ASSIGNMENT

CE/CZ2006: Software Engineering

Building SmartNation Application

2019/2020 SEMESTER 1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING NANYANG TECHNOLOGICAL UNIVERSITY

Table of Contents

1. Product Description

1.1 Purpose

Working towards a greener society, GreenRoutine will be developing a mobile application that enables users to easily obtain the availability and details of all Cash-for-trash sites. In addition, users can find out their estimated returns and send any questions they have to a Cash-for-trash site. The objective of the application is to make recycling a social norm in Singapore by making recycling more convenient.

1.2 Scope

Objective:

- To spur the society to go green by encouraging recycling with this mobile application

Goal:

- To create a mobile application capable of:
    - Extracting details of recycling sites from an API provided by data.gov.sg
    - Organising the extracted information to make it easy for users to access the relevant information that they need
    - Allowing users to calculate their estimated returns by inputting the amount of recyclables they wish to recycle
    - Allowing users to send questions that they have to a recycling site

Limitations:

- The application is limited to use in Singapore only because all the recycling sites that the API from data.gov.sg provides information on are situated in Singapore

1.3 User and StakeHolders

The stakeholders of the project are the National Environment Agency (NEA) who manages the recycling sites, Android mobile phone users and GreenRoutine project team.

1.4 Assumptions and Constraints

Users are assumed to have an android device that has GPS functions.
Users are assumed to have Internet access.

2. Functional Requirements

1. The system must allow the user to register a new account.

1.1. The user must be able to provide his/her desired Username and Password.

1.2. The system must validate that the required fields are filled up.

1.3. The system must validate the account availability.

2. The system must allow the user to login to their account.

2.1. The system must validate that the required fields are filled up.

2.2. The system must validate that the login credentials are valid.

2.3. The system must display an error message if the login credentials are invalid.

3. The system must allow the user to use the search function to search for information of individual sites.

3.1. The user must be able to search for the locations of different sites on the map.

3.1.1. The user must be able to filter the sites by location.

3.1.1.1. Available options must be provided in the form of a drop-down list.

3.1.2. The user must be able to filter sites by the type of recyclables accepted by individual sites.

3.1.2.1. Available options must be provided in the form of checkboxes.

3.1.3. The system must display the selected site's information page when the user selects a map marker.

3.1.3.1. The information page must display relevant information on the selected site: operating days and hours, types of recyclable accepted, full address, and contact information.

3.1.3.2. The user must be able to send their questions to the selected site.

3.1.3.2.1. The length of the message must not exceed 256 characters.

3.1.3.3. The user must be able to get an estimated monetary return for their recyclables.

3.1.3.3.1. For each type of recyclables, the user must be able to enter the quantity that they want to recycle by using a number pad.

3.1.3.3.2. The system must display the estimated money return rounded to 2 decimal places.

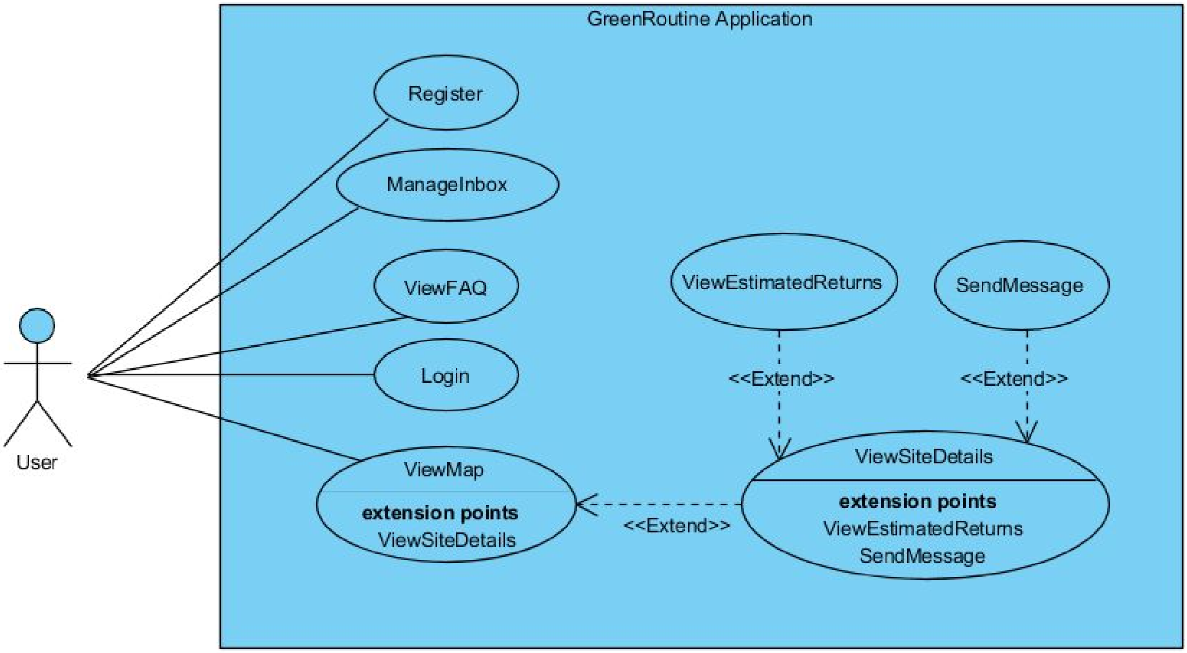
4. The system must allow the user to view their inbox for the messages they have sent and check for replies.

4.1. The user must be able to view the messages they have sent.

5. The system must have a FAQ page to help answer some common questions that users have.

6.	The system must be able to retrieve the location of user's device via Geolocation API.
7.	The system must be able to retrieve the site's information from the API provided by data.gov.sg.
8.	The system must display environmental news on the home page.

2.1 Use Case Diagram



2.2 Use Case Descriptions

Use Case ID:	1		
Use Case Name:	Register		
Created By:	Benedict Ng	**Last Updated By:**	Terry
Date Created:	1/9/2019	**Date Last Updated:**	23/10/2019

Actor:	User (Initiating Actor)
Description:	This use case allows new users to register for an account to use the system.
Preconditions:	-
Postconditions:	1. New account successfully created.
Flow of Events:	1. System displays the login page. 2. User clicks on Register button. 3. System displays register form. 4. System validates user's inputs. 5. If inputs are valid, new account is created. 6. Redirect to homepage.
Alternative Flows:	AF-05: If inputs are invalid 1. If username is too short, system displays “Username must be at least 8 characters long”. 2. If password is too short, system displays “Password must be at least 8 characters long”. 3. If username already exist, system displays “Username has been taken”.
Exception:	-
Includes:	-
Extends:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	2		
Use Case Name:	Login		
Created By:	Benedict Ng	**Last Updated By:**	Terry
Date Created:	1/9/2019	**Date Last Updated:**	23/10/2019

Actor:	User (Initiating Actor)
Description:	This use case allows the user to login to their account.
Preconditions:	1. User has an account in the system.
Postconditions:	1. User successfully login to the system.
Flow of Events:	1. System displays the login page. 2. User enters username and password. 3. User clicks on the Login button. 4. System validates the login credentials. 5. Redirect to homepage
Alternative Flows:	AF-S4: If the username and/or password is invalid. 1. System displays error message. 2. System returns to step 2.
Exception:	-
Includes:	-
Extends:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	3		
Use Case Name:	ViewMap		
Created By:	Benedict Ng	**Last Updated By:**	Yuen
Date Created:	1/9/2019	**Date Last Updated:**	8/9/2019

Actor:	User (Initiating Actor)
Description:	This use case allows the user to view map with markers indicating the sites.
Preconditions:	1. The user must be logged in.
Postconditions:	1. System displays map with markers indicating the sites.
Flow of Events:	1. User clicks on the Drawer. 2. User clicks on the “Map” option from the drawer. 3. System displays a map and filter options. 4. System displays map markers indicating the locations of individual sites. 5. User clicks on a map marker, the extended use case ViewSiteDetails will be used.
Alternative Flows:	AF-S3: User selects filter options 1. System reads user’s filter option(s) 2. System displays on the map markers of the corresponding filtered sites.
Exception:	-
Includes:	-
Extends:	1. ViewSiteDetails
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	4		
Use Case Name:	ViewSiteDetails		
Created By:	Benedict Ng	**Last Updated By:**	Terry
Date Created:	1/9/2019	**Date Last Updated:**	23/10/2019

