IGC Market
Online Platform
Software Design Document

Team Members:

Project Manager

Jeongmin Yoo

Lead Programmer

Jey Kang

Project Owner

Yeeun Sohn

Designer

Jaewon Lee

Table of Content

1. Introduction

1.1 Product Description

The IGC Market is a web-based used goods market for students at an IGC-based university. IGC Market's goal is to make a community where IGC students can sell and buy their goods and reduce throwing out of items of IGC students which causes additional environmental damage.

There is no community where IGC students can sell and buy their goods now, so SUNY students send messages about buying and selling textbooks and home appliances in SUNY group chat whenever a new semester begins. This can cause other students to miss important information, and some students leave the group chat because the chat room is noisy. Also, though there are IGC students nearby, only SUNY students participate in the group chat, so many potential buyers and sellers or items are unable to connect with each other.

Hence, our solution is to create a platform, replacing SUNY group chat, that IGC students can sell and buy their items from each other and reduce the losses of IGC's potential sales and the throwing out of items causing additional environmental damage. Several main sections will be included on the website:

1. Listing Creation by Users - Users (students with accounts) must be able to create a product listing by providing images of their item, its price, its category (from a fixed list of categories), its status (on sale, sold, reserved) and a description of its specifics, as well as a title for the entire listing. This listing is viewable by other users.
2. Search - Users will be able to type in a keyword to a search box at the top. Then the application will list all the products related to the keyword. Users can also filter out the categories that they are interested in.
3. Product Description - Users will be able to view the details of the product by clicking on the image or text of the product. The description will contain the following information.
    a. images of the product posted by the product owner
    b. information given by the product owner
    c. time of the product posted, number of views
    d. price of the product
    e. status of product (e.g., on sale, sold, reserved)
    f. contact information of the product owner (e.g., SUNY email address)
4. Inter-user Chatting - Users that are interested in buying a product will be able to send a message to the product owner by chatting application installed. Users choose how to deliver the product (e.g., meeting in person), share information needed to make the delivery (e.g., meeting time and location), and share a review of it.
5. Set Favorite - Users who are buying a product will set a favorite for a product that they are interested in. Users can see the list of products that they set as favorite.
6. About us - The application will include a page where users can read about the scope and purpose of the application and the contact information of the managers of the application. Contact information will include name, email addresses.

1.2 Scope

"IGC-Market" is a web application for trading used items, localized within the IGC campus area. It will run on any device that supports a web browser. The application will allow users to find cheap used items for sale by other users and provides multiple methods of filtering items to match user preferences. Transactions are carried out entirely offline, after negotiations between users through the integrated chat feature.

Users may also add their own items to the pool of searchable listings provided detailed information about the items, such as item category, item price, and item description. Other users may use these details to filter the listing pool via the search bar, which displays a page with links to the detail pages for listings that contain content that matches entered keywords.

Every listing will have associated detail pages, which display a wide range of information about the listing, including the information inserted by the user who owns the product being sold as well as other minutiae such as owner email, view count, date created, comments and ratings (on a scale from 0 to 5).

"IGC-Market" will utilize users' email addresses to serve as unique identifiers visible to all registered users. Email addresses are collected during account registration and are restricted to addresses associated with the schools currently located within the IGC multicampus complex.

1.3 Users

The demographics of the intended audience for this app are people who either reside in or make frequent visits to the Incheon Global Campus area. This includes students who attend institutions currently located within the IGC, professors who teach at said institutions, and other faculty members who have access to a faculty email account, which is required for signup. However, we expect that the majority of active users (as in users who actively make use of the application) will be students.

1.4 User feedback

We anticipate that user feedback will be most urgently required when designing the user interface in order to determine the most frequently used features and how they can be improved. This will allow for a better focus of development efforts. If possible, user feedback should be obtained periodically during development to accurately gauge progress. So we would get the feedback by posting a survey in SUNY and other universities in IGC group chat.

