[Software Engineering]

Final Report

Information Technology and Management

Seoul National University of Science and Technology

17101992 Sumin Hong, workingsumin@gmail.com

17102044 Jeewon Kim, jeewoncoding@gmail.com

20102123 Jina Choi, cjina1102@seoultech.ac.kr

Contents

XIII. Reference

I. Abstraction

The application 'UglyBut' is a live-commerce software that helps connection between the farmers who worried about handling ugly agricultural product, especially fruits, and customers who want to purchase the fruits at a low price, without considering the shape of fruits. First, we described the context with the actors that they use our software. Following this, functional requirements and non-functional requirements are established. As we assume that our application drives sales of fruits through live, so these include the functions related live. The scenario indicates how the defined functions are used to the actors, such as farmers, customers. Also, we suggest UML Diagrams for more clear and easier understanding of our application. Use Case Diagram shows the connections between the functionalities and actors who use the functions. Class Diagrams helps understands the structure of our application design. Sequence Diagrams describes the relations and times that suggested functions used.

In Design phase, the details for our application design are presented. First, the ER Diagram shows the database design and the * and * relationships of each database. It follows the Class Diagram below of our report and. Also, we depicted User Interface design. Finally, Interface Design and Detailed Design both explained the parameters and technical logics of the functionalities of our software in detail to implement the software.

II. Introduction

II-a. Necessity of Term Project

Ugly agricultural products, which are discarded by 230 tons (about 4 trillion won in production) every year, are classified out of grade by simple external factors such as shape and size and are sold at a bargain price or discarded. There are huge social costs and environmental pollution to deal with this. However, the reality is that there is no suitable platform to purchase these agricultural products.

II-b. Goal of Term Project

Our goal is to make a live commerce application that connects farmers who produce ugly produce with consumers who want to buy it. If accurate and reliable product information is delivered to consumers by performing live directly from farmers, consumers' anxiety can be lowered, and accessibility can be improved. In addition, the simplification of distribution channels through live commerce enables consumers to purchase delicious fruits at an affordable price. Farmers can also sell ugly produce in large quantities at a time that cost a lot to process, secure profits through relatively more sales, and increase the overall interest in ugly produce.

II-c. Problem Description

There are eight actors, Diana, Jack, administration, and five kinds of APIs(Road Name Address Open API, Phone Number verification API, Live Streaming API, Delivery API, Payment API) who are involved in our application. Diana and Jack are the main actors.

- Diana : She is 25 years old and currently living in Seoul. She recently left her parents' house in countryside and started to live by herself for the first time for her education. She wants to have fruits, but she is hesitant to buy them because she is not a financially available student. Moreover, she wants to go to a local supermarket that sells fruits with cheaper prices, but she does not have enough time to spare due to her busy school life. She started to look for how to get fruits with cheaper price.
- Jack : He is 35 years old and currently living in Mokpo, working as an owner of apple farm. Due to the bad weather in the summer, he harvested many ugly apples. The taste of the apples was no different from other apples, but the farmer had to throw away many ugly apples because they were out of the marketable standard because of their appearance. He started to think how not to waste those apples.

In this situation, Diana and Jack get to know about our application.

Diana, who is a user of IPhone10, downloads our application from App Store and opens the application. When she opens the application, a main login page comes up. As she does not have an account yet, she clicks register button. She needs to write email, passwords, name, address, and phone number.

Jack, who is a user of Galaxy flip3, downloads our application from Play Store and opens the application. When he opens the application, a main login pages comes up. As he does not have an account yet, he clicks register button. He needs to write email, passwords, name, profile picture, address, phone number, and upload farmer authentication document.

When Diana and Jack both try to search address, Road Name Address Open API returns selected address data, so that Diana and Jack can write down their address more conveniently.

When Diana and Jack both authenticate themselves through phone number, Phone Number verification API is involved returning result code if the verification performs correctly by them.

Jack authenticates himself as a farmer through uploading a relevant document. Administration receives this document and checks it. If the document is the proper one, the farmer can end the register procedure.

After registration procedure is done, they are directed to the main service page.

After log-in, Diana can see a list of lives on the main screen. She accesses through LiveStream API one of lives she wants to watch. She can leave chat during live to ask questions about the products. If she likes the product, she further can see the information of the product in the product detail page, clicking the bottom below the screen. If she likes it, she can choose options and proceeds payment. When she clicks the payment button on the detail page of the product, the payment

request is passed on to the Payment API.

If she wants to change or cancel order, she can access order confirmation in mypage. She clicks one of orders from the list and can either change options or cancel the order through clicking change/cancel buttons.

If the order is successfully done, she can check delivery status of the order request. The URL containing information of delivery status is offered by Delivery API.

After log-in, Jack can register live and products upon the live in mypage. The information of the live is passed to Live Streaming API. If Jack wants to change information of the live or products he registers, he can also access mypage and change options for them.

When Jack is ready to start live, in mypage after clicking the live he wants to start, he can begin the live. Now he can communicate with customers who enter the live and can read chat that customers leave. If he wants to stop live, he can click end button to stop streaming.

After that, he can check orders he received in order management. He can confirm customers' orders in order management page. Also, he can change customers' order information such as name or address if customer asks him to do. If a customer wants to change an address, he can easily search the address through Road Name Address Open API. Finally, Jack can request delivery, uploading an extracted file on the delivery company website.

If Diana and Jack want to log out or remove the account, they can click 'Mypage' and 'Edit Profile' button one after another. They can either click remove account or logout button.

Our application can be used for many people like Diana who wants to buy fruits with cheaper price and for farmers like Jack who wants to sell discarded fruits.

II-d. Background (Market Research)

[못난이 농산물 구매 경험 및 인지]

(n=2,000)

구분	응답자수(비율)	
있다	1,210명(60.5%)	
없다	790명(39.5%)	
	알지만 구입하지 않음	565명 (71.5%)
	몰라서 구입하지 않음	225명 (28.5%)

[구입이유]

(단위 : 명, %)

구분	응답자수(비율)
못난이 농산물의 가격이 더 저렴해서	561(46.4)
맛·식감 등 품질이 일반 농산물과 차이가 없어서	344(28.4)
주스 등 외관이 중요하지 않은 요리를 위한 용도로 사용하기 위해서	172(14.2)
최근 언론보도 등을 통해 못난이 농산물에 대한 호기심이 생겨서	87(7.2)
음식물 폐기물 저감, 농가소득 증대 등 착한 소비 실천을 위해서	44(3.6)
기타	2(0.2)

[구입 종류(중복응답)]

(단위 : 명, %)

구분	응답자수(비율)
과실류(사과, 배 등)	880(72.7)
서류(감자, 고구마 등)	625(51.7)
채소류(호박, 가지 등)	482(39.8)
버섯류(느타리, 양송이 등)	125(10.3)
기타	5(0.4)

[재구매 의사]

(단위 : 명, %)

구분	응답자수
예	1,155(95.5)
아니요	11(0.9)
잘 모르겠다	44(3.6)

[추후 구매의사]

(단위 : 명, %)

구분	못난이 농산물에 대해 알고 있으나, 구매하지 않은 집단(n=565)	못난이 농산물에 대해 모르고 있었던 집단(n=225)
예	486(86.0)	147(65.3)
아니오	5(0.9)	10(4.4)
잘 모르겠다	74(13.1)	68(30.2)

[활성화 방안]

(단위 : 명, %)

구분	응답자수(비율)
쉽게 구입할 수 있도록 구매 접근성 확보	673(55.6)
못난이 농산물 인지도 향상을 위한 홍보 강화	209(17.3)
농민-소비자 간 직거래 시스템 구축을 위한 정부지원	158(13.1)
못난이 농산물에 대한 긍정적 이미지 제고 등 인식개선 활동	131(10.8)
수요 확보를 위한 제품 및 레시피 개발·홍보	32(2.6)
기타	7(0.6)

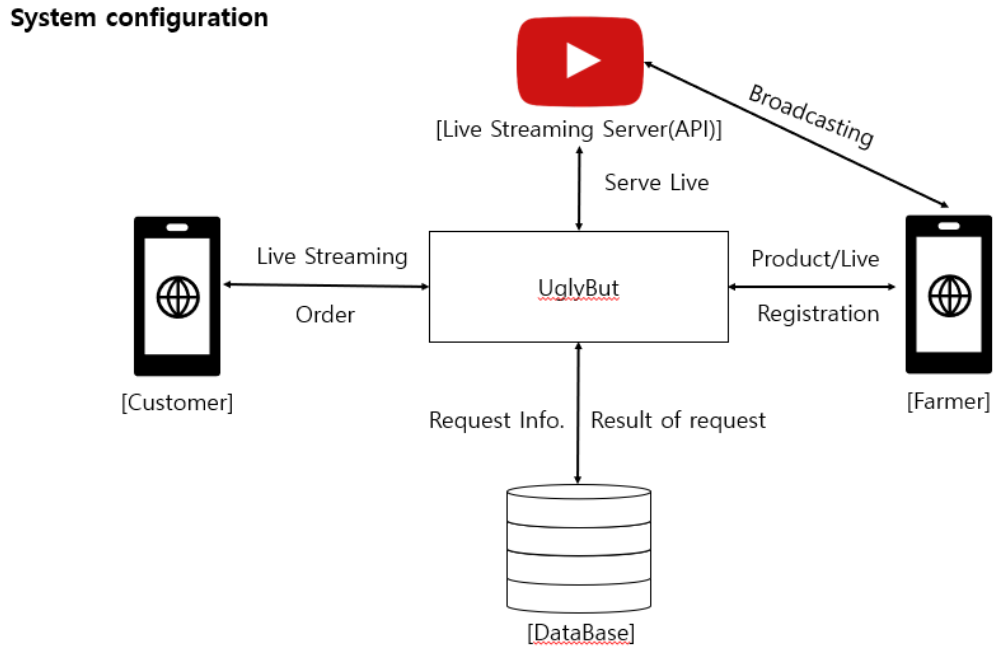
This is the analysis for the consumer perception of ugly agricultural products. The statistics represented the kind of the agricultural product (fruits or vegetables, etc.) that the consumer purchased mainly, and the high satisfaction with the ugly agricultural products. The analysis shows the application is needed to market

	UglyBut	Live Commerce Platform	Ugly agricultural product market	Online food market	Food market with multi channel
Sell ugly agricultural product	O	X	O	X	△
Live & Communication	O	O	X	X	△
Filtering for fruits	O	X	X	X	X
Live support kit	O	X	X	X	X

We investigated the similar service with our application. Even though they have similar functions, the focus of each app is different, and the main concept of our application (i.e., sell ugly fruits with live-commerce) is unique.

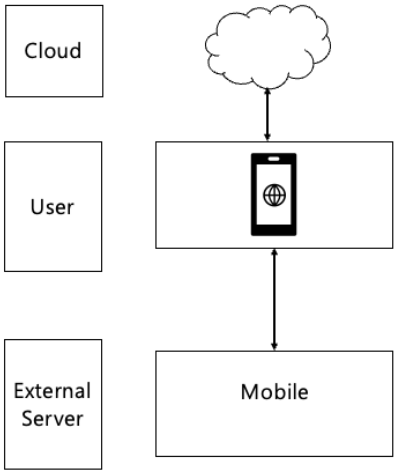
II-e. System Configuration

1) System Configuration



2) Hardware and Network Configuration

Hardware and Network configuration using cloud service



Client
• Mobile 4G LTE, Android 11, One UI 3.1.1, CPU : Snap Dragon 855+ GPU : Adreno 640 8GB / 256 GB / 330 mAh
Cloud
• AWS EC2 • Instance : t2.micro • vCPUs : 1 • Memory : 1GiB

3) Software Configuration

Software Configuration with cloud

Client	Server (Cloud)
Kotlin	Server OS : Amazon Linux AMI Framework : Spring Framework Test Server : Tomcat 8.0 DB : MariaDB

II-f. Team Roles

- Sumin: Market Research, Functional Requirements (Mypage(farmer), Delivery, Live (Customer)), Scenario, DB, Design
- Jeewon: Problem Description, Functional Requirements (Account, Mypage(customer)), Scenario, UML Diagram, User Interface Design
- Jina: Functional Requirements (Mypage(farmer), Live Management), Scenario, Interface Design, Detailed Design

II-f. Work Breakdown Structure

(1) Planning

Task	
Scheduling	Establish the project schedule and estimated efforts with splitting the sub-tasks following the software life cycle

(2) Requirement Analysis

Task	
Market Research	Research analysis and statistics for user demand, competitors, profitability
Functional Requirement	Analysis the functionalities of the service
Non-functional Requirement	Analysis the system properties and constraints

(3) Design

Task	
UML Diagram	Design the Usecase Diagram. Class Diagram, Sequence Diagram
User interface	Design the UI Wireframe that describe main functions
DB Design	Design the database structures and their relationships

(4) Implementation

Task	
Front-end	Component architecture handling, Data state managing
Back-end	Database managing, Server deployment, Handle API

(5) Testing & Integration

Task	
Unit test	Implementation test by original developer
Alpha test	Testing by customer at developer's site
Beta test	Testing by customer at customer's site

(6) Maintenance

Task	
Consistent function update	Functions add for following the change of

	requirements
Security managing	Protection of users' personal information such as phone number, address, etc.
Error Fixing	Fixing discovered errors and latent errors
Reliablity & Performance update	Improve reliability and performance to prepare for a growing number of customers

III. Domain Specification

- When a farmer joins the application, he/she needs to be a legally proved farmer having a business number.
- When a farmer registers products, he/she can only do it in 'kg' units.

