Used** |
| 1 | Clicks on the “deliver food” button at the main page | The system will turn to the delivery | none |
| 2 | Do nothings | The system will show the delivery information | Delivery information |
| 3 | Clicks on the “manageDelivery” button | The system will turn to the “ManageDelivery” page | no |
| 4 | Clicks on the skip button to cancel the current delivery | The system will show the skip alart | no |
| 5 | Do nothings | The system will store the current delivery information to waitlist | Delivery information |
| 6 | Do nothings | The system will turn back to the “startDelivery” page. | no |

* **Successful Post-Conditions:**
* The driver skips one delivery.

# Casual Use Case Specification: UC17 ConfirmDelivery

* **Brief Description:**
* this use case enables how the drive confirms the current delivery;

# 

# 1: Business Rule(s):

* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The delivery man wants to confirm the current delivery**

# Preconditions

* The user logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( driver )** | **System** | **Data Used** |
| 1 | Clicks on the “deliver food” button at the main page | The system will turn to the delivery | none |
| 2 | Do nothings | The system will show the delivery information | Delivery information |
| 3 | Clicks on the “manageDelivery” button | The system will turn to the “ManageDelivery” page | no |
| 4 | Clicks on the confirmation button to cancel the current delivery | The system will show the confirmation alart | no |
| 5 | Do nothings | The system will turn back to the “startDelivery” page | no |

* **Successful Post-Conditions:**
* The driver skips one delivery.

# Casual Use Case Specification: UC18 ShowWorkSummarization

* **Brief Description:**
* this use case enables how the drive see the work progress;
* 1: Business Rule(s):
* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The delivery man wants to see the work progress**

# Preconditions

* The user logged in.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( driver )** | **System** | **Data Used** |
| 1 | Clicks on the “deliver food” button at the main page | The system will turn to the delivery | none |
| 2 | Do nothings | The system will show the delivery information | Delivery information |
| 3 | Clicks on the “manageDelivery” button | The system will turn to the “ManageDelivery” page | no |
| 4 | Clicks on the “progress” button to cancel the current delivery | The system turns to “progress” page | no |

* **Successful Post-Conditions:**
* The driver can see the work progress

# Casual Use Case Specification: UC19 StopDelivery

* **Brief Description:**
* this use case enables how the drive stops the delivery job
* 1: Business Rule(s):
* None;

# Section 2: Scenarios (HD):

* **Scenario 1: The delivery man wants to see the work progress**

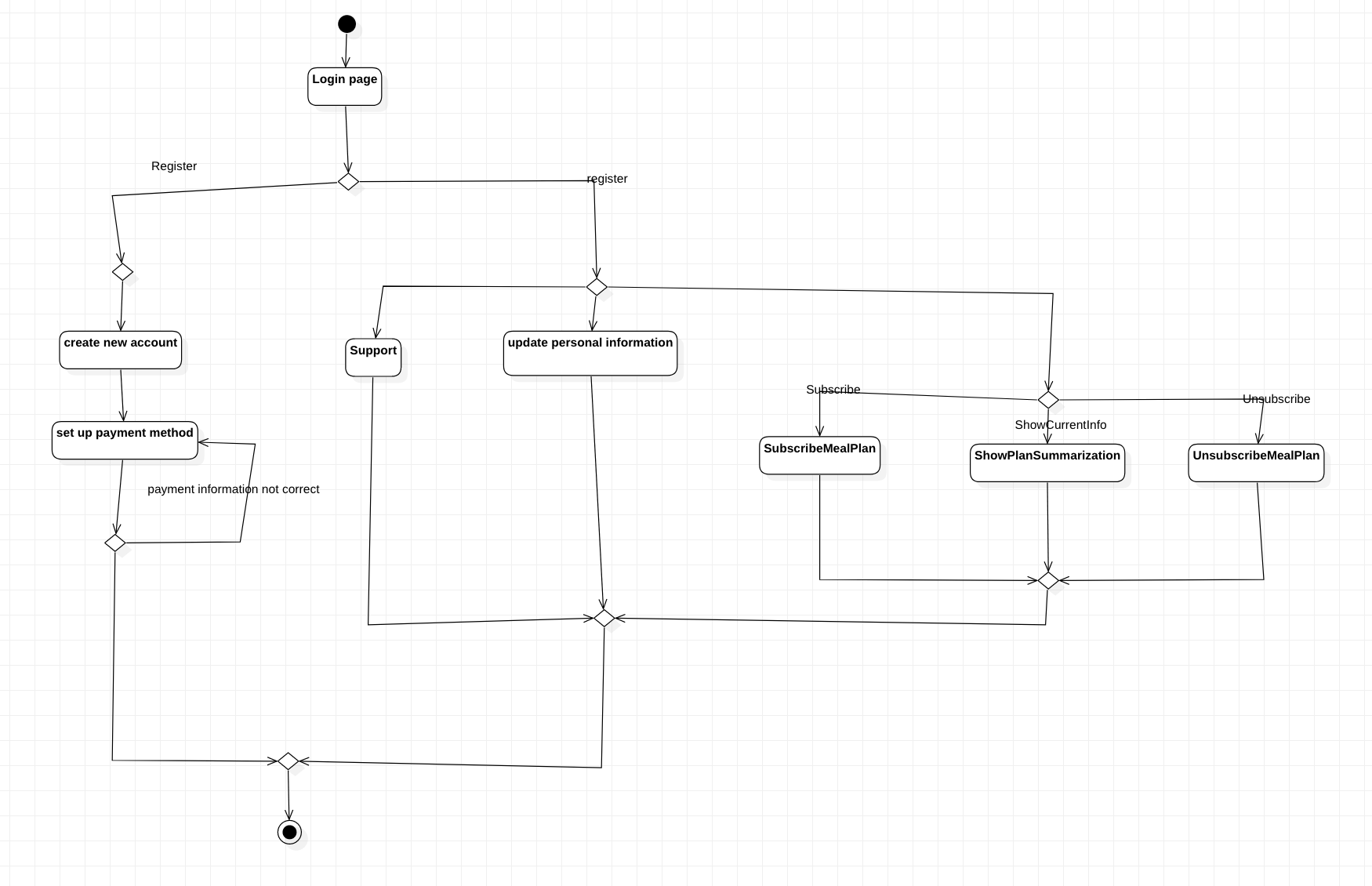
# Preconditions

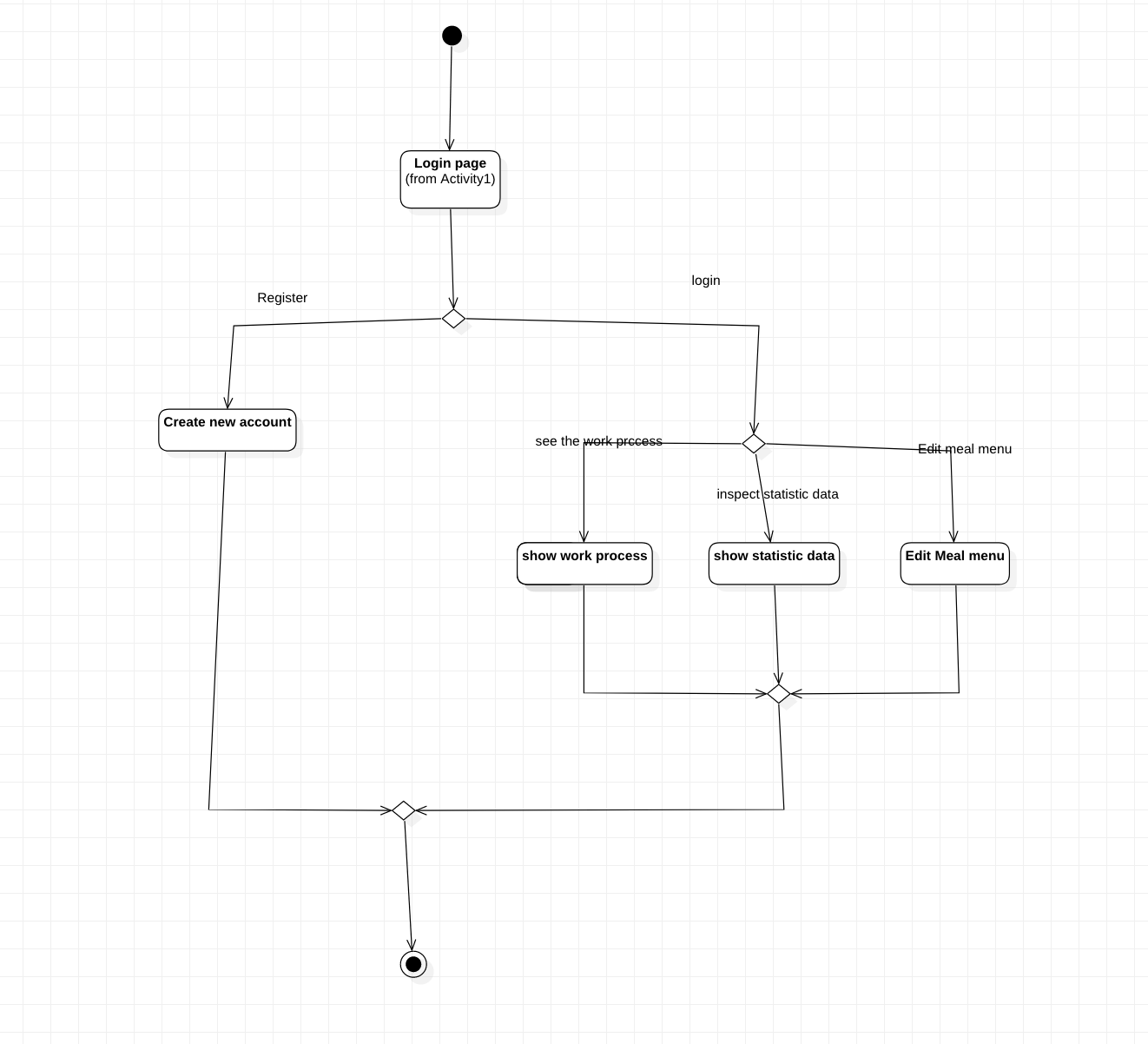
* The user logged in.

