Korea Advanced Institute of Science and Technology

School of Computing

CS350 - Introduction to Software Engineering

TRAD’M Application - Software Requirements Specification

Team 7

Members:

- Kern Fowler - 20196550
- Nabila Sindi - 20180744
- Mohammed Almaazmi - 20170850
- Zihan Qi - 20196362

Project Supervisors:

- Doo-Hwan Bae
- Sumin Park (Teaching Assistant)
- Sangwon Hyun (Teaching Assistant)

6th October 2019

Contents

1 - Project Introduction

1.1 - Purpose

The following SRS document describes our in-development project “TRAD’M”, which is a second-hand peer-to-peer marketplace mobile application.

This document outlines the overall product description, functional and non-functional requirements of the product, use cases for user interactions with the product and gives a basic possible UI example. It is intended to be viewed by the project managers, designers, programmers and testers; as well as any end user wishing to learn what the project can do. However, this document will not fully explain how the program works and the user should refer to the manual for more information.

1.2 - Scope

TRAD’M will allow users to sell their unwanted items for money, and allow buyers to get certain items at a reduced price. It also allows people to list services they are willing to provide to other people, and the price. We are currently focusing our clients on only students at KAIST, with international students being the main clientele. However, we can later adapt our application to include users of other neighbouring universities. We believe our application will help international students as it will allow them to save money on arrival to KAIST as they can buy required items at a reduced price. It will also allow them to sell on these items to new students when they are leaving KAIST, this reduces waste and again helps them save money. The services section will allow full-time KAIST students, or other students with certain skills or free time, to help new students and earn money.

We will be including basic features for this type of application, such as; marketplace area, search feature and account creation and login. This should get our app working at a base level, we can then add more features as the project develops over time. Later on in this document we list all the features with their priority.

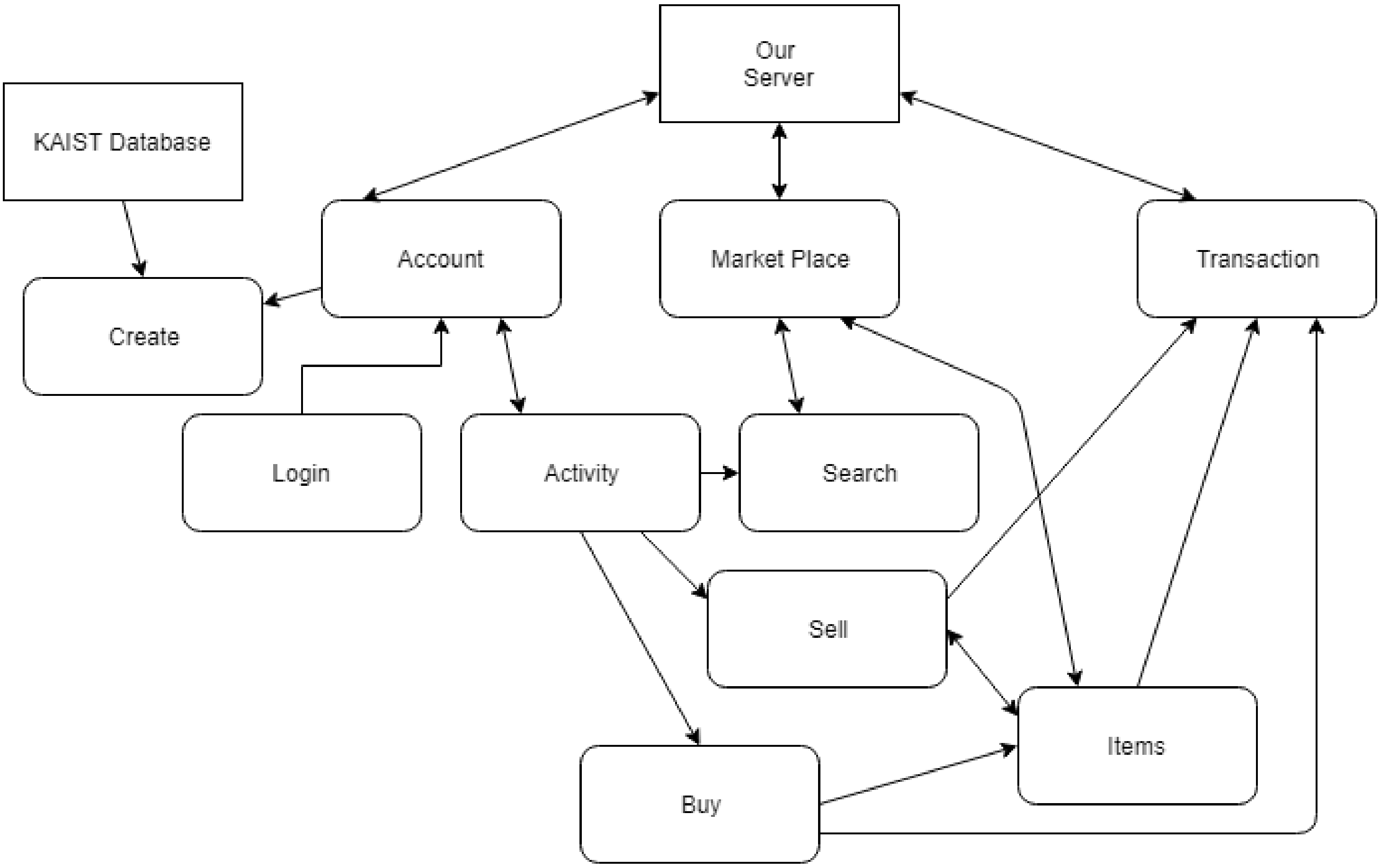
1.3 - Team Vision Statement

i) Our goal in creating TRAD’M is to create an app that will help international students in KAIST find the items they need and avail of the services they need help with without having to suffer through the language barrier since most international students in KAIST are not fluent enough in Korean to communicate with locals. Our hope is that this will lead to an easier and happier life at university for international students at KAIST.
ii) All of us in the group are busy to varying degrees. As such, we have to manage our time efficiently to complete this project. Furthermore, we are using a democratic team organizational style to foster creativity and innovation. This allows to come up with different ideas for our application that help fulfill our goal in creating TRAD’M. We are also dividing the work on the basis of capabilities of motivation of the people involved; assigning members to the tasks they suit and want. This allows us to work efficiently to complete the project in what precious little time we have.
iii) Our app should fulfill the niche role of bringing sellers and buyers together to help international students in KAIST acquire the goods and services they require. For that purpose, our app should be able to create and show offers to prospective buyers as well as let buyers purchase the offers on display. The app should include auxiliary functions that help facilitate this exchange of goods and services such as searching. Furthermore, to entice people to use the app, the app should include features that would attract and retain potential users as well as basic features that are needed in a goods/service market app. These include features such as reliability, usability, availability and performance.

2 - Overall Description

2.1 - Product Perspective

TRAD’M is a new stand-alone application that is aimed to reduce the hassle of moving into KAIST as a new international student. Its marketplace feature will allow, old and new KAIST international students to help each other save money and reduce waste items, benefiting the environment. The services feature allows full-time KAIST students to gain money by lending their time to help others. TRAD’M will be free to use, and easy to use on the go as it is a mobile application.