Actor:	-
Description:	This use case allows the user to view the information page of their selected site.
Preconditions:	1. User must select a site from ViewMap
Postconditions:	1. System displays selected site’s information page.
Flow of Events:	1. System displays selected site’s information page. 2. System displays “Estimate Monetary Return” and “Send Message” options. 3. If user clicks on the “Estimate Monetary Return” option, the extended use case ViewEstimatedReturns will be used. 4. If user clicks on the “Send Message” option, the extended use case SendMessage will be used.
Alternative Flows:	AF-S1: User press back 1. System return to ViewMap
Exception:	-
Includes:	-
Extends:	1. ViewEstimatedReturns 2. SendMessage
Assumptions:	-
Notes and Issues:	-

Use Case ID:	5		
Use Case Name:	ViewEstimatedReturns		
Created By:	Benedict Ng	**Last Updated By:**	Yuen
Date Created:	1/9/2019	**Date Last Updated:**	3/10/2019

Actor:	-
Description:	This use case allows user to view the estimated returns based on the quantity of recyclables entered.
Preconditions:	1. System must be in ViewSiteDetails use case.
Postconditions:	1. System displays the estimated returns.
Flow of Events:	1. System redirects to a new page. 2. System displays the site's available recyclable types. 3. User enters quantity for each recyclable type by using a number pad. 4. System calculates estimated returns based on user's inputs. 5. System displays the estimated returns.
Alternative Flows:	AF-S1: User press back 2. System return to ViewSiteDetails
Exception:	-
Includes:	-
Extends:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	6		
Use Case Name:	SendMessage		
Created By:	Benedict Ng	**Last Updated By:**	Terry
Date Created:	1/9/2019	**Date Last Updated:**	23/10/2019

Actor:	-
Description:	This use case allows user to send questions to specific sites.
Preconditions:	1. User must be in ViewSiteDetails use case.
Postconditions:	1. System stores the user’s message.
Flow of Events:	1. System redirects to a new page. 2. System displays a message form. 3. User enters title and message. 4. System validates user’s input. 5. System stores the user’s message into the database. 6. System displays toast “Message successfully sent” 7. System redirects to ViewSiteDetails.
Alternative Flows:	AF-S6: If there are invalid inputs 1. System displays error message. 2. System returns to step 3.
Exception:	-
Includes:	-
Extends:	-
Assumptions:	-
Notes and Issues:	-

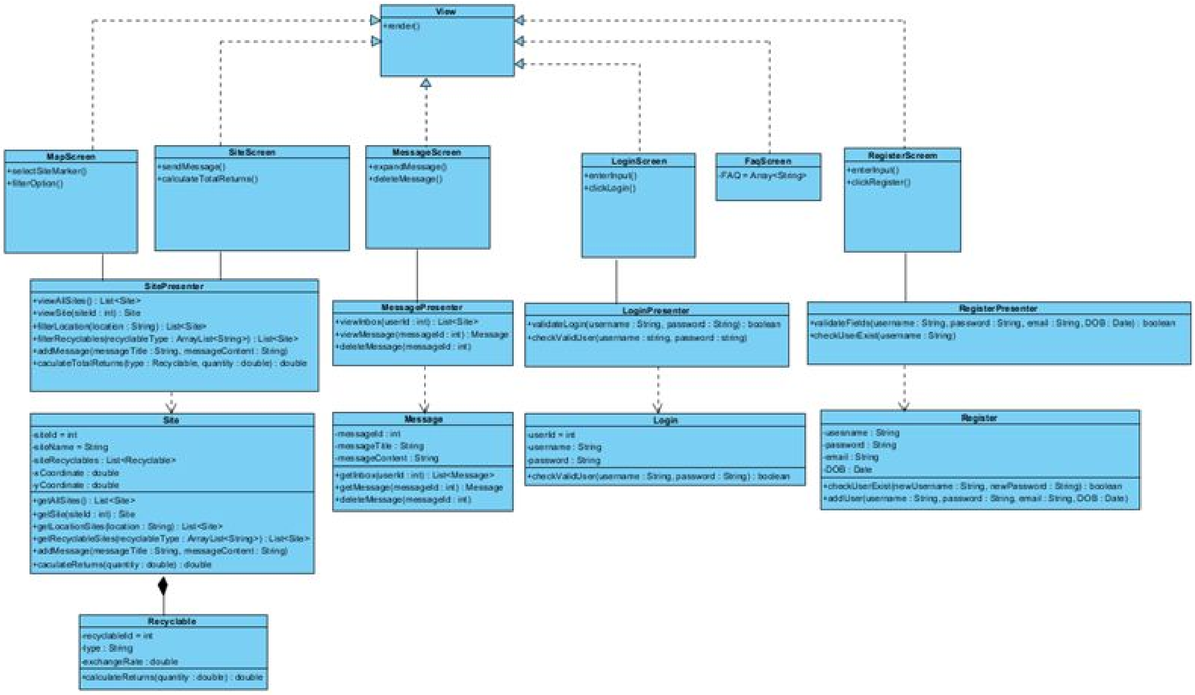
Use Case ID:	7		
Use Case Name:	ManageInbox		
Created By:	Benedict Ng	**Last Updated By:**	Benedict Ng
Date Created:	1/9/2019	**Date Last Updated:**	20/10/2019

Actor:	User (Initiating Actor)
Description:	This use case allows user to manage their Inbox.
Preconditions:	1. User must be logged in.
Postconditions:	1. System successfully displays user’s Inbox.
Flow of Events:	1. User selects the “View Inbox” option from drawer. 2. System displays the list of messages the user has sent. 3. When user clicks on one of the message titles, it expands to show the full message on the same screen.
Alternative Flows:	-
Exception:	-
Includes:	-
Extends:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	8		
Use Case Name:	ViewFAQ		
Created By:	Benedict Ng	**Last Updated By:**	Low Yu Benedict
Date Created:	1/9/2019	**Date Last Updated:**	15/9/2019

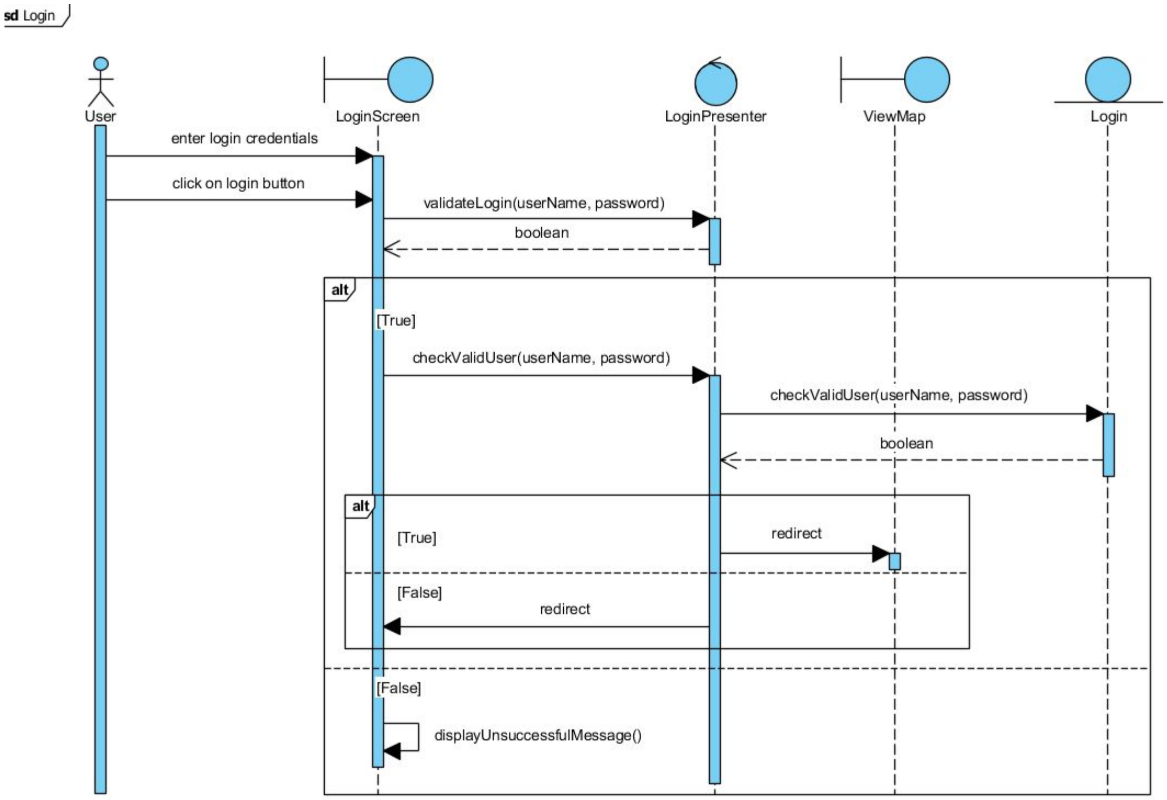
Actor:	User (Initiating Actor)
Description:	This use case allows user to view the FAQ.
Preconditions:	1. User must be logged in.
Postconditions:	1. System successfully displays the FAQ.
Flow of Events:	1. User selects the “FAQ” option from drawer. 2. System displays the FAQ.
Alternative Flows:	-
Exception:	-
Includes:	-
Extends:	-
Assumptions:	-
Notes and Issues:	-