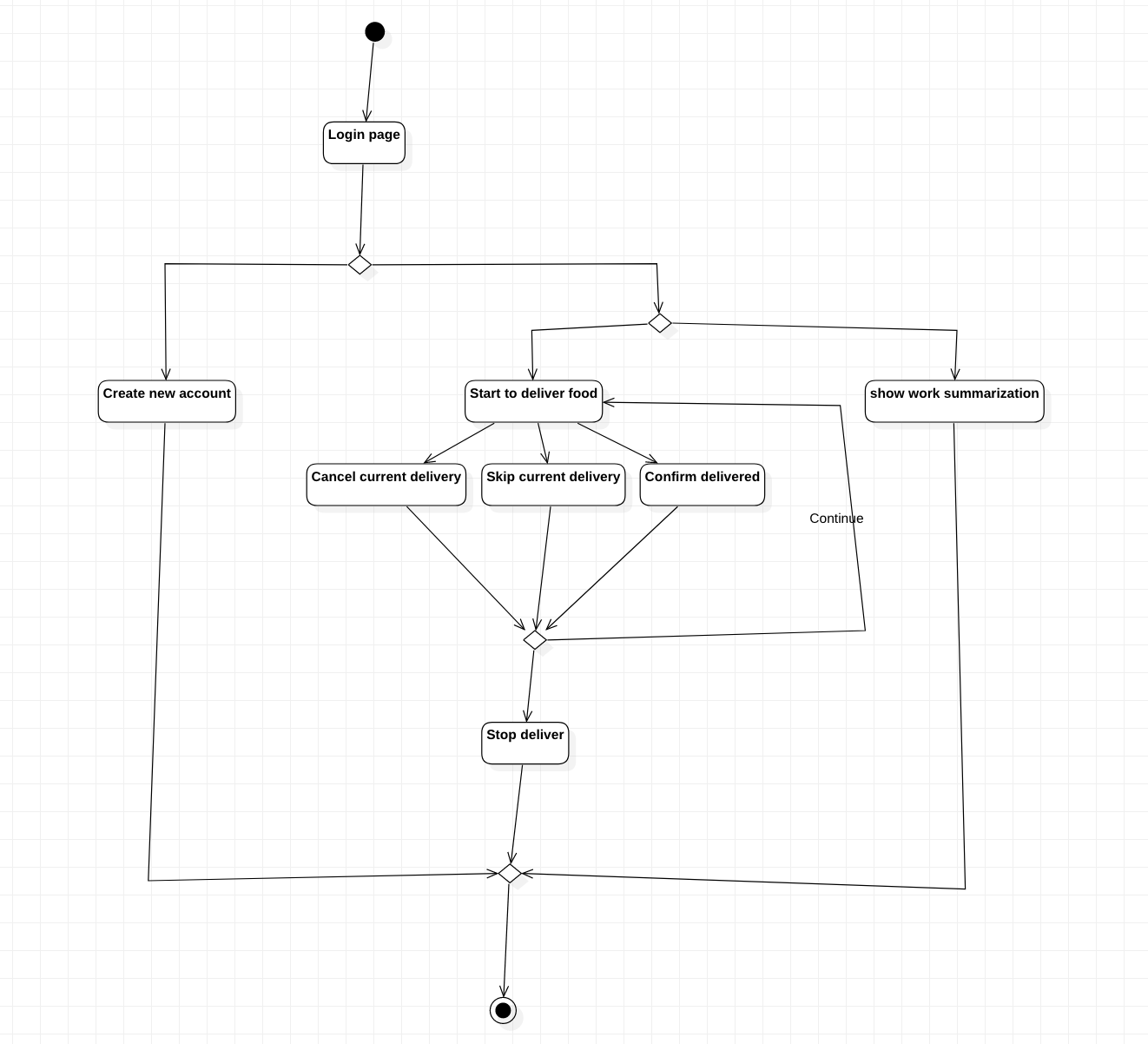
|  |  |  |  |
| --- | --- | --- | --- |
| **Step#** | **Actor ( driver )** | **System** | **Data Used** |
| 1 | Clicks on the “deliver food” button at the main page | The system will turn to the delivery | none |
| 2 | Do nothings | The system will show the delivery information | Delivery information |
| 3 | Clicks on the “manageDelivery” button | The system will turn to the “ManageDelivery” page | no |
| 4 | Clicks on the “stop” button to cancel the current delivery | The system will show the “cancellation” alert | no |
| 5 | Do nothings | The system will turn back to the main page. |  |

* **Successful Post-Conditions:**
* The driver stop the delivery.

The active case diagram and description







User Case Name: EditNextMenu ID:1 Important level: high

Primary actor: Manager User case type: Detail, essential

Stackholders ahd interests:

Manager-want to set the next food menu

Computer server-want to ensure the manager can input the new menu;

Brief description: this use case describe how the manager edits the food menu;

Trigger: when the manager clicks on the dedicated button in his profile.

Type: internal

Relationship:

Association:

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The manager logs in.
2. The manager clicks on the dedicated button on the homepage.
3. The system will show a dedicated page
4. The manager will input the meal plan details on the form chart
5. The system will validate the input
   1. If the validation is passed, the system will save the data and returns back to the main page
   2. If the validation isn’t passed, the system will show error messages, and resets the page.

User Case Name: ShowDailyProcess ID:2 Important level: high

Primary actor: Manager User case type: Detail, essential

Stackholders ahd interests:

Manager-want to see how much meals have been sent

Computer server-want to ensure the manager checks certain information;

Brief description: this use case describe how the manager sees the work process;

Trigger: when the manager clicks on the dedicated button in the homepage.

Type: internal

Relationship:

Association:Manager, Server

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The manager logs in
2. The manager clicks on the dedicated button in the homepage
3. The system will show the work process.

User Case Name: ShowStatisticData ID:3 Important level: high

Primary actor: Manager User case type: Detail, essential

Stackholders ahd interests:

Manager-want to see the relevant statistic data.

Computer server-want to ensure the manager can see the data;

Brief description: this use case describe how the manager see the relevant statistic data;

Trigger: when the manager clicks on the dedicated button on the homepage.

Type: internal

Relationship:

Association: Manager, Server

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The manager logs in
2. The manager clicks on the dedicated button in the homepage
3. The system will show the relevant statistic data.

User Case Name: CreateAccount ID:4 Important level: high

Primary actor: Manager,Driver,customer User case type: Detail, essential

Stackholders ahd interests:

Manager-want to register an account.

Customer-want to register an account.

Driver-want to register an account.

Computer server-want to ensure the user can see the data;

Brief description: this use case describe how the user register an accoutn;

Trigger: when the user clicks the registration button at the login page.

Type: internal

Relationship:

Association: Manager, Server, driver, customer

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The user clicks the registration button at the login page
2. The system will show the registration page, and the form chart.
3. The user input the personal information at the form chart.
4. The system will validate the personal information
   1. If the personal information passed the validation, the system will return to the home page.
   2. If the personal information doesn’t pass the validation, the system will show the error and refresh the page.

User Case Name: UpdateAccountInformation ID:5 Important level: high

Primary actor: Manager,Driver,customer User case type: Detail, essential

Stackholders ahd interests:

Manager-want to set personal information and subscription.

Customer-want to set personal information and subscription.

Driver-want to set personal information and subscription.

Computer server-want to ensure the user can adjust their personal information and subscription;

Brief description: this use case describes how the user updates their personal information;

Trigger: when the manager clicks on the dedicated button in his profile.

Type: internal

Relationship:

Association: Manager, Server, driver, customer

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The user logs in.
2. The user clicks on the dedicated button on the homepage.
3. The system will show a dedicated page
4. The user will change their personal information at the chart form
5. The system will validate the personal information
   1. If the personal information passes the validation, the system will show the successful information.
   2. If the personal information doesn’t pass the validation, the system will show the failure information as well as refresh the page.

User Case Name: Subscribe Meal plan ID:6 Important level: high

Primary actor: customer User case type: Detail, essential

Stackholders ahd interests:

Customer-want to set subscription.

Computer server-want to ensure the user can adjust their personal information and subscription;

Brief description: this use case describes how the user updates their personal information;

Trigger: when the manager clicks on the dedicated button in his profile.

Type: internal

Relationship:

Association: Server, customer

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The customer logs in.
2. The customer clicks on the dedicated button on the profile page
3. The system will turn to the meal plan selection page
4. The user will select meal plan and press the finish button to confirm.
5. The system will show the receipt and sent an email to the user.
6. The system returns to the homepage.