2.2 - Product Features

This application is designed to help users sell and buy idle second-hand items, thereby saving a lot of money. Users can purchase physical goods or services (such as delivery services) on this application. In terms of searching for goods, buyers can search for goods according to keywords, commodity conditions and commodity category. When the buyer wants to buy goods, he can negotiate with the seller through the message system and choose the trading place after the two sides decide to trade. In order to ensure the security of the transaction, a confirmation button is added. When both sides complete the transaction, they need to press the confirmation button to determine the success of the transaction. If any one side does not press the confirmation button, the system will determine the failure of the transaction. After the transaction, the seller and buyer can evaluate the transaction as well as their trader. A powerful map positioning support is also in the application, so that users can easily and quickly find the information of goods sold nearby and determine the location of transactions by using it.In addition, the wish list function allows the buyer to record the items he wants to buy, and the system will notify the user when such goods are sold.

2.3 - Operating Environment

The operating environment for the flea market application is as listed below:

1) Distributed database. Database will be taken mainly from the KAIST Server, connecting mainly through an SSO KAIST Login.
2) Client/Server system.
3) Operating System: Android 7.0 and above
4) Database: SQLite
5) Platform: Java, Android Studio, Sketch

2.4 - Assumptions and Dependencies

1) Inexistence of gates for getting and keeping TRAD'M in the Android Market
2) Users have continuous and unhindered access to WiFi or mobile data
3) Users own an Android device and have access to Play Store
4) No user interface bottleneck on mobile
5) TRAD'M can update via the Play Store whenever updates become available
6) TRAD'M uses safe encrypted communication and if related security issues happen, related patches will be automatically applied.
7) Critical bugs and errors rarely occur and if they occur, patches are quickly applied to fix the issues
8) Portability across all Android devices
9) Back compatible up to Android 7.0
10) TRAD'M will only support English
11) TRAD'M will be built using Java as the main language as per the project requirement
12) Existence of an external database to store all information regarding users, products, offers etc. The server must be constantly available to interact with. This might be done virtually using existing and trustworthy cloud databases
13) Use of a mobile app testing platform such as Experitest
14) Use of a mobile analytic tool such as App Watch
15) Use of a mobile app development framework for Android either Native or Cross-Platform such as the Android SDK or Flutter
16) Use of a mobile app design tool to design the app such as Sketch
17) Use of third party dependencies such as Glide, Crashlytics, Guava, (list to be updated as project evolves)
18) Use of APIs to enable access to other applications and platforms such as Facebook API, Twitter API, Google Maps API, YouTube API (list to be updated as project evolves)

3 - System Features

3.1 - Functional Requirements

3.1.1 - Must Have Features

1) **Account creation:** The system will allow the students to create an account using their student IDs.
2) **Login:** Allow registered users to login.
3) **Login Authentication:** The system should authenticate users during login.
4) **Market for buyers:** Create a market that shows all currently listed offers along with their prices; can click for more product information.
5) **Item and service offer creation and edit:** Allow sellers to create an item offer. Item and service offers might include pictures in which case the picture should be shown along with the actual offer. Furthermore, the offer should include the item/service name, selling price range, category

and optionally any further description the seller would like to convey to potential buyers. Furthermore, the seller should have the option to edit any of the details in his/her offer at any point he/she wishes to.

6) **Offer purchase:** Users can purchase offers on display in the marketplace. This sends the seller of the offer a button prompt to confirm the purchase.
7) **Search function:** Allow users to easily filter through all the available offers with certain keywords.
8) **Confirm button:** Add a confirm button for sellers to complete the transaction.

3.1.2 - Should Have Features

1) **Message system function:** Create a built-in messaging system where users can get more information, bargain or arrange meetups.
2) **Settings page:** Creation of a settings page where users can update their account information and apply for deletion of their accounts if they so wish.
3) **External sharing and access:** Allow commodity information to be shared through links to users' friends and to users' social media(through other social software). Product information, pictures, videos, etc can be viewed if available externally such as on Youtube etc.

3.1.3 - Could Have Features

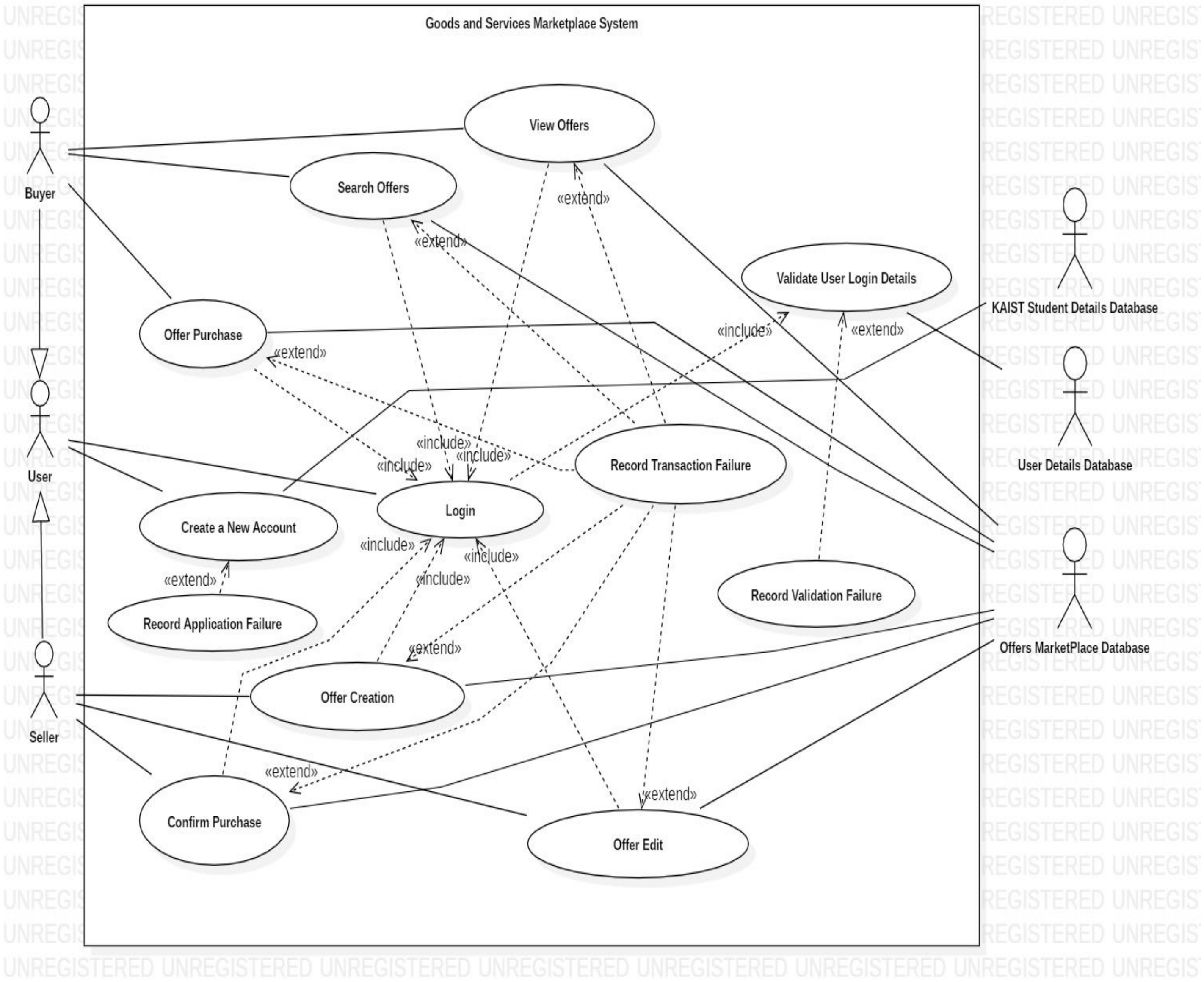
1) **Categoric sorting:** Seller will state what category the offer falls under. Allow buyers to search for offers in their selected category with their preferred price range, offer condition/features and borrowing length.
2) **Shopping basket:** Create a shopping basket where users can see the offers they have chosen while they are shopping as well as click on a seller to view the offers currently being sold by him/her.
3) **Feedback function**: Allow both sides to give reviews and star ratings according to their trading experience.
4) **Reminders:** Notifications and reminders for users of incomplete transactions or approaching borrowing deadlines.

