Software Requirements Specification

for

Stray Finder

Prepared by:

Chen Xingyu
Hrishikesh Harish Pai
Hu Zhuangyu
Jacintha Wee Yun Yi
Shivangi Krishna
Valencia Lie

Lab Group DSS3

17 April 2022

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document is primarily for the sake of providing a full in-depth description of Stray Finder – a mobile application designed to help volunteer organizations look out for and take care of stray cats in Singapore and also for members of the public to support this cause. This document outlines the system boundaries, interface, as well as communications used alongside an external API and how it interacts with our application. Also included is the project scope, an overall description, interface requirements, functional requirements as well as non-functional requirements.

Effectively, Stray Finder serves two main purposes:

1) The first purpose is to allow the user(volunteer) to attend to and access the information of injured cats. We want the user to use this app to ensure that as many cats as possible that are reported, are taken care of and brought to safety.

2) The second purpose is to allow the user(passerby) to report any cats that they come across (uninjured and injured). By doing this, more cats can be identified and addressed by the user(volunteer).

1.2 Intended Audience and Reading Suggestions

This document is meant for project managers, developers, testers and general users, as it provides a detailed guideline to the architecture of the project. It is highly recommended that project managers and developers read the entire document. Testers can use this document for building test strategies, and should refer to the testing process outlined in Section 6.5.

1.3 Product Scope

Stray Finder is a free app that can be accessed by two types of users, any member of the public (considered as user passerby) or a member of a volunteer group (considered as user volunteer). Depending on which type of user is accessing the app, the functionalities available vary.

Users(volunteers) have to login into their account using their email (unique per volunteer organization) and password for the account. If the account is not already present they can register a new account. When a user is registering for an account, the organization name they register under will be checked with all organization names present in our database. This is done to prevent duplicates and ensure that there is only one account for each organization. Once logged in, the

user(volunteer) has access to viewing all the injured cats in the apps database displayed on a noticeboard, view all the cats(injured and not injured) on the cat map, and access the vet function to locate all the registered vets in Singapore or filter the 5 closest vets in accordance to the user's current location.

There is no login requirement for users(passerby) and any member of the public can access the functionalities of this app directly by clicking on 'I am a passerby' on the very first screen. After they clicked on the button, the user(passerby) can access the "report a cat" function (both injured or uninjured), the list of several registered adoption agencies in Singapore, and lastly the same vet function that is available to the user(volunteer).

1.4 Users and stakeholders

The stakeholders for this project consist of volunteer organizations that are dedicated to the rescue and care of stray cats and Android and iOS mobile users of Stray Finder. Information and locations of the licensed vets will be collected through the application with the assistance of data.gov.sg API. Stary Finder strives to be a platform where users(passerby) can easily access information of organizations that offer adoption of cats, while also giving them the option to visit any nearby licensed vets according to their convenience by accessing their current location. The aim is to provide a smooth and seamless platform for digital interaction between passersby, who report stray cats to the app, and volunteers, who will be aiding in rescuing and bringing cats reported to the app to safety.

2. Overall Description

2.1 Product Perspective

Stray Finder is a self-contained mobile application that does not require external components. This project was first developed and deployed on an iPhone simulator. Its main responsibilities are to provide a means to report and locate injured stray cats, and store their records within the database. The usability of this application allows passersby and volunteers of animal welfare organizations in Singapore to better handle injured stray cats if they were to ever encounter one. It also provides a way for the general public to contact any registered pet adoption agencies that are compiled within the app in order to facilitate the adoption process (which are not affiliated to Stray Finder in any way).

2.2 Product Functions

Major functions of Stray Finder:

1) User(volunteer):
    - Login to App and logout of App
    - Register an account
    - Edit profile details (organization name, contact number and address)
    - Reset password via forgot password
    - View noticeboard of all the injured cats
    - Handle injured cats from the noticeboard
    - View cat map (display of all cats in the database, injured and non-injured)
    - View all the vets on the map or filter it to the five closets vets according to the users'(volunteer) current location

2) User(passerby):
    - Report a stray cat
        - Identify if the cat already exists in the database, if not, the user can name the cat since it's a new cat
        - Specify if a cat is injured, and if so, what the injury is along with pictures of it
        - Indicates if they are personally bringing the reported cat to the vet or not if the cat is specified as injured
        - Leave contact details if they are personally bringing the cat to the vet so that a volunteer can take over the case seamlessly
    - View a list of some of Singapore's registered pet adoption agencies
    - View all the licensed vets in Singapore on a map (data taken from govdata) or filter it to the five closest vets according to their users(passerby) current location

2.3 User Classes and Characteristics

Users of the software system include any public member willing to use the app (user(passerby)) and members of volunteer groups (user(volunteer)). All users should be familiar with how to install and use a basic mobile application. Users may not be fully knowledgeable of software-development or manufacturing processes. However, users should have a good understanding of the tasks and activities that they are interested in, and aim to complete and achieve via the use of this app.

2.4 Operating Environment

The Stray Finder application is built on Android Studio (Bumblebee and Chipmunk) and Visual Studio Code, and is able to run on both Android and iOS devices. Since the minimum API level used is API30: Android 10 as well as iOS 11, the app should be able to run on roughly 67% of all android devices and 100% of all iOS devices. The app will be coordinated with the photo gallery, camera and location services of the phone. It will also coexist with other applications as long as it is compatible with the operating system.

2.5 Design and Implementation Constraints

Database Used: Firebase Cloud Firestore and Cloud Storage

2.6 User Documentation

The target audience of this software system is the general public, and therefore does not require extensive training, nor need to read the documentation extensively in order to understand how to use this application.

2.7 Assumptions and Dependencies

Users should have a GPS-enabled Android and iOS device capable of running the application. Gallery and camera access are required for the report function available to users(passerby).
The app also needs access to the Internet in order for it to work.

3. External Interface Requirements

3.1 User Interface

Stray Finder focuses on two different categories of users: volunteer group members and the general public who are considered as passersby. The user(volunteer) can use the app to view the list of all injured cats, access a map that displays all the cats stored in the app's database, as well as a map to locate all licensed vets in Singapore or those that are close to their current location. The user(passerby) can use the app to report cats and ascertain if they are injured, access information about organizations or groups that allow in adopting cats, as well as the same map available to the user(volunteer).

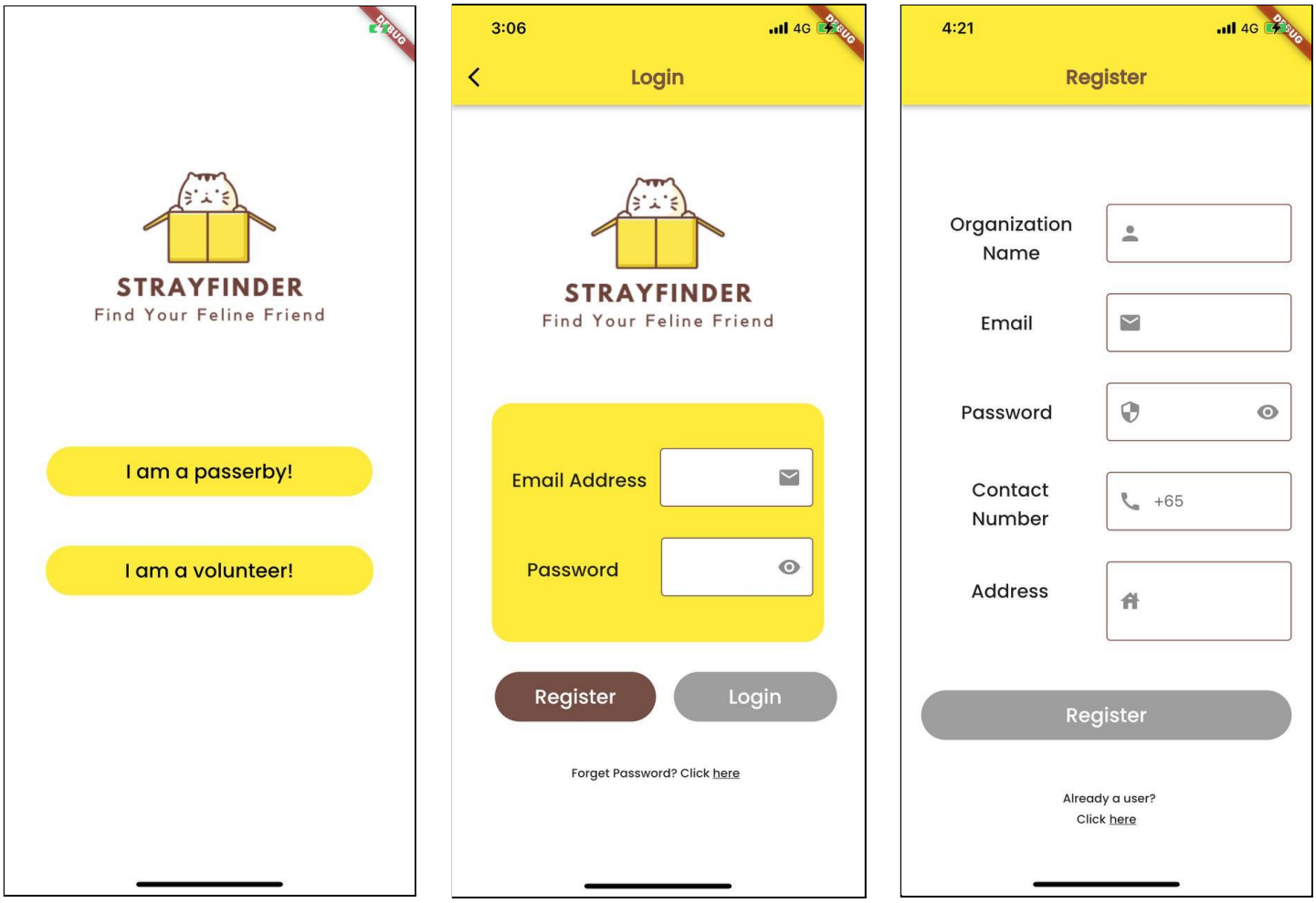


Figure 1: Login and Registration pages

Figure 1 shows the user interface for login and registration. The user interface is designed in such a way that the format and layout is simple and easy to follow for the user(volunteer), similar to other app applications. This allows the user to follow the instructions and login or register on the app with ease. The app is also equipped with a forget password function that allows the user (volunteer) to reset the password of their account by entering their registered email address (refer to Figure 2). An email will then be sent to the email address, allowing the user(volunteer) to reset their password through an external link (refer to Figure 3)