User Case Name: Unsubscribe Meal plan ID:7 Important level: high

Primary actor: customer User case type: Detail, essential

Stackholders ahd interests:

Customer-want to unsubscribe meal plan.

Computer server-want to ensure the user can adjust their personal information and Unsubscribe meal plan;

Brief description: this use case describes how the user unsubscribe their meal plans;

Trigger: when the manager clicks on the dedicated button in his profile.

Type: internal

Relationship:

Association: Server, customer

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The customer logs in.
2. The customer clicks on the dedicated button on the profile page
3. The system will turn to the meal plan selection page
4. The user will select meal plan to unsubscribe and press the finish button to confirm.
5. The system will show the receipt and sent an email to the user.
6. The system returns to the homepage.

User Case Name: ShowPlanSummarization ID:8 Important level: high

Primary actor: customer User case type: Detail, essential

Stackholders ahd interests:

Customer-want to see the meal plan’s details and delivery schedule.

Computer server-want to ensure the user can see the meal plan menu and delivery schedule;

Brief description: this use case describes how the user understands their meal plans;

Trigger: when the user clicks on the dedicated button in his profile.

Type: internal

Relationship:

Association: Server, customer

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The customer logs in.
2. The customer clicks on the dedicated button on the profile page
3. The system will show the delivery schedules and meal plan menu.

User Case Name: LogoutAccount ID:9 Important level: high

Primary actor: Manager,Driver,customer User case type: Detail, essential

Stackholders ahd interests:

Manager-want to logout.

Customer- want to logout.

Driver- want to logout.

Computer server-want to ensure the user can logout;

Brief description: this use case describes how the user logout;

Trigger: when the user clicks the logout button at the home page.

Type: internal

Relationship:

Association: Manager, Server, driver, customer

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The user clicks the logout button at the home page
2. The system will return back to the login page

User Case Name: LoginAccount ID:10 Important level: high

Primary actor: Manager,Driver,customer User case type: Detail, essential

Stackholders ahd interests:

Manager-want to login.

Customer- want to login.

Driver- want to login.

Computer server-want to ensure the user can login;

Brief description: this use case describes how the user logs in;

Trigger: when the user clicks the login button at the login page.

Type: internal

Relationship:

Association: Manager, Server, driver, customer

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The user clicks the login button at the home page
   1. If the password and account number is matched, the system will turns to the main page(profile page).
   2. If the password doesn’t match the account number, the system will show error messages.

User Case Name: SupportChat ID:11 Important level: high

Primary actor: Manager,server, customer User case type: Detail, essential

Stackholders ahd interests:

Manager-want to support customers.

Customer- want to get support.

Computer server-want to ensure the user can get support;

Brief description: this use case describes how the user logs in;

Trigger: when the user clicks the login button at the login page.

Type: internal

Relationship:

Association: Manager, Server,customer

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The user clicks the login button at the home page
   1. If the password and account number is matched, the system will turns to the main page(profile page).
   2. If the password doesn’t match the account number, the system will show error messages.
2. The user clicks at the support button the button at the main page
3. The system will turn to the support chat page, and reminds the manager to listen
4. The system will record down the chat, sends a record through email to the customer
5. The user clicks on the close button

User Case Name: DeliverFood ID:12 Important level: high

Primary actor: server, Driver User case type: Detail, essential

Stackholders ahd interests:

Driver-want to deliver food

Computer server-want to ensure the driver can deliver food;

Brief description: this use case describes how the deliver begin to deliver;

Trigger: when the user clicks the login button at the login page.

Type: internal

Relationship:

Association: Driver, server

Include:ManageDelivery

Exclude:

Generalization:

Normal Flow of events:

1. The driver logs in the system
2. The driver clicks on the “Start deliver” button at the main page
3. The system will turn to the “start delivery” page.

User Case Name: DeliverFood ID:13 Important level: high

Primary actor: server, Driver User case type: Detail, essential

Stackholders ahd interests:

Driver-want to deliver food

Computer server-want to ensure the driver can deliver food;

Brief description: this use case describes how the deliver begin to deliver;

Trigger: when the user clicks the login button at the login page.

Type: internal

Relationship:

Association: Driver, server

Include:ManageDelivery

Exclude:

Generalization:

Normal Flow of events:

1. The driver logs in the system
2. The driver clicks on the “Start deliver” button at the main page
3. The system will turn to the “start delivery” page.

User Case Name: ManageDelivery ID:14 Important level: high

Primary actor: server, Driver User case type: Detail, essential

Stackholders ahd interests:

Driver-want to manage delivery

Computer server-want to ensure the driver can manage delivery;

Brief description: this use case describes how the deliver manage delivery;

Trigger: when the user clicks the manage button at the main page.

Type: internal

Relationship:

Association: Driver, server

Include:CancelDelivery, ConfirmDelivery, SkipDelivery

Exclude:

Generalization:

Normal Flow of events:

1. The driver logs in the system
2. The driver clicks on the “Start deliver” button at the main page
3. The system will turn to the “start delivery” page.
4. The system will show the current deliver route and certain information
5. The driver clicks on the “ManageDelivery” button

User Case Name: CancelDelivery ID:15 Important level: high

Primary actor: server, Driver User case type: Detail, essential

Stackholders ahd interests:

Driver-want to cancel delivery

Computer server-want to ensure the driver can cancel delivery;

Brief description: this use case describes how the drive cancels delivery;

Trigger: when the user clicks the cancel button at the deliver management page.

Type: internal

Relationship:

Association: Driver, server

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The driver logs in the system
2. The driver clicks on the “Start deliver” button at the main page
3. The system will turn to the “start delivery” page.
4. The system will show the current deliver route and certain information
5. The driver clicks on the “ManageDelivery” button
6. The system will turn to the “ManageDelivery” page and show certain information
7. The Driver clicks on the “cancel delivery button”
8. The system will send the cancellation through an email to the client

User Case Name: SkipDelivery ID:16 Important level: high

Primary actor: server, Driver User case type: Detail, essential

Stackholders ahd interests:

Driver-want to cancel delivery

Computer server-want to ensure the driver can skip the current delivery;

Brief description: this use case describes how the drive skips the current delivery;

Trigger: when the user clicks the skip button at the deliver management page.

Type: internal

Relationship:

Association: Driver, server

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The driver logs in the system
2. The driver clicks on the “Start deliver” button at the main page
3. The system will turn to the “start delivery” page.
4. The system will show the current deliver route and certain information
5. The driver clicks on the “ManageDelivery” button
6. The system will turn to the “ManageDelivery” page and show certain information
7. The Driver clicks on the “skip delivery button”
8. The system will skip the current delivery task and store it to the waitlist

User Case Name: ConfirmDelivery ID:17 Important level: high

Primary actor: server, Driver User case type: Detail, essential

Stackholders ahd interests:

Driver-want to confirm delivery

Computer server-want to ensure the driver can confirm the current delivery;

Brief description: this use case describes how the drive confirms the current delivery;

Trigger: when the user clicks the comfirm button at the deliver management page.

Type: internal

Relationship:

Association: Driver, server

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The driver logs in the system
2. The driver clicks on the “Start deliver” button at the main page
3. The system will turn to the “start delivery” page.
4. The system will show the current deliver route and certain information
5. The driver clicks on the “ManageDelivery” button
6. The system will turn to the “ManageDelivery” page and show certain information
7. The Driver clicks on the “confirm delivery button”
8. The system will confirm the current delivery task and send the confirmation through an email to the user

User Case Name: ShowWorkSummarization ID:18 Important level: high

Primary actor: server, Driver User case type: Detail, essential

Stackholders ahd interests:

Driver-want to see the work progress

Computer server-want to ensure the driver can see the work progress;

Brief description: this use case describes how the drive see the work progress;

Trigger: when the user clicks the “Summarization” button at the deliver management page.

Type: internal

Relationship:

Association: Driver, server

Include:

Exclude:

Generalization:

Normal Flow of events:

1. The driver logs in the system
2. The driver clicks on the “Summarization” button
3. The system will turn to the Summarization page

User Case Name: StopDelivery ID:19 Important level: high

Primary actor: server, Driver User case type: Detail, essential

Stackholders ahd interests:

Driver-want to stop the job

Computer server-want to ensure the driver can store the job;

Brief description: this use case describes how the drive can stop the job;

Trigger: when the user clicks the “Stop” button at the deliver management page.

