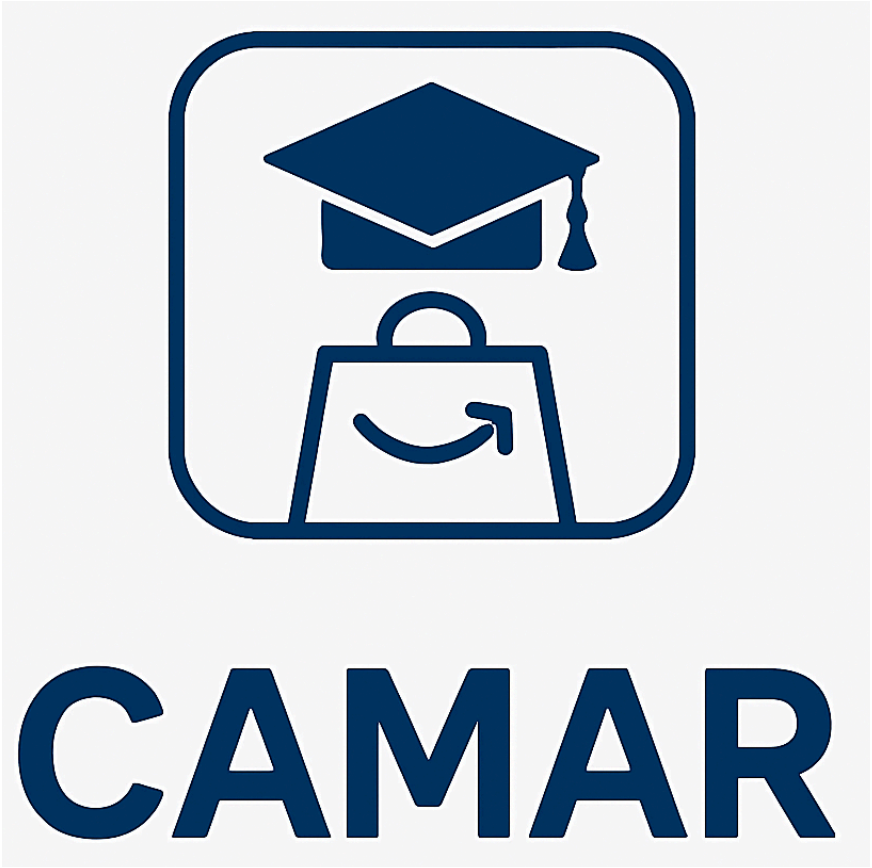


CAMMAR (Campus Market)

10.21.2025

Kitae Kim, Jaeheon Park, Sanghoon Lee

Software Design Document

CSE 416 - Team 2

Table of Contents

1. Introduction

1.1 Product Description

The proposed product is an **AI-powered secondhand marketplace web application** tailored specifically for SUNY Korea students. Unlike existing platforms such as Carrot or Bungaejangter, this marketplace directly addresses the unique needs of students by providing a **trustworthy, localized, and student-focused trading environment**.

The primary problem it addresses is the **lack of trust and relevance** of existing secondhand platforms. Public platforms allow anyone to sign up, creating risks of scams, unreliable sellers, and irrelevant listings. Our solution ensures that only verified students with university email addresses can participate, fostering a safe and student-centered community.

The **target audience** consists of SUNY Korea students, including undergraduates, exchange students, and seniors. These groups frequently need items such as textbooks, dorm supplies, and lab equipment, and they often seek affordable, fast, and convenient exchanges within the campus area.

Key product features include:

1. **AI-Powered Smart Recommendations** – Personalized suggestions based on browsing and transaction history.
2. **Automatic Item Categorization** – AI-driven image recognition to classify uploaded items (e.g., textbooks, electronics, furniture).
3. **Community Page & Interaction** – Allows users to post, comment, and connect beyond item trading, strengthening student ties.
4. **Localized Trading Environment** – Restricts transactions to campus and nearby areas, making meetups fast and practical.

Together, these features ensure a secure, efficient, and socially enriching trading experience for students.

1.2 Scope

The marketplace will:

- **Provide** a secure, student-only trading space.
- **Enable** item posting with images, descriptions, and prices.
- **Support** AI-powered recommendations and categorization.
- **Foster** community-building through posts and interactions.
- **Facilitate** chat-based transactions between buyers and sellers.

The product will **serve SUNY Korea students** and, in the future, may be generalized to other campuses by adapting university-domain-based verification.

It will **not support**:

- Nationwide or public trading beyond the campus.
- Direct financial transactions (payments remain user-handled).
- Anonymous user activity (all tied to verified university accounts).

Platforms & Devices: Runs as a responsive web application supporting both desktop and mobile browsers.

1.3 Users

The product is designed for three primary user groups:

1. **Undergraduate and Exchange Students** – Often need temporary or affordable items such as dorm supplies and textbooks. They typically have moderate technical expertise and are accustomed to online platforms.
2. **Seniors and Graduate Students** – May be selling used lab equipment, notes, or dorm furniture. They seek convenient ways to dispose of items while helping juniors.
3. **Admins** – Manage the system, handle disputes, and ensure the integrity of the marketplace. Admins are technically proficient.

Accessibility considerations:

- **Educational level**: All users are university-level students.

- **Experience**: Varies; some users may be new to secondhand trading.
- **Technical expertise**: Most users are familiar with mobile/web platforms.
- **Potential disabilities**: Platform should support basic accessibility features such as text clarity and mobile-friendly design.

1.4 User Feedback

Feedback will be collected at several stages:

- **Prototype testing**: During UI mockups and functional requirement reviews, feedback from peers and classmates will be used to refine navigation and usability.
- **Beta launch**: Students will test posting, searching, and AI recommendations; feedback will guide adjustments.
- **Community engagement**: Community posts and chats provide real-time user feedback on what works and what needs improvement.

Initial feedback from peers highlighted the importance of **restricting sign-ups to campus email addresses** to ensure trust, which shaped the authentication design.

1.5 Existing Alternatives

Popular platforms like **Carrot Market** and **Bungaejangter** dominate the Korean secondhand market, but they have drawbacks for students:

- **Strengths**: Large user bases, wide item availability, and established trust systems.
- **Weaknesses**: Open access (anyone can join), higher scam risks, irrelevant items for students, and inconvenient off-campus meetups.

Our marketplace borrows **popular features** (e.g., item posting with images, chat with sellers) while improving them with **AI categorization** and **student-only access**. Unlike competitors, it emphasizes **campus localization** and **student-specific categories**.

1.6 Definitions

- **AI Categorization**: Automatic sorting of uploaded items into relevant categories using machine learning image recognition.
- **Smart Recommendations**: Personalized item suggestions generated by analyzing user activity patterns.
- **Community Page**: A feature enabling non-transactional posts, discussions, and peer connections.

2. Requirements

1. Lorem ipsum dolor sit amet, consectetuer adipiscing elit
2. Sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

2.1 Functional Requirements

The product must meet the following functional requirements:

2.1.1. Visitor (Non-authenticated Users)

Authentication & Access:

- Visitor can view the application logo and name on the landing page.
- Visitor can navigate to the Sign In page by clicking the Sign In button.
- Visitor can view SUNY Korea branding on the sign-up page.
- Visitor can initiate Google authentication by clicking "Continue with Google" button.

2.1.2. User (Authenticated Users)

Authentication & Profile Management:

- User can enter their email address and password to sign in.
- User can authenticate their credentials by clicking the Sign In button.
- User can return to previous pages using the back button in headers.

Marketplace & Item Management:

- User can view latest listed items with images and titles on the homepage.
- User can navigate to different sections using bottom navigation bar (Home, Community, Chats, Profile)

- User can search for items using the search button (🔍) to navigate to Search Page
- User can view notifications by clicking the notification button (🔔).
- User can create new listings by clicking the Post button (➕) to navigate to Posting Page.
- User can view detailed item information by clicking/tapping on any item.
- User can view large product images, seller information, item names, prices, and descriptions on Item Detailed Pages.
- User can add items to wishlist using the heart button (❤️).
- User can contact sellers through one-on-one chat using the chat button (💬).

Posting & Selling:

- User can upload up to 10 pictures when creating a listing.
- User can input title, description, and price for their items.
- User can publish items immediately to the marketplace by clicking "Post" button.

Search Functionality:

- User can search for items using keywords in titles/descriptions.
- User can view their search history.
- User can delete individual search history items or clear all history.
- User can trigger search results by selecting previous search history items.

Community Features:

- User can view all community pages (default view).
- User can view specific community pages.
- User can view list of available communities.
- User can create community posts.
- User can view community posts when clicked.

- User can view community descriptions.
- User can create new communities by clicking the ➕ button.
- User can input community name and description (maximum 20 words) when creating communities.
- User can view content of community posts.
- User can comment on community posts.
- User can contact post authors by clicking on their usernames.
- User can create community posts with titles, community categorization, content (maximum 50 words), and media (maximum 4 pictures or 1 video).

Communication:

- User can view list of previous chat rooms.
- User can access specific chat rooms by clicking on them.
- User can view previous chat conversations.
- User can see opponent's names in chat.
- User can send messages to other users.

2.1.3. System (Automated Functions)

Authentication & Security:

- System can validate user credentials and redirect authenticated users to Homepage upon successful login.
- System can restrict registration to only Google accounts.
- System can display error message "Please use your SUNY Korea email account" when non-campus email is attempted.
- System can create new user profiles upon first successful sign-up with university email.

- System can redirect new users to Homepage after successful registration.

AI & Automation:

- System can automatically categorize items using AI when posts are created.

Content Management:

- System can immediately publish items to marketplace when users click "Post" button.
- System can manage chat room access only via contact features in market and community (no individual chat room creation).

2.1.4. Stretch Goals (Future Features)

Enhanced Community Features:

- User can upvote and downvote community posts.
- User can view number of unread chats.
- User can upload media in chat conversations.
- System can implement Twitter/X-like community format.

