TAYLOR'S UNIVERSITY
Wisdom · Integrity · Excellence

ITS66404 Software Engineering
ASSIGNMENT 2
Project Based Learning (PBL) with Purpose Learning

HAND OUT DATE: 14 October 2022 (Friday)
HAND IN DATE: 18 November 2022 (Friday)
WEIGHTAGE: 30%

Instructions to students:

- The assignment should be attempted in groups of 4-5 students.
- Complete this cover sheet and attach it to your assignment – this should be your first page.

Student declaration:	
I declare that: • ***I understand what is meant by plagiarism*** • ***The implication of plagiarism have been explained to us by our lecturer*** ***This project is all our work and I have acknowledged any use of the published or unpublished works of other people.***	
Names of Group Members	
Name	**Student ID**
Ishihara Satoaki	0354208
Aman Mahmud Bin Amer Mahmud	0348725
Mohamed Fahad Farhan	0354487
Mohammad Sameed Khan	0353846
Thua Sin Wei	0354566
Jinan Nisar	0354690

Avoid copy and paste job in your report and it is considered as plagiarism. Plagiarism in all forms is forbidden. Students who submit plagiarised assignment will deserved a 0 marks.

Table of contents

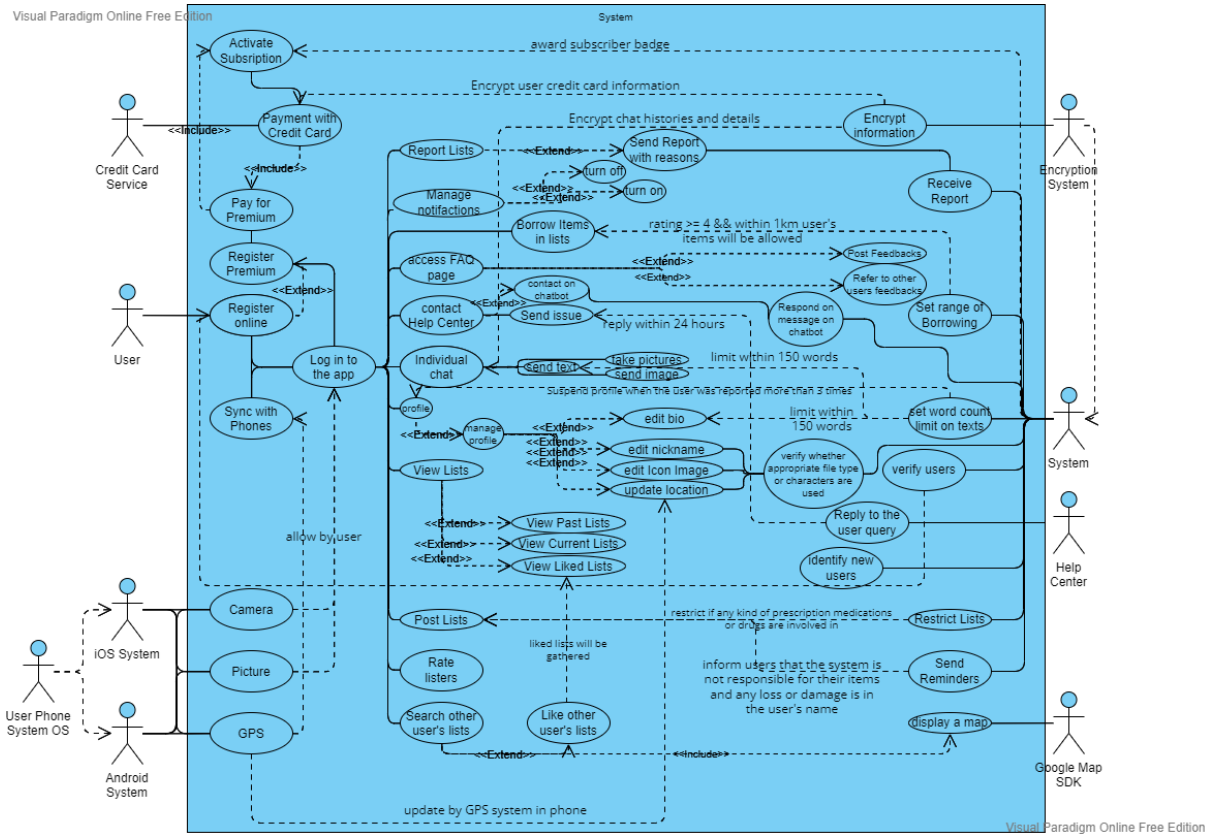
Introduction

Our group is led by Ishihara Satoaki -leader of this group-, and other members are Mohammad Fahad Farhan, Mohammad Sameed Khan, Jinan Nisar, Aman Mahmud Bin Amer Mahmud, and Thua Sin Wei.

Our team is working to minimise the impact of waste on the environment by integrating technological pursuits in order to make a difference, under the vision of "**Create a sustainable ecosystem**". In assignment 1, we have proposed the ConserveR app which allows users to give away or lend their goods for a specific time-frame, with UML use case diagram, class diagram, and 4 object diagrams. We have also defined functional requirements, non-functional requirements, and scope definitions.

Assignment 2 is the extension of assignment 1. In addition to delving into the predicted cases by conducting simulations in detail, we actually made a prototype and reached the point where it is advertised via presentation in class. We have brushed up our use case diagram and class diagram as modified diagrams, added descriptions and specifications on some of use case diagrams. Modified use case diagrams, modified class diagrams, three UML sequence diagrams, activity diagrams, and state chart diagrams are used in this assignment. Actually interfaces are also created, and we will propose our software via physical presentation. The project will be completed to the point of presenting the software developed to achieve the goals that have been set for some time.

Modified Use Case Diagram

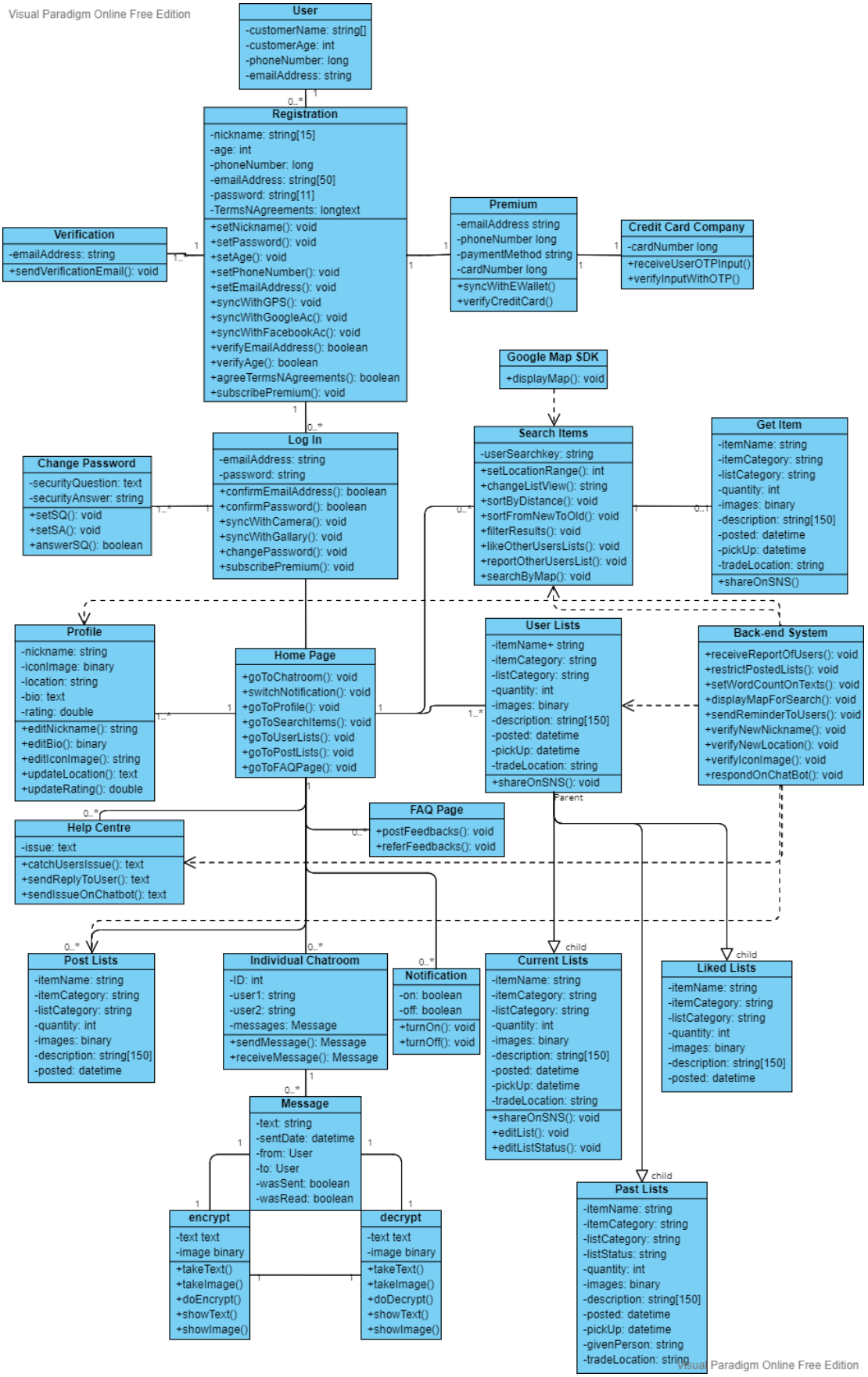


In this use case, three attributes have been added which are “verify whether appropriate file type or character is used” “contact on chatbot” and “respond on message on chatbot”. The first one is the attribute for System that verifies the user’s profile details to approve.

Now, the system can verify not only the word count of bio, but also if the appropriate file type is used for the icon image, or if there are any special characters used in the location. If unsupported file types are used for the new icon image, and unsupported special characters are used in the new location, then the system can deny the approval, request the user to use the acceptable characters or file types for the new profile details.

Also, the user not only needs to send an issue via the forum in the Help Centre, but the user can directly chat with the system on chatbot. When the user writes an issue and sends it to the chatbot, the system will receive it and will send the reply as soon as possible.

Modified++++++++++0 Class Diagram



In the class diagram, five new operations have been added in the Back end System and Help Centre. They are +verifyNewNickname(): void -to verify the new nickname user typed in is using only supported characters but unsupported special characters- , +verifyNewLocation(): void -to verify the new location user typed in is using only supported characters but unsupported special characters- , +verifyIconImage(): void -to verify the new icon image user has chosen is the supported file type but unsupported file type-, and +respondOnChatBot(): void -to send reply on issue sent by user through the chatbot-. +sendIssueOnChatbot(): text -to send the issue to the backend system via chatbot- was added in Help Centre class. With the three new verification operations, and +setWordCountOnTexts(): void will be verifying changes made on user profile, check the suitability of alterations and final application. For those two chatbot-related operations will work on communication between user and system.

Use Case Descriptions and Specifications

1. **Profile**

Use Case Name	Basic Flow @ Happy Path	Alternate Flow @ Alternate Path	Exception Flow @ Exception Pathway
Profile	1. User clicks the "Manage Profile" button. 2. User chooses which elements in the profile to change/edit. 3. User decides to change the nickname. 4. System checks whether the new profile details are approvable. 5. User Press "save" to apply the change on profile. Use case success.	1. User clicks the "Manage Profile" button. 2. User chooses which elements in the profile to change/edit. 3. User changes the nickname and saves it. [A1: User changes the bio and saves it.] 4. User decides to change the nickname. [A2: User changes the icon and saves it.] 5. User decides to change the nickname. [A3: User changes the location and saves it.] 6. System checks whether the new profile details are approvable. 7. User Press "save" to apply the change on profile. Use case success. **Alternative Flows:**	1. User clicks the "Manage Profile" button. 2. User chooses which elements in the profile to change/edit. 3. System checks whether the new profile details are approvable. Use case failed. **Exception Flows:** E1: System confirmed unacceptable format of files or characters in the new details. 1. System denies the approval and sends errors. 2. The error message will be displayed to the user and the user cannot save the change. E2: System confirmed that the word count of the bio exceeds 150 words.

		A1: User changes the bio and saves it. A2: User changes the icon and saves it. A3: User changes the location and saves it.	3. System denies the approval and sends errors. 4. The error message will be displayed to the user and the user cannot save the change.

Use Case Specifications

Name: Profile

Description: Profile use case describes the process of a user managing profiles with editing icon, nickname, location, or bio.

Author(s): Ishihara Satoaki

Actor(s): User, System

Location(s): malaysia

Status: Success

Priority: Medium

Assumption(s): User knows how to change the nickname.

User knows how to change the bio.

User knows how to change the icon image.

User knows how to update the location.

Precondition(s): User must be logged into the system as an existing user. User must have agreed to the ToS of the app. User must allow the app to access their gallery.

Postcondition(s): User will see the updated new profile details on the screen.

2. Post Lists

Use Case Name	Basic Flow @ Happy Path	Alternate Flow @ Alternate Path	Exception Flow @ Exception Pathway
Post List	1. User chooses my listings option from the hamburger menu. 2. User adds a list and chooses the giveaway option. 3. User selects category of the listed item. 4. User adds title and description of the listing. 5. User chooses to add at least five photos of the item being listed. 6. System provides with the option to choose from library or take photo. 7. System checks the photos for suspicious content and confirms adding photos to the listing. 8. User specifies the quantity of the item being listed. 9. User specifies the pickup time. 10. User adds pick up location manually.	<u>Basic Flow:</u> 1. User chooses my listings option from the hamburger menu. 2. User adds a list and chooses the giveaway option. [A1: User chooses the borrow category instead] <u>Select the category of item.</u> 5. User chooses to add at least five photos of the item being listed. 6. System provides with the option to choose from library or take photo. 7. System checks the photos for suspicious content and confirms adding photos to the listing. 8. User specifies the quantity of the item being listed. 9. User specifies the pickup time. 10. User adds pick up location manually. [A2:	<u>Basic Flow:</u> 1. User chooses my listings option from the hamburger menu. 2. User adds a list and chooses the giveaway option. 3. User selects category of the listed item. 4. User adds title and description of the listing. 5. User chooses to add at least five photos of the item being listed. 6. System provides with the option to choose from library or take photo. 7. System checks the photos for suspicious content and confirms adding photos to the listing. [E1: Suspicious items detected in the photos] The use case fails. **Exception Flows:** E1: Suspicious items detected in the photos.