2.3 Class Diagram

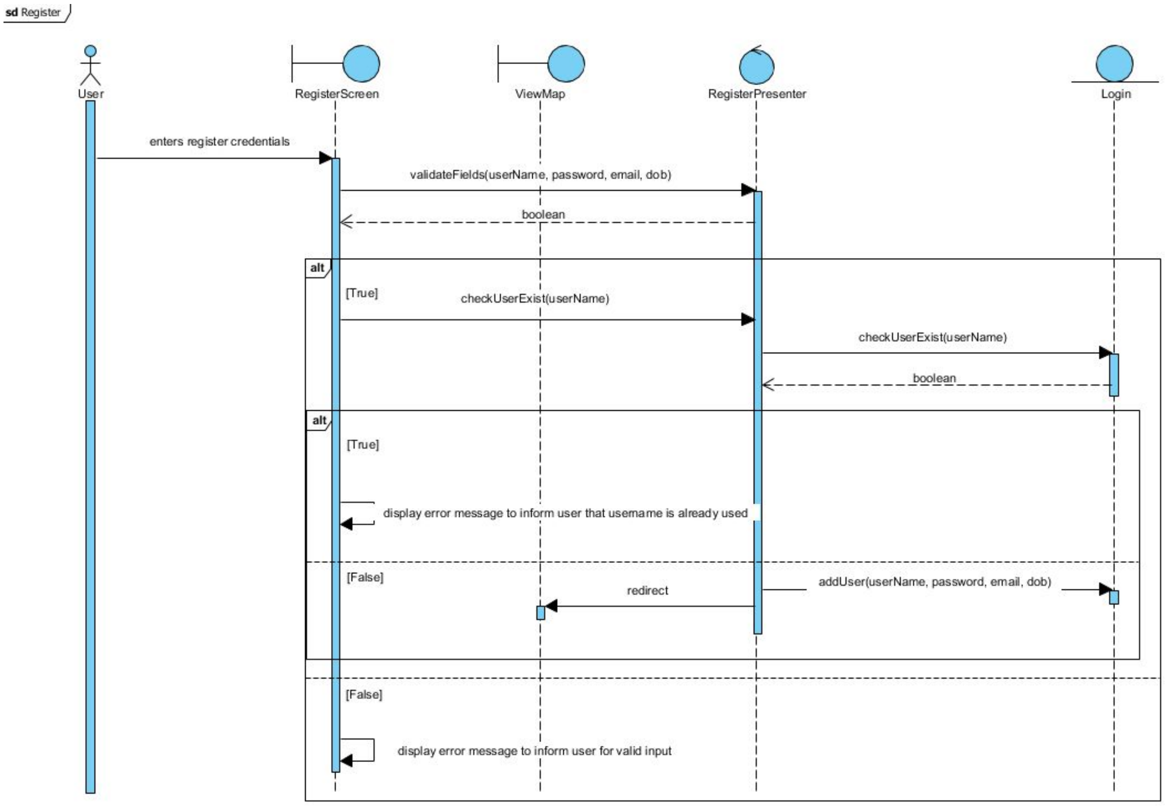

View
+render()
MapScreen
+selectSiteMarker()
+filterOption()
SiteScreen
+sendMessage()
+calculateTotalReturns()
MessageScreen
+expandMessage()
+deleteMessage()
LoginScreen
+enterInput()
+clickLogin()
FaqScreen
-FAQ = Array<String>
RegisterScreen
+enterInput()
+clickRegister()
SitePresenter
+viewAllSites() : List<Site>
+viewSite(siteId : int) : Site
+filterLocation(location : String) : List<Site>
+filterRecyclables(recyclableType : ArrayList<String>) : List<Site>
+addMessage(messageTitle : String, messageContent : String)
+caculateTotalReturns(type : Recyclable, quantity : double) : double
MessagePresenter
+viewInbox(userId : int) : List<Site>
+viewMessage(messageId : int) : Message
+deleteMessage(messageId : int)
LoginPresenter
+validateLogin(username : String, password : String) : boolean
+checkValidUser(username : string, password : string)
RegisterPresenter
+validateFields(username : String, password : String, email : String, DOB : Date) : boolean
+checkUserExist(username : String)
Site
-siteId = int
-siteName = String
-siteRecyclables : List<Recyclable>
-xCoordinate : double
-yCoordinate : double
+getAllSites() : List<Site>
+getSite(siteId : int) : Site
+getLocationSites(location : String) : List<Site>
+getRecyclableSites(recyclableType : ArrayList<String>) : List<Site>
+addMessage(messageTitle : String, messageContent : String)
+caculateReturns(quantity : double) : double
Message
-messageId : int
-messageTitle : String
-messageContent : String
+getInbox(userId : int) : List<Message>
+getMessage(messageId : int) : Message
+deleteMessage(messageId : int)
Login
-userId = int
-username : String
-password : String
+checkValidUser(username : String, password : String) : boolean
Register
-username : String
-password : String
-email : String
-DOB : Date
+checkUserExist(newUsername : String, newPassword : String) : boolean
+addUser(username : String, password : String, email : String, DOB : Date)
Recyclable
-recyclableId = int
-type : String
-exchangeRate : double
+calculateReturns(quantity : double) : double