IV. Functional Requirements

Number	Category	Requirement Name
FR-01	Account	Register
FR-02		Login
FR-03		Logout
FR-04		Remove Account
FR-05		Farmer Authentication
FR-06		Edit Profile
FR-07	Mypage(customer)	Order Confirmation
FR-08		Order Cancelation
FR-09		Change Order
FR-10	Mypage(farmer)	Register Live
FR-11		Manage Live
FR-12		Register Product
FR-13		Manage Product
FR-14		Order Management
FR-15		Order Confirmation
FR-16	Live Management	Start Live

FR-17		Stop Live
FR-18	Delivery	Delivery Status Request
FR-19	Live(customer)	Live Access
FR-20		Chat
FR-21		Payment Request

FR-01	
Req. Name	Register
Req. Category	Account
Req. Description	- The user (customer, farmer) needs to be able to registers application first - The user needs to be able to create an account with a password, an email, name, address, phone number, and profile picture - The user needs to find their address via Road Name Address Open API - The user needs to be certified their identification using a cell phone verification via Phone Number verificiation API - The system needs to ask for the agreement of the new user on the terms of use of his/her personal data - A successful registration redirects the user to the login page

FR-02	
Req. Name	Login
Req. Category	Account
Req. Description	- The user (customer, farmer) needs to be able to log in to the application using his password and email which they provided during the registration - A successful login redirects the user to the service

	main page - If the login is not successful, the user is notified "Try again"

FR-03	
Req. Name	Logout
Req. Category	Account
Req. Description	- The user (customer, farmer) needs to be able to log out of the application from edit profile section - A successful logout redirects the user to the login page

FR-04	
Req. Name	Remove account
Req. Category	Account
Req. Description	- The user (customer, farmer) should be able to remove account from edit profile section - A successful removing account redirects the user to the login page

FR-05	
Req. Name	Farmer Authentication
Req. Category	Account
Req. Description	- The user(farmer) needs to be able to authenticate themselves as a farmer through providing a relevant document such as business registration - The document provided is passed to administration - A successful authentication allows farmers to receive live kit set such as camera and to start live broadcast - If the authentication is not successful, the farmer is notified "Invalid authentication. Try again"

FR-6	
Req. Name	Edit Profile
Req. Category	Account
Req. Description	- The user (customer, farmer) needs to be able to change their personal data when they provided during registration - The user needs to be able to change password, name, address, phone number, and profile picture - If the user wants to change address, he/she needs to find address via Road Name Address Open API - If the user wants to change his/her phone number, he/she needs to be certified their identification using a cell phone verification via NICE API - The user needs to be able to log out and remove account from this page - A successful editing profile applies the information that a user wants to change - If it is not successful, the user is notified "Invalid authentication. Try again"

FR-07	
Req. Name	Order Confirmation
Req. Category	Mypage(customer)
Req. Description	- The user needs to be able to check what she/he orders during live - The user needs to be able to check order number, product code, product name, quantity, and status of whether farmer confirms the order - A successful order confirmation brings all information of what users order and shows it to users

	- Order confirmation page includes order cancelation, change order and delivery confirmation buttons

FR-08	
Req. Name	Order Cancelation
Req. Category	Mypage(customer)
Req. Description	- The user needs to be able to cancel their orders before farmer confirms the orders - A successful cancelation removes a list of the order from order confirmation section

FR-09	
Req. Name	Change Order
Req. Category	Mypage(customer)
Req. Description	- The user needs to be able to change their orders before farmer confirms the orders - A successful change his/her options of the order only during live broadcast

FR-10	
Req. Name	Register Live
Req. Category	Mypage(farmer)
Req. Description	- The user needs to be able to register live information before starting live - The user needs to be able to pre-register a live broadcast with title, start date, start time, registered product information - If user missed some input information, the live broadcast registration will be denied - Using Live Streaming API, the farmer needs to be able

to set his/her broadcast

FR-11	
Req. Name	Manage Live
Req. Category	Mypage(farmer)
Req. Description	- The user needs to be able to check registered live list and each specific information - The user can modify live information such as registered live title, start date, start time, registered product information - Using Live Streaming API, user can update his/her registered broadcast and stream information - A successful modification changes information of the registered live from live list section

FR-12	
Req. Name	Register Product
Req. Category	Mypage(farmer)
Req. Description	- The user needs to be able to register information of their goods for sales. It is used to register the live broadcast information - The user needs to be able to input product name, price, quantity, type(code), image, and additional description of the product - Product codes are attached in appendix

FR-13	
Req. Name	Manage Product
Req. Category	Mypage(farmer)
Req. Description	- The user needs to be able to check registered product list and each specific information

	- The user needs to be able to modify registered product information - The user can modify product information again such as registered name, price, quantity, type(code), image, and additional description of the product - A successful modification changes information of the registered product from product details section

FR-14	
Req. Name	Order Management
Req. Category	Mypage(farmer)
Req. Description	- The Farmer can check the orders got from the Lives - The orders are distinguished by the Lives that the farmer opened - The orders of each Live are extinct after 14 days of delivery is started - The Farmer can change/cancel the orders options - The Farmer can extract the order information such as the customer's address, phone number

FR-15	
Req. Name	Order Confirmation
Req. Category	Mypage(farmer)
Req. Description	- The Farmer can confirm the orders got from the Lives before delivery - If the Farmer confirms the orders, the order information (name, phone number, address, etc.) can be exported to *csv* format for passing Logistic Program

FR-16	
Req. Name	Start Live
Req. Category	Live management
Req. Description	- After the registered live start time, the user needs to be able to start that live - When the before the registered live time, the user cannot start that live - Live Streaming API transmits video on user's video stream - The farmer installs an AWS Elemental MediaLive software encoder to convert his/her video into a digital format to connect his/her stream to encoder, and then go live - Other users need to be able to watch the that live in real time - When farmer starts the live, the farmer needs to be able to see the chat of the viewers(customers) and take orders - The farmer needs to be able to check the remaining quantity of products in real time during the live broadcast

FR-17	
Req. Name	Stop live
Req. Category	Live management
Req. Description	- If the registered product is exhausted or user(farmer) wants to end live, the user needs to be able to terminate live at any time

	- To end the stream, stop sending content from farmer's encoder - Live Stream API automatically ends the broadcast about a minute after user stop transmitting video - After the live end, other users cannot see that live - When the live ends, product order information is needed to be able to pass to the order management section

FR-18	
Req. Name	Delivery Status Request
Req. Category	Delivery
Req. Description	- The Farmer can request the status of delivery of the orders he delivered by the Delivery service - If the Farmer requests the delivery status of one order, the order's status is shown by inputting the order's tracking number - The Delivery API the service use is "Delivery Tracker" - The service passes the *carrier_id* (Predefined delivery service company id), *track_id* (tracking number) with the URL of the API

FR-19	
Req. Name	Live Access
Req. Category	Live (customer)
Req. Description	- The Customer can access the Live that goes in real-time - The Live isn't shown to customer after it is extinct

FR-20	
Req. Name	Chat

Req. Category	Live (customer)
Req. Description	- The Customer can chat on the screen of Live when Live goes - If the customer inputs the message at the chat bar below the screen and clicks the send button, the chat sends to Live Streaming API and pops up on the screen - The Customer can check the chats of his own/others - The Chats are extinct when the Live is over

FR-21	
Req. Name	Payment Request
Req. Category	Live (customer)
Req. Description	- The Customer can request payment for their order - The Customer only orders the product when the Live goes on - The customer can watch the product detail page that connected to the Live screen - The Customer selects the product, product options, and payment method in the services for payment requests - The Payment API, "Import", processes the order with order information such as name, address, phone number, product, product options, message

V. Nonfunctional Requirements

Req. Number	Req. Type	Requirement Name	
NFP-01	Product Requirement	Usability	
NFP-02		Efficiency	Performance
NFP-03			Space
NFP-04		Reliability	
NFP-05		Portability	
NFO-01	Organizational Requirement	Delivery	
NFO-02		Implementation	
NFO-03		Standards	
NFE-01	External Requirement	Interoperability	
NFE-02		Ethical	
NFE-03		Legislative	Privacy
NFE-04			Safety

NFP-01	
Req. Name	Usability
Req. Type	Product
Req. Description	- When the user turns on the application, the main page is Log in page for quick access - The user is only able to access the application using data or Wi-Fi - If the user permits using personal information for automatic login, as soon as the user clicks the application icon, the user is able to log in straight away - If there is no data or Wi-Fi, an error message says 'Network Problem'. - Once the user logs in, tabs(menus) at the bottom of

	the screen are divided - The user's information that she/he input when she/he joined the application is automatically saved in my profile section - All the functions are available all the time.

NFP-02	
Req. Name	Efficiency - Performance
Req. Type	Product
Req. Description	- If there are any errors, the system alerts to users straight away and goes to the initial state - Quick responsiveness through touching screen is guaranteed

NFP-03	
Req. Name	Efficiency - Space
Req. Type	Product
Req. Description	- Unnecessary and useless memories are deleted every day - My orders are deleted every 30 day under users' permission - Data that the user wants to delete is deleted straight away from user's interface but is deleted after 7 days from database

NFP-04	
Req. Name	Reliability
Req. Type	Product
Req. Description	- All the data are saved automatically when users terminate the application - All the data are maintained on the background of the

	application - Under the user's permission, the application needs to be linked to device's gallery or files

NFP-05	
Req. Name	Portability
Req. Type	Product
Req. Description	- The application is compatible with both Android and Apple users because the application is developed by ReactNative and Flutter

NFO-01	
Req. Name	Delivery
Req. Type	Organizational
Req. Description	- The application is registered in Google Play Store and App Store - Users can download the application either from Google Play Store or App Store

NFO-02	
Req. Name	Implementation
Req. Type	Organizational
Req. Description	- Flutter - AWS - ReactNative - Maria DB

NFO-03	
Req. Name	Standards
Req. Type	Organizational

Req. Description	- System identifies users by his/her login information, email and password or, under the user's permission, any Bio information.

NFE-01	
Req. Name	Interoperability
Req. Type	External
Req. Description	- The user needs to be able to use application on any devices if the user is able to download either from Google Play Store or App Store

NFE-02	
Req. Name	Ethical
Req. Category	External
Req. Description	- The user needs to be able to certify themselves via verification via mobile phone - The user needs to be able to report unwholesome lives or chats

NFE-03	
Req. Name	Legislative - Privacy
Req. Category	External
Req. Description	- When joining the app, the user needs to be able to receive consent of using personal data - The user needs to be able to receive permission of using Bio information - The user needs to be able to give permission of using device's data

NFE-04

Req. Name	Legislative - Safety
Req. Category	External
Req. Description	- To prevent the leakage of information, the information is encrypted

VI. Scenarios

Scenario Title	Register
Actors	Customer, Farmer
Initial Condition	The user has no account
Final Condition	Successful registration
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If the user is not able to authenticate their identity, "Invalid authentication. Try again" message pops up 3. If the user inputs a duplicated ID, "Duplicated ID" message pops up 4. If the user inputs a different password for validation, "Validation Fail" message pops up
Process	1. The user opens the application 2. The main page of application comes up 3. The user clicks 'Join' button 4. The user needs to set email and clicks duplication check 5. The user needs to set passwords twice for validation 6. The user needs to set name 7. The user needs to set address through clicks 'Search 'button A. In server-side, Open API URL is called B. In client side, the user input their specific address which is used as inputYn parameter in API and click 'Search' button C. In server-side, api returns selected address data through roadFullAddr, jibunAddr, zipNo, addrDetail, totalCount, currentPage, countPerPage, errorCode, errorMessage

	8. The user needs to set phone number 9. The user clicks verification button A. In client-side, the user input personal information B. In server-side, api extracts requestno, returnurl, sitecode, authtype, mobilceco, businessno, methodtype, popupyn, receivedata C. In client-side, api returns resultcode(String type) if the verification performed correctly D. In server-side, authentication proceeds after encrypting the request data in the form of JSON 10. The user can see a message, 'Verification is successfully done.' 11. The user should be able to set a profile picture if they want 12. The user needs to click 'Yes' button for the agreement of the terms of use of his/her personal data 13. After registration procedure, the user is directed to the main service page

Scenario Title	Login
Actors	Customer, Farmer
Initial Condition	The user has an account and is on the login page of the app
Final Condition	The user successfully logs in and on the main service page of the app
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If the user put wrong email or password, "Try again" message pops up 3. If the user searches an invalid address, no address search result comes up
Process	1. The user can log in through email and password 2. After log-in procedure, the user is directed to the main

	service page

Scenario Title	Logout
Actors	Customer, Farmer
Initial Condition	The user already logs in the application
Final Condition	The user successfully logs out
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable
Process	1. The user clicks 'Mypage' and 'Edit Profile' button one after another 2. The user can see 'Logout' button and clicks it 3. The application asks the user about confirmation, 'Are you sure?' 4. If the user clicks 'Yes', then the user is directed to the login page

Scenario Title	Remove Account
Actors	Customer, Farmer
Initial Condition	The user already logs in the application
Final Condition	The user successfully removes account
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable
Process	1. The user clicks 'Mypage' and 'Edit Profile' button one after another 2. The user can see 'Remove Account' button and clicks it 3. The application asks the user about confirmation, 'Are you sure?' 4. If the user clicks 'Yes', then the account is deleted, and the user is directed to the login page

Scenario Title	Farmer Authentication

Actors	Farmer
Initial Condition	The farmer is on the process of registration
Final Condition	The farmer successfully authenticates their identity
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If the authentication is not valid, the farmer is notified "Invalid authentication. Try again"
Process	1. The farmer clicks 'Farmer Authentication' button 2. The farmer needs to upload relevant document such as business registration 3. The farmer needs to wait until the administration confirms the document 4. If the authentication is successful, then the farmer is directed back to registration procedure

Scenario Title	Edit Profile
Actors	Customer, Farmer
Initial Condition	The user already logs in
Final Condition	The user successfully can change their personal profile
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If the user searches an invalid address, no address search result comes up
Process	1. The user clicks 'Mypage' button 2. The user clicks 'Edit Profile' button 3. The user can change their password, name, address, phone number, and profile picture 4. If the user wants to change their password, they need to set twice for validation 5. If the user wants to change their address, the user needs to set address through clicking 'Search' button A. In server-side, Open API URL is called

	B. In client side, the user input their specific address which is used as inputYn parameter in API and click 'Search' button C. In server-side, api returns selected address data through roadFullAddr, jibunAddr, zipNo, addrDetail, totalCount, currentPage, countPerPage, errorCode, errorMessage 6. If the user wants to change their phone number, the user needs to authenticate phone number 7. If the user wants to change their phone number, set phone number and clicks verification button A. In client-side, the user input personal information B. In server-side, api extracts requestno, returnurl, sitecode, authtype, mobilceco, businessno, methodtype, popupyn, receivedata C. In client-side, api returns resultcode(String type) if the verification performed correctly D. In server-side, authentication proceeds after encrypting the request data in the form of JSON 8. The user can see a message, 'Verification is successfully done.' 9. After changing personal information, the user clicks 'Change' button 10. The user can see a message 'Change is successfully done.'