	11. System checks if the location is valid. 12. User submits the listing. Use case succeeds.	User chooses map option instead.] <u>Listing is submitted.</u> 11. User submits the listing. Use case succeeds. **Alternative Flows:** A1: User chooses the borrow category instead. 1. User adds a list and chooses the borrow option. The use case continues at <u>Select the category of item</u> in basic flow. A2: User chooses map option instead. 1. User adds pick up location through the map. 2. System doesn't validate location since all locations on map are pre-validated. The use case continues at <u>Listing is submitted</u> in basic flow.	1. System suspends the adding list activity. 2. The system informs the user about the suspicion detected and warns them against violating ToS.

Use Case Specifications

Name: Post List

Description: Post List use case describes the process of a user adding a listing on the app under either the giveaway or borrow category.

Author(s): Ishihara Satoaki

Actor(s): User, System

Location(s): Malaysia

Status: Success.

Priority: High.

Assumption(s):

User has a valid item to donate or share.

User knows how to access the hamburger menu.

User knows the category their listing belongs to.

User knows their pickup location.

Precondition(s): User must be logged into the system as an existing user. User must have agreed to the ToS of the app. User must have their location open to be detected by GPS. User must allow the app to access their camera.

Postcondition(s): User will see “List posted” prompt displayed on screen immediately after completion of the use case. User will see a list in the current list category of “My lists” in the main menu.

3. Search Lists

Use Case Name	Basic Flow @ Happy Path	Alternate Flow @ Alternate Path	Exception Flow @ Exception Pathway
Search List	1. By clicking the search symbol, the user may search the list. 2. The user can select the item category. (All, Borrow, and Donate) 3. To find the lists, type the name of the item. 4. The system will filter out the items that the user searches for that are within their area and in the category that the user selects. 5. The user can select out objects that are edible or inedible. 6. The system will filter the items based on the user's preferences. 7. The user will be presented with a selection of options from which to pick.	1. By clicking the search symbol, the user may search the list. 2. The user can select the item category. (All, Borrow, and Donate) <u>Search items</u> 3. To find the lists, type the name of the item. 4. To find the lists, type the name of the item. [A1: User uses voice typing instead] 5. The system will filter out the items that the user searches for that are within their area and in the category that the user selects. 6. The user can select out objects that are edible or inedible. 7. The system will filter the items based on the user's preferences. 8. The user will be presented with a selection of options from which to pick.	1. Not able to locate any of the items that have been typed in by the users in the particular area. 2. There were no results found for the user's search. 3. Trying to search without entering any keyword.

		A1: User uses voice typing instead 1. User chooses voice typing for search by clicking on the microphone icon near the search bar. 2. User speaks the item name and the system processes the speech output to display results.	

Use Case Specifications

Name: Search List

Description: Search List use case describes the process of users searching the items and sorting the category of items that near their area.

Author(s): Ishihara Satoaki

Actor(s): User, System

Location(s): Malaysia

Status: Success.

Priority: High

Assumption(s):
User knows how to search for the items they want.
User knows how to sort the category of the items.

Precondition(s): User must be logged into the system as an existing user. User must have agreed to the ToS of the app. User must have their location open to be detected by GPS. User must allow the app to access their camera.

Postcondition(s): User will see the screen involves a lot of lists have been gathered due to the keywords of images that the user has posted.

Interface Design Proposal

The user interface of the application follows a minimalistic theme with enough illustrations to enhance user experience and strengthen the emotional and aesthetic appeal of the app. It makes use of menu selection interaction style to simplify navigating processes and minimise errors. On launch, the app features a splash screen that displays the app logo for three seconds to attract a user's interest. (Fig 1) Once the app is launched completely, the app presents the user with an attractive onboarding screen displaying the app's motto, where the user can register or log in. (Fig 2) The app features a simplistic home screen which facilitates ease of access to all the app's functions and enhances app usability. (Fig 3) The user can see listings nearby and post a list directly from the home screen, without opening the menu.

Fig 1: Splash Screen

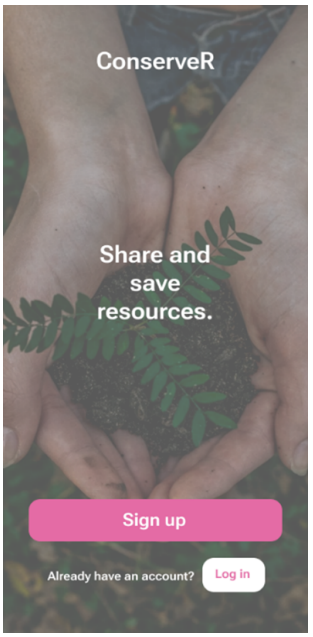


Fig 2: Onboarding Screen

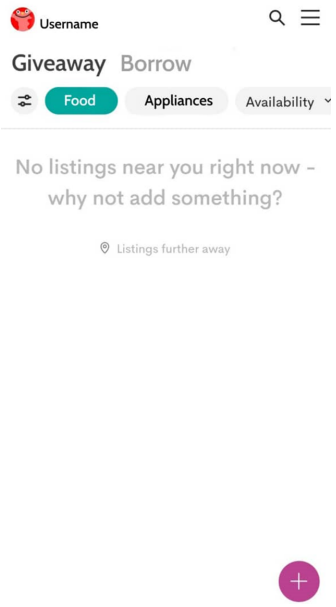
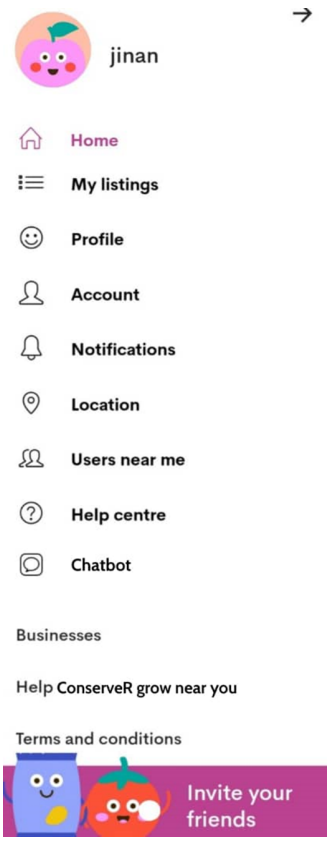
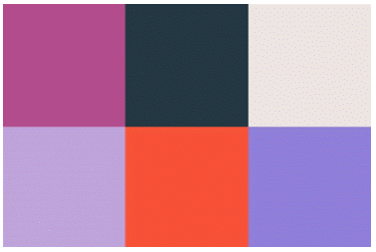


Fig 3: Home Screen

The app features a useful hamburger menu that displays all the functions offered by the platform besides ways to share and support. The menu includes colourful illustrations to further attract users.

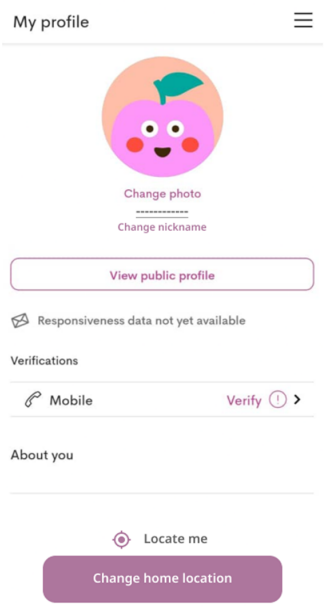


The app uses a variety of attractive colours as shown in the colour pallete and mostly uses minimalistic fonts to give the app a simplistic look.



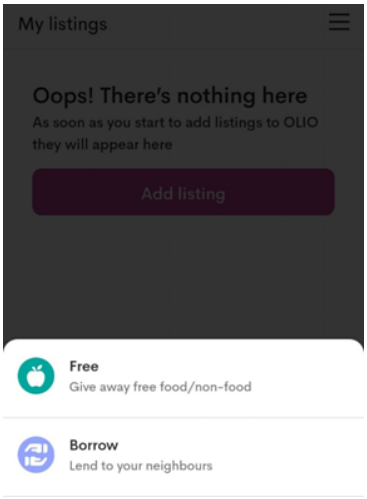
Profile:

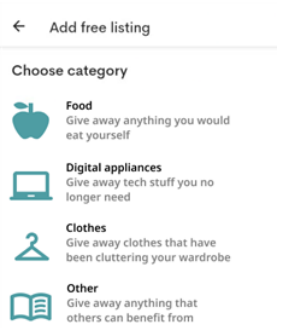
A user can easily edit their profile by selecting the profile option from the hamburger menu. On selection, user would be displayed their current profile along with options to customise and change it. The app would offer default fruit themed icons besides giving the user the option to add their own image. User can view how their public profile looks to others and also add a short description about themselves.

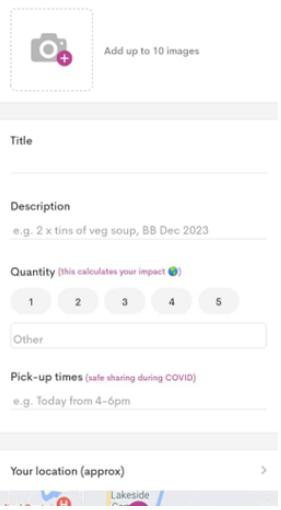


Post List:

Users can post lists belonging to various categories offered by the app. Each category is accompanied by an icon to enhance the visual attractiveness as well as the ease of access of the app. The process of adding a list is kept simple with minimal options on the post list screen which displays all necessary steps on one page.



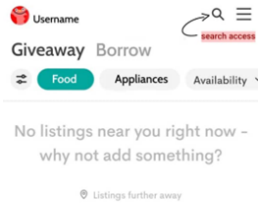


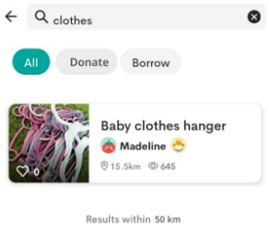


1. User selects listing type. 2. User selects the listing category. 3. User posts the list.

Search List:

The users can search the listings near them by clicking on the search button conveniently placed on the home screen. All listings are shown along with a picture to help give the users a better idea of the item listed.





Test Plan

1. **Case 1 (Profile)**

Use Case	Test Case	Description	Procedure	Expected Results
UC03: Profile List	TC01-01: Managing profile	Users are entering the profile tab to change the nickname displayed on profile and the photo of the profile.	• The user will click the "Manage Profile" button • The system will open the profile tab • The users will observe various profile altering and visiting options such as: -view public profile -change nickname -change profile photo -verify phone number -your description	Once the user clicks the "Manage Profile" button, it will proceed to the profile tab where users have the option to view public profile, change location,change nickname, change phone number, change profile photo, verify phone number and about your description.
	TC01-02: Viewing public profile	Users will check their public profile to view their profile information	• The user will click "View Public Profile" button • The system will display: -profile photo -phone number -the location of the user -profile nickname	Once the user clicks the "View Public Profile", it will display the their public information with their nickname, profile photo, phone number and the location of the user
	TC01-03:	Users will be	• The user will	Once the user

	Change Nickname and Profile Photo	able to change their profile photo and profile nickname in the “Change Nickname” and “Change Photo” button	click “Change Nickname” and “Change Photo” button ● The user will input their nickname and profile photo ● The user will save their profile photo and their nickname by clicking the save button ● The user’s profile photo and their nickname will be display in the profile tab or view public profile	changes their nickname and profile photo, a new nickname and picture is displayed on their profile screen when visited as well as on the left hand top corner of the homepage.
	TC01-04: Verifying phone number	Users will need to verify their phone number to authenticate themselves and secure their engagements on the app.	● The user will click the “Verify” button ● The system will call the user’s phone number using the 2FA system or known as two-factor authentication system ● The user will input the code from the 2FA system into the the app and clicking the “Verify” button ● The user will	Once the user clicks the “Verify” button, the system will call the user’s phone number for the verification code. Therefore, the system will verify the user’s own phone number granting the security verification. Once verified, the option is no longer visible in the profile section.

			have the authenticity once the verifying code is matched	
	TC01-05: Change home address and current location	Users are able to change their home address and their own current location	• The user will click the “Change Home Location” or “Locate Me” button • The user will input the their coordinates or turning on their GPS signal • The user’s location will be display on the profile tab or view public profile	The user’s home address and location is changed once the user sets their GPS signal or inputs their coordinate of the map. The location is displayed on their profile when viewed.

2. **Case 2 (Search Lists)**

Use Case	Test Case	Description	Procedure	Expected Results
	TC02-01 User is presented with the option to filter through a list	The user is presented with an array of options to filter through posts that users have set for give away such as by distance, condition of product and the borrowing period.	- User clicks on the funnel button located at the search bar. - The user is then met with options to select the category to filter. - The user then proceeds to select the necessary criteria which is checkboxed to help filter the criteria.	The user is allowed to click on the filter button located at the top of the lists main page which has a funnel icon. This allows the user to checkmark on necessary conditions that need to be met before purchasing an intended product.
	TC02-02 User is presented with an option to search using the speech option.	This is a feature which can be integrated with google's voice assistant in order to search for products across the ConserveR app.	- The user clicks on the microphone button located on the search bar. - This then prompts the apps server to connect to google's voice assistant via a SDK. - This is then matched with the words the user speaks to find the given product,.	This feature allows users to search for goods within the ConserveR app using Google's voice assistant which will be added as a sdk onto the app's platform to deliver greater search optimization.

	TC02-03 User is presented with an option to search for an item using an image.	This feature integrates google lens into the app so that users can search for the product by just capturing an image of an object they wish to purchase.	-User clicks on the camera icon located at the search bar. -This then connects with the app google lens in order to use AI tracking to identify the object that the user captured. -This is then correlated with search results to find the users desired product.	This allows users to integrate artificial intelligence and algorithms in order to recgnize different objects and produce search results that match the given item in the app.
	TC02-04 The user is able to search for an item using text.	The user inputs text and this is then matched with items that belong in lists posts.	-User clicks on the search bar and types in the desired product. -The app then connects to the backends server and locates the product with it's description. - The server, then relays the information back to the user in the front end by listing all the product's related to the user's search.	The users inputted search item is matched with the conserveR back-end database in order to produce accurate search results that match the user description.
	TC02-05 The user is able to search by location.	This allows the user to help products located within close vicinity only.	-The user clicks on the location pin icon located next to the search navigation bar. - This connects with google maps SDK in order to track the longitude and latitude of the user's location. -This prompts the	This allows the user to find items within a 3km radius as this feature is useful for users who wish to travel less.

			app's system to relay back a response where item listings that are close to the user are displayed.	