1.5 Existing alternatives

Two of the most popular web-based used goods trading applications in Korea are Dang-geun Market and Bungae Jangter. Of the two, we were particularly interested in the former, as it requires users to communicate and negotiate a way to facilitate payment, which restricts transactions to a small area around the seller’s location. Our vision for IGC Market is a hyper-localized form of this, restricted to the relatively tiny area that is the IGC. This allows for easier and quicker meetups for trading and allows

students to search for items relevant to their on-campus life more easily. Not only that, but it also removes the buying/selling posts made in semi-official group chats, allowing for more focus on study-relevant matters.

1.6 Definitions

- **IGC:** Incheon Global Campus
- **UI:** User Interface

1.7 References

- Dang-geun Market Website: https://www.daangn.com/
- Bungae Jangter Website: https://m.bunjang.co.kr/

2. Requirements

2.1 Functional Requirements

Users can:

1. Read about the purpose of this website and the benefits when students use our website.
2. View all the categories on sale
3. View IGC goods in a variety of methods.
    a. Use a filter that sort goods by categorizing several features
    b. Sort items by using price, upload date, and the number of viewers
4. View IGC goods with a variety of information
    a. Item name
    b. Category
    c. Picture of the item good
    d. Price
    e. Product Status
    f. Features
5. View the information about the post
    a. View
    b. Upload date and time
    c. Uploader name
    d. The status of products
        i. On sale
        ii. Reserved
        iii. Sold
6. Upload an item that the user wants to sell with specific information
    a. Item name
    b. Category
    c. Picture of the item
    d. Price

    e. Product Status
    f. Features
7. Save the item as “Favorites”
8. View the “Favorites” list
9. View history that the user try to sell
10. Edit the post that the user uploaded
11. Sign up with a group(university) email address
12. Log in with the group(university) email address

Our Team administrators can:

1. Request to delete the contents displayed on the website.
2. Manage the group(university) mailing lists and send emails about any alarms regarding their items.
3. Manage the group(university) mailing lists and send emails about any promotions regarding the website.
4. Ban a user who gets a report from other users.
5. Designate administrators’ accounts and make them manage the website.

2.2 Use Cases

2.2.1 User class 1 - User and Administrator

2.2.1.1 Sign-up

Use Case:	Sign-up
Primary Actor:	Buyer, Seller, Administrator
Priority:	High
Goal in Context:	A user wants to sign up to perform activities within the application.
Precondition:	The user is viewing on a sign-up page.
Scenario:	1. The user clicks the “Sing-up” button for their university email. 2. The user enters their university email and password. 3. The server verifies the user's university email and password. 4. If it is verified well, then the user is given a sign-up token to be granted access.
Postcondition:	The user is viewing the main page with log-in.
Extensions:	3.1 If the user doesn’t have the university email or the server fails to verify the user’s university email, it will alert the user with a “Verification Failed” message.

2.2.1.2 Login

Use Case:	Login
Primary Actor:	Buyer, Seller, Administrator
Priority:	High
Goal in Context:	A user wants to log in to perform activities within the application.
Precondition:	The user is viewing on a login page.
Scenario:	1. The user clicks the “Log-in” button for their university email. 2. The user enters their university email and password. 3. The server verifies the user's university email and password. 4. If it is verified well, then the user is given a log-in token to be granted access
Postcondition:	The user is viewing the main page with log-in.
Extensions:	3.1 If the user enters the wrong email or password and the server fails to verify their university email, it will alert the user with a “Verification Failed” message.

2.2.1.3 Log-out

Use Case:	Log-out
Primary Actor:	Buyer, Seller, Administrator
Priority:	High
Goal in Context:	A user wants to log out and stop the use of the application.
Precondition:	A user is logged in.
Scenario:	1. The user clicks the “Logout” button. 2. The log-in token gets deleted.
Postcondition:	The user is viewing the main page with log-out.
Extensions:	N/A

2.2.2 User Class 2 - Buyer

2.2.2.1 Search item by category

Use Case:	Search item by category
Primary Actor:	Buyer

Priority:	High
Goal in Context:	A user wants to search available item listings by category.
Precondition:	The buyer must be logged in to their account and view the “Search” bar.
Scenario:	1. The buyer clicks on the search dropdown bar that contains “Category” options. 2. The buyer clicks the option that they want. 3. The buyer enters keywords they want to find and presses the “Search” button or the enter key on their keyboard.
Postcondition:	The buyer is shown a result page containing listings that match keywords and the selected options. Each individual listing can be clicked to show its details page.
Extensions:	2a.1 If the buyer selects the option “All”, the buyer can view a result page containing listings that match keywords in all categories. 2b.1 If the buyer does not select any option, the buyer can view a result page containing listings that match keywords in all categories. 3.1 If there is no item matching with the keywords, the message “No matching result” is shown.