Scenario Title	Order Confirmation
Actors	Customer
Initial Condition	The user already orders item
Final Condition	The user can check the list of orders
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable

Process	1. The user clicks 'Mypage' button 2. The user clicks 'Order' button 3. The user can see a list of the orders 4. If the user clicks one of the orders, the user can check order number, product code, product name, price, quantity, and status of whether farmer confirms the order 5. The user can see cancel, change and delivery status, confirm, and export csv buttons 6. If the user clicks confirm button, the status of order is changed to confirmed 7. If the user clicks export csv button after confirming the order, the user can extract csv file to pass Logistic Program for delivery

Scenario Title	Order Cancelation
Actors	Customer
Initial Condition	The user already orders item
Final Condition	The user successfully cancels the order
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If the user tries to cancel the order after the farmer confirms the order, then the user fails to cancel the item and gets a 'You cannot cancel your order because your order is confirmed.' message
Process	1. The user clicks 'Mypage' button 2. The user clicks 'Order' button 3. The user clicks 'Cancel' button of the order he/she wants to cancel 4. The application asks the user about confirmation, 'Are you sure?' 5. If the user clicks 'Yes', then the order on the order

	confirmation list is removed

Scenario Title	Change Order
Actors	Customer
Initial Condition	The user joins live broadcast item
Final Condition	The user successfully changes the order
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If the user tries to change the order after the farmer confirms the order, then the user fails to change the item and gets a 'You cannot change your order because your order is confirmed.' message
Process	1. The user clicks 'Change' button of the order he/she wants to change during live broadcast 2. The user can change option 3. The application asks the user about confirmation, 'Are you sure?' 4. If the user clicks 'Yes', then the order on the order confirmation list is changed

Scenario Title	Register Live
Actors	Farmer, Live streaming API
Initial Condition	The user has no registered live
Final Condition	Successful registration
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. Except for the stream's title, user cannot create a

	stream successfully 3. If user set contentDetails.isReusable property value to false, then user need to create a new stream for each broadcast
Process	1. In my page, farmer can select 'Live Information' 2. The live details page of application comes up 3 The farmer clicks 'Register' button A. In client-side, the farmer input snippet.title, snippet.scheduledStartTime, status.privacyStatus, other contentDetails to add his/her broadcast and call the liveBroadcasts.insert method B. In server-side, api extracts his/her code and store the id from liveBroadcast resource. API response contains the liveBroadcast resource that farmer created C. In client-side, the farmer set snippet.title, cdn.frameRate, cdn.ingestionType, cdn.resoultion and call liveStream.insert method to create the video stream D. In server-side, api extracts & store the id from that liveStream resource. API response contains the liveStream resource that farmer created. E. Having created his/her liveBroadcast and liveStream resources, api calls liveBroadcasts.bind method to link the video bits and sets the id parameter to the broadcast ID obtained and the streamId parameter to the stream ID obtained. This method returns a liveBroadcast resource in the response body 3. The user can click 'Yes' button to save it 4. After registration procedure, the user is directed to the live details page

Scenario Title	Manage Live
Actors	Farmer, Live Stream API
Initial Condition	The user has registered live
Final Condition	Seeing registered information, successful modification
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If broadcast's status is active or error, the farmer cannot update any of the contentDetails properties and broadcast's scheduled start time
Process	1. In my page, user can select 'Live Information' 2. In live details page, user needs to be able to check registered live list and its information 3 If the user clicks a specific live and click 'Change' button, user can fill up that live information again A. In client-side, api calls snippet.description B. In server-side, the broadcast's title, description, and privacy status, and other metadata fields that are part of the broadcast's video resource is updated and return result string 4. After changing live information, the user clicks 'Yes' button 5. The user can see a message 'Change is successfully done' 6. After registration procedure, the user is directed to the live details page

Scenario Title	Register Product
Actors	Farmer
Initial Condition	The user has no registered product
Final Condition	Successful registration

Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If the users enter a product code that is not in the code classification table, "Invalid product code. Try again" message pops up
Process	1. In my page, user can select 'Product Information' 2. The product details page of application comes up 3. If user clicks 'Register' button, user can enter a new product information such as price, quantity, type(code), image, and additional description of the product 4. The user can click 'Yes' button to save it 5. After registration procedure, the user is directed to the product details page

Scenario Title	Manage Product
Actors	Farmer
Initial Condition	The user has registered product list
Final Condition	Seeing each product information, successful modification
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable
Process	1. In my page, user can select 'Product Information' 2. In product details page, user needs to be able to check registered product list and its information 3. If the user clicks a specific product and click 'Change' button, user can fill up that product information again 4. The user can click 'Yes' button to save modification 5. The user can see a message 'Change is successfully done'

	6. After registration procedure, the user is directed to the product details page

Scenario Title	Order Management
Actors	Farmer, Road Name Address Open API
Initial Condition	Order information collected
Final Condition	Correct Order information
Exceptional Case	1. If the network is not stable, "Network Problem" pops up and this message will not disappear until the network gets stable 2. If the change isn't reflected successfully because of DB transaction work, the "Change isn't reflected" message pop up and disappear
Process	1. The Farmer clicks the 'Mypage' button 2. The Farmer clicks the 'Order' button 3. The Farmer clicks the Order group (grouping by Live) list he wants to check 4. The Farmer can show an order detail by clicking an order in the order list of the group 5. If the Farmer wants to change the customer information such as phone number, he clicks the edit text and types the new information 5-1. If the farmer wants to change address, he/she needs to set address through clicking 'Search' button. In server-side, Open API URL is called. 5-2. In client side, the user input their specific address which is used as inputYn parameter in API and click 'Search' button 5-3. In server-side, api returns selected address data through roadFullAddr, jibunAddr, zipNo,

	addrDetail, totalCount, currentPage, countPerPage, errorCode, errorMessage. 6. The farmer clicks the 'Yes' button when finishes the changing 7. The changed information passed to the DB by POST method 8. In the DB, the changed information is inputted in the proper column followed by *order_id* (primary key) 9. If the change is reflected successfully, the message pop up to Farmer 10. If the Farmer wants to cancel the order, he clicks the 'Cancel' button on the order detail page 10-1. The changed status information passed to DB by POST 10-2. In the DB, the status column of the order followed by *order_id* (primary key) 10-3. If the cancel is reflected successfully, the message pop up to Farmer

Scenario Title	Order Confirmation
Actors	Farmer
Initial Condition	The farmer didn't get the order information file
Final Condition	The farmer gets the correct order information file
Exceptional Case	1. If the export work didn't work well because the network is not stable, the "Export failed" message pop up and disappears
Process	1. The Farmer clicks the 'Mypage' button 2. The Farmer clicks the 'Order' button 3. The Farmer clicks the Order group (grouping by Live) list he wants to check

	4. The Farmer can export the order information to *csv* format when he clicks the "export" button on the screen 5. The order information is got from DB and is written to csv file 6. The order information file saved to Farmer's

Scenario Title	Start Live
Actors	Farmer, Live Stream API
Initial Condition	The user does not start live-streaming
Final Condition	The farmer's video is delivered to the video player of the viewers in real time
Exceptional Case	1. If the average CPU utilization rate for the encoding operation exceeds 80%, the transmitting live may be constantly cut off or stuttering due to the CPU load. 2. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 3. If the user set the liveBroadcast resource's contentDetails.enableAutoStart property to false, then user need to call the liveBroadcasts.trainsition method 4. If user click 'start' button before the registered live start time, 'Not time for the live yet' message pops up.
Process	1 In my page, user can select 'live information' 2 In live details page, user needs to be able to click a specific live which want to start streaming 3 When user selects a specific live and click 'Start' button after registered start time, user can start live-streaming A. In client-side, api calls the liveStreams.list method to retrieve the liveStream resource associated with

	farmer's broadcast B. In server-side, api confirms that the status.streamStatus property's values is active, which indicates that server are receiving data from your encoder correctly C. And it automatically updates the broadcastStatus parameter value to live. And server automatically starts the broadcast 4 Connect farmer's stream to his/her encoder, and then go live 5 After starting procedure, the user is directed to the live-streaming page

Scenario Title	Stop live
Actors	Farmer, Live Streaming API
Initial Condition	The user doing a live-streaming
Final Condition	Successful end live-streaming
Exceptional Case	1. If network is not stable, "Network Problem" pops up and this message will not disappear until network gets stable 2. If the user set the liveBroadast resource's contentDetails.enableAutoStop property to false, then user need to call the liveBroadcasts.trainsition method to update the broadcast's status
Process	1. In live-streaming page, user can click 'End' button to stop streaming user's video at any time user want A. If contentDetails.enableAutoStop property to true, server automatically set the broadcastStatus parameter value to 'complete' using 'id' parameter value to the broadcast ID obtained

	B. Server will end the broadcast about a minute after user stop transmitting video and stop sending content from farmer's encoder 2. After stopping procedure, the user is directed to the live detail page

Scenario Title	Delivery Status Request
Actors	Farmer, Customer, Delivery API
Initial Condition	The user watches the 'Mypage' screen
Final Condition	The user goes to the external delivery status inquiry page
Exceptional Case	1. If the external page didn't work well, the external page shows an "Inquiry failed" message 2. If the tracking number is invalid, the external page shows "there's no delivery information" message
Process	1. The Farmer clicks the 'Mypage' button 2. The Farmer clicks the 'Order' button A. The Farmer clicks the Order group (grouping by Live) list he wants to check B. The Farmer clicks an order in the order list of the group and accesses to order detail page 3. The Customer clicks the 'Order' button A. The Customer clicks the order that he/she wants to check in the order list 4. The user (farmer, customer) clicks the "Delivery status" button A. The *carrier_id* (Predefined delivery service company id), and *track_id* (tracking number) are passed to the URL connected to the button B. The URL is offered by Delivery API, "Delivery Tracker"

	5. The user goes to the external delivery status page and check the delivery status

Scenario Title	Live Access
Actors	Customer, Live Streaming API
Initial Condition	The customer is on the main screen
Final Condition	The customer watches the Live stable
Exceptional Case	1. If the Network connection is bad, the Customer can watch the "Network condition bad" pop-up message with some delaying 2. If the Live encoder didn't work well, the Customer can watch the "Live inaccessible" message and can't watch the Live
Process	1. The customer clicks the Live thumbnail screen (button) in the ongoing live list on the main screen 2. The customer is connected to the Live server URL, and he/she can watch the Live streaming 3. If the customer wants to leave the live, he/she can click exit button.

Scenario Title	Chat
Actors	Customer, Live Streaming API
Initial Condition	The customer is on the main page
Final Condition	The customer sends the chat on the Live stable
Exceptional Case	1. If the Network connection is bad, the Customer can watch the "Network condition bad" pop-up message and Chat didn't be sent or delay 2. If the specified live chat is no longer live, then forbidden (403) error will be occurred

Process	1. The Customer opens the application 2. The main page of the service comes up 3. The Customer clicks the Live thumbnail screen (button) in the ongoing live list on the main screen to watch Live 4. The Customer can access the Live 5. The Customer input the chat using the editText area on the Live screen and clicks "SEND" button A. In client-side, api calls LiveChatMessages.insert method with snippet.liveChatId, snippet.type=textMessageEvent, snippet.textMessageDetails.messageText B. If successful, this method returns a liveChatMessage resource in the response body 6. The Customer can check the chats of his own/others

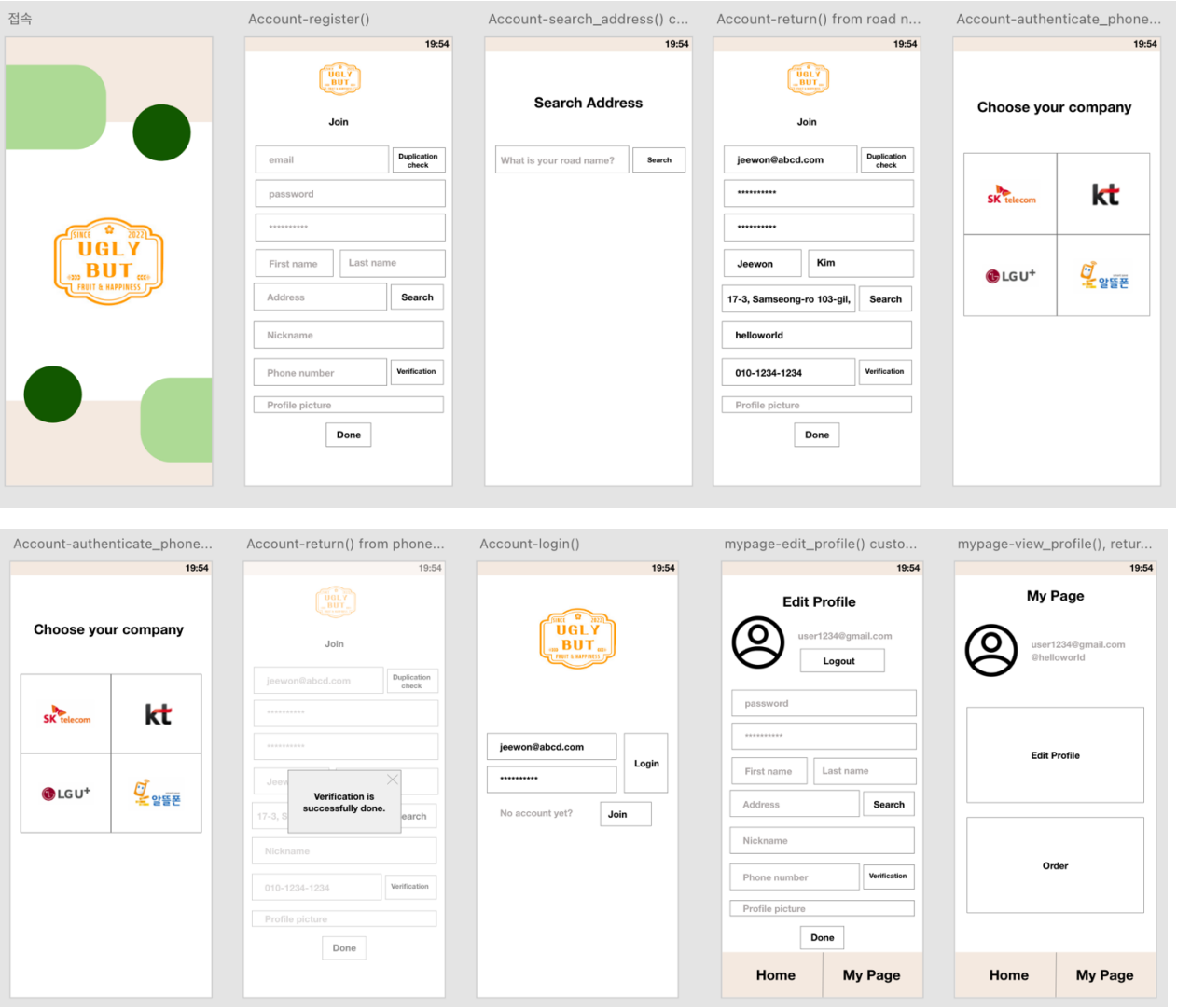
Scenario Title	Payment Request
Actors	Customer, Payment API
Initial Condition	The customer watches the live
Final Condition	The customer purchased the product successfully
Exceptional Case	- If the Network connection is bad, the Customer can watch the "Network condition bad" pop-up message and the screen returns to the product detail page - If the payment is invalid, the API returns an Error page and returns to the product detail page - If the payment is forgeries, the service shows the "Payment failed" message and returns to the product detail page
Process	1. The product detail page of the live appeared when the Customer clicked the shop button below the

screen

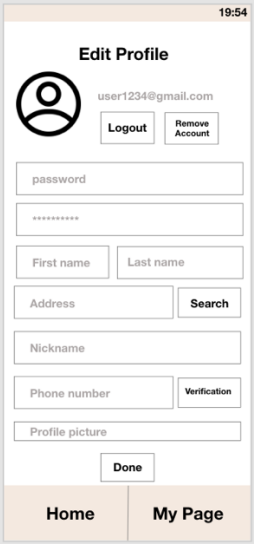
2. The Customer pick the options (e.g., amount) of the product
3. If the Customer clicked the payment button on the detail page of the product, the payment request (Including order information that the customer inputted) is passed on to the Payment API
4. The order information saved to DB grouping by the Live
5. In Client-side, the API returns *rsp* (response object) if the request with payment information passed correctly.
6. In Server-side, API extracts *imp_uid* (payment), *merchant_uid* (order) from *req* parameter. The API server verifies payment information and returns *status*.
7. The services receive the status. If payment status is '*success', the* service shows the order success screen.