3. Case 3 (Post Lists)

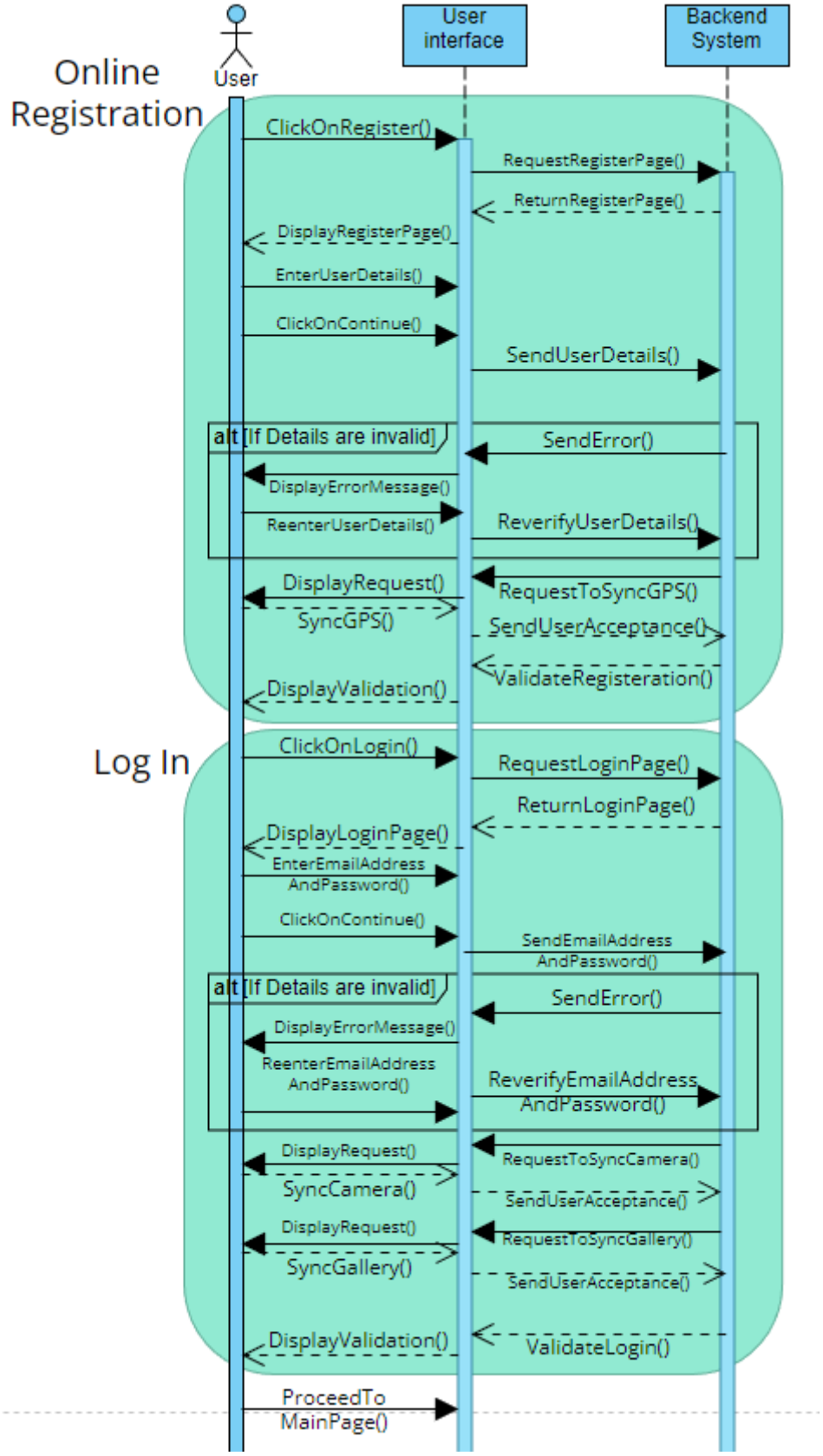
Use Case	Test Case	Description	Procedure	Expected Results
UC01: Post List	TC01-1: Giveaway list successfully posted.	User successfully posts a give away list by specifying the category and adding item description and pictures.	Follow UC01 normal flow (refer to use case description) with selecting giveaway category and choosing location manually.	“List posted” message is prompted on the screen. User is able to see the added list when choosing the “My lists” option from the menu under the giveaway category, indicating that the item has been successfully listed for giveaway or donation.
	TC01-2: Borrow list successfully posted.	User successfully posts a borrow list by specifying the category and borrowing duration and adding item description and pictures.	Follow UC01 alternate flow with selecting borrowing option instead of giveaway.	“List posted” message is prompted on the screen. User is able to see the added list when choosing the “My lists” option from the menu under the borrowing category, indicating that the item has been successfully listed for borrowing.

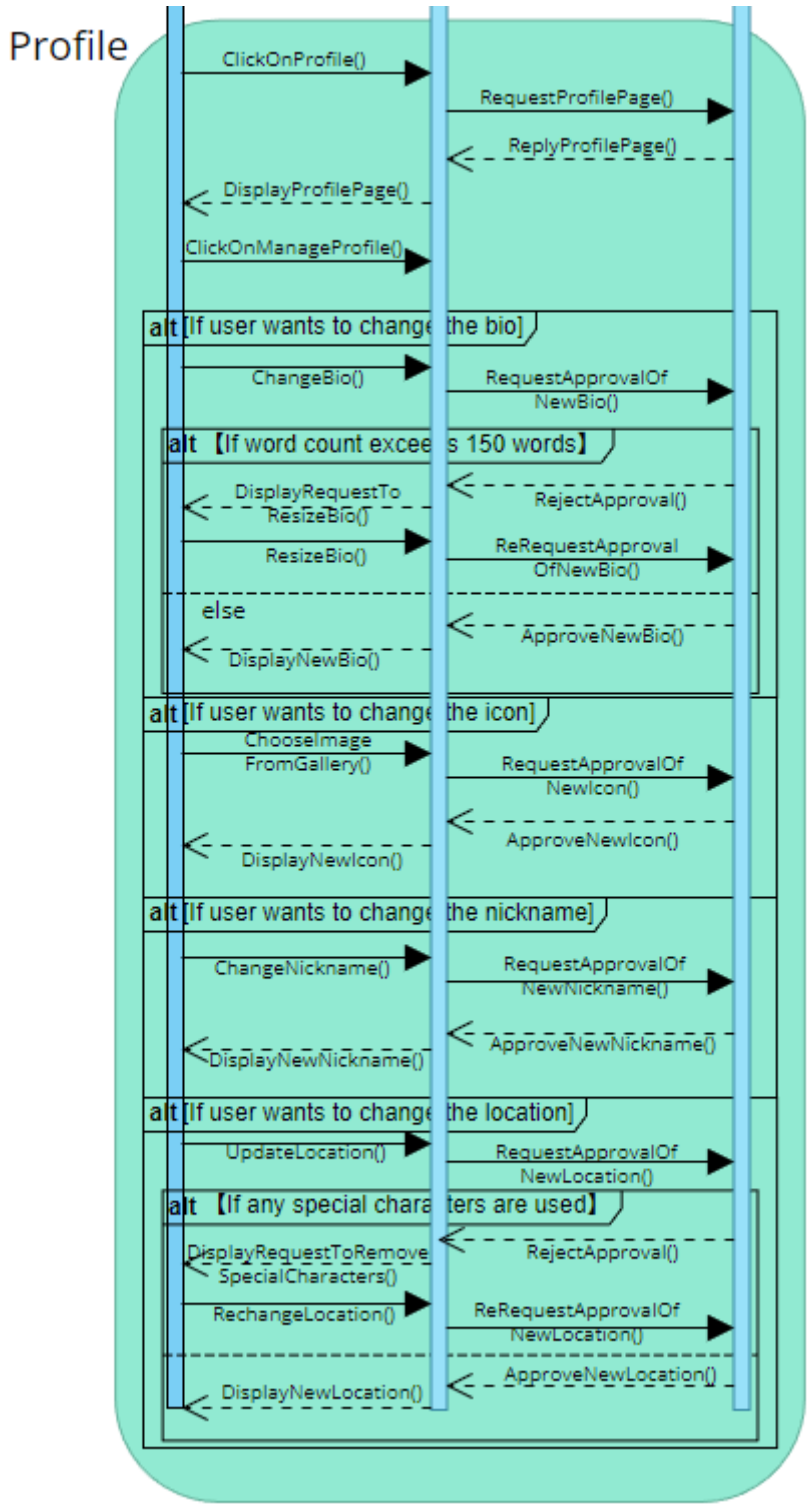
	TC01-3: Giveaway list posted with the help of map.	User successfully posts a give away list by choosing the location on a map instead of entering it manually.	Follow UC01 normal flow till event 9. - User chooses the “add location from map” option. - User adds the pickup location from the map by dragging the location pin on the preferred location. - Selected point is added as the pickup location. - User continues from event 12.	App’s map is displayed on the screen when the user selects the “add location through map” option. User is able to add location from the map. “List posted” message is prompted on the screen once the user adds location and clicks on “submit”.
	TC01-4: Photos added with camera.	User directly takes photos of the item through the app’s camera and automatically adds it to the list.	Follow UC01 normal flow till event 6. - User chooses the “Take photo” option and clicks 5 photos from the app’s camera. - The photos are automatically added after being clicked. - User continues from event 7.	App’s camera is opened when the user wants to take a photo. User is able to add photos directly from the app’s camera. Photos are added at the top of the listing with item description.

	TC01-5: Suspicious item post attempt made.	User attempts to post a listing which includes an item prohibited by the platform for legal and safety reasons like prescription drugs or firearms.	Follow UC01 normal flow till event 6. - User adds a prohibited item image. - System detects a suspicious item in the images posted. - Add list process is suspended.	User is unable to post the list. A warning message is prompted on the screen reminding user not to list any prohibited items.

Sequence Diagram

case 1





The first case of a sequence diagram is changing the profile. First, the user clicks the "Register" button (ClickOnRegister()) on the first page of the application. User interface receives the request from the user and then sends the request data packet of the register page to the backend system (RequestRegisterPage()). The application backend system will reply

with the data packet of the register page (ReplyRegisterPage()), and the user interface will receive it and create a register page, displayed for the user (DisplayRegistrationPage()).

User enters personal information such as nickname, age, phone number, email address, etc. (EnterUserDetails()) and submit by clicking “continue” (ClickOnContinue()). User interface receives that information and sends it to the backend system (SendUserDetails()), then the backend system will verify user information details. If the nickname or email address, phone number are invalid, the backend system will send an error to the user interface (SendError()) and the user interface will display an error message for the user (DisplayErrorMessage()). In this case, the user must re-enter information (ReenterUserDetails()) and submit again to the backend system via user interface (ReverifyUserDetails()). Otherwise, the backend system will request to sync the GPS of the user’s phone (RequestToSyncGPS()): user interface will display the request message (DisplayRequest()). The user must accept the sync if the user wants to proceed (SyncGPS()): the user interface will send the data of acceptance to the backend system (SendUserAcceptance()). When all of these are finished, the backend system will validate the user registration (ValidateRegistration()), user interface receive it and display validation for the user (DisplayValidation()).

After the registration, user press “Log In” (ClickOnLogin()) and user interface will request login page (RequestLoginPage()), backend system will return login page data packet (ReturnLoginPage()), therefore user interface will display login page (DisplayLoginPage()). Then, the user needs to input the email address and password already registered (EnterEmailAddressAndPassword()) and click “continue” to submit (ClickOnContinue()). User interface will send the data of user login information to the backend system (SendEmailAddressAndPassword()), and the backend system will verify that information. Except for the case that both email address and password are correct, the backend system will send an error to the user interface (SendError()) and the user interface will display an error message for the user (DisplayErrorMessage()). In this case, the user must re-enter information (ReenterUserDetails()) and submit again to the backend system via user interface (ReverifyUserDetails()). If both information is correct, the backend system will request to sync the Camera of the user’s phone (RequestToSyncCamera()): user interface will display the request message (DisplayRequest()). The user must accept the sync if the user wants to proceed (SyncCamera()): the user interface will send the data of acceptance to the backend system (SendUserAcceptance()). When all of these are finished, the backend system will require the user to sync the user phone’s gallery (RequestToSync). Lastly the backend

system will validate login (ValidateLogin()), and the user interface will display validation information for users (DisplayValidation()).

When both registration and login are completed appropriately, the user will automatically proceed to the main page (ProceedToMainPage()).

In the main page, the user will choose "Profile" (clickOnProfile()), and the user interface will request a profile page to the backend system (requestProfilePage()). The backend system will reply by sending a profile page (replyProfilePage()), the user interface will receive the data and display profile page for the user (displayProfilePage()).

Then, the user will click "Manage Profile" to edit a profile "clickOnManageProfile()):

1. If the user wants to change the bio, the user will re-write the bio and click "save" (changeBio()) and the user interface will request the backend system to approve the user's new bio (requestApprovalOfNewBio()). Nevertheless, if the word count of new bio exceeds 150 words, the backend system will reject the approval, send error (rejectApproval()): user interface will display a request to resize the word count of new bio to user (displayRequestToResizeBio()). User must re-write new bio within the word count and re-save it (resizeBio()), and when user finished resizeBio(), the user interface will re-request approval of the user's new bio (reRequestApprovalOfNewBio()). Then, if the word count is not exceeding a maximum of 150 words, the backend system will approve and apply the user's new bio, and the new bio will be displayed by the user interface (displayNewBio()).

