**CS673 Software Engineering** 

**Team 4 - Moo Dairy Online Store**

**Project Proposal and Planning**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Duan Lin | Team leader / QA leader | *Duan Lin* | 09/24/2020 |
| HussaiN Alibrahim | Configuration leader / Front-End Designer 1 | *Hussain alibrahim* | 09/24/2020 |
| Shuyi Zheng | Requirement leader / Front-End Designer 2 | *Shuyi Zheng* | 09/24/2020 |
| Shreyas Prakash | Design and Implementation Leader | *Shreyas Prakash* | 09/24/2020 |
| Neha Jadhav Sarnaik | Design and Implementation Leader | *Neha Jadhav Sarnaik* | 09/24/2020 |
| Ming Yuan | Security Leader | *Ming Yuan* | 09/24/2020 |
| Bhavesh Tadikonda | Backup Team leader | *Bhavesh* | 09/24/2020 |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| 1.0 | Shuyi Zheng | 09/24/2020 | Functional requirement |
| 1.1 | **Duan Lin** | 09/24/2020 | Overview, Related Work, Quality Assurance Plan |
| 1.2 | Bhavesh Tadikonda | 09/24/2020 | High level design,  Databases, Endpoints for application |
| 1.3 | Shreyas Prakash | 09/24/2020 | Non Functional Requirements. Objectives and Priorities. |
| 1.4 | Ming Yuan | 09/24/2020 | Risk Management and Mitigation Plan |
| 1.5 | Neha Jadhav Sarnaik | 09/24/2020 | Management Plan :Process Model |
| 1.6 | Duan Lin | 10/22/2020 | Update QA. |

[Overview](#_87t9hln2vjz0)

[Related Work](#_mps353x5ezyl)

[Detailed Description](#_fg3z0hpd4q9v)

[Management Plan](#_ds8oyr75pnh1)

[Process Model](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.27177f40uci)

[Risk Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.a4oqwntk3mw)

[Monitoring and Controlling Mechanism](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.ywdoc2clc9yt)

[Schedule and deadline](#_tadq5mb0pici)

[Quality Assurance Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.72e1f4uawy2r)

[Metrics](#_b2haznn3yyz2)

[Standard](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.vc72k6dweldv)

[Inspection/Review Process](#_f1c69ifi68h7)

[Testing](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.r5d5mhtlf0kq)

[Defect Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.54a4wuncjg1c)

[Process improvement process](#_jhct37ebxxpn)

[Configuration Management Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.hw41vg4ykxen)

[Configuration items and tools](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.bwlb4d4vdox2)

[code commit guidelines](#_yyauft6zr9hw)

[References](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.8mva2050iy7t)

[Glossary](#_ty3i2nqffhtc)

# Overview

Our project aims to build Dairy Products Online E-commerce Store, provide a web application that will help the consumer get their dairy products on time. The website will contain features such as cart, checkout, user profile, and a products information page.

For our ecommerce website, we will run a B2C(Business to Customer) model. In addition, we only sell dairy products such as milk, cheese, cream, butter and yogurt. Ever since we first met, our team has had a strong aspiration to support farms. We saw the icon that is the cow as inspiration and wanted to share our love of Dairy Products all across the world. So we formed Moo Dairy, fresh dairy for fresh folks. We advocate healthy diet style and provide dairy products facts and nutrition facts. Furthermore, we guarantee simple, practical and reliable operations in our App that it is also suitable for all kinds of customers.

# Related Work

When you think of ecommerce in the United States, brands such as Amazon, Walmart and eBay likely come to your mind. Considering the widespread success of many American e-commerce brands, it’s unsurprising to learn that the US is a notable leader in the global e-commerce sector.

The giant population mentioned earlier amounts to 327.9 million — 312 million of whom use the internet, and 209.6 million of whom shop online, according to figures published from Statista in 2016.

In 2018 sales from e-commerce reached a whopping $517.36 billion, and retail e-commerce alone is predicted to surpass $740 billion by 2023.

As of 2018, online retail made up 14.3 percent of the total retail sales in the US, up from 11.6 percent in 2016. This makes the US the third most advanced e-commerce market in the world, bested by the UK and Germany, which each enjoying a higher percentage of total retail sales.