3.1.4 - Won’t Have Features

1) **Home page:** Creation of a homepage which serves as a main hub for all app activity. It should include links to all other aspects of the app as well as show the information most relevant to the logged in user such as settings, discounts, suggestions, wishlist, and shopping basket.
2) **User page:** Creation of a user page where a user’s information, reviews and offers on sale can be viewed by other users and prospective buyers/sellers. The user should be able to customize the information that appears on this page.
3) **Wish List function:** Create a wish list where users can add a list of goods and/or services they wish to purchase and the price they are willing to pay. The app notifies users if such offers become available.
4) **Suggestion menu:** Suggest hot categories and offers, as well as categories and offers similar to previous user purchases.
5) **Borrowing and returning system for goods:** Allows users to temporarily lend items for free or for a small price. The buyer will return the item after the allocated time.
6) **Map function:** Allows users to pick a meetup location; may have designated locations for security and safety, and may support external map interaction such as Google Maps.
7) **Discount feature:** A feature where sellers can offer to place their offers on discount for a specified duration, in which case a notification is sent to all buyers with this offer in their wish lists informing them of the discount. Furthermore, discounts could be promoted on the main homepage for all to see.
8) **Purchase history:** Track and show a history of users’ purchases. This should only be available to the system and the user and no one else.

3.2 - Use Case Diagrams and Descriptions

3.2.1 - Main Use Case


Goods and Services Marketplace System
Buyer
User
Seller
View Offers
Search Offers
«extend»
Validate User Login Details
KAIST Student Details Database
Offer Purchase
«include»
«extend»
User Details Database
Record Transaction Failure
«include»
Login
Create a New Account
«include»
Record Validation Failure
«extend»
Record Application Failure
Offers MarketPlace Database
Offer Creation
Confirm Purchase
Offer Edit

3.2.2 - Create New Offer

Use case name	Item and service offer creation
Related Requirements	Requirement i. 6).
Goal in Context	An existing user creates a new item or service offer.
Preconditions	User must be logged in.
Successful End Condition	A new item or service offer is created for the user.
Failed End Condition	The item or service offer is not created.
Primary Actors	User (Seller).
Secondary Actors	
Trigger	User clicks on create offer button to make a new offer.

Main Flow	1. User clicks on create offer button. 2. User selects offer type, item or service. 3. User enters the offer details. Details to enter may be different depending on offer type. 4. User's entered details are validated. 5. The new offer is created 6. A summary of the new offer's details is then emailed to the user.
Extensions	4.1. User's entered details could not be validated. 4.2. Display why details which could not be validated. User can attempt to fix details in which case go back to 3. or user can click cancel in which case go to 4.3. 4.3. Offer creation is cancelled.

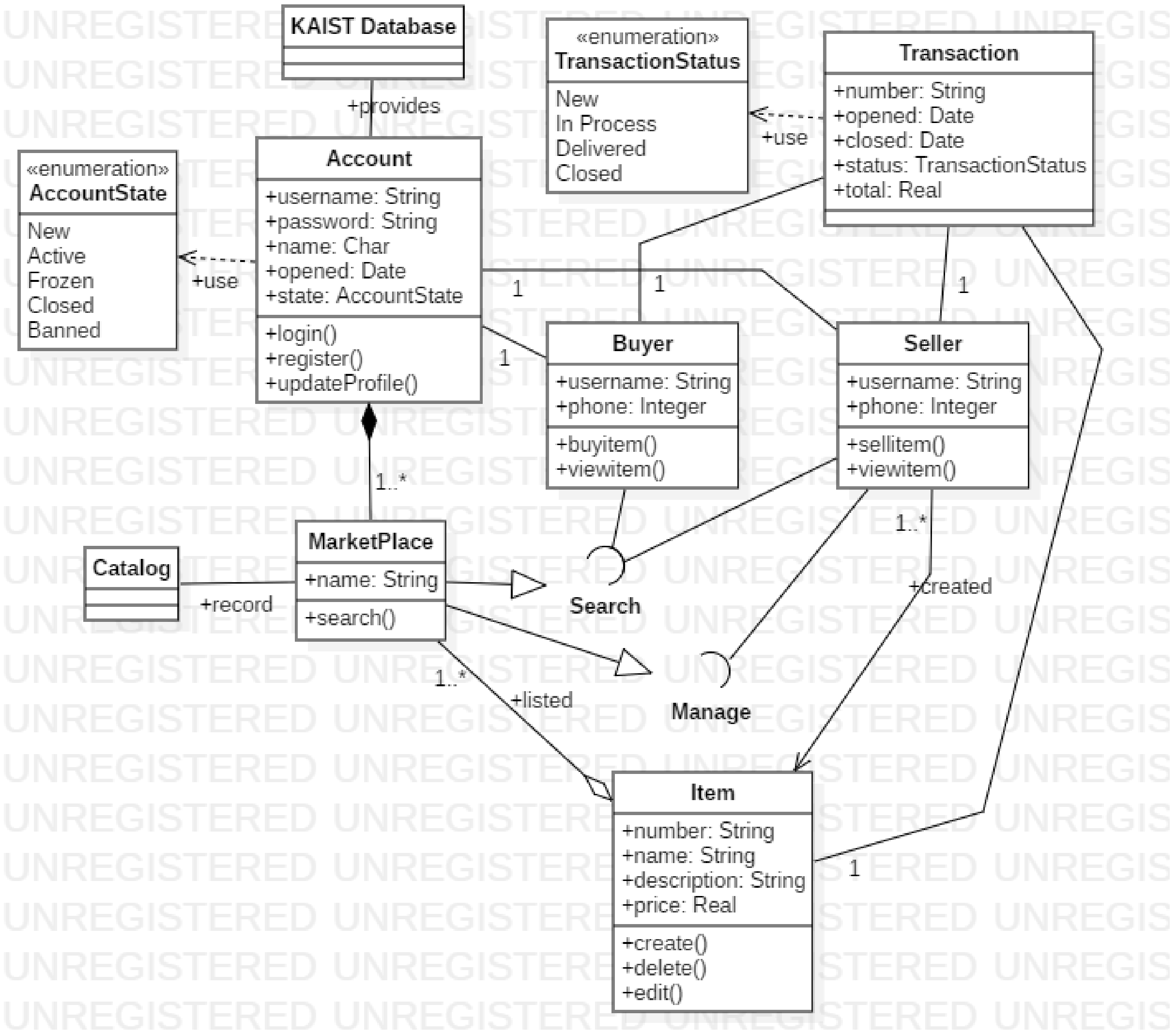
3.2.3 - Create New Account

Use case name	**Item and service offer creation**
Related Requirements	Requirement i. 1)
Goal in Context	A new or existing user requests a new account from the Administrator
Preconditions	User must have a KAIST SSO Account.
Successful End Condition	A new account is created for the user. Users are able to create and buy offers
Failed End Condition	The application for a new account is rejected.
Primary Actors	Administrator
Secondary Actors	User's KAIST SSO Account Database
Trigger	The administrator asks the CMS to create a new account.
Main Flow	1. The user clicks on create new account button. 2. User enters details. 3. Administrator sends user's details to KAIST SSO Account Database and are verified. 4. The new account is created. 5. A summary of the new account's details are emailed to the user.
Extensions	3.1. User's entered details could not be validated. 3.2. Display why details which could not be validated. User can attempt to fix details in which case go back to 3. or user can click cancel in which case go to 3.3. 3.3. Account creation is cancelled.