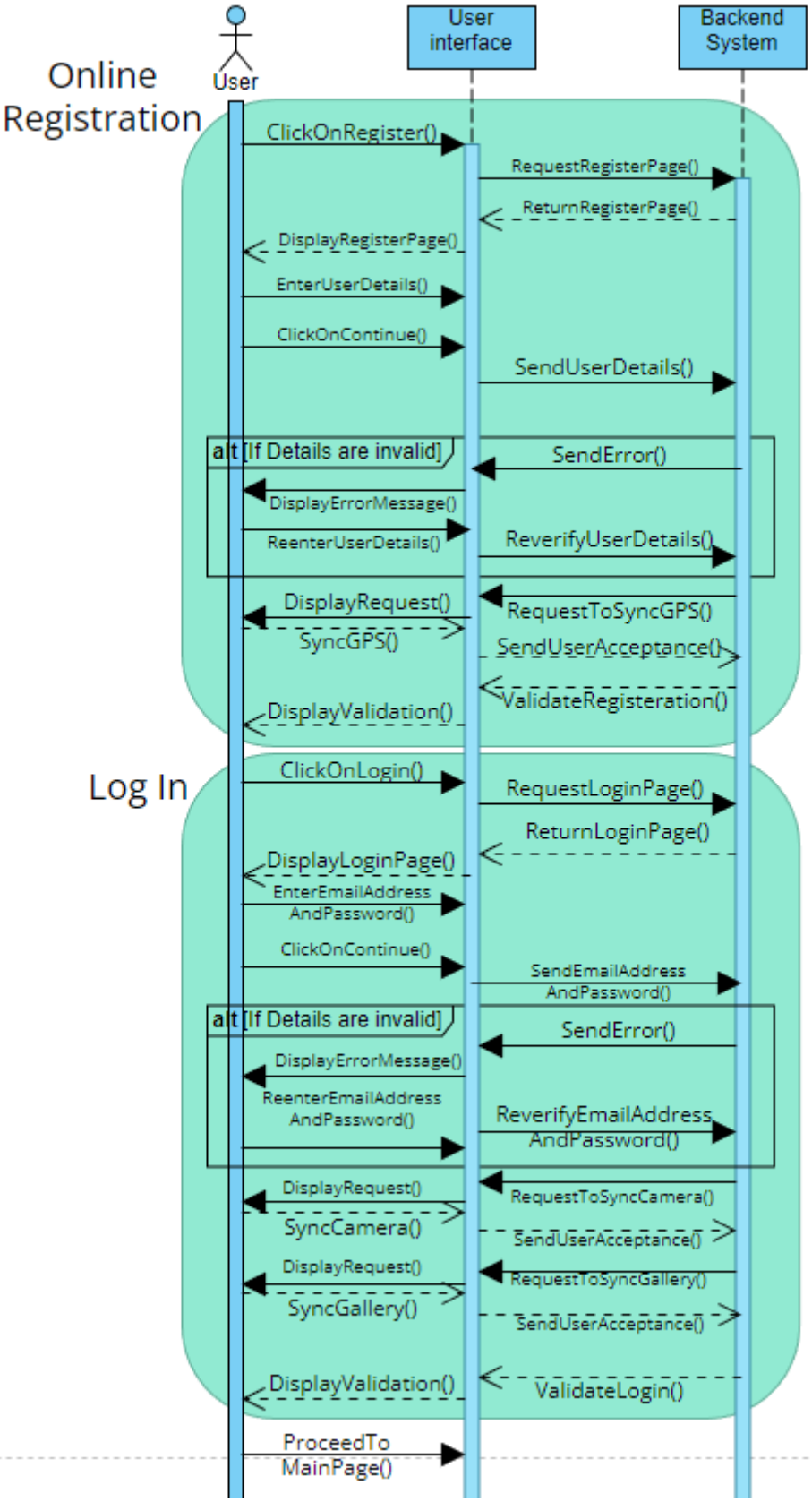
2. If the user wants to change the icon, the user will click "choose new image" on the position of the icon, select a new image from gallery to use (chooseImageFromGallery()). The user interface will request the backend system to approve a new icon (requestApprovalOfNewIcon()), and the backend system will approve and apply a new image to use for the icon (approveNewIcon()). Finally the new icon with the new image will be displayed by the user interface (displayNewIcon()).

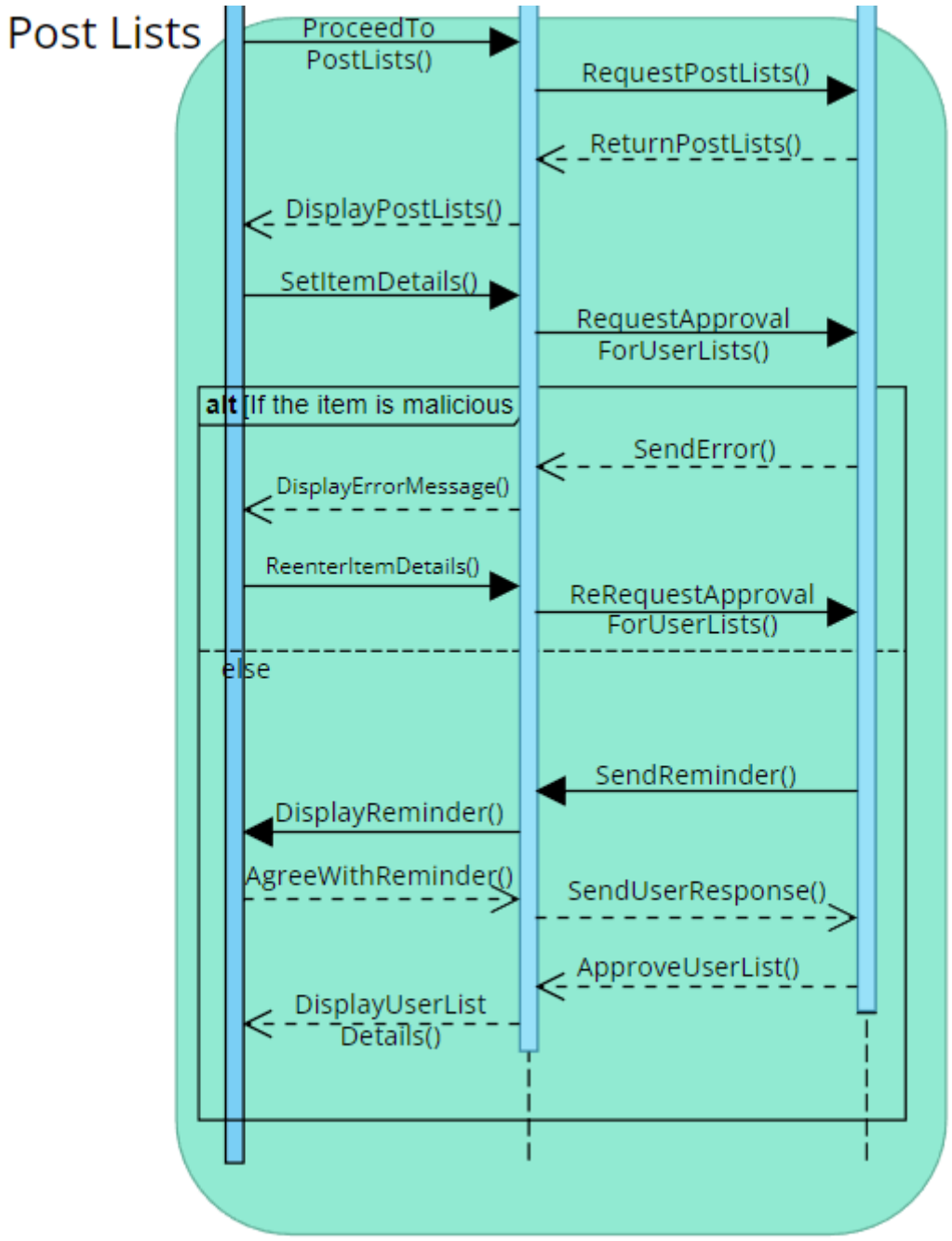
3. If the user wants to change the nickname, the user will re-write the nickname and click “save” (changeNickname()) and the user interface will request the backend system to approve the user’s new nickname (requestApprovalOfNewNickname()). Then the backend system will approve and apply a new nickname

(approveNewNickname()). Finally the new nickname will be displayed by the user interface (displayNewNickname()).

4. If the user wants to change the location, the user will re-write the nickname and click “save” (updateLocation()) and the user interface will request the backend system to approve the user’s new location update (requestApprovalOfNewLocation()). Nevertheless, if any special characters are used for location states, the request will be rejected (rejectApproval()), and the backend system will require the user to only use string characters and special characters are not admitted for location: user interface will display this request to user as an error message (displayRequestToRemoveSpecialCharacters()). User must rewrite the new location and save again (rechangeLocation()), and the user interface will request again to the backend system to approve (reRequestApprovalOfNewLocation()). If the location is only used text but does not involve any special characters, then the backend system will approve and apply new Location (approveNewLocation()). Eventually, the new location will be displayed by the user interface (displayNewLocation()).

case 2


Online Registration
User
User interface
Backend System
ClickOnRegister()
RequestRegisterPage()
ReturnRegisterPage()
DisplayRegisterPage()
EnterUserDetails()
ClickOnContinue()
SendUserDetails()
alt [If Details are invalid]
SendError()
DisplayErrorMessage()
ReenterUserDetails()
ReverifyUserDetails()
DisplayRequest()
SyncGPS()
RequestToSyncGPS()
SendUserAcceptance()
ValidateRegisteration()
DisplayValidation()
Log In
ClickOnLogin()
RequestLoginPage()
ReturnLoginPage()
DisplayLoginPage()
EnterEmailAddress AndPassword()
ClickOnContinue()
SendEmailAddress AndPassword()
alt [If Details are invalid]
SendError()
DisplayErrorMessage()
ReenterEmailAddress AndPassword()
ReverifyEmailAddress AndPassword()
DisplayRequest()
SyncCamera()
RequestToSyncCamera()
SendUserAcceptance()
DisplayRequest()
SyncGallery()
RequestToSyncGallery()
SendUserAcceptance()
DisplayValidation()
ValidateLogin()
ProceedTo MainPage()



The second case of a sequence diagram is posting lists. First, the user clicks the “Register” button (ClickOnRegister()) on the first page of the application. User interface receives the request from the user and then sends the request data packet of the register page to the backend system (RequestRegisterPage()). The application backend system will reply with the data packet of the register page (ReplyRegisterPage()), and the user interface will receive it and create a register page, displayed for the user (DisplayRegistrationPage()).

User enters personal information such as nickname, age, phone number, email address, etc. (EnterUserDetails()) and submit by clicking “continue” (ClickOnContinue()). User interface receives that information and sends it to the backend system (SendUserDetails()), then the backend system will verify user information details. If the nickname or email address, phone number are invalid, the backend system will send an error to the user interface (SendError()) and the user interface will display an error message for the user (DisplayErrorMessage()). In this case, the user must re-enter information (ReenterUserDetails()) and submit again to the backend system via user interface (ReverifyUserDetails()). Otherwise, the backend system will request to sync the GPS of the user’s phone (RequestToSyncGPS()): user interface will display the request message (DisplayRequest()). The user must accept the sync if the user wants to proceed (SyncGPS()): the user interface will send the data of acceptance to the backend system (SendUserAcceptance()). When all of these are finished, the backend system will validate the user registration (ValidateRegistration()), user interface receive it and display validation for the user (DisplayValidation()).

After the registration, user press “Log In” (ClickOnLogin()) and user interface will request login page (RequestLoginPage()), backend system will return login page data packet (ReturnLoginPage()), therefore user interface will display login page (DisplayLoginPage()). Then, the user needs to input the email address and password already registered (EnterEmailAddressAndPassword()) and click “continue” to submit (ClickOnContinue()). User interface will send the data of user login information to the backend system (SendEmailAddressAndPassword()), and the backend system will verify that information. Except for the case that both email address and password are correct, the backend system will send an error to the user interface (SendError()) and the user interface will display an error message for the user (DisplayErrorMessage()). In this case, the user must re-enter information (ReenterUserDetails()) and submit again to the backend system via user interface (ReverifyUserDetails()). If both information is correct, the backend system will request to sync the Camera of the user’s phone (RequestToSyncCamera()): user interface will display the request message (DisplayRequest()). The user must accept the sync if the user wants to proceed (SyncCamera()): the user interface will send the data of acceptance to the backend system (SendUserAcceptance()). When all of these are finished, the backend system will require the user to sync the user phone’s gallery (RequestToSync). Lastly the backend system will validate login (ValidateLogin()), and the user interface will display validation information for users (DisplayValidation()).

When both registration and login are completed appropriately, the user will automatically proceed to the main page (ProceedToMainPage()).

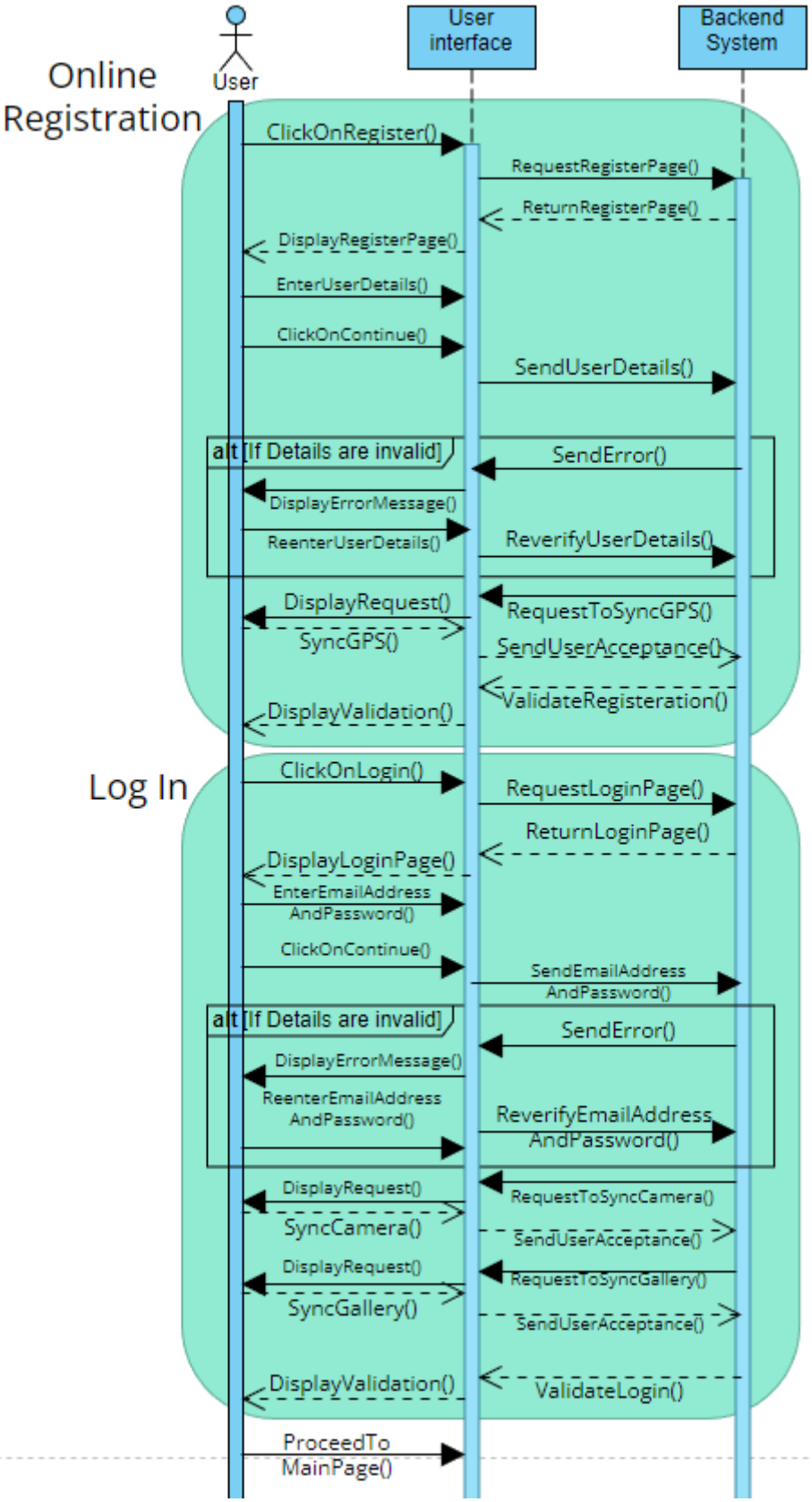
In the main page, the user will click “Post Lists” (ProceedToPostLists), and the user interface will request a “Post Lists” page to the backend system (requestPostLists()). The backend system will reply by sending a “Post Lists” page (ReturnPostLists()), the user interface will receive the data and display a “Post Lists” page for the user (displayPostLists()).