2.2 Use Cases

2.2.1. Use Case 1 : Register Product for Sale

- Use Case : Register product for sale
- Primary Actor : Student Seller
- Priority : Essential
- Scenario:

1. Student logs into the platform and clicks "Sell Item" button.
2. Student enters product information (title, price, description, category).
3. Student uploads photos of the item (maximum 5 images).
4. Student selects transaction method (meet-up/delivery, preferred location/time).
5. System verifies student's school authentication.
6. Student confirms and publishes the product listing.
7. System displays the product in the marketplace and notifies potential buyers.

- Extensions:

- 2a. Student enters invalid product information.
- 2a.1 System displays validation errors and prompts student to correct the information.
- 5a. Student is not school-verified.
- 5a.1 System redirects student to school verification process before allowing product registration.
- 4a. Student selects inappropriate transaction location.
- 4a.1 System suggests safer, school-approved meeting locations.

2.2.2. Use Case 2 : Search and Inquire about products

- Use Case : Search and inquire about products
- Primary Actor : Student Buyer
- Priority : Essential

- Scenario :

1. Student accesses the main marketplace page.
2. Student applies search filters (category, price range, school, transaction method).
3. Student browses through filtered product listings.
4. Student clicks on a product of interest to view detailed information.
5. Student clicks "Contact Seller" button.
6. System creates a chat room between buyer and seller.
7. Student sends initial inquiry message to the seller.

- Extensions :

- 2a. No products match the applied filters.
- 2a.1 System suggests alternative search criteria or displays similar items.
- 4a. Seller's account is deactivated.
- 4a.1 System displays "Seller Unavailable" message and suggests similar products from active sellers.
- 6a. Buyer and seller are from different schools.
- 6a.1 System displays warning about cross-school transactions and requires additional confirmation.

2.2.3. Use Case 3 : Negotiate Transaction Through Chat

- Use Case : Negotiate transaction through chat

- Primary Actor : Student (Buyer/Seller)

- Priority : Essential

- Scenario :

1. Buyer sends initial inquiry message to seller.
2. Seller responds in the chat room.
3. Both parties negotiate price and transaction conditions.
4. Parties agree on final terms and confirm the transaction.
5. Parties arrange meeting location and time.
6. Transaction is marked as "Confirmed" in the system.
7. Both parties receive transaction confirmation notification.

- Extensions :

- 3a. Negotiation fails to reach agreement.
- 3a.1 Either party can end the conversation, and the chat remains available for future reference.
- 4a. One party requests transaction cancellation.
- 4a.1 System prompts for cancellation reason and notifies the other party.
- 5a. Inappropriate messages are detected.
- 5a.1 System provides reporting functionality and may temporarily suspend chat privileges.

2.2.4. Use Case 4 : School-Based Product Filtering and Verification

- Use Case : School-based product filtering and verification
- Primary Actor : Student
- Priority : Expected
- Scenario :

1. Student registers using school email address.
2. System sends verification email to student's school account.
3. Student completes email verification process.
4. System automatically applies "My School" filter to student's marketplace view.
5. Student can toggle between school-only and all-schools view.
6. System displays trust indicators based on school verification status.
7. Student can verify local meeting spots for safe transactions.

Extensions:

- 2a. School email verification fails.
- 2a.1 System provides manual verification process requiring student ID upload.
- 4a. Student's school is not in the supported schools list.
- 4a.1 System prompts student to request school addition or provides limited access.
- 6a. Student's verification status is revoked.
- 6a.1 System restricts access to verified-only features and prompts re-verification.

2.2.5. Use Case 5 : Post-Transaction Review and Trust System

- Use Case : Post-transaction review and trust system
- Primary Actor : Student (Buyer/Seller)
- Priority : Essential
- Scenario :

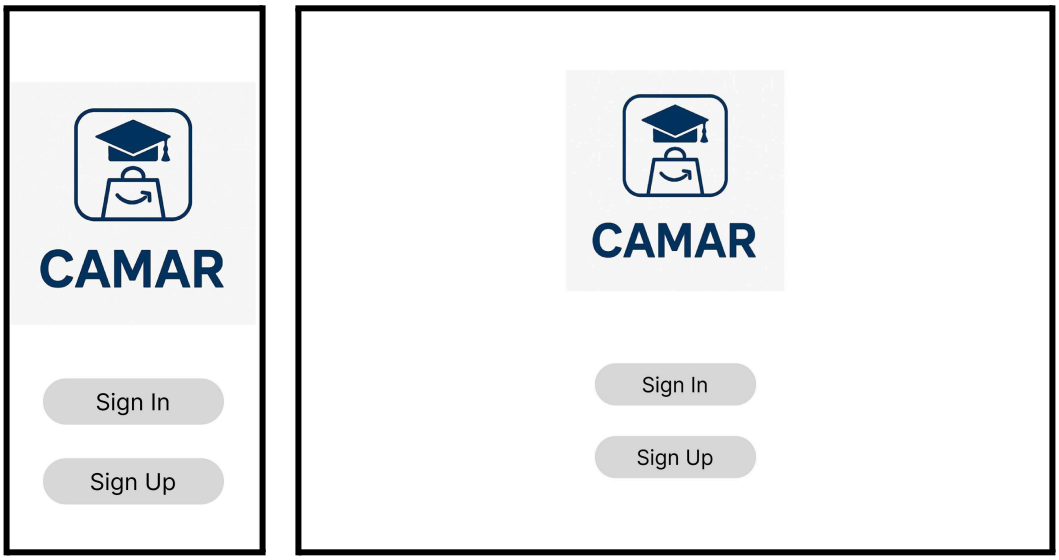
1. Transaction is marked as completed by both parties.
2. System sends review request notifications to both buyer and seller.
3. Student rates the transaction partner (1-5 stars) and writes optional review.
4. System updates trust scores based on received ratings.
5. System displays updated trust indicators on user profiles.
6. High-trust users receive priority in product listings.
7. System maintains review history for future reference.

- Extensions :

- 3a. Student provides negative rating without justification.
- 3a.1 System prompts for detailed feedback and may flag for manual review.
- 4a. One-sided negative review is submitted.
- 4a.1 System notifies the reviewed party and provides dispute resolution process.
- 6a. User's trust score falls below threshold.
- 6a.1 System temporarily limits user's marketplace privileges and suggests improvement actions.

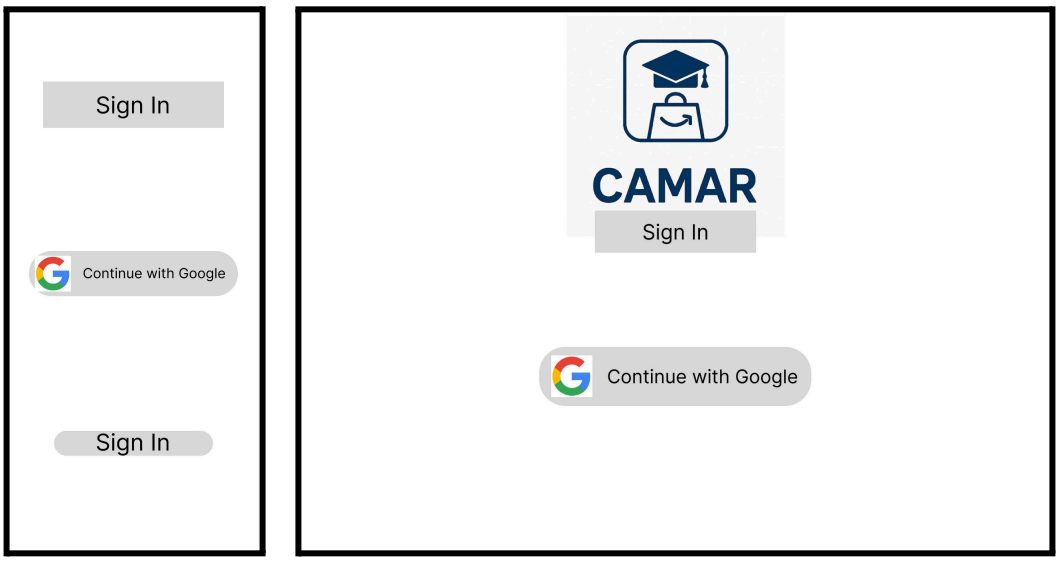
2.3 User Interfaces

2.3.1. Welcome Page



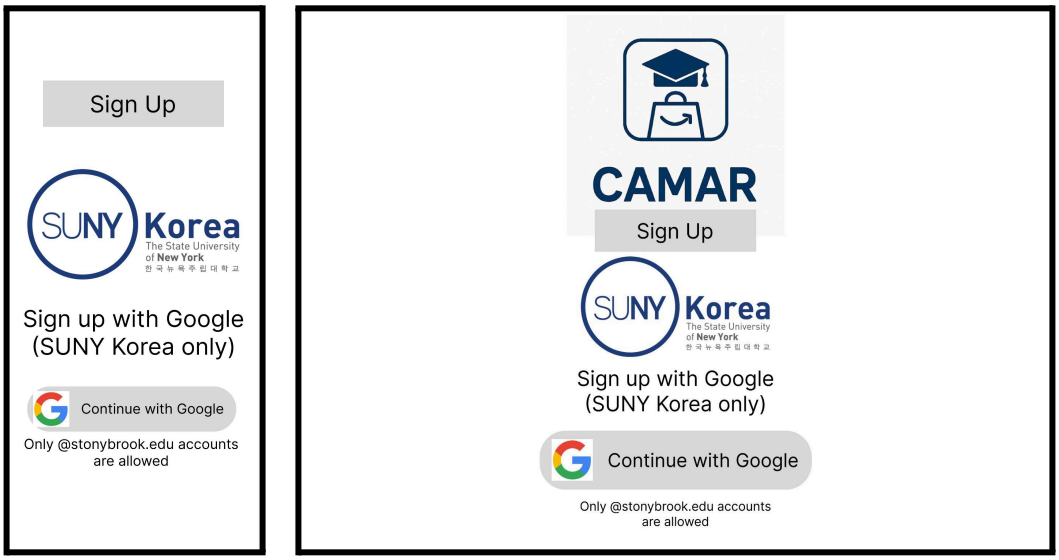
This will be the default page when the user accesses the application.