# Proposed High level Requirements

* 1. Functional Requirements  
     1. Essential Features
        1. User Profiles
           1. Register  
              “As a user, I want to sign up for an account so I can purchase products and check my order.”
           2. Login

“As a user, I want to sign in so I can purchase products and check my order.”

* + - * 1. Setting

“As a user, I want to modify my information so I can update my information.”

* + - * 1. Order history

“As a user, I want to check my order history so I can track my order.”

* + - 1. Product
         1. Product list

**“**As a user, I want to see products on the same page so I can compare the price and quality.”

* + - * 1. Productdetail

**“**As a user, I want to see productdetail so I can know more about this product.”

* + - 1. Order
         1. Cart

“As a user, I want to have a cart to save the product I am interested in so I can buy it later.”

* + - * 1. Checkout  
           “As a user, I want to check out so I can buy and pay for the products in my cart.”
      1. Others
         1. About us

“As a user, I want to know more about your company so I can decide whether this website is trustful.”

* + - * 1. Footer  
           “As a user, I want to know more about how I could contact you so I may feel this website is reliable .”
    1. Desirable Features (the nice features that you really want to have too):
       1. Three parties login

“ As a user, I want to login with my google account so I don’t need to sign up for a new account.”

* + - 1. Share

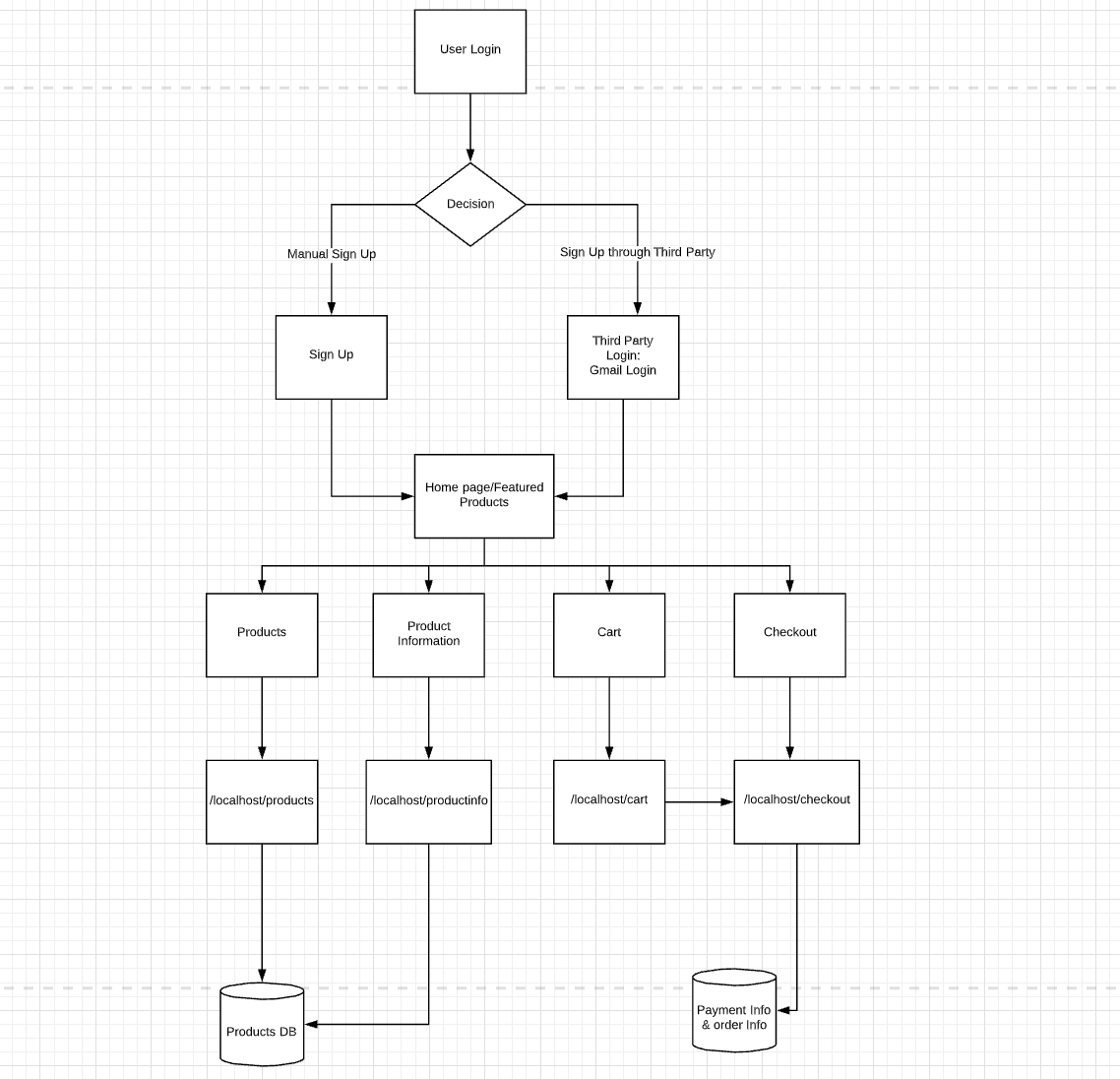
“ As a user, I want to share the product through ins or facebook so I can share with my friend.”