2.2.2.2 Sort results by options

Use Case:	Sort results by options
Primary Actor:	Buyer
Priority:	High
Goal in Context:	A buyer wants to sort the search results (displayed listings).
Precondition:	They must have search results or be on the Main page.
Scenario:	By default, the search results are sorted in order of recent uploaded. If the buyer has already selected a different sorting method, they may do the following: 1. The buyer clicks the dropdown labeled “Sort by”. It contains options of ‘Price’, ‘Recent’, and ‘Hits’. 2. The buyer selects one of the options from the dropdown list.
Postcondition:	1. If the option is set to ‘Price’, the results are reordered to prioritize cheaper listings 2. If the option is set to ‘Recent’, the results are reordered to prioritize recent listings. 3. If the option is set to ‘Hits’, the results are reordered to prioritize more hits.

	Each listing can be clicked to show the details page for the item being sold.
Extensions:	N/A

2.2.2.3 View listing details

Use Case:	View listing details
Primary Actor:	Buyer
Priority:	High
Goal in Context:	A buyer wants to view the details page of a listing.
Precondition:	The buyer must be logged in to their account, and they must have search results displayed from a previous search or be on the Main page.
Scenario:	1. The buyer clicks on one of the listings displayed in the results.
Postcondition:	The buyer is taken to the details page for the selected listing, which contains information such as item description, price, and seller id.
Extensions:	N/A

2.2.2.4 Initiate transaction (Chat with the seller)

Use Case:	Initiate transaction
Primary Actor:	Buyer
Priority:	High
Goal in Context:	A buyer wants to initiate a transaction to purchase a listing.
Precondition:	The buyer must be viewing the details page of a listing.
Scenario:	1. The buyer clicks the button labeled “Chat with seller” on the details page. 2. The buyer chats with the seller to initiate the transaction. 3. The buyer and seller decide the place where they meet and have a transaction.
Postcondition:	After completing the transaction, the seller sets the status of the listing to “Sold”. And, when the status of a product changes to “Sold”, other buyers cannot initiate chatting for that product.
Extensions:	3.1 If the communication between buyer and seller is not going well, the transaction will not complete.

2.2.2.5 Set Favorite

Use Case:	Set Favorite
Primary Actor:	Buyer
Priority:	High
Goal in Context:	A buyer wants to set a favorite for the product interested.
Precondition:	The buyer must be viewing the details page of a listing.
Scenario:	1. The buyer clicks the button labeled “Favorite” on the details page.
Postcondition:	The buyer can see the product that they set “Favorite” in the “My Account” page.
Extensions:	N/A

2.2.3 User Class 3 - Seller

2.2.3.1 Create Product Listing

Use Case:	Create Product Listing (Uploading a product post for sale)
Primary Actor:	Seller
Priority:	High
Goal in Context:	A seller wants to create a product listing for sale.
Precondition:	The seller should be logged on to the application, and access to the seller’s album.
Scenario:	1. The seller selects the “Upload” button on the main screen. 2. The seller uploads the image of the product the seller wants to sell. 3. The seller types in the product’s name in the title and its description. 4. The seller selects the category and status of the product. 5. The seller types in the price that the seller wants to offer. 6. The seller presses the “Upload” button to finish uploading the product.
Postcondition:	The seller views the uploaded product with its detailed description.
Extensions:	N/A

2.2.3.2 Edit the uploaded product post information

Use Case:	Edit the uploaded product post information

Primary Actor:	Seller
Priority:	High
Goal in Context:	The seller can edit the information that was posted.
Precondition:	The seller should be logged in, and the seller has uploaded a product. The post to edit is selected from the user page ‘history’ section.
Scenario:	1. The seller presses the ‘edit’ button on the product post. 2. The seller views the screen that allows to modify of the information. It is identical to the initial uploading page. 3. The seller can edit the title, description, image, category, status, and price. 4. The seller presses the “Upload” button to finish editing the post.
Postcondition:	The seller views the updated product listing.
Extensions:	4.1 The seller can click a ‘cancel’ button to discard the changes.