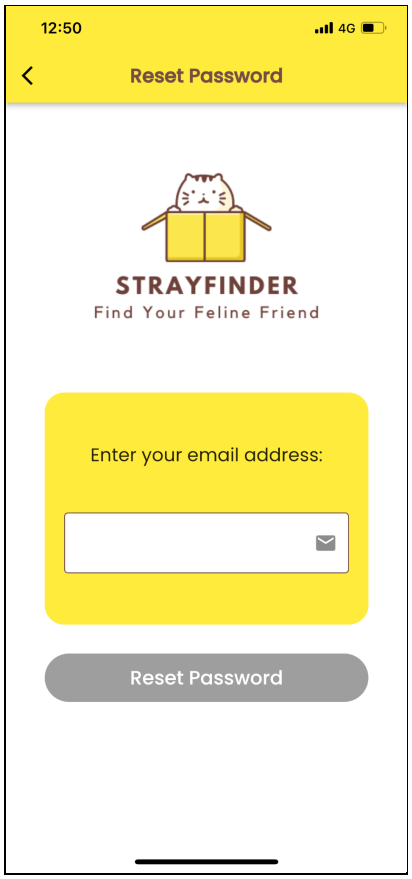


Figure 2: Forget password page

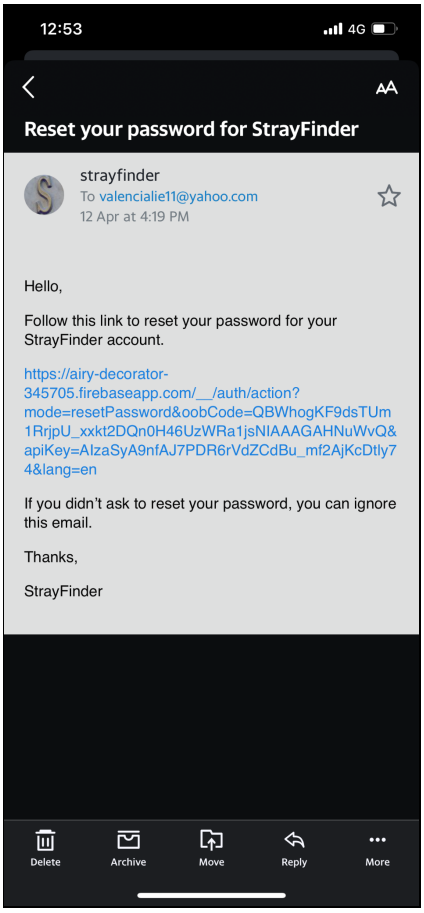


Figure 3: Forget password instruction sent through email

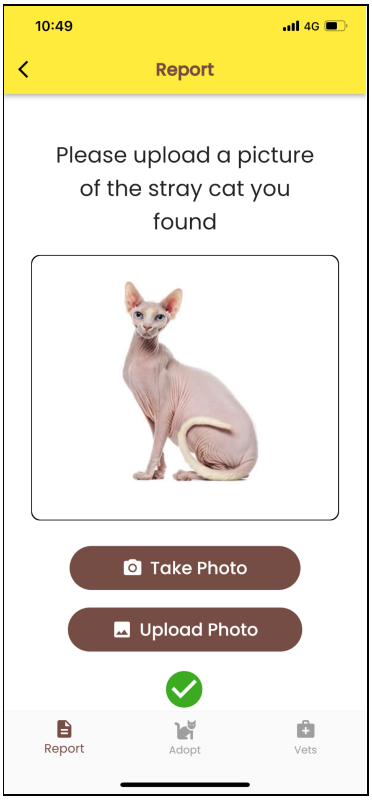


Figure 4: Starting page of the report function, prompting the user to upload a picture of a cat he/she found

Figure 4 shows the starting page of the report function which can only be accessed by the user(passerby). When using this function, the user has the option to either upload a picture (where they have to enable access to the photo gallery) or take a photo using their camera (where they have to enable access to the camera).
Once the picture is confirmed to be of a cat picture, the app predicts the breed of the cat in the picture and displays pictures of all the cats with the same breed in our database.

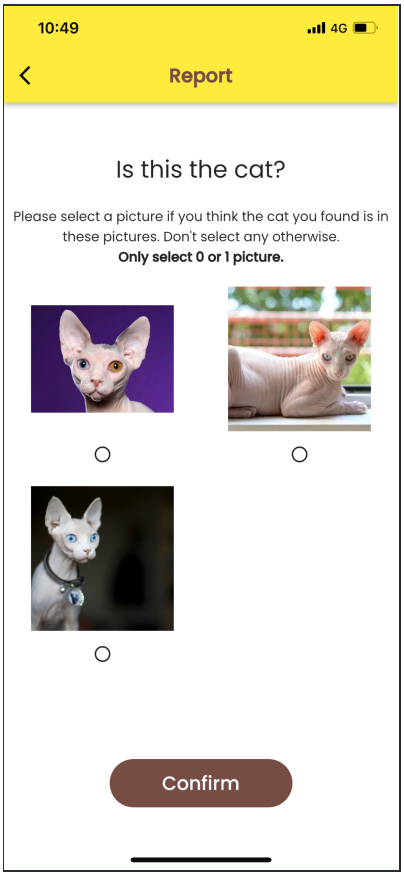


Figure 5: A page showing all cats in our database that have the same breed as the reported cat

The user(passerby) will then be prompted to identify if any of the cats displayed is the same as the cat he/she uploaded.

If he/she indicates that the cat is a newly found cat (meaning none of the cats displayed is the same cat as the one reported), he/she would have to name the cat and the page below is displayed:

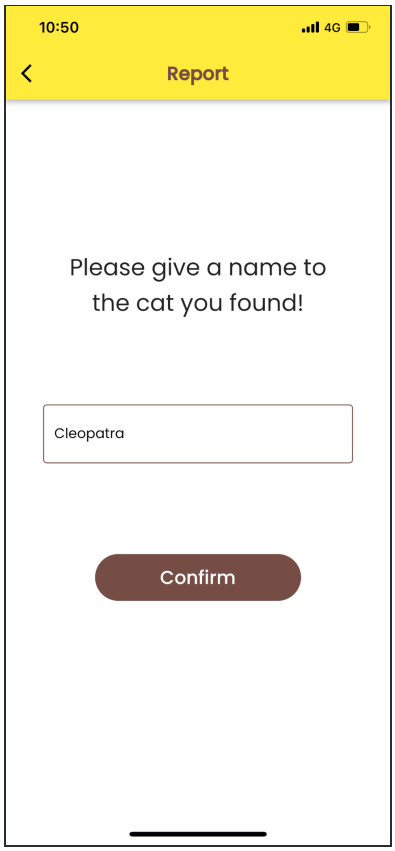


Figure 6: A page prompting the user to name a newly found cat

Afterwards, they are prompted to either enable access to their current location or to manually input a location:

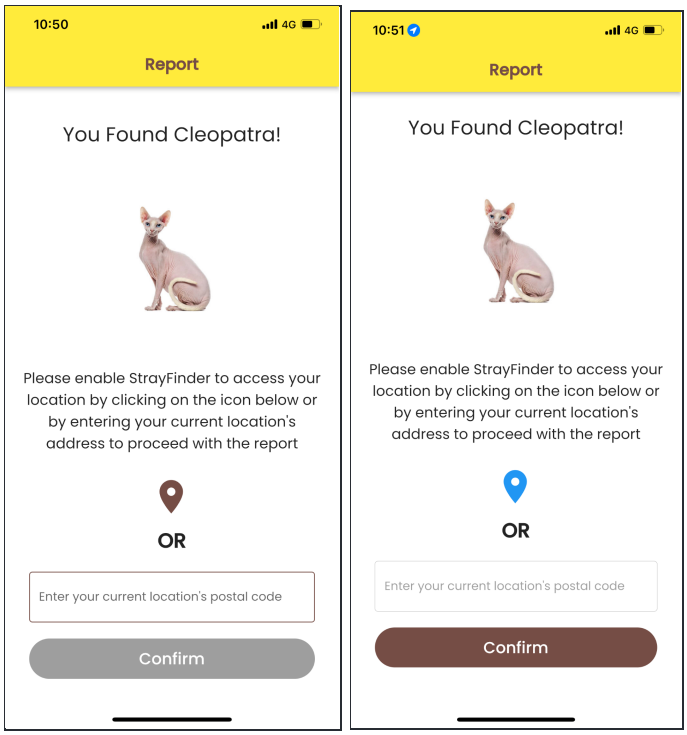


Figure 7: Ask user to give access to location

Next, the user(passerby) is prompted to specify if the cat reported is injured or not.

Figure 8: Prompt the user to specify if the cat reported is injured

If the user says the cat is injured, the page below is displayed:

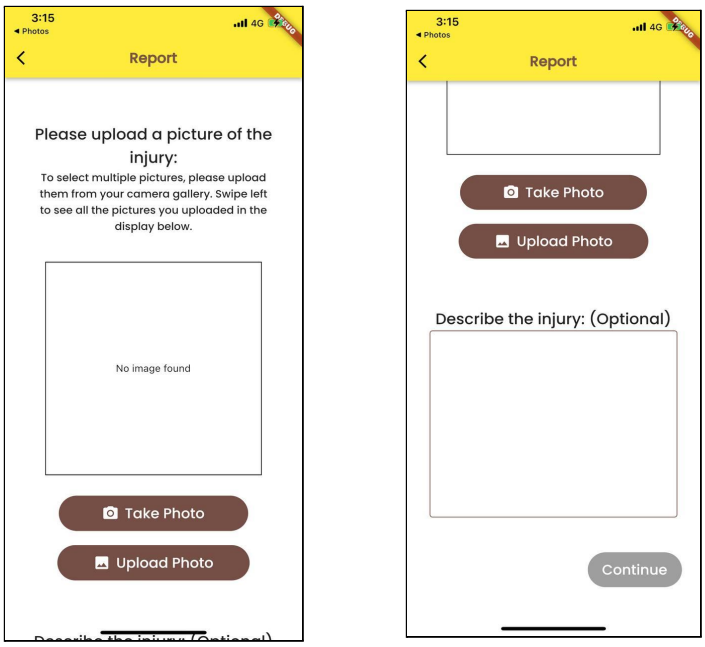


Figure 9: Report injuries of a cat pages

Figure 9 shows the pages that are displayed when the user(passerby) selects the option saying the cat reported is injured. This page can only be accessed by the user(passerby) and it is a continuation of the report page. The user has to upload a picture of the cat's injury, either using their camera or uploading from their photo library. Multiple pictures are allowed to be uploaded via the photo gallery. The description of the injury is optional. Once the required entries have been made, the grey "Continue" button will change to brown, indicating that the user can now proceed with their submission.