* + 1. Optional Features (additional cool features that you want to have if there is time):
       1. Save

“ As a user, I want to save the product but not add it to cart so even if I don’t want this product I still can keep it somewhere for comparison or sharing with my friend next time.”

* + 1. Existing Features (delete this item if your project starts from scratch):
  1. Nonfunctional Requirements:
* Scalability
  + Application should be made such that it handles large traffic of requests. Here we are testing our application on 20 requests at single point of time.
* Reliability
  + Application should be reliable which means it should provide accurate results all the time.
* Performance
  + Application should give accurate results within less amount of time.  
    Optimized API design for achieving the best performance even under large traffic.
* Availability
  + Application should be available to the users all the time for use.
* Security
  + Application should be secure and resilient in the face of attacks. The Application should always give correct and predictable results.
  + Attacks includes the following:

1. Bad Bots- Bad Bots can access the Application Database and access user login details. So this issue has to be considered.
2. Credit Card Fraud Attacks- As the user might have saved his/her credit card on the payment portal and hackers might hack these details. So this issue has to be taken care.
3. DDos Attack- This is an overload requests on the application by other IP servers which may slow down the working application and potentially hang up. So this also has to be taken care.

C. High level design   
 

Architecture:

Planned to use microservice architecture, because application functionality can be divided into multiple independent modules that can precisely perform standalone tasks.

Routes to the application:

(RESTful API endpoints)  
  
Localhost: Planned to use AWS S3 buckets for hosting the application. With static hosting enabled

localhost/products -> Method, GET list all the products available in the application.

localhost/products/<int:id> -> Method GET,POST only the details of specific product

localhost/cart -> Method GET, all the products selected by the user will be displayed in the page.

localhost/checkout -> Method GET,POST will have a Place holder’s for entering card information and quantities of the product can be changed.   
  
Databases:   
  
MongoDB, which is a NoSql database. Which is a document database, information in documents stored as Json objects. Objects are nested as Key-value pairs.  
  
Planned to have 3 different collections.   
  
User\_information:   
 {  
 id: UUID,

Email: String,

Phone: Integer,

DOB: Integer,

ProfileName: String

}

Product\_Information:

{

id:UUID:{

Name\_of\_product: String,

Description: String,

Price: Integer

}

}

Product\_Order\_Information:

{

id:UUID,

email:String,

price:Integer,

Products:{

Product\_name:qty

}

}

Product\_order\_information to keep track of all orders of the customers with their price and products.

# Management Plan

## Process Model

## The process model being used is the Agile model where there is weekly communication between the developers,owner and the team .

* + 1. Working application is the initial step to improvement.