2.3 Sequence Diagram

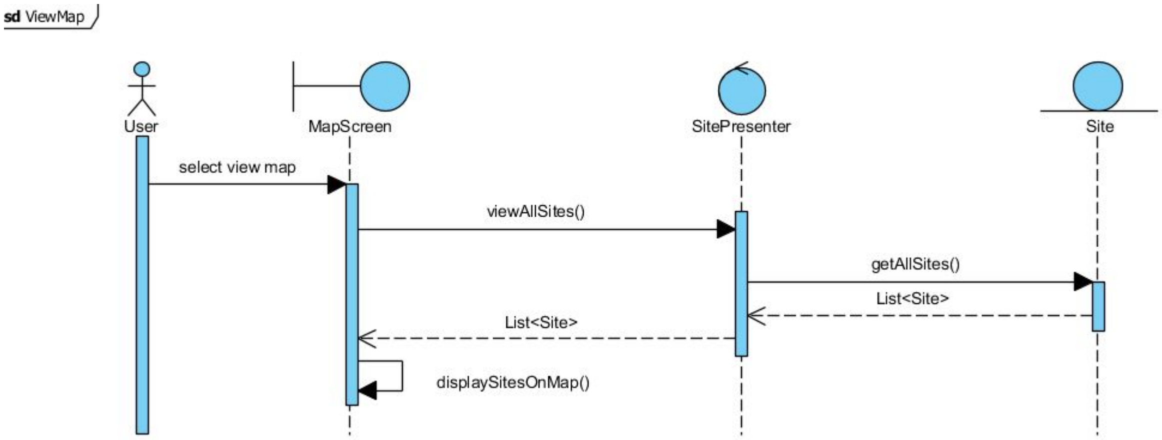
1. Login



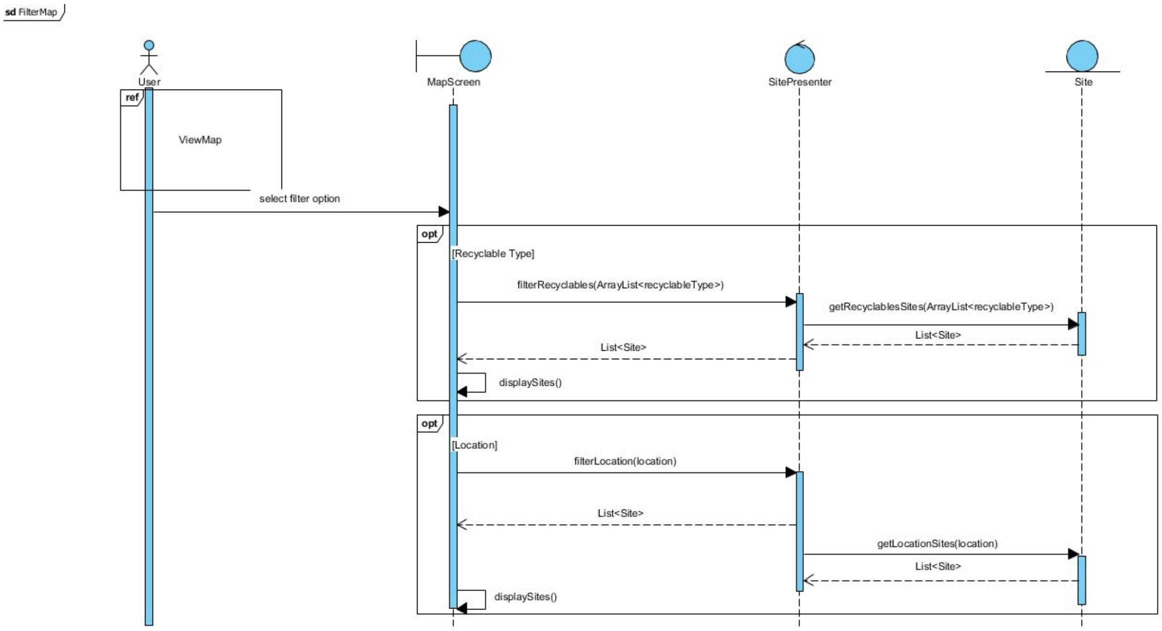
2. Register


sd Register
User
RegisterScreen
ViewMap
RegisterPresenter
Login
enters register credentials
validateFields(userName, password, email, dob)
boolean
alt
[True]
checkUserExist(userName)
checkUserExist(userName)
boolean
alt
[True]
display error message to inform user that username is already used
[False]
redirect
addUser(userName, password, email, dob)
[False]
display error message to inform user for valid input