3.2.4 - Create Purchase Request

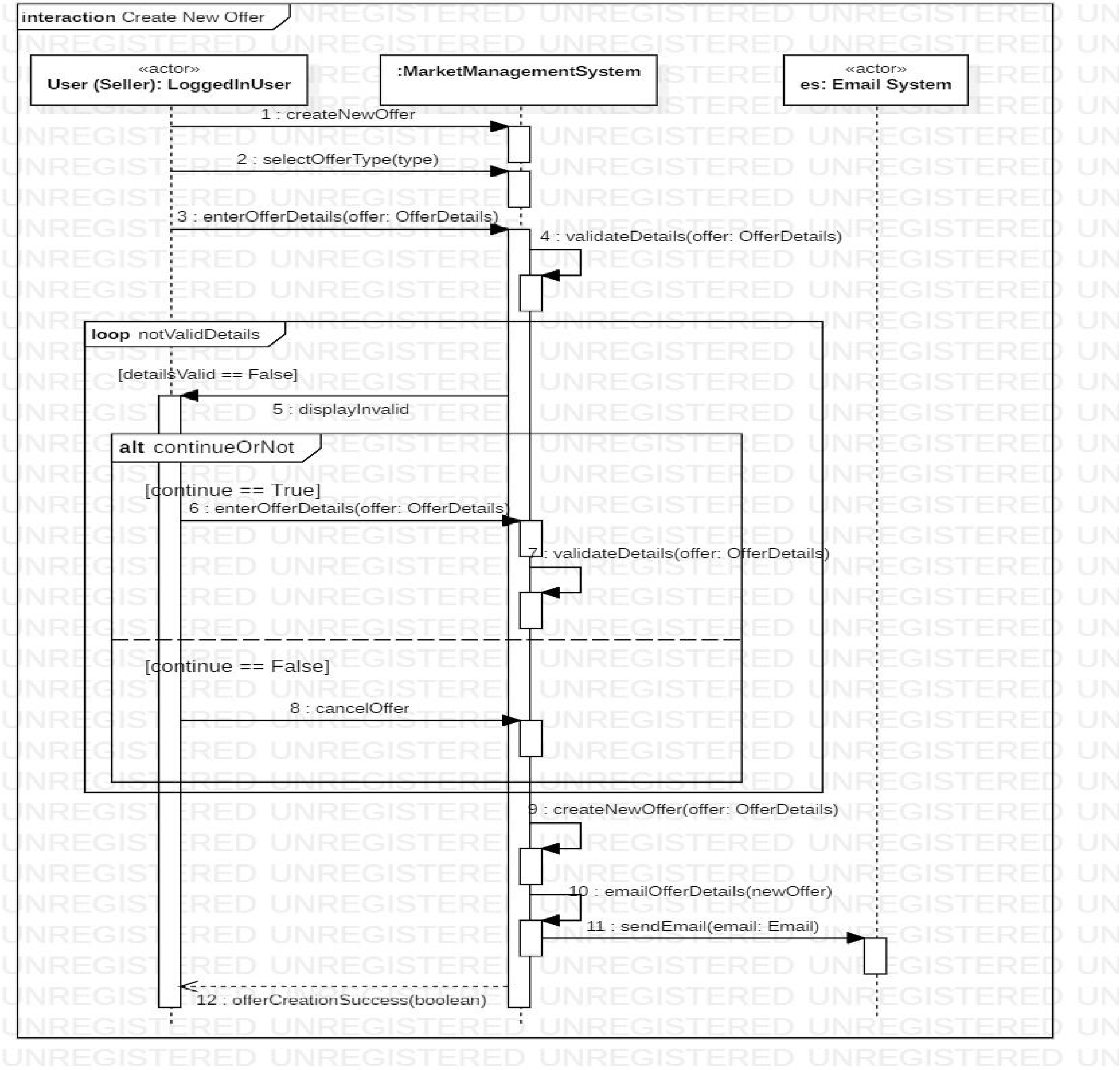
Use case name	**Create purchase request**
Related Requirements	Requirement i. 4)
Goal in Context	Users (buyers) purchase second-hand goods they require.
Preconditions	User must be logged in.
Successful End Condition	Both sides click on the confirm transaction button to finish the transaction.
Failed End Condition	Either buyer or seller cancels the transaction.
Primary Actors	User (Buyer).
Secondary Actors	User (Seller).
Trigger	The user clicks on the purchase button.
Main Flow	1. User browses commodity information. 2. User clicks on contact with seller button to send messages to the seller for detailed information(such as price,the time and the place of transaction). 3. User clicks on purchase button 4.The condition of relative item becomes sold out. 5.User clicks on the confirm transaction button after the transaction(the seller does the same). 6.The proof of purchase will be sent to the user email later.
Extensions	5.1. Either buyer or seller does not click on the confirm transaction button will lead to the failure of the transaction. 5.2. If the transaction was canceled, the condition of relative item becomes available. 5.3. Both sides can write comments of the trade experience after the transaction(either it is successful or not).