VIII. UI/UX

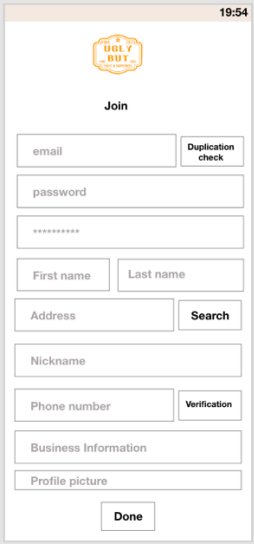
Account



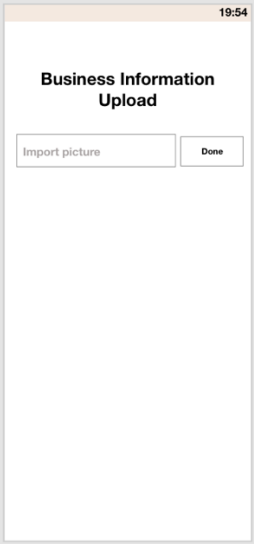
mypage-logout() customer f...



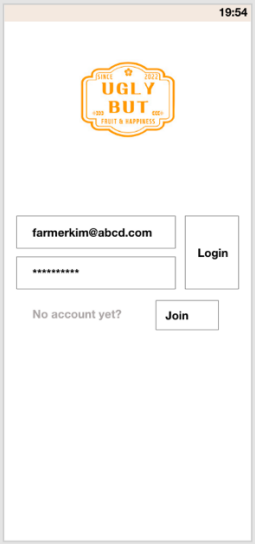
Account-register() farmer



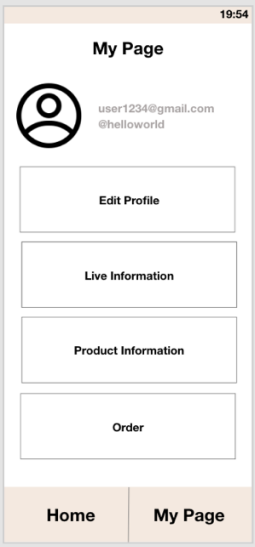
Account-send_authenticate_...



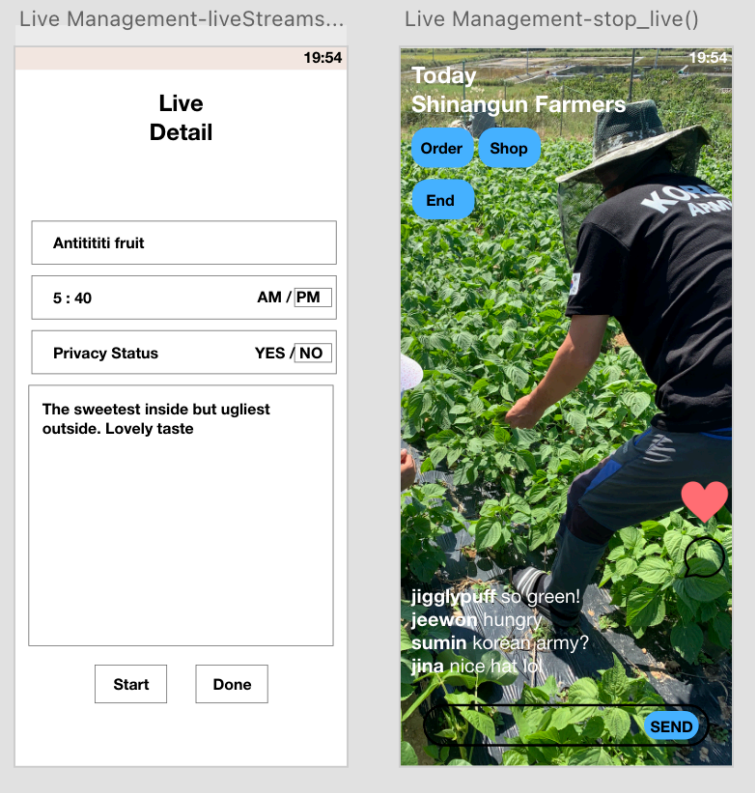
Account-login() farmer



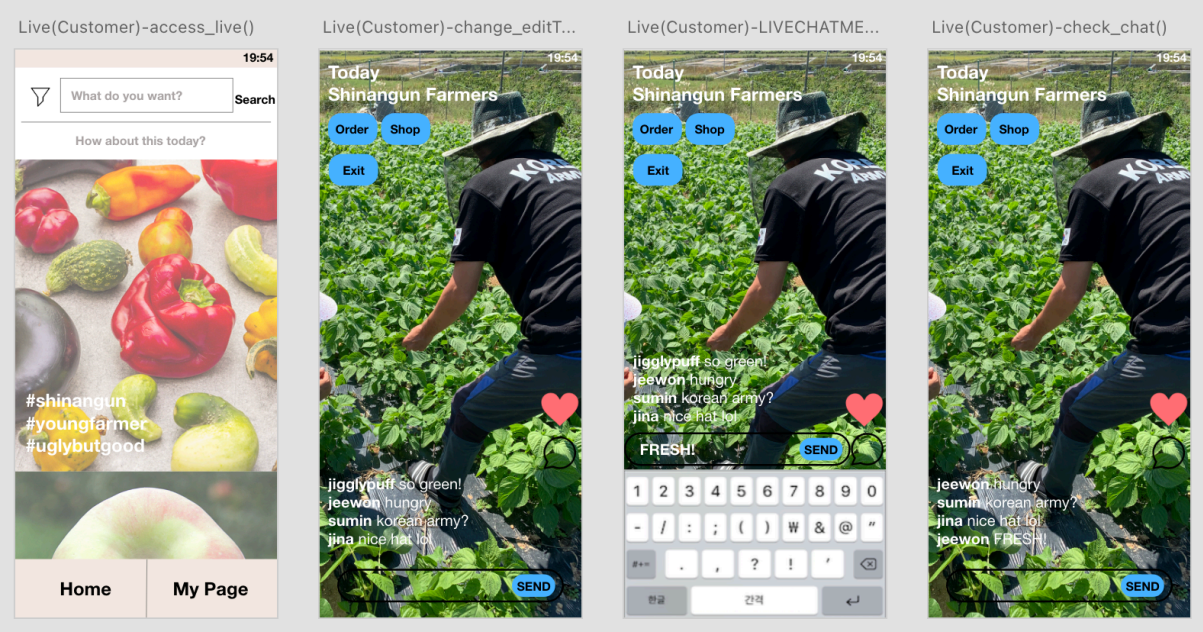
mypage-view_profile(), retur...



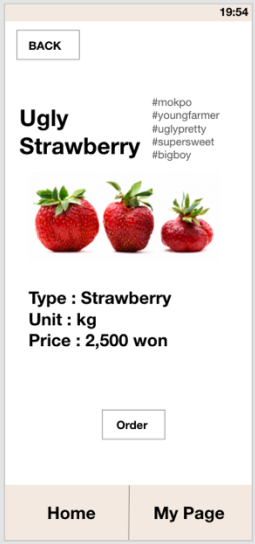
Live Management



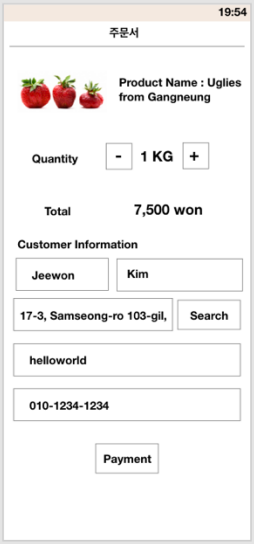
Live(Customer)



Live(Customer)-view_product...



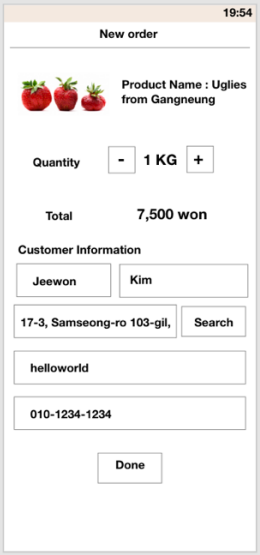
Live(customer)-choose_produ...



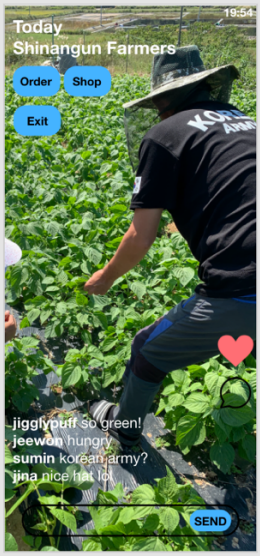
Live(customer)-Payment()

Live(customer)-cancel_order()

Live(customer)-change_order()

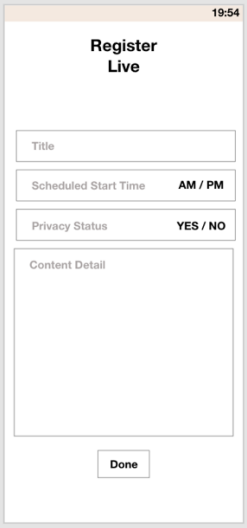


Live(Customer)-leave_live()

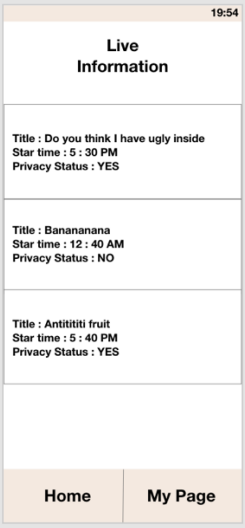


Mypage(Farmer)

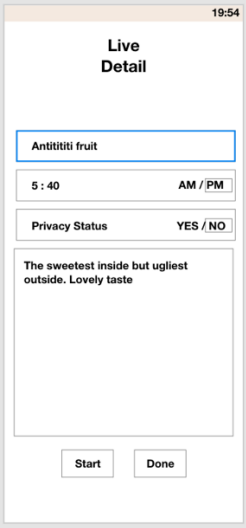
Mypage(farmer)-register_live()



Mypage(farmer)-get_live_info()

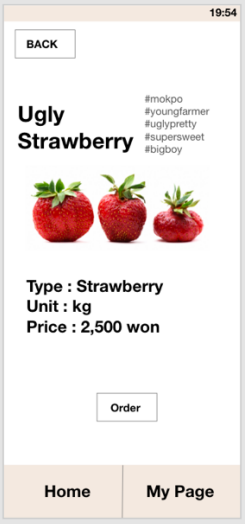


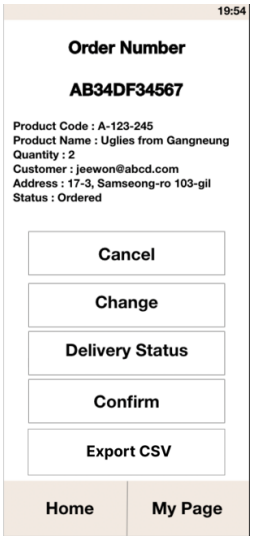
Mypage(farmer)-view_live_det...



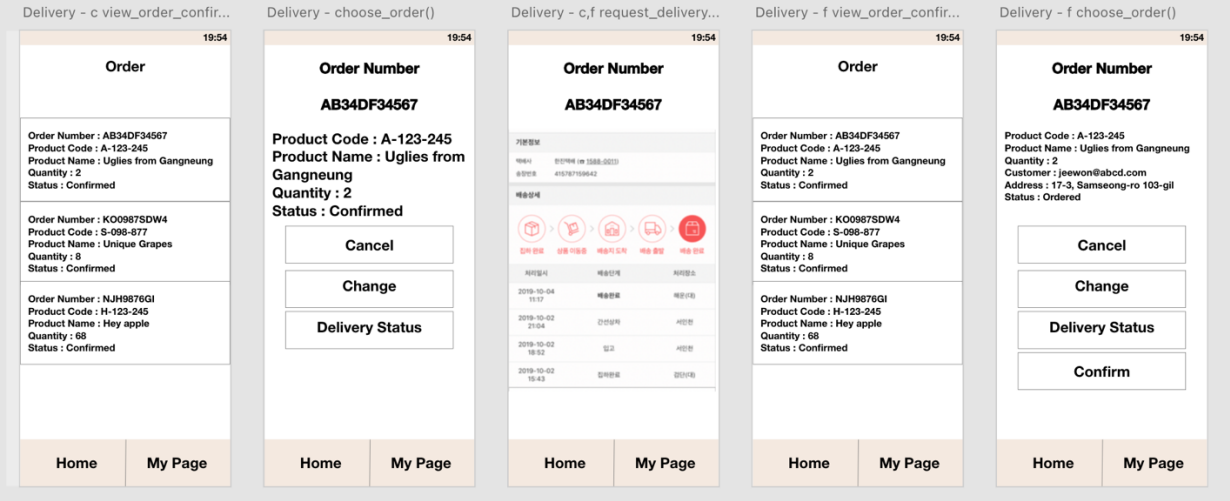
Mypage(farmer)-register_prod...

Mypage(farmer)-get_product...