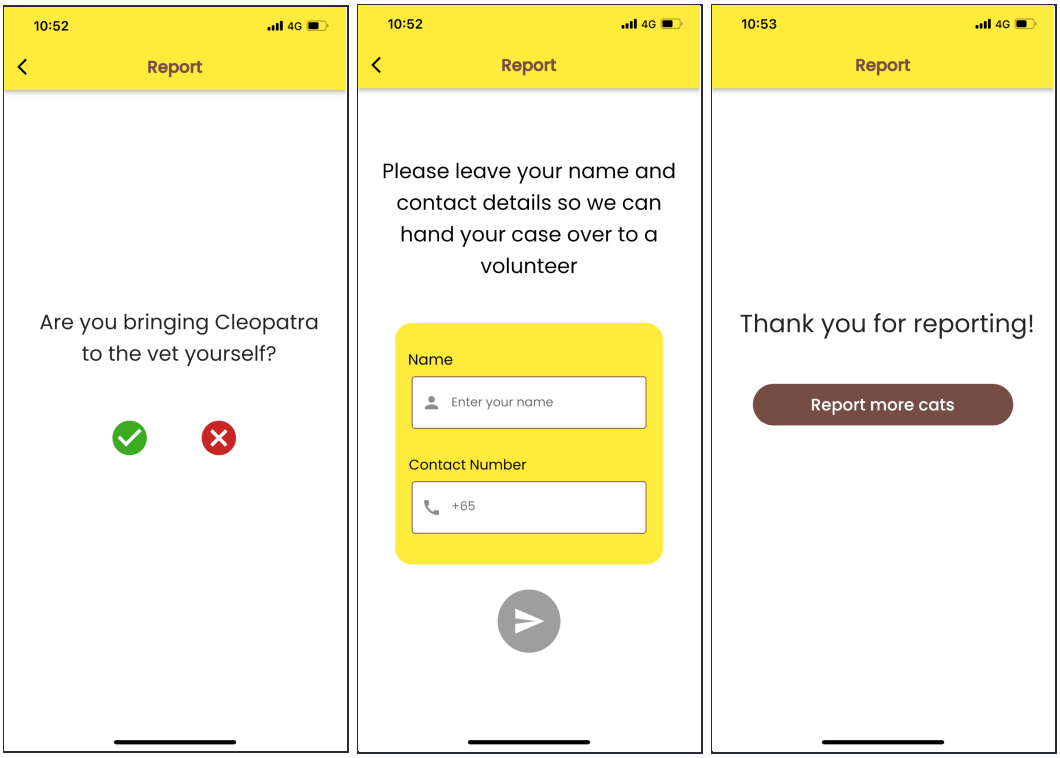


Figure 10: Page asking user to leave contact details and name if they choose to bring the injured cat to the vet themselves as well as the thank you page at the end of the reporting process

Figure 10 shows the page that questions the user(passerby) if they want to take the cat to the vet themselves, as a continuation of the report page. If the user chooses to take the cat they are prompted to enter both their name and contact number in order to continue, so that volunteers can take over the injured cat case from the passerby once the injury is reported.
Once completed, a thank you message will be displayed at the end of the reporting process. This same message will be displayed if the user(passerby) had initially said that the cat had no injuries, or even if they clicked no for taking the cat to the vet themselves.

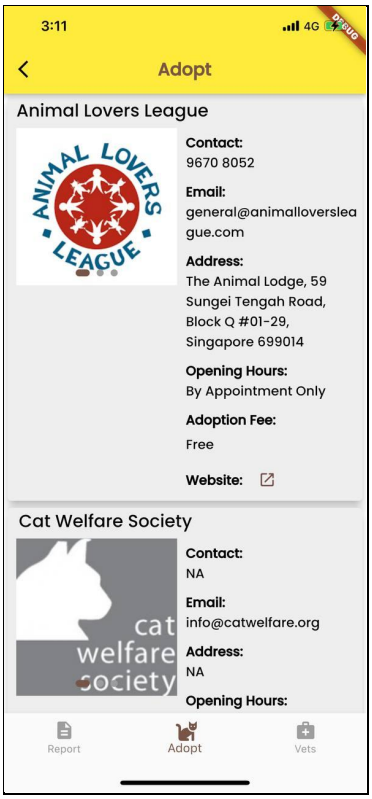


Figure 11: Adopt page that displays the list of some adoption agencies in Singapore

Figure 11 shows the adopt page which shows a static list of organizations that have cat adoption as an option. This function and page can only be accessed by the user(passerby). The user can view more organizations by using the scroll down option. When the user clicks on the website button, they will be directed to the website of that respective organization.

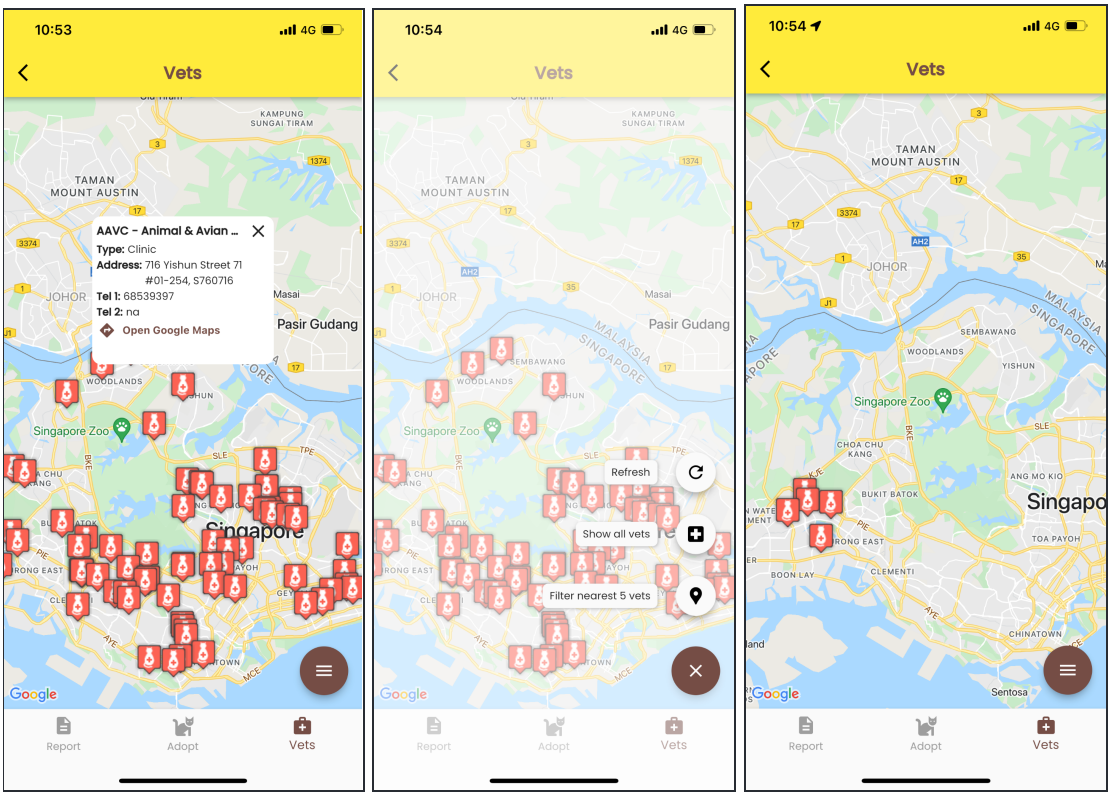


Figure 12: Vets page that displays licensed vets from the data.gov.sg API which features a filter function

Figure 12 shows the map that displayed all the licensed vets. This map function can be accessed by both user(volunteer) and user(passerby). The red drop pins show all the vets, however clicking on the floating action button followed by the 'Filter Nearest 5 Vets' button will filter the vets to display only 5 of the closest vets according to the user's(volunteer & passerby) current location. Clicking on a red marker displays the information of that specific vet, such as the type of vet, address, telephone number. Clicking on 'Open Google Maps', directs the user to Google maps where the direction, path and estimated time to reach this specific vet will be displayed.

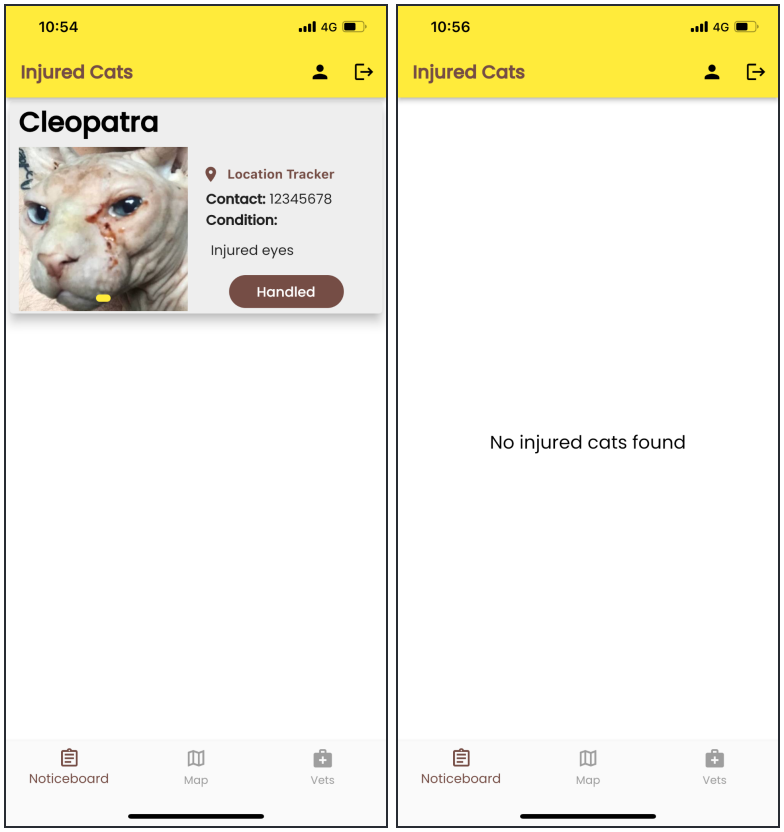


Figure 13: A page that displays all injured cats in the database. Cat details disappears from the page once the handled button is pressed

Figure 13 shows the noticeboard of all injured cats in our database that can only be accessed by the users(volunteers). The noticeboard features a list of all injured cats in our database, including a picture of the cat's injuries, the name and the condition of the cat, as well as the name and contact details of the passerby if the passerby decides to bring the cat personally to the vet. If the user (volunteer) clicks on the handled button, the cat is no longer considered as injured and the cat will disappear from the noticeboard. There will also be a location tracker button that allows the user(volunteer) to see the last seen location of the cat on a map (refer to figure 12).

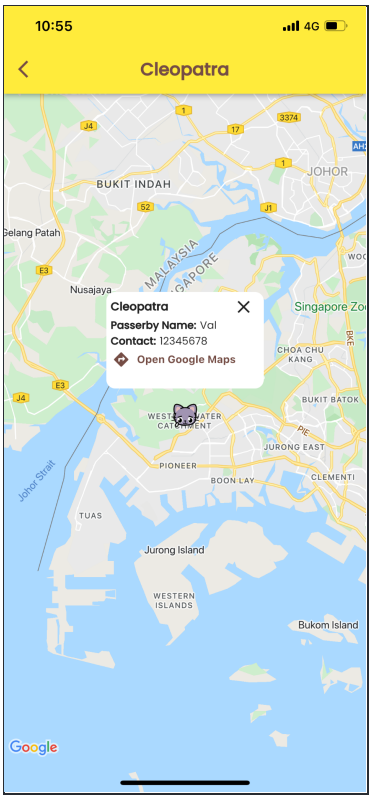


Figure 14: A map that displays the last seen location of the cat when the location tracker is pressed

Figure 14 shows a map of the last seen location of the cat. When the marker is pressed, an information window will pop up and show additional details about the cat such as its name, and the name and contact details of the passerby who reported the cat (if applicable). There will also be a button that allows the user(volunteer) to navigate to the location of the cat via Google Maps.