2.2.3.3 Delete the uploaded product post

Use Case:	Delete the uploaded product post
Primary Actor:	Seller
Priority:	High
Goal in Context:	The seller can delete the post that the seller has uploaded.
Precondition:	The seller should be logged in, and the seller has uploaded a product. The post to delete is selected from the user page ‘history’ section.
Scenario:	1. The seller presses the ‘delete’ button on the product post. 2. A popup appears with a confirmation message and has a ‘confirm’ button and ‘cancel’ button. 3. The seller presses the ‘confirm’ button.
Postcondition:	The seller views the user page. The deleted post is erased from the ‘history’ section.
Extensions:	1.1 If another user has initiated a transaction to buy the product, it will alert the seller that there is a buyer wanting to make a purchase. The seller may choose the ‘delete anyways’ button to proceed with the deletion.

2.2.3.4 Delete uploaded product list in history

Use Case:	Delete uploaded product list in history

Primary Actor:	Seller
Priority:	Medium
Goal in Context:	A user wants to delete the history of uploaded product lists from the user's account.
Precondition:	The user must be logged in and should be viewing the "My Page".
Scenario:	1. The user clicks on the "My Product" button on the "My page". 2. The user views a list of products that the user has uploaded. Each element of the list shows a product's name and the "Delete" button. 3. The user can click the product's name to view the product information or click the "Delete" button to delete it from the history. 4. If the user selects the "Delete" button, the product name will be erased from "My Product".
Postcondition:	The user views "My Product" without the deleted product name.
Extensions:	N/A

2.2.4 User Class 4 - Administrator

2.2.4.1 Ban user

Use Case:	Ban user
Primary Actor:	Administrator
Priority:	Medium
Goal in Context:	An administrator may stop a user from being active.
Precondition:	The administrator should be logged in with an administrator account and be looking at its 'user page'. Also, there should be at least one user registered to the application.
Scenario:	1. The administrator clicks the 'view users' section in the 'user page'. It shows a list of users registered to the application. 2. The administrator clicks on a user's name to be banned. A brief profile of the user containing ID and a list of products the user has uploaded pops up. 3. The administrator clicks on a 'ban this user' button under the product list.
Postcondition:	The administrator views a list of users without the banned user.
Extensions:	3.1 If the administrator chooses to not ban the user, the administrator may click on the 'cancel' button next to the 'ban this user' button.

2.2.4.2 Remove listing of product

Use Case:	Remove listing of product
Primary Actor:	Administrator
Priority:	Medium
Goal in Context:	The administrator may remove an uploaded product from the database.
Precondition:	The administrator should be logged in with an administrator account and be looking at its 'user page'. Also, there should be at least one user registered to the application with a product listing created.
Scenario:	1. The administrator clicks "View product lists" on its user page. 2. The administrator views a page with a search bar and a list of all the product posts created in the application. 3. The administrator searches for a product name or type to be deleted in the search bar. When the administrator selects "Search", it will show a list of products that are relevant to the search result. 4. The administrator clicks the "Delete" button next to the product's name in the list. A confirmation alert will be shown, "OK" button will be clicked to proceed.
Postcondition:	The administrator views a list of searched product lists without the deleted product post. When the deleting process is done, the administrator may go back to the user page by clicking the "Done" button.
Extensions:	4.1 If the administrator wants to cancel the deletion, the administrator clicks the "NO" button next to the "OK" button.

2.2.4.3 Send alarm

Use Case:	Send alarm
Primary Actor:	System
Priority:	Medium
Goal in Context:	Send alert when the user receives chat
Precondition:	The user receives a chat from another user
Scenario:	1.When a user receives a chat, the system will automatically send the email to the user's university email.