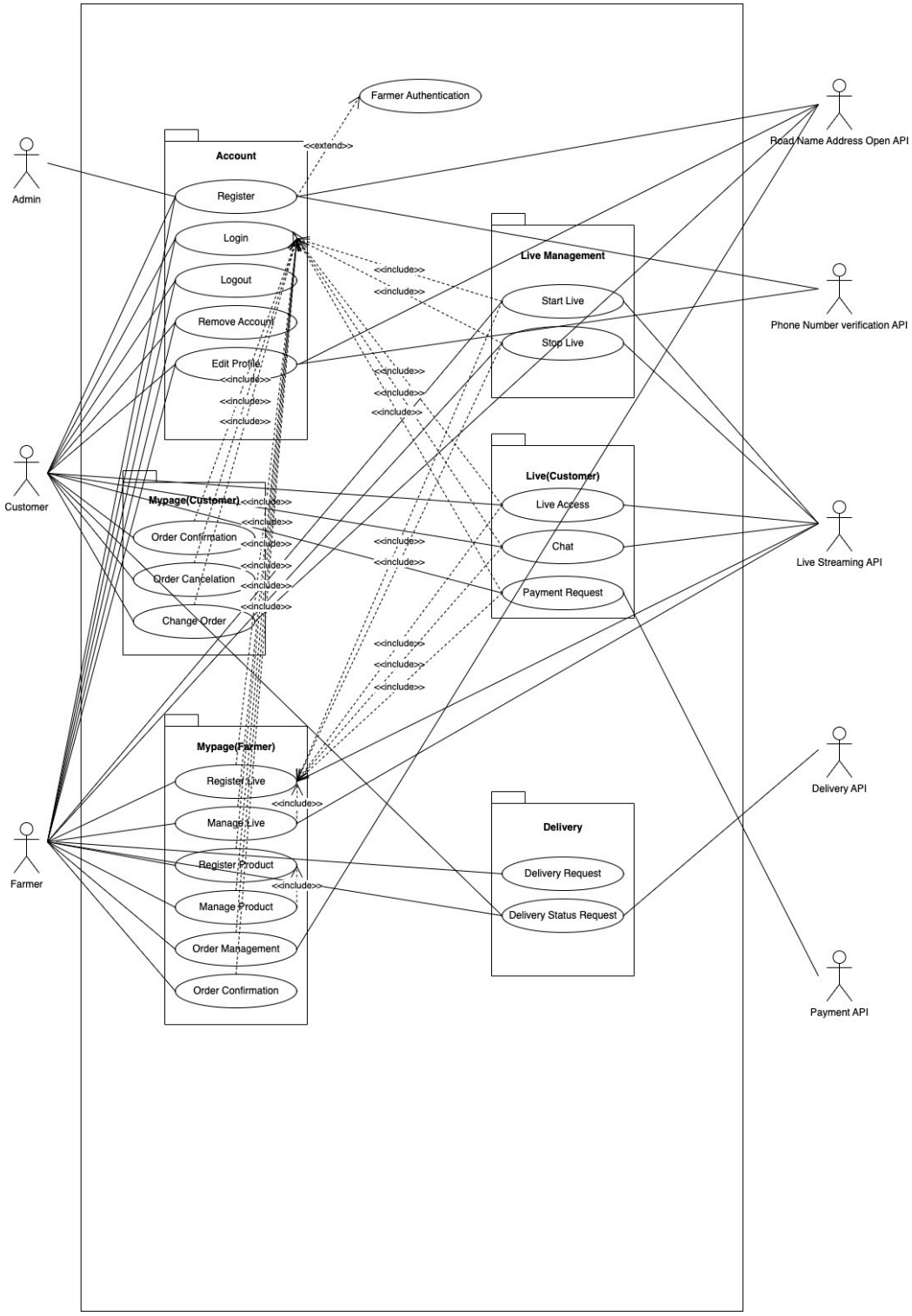


Delivery


Delivery - c view_order_confir...
19:54
Order
Order Number : AB34DF34567
Product Code : A-123-245
Product Name : Uglies from Gangneung
Quantity : 2
Status : Confirmed
Order Number : KO0987SDW4
Product Code : S-098-877
Product Name : Unique Grapes
Quantity : 8
Status : Confirmed
Order Number : NJH9876GI
Product Code : H-123-245
Product Name : Hey apple
Quantity : 68
Status : Confirmed
Home
My Page
Delivery - choose_order()
19:54
Order Number
AB34DF34567
Product Code : A-123-245
Product Name : Uglies from Gangneung
Quantity : 2
Status : Confirmed
Cancel
Change
Delivery Status
Home
My Page
Delivery - c,f request_delivery...
19:54
Order Number
AB34DF34567
Home
My Page
Delivery - f view_order_confir...
19:54
Order
Order Number : AB34DF34567
Product Code : A-123-245
Product Name : Uglies from Gangneung
Quantity : 2
Status : Confirmed
Order Number : KO0987SDW4
Product Code : S-098-877
Product Name : Unique Grapes
Quantity : 8
Status : Confirmed
Order Number : NJH9876GI
Product Code : H-123-245
Product Name : Hey apple
Quantity : 68
Status : Confirmed
Home
My Page
Delivery - f choose_order()
19:54
Order Number
AB34DF34567
Product Code : A-123-245
Product Name : Uglies from Gangneung
Quantity : 2
Customer : jeewon@abcd.com
Address : 17-3, Samseong-ro 103-gil
Status : Ordered
Cancel
Change
Delivery Status
Confirm
Home
My Page