## Reviews of progress of the application have been made more often.

* + 1. Two way communication methods have been utilized in all the stages of development of the application.
    2. Functionalities of the application are developed rapidly and illustrated.
    3. Utilization of this model has also given flexibility to the development team where changes can be made at the middle of the project .
    4. As the development process is iterative, less upfront planning is required to develop the application .
    5. Quick changes can be made as per the requirement from the customer regarding the application.

Phases of Agile development for the application

1. Requirement gathering Stage : In this stage the requirements were defined.Business opportunities were described and planning of the time done accordingly.
2. Design the requirements Stage: Once the application was identified , further work with the team and the customer was carried out to define the requirements.The working of the application with the main components was shown through the high level design.
3. IterationStage : The team defined the requirements and then the work started.Development team started working on the application with a goal to deploy the working application. The application will go through several stages of improvement keeping the application simple and with minimum functionalities.
4. Testing the application: In this stage Quality Assurance team will review each iteration's performance and check for any errors.
5. Deployment Stage:In this stage the application is given by the team for the customers workplace (web).
6. Feedback Stage: After the release of each iteration ,the final stage will be the feedback of that iteration and its working.

## Objectives and Priorities:

Objectives and priorities of the project are broken down in stages. We are basically having 4 stages which describes the objectives in that particular stage. These stages are numbered in ascending order from Stage-1 to Stage-4 with Stage-1 taken as the highest priority stage and Stage-4 taken as lowest priority stage at this point of time.

The four stages are as below:

Stage-1:

Objectives of this stage are as follows:

i) Allow the user to create an account on our Application.

ii) Password and Email verification of the credentials given by the user.

Stage 2:

Objectives of this stage are as follows:

i) Allow users to do either Manual sign In or Third Party Sign in into our Application.

ii) Gmail ID verification by OAuth and storing user login credentials into the Database

With encryption mechanism(IP hashing)

iii) Creating dynamic front end pages which interacts with the initial routes of API

Designed for our application.

iv) Integrating Google API with API’s built along with front end HTML pages and

MongoDB for successful login into the system through Google Sign In.

v) Testing the Manual Sign In and Google Sign In features by QA team. Finding the

Bugs which have to be resolved in the next stage.

Stage 3:

Objectives of this stage are as follows:

i) Taking initiative to fix the bugs found by the QA team.

ii) Allow users to see the product information. This is a fixed process and the

Users can view only one product information by clicking on the product.

iii) Allow users to add multiple products into the cart.

iv) Allow users to remove multiple products from the cart.

v) Dynamic content creation on the front end using the response from the API.

vi) Verifying if the user is able to see the product information successfully and also

Add products into the cart by the QA team.

Stage 4:

Objectives of this stage are as follows:

i) Allow users to update the cart and add payment details.

ii) Allow users to make the payment successfully.

iii) Allow users to cancel the payment upon entering the payment gateway.

iv) Creating end to end proper workflow taking security features into account which

Reduces external attacks such as Bad Bots or DDos.

Tasks required to achieve the objectives with highest priority taken first into consideration is as follows:

1. Analyze Functional and Nonfunctional Requirements.
2. Low level Design of the application.
3. Cost, Frequency and Bandwidth Estimation.
4. Designing of Database Schema.
5. Class Diagram analysis before coming up with the API design.
6. API design according to the requirements and class diagram analysis.
7. Database Sharding Mechanism for partitioning into 2 Databases depending on the functionality and load traffic.
8. High level Design of the system including Sharding mechanisms for better performance of the application.
9. Auto Integration of API responses with databases and also with front end to create dynamic content using Jinja2.
10. Adding more routes and functionalities to the application.
11. Testing the application with Unit Test Cases.
12. Performing Integration test on the application.
13. Fixing the bugs(if found) which are found by the QA team.
14. Optimizing the performance of the Application for better time and space.
15. Performing Unit and Integration Test by QA team.
16. Adding encryption and decryption techniques(security features) for the application.
17. Ready to release.