3.3 - Domain Model

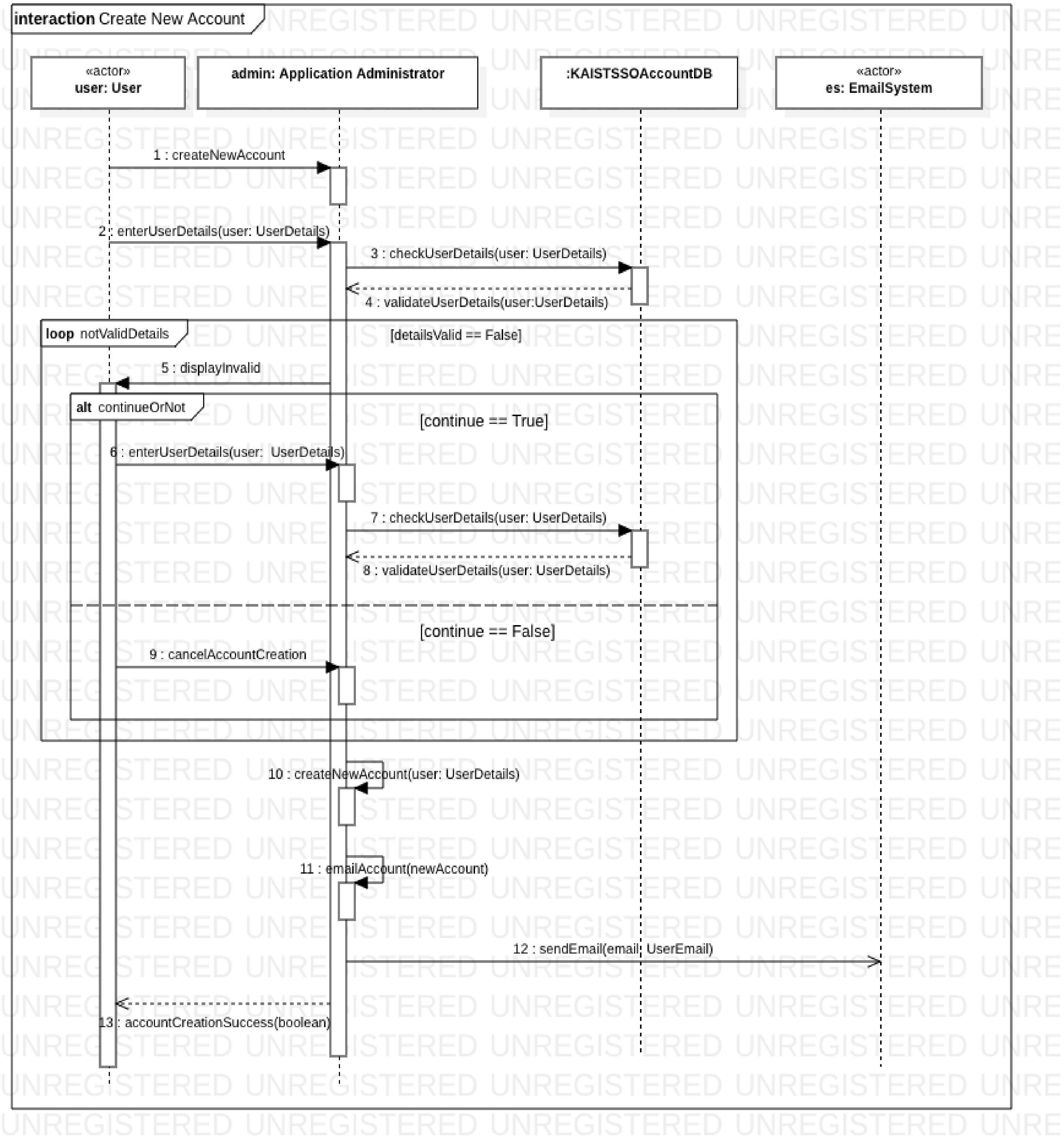


3.4 - Sequence Diagrams

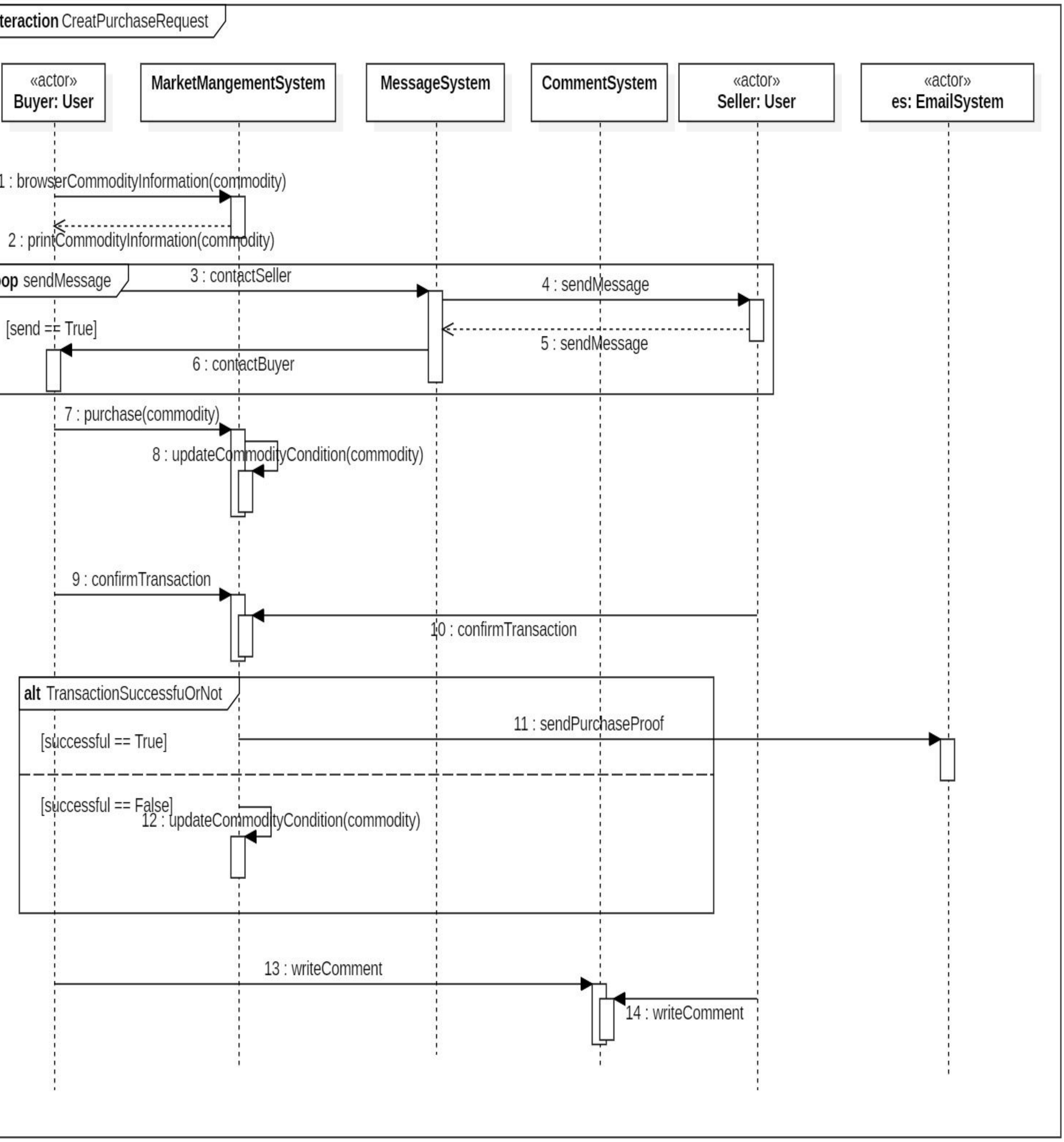
3.4.1 - Create New Offer


interaction Create New Offer
«actor»
User (Seller): LoggedInUser
:MarketManagementSystem
«actor»
es: Email System
1 : createNewOffer
2 : selectOfferType(type)
3 : enterOfferDetails(offer: OfferDetails)
4 : validateDetails(offer: OfferDetails)
loop notValidDetails
[detailsValid == False]
5 : displayInvalid
alt continueOrNot
[continue == True]
6 : enterOfferDetails(offer: OfferDetails)
7 : validateDetails(offer: OfferDetails)
[continue == False]
8 : cancelOffer
9 : createNewOffer(offer: OfferDetails)
10 : emailOfferDetails(newOffer)
11 : sendEmail(email: Email)
12 : offerCreationSuccess(boolean)

3.4.2 - Create New Account


interaction Create New Account
«actor»
user: User
admin: Application Administrator
:KAISTSSOAccountDB
«actor»
es: EmailSystem
1 : createNewAccount
2 : enterUserDetails(user: UserDetails)
3 : checkUserDetails(user: UserDetails)
4 : validateUserDetails(user:UserDetails)
loop notValidDetails
[detailsValid == False]
5 : displayInvalid
alt continueOrNot
[continue == True]
6 : enterUserDetails(user: UserDetails)
7 : checkUserDetails(user: UserDetails)
8 : validateUserDetails(user: UserDetails)
[continue == False]
9 : cancelAccountCreation
10 : createNewAccount(user: UserDetails)
11 : emailAccount(newAccount)
12 : sendEmail(email: UserEmail)
13 : accountCreationSuccess(boolean)