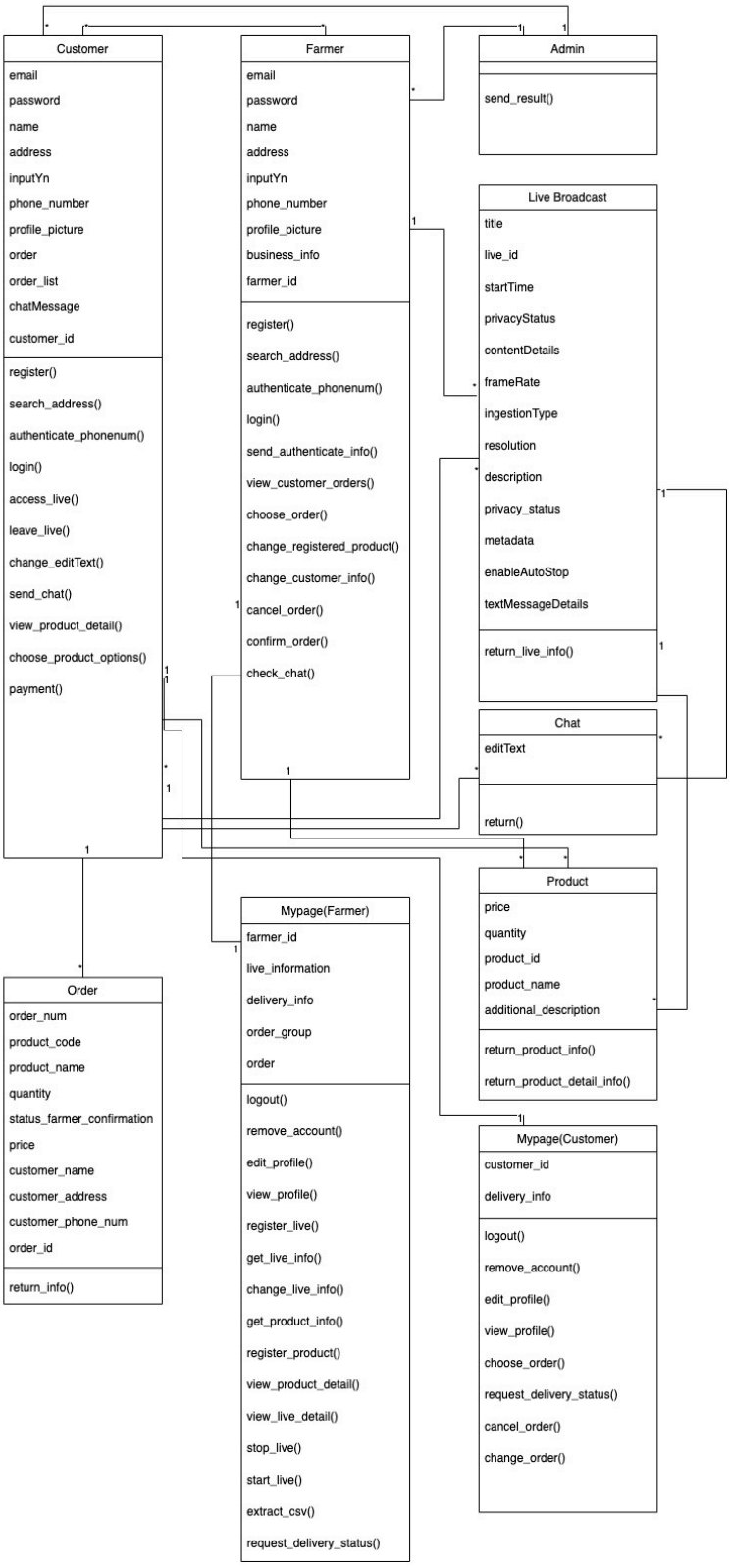
VIII. UML Design

VIII-a. Use Case Diagram

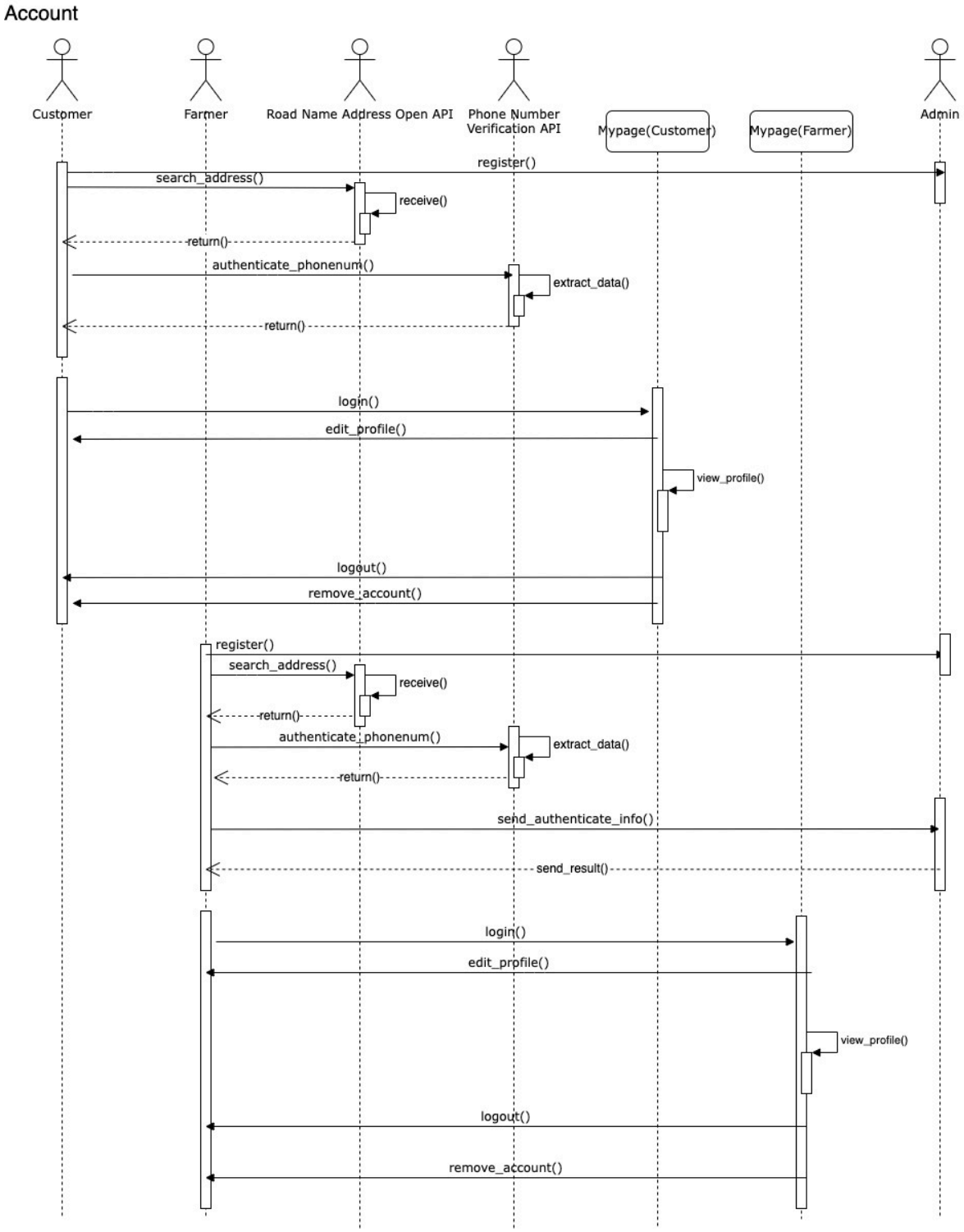


VIII-b. Class Diagram

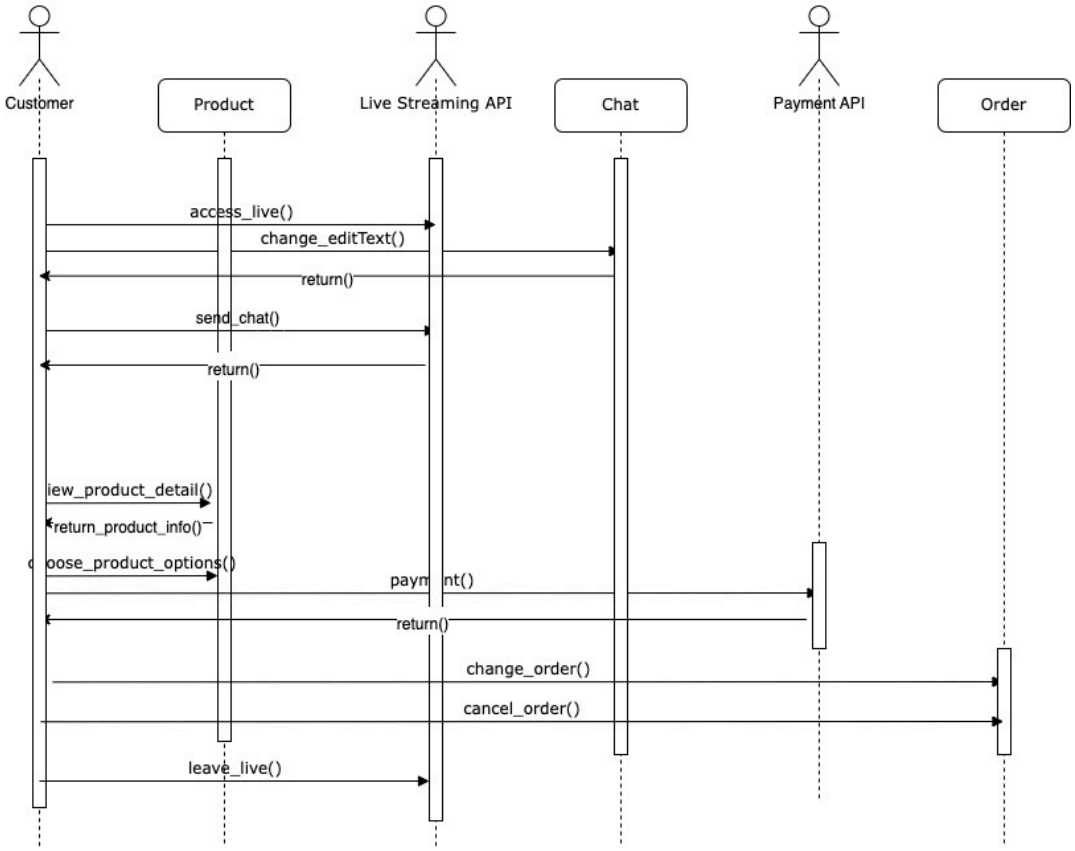
Class Diagram



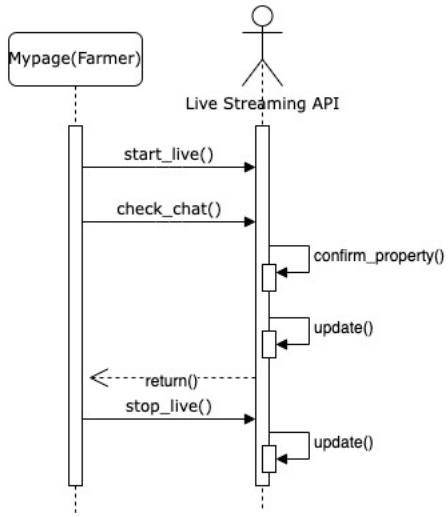
VIII-c. Sequential Diagram



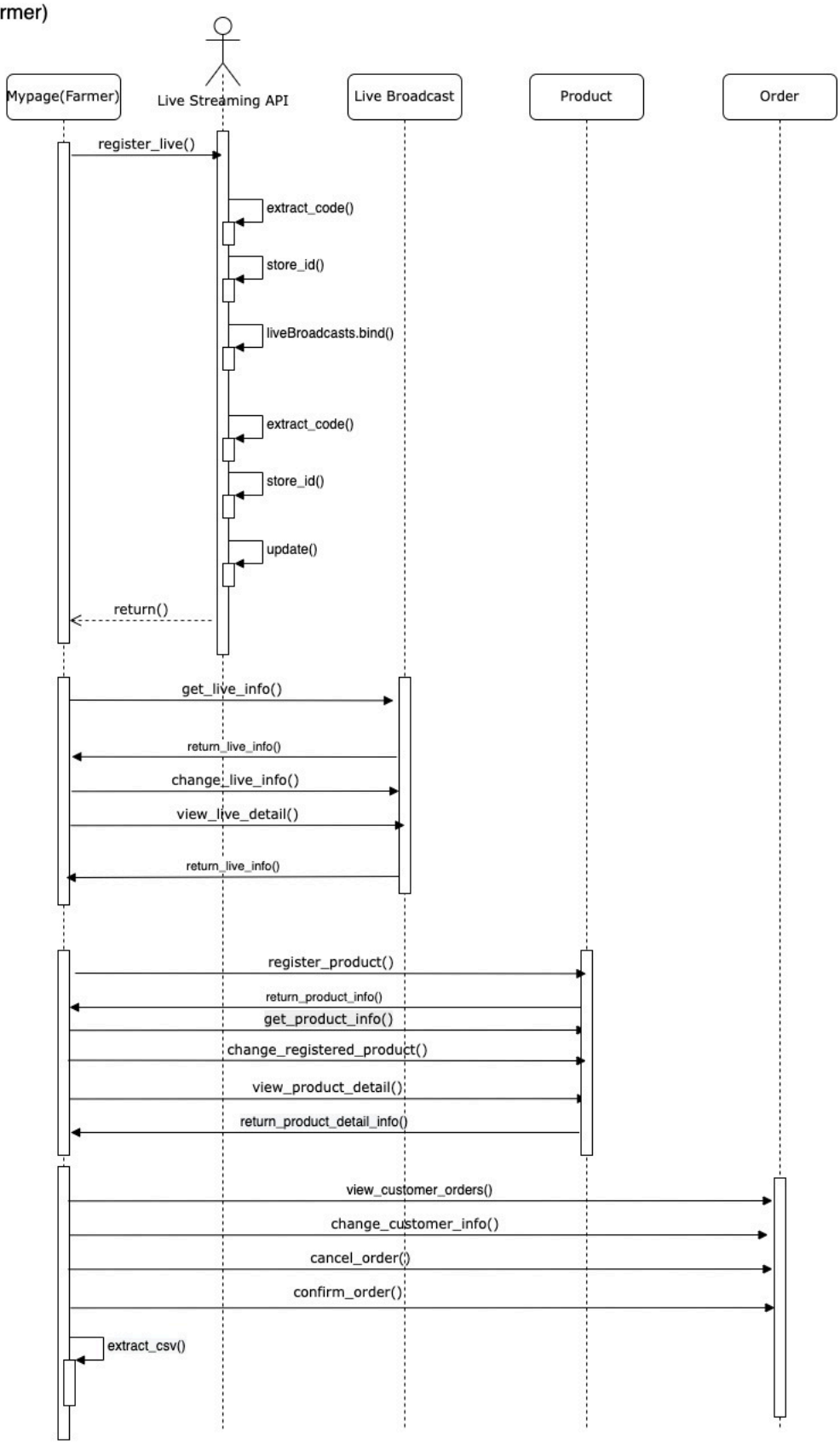
Live(Customer)



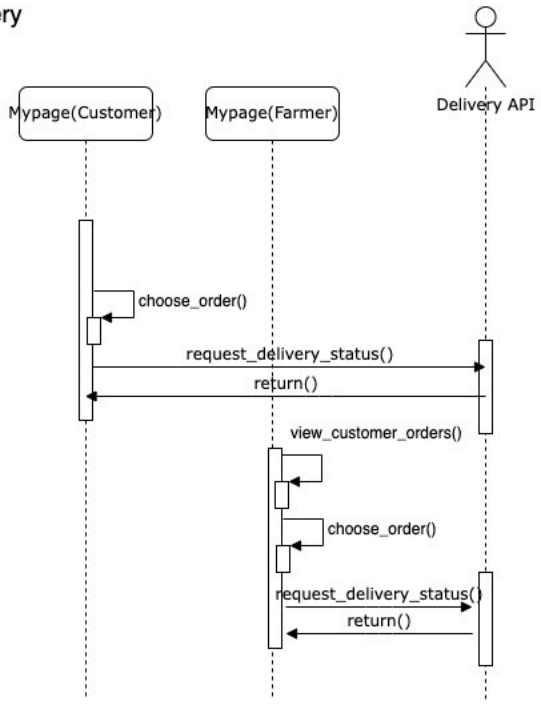
Live Management



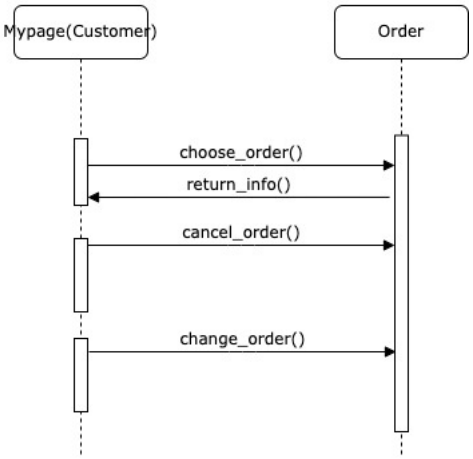
Mypage(Farmer)



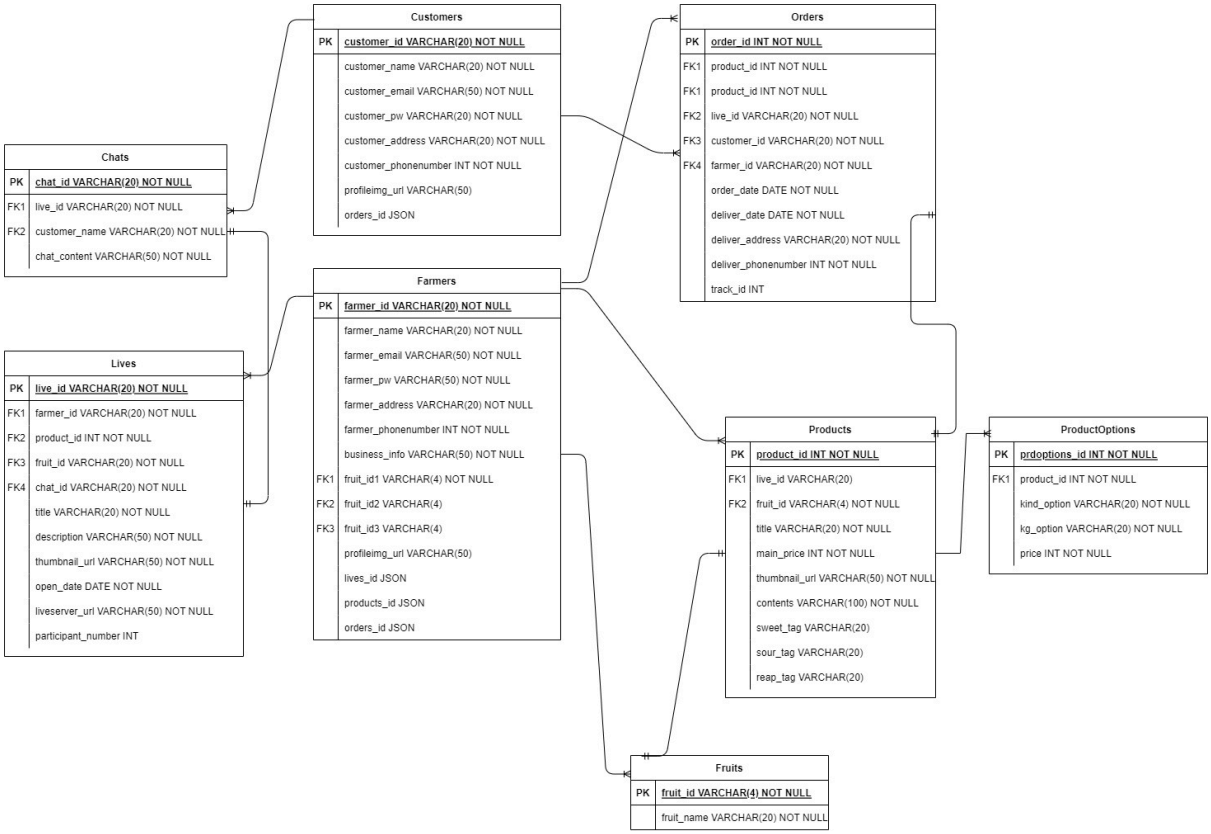
Delivery



Mypage(Customer)



IX. ER Diagram



X. Interface Design

Customer

function	Parameter	Return	process
register	String	void	Email, password, first_name, surname, nickname, inputYn, phone_number, profile_picture are inputted as a parameter which is necessary information for registration. And inputYn is sended to search_address()
	email, password, name, inputYn, phone_number, profile_picture	-	

			function, and get address. Phone_number is sent to authenticate_phonenum () function. And that information is stored in customers database.
search_address	String	string	inputYn is inputted as a parameter. It is passed to the Road Name Address Open API and real road address is returned.
	inputYn	Road address	
authenticate_phonenum	String	String	Phone_number is inputted as a parameter, and it is passed to NICE. And resultcode is returned which indicates this phone number is valid or not.
	Phone_number	resultcode	
login	String	void	Get the string type value of the email and password. After successful authorization, users can log in and enter the main page.
	email, password	-	
change_editText	String	String	Get the chatMessage and customerId as a parameter. Check there is same customer ID and its chat message in the chat database. If there is the same Id, replace the
	Changing chat message, name, live_id	Chat message	

			existing chat message with the input string.
access_live	String	Void	When the customer clicks a specific live, liveId and customerId are inputted as a parameter. And they are added to lives database. liveId is passed to check_chat() function.
	live_id	-	
leave_live	string	Void	When the customer clicks 'end' button, live ID and customer ID are inputted as a parameter. Then in lives database, remove the customer ID corresponding to the inputted live ID.
	live_id	-	
send_chat	String	Void	Get the inputted chat message and customer ID as a parameter. They are stored in lives database and chats database. And that inputted chat is passed to Live Stream API, and then it is shown in the live broadcast page.
	chat_id	-	
view_product_detail	String	void	When the user selects one product of the product list, selected product ID is used as a
	product_id	-	

			parameter. If it finds the product type, kg, product image on the products database.
choose_product_options	Int	Void	When the customer enters the quantity of product to order, product quantity is used as a parameter and stored in orders database.
	product_id, product_quantity, inputYn, phone_number	-	
payment	String	Boolean	If the Customer clicked the 'payment' button on the detail page of the product, the order information is passed on to the Payment API. If the payment status is 'success', it returns true. If not, false.
	order_id	True or False	

Farmer

register	String	void	Email, password, first_name, surname, nickname, inputYn, phone_number, and profile_picture are inputted as a parameter which is necessary infromation for registration. And inputYn is sended to
	email, password, name, inputYn, phone_number, profile_picture	-	

			search_address() function, and get address. Phone_number is sended to authenticate_phonenum() function. And that information is stored in customers database.
search_address	String inputYn	string address	inputYn is inputted as a parameter. It is passed to the Road Name Address Open API and real road address is returned.
authenticate_phonenum	String phone_number	String resultcode	Phone_number is inputted as a parameter, and it is passed to NICE. And resultcode is returned which indicates this phone number is valid or not.
login	String email, password	void -	Get the string type value of the email and password. After successful authorization, users can log in and enter the main page.
send_authenticate_info	String business_info	Void -	When farmer import business information image and click 'done'

			button in the business information upload page, business information image's uri is used as a parameter. And it is stored in farmers table with farmer ID.
view_customer_orders	String	List	When the farmer click 'order' button on the my page, the current user's id is used as a parameter. And all order list in the orders table is returned, and it is shown in the screen.
	farmer_id	Order list	
choose_order	String	List	When the farmer clicks a specific order in the order list, the corresponding order ID is used as a parameter. And order information in the orders table is returned.
	order_id	selected Order's information	
change_registered_product	String	Void	When the farmer selects a specific product in the register product page, product ID is used as a parameter. Then, find that product's information in the products table and
	product_id	-	

			change it with using update query.
change_customer_info	String	Void	customerID is used as a parameter. Then, find that customer's information in the customers table and change it with using update query.
	customer_id	-	
cancel_order	String	Void	According to the parameter, which is order ID, corresponding order data is deleted in all tables which have it.
	order_id	-	
confirm_order	String	Void	When the farmer selects an order in the order page and click 'confirm' button, order ID is used as a parameter.
	order_id	-	
check_chat	String	List	Get the string type value of the live Id. And show chat message list which included in that live ID on that live page.
	live_id	Chat message list	