2.3.2. Sign In Page



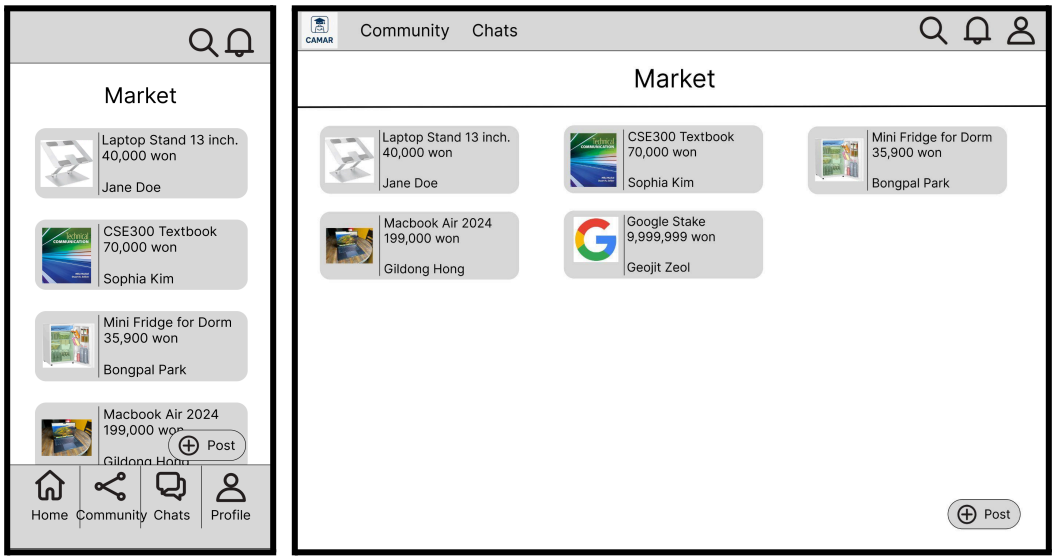
After clicking “Sign In”, the user will be redirected to this page.
All authentication will be done via Google Auth 2.0

2.3.3. Sign Up Page



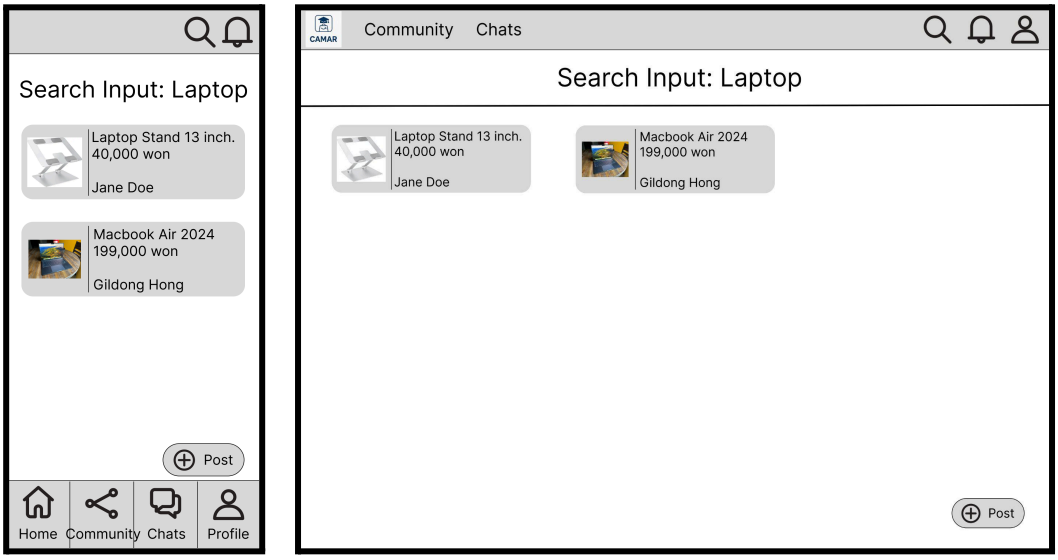
After clicking “Sign Up”, the user will be redirected to this page.
All authentication will be done via Google Auth 2.0

2.3.4. Market Page (Default Home Page)



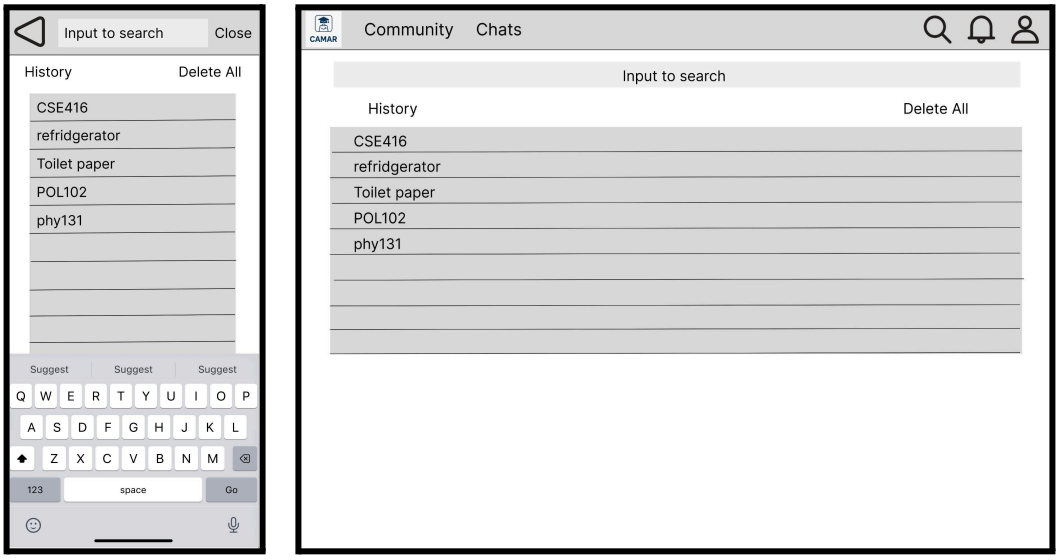
This will be the default home page when the user successfully logs in.
It will show a list of products uploaded by other users.
Pressing the post button (➕) will redirect the user to Item Description Creation Page.

2.3.5. Market Page (Search Result Page)



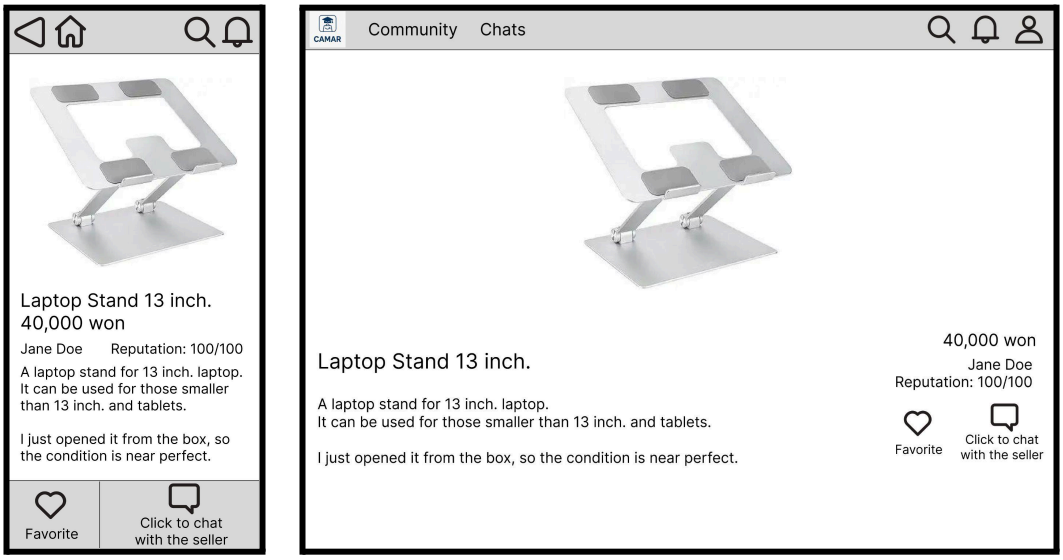
After inputting certain search keywords, the items will be filtered based on title, content, description, and categories.
Clicking on the items will redirect the user to this description page.

2.3.6. Search Page



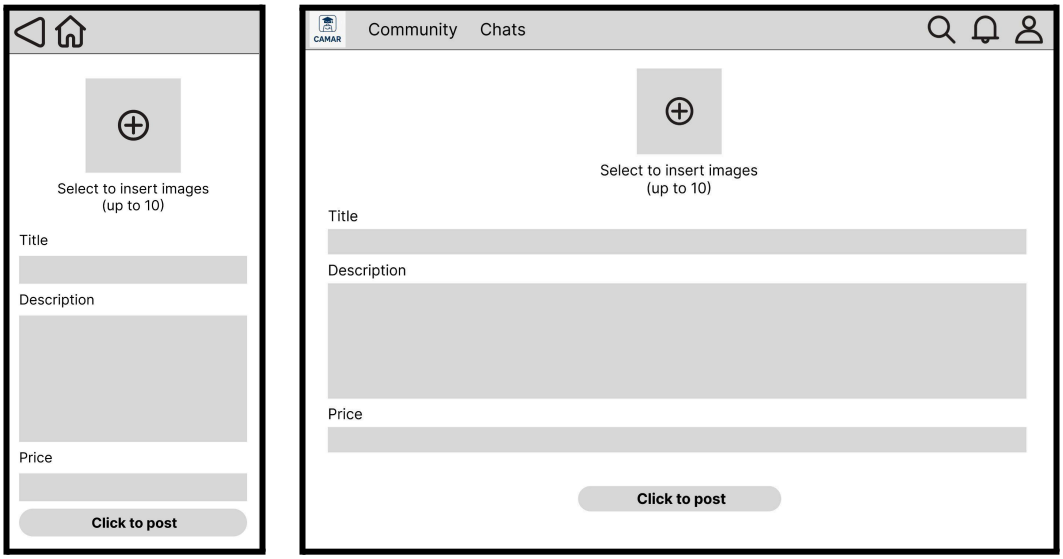
Clicking the magnifying glass icon will redirect the user to this search page.
It will display the history of the current user's search input.