## Risk Management

### Project Risks

* **Lack of financial knowledge and experience**: since our project is an E-commerce web application to sell dairy products. Some team members may not be familiar with the knowledge of online financial issues and web application programming which are essential for us to complete our web application and meet the requirements of potential customers.
* **Mitigation plan**: each team member according to the actual situation of himself/herself has to learn relevant knowledge, tools, applications and software through various possible ways like youtube, blogs etc. The team members with rich experience can help the less experienced teammates quickly acquire the knowledge and skills that are conducive to the project promotion. Anyone who has questions or problems should seek help from others in time.
* **Legal issues:** as an online shopping application, we are always inseparable from the part of payment, whether it’s direct payment or in direct payment through third party payment. If it’s direct, the payment information of users may be leaked, which may bring huge loss to users. We worry that we do not have powerful enough security measures to ensure the important information like the credit card and its password.If the payment is made through a third party, we have no license from theresince this is a student project and not an official product. There are some legal issues which also arouses the possibility of accidental payment sometime.
* **Mitigation plan:** First we will not have a real card. The front end will fake a payment screen in which the only condition is that the credit card fits the criteria needed in the appropriate fields. And then we will refrain from using third party payment tools such as paypal or that sort of thing.
* **Data Risks:** When users log in our website, they will input their own username and corresponding password. When making virtual payment, they will also enter their own credit card number and password. If these data are directly stored in our database, there may be certain risks. And if we can directly see the password plaintext through the database, it’s not ethical and users will greatly reduce their trust in our product.
* **Mitigation plan:** In the back-end, we will encrypt any passwords entered by customers whether they are logging or paying, the mechanism of which is invisible to our developers. Then we stored these encrypted information into our database. Whenever users re-enter their passwords, we only need to verify whether the encrypted string are consistent to ensure the security of customers’ information.

### Technical Risks

* **Integration risk:** Our highest technical with this project is integration risk, because in the process of our implementation, our team is divided into two subteams, one is responsible for the front-end web design and the other is responsible for the implementation of the back-end database and realization of web application functionalities in the level of code. With the front-end subteam completing design of the whole web page and its functions, our back-end subteam have realized these functionalities and established a needed database simultaneously. However, for most team members, this is the first time that we are going to design, implement and test a web application in a team project, and each team member has his/her own preferred of familiar programming languages and tools. In this way, there are likely to be consistency problems not only between the front-end and back-end teams but also within two subteams. Besides, some less advanced team members may have difficulty understanding others’ code, which can bring unpredictable risks to every stage of the integration.
* **Mitigation plan:** First of all, we will make detailed planning, what functions the front-end has and what the back-end needs to achieve. In each stage of the project, we should conduct timely integration and testing. The front-end and back-end should communicate closely, understanding each other’s work process and making good connections. After the completion of each stage of work, more experienced team members will do the code review to ensure correctness and consistency of the whole integration.
* **Continuing requirements changes:** User satisfaction is crucial to software development, especially when our project is an E-commerce web application. Therefore, our other main technical risk we may face is that our application requirements will change constantly according to the needs of customers. In this process, we need to continually modify, refactor or add our code, which is likely to delay our development plan.
* **Mitigation plan:** Each team member will make clear the framework of our project with awareness of the primaries and secondaries. We will build the framework of the whole software first, in the process of which we will try our best to ensure the flexibility of our code to face possible changes and refactorings. Only when we have completed the most important or main functions such as login, verification and payment transactions, we then extend it to satisfy the users’ personalized needs. For example, only after our completion of the payment part can we develop the functionality to detect which input card is visa or mastercard.

## 

## Monitoring and Controlling Tools and Mechanisms

We will use the following tools to facilitate group communication and monitor the project progress.

* + 1. Pivotaltracker Link: <https://www.pivotaltracker.com/n/projects/2465375>
    2. Slack Link: None
    3. Github Link: <https://github.com/BUMETCS673/CS673F20T4>
    4. Zoom meeting Link: <https://bostonu.zoom.us/j/9340853828>
    5. Weekly meeting time: Friday@9:30PM

## Timeline (need to be updated at the end of each iteration)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Functional Requirements(E/D/O) | Tasks | Estimated/real person hours | Presentation Recording Link (5-10 minutes) |
| 1 | 9.3 - 9.24 | Set up requirements, configuration environment | Refer to weekly report |  |
| 2 | 9.25 - 10.22 | Manual Sign up and login, Google login, {index login register product} HTML design | Refer to weekly report | https://drive.google.com/file/d/1zSKcAT3rpo5fmZ0MnUHUVU8Ms511O9rq/view?usp=sharing |
| 3 |  |  |  |  |

# 

# Quality Assurance Plan

## Metrics

* + 1. Product metrics:

Product metrics can vary widely between different products and different industries. However, there is a common set of metrics that is helpful for almost any product. Meanwhile, selecting the right metrics to serve as key performance indicators is an essential part of the product development process.