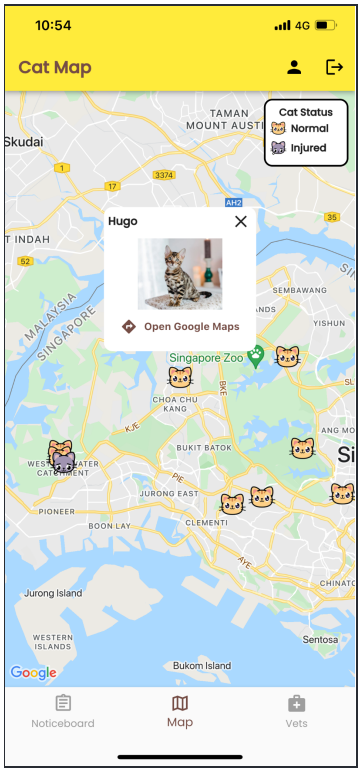


Figure 15: A map that displays the last seen location of all the cats in the database

Figure 15 shows a map of the last seen locations of all the cats in our database. The differing markers indicate different injury status of the cat: purple marker for injured cats and orange marker for uninjured cats. When the marker is pressed, an information window will pop up and show additional details about the cat such as its name and its picture. There will also be a button that allows the user(volunteer) to navigate to the location of the cat via Google Maps.

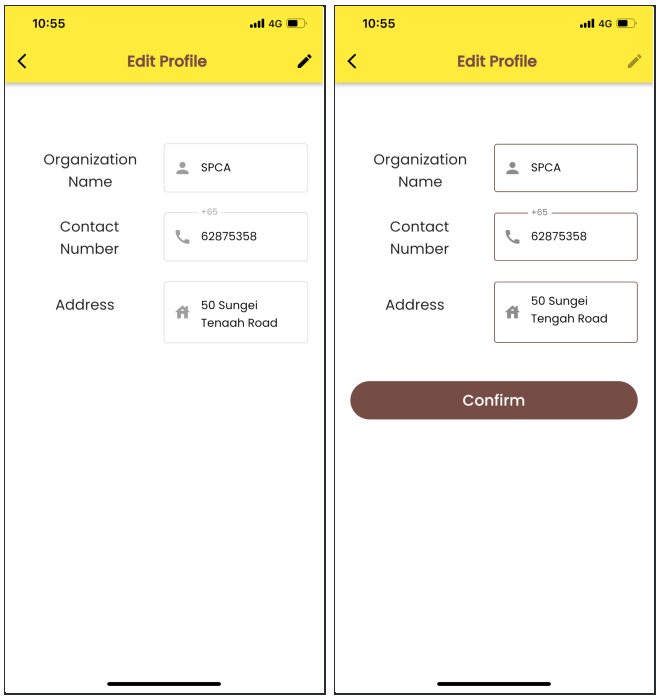


Figure 16: A page that enables the user(volunteer) to edit their profile

Figure 16 shows a screen where the user(volunteer) is able to edit their profile such as their organization name, their contact number and physical address.

3.2 Hardware Interfaces

Stray Finder is required to work on the hardware devices with location services enabled to locate the users'(volunteer & passerby) current location. The current location of the user(passerby) will be used to update the cats' last seen location on the "Cat map". The current location of the user(volunteer & passerby) will be used to filter the vets map to the 5 closest ones. The hardware devices are also required to enable camera and photo gallery access for the report function available to the user(passerby).

3.3 Software Interfaces

Stray Finder is being developed so that it can be used on both Android and iOS devices.

APIs used in the application:

1) data.gov.sg API - list of licensed vet centers to be shown on the Vet Map
2) Google map API - to integrate map functionalities in our app
3) Breed Predictor API - to predict the breed of the cat image input by the user(passerby) in the report function
4) Object Classification Model - to ascertain if the input image by the user(passerby) in the report function is that of a cat. If the image is not that of a cat, an error message is shown and the user(passerby) is required to input an image of a cat

5) Firebase Authentication - to store, retrieve and update password associated with user(volunteer) account
6) Firebase Cloud Firestore - to store, retrieve and update information associated with stray cats, injuries and user(volunteer)
7) Firebase Cloud Storage - to store, retrieve, update and delete the stored cat pictures and injury pictures.

3.4 Communications Interfaces

Stray Finder will be accessed as a mobile application. All the features mentioned in the app will be accessed via the application.

- Communication with the APIs uses HTTPS protocol.
- Flutter provides libraries with HTTPS native support.

4. System Features and Functional Requirements

4.1 Onboarding to Application

1.1 The application must allow the user(passerby) direct access to the main functions of the app if the user indicates he/she is a passerby.

1.2 The application must request the user(volunteer/passerby) for permission before accessing the user's location.

1.2.1 If the user denies access, should the user want to use any location enabled features within the application, he/she must provide a location manually.

1.2.2 If a user grants access, all location enabled features should automatically retrieve the user's current location via GPS.

1.3 The application must request the user(passerby) for permission before accessing the user's mobile phone camera and photo library.

1.3.1 If user denies access, any features which require camera access must display a warning to the user and further access to the feature will be denied.

1.3.2 If user grants access, any features which require camera access will automatically open up the camera/photo library.

4.2 Registration and Login

2.1 The application must prompt the user(volunteer) for the registered email, which is unique to the volunteer organization. The volunteer organization must share one account for all members, which is used for login.

2.2 If the user(volunteer) forgets the password of the account, he/she will be able to reset the password by providing their registered email address to the app under the 'Forgot Password' button. The app will send an email and provide an external link on how to reset password to their email address.

2.3 User(volunteer) must be able to logout after successful login.

2.4 When a user(volunteer) registers for a new account, he/she must input the organization name, address, contact number, email and password to successfully complete registration. If the organization name or email provided already exists in the database, the user(volunteer) must be prompted again to input a different organization name/email.

4.3 Profile Customization

3.1 After login, the user(volunteer) shall be able to customize his/her profile.

3.2 The user(volunteer) must be able to only edit:

3.2.1 Organization name

3.2.2 Organization contact number

3.2.3 Organization address

3.3 The user(volunteer) must fill in all text fields when editing his/her profile for successful completion of profile update.

3.4 The user(volunteer) must adhere to the app's guidelines for editing of profile, mainly:

3.4.1 Organization contact number must be exactly 8 digits, no more and no less.

4.4 Cat Reporting

4.1 When the user(passerby) submits a photo of a stray cat, the photo must be first analyzed to determine if the image is that of a cat.

4.1.1 If the image is not that of a cat, the user will be prompted to resubmit a photo of a cat.

4.2 The photo must then be analyzed to identify the breed of the cat through a pre-trained machine learning model.

4.2.1 The application must display all the cats of the same breed which currently exist in the database and must prompt the user to match the reported cat to one or none of the displayed stray cats.

4.2.1.1 If the user cannot find a match, the cat shall be considered a newly found stray cat.

4.2.1.2 If the user finds a match, the cat is an existing stray cat in the database.

4.2.1.3 If the user selects more than one of the displayed cats, the app must warn the user that he/she is only permitted to select either 0 or 1 of the cat images displayed.

4.3 When the cat exists in the database, the application must update the last seen location of the stray cat.

4.4 When the cat does not exist in the database, the application must prompt the user(passerby) to input details of the stray cat where applicable:

4.4.1 Name

4.4.1.1 Users are reminded to ensure that no rude words are allowed as names.

4.4.2 Last seen location of stray cat

4.4.2.1 If location services are enabled, the last seen location of the stray cat must be automatically updated to be the current location of the user.

4.4.2.2 If location services are disabled, the last seen location of the stray cat must be manually provided by the user in the form of an address.

4.5 The user(passerby) must confirm if the stray cat found is injured or not.

4.5.1 If the cat is injured, the user(passerby) must provide details of the injury including photos of the injury and an optional description of the injury.

4.5.2 The user(passerby) must confirm if he/she is going to bring the injured cat to the vet.

4.5.2.1 If the user(passerby) is going to take the injured cat to the vet, he/she must leave his/her contact details including name and contact number.

4.5.2.2 If the user(passerby) is not going to take the injured cat to the vet, he/she shall not need to leave his/her contact details and a thank you message must be shown.

4.5.3 If the cat is not injured, a thank you message must be shown to the user(passerby) as closure.

4.5 Cat Map

5.1 The application must display last seen locations as well as brief descriptions of all the stray cats in the database when the user(volunteer) clicks on any of the map markers shown in the map, such as:

5.1.1 The cat's name

5.1.2 The cat's image

5.1.3 A button to navigate via Google Maps to the last seen location of the cat

5.2 Different markers shall be used to indicate injury status of the cats:

5.2.1 Purple cat icon must indicate an injured stray cat

5.2.2 Orange cat icon must indicate an uninjured stray cat.

4.6 Injury Management

6.1 When the user(passerby) files a report of an injured stray cat, the application must prompt the user to write a description of any visible injuries and upload a picture of said injuries.

6.1.1 The description of the injury shall be optional.

6.1.2 A picture of the injury must be uploaded to proceed to the next page.

6.2 The application must display all reports of injured stray cats in the injury noticeboard to aid volunteer groups in tending to them.

6.2.1 If user(passerby) does not want to send the injured stray cat to the vet:

6.2.1.1 No name and contact details shall be required from the passerby who reported the case.

6.2.1.2 If a particular case has been handled by a user(volunteer), it must automatically disappear from the noticeboard of injured stray cats.

6.2.2 If user(passerby) wants to send the injured stray cat to the vet:

6.2.2.1 Contact number and name of passerby must be taken in as input for the user(volunteer) to liaise with the user(passerby) directly.

6.2.2.2 After the case is taken over by a user(volunteer), it must automatically disappear from the noticeboard of injured stray cats.

4.7 Vet Clinics

7.1 The application must display detailed information about the vet when the user(volunteer/passerby) clicks on any of the map markers shown in the map, such as:

7.1.1 The vet's full name

7.1.2 The vet's type

7.1.2 The vet's full address

7.1.3 The vet clinic's telephone number(s)

7.1.4 A button to navigate to the location of the vet via Google Maps

7.2 The application must allow the user(volunteer/passerby) to filter to the five nearest vets in accordance to their current location

4.8 Adoption

8.1 The application must display detailed information about existing adoption centers in Singapore to aid the user in enquiring for adoption, such as:

8.1.1 The adoption agency's full name

8.1.2 The adoption agency's telephone number

8.1.3 The adoption agency's email address

8.1.4 The adoption agency's address

8.1.5 The adoption agency's opening hours

8.1.6 The adoption agency's adoption fee (if applicable)

8.1.7 The hyperlink to the adoption agency's website

4.9 Interface with Other Systems

9.1 The application must be able to use GPS to track user's(volunteer/passerby) current location when submitting reports of stray cats.