2.3.7. Item Description Page



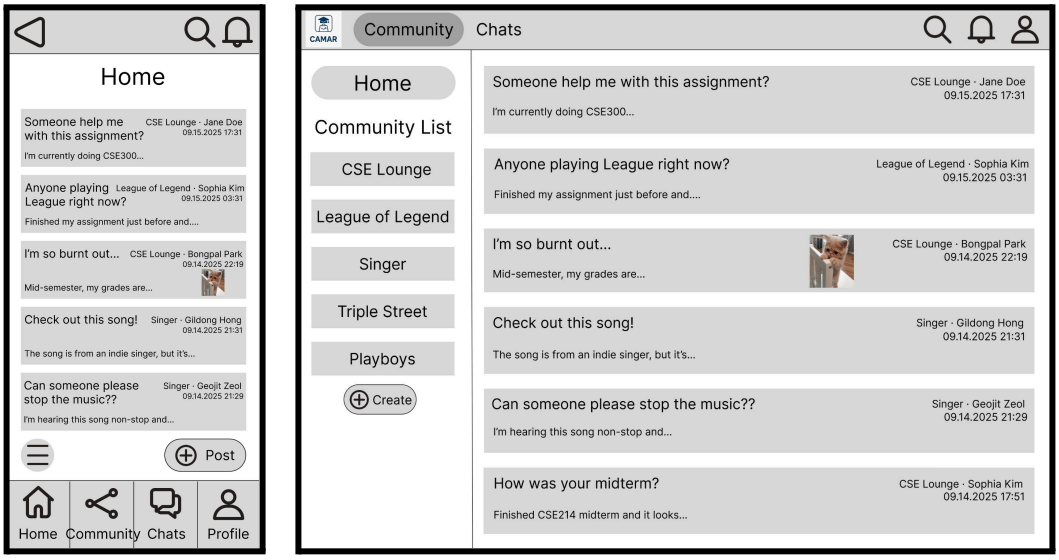
It will show the picture, title, description, and price the writer inputted, with the name and reputation of the user.
After clicking the favorite icon (❤️) , this post will be added to the current user's favorite post list.
After clicking the chat icon (💬), it will redirect the user to the chatting room with the writer.

2.3.8. Item Description Creation Page



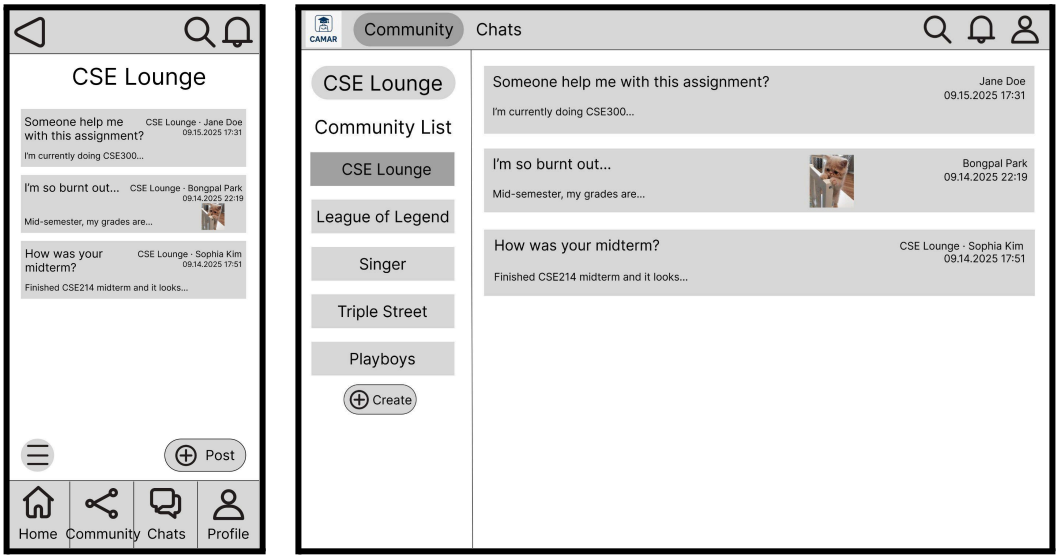
The user will be able to input images (up to 10), title, description and price of the item.
Clicking the "Click to post button" will post the page to the application.

2.3.9. Community Page (Default)



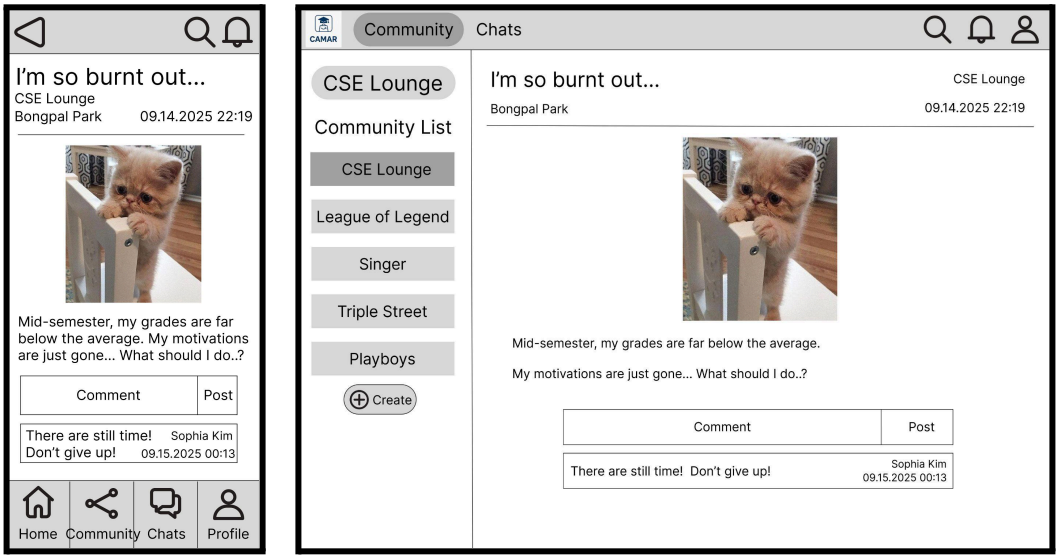
Default page when the user clicked the “Community” button.
It will display all posts in all communities.
Clicking the post items will redirect the user to the Community Post Page.
Community List will be visible to Desktop users on the sidebar.
Mobile users need to click the Community List button on bottom-left.

2.3.10. Community Page (Specific Community)



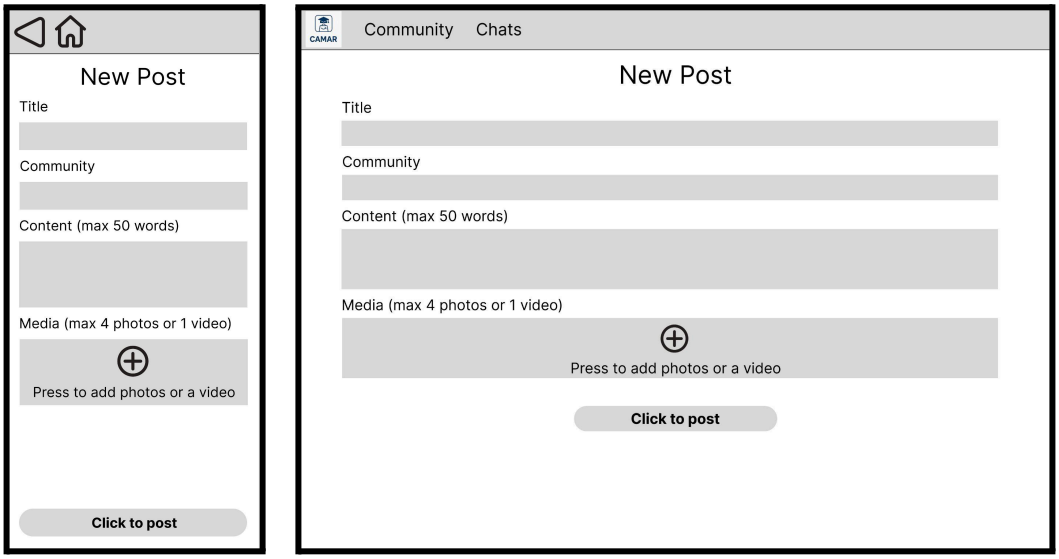
Different view when the user clicks specific community
(In the example, “CSE Lounge” community)

2.3.11. Community Post Page



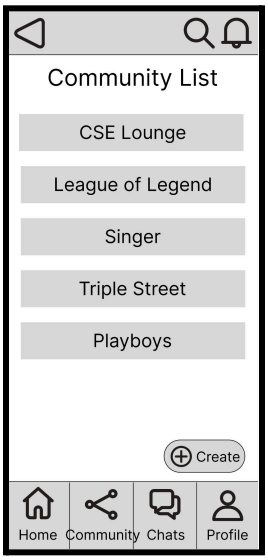
It will display the picture, content, comment input bar, and comments of the current post. Users will be able to post their comments via the comment input bar.