Type: internal

Relationship:

Association: Driver, server

Include:

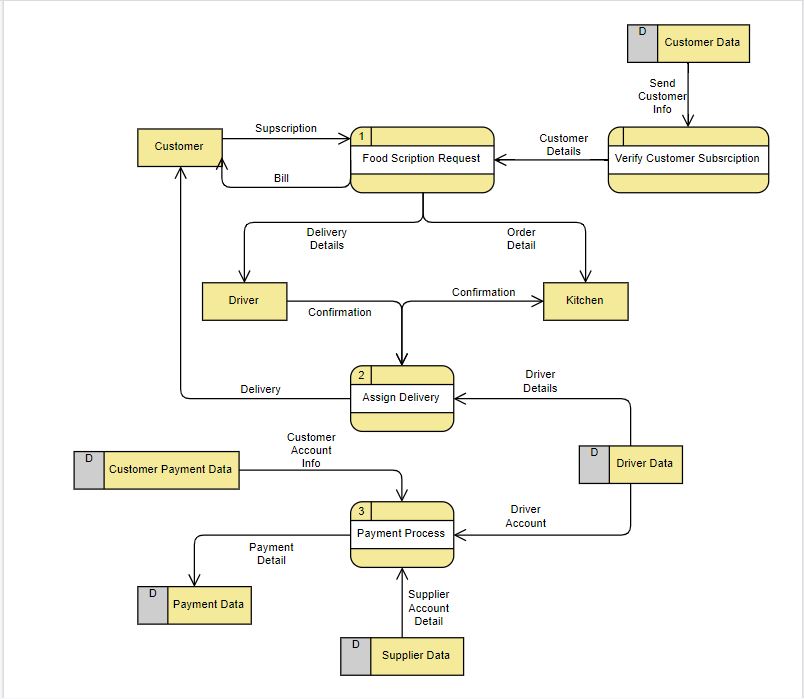
Exclude:

Generalization:

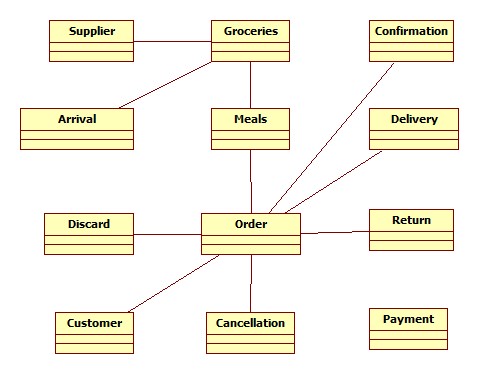
Normal Flow of events:

1. The driver logs in the system
2. The driver clicks on the “Start deliver” button at the main page
3. The system will turn to the “start delivery” page.
4. The system will show the current deliver route and certain information
5. The drive clicks on the “stop” button
6. The system will return back to the main page

The DFDs diagram version-1.