Postcondition:	The user will get an alarm in their email.
Extensions:	N/A

2.3 User Interface

- Figma prototype of the platform (We updated some features): https://www.figma.com/file/91tCZws4DP3u2sZmMojST8/IGC-Market?node-id=0%3A1
- Use flow diagram: https://www.figma.com/file/ztOAzHp9VvjYiMYOgCHTeC/user-flow-diagram-template?node-id=0%3A1

2.4 Non-Functional Requirements

2.4.1 Performance requirements

All requests made by users will be processed in 500ms or lower. Here, the processing time is defined as the time required for the database to process queries incurred by a single user action. In the event that processing takes longer than 500ms, the case will be logged internally and is counted as an error.

2.4.2 Operating constraints

IGC Market requires an active internet connection for all of the services it provides, as it must constantly fetch up-to-date information from the server backend. However, other than this there are no other specific runtime constraints other than those presented by the web browser used.

2.4.3 Platform constraints

The application is designed to be viewed from any React.js-capable web browser. As such, we assume an operating system of either Windows, Mac OS, or Linux for desktops, and Android or iOS for mobile devices. The user interface will be responsive and scale to screen resolution, and as there are no device-exclusive libraries used all functions will be usable from all devices.

2.4.4 Modifiability

For all major feature changes, changes must be made to both the code for the backend as well as the frontend, which will require approximately a month of work from at least two dedicated developers on each side of the project.

2.4.5 Portability

There should not be much effort required to make the application available on other platforms (such as a mobile-specific version). It is already compatible with most, if not all, modern desktop platforms by virtue of requiring no executable component, which also allows for easier redesigning for mobile viewing.

A mobile application will most likely consist of a web-view packaged as an application rather than a native UI.

2.4.6 Reliability

The application must have at least an MTBF (mean time between failures) of one week. Here, failure is defined as an instance where either a response is not returned to the user within the time restriction (500ms), or the request fails to complete due to any non-user cause and an error message is shown. To clarify, this includes cases where there is an error updating the database and other problems server-side. However, this does not include cases where the user deliberately induces a commonly reproducible error such as reaching a 404 page via an inappropriate URL, nor does it include cases where a minor error is shown due to user incompetence, like a mistyped form. Cases where such a page is displayed from an incorrect hyperlink within the page itself still counts as an error. Timeout errors can only be mitigated through rigorous optimization while avoiding server-side errors requires rigorous regression testing for every feature addition.

2.4.7 Legal

IGC Market must expunge all personal information from accounts that have not been logged into for 2 years, in order to comply with both Korean limits on data retention and the GDPR's standard for non-anonymized personal data.

3. System Architecture

3.1 Overview

We will be using the following technologies in the development of our application:

1. React.js
   React.js is a front-end JavaScript framework for building responsive web UI. It will be used to build our entire front-end.
2. Express.js
   Express.js is a framework for Node.js intended for API development. Due to its easy integration with the other technologies on this list, we chose it for our own API.
3. Node.js
   Node.js is a JavaScript runtime designed for server development. This is an essential part of our stack because other technologies in it depend on it or are libraries built for it.
4. MongoDB
   MongoDB is a NoSQL, non-relational, document-based database framework. We wanted to move away from SQL for our project, and we chose MongoDB partly due to its role in the MERN stack and partly because of its comparatively familiar syntax.
5. Mongoose
   Mongoose is a library for communication between Node.js and MongoDB databases. This is a required library for use of Express.js.
6. NPM (Node Package Manager)