9.2 The application must be able to use Wi-fi to communicate all actions executed with Firebase.

9.3 The application must be able to retrieve data on locations of licensed vet clinics in Singapore through data.gov.sg API.

9.4 The application must be able to use Google Maps API to show locations of all cats in the database, unless cat is removed from the database.

5. Other Nonfunctional Requirements

5.1 Usability – Adherence to Schneiderman's 8 Golden Rules

All usability requirements comply with the guide according to the Schneiderman's Eight Golden Rules:

5.1.1 Strive for consistency
5.1.2 Cater to universal usability
5.1.3 Offer informative feedback
5.1.4 Design dialog to yield closure
5.1.5 Permit easy reversal of actions
5.1.6 Support internal locus for control
5.1.7 Reduce short-term memory load
5.1.8 Prevent errors

5.1.1 Strive for consistency

Consistent sequence of action should be applied when users are in a similar situation. The app design uses many consistent buttons, fonts, actions and navigation bars in all the pages of the app. As seen in Figure 7, the navigation bar for user(passerby) is consistently that of "Report", "Adopt" and "Vets" in that fixed order whereas for user(volunteer) the order is "Noticeboard", "Map" and "Vets". Hence users can very easily find functions that they require and it takes a small amount of time to get used to navigating the app. Similar components are also used in the Map page so that users(volunteer & passerby) will feel familiar when using the map for the first time, such as using red drop pins to denote the location of a vet clinic. This further strengthens the consistency of the app.

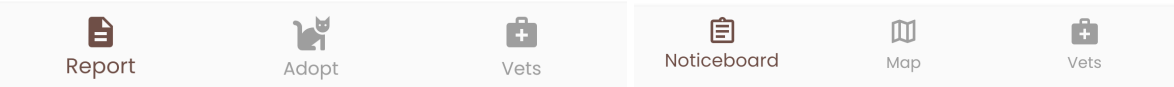


Figure 7: Bottom navigation bar for: user(passerby) (left) and user(volunteer) (right)

5.1.2 Cater to universal usability

The app design is simple and intuitive to use. There are not many pages on the app, and they can all be accessed using the bottom navigation bar. This allows most users(volunteers and passersby) to use the app easily. Since the design of the button and working mechanisms we implemented are similar to common apps used, users will not have an issue adjusting to the design of this app. For example, familiar format and arrangement for filling in the required fields for the login page including fields for email and password.

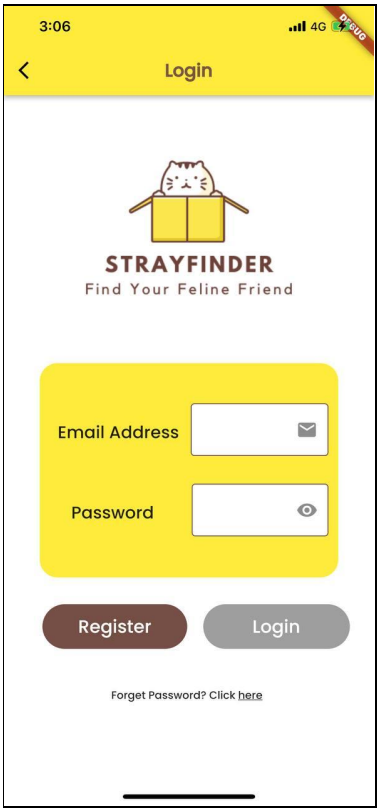


Figure 8: Login page

5.1.3 Offer in informative feedback

Depending on the actions performed by the users(volunteer & passerby), different informative feedback will be displayed to the user. This allows users to be aware of how their interactions with the app has yielded the appropriate functions to be performed by the app. For example in Figure 9 a confirmatory thank you page is shown to the user(passerby) after completion of the report function.

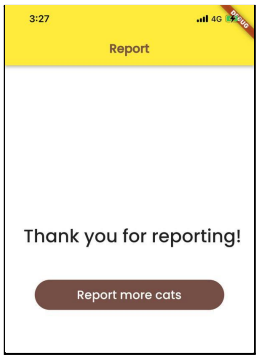


Figure 9: Confirmation message display after successful reporting

5.1.4 Design dialog to yield closure

Sequences of actions should be organized into groups with beginning, middle and end. The user is prevented from performing any unwanted actions, for example, when the user(volunteer) is logging in, the user cannot press on the "Login" button until all the required fields and details (Email Address and password) have been entered. Only after the correct details are keyed into all text fields can the user(volunteer) log into the app and access its functionalities.

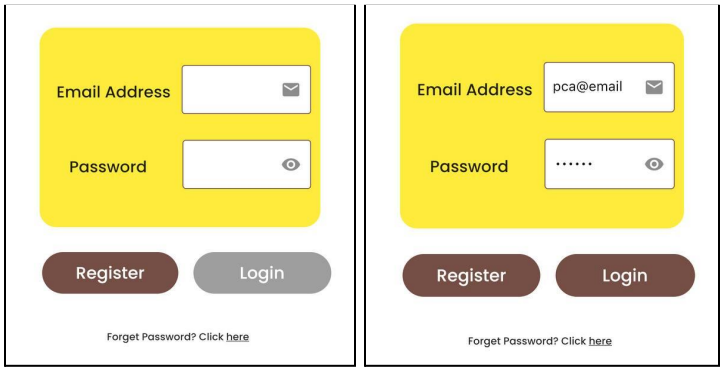


Figure 10: Login page entries

5.1.5 Permit easy reversal of actions

This feature relieves anxiety on users as they know an error can be undone. For example, if a user(passerby) by mistake clicks "I am a volunteer", they will be directed to the login page as seen below in Fig 11(a). By pressing the back arrow on the top left-hand corner of the page, they can return to the homepage and change their option to "I am a passerby", as seen in Fig 11(b). When the user(volunteer & passerby) wants to change the feature they are using, e.g. "Vets", they can easily reverse their action by clicking on a different feature available on the bottom navigation bar, as seen in Figure 12.

Figure 11: Reversal of action from login page back to homepage of StaryFinder

Figure 12: Navigation bar on user(passerby) page. All function options available displayed

5.1.6 Support internal locus of control

Experienced operators strongly desire the sense that they are in charge and the system responds to their actions. When a user(volunteer) is logged in, they have the freedom to change their account setting any time, e.g. organization name, contact number and address as seen in Figure 13. Thus, all actions that can be performed on the app will be based on how the user interacts with the app. The system will respond according to the type of actions they performed.

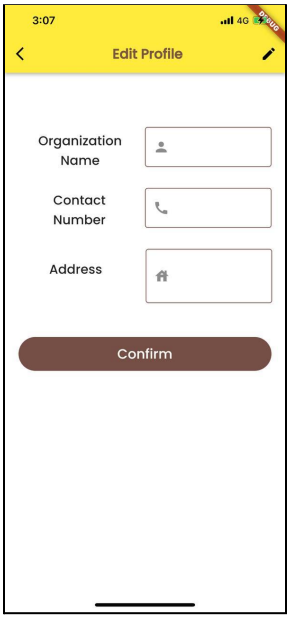


Figure 13: Edit profile page that is accessible to users(volunteer)

5.1.7 Reduce short-term memory load

Human information processing in short term memory is limited. We need our app design to be simple and for multiple page displays to be consolidated. This prevents the app from displaying too much information and overwhelming the user. For example, the report function made available to users(passerby) will sequentially reveal more questions to be answered by the user (e.g. ask if the cat is injured, ask if the user is bringing the injured cat to the vet). This is done to prevent overwhelming the user(passerby) by asking too many questions at once. Furthermore, as seen in Figure 7, only three tabs are made available per type of user in the bottom navigation bar, and these three tabs encapsulate the necessary functions that each user requires. This makes it easier for each user to quickly navigate the functions they need without having to remember too many steps.

5.1.8 Prevent errors

The app is designed such that users cannot make a serious error. If an error is made, a message will pop up telling them about the error made and what they should do to correct the error. Informative messages like these can be done to help reduce the number of errors a user can perform. For example, if the user(volunteer) enters an incorrect email during login in, an error message will pop up on the screen prompting the user(volunteer) to try again with a valid email address. There is exception handling in place to catch errors and provide feedback for the user to enter the correct input.

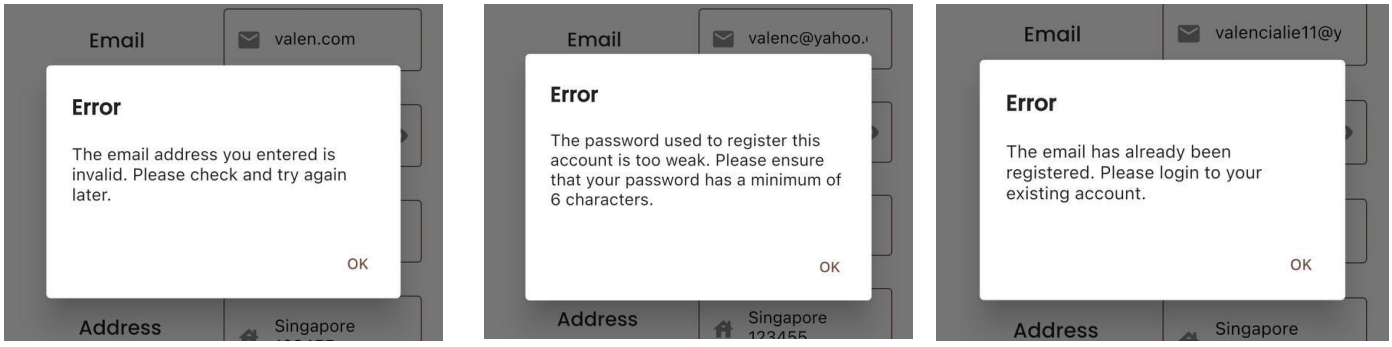


Figure 14: Different types of error messages displayed

5.2 Reliability

5.2.1 All information displayed must be accurate as of last updated information from the database.

5.2.1.1 For example, in the report function when the user(passerby) finds an existing cat and there is a change to its injury status to that of being injured, the change is registered by the StrayCatMngr and reflected in the 'cats' collection of the Cloud Firestore database. Once the injury of the cat is added to the 'injuries' collection of the Cloud Firestore database as well as corresponding photos in the Cloud Storage database, the injured cat markers (denoted by purple cat icon) can be seen on the cap map for users(volunteer). Under the injury noticeboard for users(volunteer), the injured cat information will be simultaneously added too. This shows the capability of the app to reflect changes to both types of users instantaneously so that information is as accurate as of recent updates.

5.3 Performance

5.3.1 The application must be able to retrieve/upload data for not more than 20 seconds.

5.3.2 The application must be able to update the location of the cats in the database or capture location of a new cat reported via GPS.