3.4.3 - Create Purchase Request

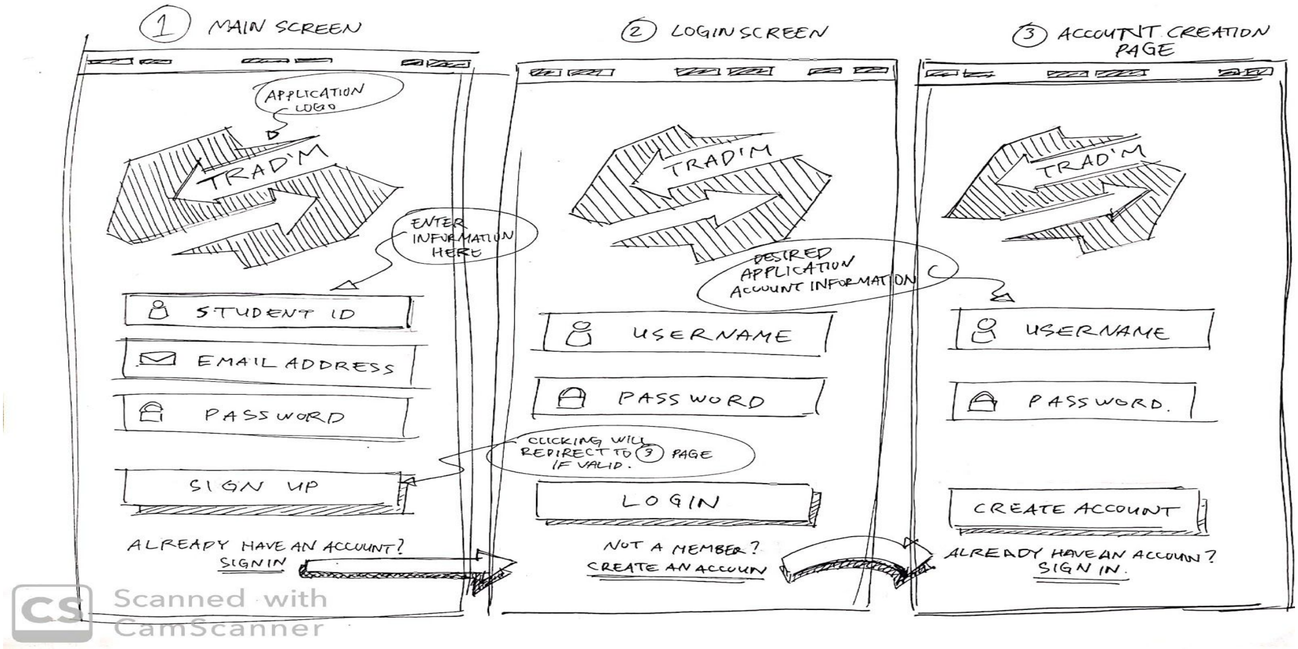


4 - Preliminary User Manual

This guide is intended for students at KAIST who want to use their Android phone to sell and buy services and goods to other KAIST inhabitants.

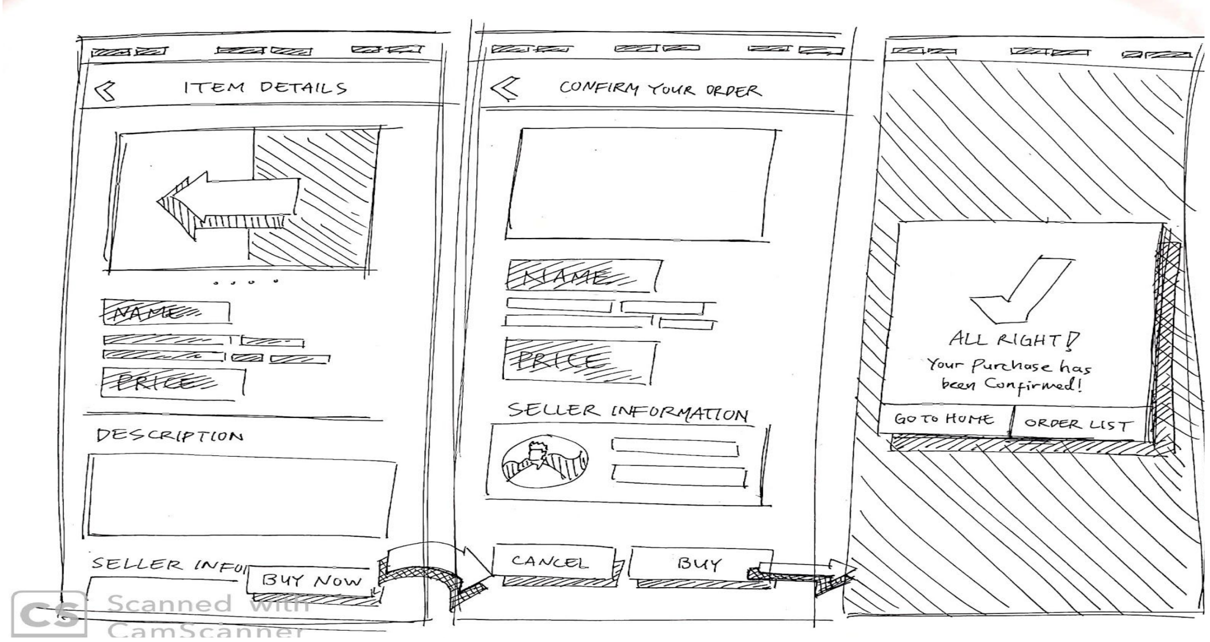
1) Account creation

- Input your KAIST SSO Student ID, email address, password, then press SIGN UP.
- An error will pop up in case of invalid information.
- Input all the required information. Once completed, press CREATE ACCOUNT.