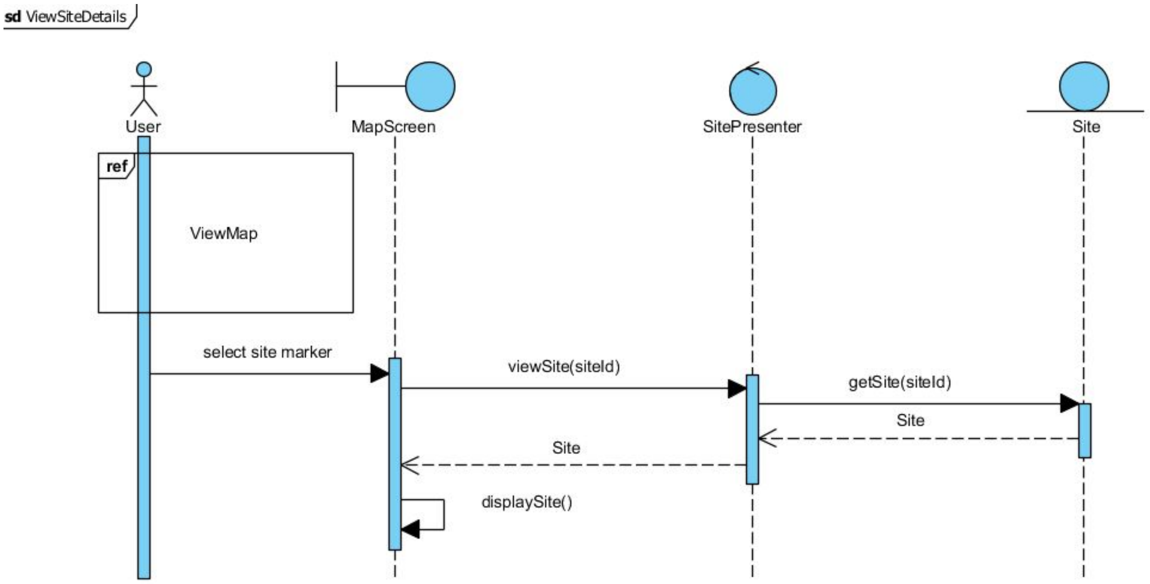
3. ViewMap



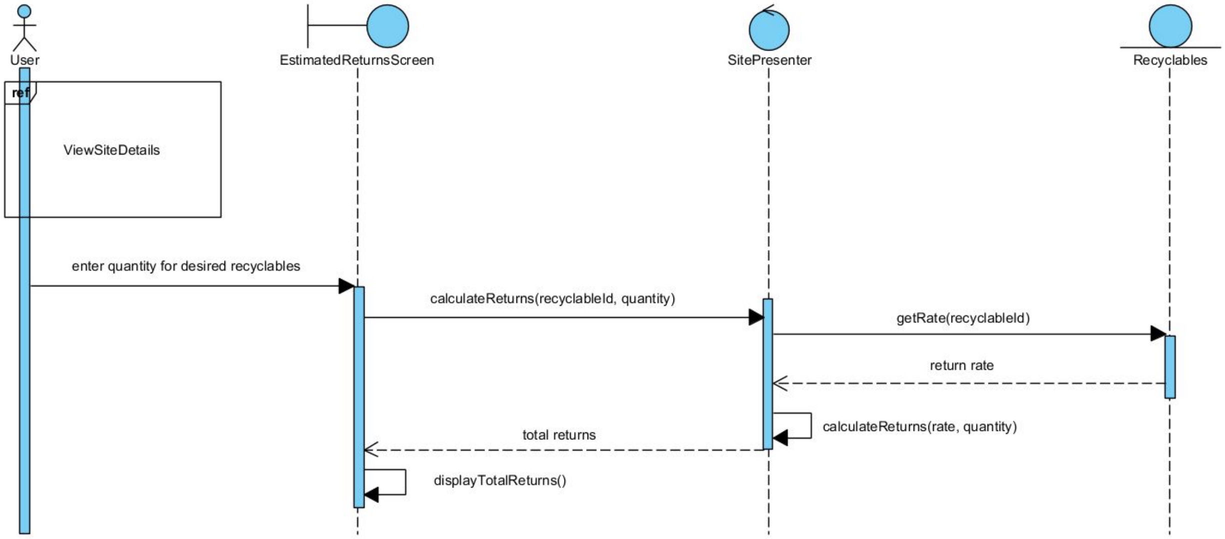
3.1. FilterMap



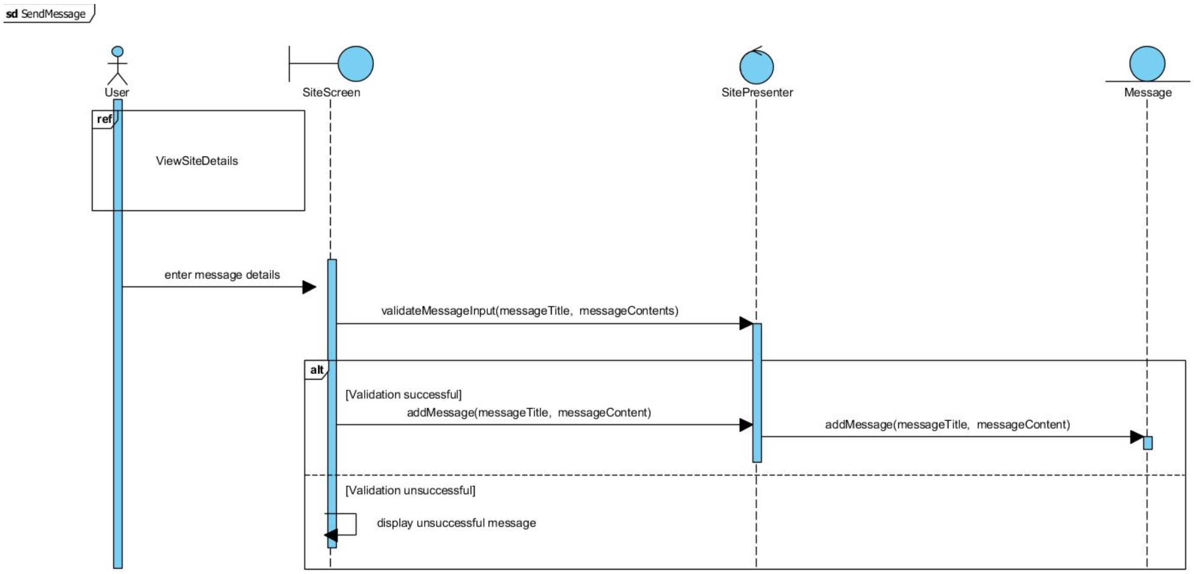
4. ViewSiteDetails



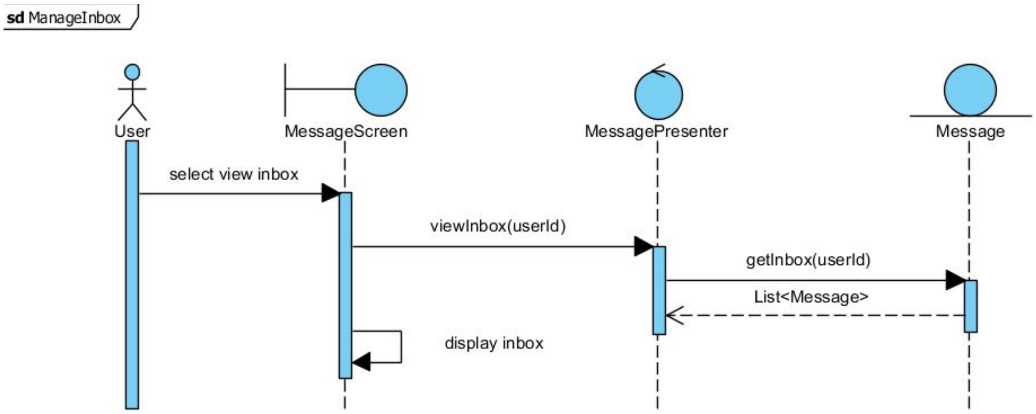
5. ViewEstimatedReturns



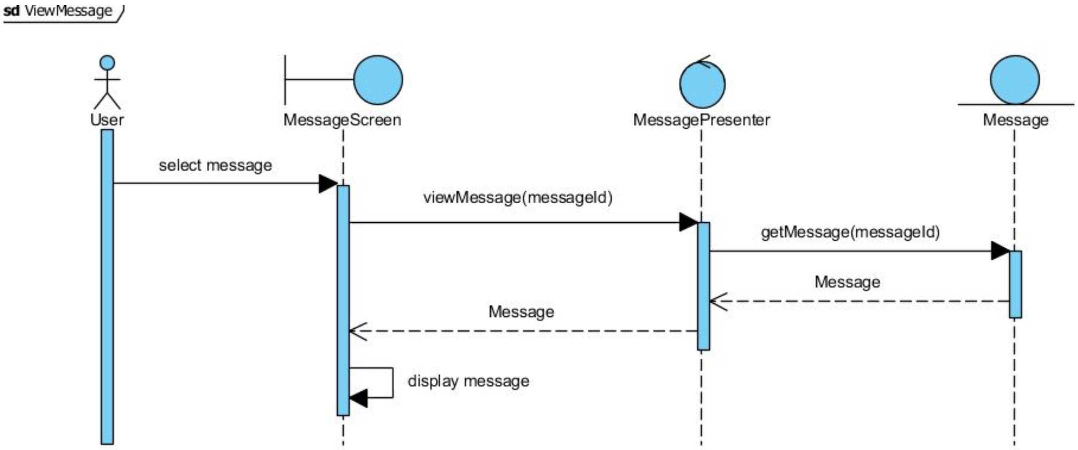
6. SendMessage


sd SendMessage
User
SiteScreen
SitePresenter
Message
ref
ViewSiteDetails
enter message details
validateMessageInput(messageTitle, messageContents)
alt
[Validation successful]
addMessage(messageTitle, messageContent)
addMessage(messageTitle, messageContent)
[Validation unsuccessful]
display unsuccessful message