Conceptual Model verson-1

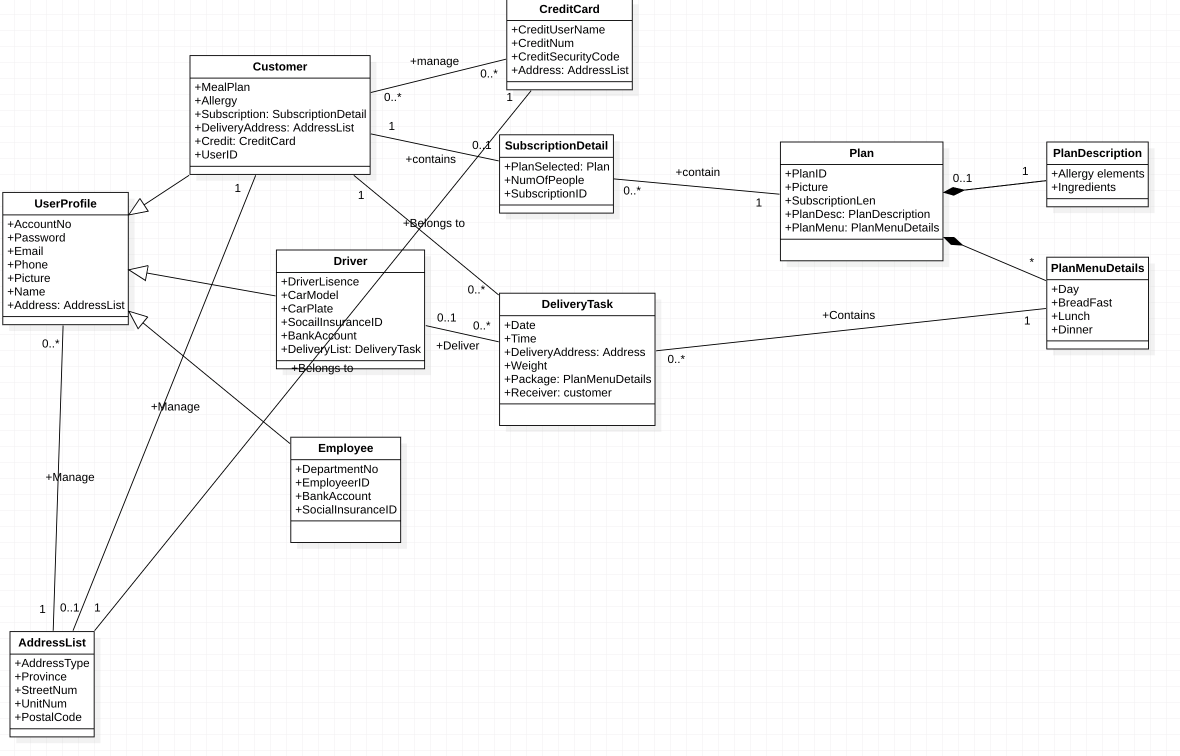


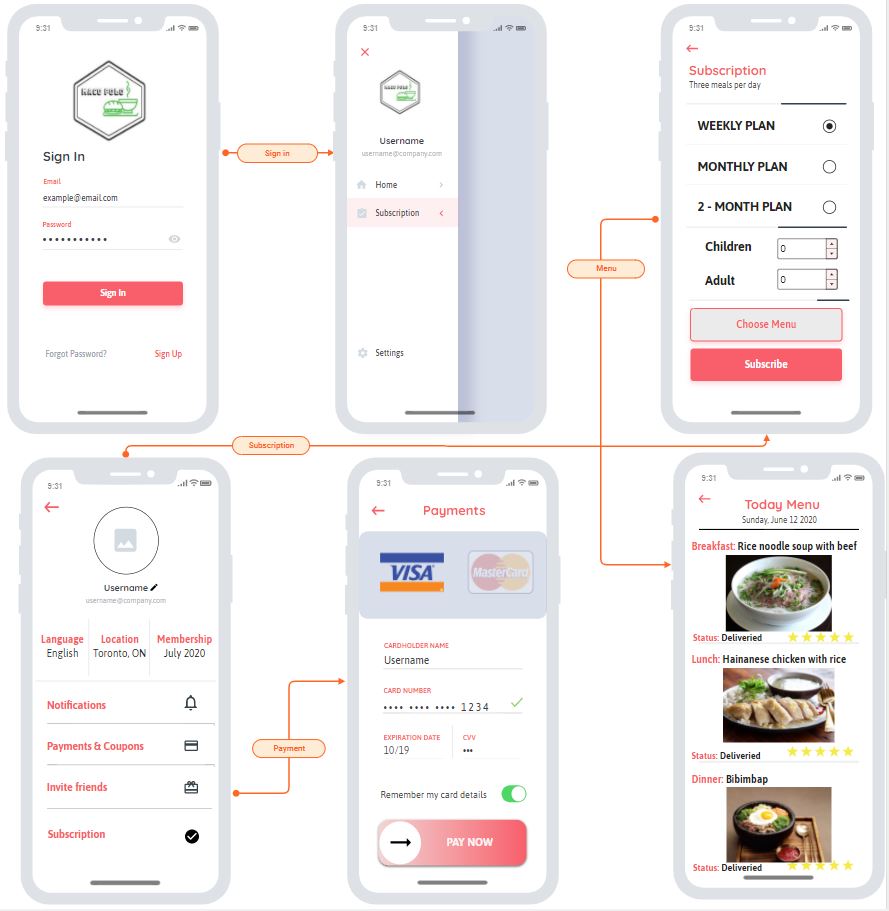
**3.3 Interface Mock-ups**

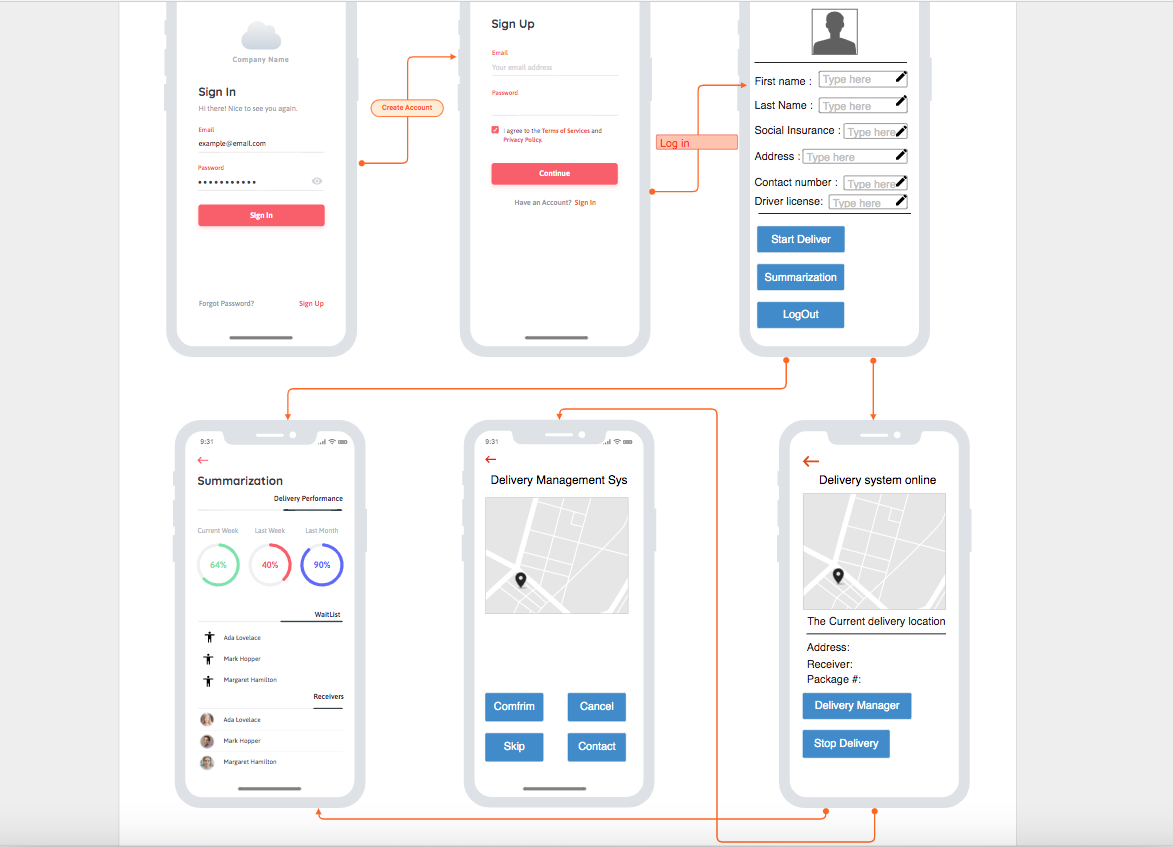
* **Website Interface Mockup:**

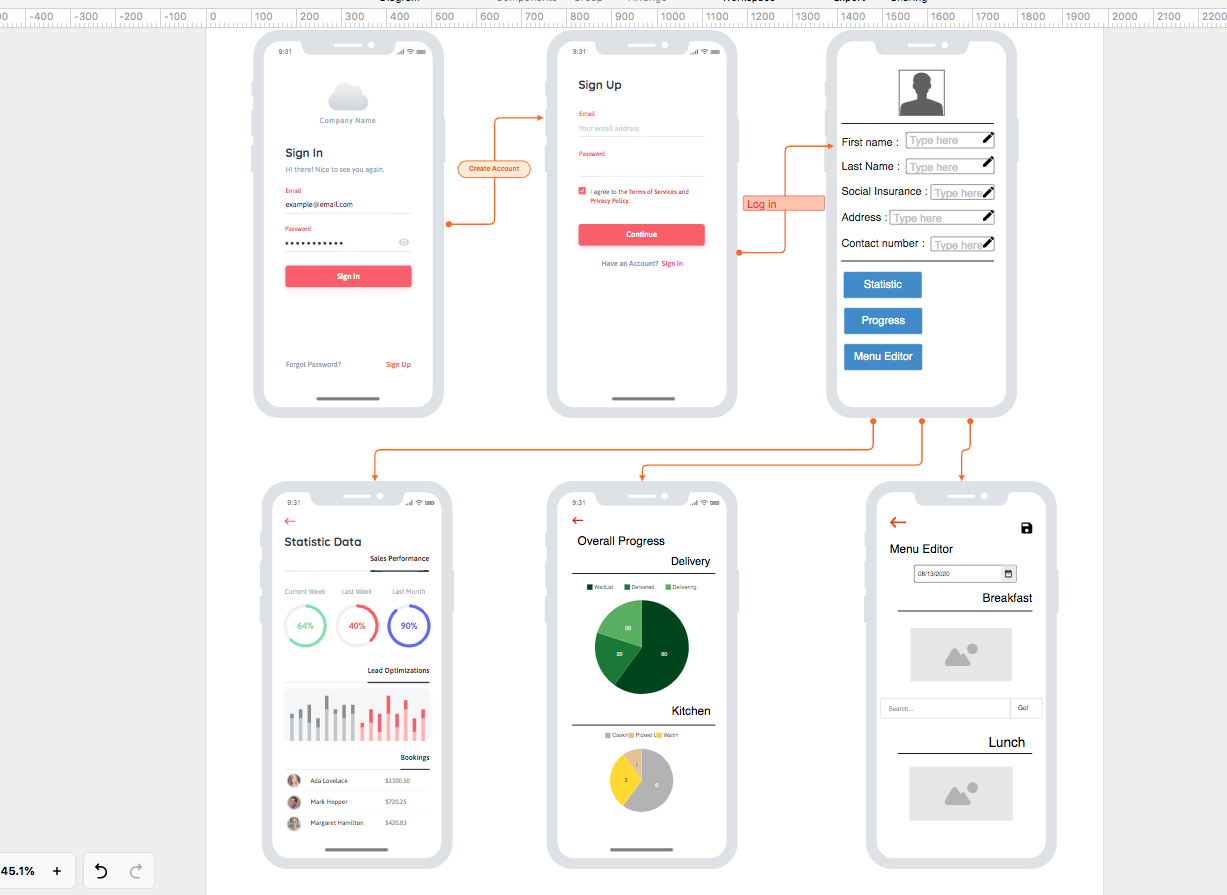
<https://xd.adobe.com/view/e05f0281-c20d-4a04-aeb1-786d7b085f00-b50c>