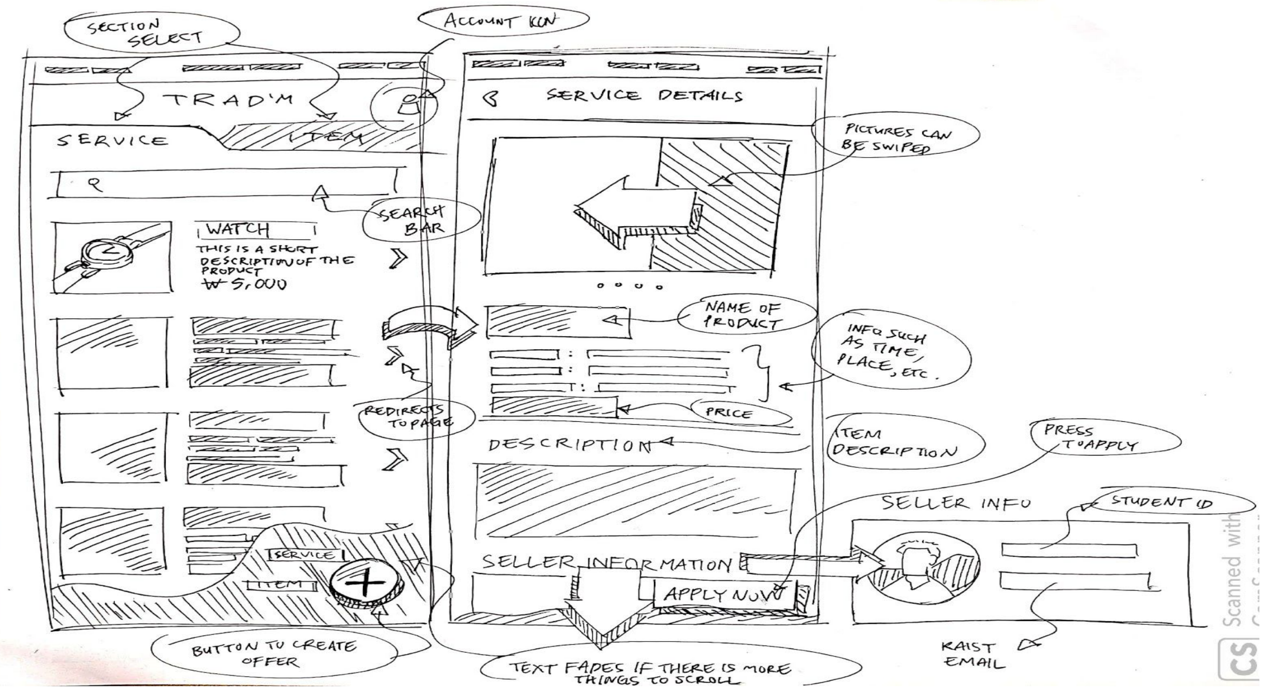
2) Login

- After registration, when you open the Mobile App, you may be asked to Login.
- Under the SIGN UP Icon, press SIGN IN.
- Input your Username and Password, then select Login.



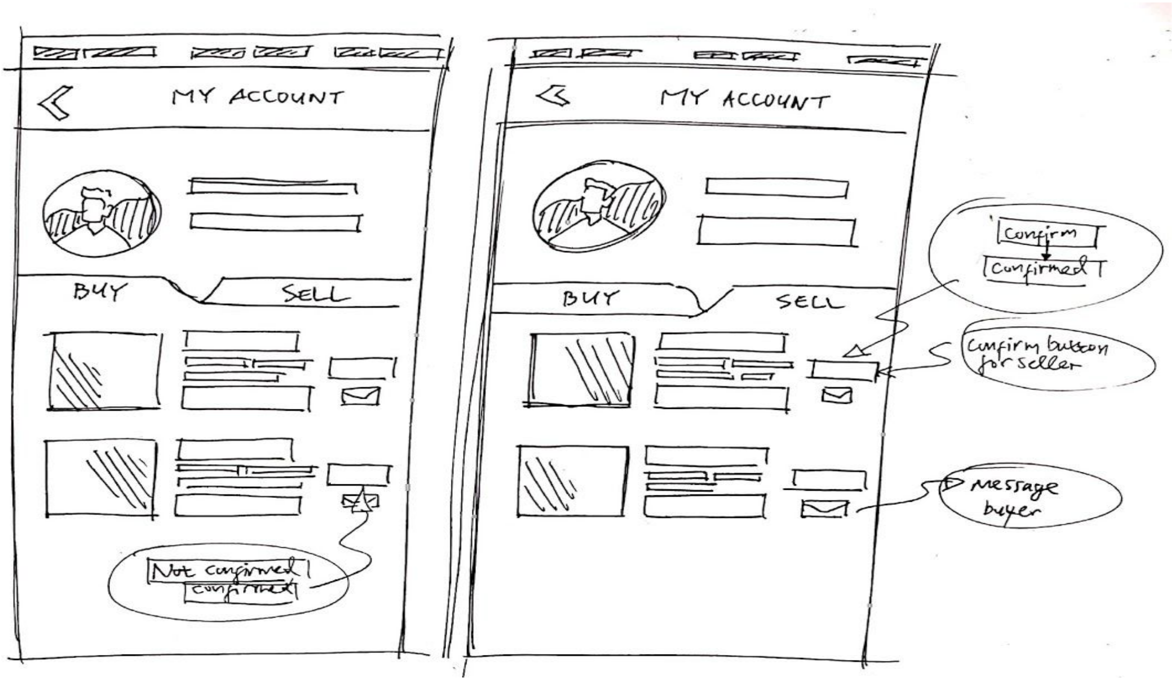
3) Item Buying:

- Select the ITEM icon above the search bar. Items will appear in order of most recent. Click on desired item.
- Select BUY NOW to proceed with order.
- Confirm whether the provided information is correct and press BUY to confirm order. If cancellation is desired, press CANCEL.
- The TRANSACTION COMPLETE pop up will appear to notify a successful order.
- The item will be added to the user's account under the BUY section.



4) Service Buying

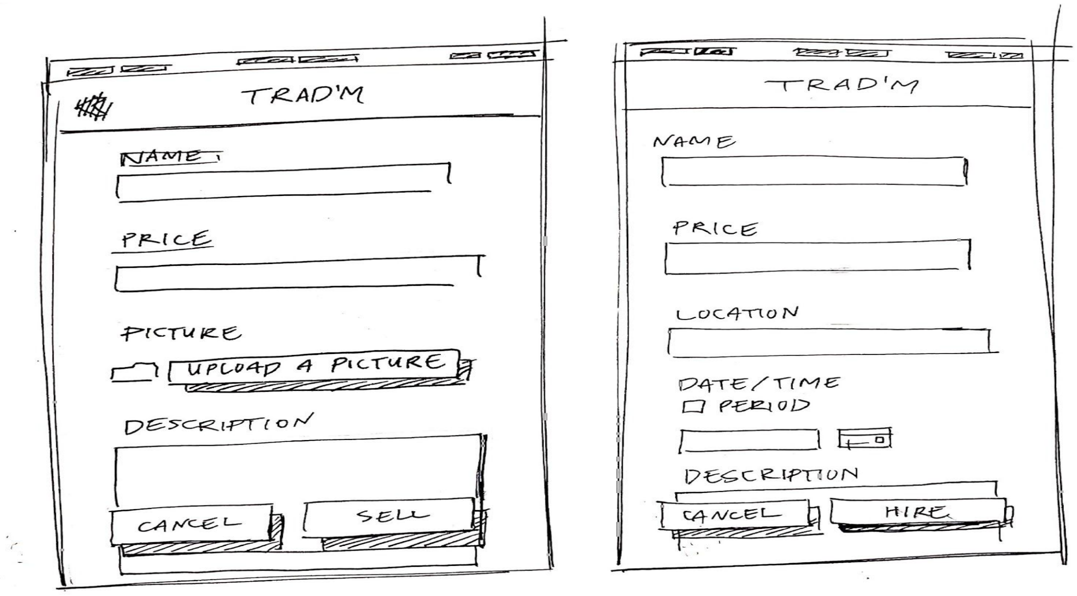
- Select the SERVICE icon above the search bar. Items will appear in order of most recent. Click on desired service.
- Select HIRE NOW to apply for the job.
- Confirm whether the provided information is correct and press BUY confirm order. If process cancellation is desired, click CANCEL.
- The TRANSACTION COMPLETE pop up will appear to notify a successful order.
- The item will be added to the user's account under the BUY section.



5) Confirmation Button

- From the main page, click on the account button on the upper right corner.
- Under the sell section. For users who are selling items, if there was a successful order, the icon on the right of the item will be NOT CONFIRMED. Pressing it will turn it to CONFIRMED. On the buyer's side, the item bought will be put on the

6) Search function (Service and Items)
- Select the category icons above the search bar.
- Type desired item in the search bar and press enter.
- Item will appear in a list in alphabetical order



7) Item offer creation and edit
- Press + button on the right bottom part of the main screen. Select desired item to sell.
- Enter the required information and press SELL.
- Item will be added to the user's account under the SELL section.