2.3.12. Community Post Creation Page



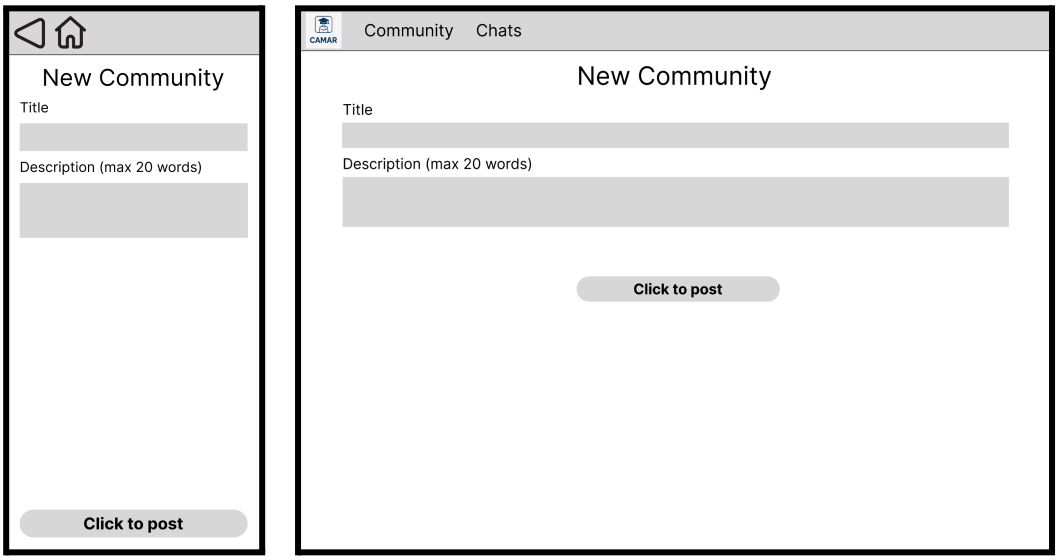
The user will be able to input images (up to 4 photos or 1 video), title, and content of the post.
Clicking the "Click to post button" will post the page to the application.

2.3.13. Community List Page (Mobile only)



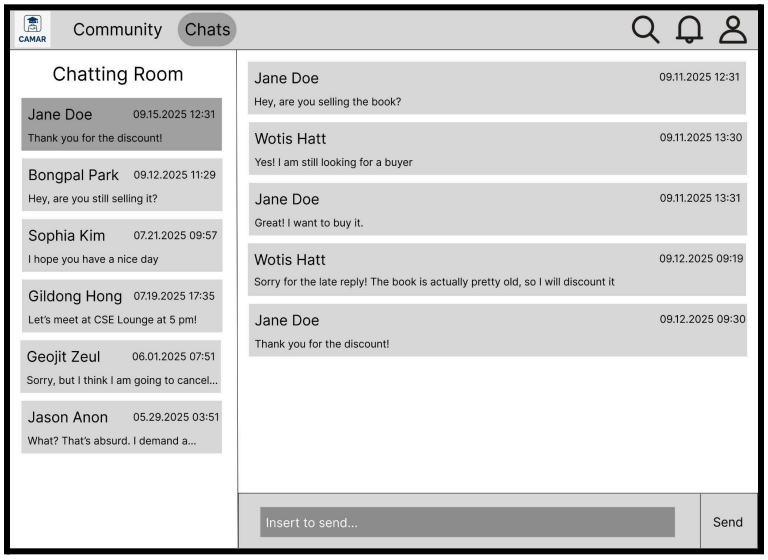
This page will be only visible on mobile view
since the community list will be on the sidebar on the desktop view.

2.3.14. Community Creation Page



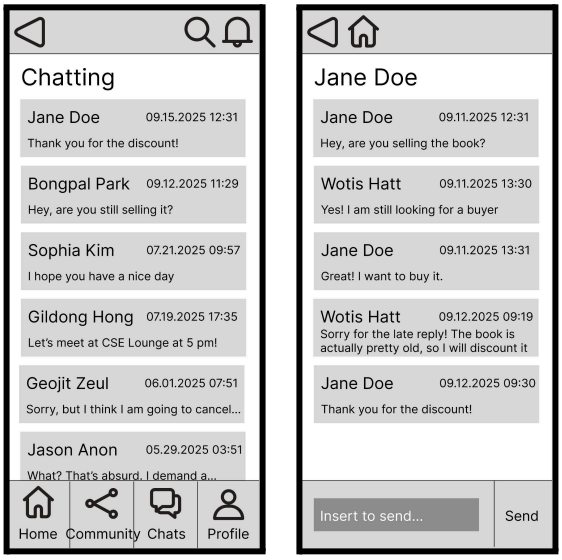
The user will be able to input the title and description of the post.
Clicking the “Click to post” will create the community to the application.

2.3.15. Chatting Page (Desktop only)



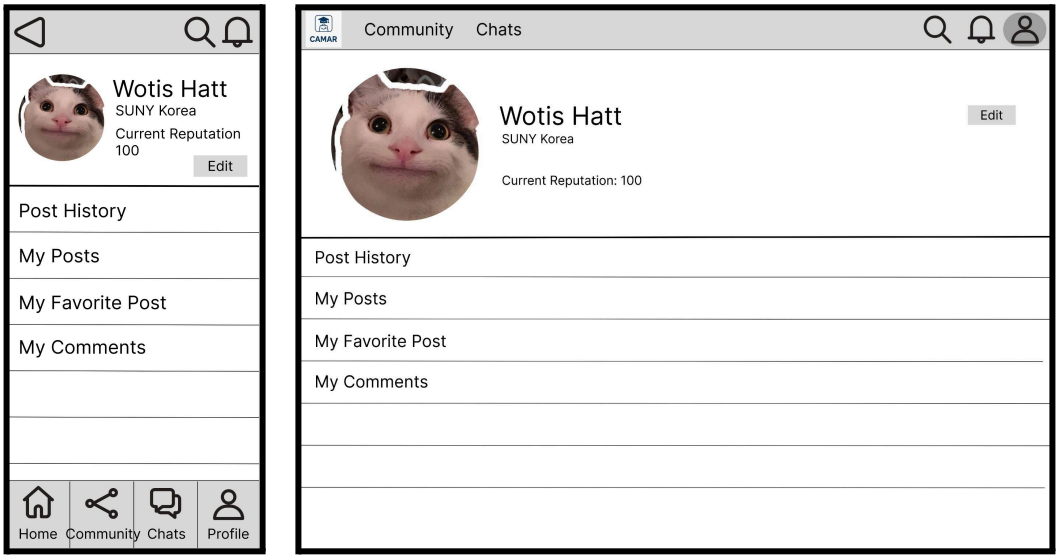
Clicking the “Chats” will redirect the user to the Chatting Page.
Desktop users will see a combined view of list and room.
The user will be able to input chatting to send to buyers.

2.3.16. Chatting Room List & Chatting Room Page (Mobile only)



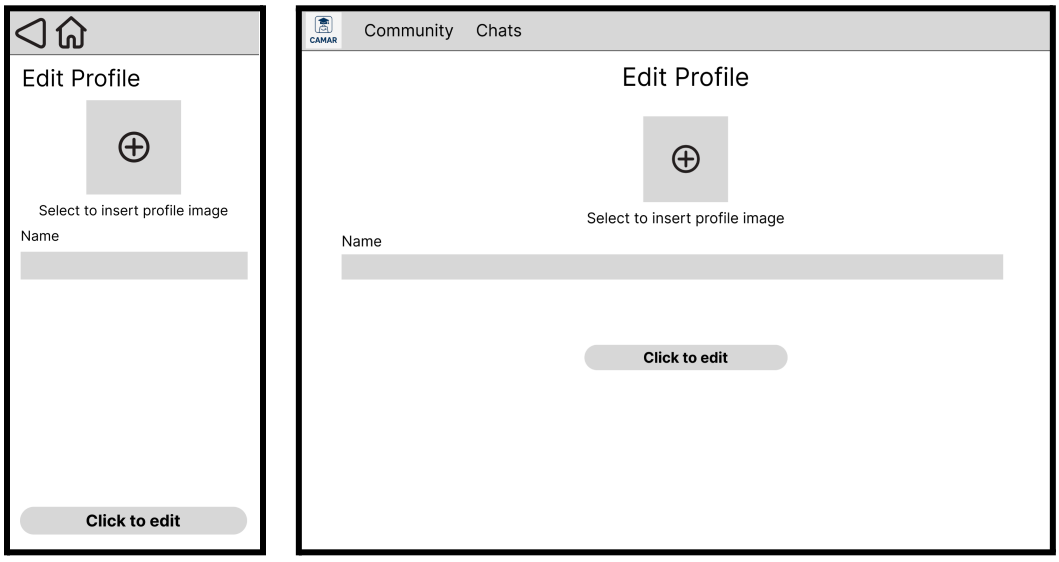
Chatting Room List and Chatting Room will be separated on the mobile view.
The user will be able to input chatting to send to buyers.

2.3.17. Profile Page



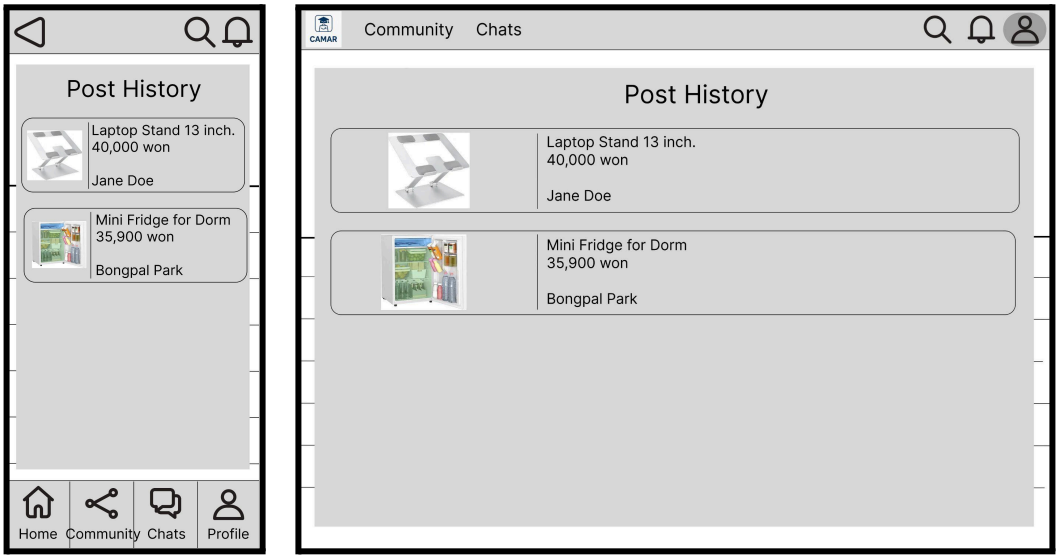
Clicking the Profile button will redirect the user to this page.
Users will be able to view the profile pictures, name the user has set, and the options.
Pressing the Edit button will redirect the user to Profile Edit Page.