Mypage(Farmer)

logout	Void	Void	When the user clicks

		-	the Logout button, the user state will be changed to logout state and will move to the login page.
remove_account	String	Void	The current user's id is used as a parameter, corresponding farmer data is deleted in all tables which have it.
	farmer_id	-	
edit_profile	String	Void	When a user enters Edit profile page, the current user's id is used as a parameter. Then, find that farmer's information in the farmers table and change it with using update query.
	farmer_id	-	
view_profile	String	Void	The current farmer's ID is inputted as a parameter. Each user's profile picture, password, name, address, nickname, email, phone number is returned on the profile page.
	farmer_id	-	
register_live	List	Void	Live information which

	Live information	-	user entered are inputted as a parameter and added to the lives table. Then liveBroadcasts.insert() and liveStream.insert() function in the Live Stream API is called.
get_live_info	String, Int	List	When the farmer enters the live information page, the farmer's Id is used as a parameter and the farmer's live list is returned.
	farmer_id	live list	
change_live_info	String	Void	Live ID is used as a parameter. Then, find that information of the live in the lives table and change it with using update query.
	live_id	-	
get_product_info	String, Int	List	When the farmer enters the register product page, the farmer's Id is used as a parameter and the farmer's product list is returned.
	farmer_id	Product list	
register_product	List	Void	Product information which user entered are inputted as a
	Product information	-	

			parameter and added to the products table.
view_product_detail	String	List	When the farmer enters register product page and clicks a specific product, the corresponding product ID is used as a parameter. It finds the product information on the products database table and gets the information as List type.
	product_id	Product information	
view_live_detail	String	List	When a farmer enters live information page and clicks a specific live, the corresponding live ID is used as a parameter. It finds the live information on the lives database table and gets the information as List type.
	live_id	Live information	
start_live	Void	Void	When the farmer click 'start' button in the live detail page, corresponding live ID
	-	-	

			is inputted and liveStreams.list method in the Live Stream API is called. Then the server automatically starts the broadcast.
stop_live	String	void	When the farmer clicks 'end' button in the live page, live ID is inputted as a parameter.
	live_iId	-	
extract_csv	String	Void	When the farmer clicks 'export csv' button in the selected order, corresponding product's ID is used as a parameter. If that order is confirmed before, this order's information is added on 'order.csv' file.
	product_id	-	
request_delivery_status	String	List	When the user selects a specific order in the order page and click 'delivery status' button, the string type value of the order ID is used as a parameter. It is passed to Delivery API. And it returns the delivery status and
	track_id	Delivery information	

			views on the screen.

Mypage (customer)

logout	Void	Void	When the user clicks the logout button, the Boolean value will be changed. Then this Logout method will get the Boolean value 0 or 1 and return logout state.
	-	-	
remove_account	String	void	The current user's id is used as a parameter, corresponding customer data is deleted in all tables which have it.
	customer_id	-	
edit_profile	String	void	customer ID is used as a parameter. Then, find customer's information in the customer table and change it with using update query.
	customer_id	-	
view_profile	String	void	The current customer's ID is inputted as a parameter. Each user's profile picture, password, name,
	customer_id	-	

			address, nickname, email, phone number is returned on the profile page.
choose_order	String	List	When the user clicks one of the orders, order ID is used as a parameter. Order number, product code, product name, quantity, status is returned.
	order_id	Order information	
request_delivery_status	String	List	When the user selects a specific order in the order page and click 'delivery status' button, the string type value of the order ID is used as a parameter. It is passed to Delivery API. And it returns the delivery status and views on the screen.
	track_id	Delivery information	
cancel_order	String	Void	When the customer selects an order and click 'cancel' button, selected order id is used as a parameter, corresponding order
	order_id	-	

			data is deleted in all tables which have it.
change_order	String	Void	When the customer selects an order and click 'change' button, selected order ID is used as a parameter. Then, find that order's information in the orders table and change it with using update query.
	orderId	-	

Admin

send_result	String	String	When farmer upload his/her business information image, then farmer ID is inputted as a parameter. If admin don't allow authentication, then return "Invalid authentication. Try again" message.
	farmer_id	Result message	

Product

return_product_info	List	Void	Get registered products list as a parameter and show
	Product list	-	

			in the product information page.
return_product_detail_info	List	Void	Get product information list as a parameter and show in the register product page.
	product_id	-	

Live broadcast

return_live_info	List	Void	Get live information list as a parameter and show in the live detail page.
	Live information	-	

Order

return_info	List	Void	Get order list as a parameter and show in the order detail page.
	Order information	-	

Chat

return	List	Void	Get chat message list as a parameter and show in the live screen.
	Chat message list	-	

XI. Detailed Design

Class Name	Customer		
Function	Input	Output	Process
register	email, password, name, inputYn, phone_number, profile_picture	void	Putting necessary personal data for registration
Logic	var email = readLine() var password = readLine() var name = readLine() var inputYn = readLine() var phone_number = readLine() var profile_picture = readLine() inputYn is passed to search_address() method and get return address. var address = search_address(inputYn) var resultcode = authenticate_phonenum(phone_number) SELECT customer_id FROM CUSTOMERS; INSERT INTO CUSTOMERS(customer_id, customer_name, customer_email, customer_pw, customer_address, customer_phonenumber, profileimg_url) VALUES (#{customer_id},#{name},#{email},#{password},#{address},#{phone_number},#{profile_picture});		

Class Name	Customer		
Function	Input	Output	Process
search_address	inputYn	address	Searching address
Logic	inputYn is sent to the Road Name Address Open API in server. And client side receives roadFullAddr, jibunAddr, zipNo, addrDetail, totalCount, currentPage, countPerPage, errorCode, errorMessage. Var address = {roadFullAddr, jibunAddr, zipNo, addrDetail, totalCount, currentPage, countPerPage, errorCode, errorMessage}		

Class Name	Customer		
Function	Input	Output	Process

authenticate_phonenum	Phone_number	resultcode	Authenticating identity passing phone number information
Logic	Phone_number is sent to the NICE API in server. In server-side, it extracts requestno, returnurl, sitecode, authtype, mobilceco, businessno, methodtype, popupyn, receivedata. If the verification performed correctly, return resultcode else, return errorcode		

Class Name	Customer		
Function	Input	Output	Process
login	email, password	void	Login after registration as a user
Logic	User puts the email and password as a parameter for login. If email and password in the customer database, (SELECT customer_email, customer_pw FROM CUSTOMERS;) user state will be changed to login state and will move to the main page. SELECT customer_email, customer_pw FROM CUSTOMERS; If customer_email == email && customer_pw == password: val intent = Intent(this, Main::class.java) startActivity(intent)		

Class Name	Customer		
Function	Input	Output	Process
change_editText	Changing chat message, name, live_id	Chat message	Typing chat during live broadcast
Logic	Get the value for chat message from the customer. var chat_message = readLine() And store the putted data from each customer in Chats database. SELECT chat_id FROM CHATS; INSERT INTO CHATS(chat_id, live_id, customer_name, chat_content) VALUES (#{chat_id},#{live_id},#{name},#{chat_message});		

Class Name	Customer		
Function	Input	Output	Process

access_live	live_id	void	Joining ongoing live
Logic	SELECT liveserver_url FROM LIVES WHERE live_id = ${live_id};, the customer is connected to the Live server URL, and he/she can watch the Live streaming. And updated participant number. UPDATE LIVES SET participant_number += 1 WHERE live_id = #{liveId};		

Class Name	Customer		
Function	Input	Output	Process
leave_live	live_id	void	Leaving from ongoing live
Logic	If 'Exit' button is clicked, customer exit this live. Binding.exitBtn.setOnClickListener{ UPDATE LIVES SET participant_number -= 1 WHERE live_id = #{live_id}; finish() }		

Class Name	Customer		
Function	Input	Output	Process
send_chat	chat_id	void	Sending chat after finishing writing chat(editText)

Logic	Chat is passed to Live Stream API, and then it is shown in the live broadcast page. If 'SEND' button clicked, then chat_id is sent to Live Stream API in server.
	
	Binding.sendBtn.setOnClickListener{
	LiveCatMessages.insert(snippet.liveChatId,
	snippet.type=textMessageEvent, snippet.textMessageDetails.messageText)
	if successful return liveChatMessage
	else return 0
	}		

Class Name	Customer		
Function	Input	Output	Process
view_product_detail	product_id	void	Viewing product detail during live
Logic	When the user selects one product, it shows the product detail. The registered product's information is shown to each customer with the below query. SELECT * FROM PRODUCTS WHERE product_id = #{product_id};		

Class Name	Customer		
Function	Input	Output	Process
choose_product_options	product_id, product_quantity, inputYn, phone_number	void	Choosing product options in product detail page
Logic	Get the value for quantity, name, inputYn, phone_number from the customer. var quantity = readLine() var name = readLine() var inputYn = readLine() var phone_number = readLine() var address = search_address(inputYn) And update product option with below query. SELECT productoptions_id FROM PRODUCTOPTIONS; INSERT INTO PRODUCTOPTIONS(product_id, kind_option, kg_option, price) VALUES (#{product_id},#{kind_option}, #{product_quantity}, #{price}) And store the putted data from each customer in ORDERS database. SELECT order_id FROM ORDERS; INSERT INTO ORDERS(order_id, product_id, live_id, customer_id, farmer_id, order_date, deliver_address, deliver_phonenumber) VALUES (#{order_id},#product_id},#{live_id},#{customer_id},#{farmer_id},#{order_date},#{address});		

Class Name	Customer		
Function	Input	Output	Process
payment	order_id	Boolean	Payment after choosing options
Logic	When the customer clicks the payment button on the order page, price information is passed on to the Payment API with below queries. SELECT product_id FROM ORDERS WHERE order_id = #{order_id}; SELECT price FROM PRODUCTOPTIONS WHERE product_id = #{product_id}; In the client-side, if the request with payment information passed correctly, return rsp(response object). In server-side, API extracts imp_uid(payment), merchant_uid(order) from req parameter. The server verifies payment		

	information and returns status. If status == 'success', return true else, return false

Class Name	Farmer		
Function	Input	Output	Process
register	email, password, name, inputYn, phone_number, profile_picture	void	Putting necessary data for registration
Logic	Get the value for email, password, name, inputYn, phone_number, profile_picture from the user. var email = readLine() var password = readLine() var name = readLine() var inputYn = readLine() var phone_number = readLine() var profile_picture = readLine() inputYn is passed to search_address() method and get return address. Var address = search_address(inputYn) Call Authenticate_phonenum(phone_number) and get resultcode in the form of JSON. And store the putted data from each farmer in FARMERS database. SELECT farme_id FROM FARMERS; INSERT INTO FARMERS(farmer_id, farmer_name, farmer_email, farmer_pw, farmer_address, farmer_phonenumber, profileimg_url) VALUES (#{farmer_id},#{name},#{email},#{password},#{address},#{phone_number},#{profile_picture});		

Class Name	Farmer		
Function	Input	Output	Process
search_address	inputYn	address	Searching address
Logic	inputYn is sent to the Road Name Address Open API in server. And client side receives roadFullAddr, jibunAddr, zipNo, addrDetail, totalCount, currentPage, countPerPage, errorCode, errorMessage. And return address Var address = {roadFullAddr, jibunAddr, zipNo, addrDetail, totalCount, currentPage, countPerPage, errorCode, errorMessage}		

Class Name	Farmer		
Function	Input	Output	Process
authenticate_phonenum	phone_number	resultcode	Authenticating identity passing phone number information
Logic	Phone_number is sent to the NICE API in server. In server-side, it extracts requestno, returnurl, sitecode, authtype, mobilceco, businessno, methodtype, popupyn, receivedata. If the verification performed correctly, return resultcode else, return errorcode		

Class Name	Farmer		
Function	Input	Output	Process
login	email, pssword	void	Login after registration as an user
Logic	User puts the email and password as a parameter for login. SELECT farmer_email, farmermer_pw FROM FARMERS; if farmer_email == email && farmer_pw == password: val intent = Intent(this, Main::class.java) startActivity(intent)		

Class Name	Farmer		
Function	Input	Output	Process
send_authenticate_info	business_info	void	Authenticating business information
Logic	Get the value for business_info from the user. var business_info = readLine() And store the putted image url from each farmer in FARMERS database. SELECT farme_id FROM FARMERS; INSERT INTO FARMERS(farmer_id, business_info) VALUES (#{farmer_id},#{business_info});		

Class Name	Farmer		
Function	Input	Output	Process
view_customer_orders	farmer_id	Order list	Viewing a list of customers' orders
Logic	When the farmer clicks 'order' button, all order list is shown to each farmer with the below query. SELECT order_id FROM FARMERS WHERE farmer_id = #{farmer_id};		

Class Name	Farmer		
Function	Input	Output	Process
choose_order	order_id	selected Order's information	Choosing one of the orders from the list of customers' orders
Logic	When the farmer clicks a specific order in the order list, that information shown to each farmer with the below query. SELECT * FROM ORDERS GROUP BY #{order_id};		

Class Name	Farmer		
Function	Input	Output	Process
Change_registered_product	product_id	void	Changing information of products that are registered already
Logic	Registered product information is revised. And the corresponding data is updated into PRODUCTS table using UPDATE query below. SELECT * FROM PRODUCTS GROUP BY #{product_id}; UPDATE PRODUCTS SET fruit_id = #{fruit_id}; UPDATE PRODUCTS SET main_price = #{main_price}; UPDATE PRODUCTS SET detailimg_url = #{detailimg_url}; UPDATE PRODUCTS SET sweet_tag = #{sweet_tag}; UPDATE PRODUCTS SET sour_tag = #{sour_tag}; UPDATE PRODUCTS SET reap_tag = #{reap_tag};		