8) Service offer Creation and edit
- Press + button on the right bottom part of the main screen. Select desired item to sell.
- Enter the required information and press HIRE.
- Item will be added to the user's account under the SELL section.

5 - Acceptance Criteria

- All mandatory fields must be completed before a user can submit any form. This includes account creation, offer creation, offer edit…
- Information from the forms ends up stored in the correct databases. For account creation, this implies the account data is stored in the user details database, and for offer creation and edit, this implies the account data is stored in the offer marketplace database…
- Both sellers and buyer should be able to verify the accuracy of the order.
- System should accept correct information during account creation and login and reject incorrect information.
- An acknowledgement email is sent to the user upon the completion of the account creation process.
- Registered users should be able to login using the credentials they entered during account registration
- Users must be logged in before they can use view, search, buy, or sell products and services.
- Users should be able to view all available offers in the marketplace. Furthermore, the user must be able to further view the details of offers by clicking on them.
- Users should be able to input search criteria and then view the offers matching the criteria.
- Users should be able to create offers for goods and services after inputting all the required details about said offer. The offer should then be viewable through the marketplace and search function.
- An acknowledgement email is sent to the user upon the completion of the offer creation process

- Users should be able to edit their previously created offers by changing details of the offers should they want to. The offers should then be updated to reflect the changes made.
- An acknowledgement email is sent to the user upon the completion of the offer edit process.
- Users should be able to buy offers for goods and services sold by other users by clicking on the purchase button.
- An acknowledgement email is sent to both buyer and seller of the offer after the buyer request to purchase an offer of the seller's.
- After the buyer request to purchase a seller's offer, the seller should be prompted with a button either confirming and finalizing the purchase or rejecting it.
- After the seller responds to the confirm button prompt, an acknowledgement email is sent to both buyer and seller of the offer informing them of the results, be it a completion of said offer or rejection of it.
- All buttons and links should function correctly.
- Sensitive information such as password is hidden.
- App can correctly deal with incorrect, too large, too small, foreign (etc) input and should produce no errors.
- Users should be informed of any issues that may arise ex: if they have inputted incorrect or unacceptable data.

6 - Non-functional Requirements

1) **Usability:** The whole process of searching and buying should be streamlined from start to finish. Furthermore, the gui should be intuitive and easy to read and understand. Buttons should clearly show their functions, drop-down menus should be easy to use, and text should be easy to read.
2) **Reliability:** The user's input should be acknowledged as soon as it is input. For example, as soon as the user is done selecting and handling the item, there should be a confirmation of the user purchase via transaction confirmation sent to both parties, the seller and the buyer, and a redirection to a chatbox to discuss the handover plans with the seller. System should also make sure to choose the correct item and seller chosen by the buyer.
3) **Scalability:** Application should be able to handle massive increases in student and item data without delay by optimizing data storage design and access. It should also allow for the addition of more locations to match increased usership.
4) **Security:** User info such as username, ID, password, personal contact, wishlist and chat history should be should be protected and should not be accessible to unauthorised personals and also there should not be a way for user to manipulate the application for their gain or bypass necessary means.
5) **Performance:** Less than 3 seconds to execute most search, chat, and filter functions, handle input and open the app.If the execution time required is too long (more than 3 seconds), small animations or prompts need to be added to optimize the user's waiting experience so that users can easily receive longer latencies.
6) **Android compatibility:** Our app should be usable across all Android devices.
7) **Documentation:** System will provide documentation to inform users of system functionality and any change to the system.For novices,they can find instructions of the application.
8) **Responsiveness:** Application should be responsive to the user's Input or to any external interrupt of high priority and return back to the same state before the interrupt ,be it external or internal, occurred. App should allow the user to easily pick the required items or enter a specific search as well as specify the conditions of the item without any hassle.
9) **Integrability:** The system should be able to be easily integrated in the future if the user base and location base expands.
10) **Availability:** Users should be aware of the existence of our app and be able to install the app from some publicly available platform such as play store. Users should also be able to rate the app and contact the necessary people via the app.
11) **Network coverage:** App should work with Wi-Fi or mobile networks and be able to handle slow connections. App should be able to look for a Wi-Fi connection and if not available then look for a mobile network.

12) **Traceability:** Users can view previous browsing or transaction records,and return to the previous interface or view details through the corresponding buttons.
13) **User Interface and Human Factors:** The app is mainly targeting international students in KAIST but could be scaled in the future as the application grows to be available for use by anyone looking for a good or service. Users will require no training to use our app, and our GUI should help our app's usability.
14) **Documentation:** Documentation should include all the functionalities, behaviors and designs (including GUI designs) in the app. It should also include possible errors, conditions, examples (list to be updated as app evolves). Documentation is intended for programmers, project managers, designers, users and testers.
15) **Error Handling and Extreme Conditions:** App will attempt to avoid critical errors as much as possible and may have catches for certain exceptions, but in the case of an error occuring the app will first try to correct the error and if it can't then restart the app. Certain constraints may also be in place to prevent things that may cause an error such as user input being limited to a certain range.
16) **System Interfacing:** App will use Android device's default keyboard and screen for input and output respectively.
17) **Quality Issues:** The system does not have to go out of its way to catch faults, usually a restart will be applied, and if the issue persists the user can send an error report to the developers. The system should however be portable across all Android devices and be back compatible up to Android 7.0.
18) **System Modifications:** Database upgrade and expansion. We expect that as more users start using the app, we would have to expand the database to allow for more information to be stored.
19) **Security Issues:** Access to both data and system should be restricted; the user should only be able to access his/her information, and users should only be able to access the system after registration and login. System could be automatically backed up weekly since the database would be virtual and external. For the same reason, physical security won't be an issue.
20) **Resources and Management Issues:** This app is to be built and maintained by the members of group 7 in the CS350 course at KAIST Fall semester 2019. The deadline for the development is the same as the deadline for the project. To build and maintain the app, the group members will rely on a mobile app development framework for Android and will be using Java as the programming language. As such the group members must have decent knowledge of these skills.

7 - Acknowledgements

7.1 - Kern Fowler

Title Page / Contents Page / Project Introduction (All Sections) / Overall Description (2.1 - Product Perspective) / System Features (3.3 - Domain Model)

7.2 - Nabila Sindi

Overall Description (2.3 - Operating Environment) / System Features (3.2.3 - Create New Account / 3.4.2 - Create New Account) / Preliminary User Manual

7.3 - Mohamed Almaazmi

Project Introduction (1.3 - Team Vision Statement) / Overall Description (2.4 - Assumptions and Dependencies) / System Features (3.1 - Functional Requirements / Use Case Diagram/ 3.2.2 - Create New Offer / 3.4.1 - Create New Offer) / Non-functional Requirements/ Acceptance Criteria

7.4 - Zihan Qi

Overall Description (2.2 - Product Features) / System Features (3.2.4 - Create Purchase Request / 3.4.3 - Create Purchase Request

7.5 - Editor

This document was formatted and edited by Kern Fowler.

7.6 - References

No reference material was used in the creation of this document.

7.7 - GitHub

This document is to coincide with work done on the following GitHub project - https://github.com/el17kjtf/CS350_Group_7.git. More information about this project and its code can be found there, as well as updates and change logs of the report.