Product metrics are often categorized into different groups. This can be quite a useful way to evaluate different aspects of how the product is performing. Here, we categorize our product metrics into two divisions: economic metrics and engagement metrics.

* Economic Metrics:
* Conversion rate

Whenever customers go through a set of steps, called a funnel, towards a certain goal. The conversion rate measures the percentage of users who make it from one step to the next. For example, this could be from a dairy product page to a checkout page or from other new customers to repeat customers.

* New customer growth rate

New customer growth measures the number of brand new customers a company acquires. This is a useful measure because it differentiates between new and repeat business and is a leading indicator of future growth. New customer growth rate is measured by taking the difference in customers from one period to the previous period, then dividing by the number of customers in the previous period.

* Customer lifetime value

Customer lifetime value (CLTV) measures how much value the average customer is worth to the company.

* Customer satisfaction

Definitely we need to acquire the feedback from customers and survey results from various sources in the real world. And we are going to collect them as an overall rating and actively adjust our product metrics.

* Engagement Metrics:
* Net promoter score

It’s a measure of how likely customers are to recommend your product or service to others.

* Active user

It’s a measure of the number of users who actually use the product during a certain period, such as daily, weekly or monthly.

* Feature usage

This metric measures the percentage of users who use different product features. This can help product leaders prioritize the features that are most important to users and potentially remove or deprioritize features that are not used more frequently.

* 1. Standard  
     Coding standard:
* Avoid code duplication. Using as many modules or methods to reduce the duplication and also it is convenient to make your code more readable and manageable.
* Adhere to conventions of PEP 8, including the indentation, string quotes, whitespace in expressions and statements, readable comments, naming conventions and others.
* Be aware that refactoring variables or functions would affect multiple references in all places.

Documentation:

* Meeting minutes: Everyone should engage and take note at the end of weekly meetings and should be filled out by corresponding members.
* Weekly progress reports: Everyone should update their weekly progress by the weekend.
* SPPP: The details and plans for the whole project. Written by all team members and updated as needed.

## Inspection/Review Process

Code inspections:

* Peer review: Since we have the Front-End team and Back-End team, each team member should review your code when you commit and push it to GitHub.
* Team review: After accomplishment of each iteration, our team should hold a review meeting and discuss individual works, experiences, setbacks as well as valuable thoughts during each process. Having code review will be beneficial so that we could learn more, and reflect ourselves impersonally.
* Content: Basic frameworks, coding logics and main function designs.

## Testing:

* Individual test: Before you submit your code to the QA leader, you should implement several test cases to test your own modules. Please make sure the code is compiled and tested successfully.
* QA test: Once someone completed their parts and submitted, QA should begin to make measurements and tests on these new submitted modules or functions. In terms of testing methods, QA needs to use black box testing and whitebox testing. Once the defects or bugs are found, QA should send the codes back to related Front-End or Back-End developers. Such a progress will be repeated until we get into the bug-free stage.

# Configuration Management Plan

(For more details, please refer to SCMP document for encounter example)

## Configuration items and tools

* Github shall be used

## Change management and branch management

* Branches shall belong to individual members. There will be two types of branches: Frontend and Backend. Then, after a screening process, a member can commit his or her work into their branch. Once all of the branches in a section are ready, we merge it onto the leader of the type of end they are working on.

## Code commit guidelines

* Code will be reviewed by 2 members each week line by line and approved before committing.

## **Integration and deployment plan**

# References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard. )

[1] us dairy <https://www.usdairy.com/>

[2] instacart <https://www.instacart.com/>

# Glossary