2.3.18. Profile Edit Page



The user will be able to input the image of their profile picture and their name.
Clicking the “Click to edit" will edit the user’s profile.

2.3.19. Profile Setting Popup



Clicking the options will open a popup window viewing the list of posts for each individual option.

2.4 Non-functional Requirements

The product must meet the following non-functional requirements:

2.4.1. Performance

- The system shall return search results within **2 seconds** for queries.
- AI categorization of uploaded items shall complete within **5 seconds** after submission.
- The marketplace shall support at least **500 concurrent users** without service degradation.

2.4.2. Security

- Only users with verified SUNY Korea email addresses may create accounts.
- All login sessions shall use **encrypted authentication (HTTPS + JWT)**.

- User data (chat logs, item postings) shall be securely stored and protected against unauthorized access.

2.4.3. Usability

- The web application shall be fully responsive on both desktop and mobile.
- Navigation across core pages (Marketplace, Item Details, Community, Chats, Profile) shall require no more than **3 clicks/taps**.
- The interface shall maintain accessibility standards (WCAG 2.1 AA compliance).

2.4.4. Reliability & Availability

- The platform shall be available **99% of the time**, excluding scheduled maintenance.
- Any service downtime shall not exceed **4 hours per month**.

2.4.5. Maintainability

- The system shall be modular, allowing for updates (e.g., AI model improvements) without disrupting the entire platform.

These requirements are **verifiable** through testing, monitoring, and performance benchmarking.

3. System Architecture

3.1 Overview

The proposed product is an **AI-powered secondhand marketplace web application** tailored specifically for SUNY Korea students. Unlike existing platforms such as Carrot or Bungaejangter, this marketplace directly addresses the unique needs of students by providing a **trustworthy, localized, and student-focused trading environment**.

The primary problem it addresses is the **lack of trust and relevance** of existing secondhand platforms. Public platforms allow anyone to sign up, creating risks of scams, unreliable sellers, and irrelevant listings. Our solution ensures that only verified students with university email addresses can participate, fostering a safe and student-centered community.

3.1 Overview

3.1 Overview

The **Campus Marketplace** system is designed as a full-stack web application that enables SUNY Korea students to buy and sell second-hand items securely and conveniently. The architecture consists of three major components: the **front-end**, the **back-end**, and the **database**, with clear separation of concerns to ensure maintainability and scalability. The system follows the **Model–View–Controller (MVC)** design pattern to organize user interface, application logic, and data management effectively.

3.1.1 Front-End (Client Side)

- **Language & Framework:**
    - **JavaScript (ES6+)**
    - **React 19.2.0** — Used as the main front-end library to create reusable UI components and handle dynamic user interactions efficiently. React's component-based architecture simplifies rendering logic and promotes maintainable code organization.
    - **react-router-dom 7.9.4** — Manages client-side routing, allowing seamless page navigation without full-page reloads. It enables dynamic loading of

different components (e.g., item list, item details, reservation page) based on the URL path.

- **Bootstrap 5.3.8** — A front-end CSS framework used to build responsive layouts and ensure visual consistency across devices. Predefined classes simplify styling for navigation bars, modals, forms, and buttons.

- **Bootstrap Icons 1.13.1** — Provides lightweight SVG icons for UI components such as edit buttons, comment icons, and user profiles, without manually importing icon image files.

The front-end communicates with the Supabase back-end via RESTful API calls and Supabase client SDKs to perform CRUD operations, authentication, and image uploads.

3.1.2 Back-End (Server Side)

- **Platform: Supabase 2.0**
- **Language: TypeScript/JavaScript** (via Supabase Functions)
- **Purpose:** Supabase serves as a **Backend-as-a-Service (BaaS)** that handles authentication, database management, and serverless API endpoints. It eliminates the need for manually configuring servers while still supporting custom logic through **Edge Functions**, which are lightweight serverless functions written in JavaScript or TypeScript.

Key features used:

- **Supabase Auth:** Provides secure user authentication via email and password, enabling login and signup functionalities.
- **Supabase Storage:** Stores and retrieves uploaded images (e.g., item photos, profile pictures) efficiently.
- **Supabase Functions:** Allows implementing backend logic such as reservation validation, item filtering, and database triggers without deploying a separate server.
- **Supabase Realtime:** Enables real-time synchronization of posts, comments, and announcements when changes occur in the database.

3.1.3 Database Layer

- **Technology: PostgreSQL 15 (managed by Supabase)**
- **Purpose:** Stores structured data for users, items, images, tags, and reservations. Supabase provides direct access to the database using SQL queries and an ORM-like interface. The schema enforces relational integrity through foreign-key constraints (e.g., linking images to their corresponding items).

- **Advantages:** PostgreSQL offers ACID compliance, strong query optimization, and compatibility with JSON fields, making it ideal for handling structured and semi-structured marketplace data.

3.1.4 Design Pattern and Architecture

The system adopts a **Model–View–Controller (MVC)**-inspired architecture:

- **Model:** Supabase PostgreSQL tables represent the data model.
- **View:** React components render UI elements and visualize state changes.
- **Controller:** Client-side logic and Supabase Functions act as the intermediary between user input and database operations.

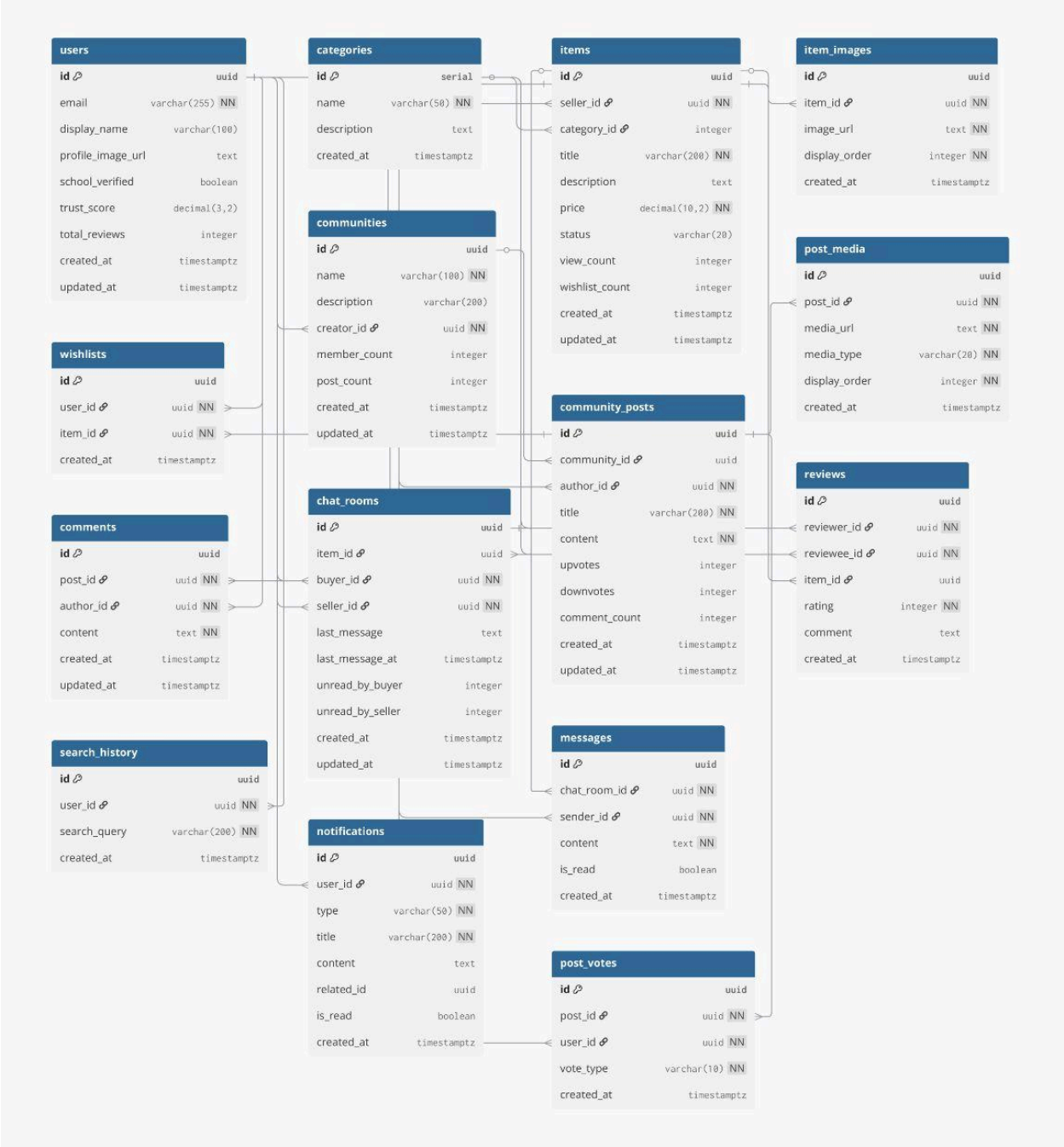
This separation ensures modularity, easy debugging, and clearer data flow.