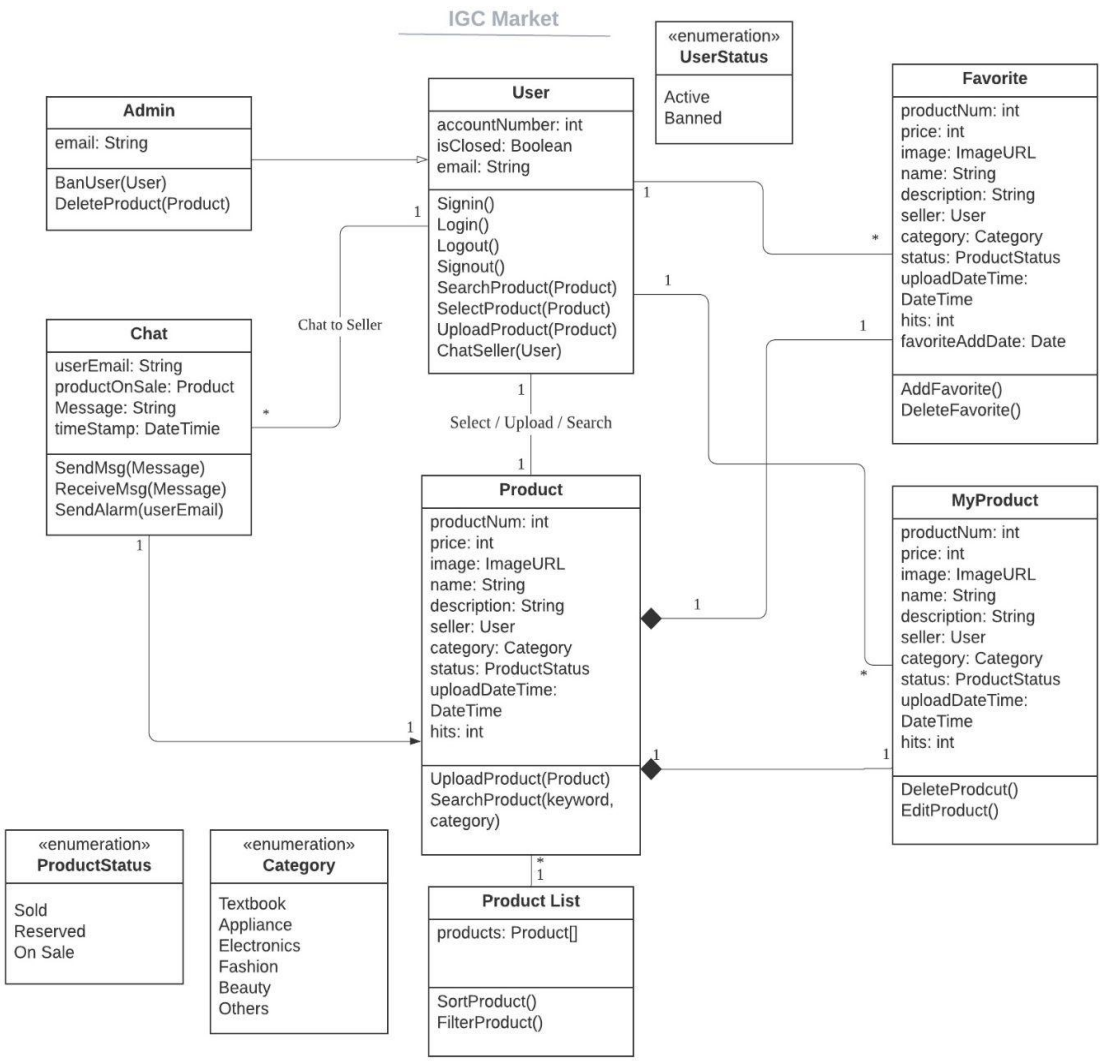
NPM is a tool for installing Node.js-related libraries and managing Node-based projects. While it is a prerequisite of being able to use Node.js, it is included on this list of technologies due to its other useful webserver management features.

7. Robo 3T
   Robo 3T is a free and open-source GUI for MongoDB. We chose this as we thought we would need a GUI for quick and easy edits to our database, and GUIs take less time to learn than command-line interfaces.
8. Postman
   Postman is a collaborative API design and development platform. We divided our API into four distinct feature sets, and so we included a collaboration platform.
9. Axios
   Axios is an API query library for React. This will be used to connect the Express.js portions and the React.js portions of our application.

We are intending to make use of the MERN stack (MongoDB-Express.js-React.js-Node.js), which is a web development stack that is known for its versatility in quickly deploying full-stack web applications. This is ideal for our project, due to the short durations between milestones.