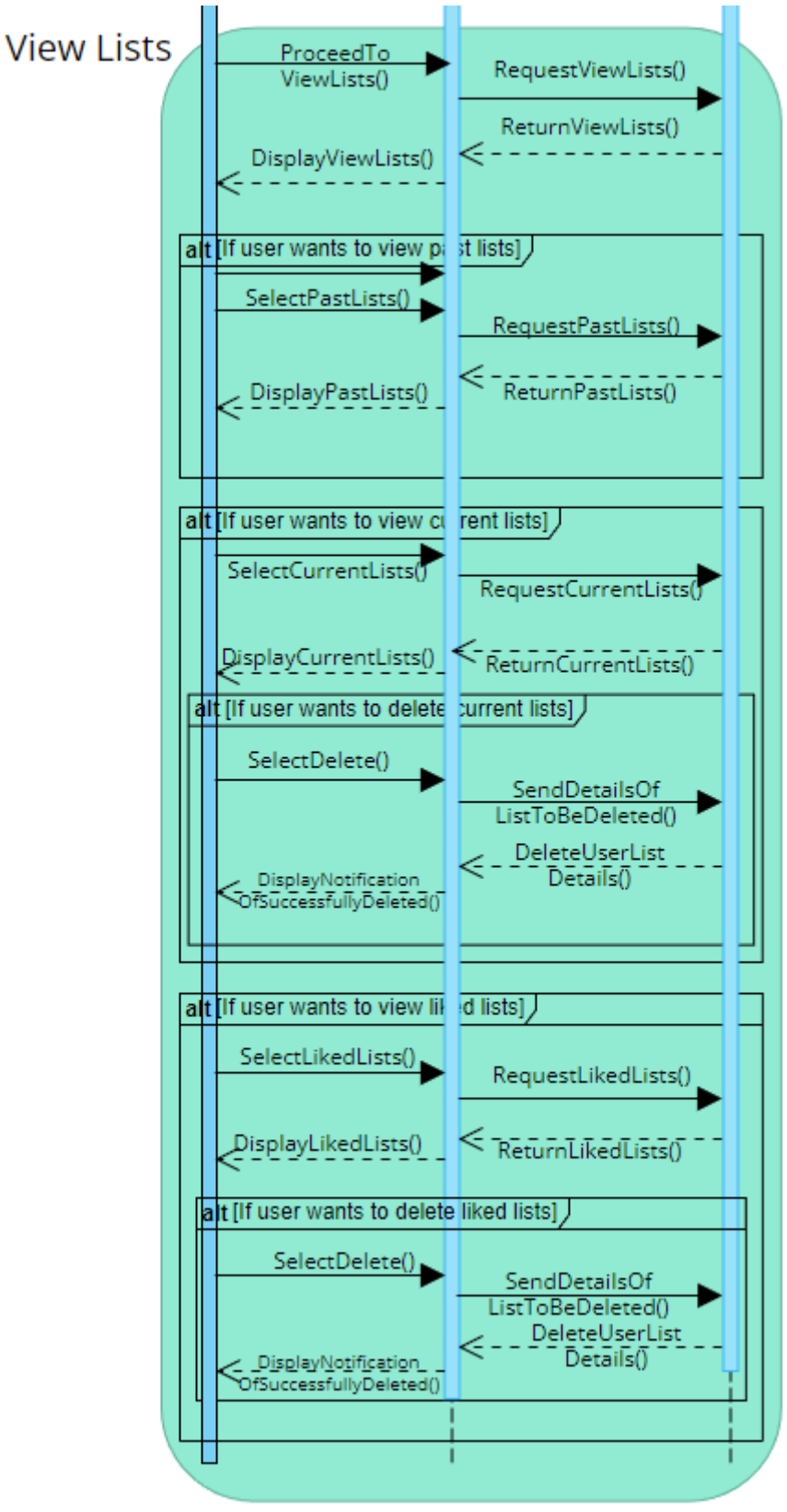
First, the user will set item details such as the name of items, quantities, etc. and click “Post” (SetItemDetails()). After the user inputs all item information, the user interface will request the backend system to approve the user to post a list with items (RequestApprovalForUserLists()):

- If any malicious items are found, the backend system will reject the list and send an error to the user (SendError()), and the user interface will display the error message for the user (DisplayErrorMessage()). The user needs to eliminate the malicious items, change the items, re-enter details of items and post again (ReenterItemDetails()), and the user interface receive and request approval to the backend system (ReRequestApprovalForUserLists()).

- If no malicious items are found, the backend system will send a reminder to the user interface that the system is not responsible for their items and any loss or damage is in the user’s name (DisplayReminder()), and the user interface will display the reminder to the user (DisplayReminder()). The user must accept the reminder to post lists so they will agree with reminder (AgreeWithReminder()), and the user interface will send the data that user has accepted with the reminder (SendUserResponse()). Finally, the backend system will approve and publish the user’s list (ApproveUserList()), and the user interface will receive the data and display the user's list and its details (DisplayUserListDetails()).

case 3


Online Registration
User
User interface
Backend System
ClickOnRegister()
RequestRegisterPage()
ReturnRegisterPage()
DisplayRegisterPage()
EnterUserDetails()
ClickOnContinue()
SendUserDetails()
alt [If Details are invalid]
SendError()
DisplayErrorMessage()
ReenterUserDetails()
ReverifyUserDetails()
DisplayRequest()
SyncGPS()
RequestToSyncGPS()
SendUserAcceptance()
ValidateRegisteration()
DisplayValidation()
Log In
ClickOnLogin()
RequestLoginPage()
ReturnLoginPage()
DisplayLoginPage()
EnterEmailAddress AndPassword()
ClickOnContinue()
SendEmailAddress AndPassword()
alt [If Details are invalid]
SendError()
DisplayErrorMessage()
ReenterEmailAddress AndPassword()
ReverifyEmailAddress AndPassword()
DisplayRequest()
SyncCamera()
RequestToSyncCamera()
SendUserAcceptance()
DisplayRequest()
SyncGallery()
RequestToSyncGallery()
SendUserAcceptance()
DisplayValidation()
ValidateLogin()
ProceedTo MainPage()



The third case of a sequence diagram is viewing lists. First, the user clicks the “Register” button (ClickOnRegister()) on the first page of the application. User interface receives the

request from the user and then sends the request data packet of the register page to the backend system (RequestRegisterPage()). The application backend system will reply with the data packet of the register page (ReplyRegisterPage()), and the user interface will receive it and create a register page, displayed for the user (DisplayRegistrationPage()).

User enters personal information such as nickname, age, phone number, email address, etc. (EnterUserDetails()) and submit by clicking “continue” (ClickOnContinue()). User interface receives that information and sends it to the backend system (SendUserDetails()), then the backend system will verify user information details. If the nickname or email address, phone number are invalid, the backend system will send an error to the user interface (SendError()) and the user interface will display an error message for the user (DisplayErrorMessage()). In this case, the user must re-enter information (ReenterUserDetails()) and submit again to the backend system via user interface (ReverifyUserDetails()). Otherwise, the backend system will request to sync the GPS of the user’s phone (RequestToSyncGPS()): user interface will display the request message (DisplayRequest()). The user must accept the sync if the user wants to proceed (SyncGPS()): the user interface will send the data of acceptance to the backend system (SendUserAcceptance()). When all of these are finished, the backend system will validate the user registration (ValidateRegistration()), user interface receive it and display validation for the user (DisplayValidation()).

After the registration, user press “Log In” (ClickOnLogin()) and user interface will request login page (RequestLoginPage()), backend system will return login page data packet (ReturnLoginPage()), therefore user interface will display login page (DisplayLoginPage()). Then, the user needs to input the email address and password already registered (EnterEmailAddressAndPassword()) and click “continue” to submit (ClickOnContinue()). User interface will send the data of user login information to the backend system (SendEmailAddressAndPassword()), and the backend system will verify that information. Except for the case that both email address and password are correct, the backend system will send an error to the user interface (SendError()) and the user interface will display an error message for the user (DisplayErrorMessage()). In this case, the user must re-enter information (ReenterUserDetails()) and submit again to the backend system via user interface (ReverifyUserDetails()). If both information is correct, the backend system will request to sync the Camera of the user’s phone (RequestToSyncCamera()): user interface will display the request message (DisplayRequest()). The user must accept the sync if the user wants to proceed (SyncCamera()): the user interface will send the data of acceptance to the backend

system (SendUserAcceptance()). When all of these are finished, the backend system will require the user to sync the user phone's gallery (RequestToSync). Lastly the backend system will validate login (ValidateLogin()), and the user interface will display validation information for users (DisplayValidation()).

When both registration and login are completed appropriately, the user will automatically proceed to the main page (ProceedToMainPage()).

In the main page, the user will click "View Lists" (ProceedToViewLists()), and the user interface will request a "View Lists" page to the backend system (RequestViewLists()). The backend system will reply by sending a "View Lists" page (ReturnViewLists()), the user interface will receive the data and display a "View Lists" page for the user (DisplayViewLists()).

- If the user wants to view past lists, then the user will click "Past Lists" (SelectPastLists()), and the user interface will receive the click response and request "Past Lists" data to the backend system (RequestPastLists()). The backend system will reply to the request with returning data of "Past Lists" to the user interface (ReturnPastLists()), and the user interface will receive data and display "Past Lists" to the user (DisplayPastLists()).

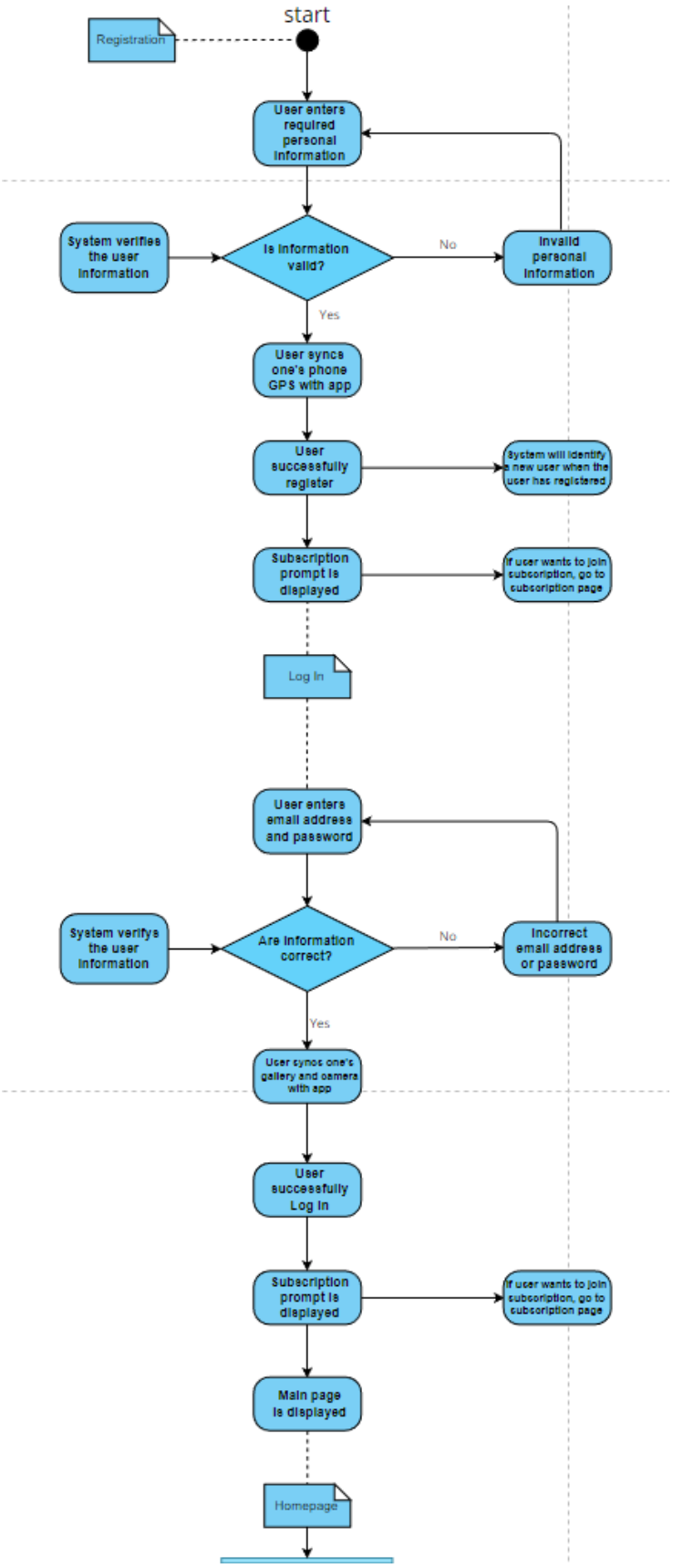
- If the user wants to view current lists, then the user will click "Current Lists" (SelectCurrentLists()), and the user interface will receive the click response and request "Current Lists" data to the backend system (RequestCurrentLists()). The backend system will reply to the request with returning data of "Current Lists" to the user interface (ReturnCurrentLists()), and the user interface will receive data and display "Current Lists" to the user (DisplayCurrentLists()).