**3.4 Interface Mock-ups Domain Case Diagram**

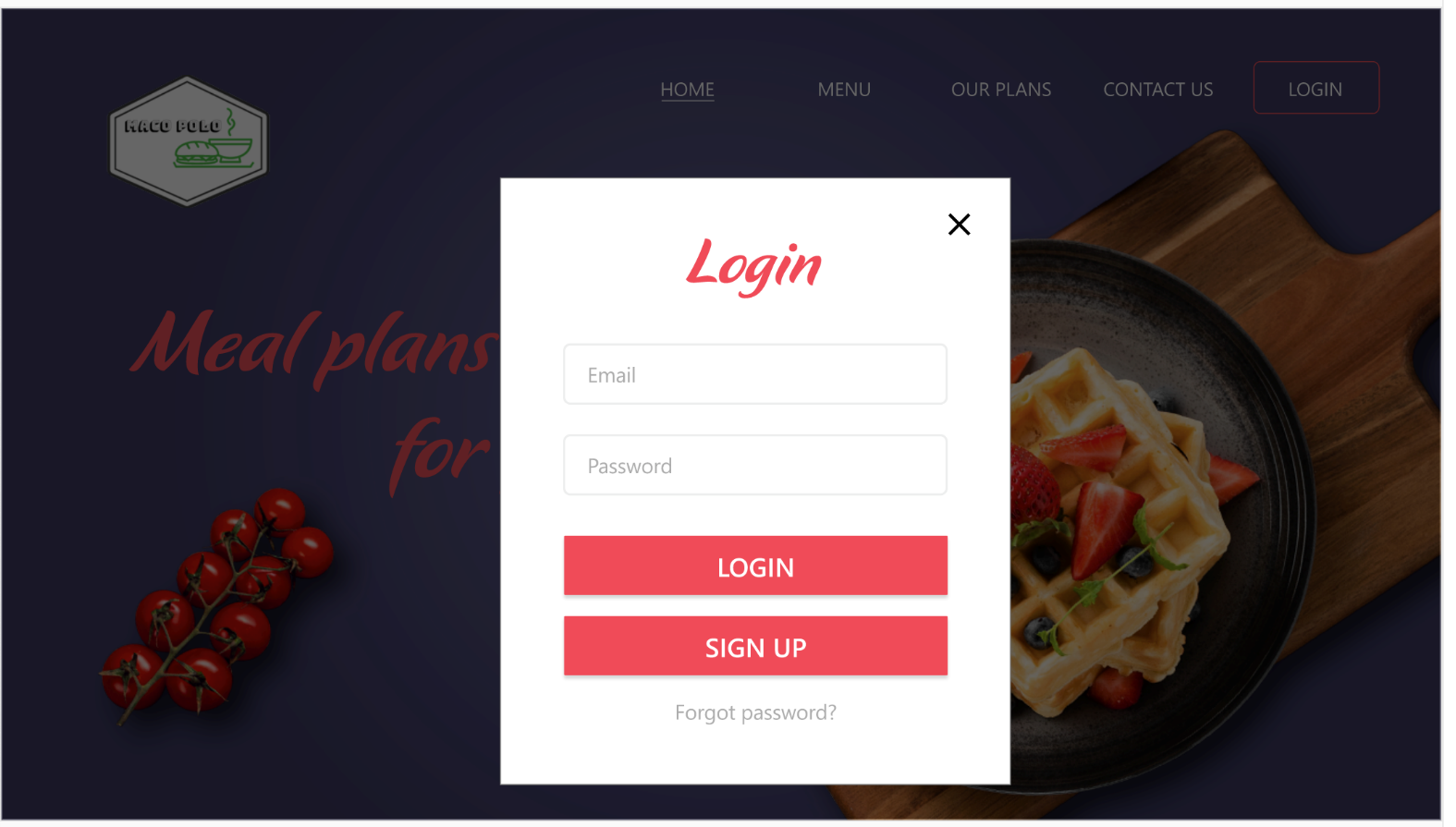








# 



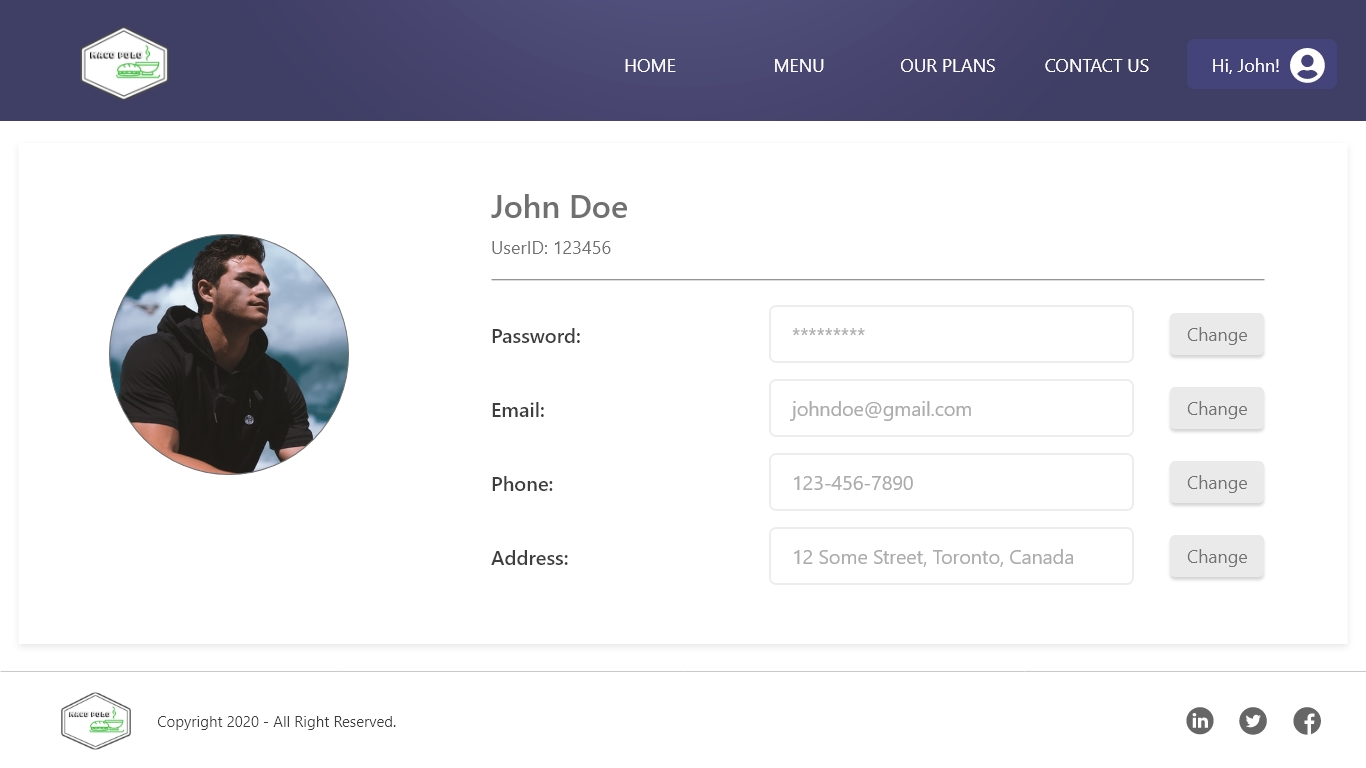
# 

# 

# 

# 

# 



# 

# 

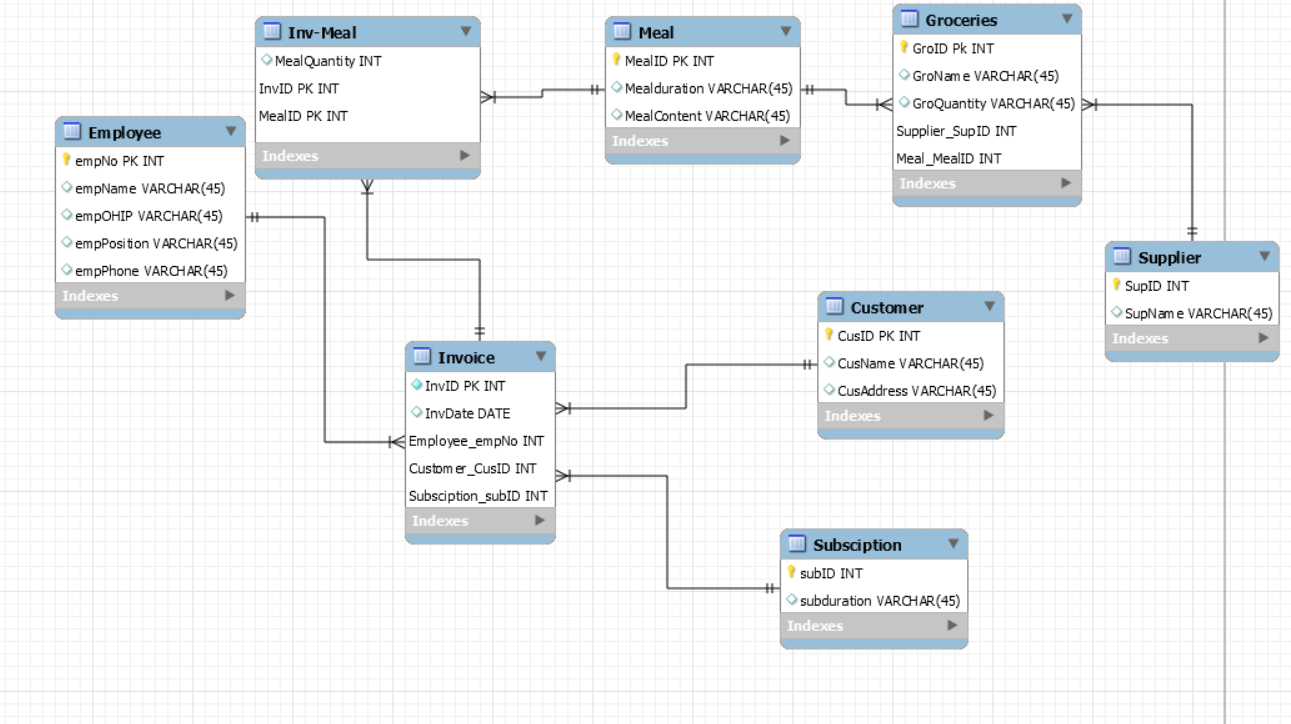
# 

# 4. Database

## 

## 4.1. Scripts to create, populate, delete tables

The first version of ERD.



### 4.1.1. Scripts to create tables

Create table Employee{

empNo number(10) primary key,

empName varchar(45),

empOHIP varchar(45),

empPosition varchar(45),

empPhone varchar(45)

}

Create table supplier{

supID number(10) primary key,

supName varchar(10)

}

Create table customer{

custID number(10) primary key,

cusName varchar(45),

cusAddress varchar(45)

}

Create table subscription{

subID number(10) primary key,

subduration varchar(45)

}

Create table Groceries{

GroID number(10) primary key,

GroName varchar(45),

GroQuantity varchar(45),

Supplier\_supID number(10)

Constraint supplier\_supID\_FK reference supplier(supID),

Meal\_MealID number(10)

Constraint Meal\_MealID\_FK reference Meal(MealID)

}

Create table Meal{

MealID number(10) primary key,

Mealduration varchar(45),

Mealcontent varchar(45)

}

Create table Inv\_Meal{

MealQuantity number(10),

InvID number(10) primary key,

MealID number(10) primary key,

Constraint primary key(InvID,MealID),

Constraint InvID\_FK foreign key (InvID) references Meal(MealID),

Constraint MealID\_Fk foreign key (MealID) references Invoice(InvID)

}

Create table Invoice{

InvID number(10) primary key,

InvDate Date,

Employee\_empNo Number(10)

Constraint Employee\_empNo\_FK referenes Employee(empNo),

Customer\_cusID number(10)

Constraint customer\_cusID\_FK references Customer(cusID),

Subscription\_subID number(10)

Constraint foreign key reference subscription(subID)

}

### 4.1.2. Scripts to populate tables (meaningful data)

use innovativedb

db.createCollection('Employees');

db.createCollection('Suppliers');

db.createCollection('Customers');

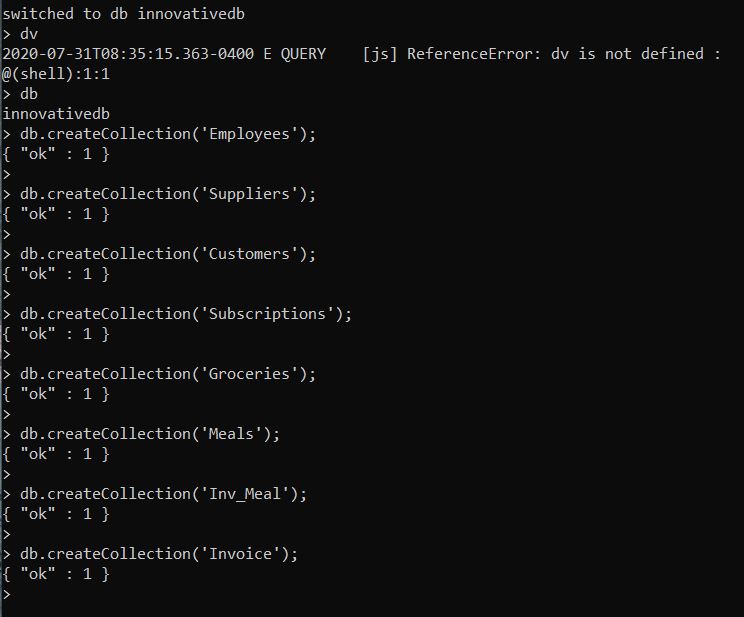
db.createCollection('Subscriptions');