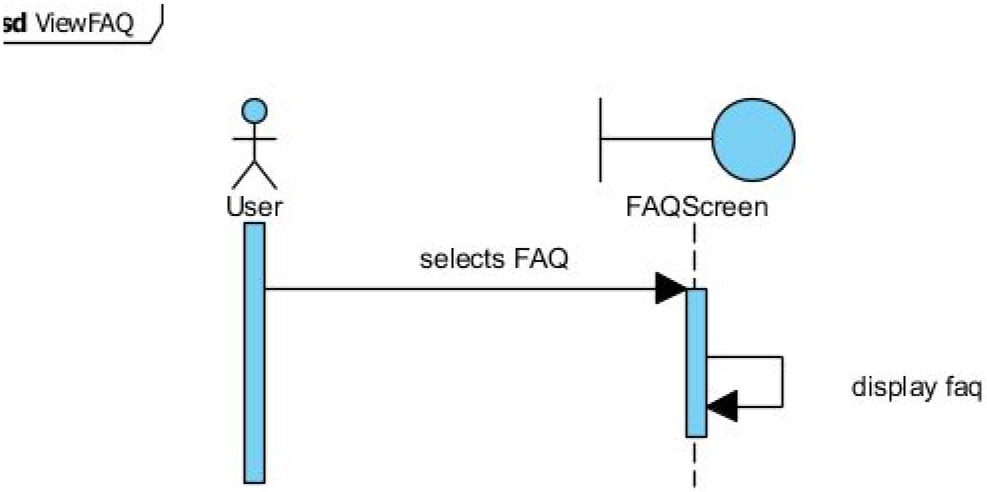
7. ManageInbox



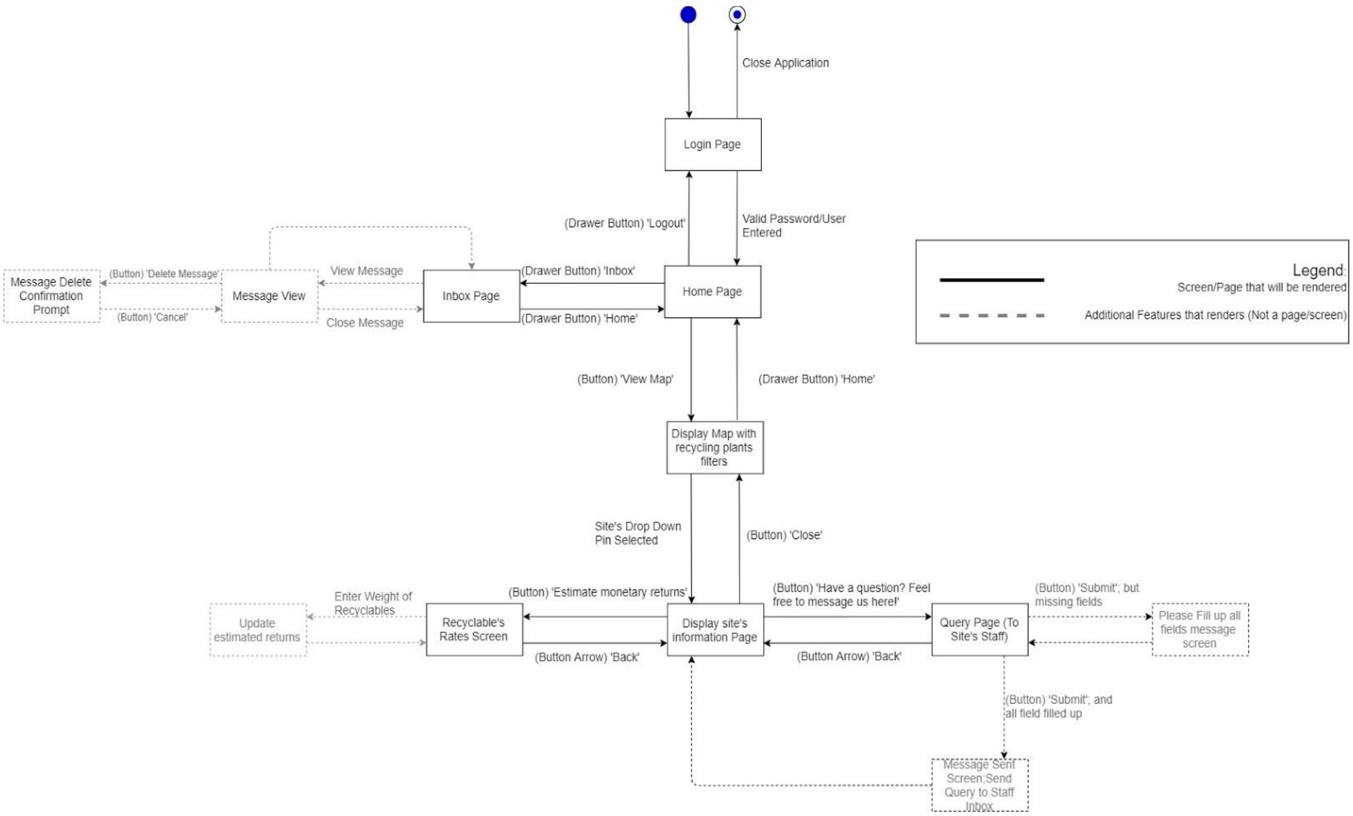
7.1. ViewMessage



8. ViewFAQ



2.4 Dialog Map


Close Application
Login Page
(Drawer Button) 'Logout'
Valid Password/User Entered
Home Page
(Drawer Button) 'Inbox'
(Drawer Button) 'Home'
Inbox Page
View Message
Close Message
Message View
(Button) 'Delete Message'
(Button) 'Cancel'
Message Delete Confirmation Prompt
Legend
Screen/Page that will be rendered
Additional Features that renders (Not a page/screen)
(Button) 'View Map'
(Drawer Button) 'Home'
Display Map with recycling plants filters
Site's Drop Down Pin Selected
(Button) 'Close'
(Button) 'Estimate monetary returns'
Display site's information Page
Recyclable's Rates Screen
Enter Weight of Recyclables
Update estimated returns
(Button Arrow) 'Back'
(Button) 'Have a question? Feel free to message us here!'
(Button Arrow) 'Back'
Query Page (To Site's Staff)
(Button) 'Submit'; but missing fields
Please Fill up all fields message screen
(Button) 'Submit'; and all field filled up
Message Sent Screen;Send Query to Staff Inbox

3. Non-Functional Requirements

1. Usability Requirements
    - 1.1. Proper reliability
        - 1.1.1. Actions function as specified
        - 1.1.2. Data on screen must reflect actual database value
        - 1.1.3. System must be available as often as possible with little or no downtime
    - 1.2. Consistency
        - 1.2.1. Consistent sequence of actions for similar situations
        - 1.2.2. Consistent terminology
        - 1.2.3. Consistent visual layout
    - 1.3. Familiar context
        - 1.3.1. Leverage common interaction styles, display organization formats, and data entry methods that most users are familiar with to reduce time to learn, rate of errors, and retention over time

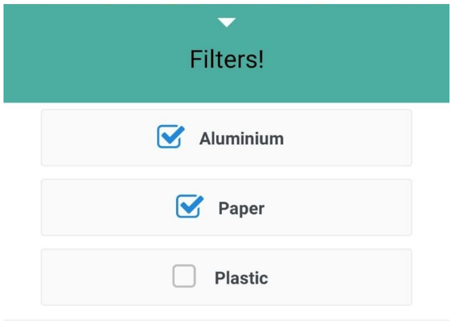
2. Performance Requirements
    - 2.1. The system must not crash for no reason
    - 2.2. The system must be able to render the map within 2 seconds
    - 2.3. When data is needed from the database, the system must be able to extract them and display it within 2 seconds

3. Extensibility Requirements
    - 3.1. The system must have an architecture that allows easy future enhancements/extensions

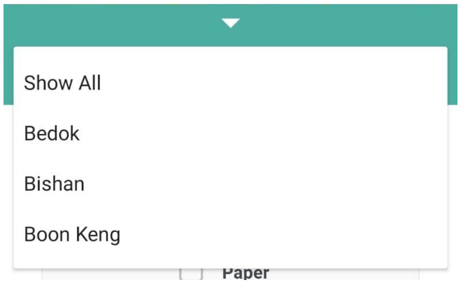
4. Interface Requirements

4.1 User

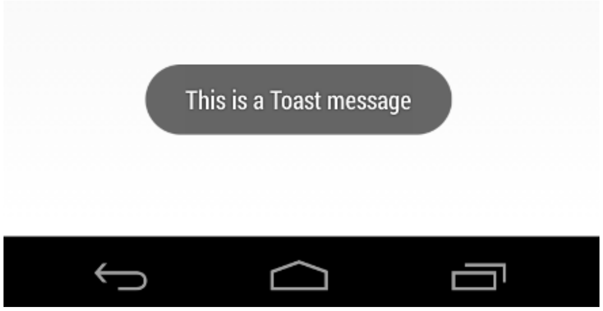
1. User interface only supports Portrait Orientation.
2. Use checkboxes when multiple options can be chosen.



3. Use scrollable dropdown list for mutually exclusive choice when there are many options are available.



4. Error messages are displayed using toast.


This is a Toast message

5. Software components used:
   - 5.1. Button - All Screens
   - 5.2. NavigationDrawer - All Screens
   - 5.3. ScrollView - All Screen
   - 5.4. Toast - All Screen
   - 5.5. TextInput - All Screens

5.6. Animated - Loading Screen
5.7. MapView - Map Screen
5.8. DropDownList - Map Screen
5.9. Marker - Map Screen
5.10. PopupDrawer - Map Screen
5.11. CheckBox - Map Screen
5.12. CustomPanel - Map Screen

4.2 Hardware

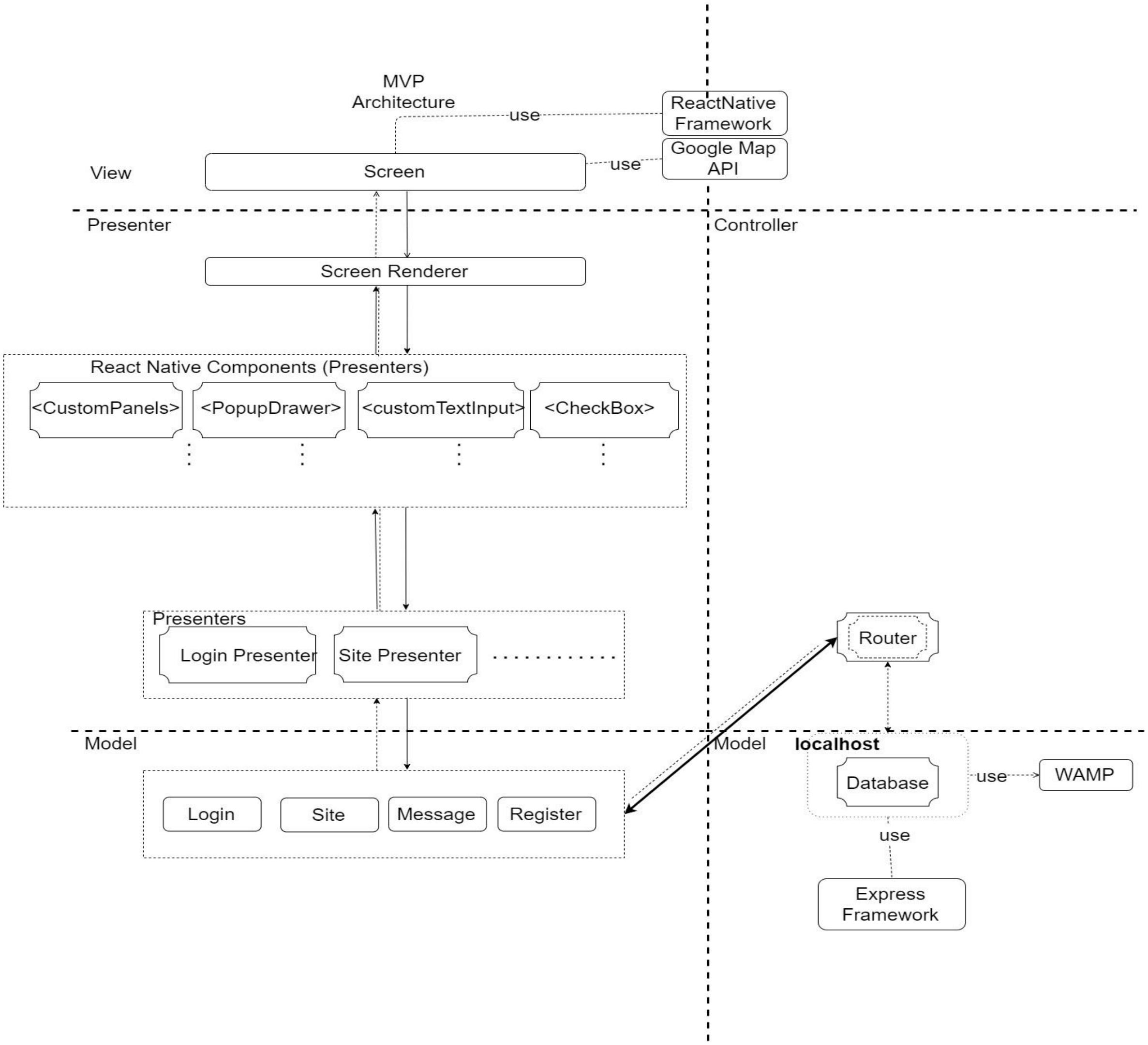
1. Android devices for running the application
2. Port 3000 is used as listening port for our database server.
3. Require the use of GPS receiver in the phone to get user location.