Class Name	Farmer		
Function	Input	Output	Process
change_customer_info	customer_id	void	Changing customers' information
Logic	Registered product information is revised. And the corresponding data is updated into PRODUCTS table using UPDATE query below. SELECT * FROM PRODUCTS GROUP BY #{product_id}; UPDATE PRODUCTS SET fruit_id = #{fruit_id}; UPDATE PRODUCTS SET main_price = #{main_price}; UPDATE PRODUCTS SET detailimg_url = #{detailimg_url}; UPDATE PRODUCTS SET sweet_tag = #{sweet_tag}; UPDATE PRODUCTS SET sour_tag = #{sour_tag}; UPDATE PRODUCTS SET reap_tag = #{reap_tag};		

Class Name	Farmer		
Function	Input	Output	Process
cancel_order	order_id	void	Canceling customers' orders
Logic	Specific order which selected by farmer is deleted in all tables which have it. Corresponding data is dropped from the ORDERS table with the below query. binding.cancelBtn.setOnClickListener{ DELETE FROM ORDERS WHERE order_id = #{order_id};}		

Class Name	Farmer		
Function	Input	Output	Process
confirm_order	order_id	void	Confirmation of customers' orders after customers place orders
Logic	Specific order which selected by farmer can confirm. Corresponding data is shown in the order page with the below query. binding.confirmBtn.setOnClickListenenr{ SELECT * FROM ORDERS GROUPBY #{order_id};}		

Class Name	Farmer		
Function	Input	Output	Process
check_chat	live_id	Chat message list	Updating customers' chats in real time
Logic	Show chat messages which included in that live ID on that live page with the below query. SELECT customer_name, chat_content FROM CHATS WHERE live_id = ${live_id}		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
logout	void	void	Logout
Logic	When the user clicks the Logout button, the user state will be changed to logout state and will move to the login page. Binding.logoutBtn.setOnClickListener(signOut() val intent = Intent(SininActivity) StartActivity(intent))		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
remove_account	farmer_id	void	Removing all the personal data
Logic	binding.withdrawBtn.setOnClickListener{ DELETE FROM FARMERS WHERE farmer_id = #{farmer_id}; intent = Intent(SigninActivity) startActivity(Intent) }		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
edit_profile	farmer_id	void	Changing personal information data
Logic	binding.modifyBtn.setOnClickListener{ var password = pwEditText.text var name = nameEditText.text var address = addressText.text var phone_number = phonenum.text var propicurl = url UPDATE FARMERS SET farmer_name = name, farmer_pw = password, farmer_address = address, farmer_phonenumber = phone_number, profileimg_url = propicurl WHERE farmer_id = #{farmer_id};		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
view_profile	farmer_id	void	Viewing personal profile

Logic	bindingprofileView.setOnclickListener{
	changeFragment(“profile“)
	}
	Fun changeFragment(string frag_Name){
	SELECT farmer_name, farmer_email, farmer_address, farmer_phonenumber, fruit_id1, fruit_id2, fruit_id3, profileImg_url, lives_id, products_id, orders_id FROM FARMERS WHERE farmer_id = #{farmer_id};
	var name = farmer_name;
	var email = farmer_email;
	var address = farmer_address;
	var phone_number = farmer_phone_number
	var string[] fruitId = [fruit_id1, fruit_id2, fruit_id3]
	Glide.profileImg.set(profileImg_url)
	}		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
register_live	Live infromation	void	Registering

Logic	binding.liveRegistesrBtn.setOnclickListener{
	var fruit_id = fruit.text
	var product_id =product.text
	var title = editTitle.text
	var thumbnail_url = Storage.profileimg.url
	var open_date = open_date.date
	curl -d {title:title, start_date:open_date}
	POST https://www.googleapis.com/youtube/v3/liveStreams
	var liveserver_url = response.getValue(live_url)
	var chat_id = response.getValue(chat_key)
	INSERT INTO LIVES farmer_id, product_id, fruit_id, chat_id, title, description, thumbnail_url open_date, liveserver_url VALUES #{ farmer_id }, #{ product_id }, #{ fruit_id }, #{ chat_id }, #{ title }, #{ thumbnail_url }#{ open_date }, #{ liverserver_url }
	}		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
get_live_info	farmer_id,	live list	Viewing information of specific live
Logic	SELECT title, open_date, fruit_id, thumbnail_url FROM LIVES GROUP BY #{farmer_id}; binding.titleText = title binding.open_date.text = open_date.toString() binding.thumbnail = Glide.set(thumbnail_url)		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
change_live_info	live_id	void	Changing information of live that is already registered
Logic	var title = titleEditText.text var description = desciptionEditText.text var open_date = open_date		

	var thumbnail_url = Storage.getURL() request.setValue(title, open_date) PUT https://www.googleapis.com/youtube/v3/liveStreams UPDATE FARMERS SET title = title, description = description, open_date = open_date, thumbnail_url = thumbnail_url WHERE farmer_id = #{farmer_id};

Class Name	Mypage(farmer)		
Function	Input	Output	Process
get_product_info	farmer_id	Product list	Viewing information of product

Logic	binding.produtView.setOnClickListener{
	SELECT title, main_price, fruit_id, sweet_tag, sour_tag, reap_tag FROM PRODUCTS GROUP BY #{farmer_id};
	var title = title;
	var main_price = main_price;
	var fruit_id = fruit_id;
	var sweet_tag = sweet_tag;
	var sour_tag = sour_tag;
	var reap_tag = reap_tag;
	}		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
register_product	Product information	void	Registering products to sell during live

Logic	INSERT INTO PRODUCTS fruit_id, title, main_price, contents, sweet_tag, sour_tag, reap_tag VALUES #{ fruitID.text }, #{ titleText.text }, #{ main_price }, #{ thumbnail_url }, #{ contents }, #{ sweet_tagText.text }, #{ sour_tagText.text }, #{ reap_tagText.text }
	Options.forEach{
	INSERT INTO PRODUCTOPTIONS product_id, kind_options, kg_options, price VALUES #{ product_id }, #{ kind_optionsText.text }, #{ kg_optionsText.text }, #{ price }
	}
	}		

Class Name	Mypage(farmer)		
Function	Input	Output	Process

view_product_detail	product_id	Product information	Viewing information of specific product in detail

Logic	SELECT title, main_price, thumbnail_url FROM PRODUCTS GROUP BY #{farmer_id}
	SELECT kind_options, kg_options, price FROM PRODUCTOPTIONS GROUP BY #{ product_id }
	var title = title;
	var main_price = main_price
	var thumbnailImg = Glide.set(thumbnail_url)
	var kind_options = arrayOf(kind_options)
	var kg_options = arrayOf(kg_options)
	}		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
view_live_detail	live_id	Live information	Viewing information of specific live in detail

Logic	binding.liveDetailView.setOnclickListener{
	SELECT * FROM LIVES GROUP BY #{farmer_id}
	binding.fruit_idText.text = fruit_id
	binding.titleText.text = title
	binding.descriptionText.text = description
	binding.opendateText.text = open_date.toString()
	Glide.set(thumbnail_url)
	}		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
start_live	Void	Void	Starting live that is already registered

Logic	binding.liveStartBtn.setOnclickListener{
	GET https://www.googleapis.com/youtube/v3/liveStreams
	}		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
stop_live	live_id	void	Stopping ongoing live
Logic	binding.stopBtn.setOnclickListener{		

	POST https://www.googleapis.com/youtube/v3/liveStreams }

Class Name	Mypage(farmer)		
Function	Input	Output	Process
extract_csv	product_id	void	Extracting csv for delivery request
Logic	SELECT * FROM ORDERS GROUP BY product_id Pd.to_csv(product_id, order_date, deliver_address, deliver_phonenumber)		

Class Name	Mypage(farmer)		
Function	Input	Output	Process
request_delivery_status	trak_id	Delivery information	Requesting status for a delivery of the user ordered
Logic	SELECT track_id FROM ORDERS WHERE order_id = #{order_id}; POST http://trace-api-dev.sweettracker.net:8012 rquest.carrier_id, track_id)		

Class Name	Mypage(customer)		
Function	Input	Output	Process
logout	void	void	Logout
Logic	When the user clicks the Logout button, the user state will be changed to logout state and will move to the login page. Binding.logoutBtn.setOnClickListener(signOut() val intent = Intent(SininActivity) StartActivity(intent))		

Class Name	Mypage(customer)		
Function	Input	Output	Process
remove_account	customer_id	void	Removing all the personal data
Logic	**if** customer_id is not null **then** SELECT customer_id FROM Customers; DELETE FROM CUSTOMERS WHERE customer_id_number = #{customer_id};		

Class Name	Mypage(customer)		
Function	Input	Output	Process
edit_profile	customer_id	void	Changing personal information data
Logic	SELECT * FROM CUSTOMERS WHERE customer_id_number= #{customer_id}; UPDATE CUSTOMERS SET customer_name = #{name}; UPDATE CUSTOMERS SET customer_pw = #{password}; UPDATE CUSTOMERS SET customer_address = (#{roadFullAddr},#{jibunAddr},#{zipNo},#{addrDetail}); UPDATE CUSTOMERS SET customer_phonenumber= #{phonenumber}; UPDATE CUSTOMERS SET profileimg_url = #{profileimage};		

Class Name	Mypage(customer)		
Function	Input	Output	Process
view_profile	customer_id	void	Viewing personal profile
Logic	**if** customer_id = #{customer_id} is not null **then** SELECT customer_name, customer_email, customer_address, customer_phonenumber, profileimg_url FROM CUSTOMERS WHERE customer_id = #{customer_id};		

Class Name	Mypage(customer)		
Function	Input	Output	Process
choose_order	order_id	Order information	Choosing one of the orders from order list
Logic	**if** order_id is not null **then** SELECT order_id, order_date, deliver_date, deliver_address, deliver_phonenumber, track_id FROM Orders WHERE customer_id = #{customer_id}; **return** order_id, order_date, deliver_date, deliver_address, deliver_phonenumber, track_id **else** error message will come up		

Class Name	Mypage(customer)		
Function	Input	Output	Process
request_delivery_status	trak_id	Delivery information	Requesting status for a delivery of the user ordered
Logic	**if** track_id is valid **then** SELECT track_id FROM ORDERS WHERE order_id = #{order_id};		

	return carrier_id, track_id **else** API will return error

Class Name	Mypage(customer)		
Function	Input	Output	Process
cancel_order	order_id	void	Canceling orders
Logic	SELECT order_id FROM Orders; **if** order_id is not null **then** DELETE FROM ORDERS WHERE order_id = #{order_id};		

Class Name	Mypage(customer)		
Function	Input	Output	Process
change_order	order_id	void	Change options of customer orders
Logic	SELECT order_id FROM Orders; **if** order_id is not null **then** UPDATE PRODUCTOPTIONS SET kind_option = #{kind_option}; UPDATE PRODUCTOPTIONS SET kg_option = #{kg_option};		

Class Name	Admin		
Function	Input	Output	Process
send_result	farmer_id	Result message	Admin checking farmer authentication information

Logic	SELECT farmer_id FROM FARMERS;
	
	if farmer_id is valid after checking physically **then**
	return Successful message
	else
	return Failure message		

Class Name	Product		
Function	Input	Output	Process
return_product_info	Product list	void	Seeing all the products that are registered
Logic	**if** product_id is not null **then** SELECT title, fruit_id, main_price contents FROM Products;		

Class Name	Product		
Function	Input	Output	Process
return_product_detail_info	product_id	void	Seeing detailed information of specific product.
Logic	SELECT product_id FROM Product; **if** product_id is not null **then** SELECT main_price, contents, sweet_tag, sour_tag, reap_tag FROM PRODUCTS WHERE product_id = #{product_id}; SELECT kind_option, kg_option FROM ProductOptions WHERE product_id = #{product_id};		

Class Name	Live broadcast		
Function	Input	Output	Process
return_live_info	Live information	void	Showing live detail about live information
Logic	SELECT live_id FROM Lives; **if** live_id is not null **then** SELECT title, description, open_date FROM LIVES;		

Class Name	Order		
Function	Input	Output	Process
return_info	Order information	void	Seeing all the orders
Logic	**if** Order Information is valid **then** SELECT order_id FROM Orders; **return** void		

Class Name	Chat		
Function	Input	Output	Process
return	Chat message list	void	Showing chats during live

Logic	SELECT chat_id FROM Chats;
	Repeat
	if chat_id is not null **then**
	SELECT customer_name, chat_content FROM Chats;
	ends for live_id = null		

XII. Appendix

- Product code of each fruit
- The name of the fruits are written in Korean because there are varieties that cannot be written in English.

1st class	2nd class	Code
사과(A)	홍로(01)	A01
	햇(02)	A02
	부사(03)	A03
	아오리(04)	A04
배(B)	장십량(01)	B01
	신고(02)	B02
	만삼길(03)	B03
귤(C)	금귤(01)	C01
	감귤(02)	C02
	깔라만시(03)	C03
만감(D)	레드향(01)	D01
	천혜향(02)	D02
	한라봉(03)	D03
	황금향(04)	D04
오렌지(E)		E
수박(F)	꿀(01)	F01
	애플(02)	F02
	망고(03)	F03
멜론(G)	파파야(01)	G01
	네트(02)	G02
	무네트(03)	G03
	머스크(04)	G04
	양구(05)	G05
	칸탈로프(06)	G06
참외(H)		H
토마토(I)	대추방울(01)	I01

	스테비아(02)	I02
	방울(03)	I03
	대과(04)	I04
	대저(05)	I05
딸기(J)	설향(01)	J01
	매향(02)	J02
	킹스베리(03)	J03
	만년설(04)	J04
키위(K)	그린(01)	K01
	골드(02)	K02
	레드(03)	K03
블루베리(L)		L
포도(M)	청(01)	M01
	캠벨(02)	M02
	블랙사파이어(03)	M03
	샤인머스켓(04)	M04
	거봉(05)	M05
	세단(06)	M06
자두(N)		N
복숭아(O)	천도(01)	O01
	백도(02)	O02
	황도(03)	O03
감(P)	단감(01)	P01
	대봉(02)	P02
바나나(Q)		Q
파인애플(R)		R
자몽(S)		S
레몬(T)		T
망고(U)	애플(01)	U01
	그린(02)	U02
	Carabao(03)	U03
	Nam Dok Mai(04)	U04
체리(V)		V
석류(W)		W
매실(Y)	청(01)	Y01

	홍(02)	Y02
	황(03)	Y03
모과(Z)		Z
유자(AA)		AA
용과(AB)	백색계(01)	AB01
	적색계(02)	AB02
	분홍색계(03)	AB03
	황색계(04)	AB04

XIII. Reference

- 못난이 농산물 구매 실태 및 인식 분석, 한국소비자원, 2021