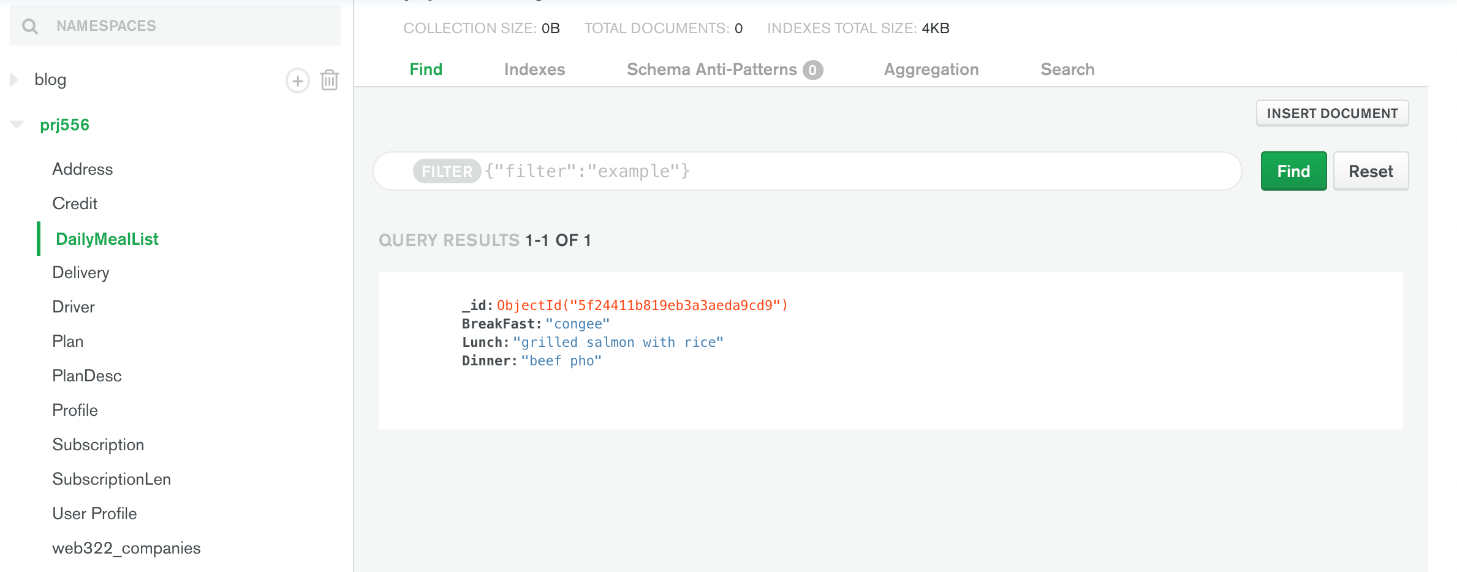
db.createCollection('Groceries');

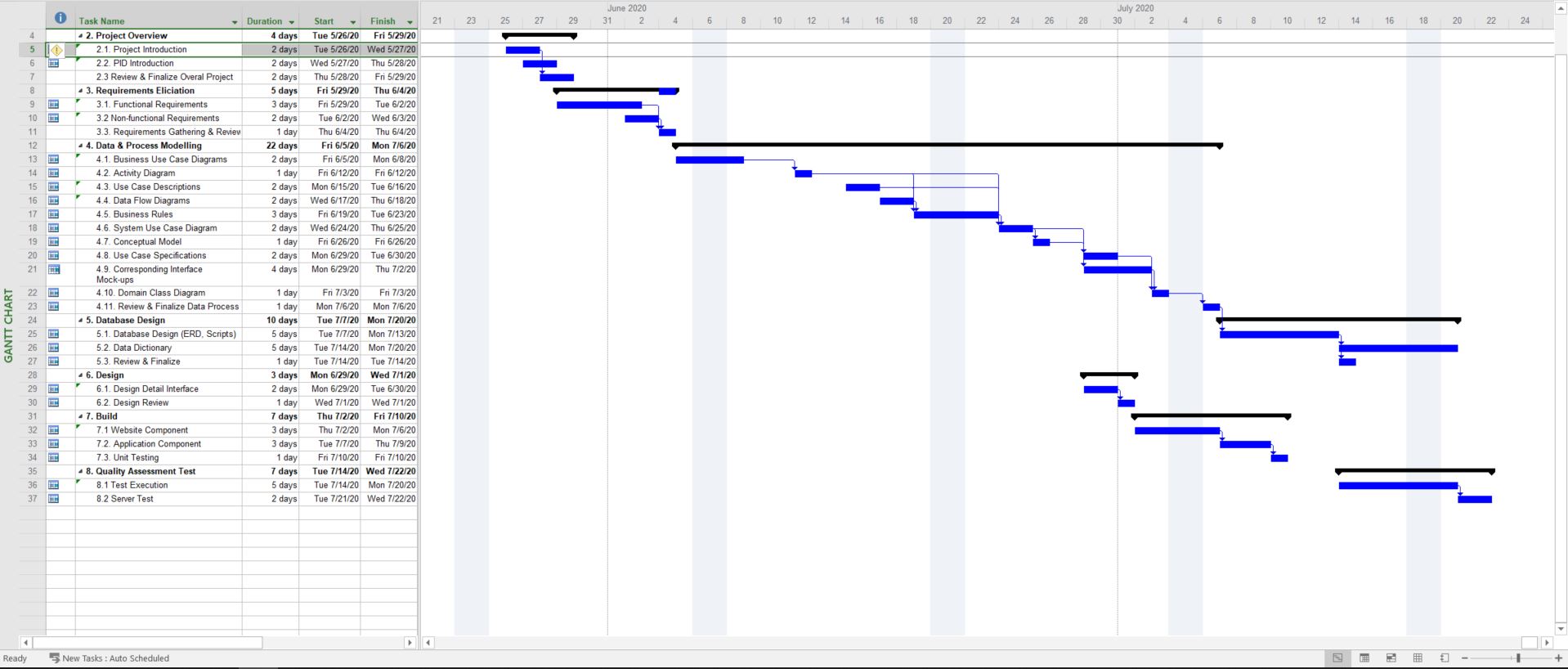
db.createCollection('Meals');

db.createCollection('Inv\_Meal');

db.createCollection('Invoice');







### 4.1.3 Scripts to delete tables

db.collection.remove('Employees ');

db.collection.remove('Suppliers'');

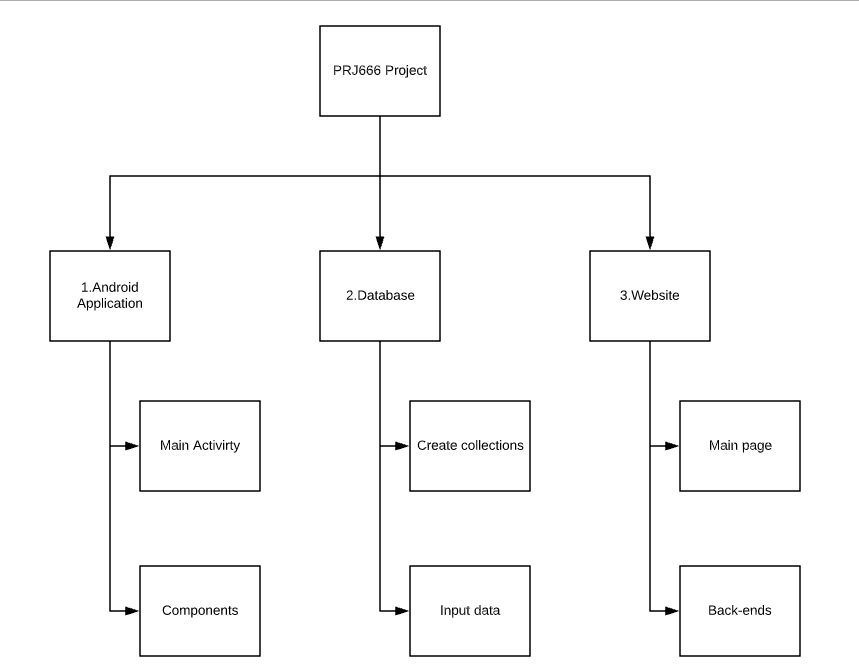
...