3.1.5 Rationale for Technology Stack

- **React** was chosen for its component reusability, fast rendering through the virtual DOM, and strong ecosystem support.
- **Supabase** provides an all-in-one backend solution—authentication, storage, serverless functions, and PostgreSQL—significantly reducing development overhead compared to building and deploying a separate Node.js/Express API.
- **Bootstrap** ensures responsive and user-friendly UI design without requiring extensive manual CSS coding.
- **PostgreSQL** was selected for its stability, relational integrity, and compatibility with Supabase's real-time features.

This stack allows rapid development, scalability, and ease of integration with modern cloud services while maintaining code simplicity and maintainability.

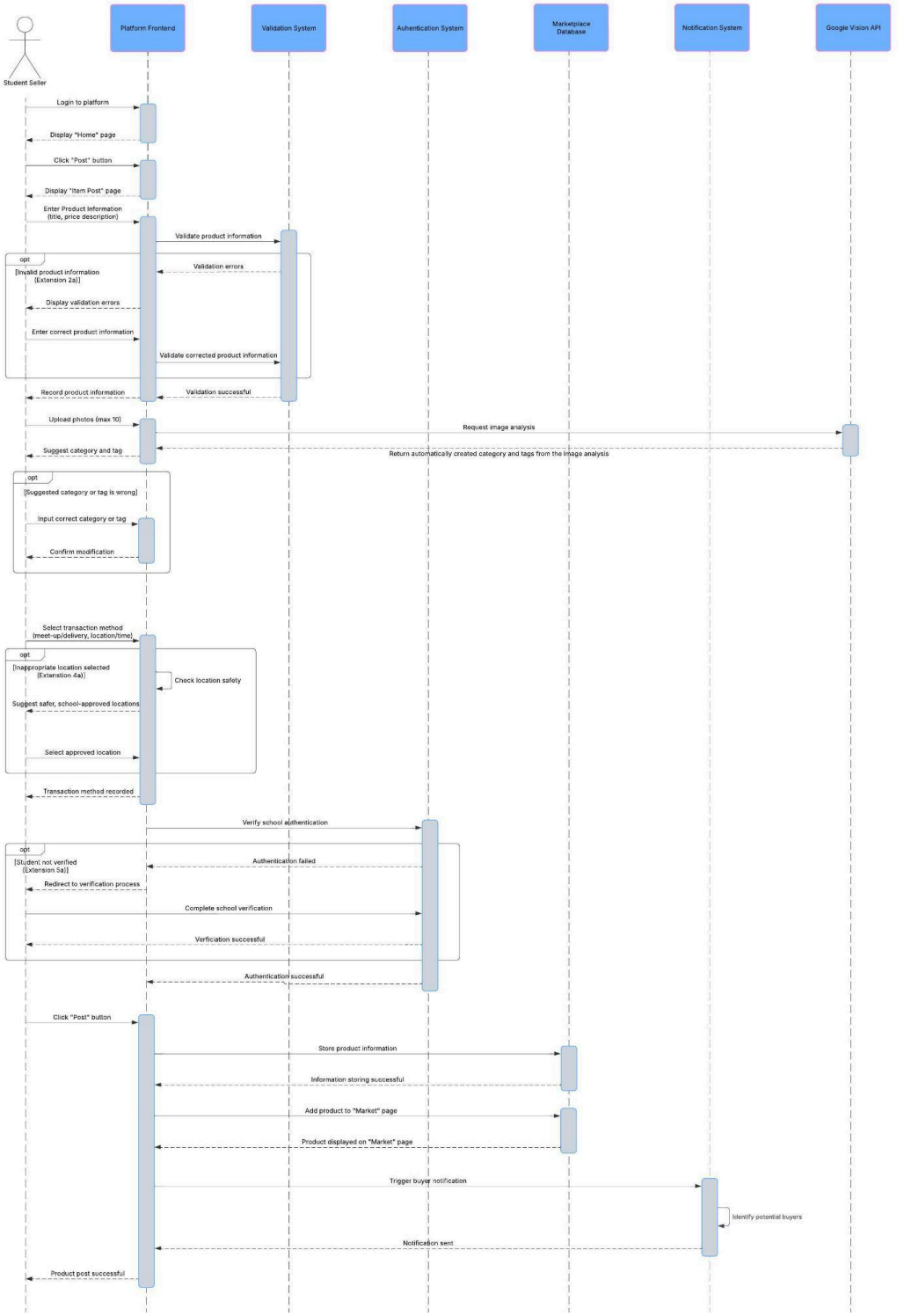
3.2 Data Design



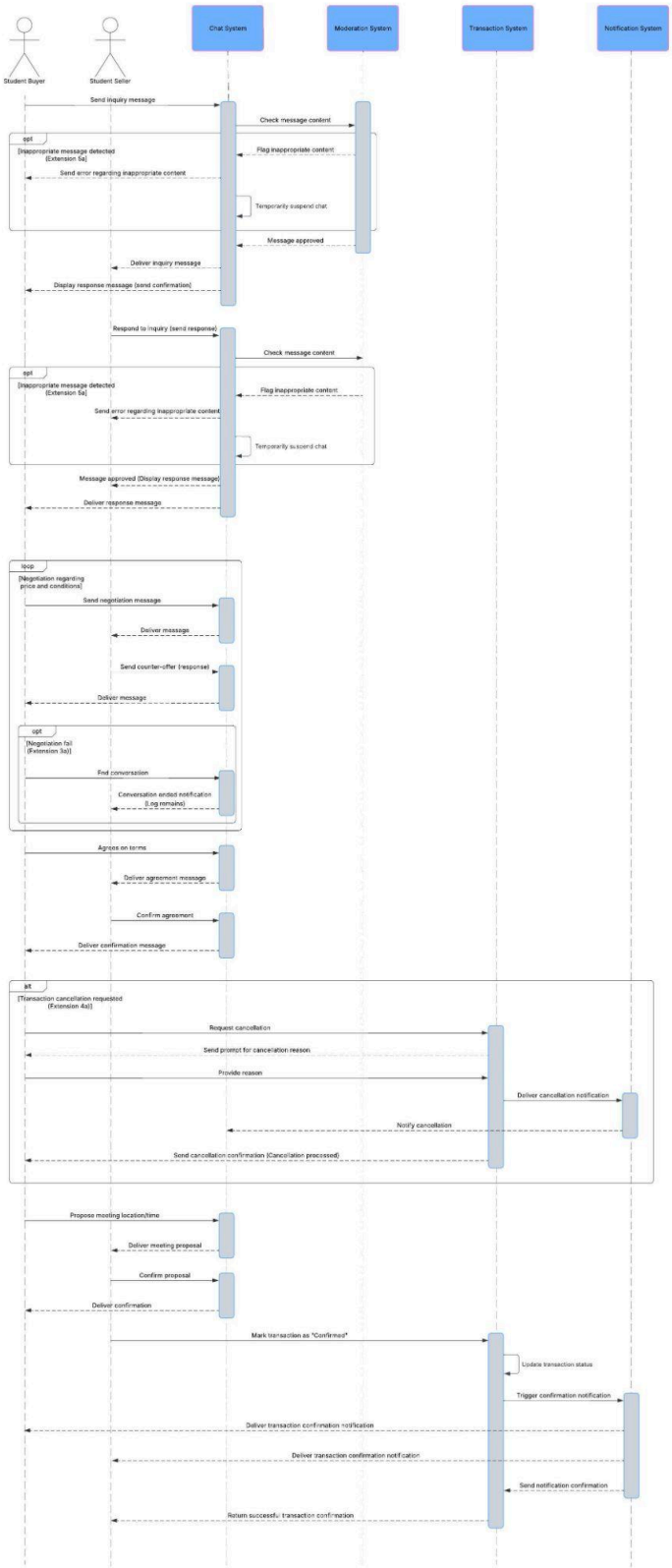

users
id uuid
email varchar(255) NN
display_name varchar(100)
profile_image_url text
school_verified boolean
trust_score decimal(3,2)
total_reviews integer
created_at timestamptz
updated_at timestamptz
categories
id serial
name varchar(50) NN
description text
created_at timestamptz
items
id uuid
seller_id uuid NN
category_id integer
title varchar(200) NN
description text
price decimal(10,2) NN
status varchar(20)
view_count integer
wishlist_count integer
created_at timestamptz
updated_at timestamptz
item_images
id uuid
item_id uuid NN
image_url text NN
display_order integer NN
created_at timestamptz
communities
id uuid
name varchar(100) NN
description varchar(200)
creator_id uuid NN
member_count integer
post_count integer
created_at timestamptz
updated_at timestamptz
post_media
id uuid
post_id uuid NN
media_url text NN
media_type varchar(20) NN
display_order integer NN
created_at timestamptz
wishlists
id uuid
user_id uuid NN
item_id uuid NN
created_at timestamptz
community_posts
id uuid
community_id uuid
author_id uuid NN
title varchar(200) NN
content text NN
upvotes integer
downvotes integer
comment_count integer
created_at timestamptz
updated_at timestamptz
reviews
id uuid
reviewer_id uuid NN
reviewee_id uuid NN
item_id uuid
rating integer NN
comment text
created_at timestamptz
comments
id uuid
post_id uuid NN
author_id uuid NN
content text NN
created_at timestamptz
updated_at timestamptz
chat_rooms
id uuid
item_id uuid
buyer_id uuid NN
seller_id uuid NN
last_message text
last_message_at timestamptz
unread_by_buyer integer
unread_by_seller integer
created_at timestamptz
updated_at timestamptz
messages
id uuid
chat_room_id uuid NN
sender_id uuid NN
content text NN
is_read boolean
created_at timestamptz
search_history
id uuid
user_id uuid NN
search_query varchar(200) NN
created_at timestamptz
notifications
id uuid
user_id uuid NN
type varchar(50) NN
title varchar(200) NN
content text
related_id uuid
is_read boolean
created_at timestamptz
post_votes
id uuid
post_id uuid NN
user_id uuid NN
vote_type varchar(10) NN
created_at timestamptz