5.3.3 The application must be able to update any new injured cats to the cat map or noticeboard in real-time.

5.4 Supportability

5.4.1 The database will be deployed to a cloud platform: Cloud Firestore and Cloud Storage.

5.4.2 The application can be downloaded on Android and iOS systems.

5.5 Security

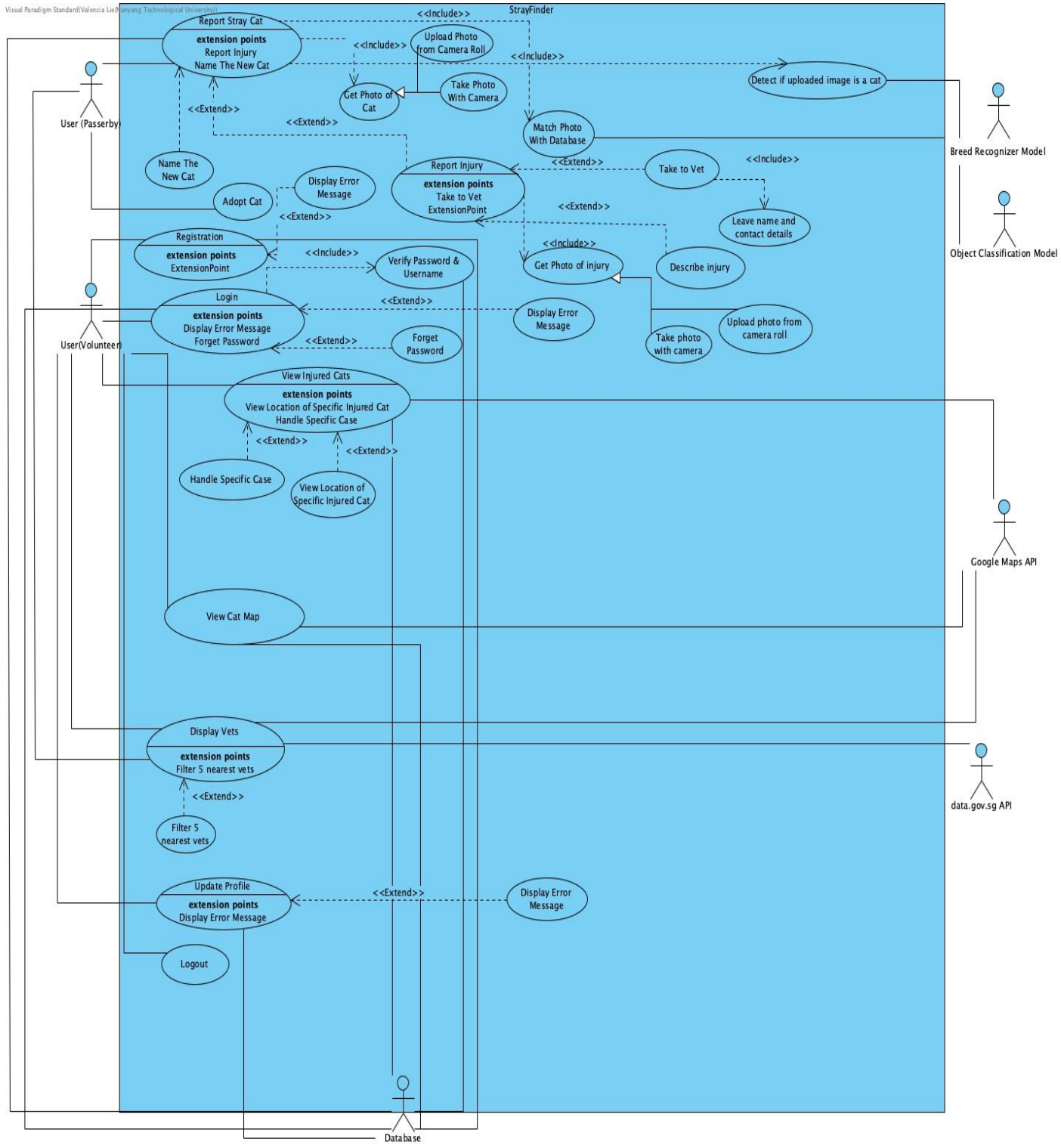
5.5.1 The application will perform encryption on newly created passwords during registration using Firebase's authentication service.

5.5.2 During user login, the encrypted passwords stored in the database will be matched with the password provided as input by the user.

5.5.3 Password field will be masked to protect user privacy, but can be unmasked if the user chooses to do so.

6. Use Cases

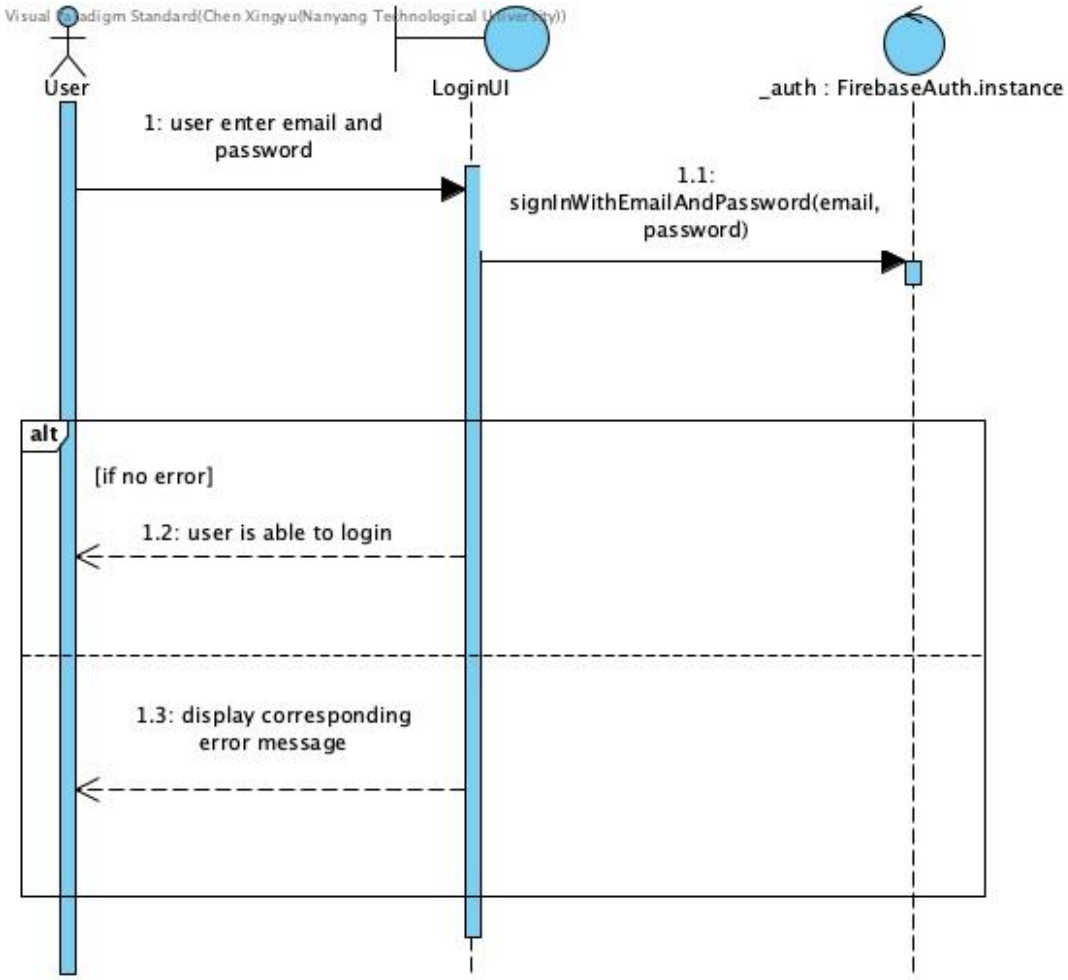
6.1 Use Case Diagram



6.2 Use Cases List

USE CASE ID	USE CASE NAME
1	Verify Login Credentials
2	Account Registration
3	Validation of account availability
4	Forgot Password
5	Edit User(volunteer) Profile Settings
6	Viewing Injury Noticeboard
7	Viewing Individual Cat Map
8	Viewing Cat Map
9	Viewing of All and Nearest Vet Clinics
10	Report Stray Cat Found
11	Report Injured Stray Cat Found
12	Confirmatory Page After Responding
13	Request for Particulars of Passerby
14	Adopt a Stray Cat

6.3 Use Case Descriptions

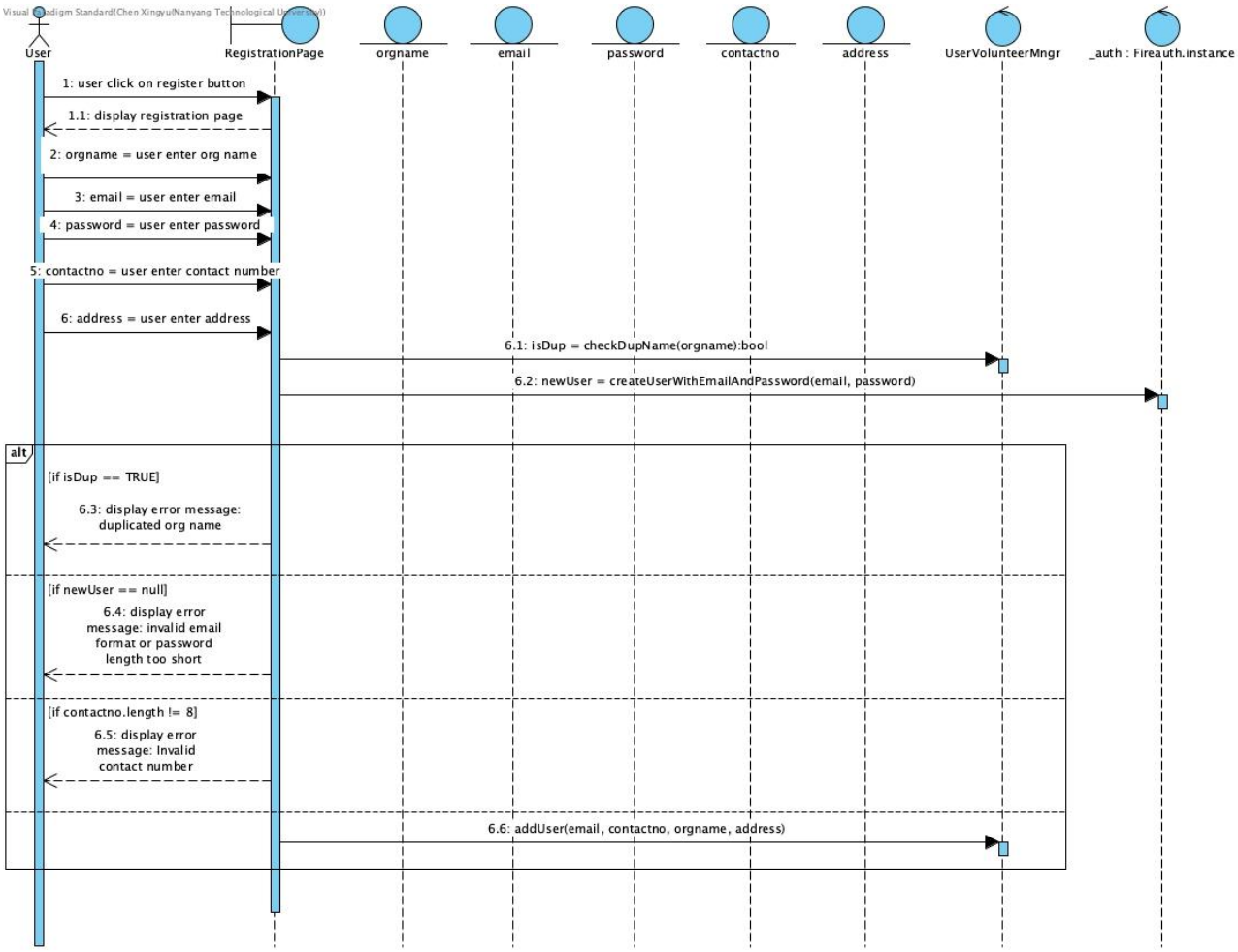


Use Case ID:	1		
Use Case Name:	Verify Login Credentials		
Created By:	Jacintha Wee Yun Yi	Last Updated By:	Jacintha Wee Yun Yi
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	Application, User(volunteer), Database
Description:	Verification of user's(volunteer) account through username and password.