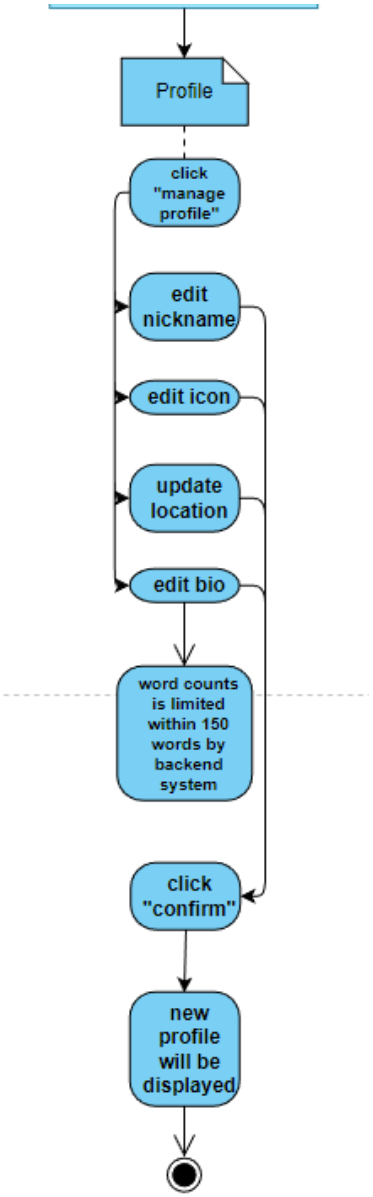
    - In the case that the user wants to delete any list in "Current lists", then the user will select the list to delete (SelectDelete()). The user interface will receive the selection, send data of details of the list to be deleted by the user to the backend system (SendDetailsOfListToBeDeleted()). The backend system will delete details of the user's list (DeleteUserListDetails()), and the user interface will receive notification that list details are deleted, display a notification for the user that the list is successfully deleted (DisplayNotificationOfSuccessfullyDeleted()).

- If the user wants to view liked lists, then the user will click “Liked Lists” (SelectLikedLists()), and the user interface will receive the click response and request “Liked Lists” data to the backend system (RequestLikedLists()). The backend system will reply to the request with returning data of “Liked Lists” to the user interface (ReturnLikedLists()), and the user interface will receive data and display “Liked Lists” to the user (DisplayLikedLists()).
    - In the case that the user wants to delete any list in "Liked lists", then the user will select the list to delete (SelectDelete()). The user interface will receive the selection, send data of details of the list to be deleted by the user to the backend system (SendDetailsOfListToBeDeleted()). The backend system will delete details of the user’s list (DeleteUserListDetails()), and the user interface will receive notification that list details are deleted, display a notification for the user that the list is successfully deleted (DisplayNotificationOfSuccessfullyDeleted()).

Activity Diagram

case 1





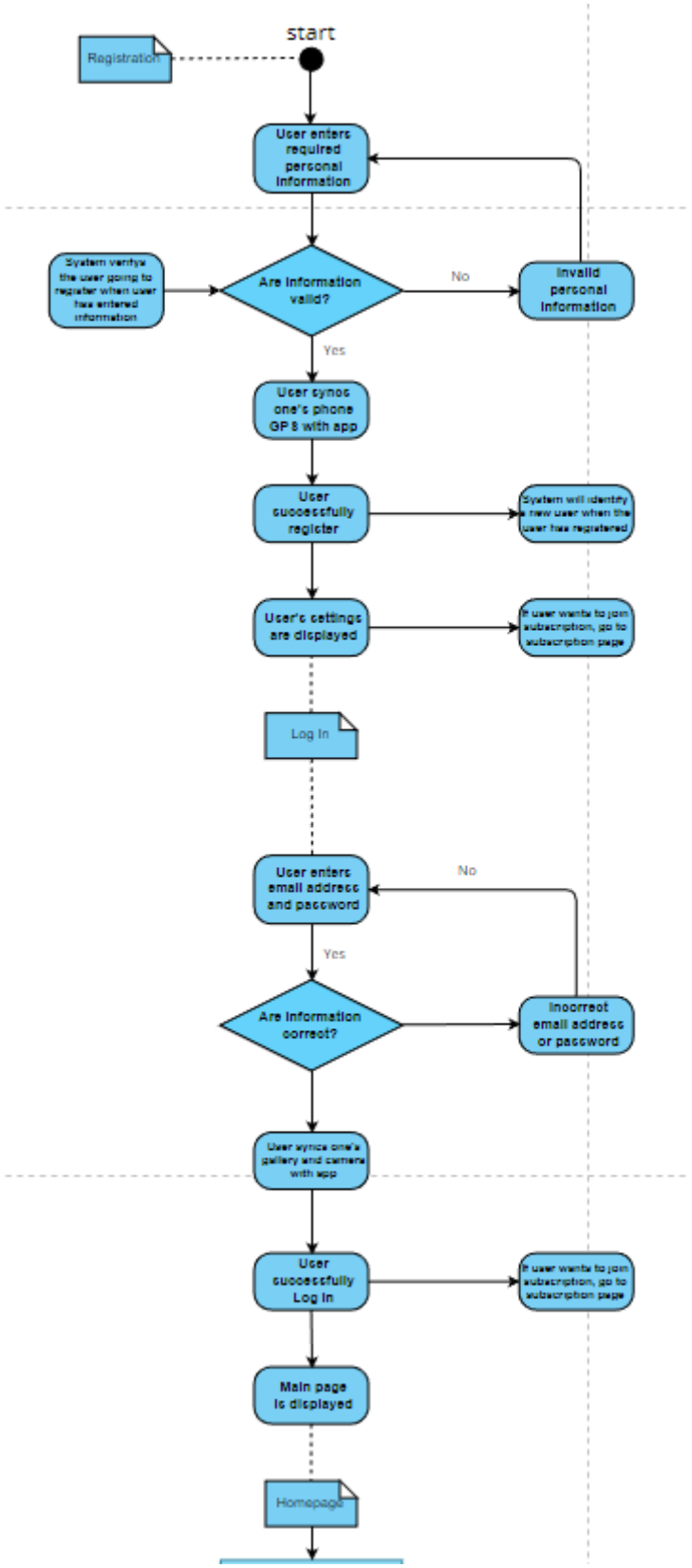
The first case of the activity diagram is to edit the user profile. First, the user enters required personal information and submit. System verifies the user information, and if the information is not valid, the user needs to go back to the step of entering required personal information. Otherwise if the information is valid, the user will be required to sync GPS with the application, and the user registration will be successfully completed after sync. (If a user wants to subscribe to a premium, the user can proceed to the subscription registration page via prompt displayed after registration is completed.)

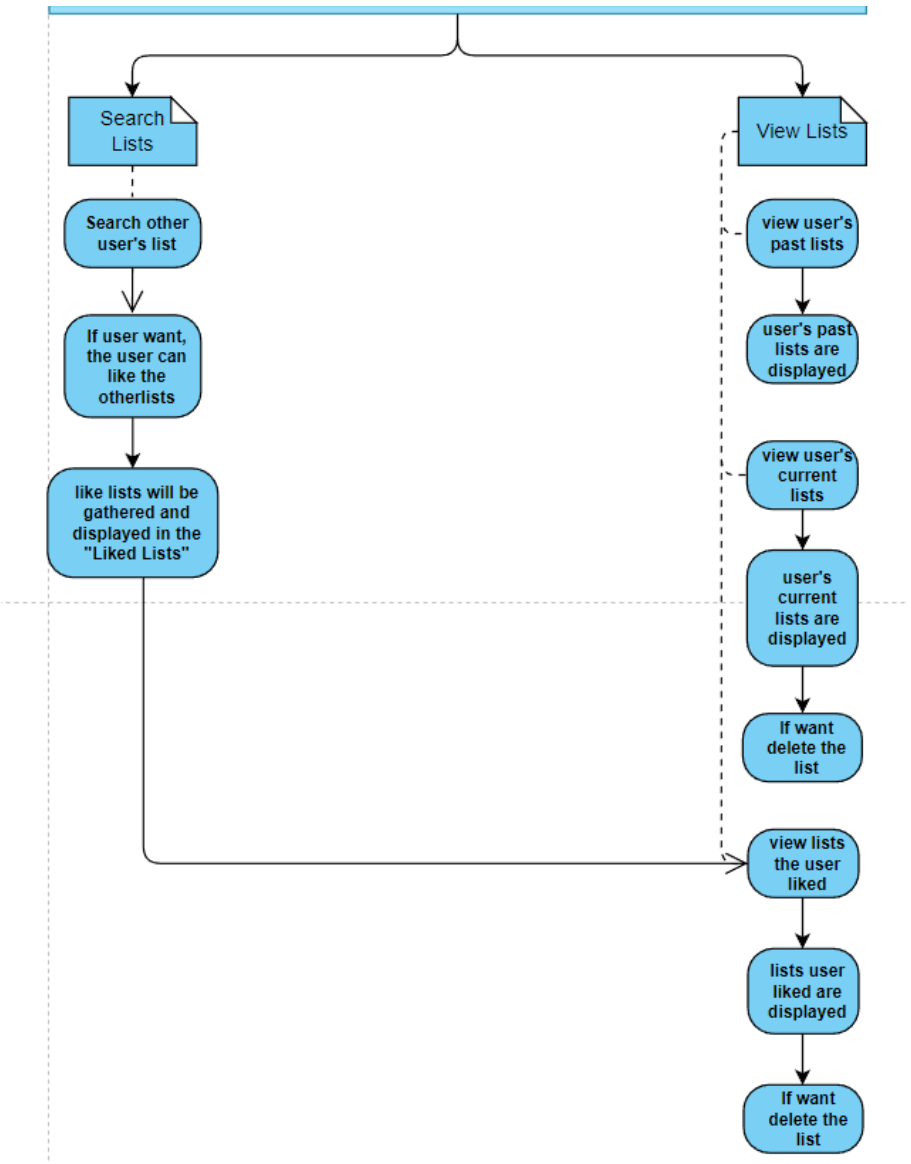
Next, the user enters an email address and password to log in. System verifies that information, and if either or both information is incorrect, the user needs to go back to the step of entering the email address and password. Otherwise if the both information is correct, the user will be required to sync gallery and phone in the user's camera with the application. Once verification and sync are all done successfully, the log in will be completed and the main page will be displayed. (As same as after the completion of registration, if the user wants to subscribe to a premium, the user can proceed to the subscription registration page via prompt displayed after login is completed.)

From the Homepage, the user will proceed to the Profile.

In the profile, the user can edit the profile by clicking “Manage profile”. The user is allowed to edit nickname, icon, bio, or update location. For editing bio, the word count is limited within 150 words by the backend system, so if the user’s bio exceeds 150 words, the user cannot save the new bio and have to write again within 150 words. Once a change is made, the user will click “confirm” to save, and new profile details will be displayed. Eventually the diagram ends, reaching the Activity final node.

case 2


start
Registration
User enters required personal information
System verifys the user going to register when user has entered information
Are information valid?
No
Invalid personal information
Yes
User synoc one's phone GPS with app
User successfully register
System will identify a new user when the user has registered
User's settings are displayed
If user wants to join subscription, go to subscription page
Log In
User enters email address and password
No
Yes
Are information correct?
Incorrect email address or password
User syncs one's gallery and camera with app
User successfully Log In
If user wants to join subscription, go to subscription page
Main page is displayed
Homepage



The first case of the activity diagram is to edit the user profile. First, the user enters required personal information and submit. System verifies the user information, and if the information is not valid, the user needs to go back to the step of entering required personal information. Otherwise if the information is valid, the user will be required to sync GPS with the application, and the user registration will be successfully completed after sync. (If a user wants to subscribe to a premium, the user can proceed to the subscription registration page via prompt displayed after registration is completed.)

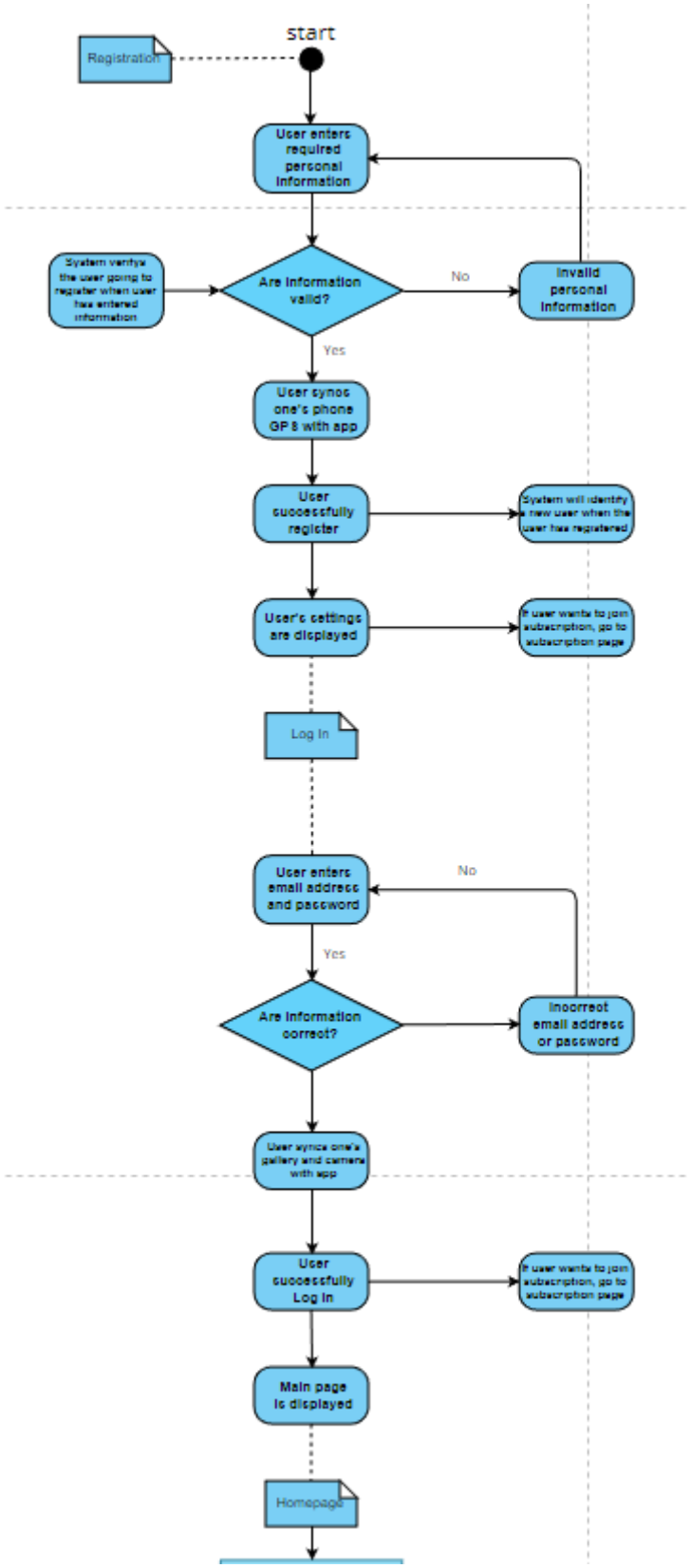
Next, the user enters an email address and password to log in. System verifies that information, and if either or both information is incorrect, the user needs to go back to the step of entering the email address and password. Otherwise if the both information is correct, the user will be required to sync gallery and phone in the user's camera with the application.