3.2 UML Class Diagrams

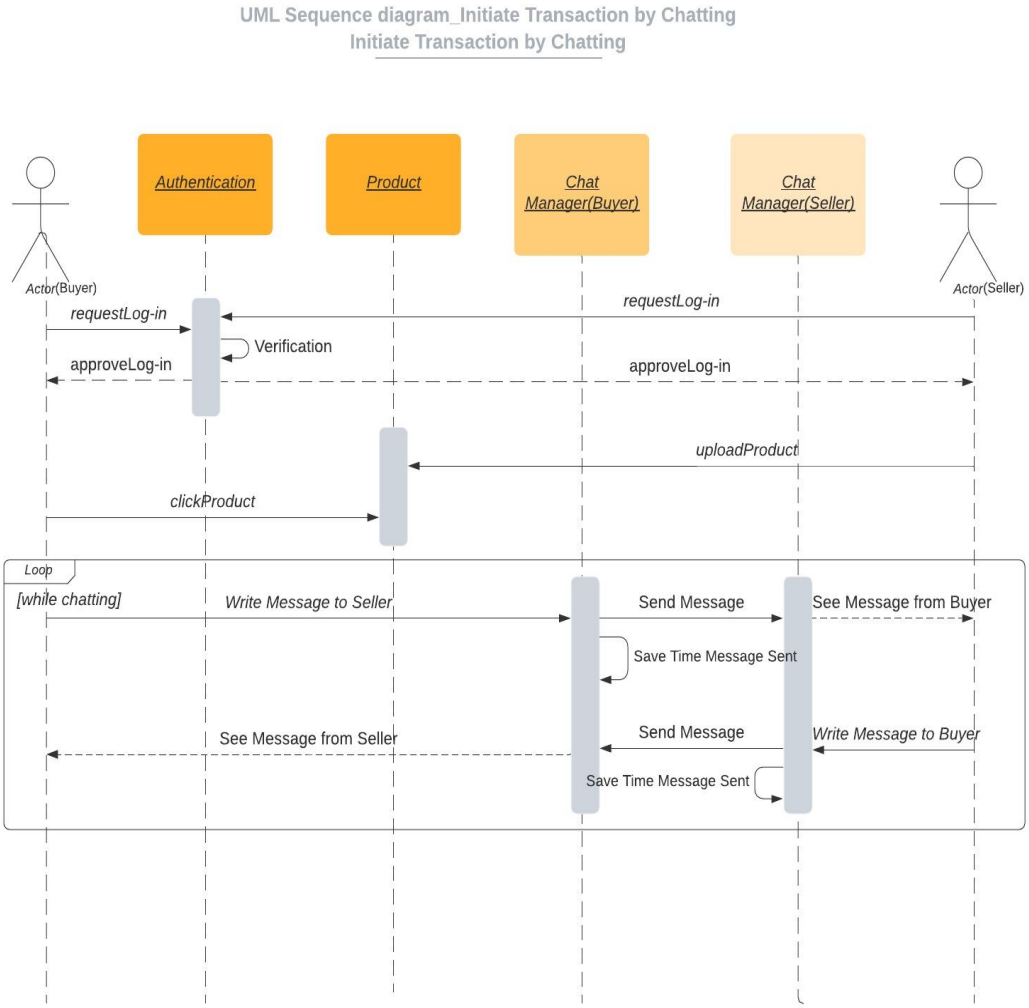
Link for UML Class Diagrams:
https://lucid.app/lucidchart/38e4b4d6-8830-43f4-bac1-9739505c9711/view?page=HWEp-vi-RSFO#