Preconditions:	1. User(volunteer) has an existing and verified account with the Stray Finder app and the account is stored within the database. 2. An active and stable internet connection is required to access the app. 3. The Stray Finder database hosted on the cloud must be online.

Postconditions:	1. User(volunteer) is able to view the noticeboard of injured stray cats. 2. User(volunteer) is able to view the cat map. 3. User(volunteer) is able to view all the vets on the map, or 5 of the nearest vets with respect to the user's location. 4. User(volunteer) is able to edit profile information including organization name, organization contact number and organization address. 5. User(volunteer) is able to log out of the app.
Priority:	High
Frequency of Use:	Every time the user(volunteer) wants to access the app.
Flow of Events:	1. User(volunteer) launches the Stray Finder app downloaded on their mobile phone. 2. The app prompts the user(volunteer) for email and password. 3. The login credentials of the user(volunteer) is verified by the application. 4. If the login credentials are valid, login is successful and the user(volunteer) is able to access functionalities as listed in the postconditions.
Alternative Flows:	AF-S1: The login credentials of the user(volunteer) is invalid 1. App will display an error message that login was unsuccessful. 2. App prompts user(volunteer) for email address and password again, returning to step 2 of flow of events.
Exceptions:	EX1: If user(volunteer) forgets password and wishes to reset password 1. Application proceeds to Use Case 4 Forgot Password.
Includes:	1. Forgot Password (Use Case 4)
Special Requirements:	-
Assumptions:	User is a volunteer.
Notes and Issues:	-

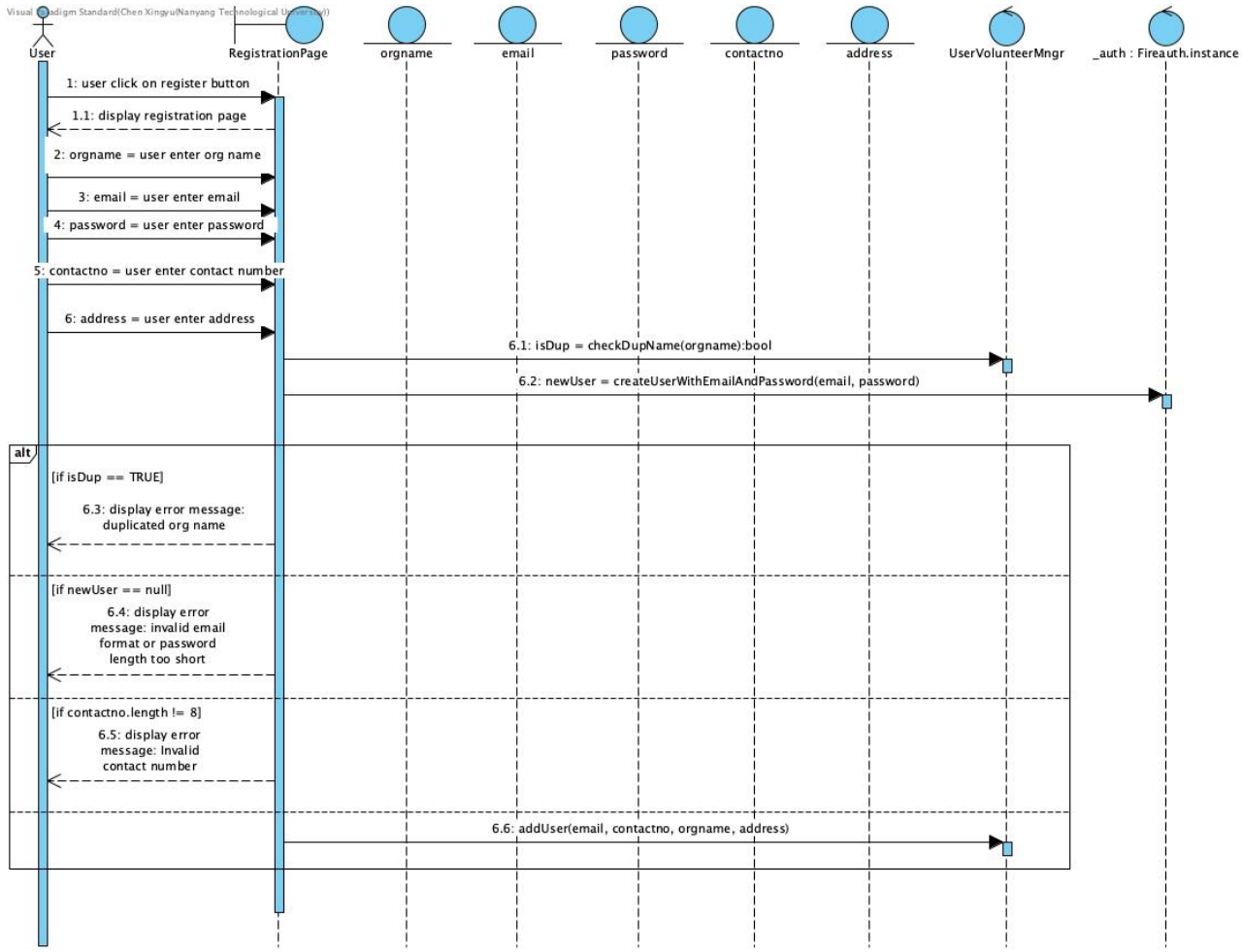


Use Case ID:	2		
Use Case Name:	Account Registration		
Created By:	Hrishikesh Harish Pai	Last Updated By:	Valencia Lie
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	User(volunteer), Database
Description:	Registration of a user(volunteer) with verifiable information e.g. organization name, contact number in order to access the volunteer pages within the mobile application.

Preconditions:	1. User(volunteer) account does not already exist within the database. 2. Active internet connection is required. 3. User(volunteer) must have initiated the process by having pressed the 'Register' button present within the "Use Case 1:Verify Login Credentials" page if the user does not already have an existing account for the Stray Finder application.
Postconditions:	1. User(volunteer) will now possess a verified Stray Finder account which has been stored within the

	application database. 2. The user(volunteer) will no longer be required to create a new account in order for them to use the application, but will only need to login to their existing account.
Priority:	High
Frequency of Use:	Once, for each unique volunteer organization.
Flow of Events:	1. User(volunteer) opens the mobile application after having downloaded it for the first time. The user(volunteer) will be brought to the login page as per "Use Case 1:Verify Login Credentials". User(volunteer) then clicks on the "Register" button to begin the process of creating an account. 2. User(volunteer) will be redirected to the account registration page. The user(volunteer) will be prompted to fill in their particulars, chiefly "Organization Name", "Email, "Password", "Contact Number" and "Address".
Alternative Flows:	-
Exceptions:	EX2: Account already exists 1. App will display an error message notifying the user(volunteer) that an account using the organization's name has already been registered. EX3: Text fields not fully filled 1. Register button will be disabled until all text fields are fully filled EX4: Text fields are not correctly filled: contact number is not of the appropriate length (<8 or >8 digits) 1. Register button will be disabled until contact number is of 8 digits EX5: Text fields are not correctly filled: email is not of an appropriate format and password is not of the appropriate length (<6 characters) 1. App will display an error message notifying the user(volunteer) that the email is not of the correct format
Includes:	1. Validation of Account Availability (Use Case 3)
Special Requirements:	-
Assumptions:	1. Account registration is done by a verified volunteer organization.
Notes and Issues:	-

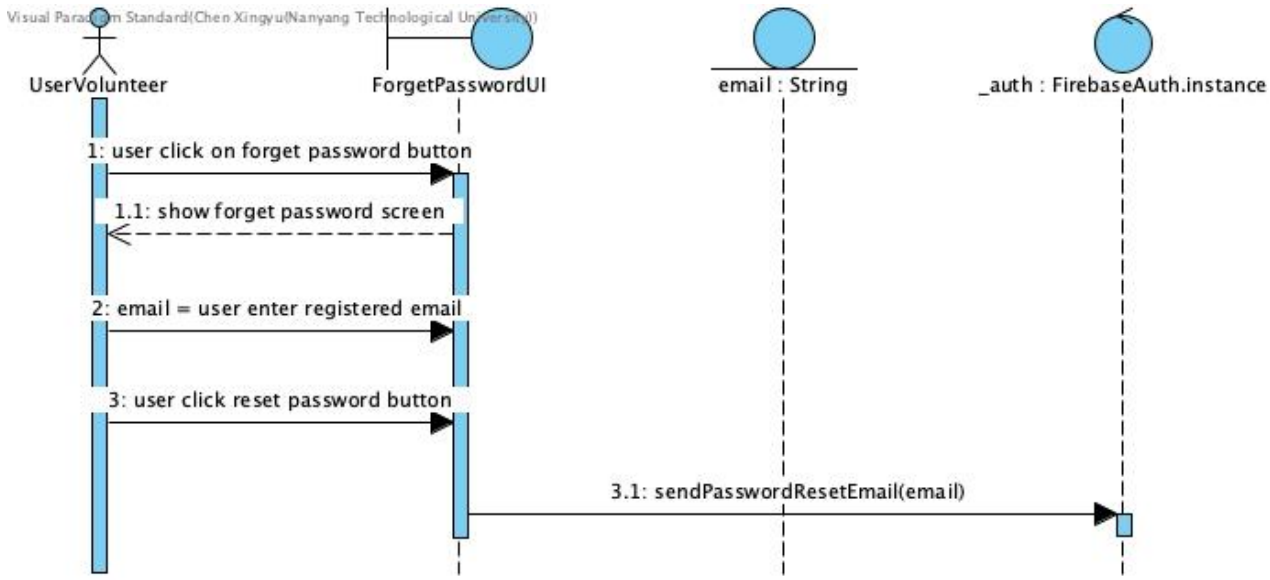


Use Case ID:	3		
Use Case Name:	Validation of account availability		
Created By:	Hu Zhuangyu	Last Updated By:	Jacintha Wee Yun Yi

Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	Application, Database
Description:	Application to validate availability of the account to be registered.
Preconditions:	1. The user (volunteer) account to be registered must not already exist in the database. 2. Mobile must be connected to the Internet (eg. WiFi). 3.Use Case 2 Account Registration check if there is an existing account in database which has already registered the email and/or organisation name to be registered.
Postconditions:	1. The user (volunteer) account is created successfully and is saved in the database. 2. The user (volunteer) is able to login with the registered account.