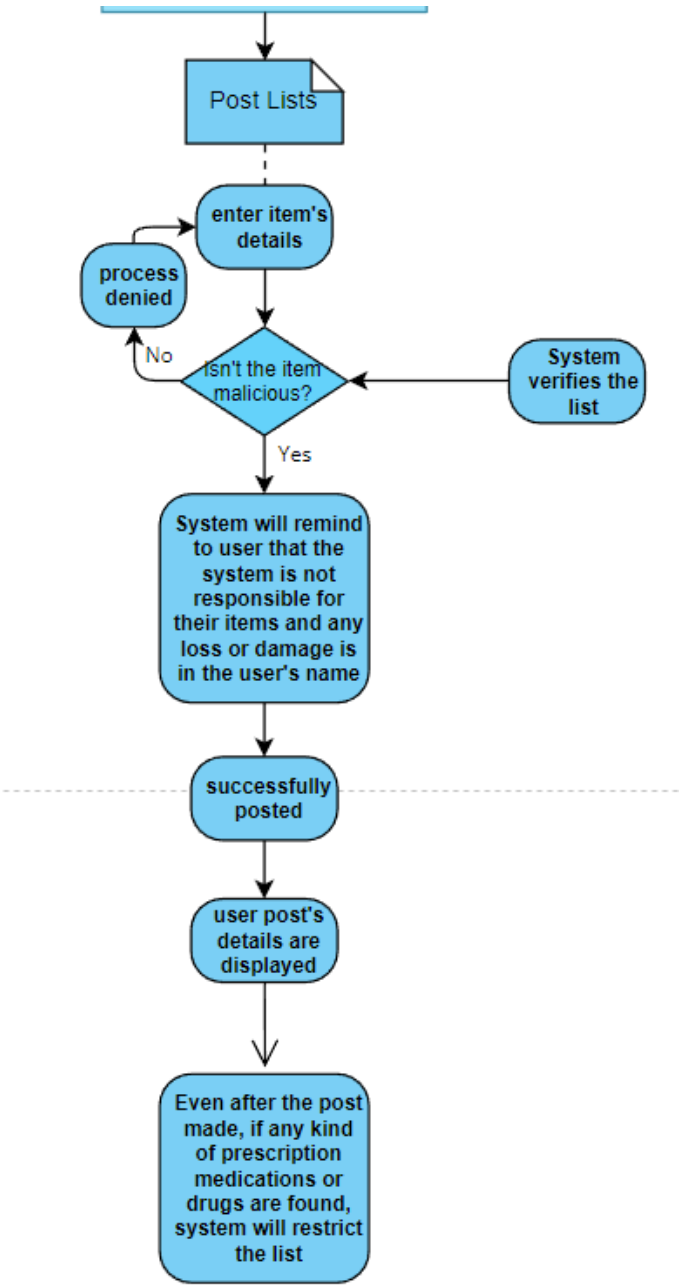
Once verification and sync are all done successfully, the log in will be completed and the main page will be displayed. (As same as after the completion of registration, if the user wants to subscribe to a premium, the user can proceed to the subscription registration page via prompt displayed after login is completed.)

From the Homepage, the user will proceed to the View Lists.

- If the user wants to view past lists already negotiated, the user will click “Past Lists” to view them. Then the past lists will be displayed.
- If the user wants to view current lists that have not been negotiated, the user will click “Current Lists” to view them. Then the current lists will be displayed. The user can also delete the current lists in some cases.
- If the user wants to view liked lists that have not been negotiated, the user will click “Liked Lists” to view them. Then the liked lists will be displayed. The user can also delete the liked lists in some cases.
    - Liked lists are also added or deleted with “Search Lists”" function. In this case, the user proceeds to “Search Lists” from Homepage, searching other user’s lists. At the sametime, if the user wants to like the list, than the user will click heart shaped button and it will be saved in “Liked Lists” category in “View Lists”.

case 3


start
Registration
User enters required personal information
System verifys the user going to register when user has entered information
Are information valid?
No
Invalid personal information
Yes
User synoc one's phone GPS with app
User successfully register
System will identify a new user when the user has registered
User's settings are displayed
If user wants to join subscription, go to subscription page
Log In
User enters email address and password
No
Yes
Are information correct?
Incorrect email address or password
User syncs one's gallery and camera with app
User successfully Log In
If user wants to join subscription, go to subscription page
Main page is displayed
Homepage



The first case of the activity diagram is to edit the user profile. First, the user enters required personal information and submit. System verifies the user information, and if the information is not valid, the user needs to go back to the step of entering required personal information. Otherwise if the information is valid, the user will be required to sync GPS with the application, and the user registration will be successfully completed after sync. (If a user wants to subscribe to a premium, the user can proceed to the subscription registration page via prompt displayed after registration is completed.)

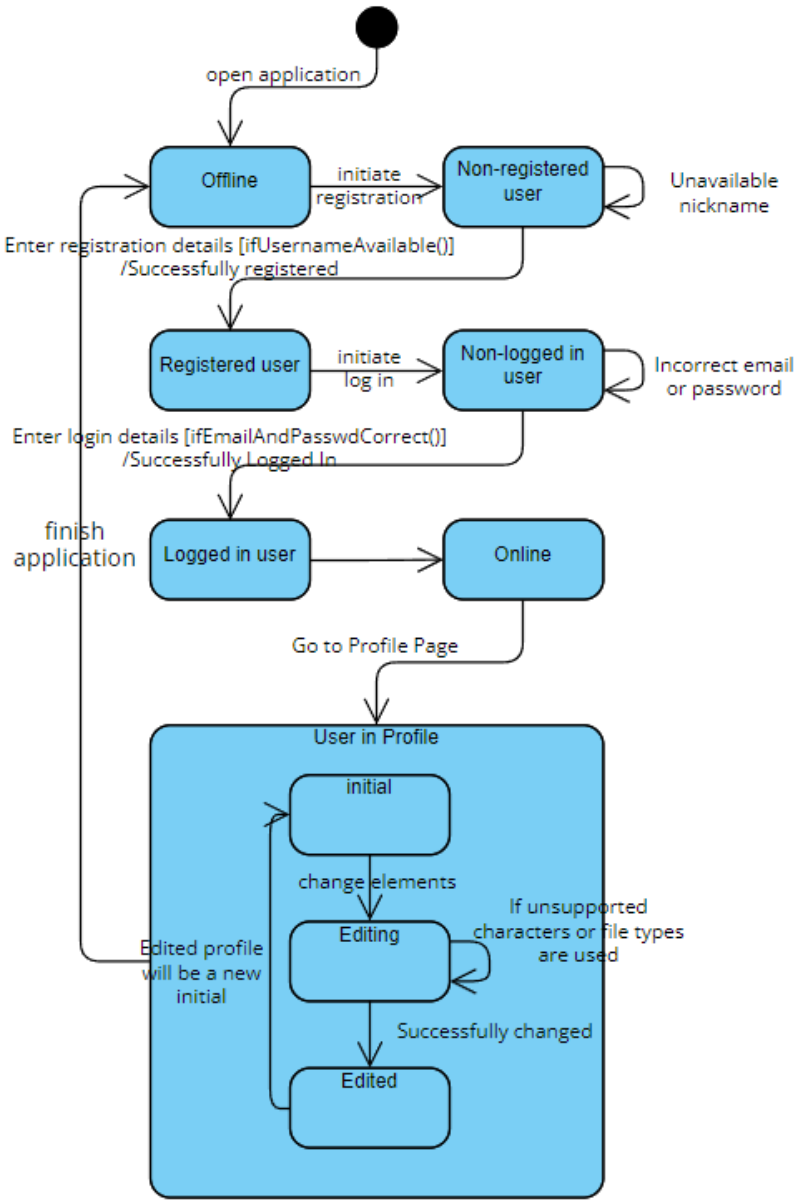
Next, the user enters an email address and password to log in. System verifies that information, and if either or both information is incorrect, the user needs to go back to the step of entering the email address and password. Otherwise if the both information is correct, the user will be required to sync gallery and phone in the user's camera with the application. Once verification and sync are all done successfully, the log in will be completed and the main page will be displayed. (As same as after the completion of registration, if the user wants to subscribe to a premium, the user can proceed to the subscription registration page via prompt displayed after login is completed.)

From the Homepage, the user will proceed to the Post Lists.

The user enters the item’s details in the list, and clicks “Post” to post the list. System will verify the list and if any malicious item is discovered, the process will be denied and the user cannot post the list, or have to change the item. Otherwise, the system will remind the user that the system is not responsible for their items and any loss or damage is in the user’s name. After the user agrees with the reminder, the list will be successfully posted, details of the list will be displayed. Nevertheless, even after the post is published, if any kind of prescription medications or drugs are found, the system will immediately restrict the list.

State Chart Diagram

case 1



The first case is the state chart diagram of “Profile” modification. Initially the status is “Offline” when the user is not using the application, and this alters to “Non-registered user” when the user launches the application and initiates registration. The user enters registration details to complete online registration, and if the nickname is available (ifUsernameAvailable()), the account will be successfully registered and the status will be switched to “Registered user”. Otherwise, if the nickname is unavailable, the user needs to change the nickname and the status will not be changed from “Non-registered user”.

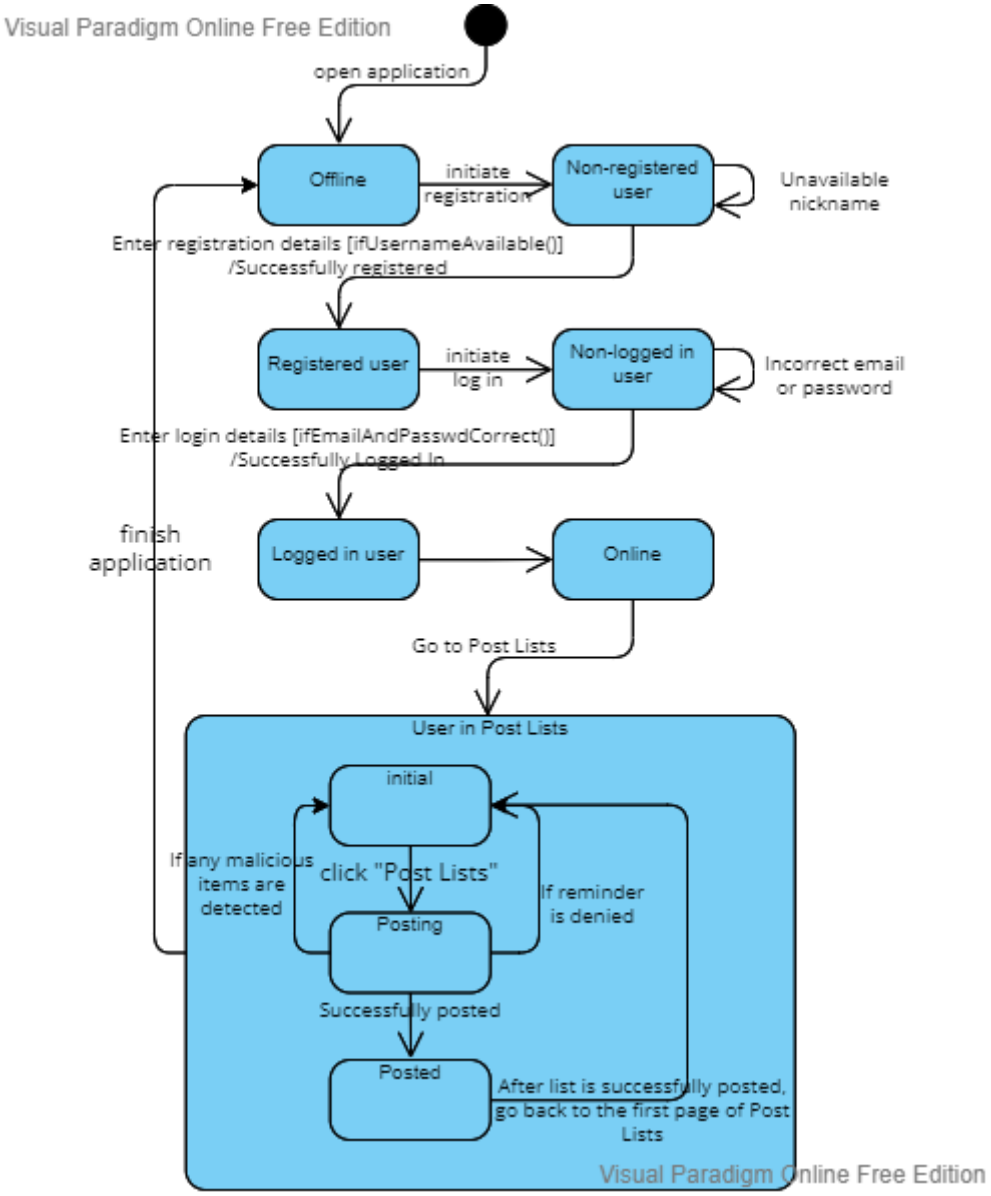
At “Registered user”, the user has completed online registration, but not login. Therefore, the user status will be changed to “Non-logged in user” when the user proceeds to the login page and initiates login. In this stage, user inputs the email address and password to log in, and if those information is correct (ifEmailAndPasswdCorrect()), the user will successfully log in and the status will be changed to “Logged in user”, Otherwise, the status will not be changed from “Non-logged in user”, and the user needs to enter the information which is valid one and would not cite the error message.

When both registration and login are completed entirely, status will be “Online” from “Logged in user”. From here, the user will proceed to “Profile”, and the status will be changed from “Online” to “User in Profile”.

The ‘Initial’ status of “User in Profile" is the profile that the user set in the steps of online registration and has not been editted. When the user clicks “Manage Profile”, the status will be changed from ‘Initial’ to ‘Editing’ and the user can change elements. After the user makes all changes in profile and tries to confirm and save, if any unsupported characters or file types are used in the profile, the status cannot be changed and remain as “Editing”, the user is required to eliminate those unsupported elements and use available characters or images. If they are available, the profile will be successfully changed, and the status will be ‘Edited’. This ‘Edited’ profile will be a new initial profile for users, so the status will be altered from ‘Edited’ to ‘initial’ again.

After the user completes all operations and quits the application, the status will be changed from “User in Profile” to “Offline”.

case 2



The second case is the state chart diagram of “Post Lists” modification. Initially the status is “Offline” when the user is not using the application, and this alters to “Non-registered user” when the user launches the application and initiates registration. The user enters registration details to complete online registration, and if the nickname is available (ifUsernameAvailable()), the account will be successfully registered and the status will be switched to “Registered user”. Otherwise, if the nickname is unavailable, the user needs to change the nickname and the status will not be changed from “Non-registered user”.