4.3 Software

1. Operating system required: Android 9 (SDK Platform 28) and above.
2. ReactNative for program structure.
3. Geolocation API to get user location.
4. MySQL as database.
5. WampServer64 as database server.
6. Express NodeJS as backend to our database server.
7. Lottie for animation.
8. Ionicons for all icons.
9. React-Native-Paper for design styling

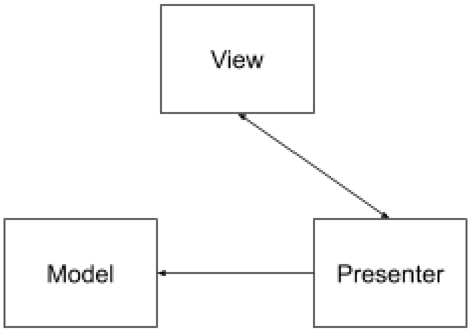
5. Architectural Design

5.1 System Architecture Diagram

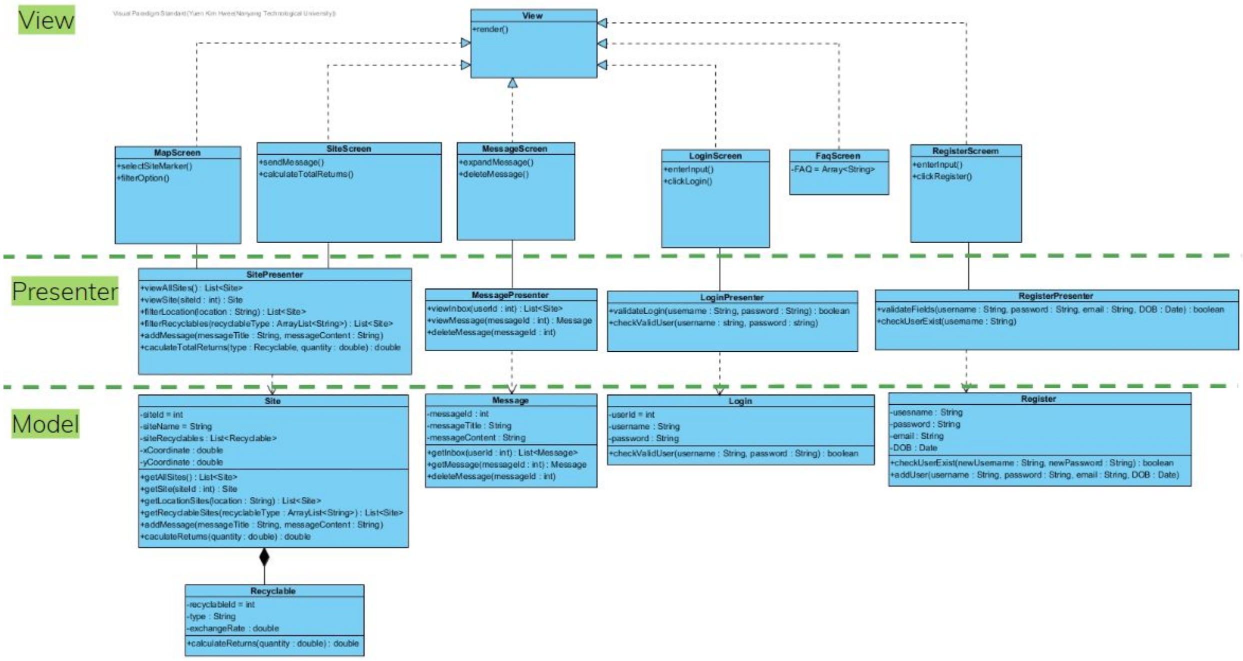


5.2 Design Pattern

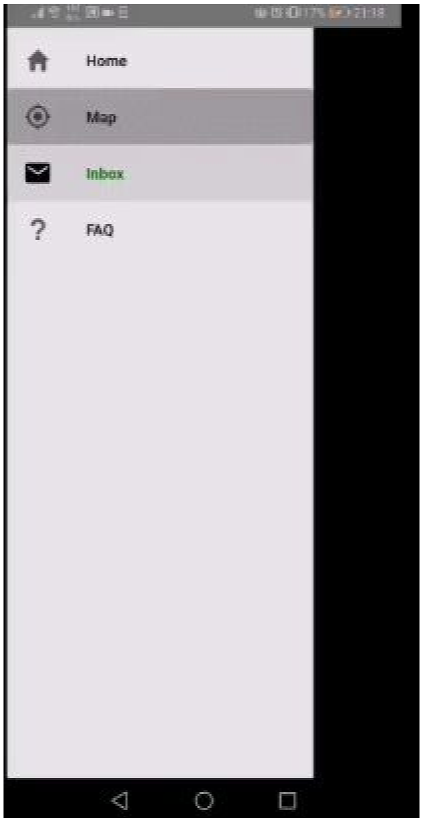
Model-View-Presenter

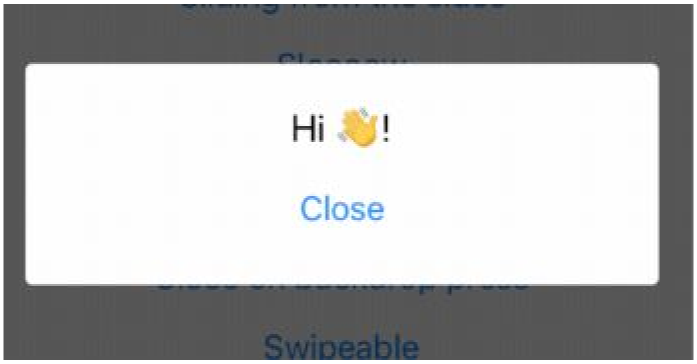


The Model is an interface defining the data to be displayed or otherwise acted upon in the user interface.
The View is a passive interface that displays data and routes user commands (events) to the Presenter to act upon the data.
The Presenter acts upon the Model and View. It retrieves data from the Model and formats it for display in the View.


View
Visual Paradigm Standard(Yuen Kim Hwee(Nanyang Technological University))
View
+render()
MapScreen
+selectSiteMarker()
+filterOption()
SiteScreen
+sendMessage()
+calculateTotalReturns()
MessageScreen
+expandMessage()
+deleteMessage()
LoginScreen
+enterInput()
+clickLogin()
FaqScreen
-FAQ = Array<String>
RegisterScreem
+enterInput()
+clickRegister()
Presenter
SitePresenter
+viewAllSites() : List<Site>
+viewSite(siteId : int) : Site
+filterLocation(location : String) : List<Site>
+filterRecyclables(recyclableType : ArrayList<String>) : List<Site>
+addMessage(messageTitle : String, messageContent : String)
+caculateTotalReturns(type : Recyclable, quantity : double) : double
MessagePresenter
+viewInbox(userId : int) : List<Site>
+viewMessage(messageId : int) : Message
+deleteMessage(messageId : int)
LoginPresenter
+validateLogin(username : String, password : String) : boolean
+checkValidUser(username : string, password : string)
RegisterPresenter
+validateFields(username : String, password : String, email : String, DOB : Date) : boolean
+checkUserExist(usename : String)
Model
Site
-siteId = int
-siteName = String
-siteRecyclables : List<Recyclable>
-xCoordinate : double
-yCoordinate : double
+getAllSites() : List<Site>
+getSite(siteId : int) : Site
+getLocationSites(location : String) : List<Site>
+getRecyclableSites(recyclableType : ArrayList<String>) : List<Site>
+addMessage(messageTitle : String, messageContent : String)
+caculateReturns(quantity : double) : double
Message
-messageId : int
-messageTitle : String
-messageContent : String
+getInbox(userid : int) : List<Message>
+getMessage(messageId : int) : Message
+deleteMessage(messageId : int)
Login
-userId = int
-username : String
-password : String
+checkValidUser(username : String, password : String) : boolean
Register
-usesname : String
-password : String
-email : String
-DOB : Date
+checkUserExist(newUsername : String, newPassword : String) : boolean
+addUser(username : String, password : String, email : String, DOB : Date)
Recyclable
-recyclableId = int
-type : String
-exchangeRate : double
+calculateReturns(quantity : double) : double