## 4.2. Data Dictionary

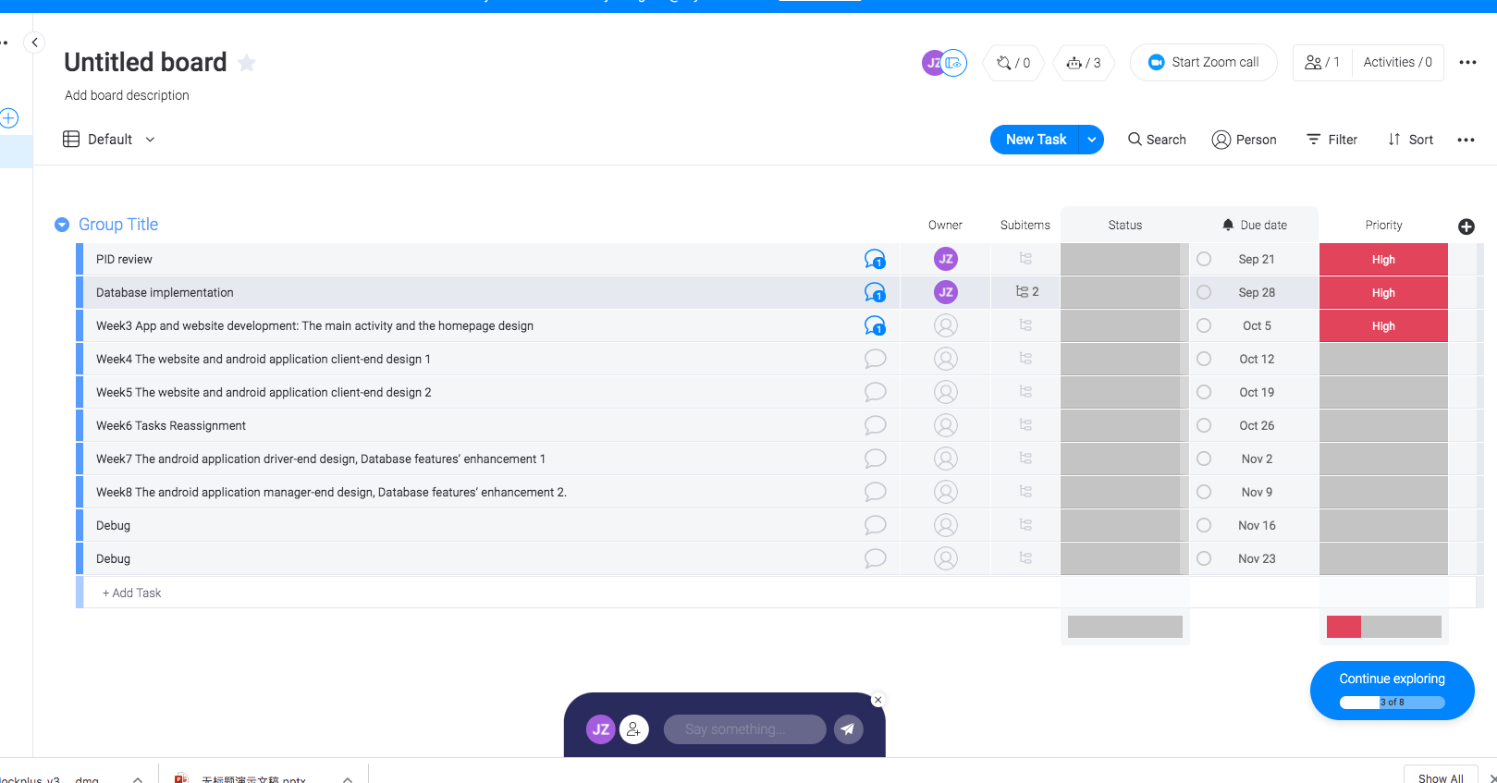
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Table** | **Data Type** | **Length** | **Read Only?** | **NULL** | **Required** | **Description** |
| ID | User | INT | 11 | No | No | Yes | Displays the ID of the user. |
| NAME | User | VARCHAR | 25 | No | No | Yes | Displays the username of the user |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

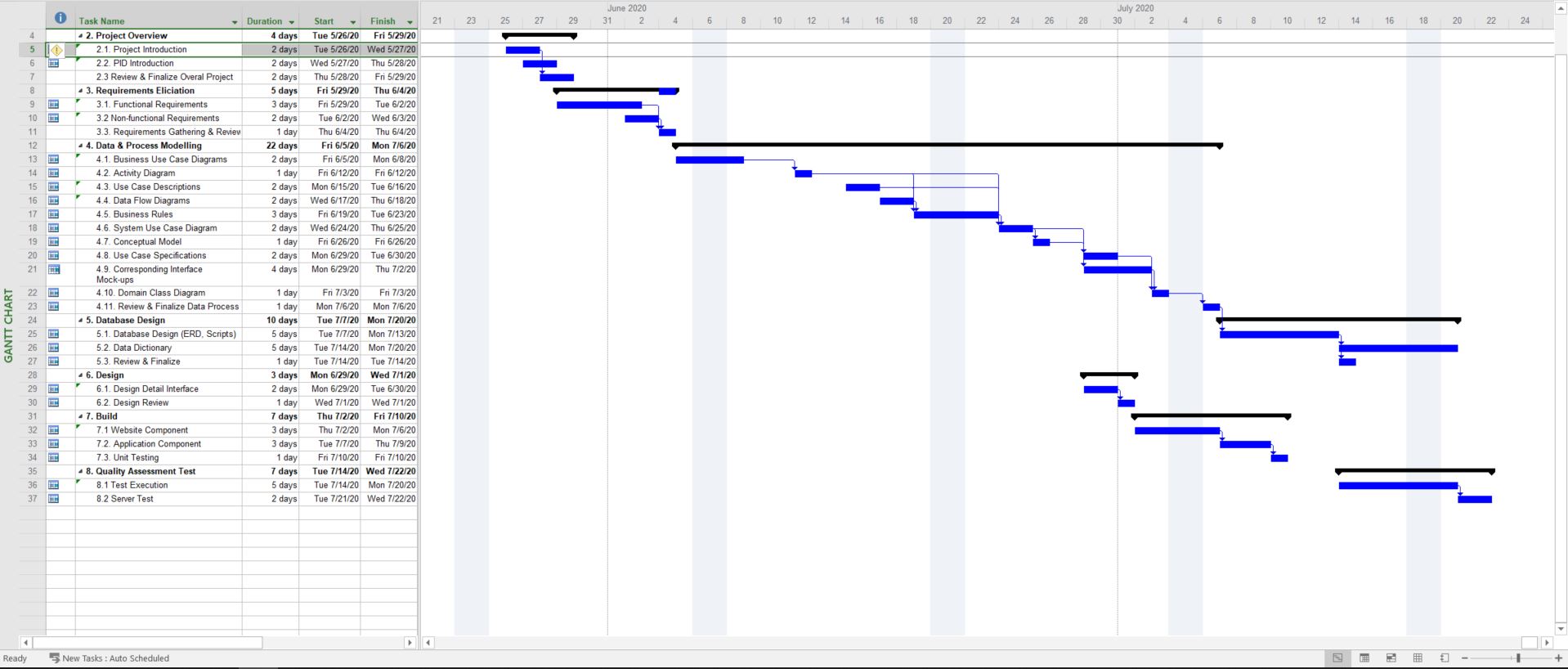
# Implementation Schedule for PRJ666

## 5.1 Work Breakdown Structure



## 5.2 Implementation Schedule using MS Project





# Measurable deliverables

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Schedule | Week 1 | Week2 | Week3 | Week4 | Week5 |
| Details | Understand the PID, and tasks assignment | Database implementation | App and website development: The Main activity and the home page design | App and website development: The website and android application client-end design 1 | App and website development: The website and android application client-end design 2 |
| Team weekly deliverables | Online meeting and Monday program management software | same | same | same | same |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Schedule | Week 6 | Week 7 | Week 8 | Week 9 | Week 10 |
| Details | Task re-assignment | App and website development:  The android application driver-end design, Database features’ enhancement 1. | App and website development:  The android application manager-end design, Database features’ enhancement 2. | Debug | Debug |
| Team weekly deliverables | same | same | same | same | same |

# Acceptance Criteria

Acceptance criteria:

User story 1: As a user, I want to be able to register an account for the app or the website.

Acceptance criteria 1:

Given: the user provided valid information

When: the customer registers an account

Then: ensure the customer has no allergy to our meal’s ingredients

And: ensure the customer agrees advertisement email or not

And: ensure the contract sends to the customer

Acceptance criteria 2:

Given: the user provided invalid information

When: the customer registers an account

Then: ensure the system notices the customer where is the error

User story 2: As a user, I want to login my account

Acceptance criteria 1:

Given: the user provided valid information

When: the customer logins our service

Then: ensure the system will turn to the user profile page

Acceptance criteria 2:

Given: the user provided invalid information

When: the customer logins our service

Then: ensure the system will notice the error, and refresh the page

User story 3: As a user, I want to subscript a service

Acceptance criteria 1:

Given: the user has login, the network condition is proper.

When: the customer chooses the meal plan, and subscripts it

And: the account of the customer has the valid debit information

And: the address is valid

Then: ensure the system will calculate the total amount of the meal plan

And: ensure the system will print out the receipt

And: ensure the system will save the data to the database

Acceptance criteria 2:

Given: the user has login, the network condition is proper

When: the customer chooses the meal plan, and subscripts it

And: the account of the customer hasn’t the valid debit information

And: the address is invalid

Then: ensure the system notice the customer to input the account profile

And: ensure the system notice the customer to try again

Employee story: As a postman, I want to get delivery routes.

Given: the employee successfully logged in the system, the network connection is proper.

When: the postman activates the dedicated delivery app

Then: ensure the system will show the route

And: ensure the route will change, once the car is not on the original route.

# Client / Faculty Sign-off