At “Registered user”, the user has completed online registration, but not login. Therefore, the user status will be changed to “Non-logged in user” when the user proceeds to the login page

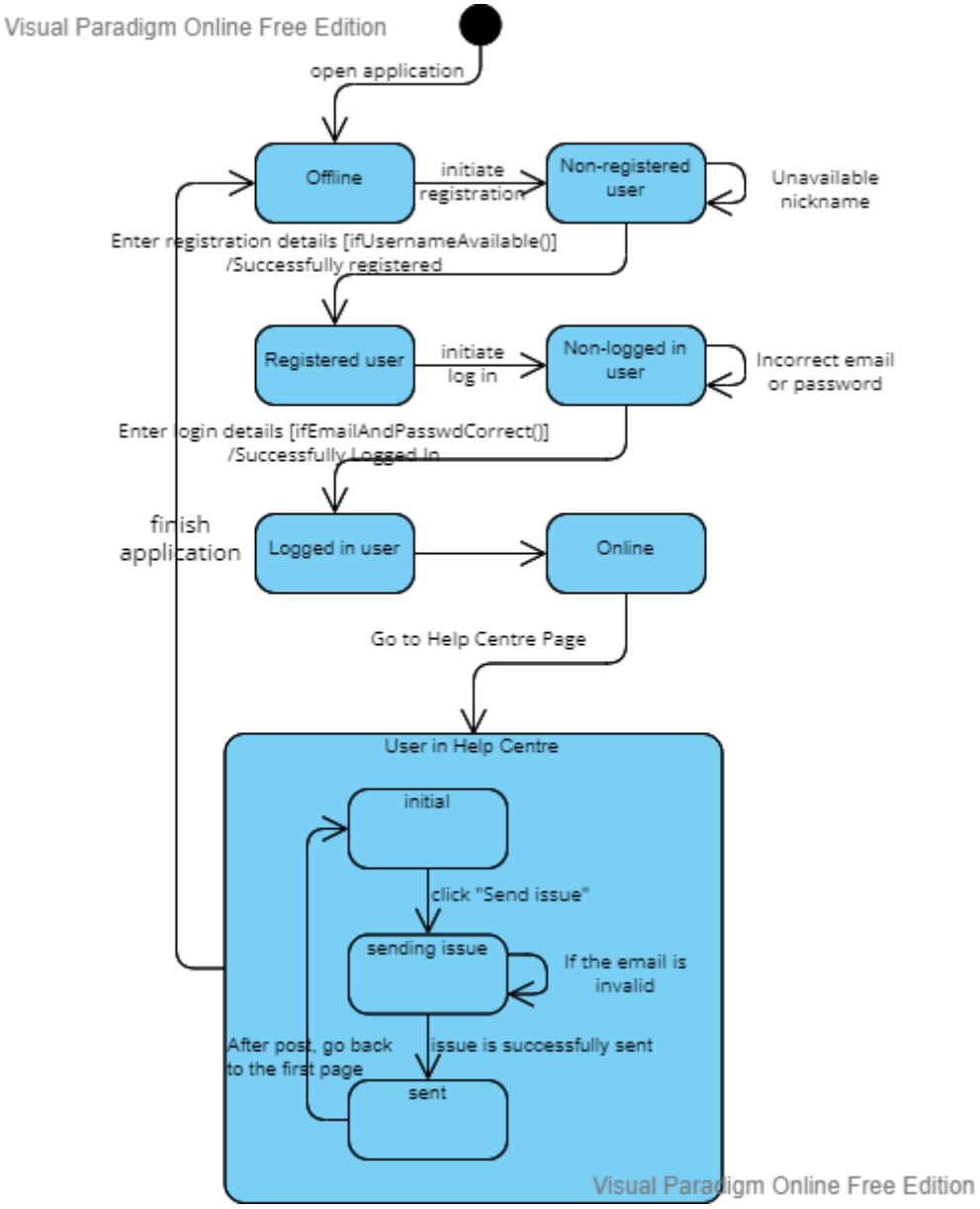
and initiates login. In this stage, user inputs the email address and password to log in, and if those information is correct (ifEmailAndPasswdCorrect()), the user will successfully log in and the status will be changed to “Logged in user”, Otherwise, the status will not be changed from “Non-logged in user”, and the user needs to enter the information which is valid one and would not cite the error message.

When both registration and login are completed entirely, status will be “Online” from “Logged in user”. From here, the user will proceed to “Post Lists”, and the status will be changed from “Online” to “User in Post Lists”.

The ‘initial’ status of “User in Post Lists” is the page that the user keys in item details. When all details are keyed in and the user clicks “Post” to publish the list, the status will be changed to ‘posting’ to be verified by the backend system. If any malicious items are detected, the approval of the new list will be denied and the status will be back to ‘initial’, the user needs to key in all details again. Even after the verification of malicious items is passed, if the user denies the reminder that the system is not responsible for their items and any loss or damage is in the user’s name, the status will not be changed to “Posted” and also back to the ‘initial’ status. Otherwise, if no malicious items are detected and a reminder is also accepted by the user, the post will be successfully posted and the status will be changed to ‘Posted’ for a while. Then soon the status will be changed to ‘initial’ that means the newly created list is already posted so the system will bring the user back to the post page.

After the user completes all operations and quits the application, the status will be changed from “User in Profile” to “Offline”.

case 3



The third case is the state chart diagram of “Help Centre” modification. Initially the status is “Offline” when the user is not using the application, and this alters to “Non-registered user” when the user launches the application and initiates registration. The user enters registration details to complete online registration, and if the nickname is available (ifUsernameAvailable()), the account will be successfully registered and the status will be switched to “Registered user”. Otherwise, if the nickname is unavailable, the user needs to change the nickname and the status will not be changed from “Non-registered user”.

At “Registered user”, the user has completed online registration, but not login. Therefore, the user status will be changed to “Non-logged in user” when the user proceeds to the login page

and initiates login. In this stage, user inputs the email address and password to log in, and if those information is correct (ifEmailAndPasswdCorrect()), the user will successfully log in and the status will be changed to “Logged in user”, Otherwise, the status will not be changed from “Non-logged in user”, and the user needs to enter the information which is valid one and would not cite the error message.

When both registration and login are completed entirely, status will be “Online” from “Logged in user”. From here, the user will proceed to “Help Centre”, and the status will be changed from “Online” to “User in Help Centre”.

The ‘initial’ status of “User in Post Lists” is the page where a lot of Q&A or feedback is displayed with division in categories. User clicks “Send issue”, writes an issue and sends if the user wants to send an issue the user has experienced and wants to ask. In this stage, the status will be ‘sending issue’. System will verify the issue sent, and if the email used for sending the issue is invalid, the issue will not be sent and sent back to the user, the user needs to write again within 150 words. When the issue is less than 150 words, the issue will be successfully sent to the admin and the status will be ‘sent’, soon it changes again to ‘initial’.

After the user completes all operations and quits the application, the status will be changed from “User in Profile” to “Offline”.

Design Rationale

Our application is designed to enable communities to share resources leading to reduced waste generation and resource scarcity. It ensures this and successfully reaches its goals by the various features embedded in the intelligent resource sharing platform of ConserveR. One of the prime features in the app is the post list feature which forms the back bone of it, allowing users to post listings for reusable items and offer them up for free or on a borrowing basis. This makes our app unique in the sense that it gives users the option to generously hand down an item that would have otherwise gone to waste. This especially reduces food wastage, allowing people to distribute surplus or unneeded food instead of dumping it away. In this way, the platform is able to be utilized as an efficient resource sharing tool and contribute positively to the environment.

Other powerful features of the app include efficient customer service through help center and chatbot embedded in the system and AI technologies like image and speech recognition to smoothen the process of searching for items on the app. These ensure a smooth customer experience and makes the interface especially user friendly, thereby resulting in greater user satisfaction. Besides this, the app incorporates efficient security processes like E2E encryption for chats, 2FA for users at registration, efficient image detection processes for detection of harmful items to ensure greater user safety as well as suspicious link detection in app's chat feature, all of which ensure a protected environment for the users which they can place their trust on. The app also adopts a minimalistic UI in order to simplify the process of navigating around the various features of the app for the users. Despite this it accommodates enough illustrations and presents each feature in an eye-catching manner which makes it attractive and helps draw in and engage users well.

While the app stands up to its commitments efficiently, it lacks certain features like rewards and metrics to measure a user's positive impact on the environment. Adoption of such features would increase user engagement and motivate them to donate and share more, keeping the platform alive with high engagement for long. Besides this, the app falls short of enough system generated notifications that keep reminding the user of the presence of the app. This increases the chances of a user installing the app but never actively using it, increasing the passive users on the platform.

Throughout the course of development of the app multiple alternatives to key features were considered but were eventually decided against. One of these was allowing the users to offer some items for sale. This was turned down in order to stay true to the social enterprise nature of the application, which doesn't align well with people seeking to make profit by posting listings on the app instead of looking to freely share resources and contribute to the environment in a positive as well as a generous manner.

The categories displayed when posting a list also went through a design change, with the initial idea being to just divide the categories into food and non-food. This design however was imprecise and the categories could be further detailed, allowing the users to categorize their listings better as well search for listings that meet their needs more accurately. The final design sorted the items into more detailed categories namely food, digital appliances, clothes and other.

Finally, the idea of using in app assistance for user onboarding was also decided against as it was proven to be resource intensive and complicated in implementation considering its benefits, which were relatively low when compared to the work done to ensure its smooth functioning. While the said feature would have ensured an app tour to familiarize the user with the app as well as greeting messages, the core functionality of the app remains intact even in its absence. Hence, in app user assistance was omitted from the current build, however with decent probability of being added in future.

Group Analysis

We, six of us (Ishihara, Jinan, Sameed, Fahad, Sin Wei, Aman) have worked together through Assignment 1 to Assignment 2. The total output of our workings were spontaneous, everybody did a good job at all, but some strengths and weaknesses were embossed in our group works in these assessments.

One major strength to discuss is the thoroughness of what we created. From the very beginning, even when we determined scope definitions, functional and non-functional requirements for our software, we reviewed each item to the utmost limit, pointed out any inconsistencies and corrected them, and strived to create the best possible product by continually pursuing rationality to the ultimate degree. This has allowed us to propose and develop a more sophisticated software with more than 10 functions and a complete software architecture that includes all of them, with fine-tuned attention to detail. I believe we were able to accomplish this because each and every one of us took the software seriously and continued to think about it without compromise whenever possible.

On the other hand, there were some difficulties in managing work progress, scheduling, and lateral cooperation. This is something that plagued us until close to the submission deadline, and there were some scenes that somewhat lacked a sense that each of us was actively completing our work in advance and submitting it early. Furthermore, each time we have postponed the collaboration and review of the project while preparing the building while affirming something, we have struggled, especially in the final stages of the project. Even so, we were able to submit high-quality output by patiently calling out to each other over and over again (sometimes through WhatsApp or Teams, sometimes directly when we met each other in face-to-face lectures) in order to finish the work well, but there is still much improvement we can make.

If we are given the opportunity for our teamwork, we will thoroughly manage the time and schedules for each work, to avoid getting caught up in an overbooked schedule with deadlines looming. From the beginning of the semester, right after the assignment is assigned to a group and made available for viewing, we would like to establish a consistent, detailed schedule and have the determination to adhere to that schedule at all costs.

This is because this gives us more time and space to review our work and extract ideas, which allows us to create higher quality, more sophisticated products. When we are driven by

deadlines and have little room in our minds, we lose the capacity to review and further refine our work, and we may not be able to produce the results that we are supposed to produce. In fact, in this case, the postponement of the project until the deadline caused the group to lose some of its composure, and instead of reviewing the work, the atmosphere within the group became so tense that even teamwork was a concern.

Nevertheless, by working diligently and systematically for weeks and sometimes months in advance, you can work to further improve your output, leaving you with the extra energy, both physically and mentally, to even achieve 120% of your original output. This can even achieve 120% of the original output. That is why, from the beginning in earnest, I would work on the job ahead of the deadline with the intention of going to finish the job early and completing it with, say, two weeks or so of significant time left.

RPA

RPA is a budding technology that helps in quicker and easier deployment plus management of software robots. Since our application's growth is crucially dependent on customer satisfaction and loyalty, addressing their concerns and queries is considered one of the top most priority. This goal is achieved by automating important functions of the help centre that ensure a smooth customer service experience.

The platform offers automated email replies when a user contacts the help centre. This feature is implemented with the popular RPA tool- UIPath. The app generates automated replies as the first acknowledgement of a user's concern as well as for common queries mailed. This is achieved by using the get IMAP mail messages function in UIPath Studio with which emails coming from any domain can be answered by the bot. From there the mail addresses can be collected and used in a send SMTP mail message function. This simple procedure helps incorporate an efficient customer query management system on the platform.

Besides the automated email replies, our app also features an attractive and robust chatbot with the help of another powerful RPA tool- Microsoft Power Virtual Agents. Power Virtual Agents offers the necessary technology needed to create and integrate a chatbot with any kind of mobile app which makes it the perfect tool for our cross-platform application. It incorporates intelligent interactions and is, therefore, able to detect users' means and goals making the chatbot even more efficient at human interactions. Bots in Power Virtual Agents are created using a code-free graphical interface which doesn't require the development team to be well-expertised in AI. This makes it a cost-effective and convenient RPA technology to include in our resource sharing application.

Reference

Power Virtual Agents, At Your Service; A Quick-Start Introduction to Creating Powerful Chatbots. [pamphlet]. Microsoft Power Platform, Microsoft.

Automate with Rakesh (2020) *UiPath Tutorial | UiPath Auto Reply Email*. 19 April, Available at: UiPath Tutorial | Uipath Auto Reply Email ()

Willem Van Galen, (2012). Use Case Goals, Scenarios And Flows. *BAtimes* [online]. 11(). []. Available at: Use Case Goals, Scenarios and Flows - Business Analyst Articles, Webinars, Templates, Jobs (batimes.com)

Individual contribution

Ishihara Satoaki	0354208	• Introduction • Modified use case diagram • Modified class diagram • Use case description & specification case 1 • Sequence diagrams case 1 to 3 • Activity diagrams case 1 to 3 • State chart diagrams case 1 to 3 • Group Analysis • Reference & Individual contribution
Jinan Nisar	0354690	• Modified use case diagram case 2 • Modified class diagram case 2 • Interface Design Proposal • Test Plan case 1 • Design Rationale • RPA
Aman Mahmud Bin Aman Mahmud	0348725	• Use case description & specification case 1 • Test Plan case 3
Mohamed Fahad Farhad	0354487	• Use case description & specification case 2 • Test Plan case 2
Mohammed Sameed Khan	0353846	• Use case description & specification case 3 • Prototype video
Thua Sin Wei	0354566	• Use case description & specification case 3 • Prototype video