3.3 UML Sequence Diagrams

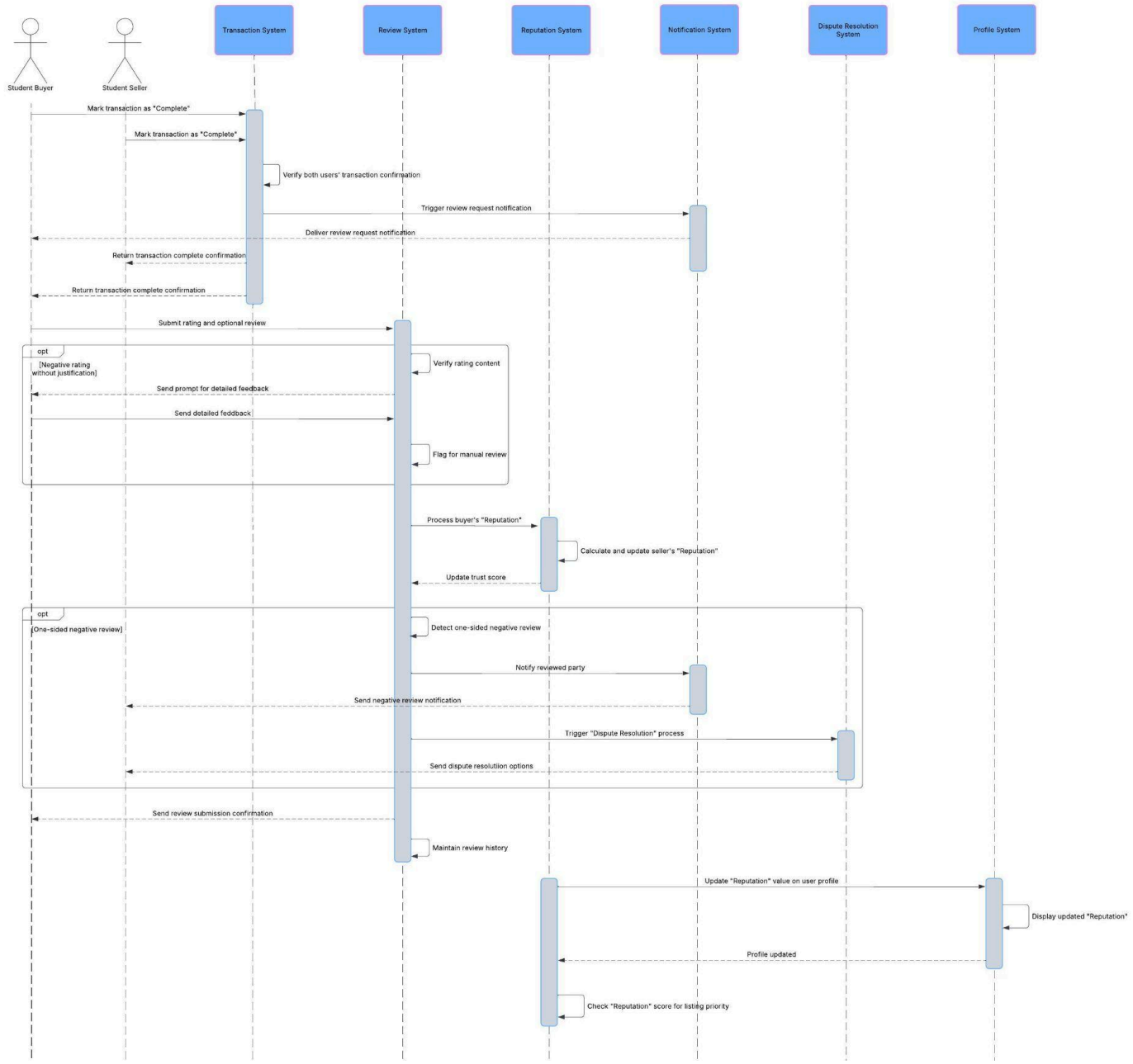
3.3.1 Use Case 1: Register Product for Sale



3.3.2 Use Case 3: Negotiate Transaction Through Chat



3.3.3 Use Case 5: Post-Transaction Review and Trust System



3.4 API Design

The following link connects to our API Design spreadsheet for our BackEnd: 416 API sheet. The format of the design document follows the example that was presented in the assignment announcement.

3.5 Deployment

The **Campus Marketplace** web application will be deployed using **Vercel** for the front-end and **Supabase** for the back-end.

Vercel hosts the React (19.2.0) client and automatically builds and deploys the project whenever changes are pushed to the GitHub repository. Each pull request generates a **preview deployment** for testing, while the **main branch** is automatically promoted to production.

Configuration

- **Build command:** `npm run build`
- **Output directory:** `build/` (or `dist/` if using Vite)
- **Environment variables:**
  - `VITE_SUPABASE_URL` – Supabase project URL
  - `VITE_SUPABASE_ANON_KEY` – public API key
  - `VITE_SITE_URL` – production domain for redirects

Supabase manages authentication, PostgreSQL database, file storage, and optional serverless functions. No separate server is required—Vercel's frontend directly interacts with Supabase through its SDK.

Domain & Security

- The site runs under a secure HTTPS domain (e.g., `https://campusmarket.vercel.app`).
- All data requests are encrypted, and Supabase enforces Row-Level Security (RLS).

Rationale:
Vercel provides effortless CI/CD, global CDN distribution, and fast rollbacks, while Supabase offers a scalable managed backend with minimal configuration. This combination allows the team to focus on features instead of infrastructure.

Project Proposal: Sanghoon Lee, Jaeheon Park, Kitae Kim

3.6 Code Convention

Our team follows the **Google JavaScript Style Guide**
(URL: https://google.github.io/styleguide/jsguide.html)
to ensure **readability, consistency, and maintainability** across all source files.

Key practices include:

- **Naming:**
  - Variables and functions use `camelCase` (e.g., `handleSubmit`, `userName`).
  - React components and classes use `PascalCase` (e.g., `ReservationForm`, `ItemCard`).
  - Constants are written in `UPPER_SNAKE_CASE` (e.g., `MAX_ITEM_COUNT`).

- **Formatting:**
  - Two-space indentation throughout the codebase.
  - Each file starts with a brief comment describing its purpose.
  - Inline comments explain non-obvious logic.
  - Imports are grouped logically (React → third-party → local).

- **Linting & Style Checking:**
  - ESLint (configured with the Google JS style preset) automatically checks adherence.
  - Prettier is used for automatic code formatting before commits.

This convention promotes clean, consistent, and professional-quality JavaScript code across the project.

4. Schedule

The **Campus Marketplace** project will be developed over four milestones, progressing from core functionality to a public beta release. The schedule divides responsibilities among three primary components: **Frontend**, **Backend**, and **AI** integration. Each milestone builds on the previous one to ensure stable and continuous development.

Milestone 1 – Initial Implementation

Goal: Build the foundation of the application (UI skeleton + basic API connectivity).

Area	Task	Assigned Member(s)	Dependencies
Frontend	Implement **Community Page UI** with navigation and post structure	Jaeheon Park	None
Backend	Implement **core Supabase API connections** for posts, users, and items	Kitae Kim	None
AI	Prototype **automated category & tag generation I** (rule-based + keyword extraction)	Sanghoon Lee	None

---

Milestone 2 – Feature Expansion

Goal: Connect backend logic and extend marketplace functionality.

Area	Task	Assigned Member(s)	Dependencies
Frontend	Develop **Marketplace UI** (item list, detail view, filter/sort)	Jaeheon Park	Milestone 1 UI
Backend	Extend **API endpoints** and integrate **AI model connections** for automated tagging	Kitae Kim + Sanghoon Lee	Milestone 1 API
AI	Implement **similar-item recommendation system** using embedding similarity	Sanghoon Lee	Milestone 1 data

---

Milestone 3 – Integration & Security

Goal: Implement secure authentication and finalize core pages.

Area	Task	Assigned Member(s)	Dependencies
Frontend	Complete **Profile UI** and **Chat UI**, integrate Supabase Auth state	Jaeheon Park	Milestone 2
Backend	Add **Google OAuth** and multi-auth (email, Google) with Supabase	Kitae Kim	Supabase setup
AI	Improve **recommendation accuracy** and refine tagging pipeline	Sanghoon Lee	Milestone 2 model

---

Milestone 4 – Beta Release (Week 10 – Week 12)

Goal: Polish UI/UX, test all components, and deploy live beta.

Area	Task	Assigned Member(s)	Dependencies
Frontend	UI refinement, mobile responsiveness, final testing	Jaeheon Park	All previous
Backend	Finalize API, enable monitoring/logging, connect production Supabase	Kitae Kim	Milestone 3
AI	Integrate final **AI tagging + recommendation** into live app	Sanghoon Lee	Milestone 3
Deployment	Deploy to **Vercel + Supabase**, conduct beta testing	Entire Team	All components ready

---

Dependencies Summary

- **Frontend → Backend:** All UI components depend on Supabase API endpoints.
- **Backend → AI:** AI outputs feed into item creation and recommendation APIs.
- **All → Deployment:** Final integration and testing depend on completion of UI, API, and AI modules.

5. Contributions

Introduction: Sanghoon Lee

Requirement: Sanghoon Lee, Jaeheon Park

System Architecture:

Overview: Sanghoon Lee
Data Design: Kitae Kim
UML Sequence Diagram: Jaeheon Park
API Design: Kitae Kim
Deployment: Jaeheon Park
Code Convention: Jaeheon Park

Schedule: Jaeheon Park, Kitae Kim, Sanghoon Lee