6. Data Dictionary

Recyclables	Recyclables are items that can be recycled, they include aluminium drinking cans, metal tins, papers, fabrics and small electrical appliances
Site	Site refers to a recycling station where people can exchange their recyclable for monetary returns. Not all sites collects the same products, or has the same operating days and hours.
User	A user is a person who is using the application..
Map Markers	Map Markers refers to red funnel-shaped markers that pin-points exact locations on a map. In this application, they represent the locations of the sites
Drawer	
Message	A message refers to a question sent by a user to a site
Information page	A page which contains the following: - Name of site - Operating days and hours - Types of recyclable accepted

	- Full address - Contact information - Estimate returns button - Send message button This page is what the users will see whenever click on a site's map marker on the map.
Form	Interface that requires user inputs / actions.
Page	Interface that only display information.
Toast	A pop-up message on the bottom of mobile screen to feedback user action that disappears after a short period of time.
Estimate returns button	A button that triggers a pop-up modal which allow user to calculate their estimated cash returns in terms of their input
Modal	 A pop up overlay display

7. Testing

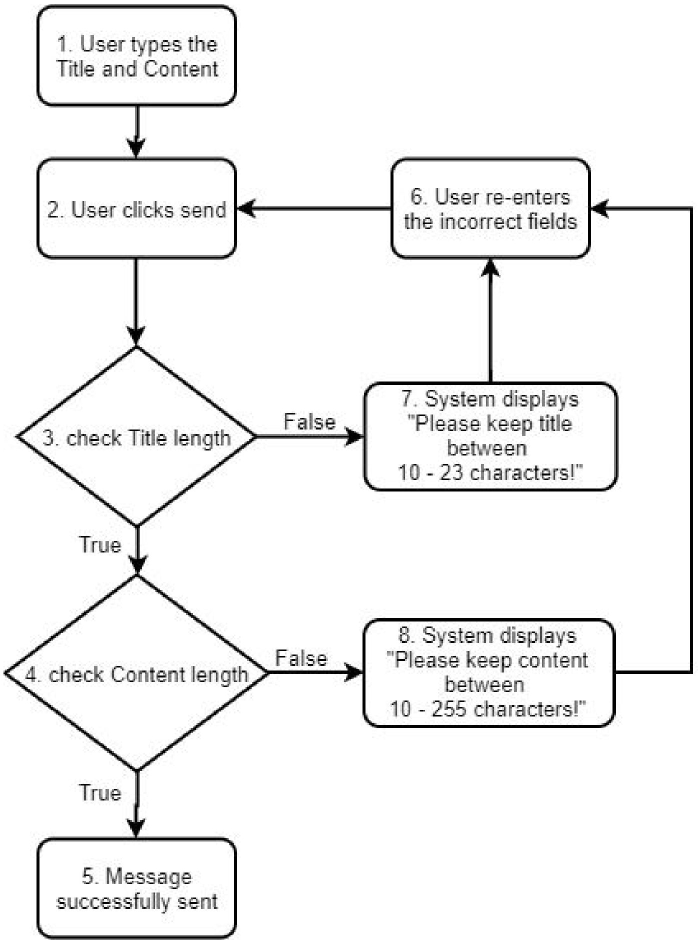
7.1 Black Box Testing

Range for number of characters for Title: 10 to 23
Range for number of characters for Content: 10 to 255

Title (length)	Content (length)	Expected Output	Actual Output
10	10	Accept	Accept
10	255	Accept	Accept
23	10	Accept	Accept
23	255	Accept	Accept
10	9	Reject	Reject
10	256	Reject	Reject
23	9	Reject	Reject
23	256	Reject	Reject
9	10	Reject	Reject
24	10	Reject	Reject
9	255	Reject	Reject
24	255	Reject	Reject

7.2 White Box Testing

Sending Message

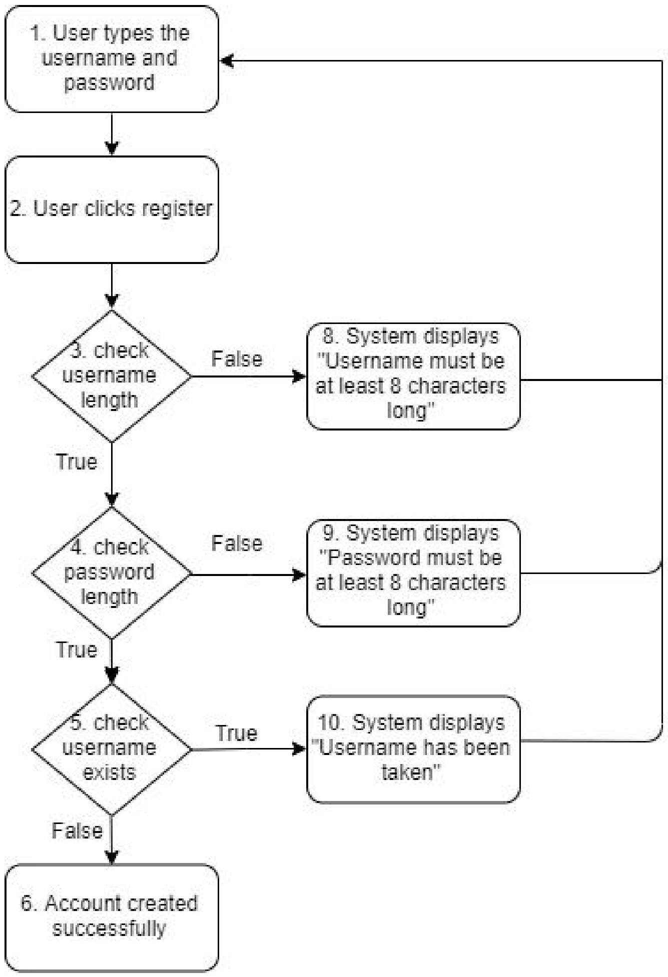

1. User types the Title and Content
2. User clicks send
3. check Title length
False
True
4. check Content length
False
True
5. Message successfully sent
6. User re-enters the incorrect fields
7. System displays "Please keep title between 10 - 23 characters!"
8. System displays "Please keep content between 10 - 255 characters!"

Basis Path	1,2,3,4,5	1,2,3,5,6,2,3,4,5	1,2,3,4,8,6,2,3,4,5

Test Input	Title.length = 11
	Content.length = 20
		Title.length = 26
		Content.length = 56
			Title.length = 15
			Content.length = 259

Expected Output	Message sent	Error message, and then message sent	Error message, and then message sent
Actual Output	Message sent	Error message, and then message sent	Error message, and then message sent

Register


1. User types the username and password
2. User clicks register
3. check username length
False
8. System displays "Username must be at least 8 characters long"
True
4. check password length
False
9. System displays "Password must be at least 8 characters long"
True
5. check username exists
True
10. System displays "Username has been taken"
False
6. Account created successfully

Basis Path	1,2,3,4,5,6	1,2,3,8,1,2,3,4,5,6	1,2,3,4,9,1,2,3,4,5,6	1,2,3,4,5,10,1,2,3,4,5,6
Test Input	Valid username and password	Invalid username length	Invalid password length	Username already exists
Expected Output	Account created	Error message, and then account created	Error message, and then account created	Error message, and then account created
Actual Output	Account created	Error message, and then account created	Error message, and then account created	Error message, and then account created

4.1 User Login

4.1.1 Description and Priority

Users can login with their Facebook account.
Priority: High

4.1.2 Stimulus/Response Sequences

Stimulus: A user enters a valid username and password to login.
Response: The user is authenticated and the application's main page is displayed.

Stimulus: A user enters either an invalid username or password.
Response: The system will display a message, "Wrong username or password, please try again".

4.1.3 Functional Requirements

1. Students and teachers must login to the system before the game can be accessed.
    - 1.1. The student and teachers must be able to login through several platforms.
        - 1.1.1. Students and teachers must be able to login with his/her Facebook account.
        - 1.1.2. Students and teachers must be able to login with his/her Twitter account.
        - 1.1.3. Students and teachers must be able to login with his/her Gmail account.
    - 1.2. The system must validate that the required fields are filled up.
    - 1.3. The system must validate that the login credentials are valid.
    - 1.4. The system must display an error message if the login credentials are invalid.