3.3 UML Sequence Diagrams

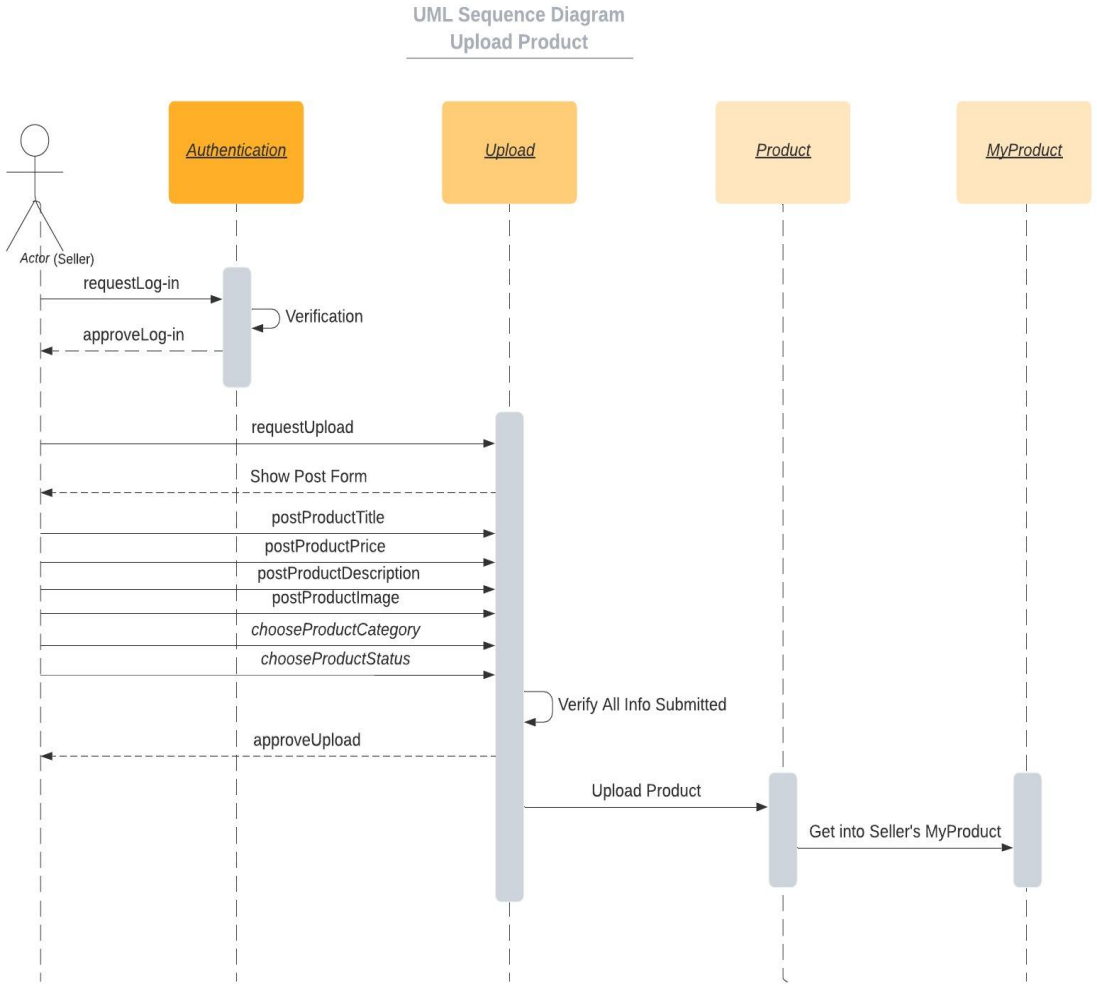
1. UML Sequence Diagram - Initiate Transaction by Chatting

   https://lucid.app/lucidchart/5c3a3a96-ab85-4ff3-97b5-71598a5bc4c8/view?page=0_0#



2. UML Sequence Diagram - Upload Product

   https://lucid.app/lucidchart/5454edf5-a96f-4e2b-8e98-ed282ef2244e/view?page=0_0#



3.4 API Design

API Design Link:
https://docs.google.com/spreadsheets/d/1gOJKWvecOOZ1FGYaBdlxIJvTfz_yvPMtkwk6DfsHSc8/edit?usp=sharing

3.5 Deployment

We will be using Heroku for deployment of our app, due to both the easy integration between Node.js and the Heroku CLI and the ready availability of information regarding the setup process.

3.6 Code Conventions

We will be using the Google JavaScript Style Guide as a baseline, with JSLint as our style checker. Modifications to our base style guide may be made if incompatibilities with the libraries we use to become an issue.

4. Schedule

Link for schedule: https://share.clickup.com/g/h/3p7r2-40/8438411cc468995

5. Contribution

Jeongmin Yoo - 4. Schedule

Jey Kang - 3.1 Overview, 3.5 Deployment, 3.6 Code Conventions, help Jeongmin for 4. Schedule, and edit 2.4 Non-functional Requirements.

Yeeun Sohn - 3.2 UML Class Diagram, 3.3 UML Sequence Diagram, and edit and organized UI Mockup and Software Design document.

Jaewon Lee - 3.4 API Design