Priority:	Medium
Frequency of Use:	Low
Flow of Events:	1. Application will check that the user (volunteer) account to be registered does not exist in the database. 2. Application will create the user (volunteer) account in the database.
Alternative Flows:	-
Exceptions:	EX6: Application detects the user (volunteer) account to be registered has already existed in the database: via checking of all registered organization name and email addresses in the database. 1. Application will show the error message if organization name or the email address provided has already been registered 2. User (volunteer) needs to enter another organization name or another email address in order to proceed.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-



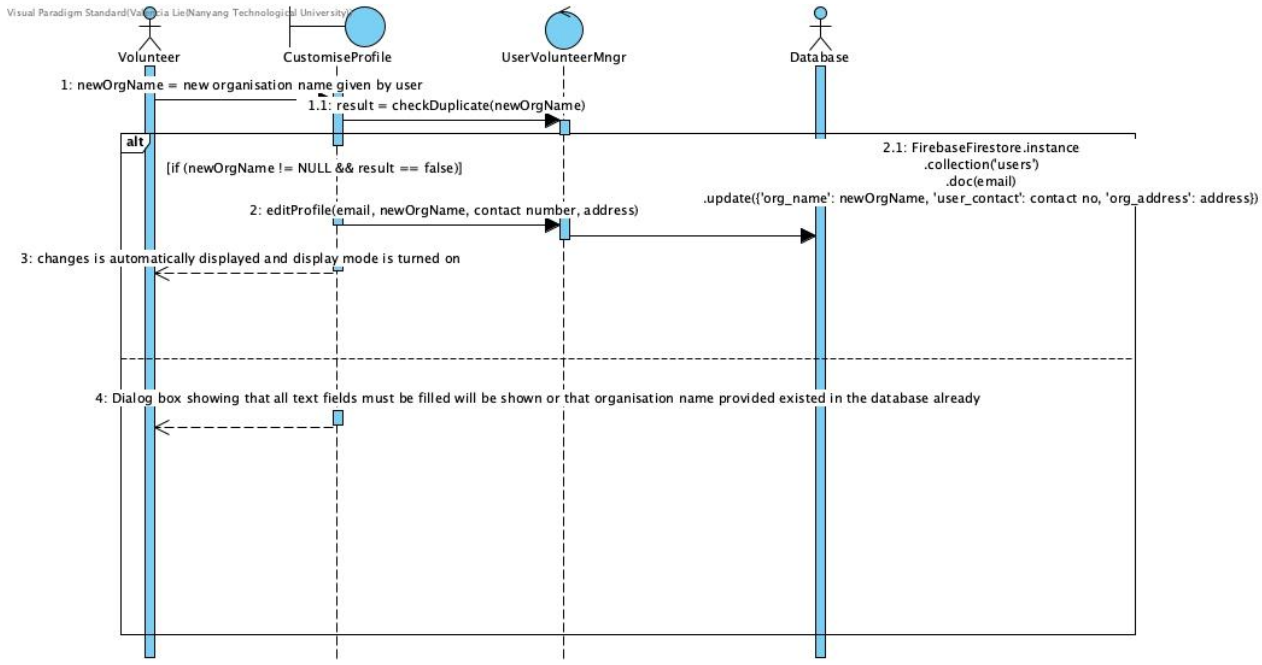
Use Case ID:	4

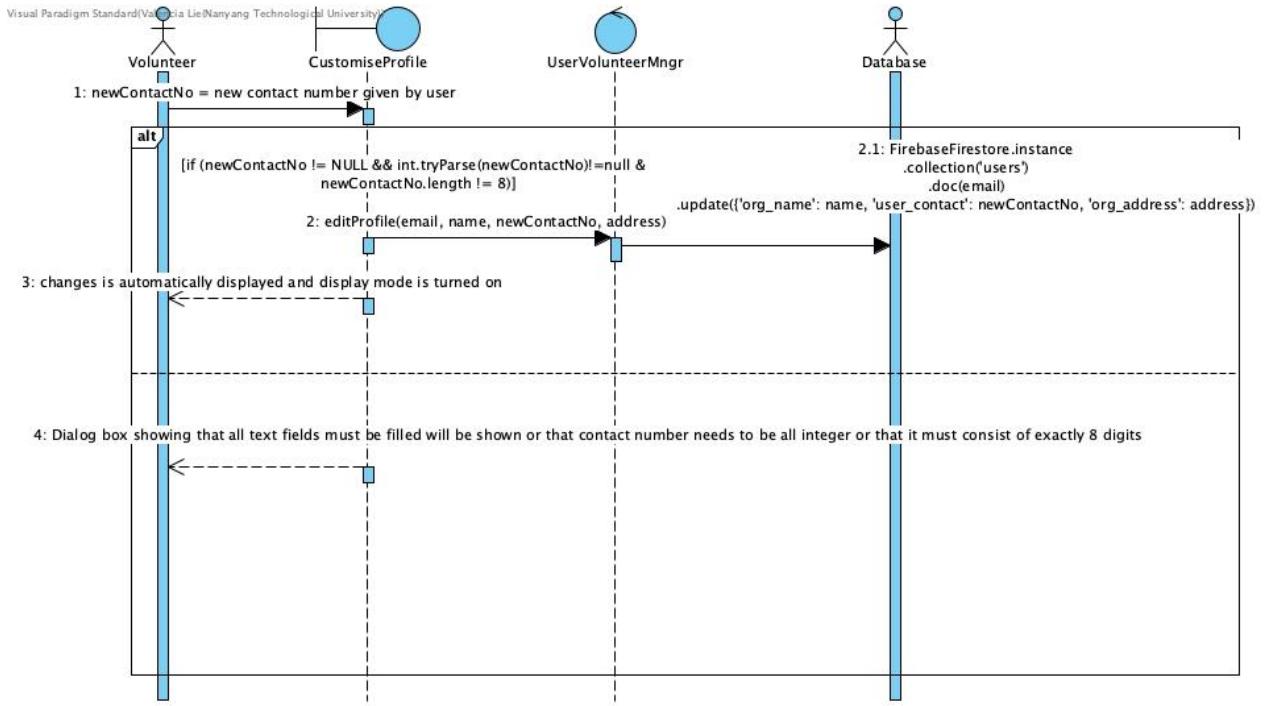
Use Case Name:	Forgot Password		
Created By:	Jacintha Wee Yun Yi	Last Updated By:	Jacintha Wee Yun Yi
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

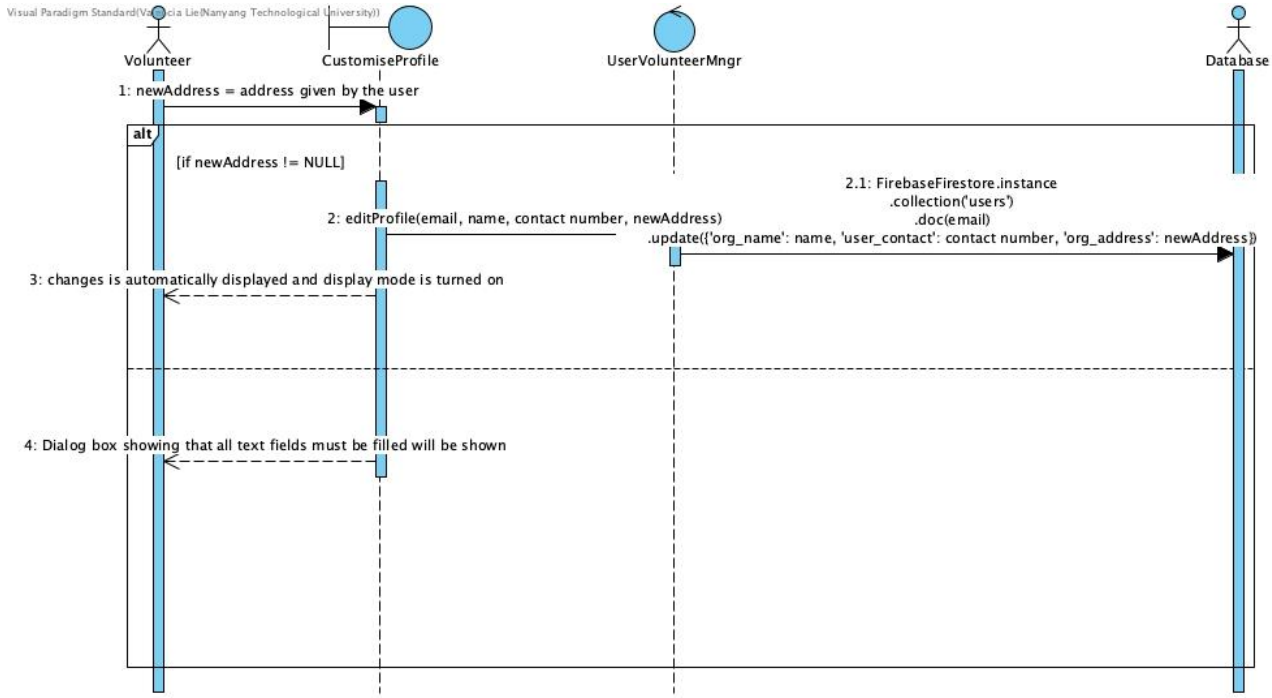
Actor:	Application
Description:	User(volunteer) can choose to reset the password if he/she has forgotten the password.
Preconditions:	1. Execution after Use Case 1 Verify Login Credentials if the exception case of the user(volunteer) having forgotten the password to the account is performed. 2. Mobile must be connected to the Internet (eg. WiFi).
Postconditions:	1. User(volunteer) reset password for account and can login into the account using the new password.
Priority:	High
Frequency of Use:	Low
Flow of Events:	1. User(volunteer) clicks on the forgot password button. 2. App will prompt the user(volunteer) to enter the organization email. 3. After retrieving the email of the verified organization, an email will be sent to the email address specified by the user(volunteer) containing the link to reset password. 4. After following the instructions from the link, the password is successfully reset. 5. Users(volunteers) can now login using email and the new password into the app.

Alternative Flows:	AF-S2: Email entered is unverifiable 1. App is unable to verify the organization email that was submitted by the user(volunteer) to reset the password. 2. App will display an error message indicating that the organization does not have a registered account yet.
Exceptions:	-
Includes:	-

Special Requirements:	-
Assumptions:	-
Notes and Issues:	-



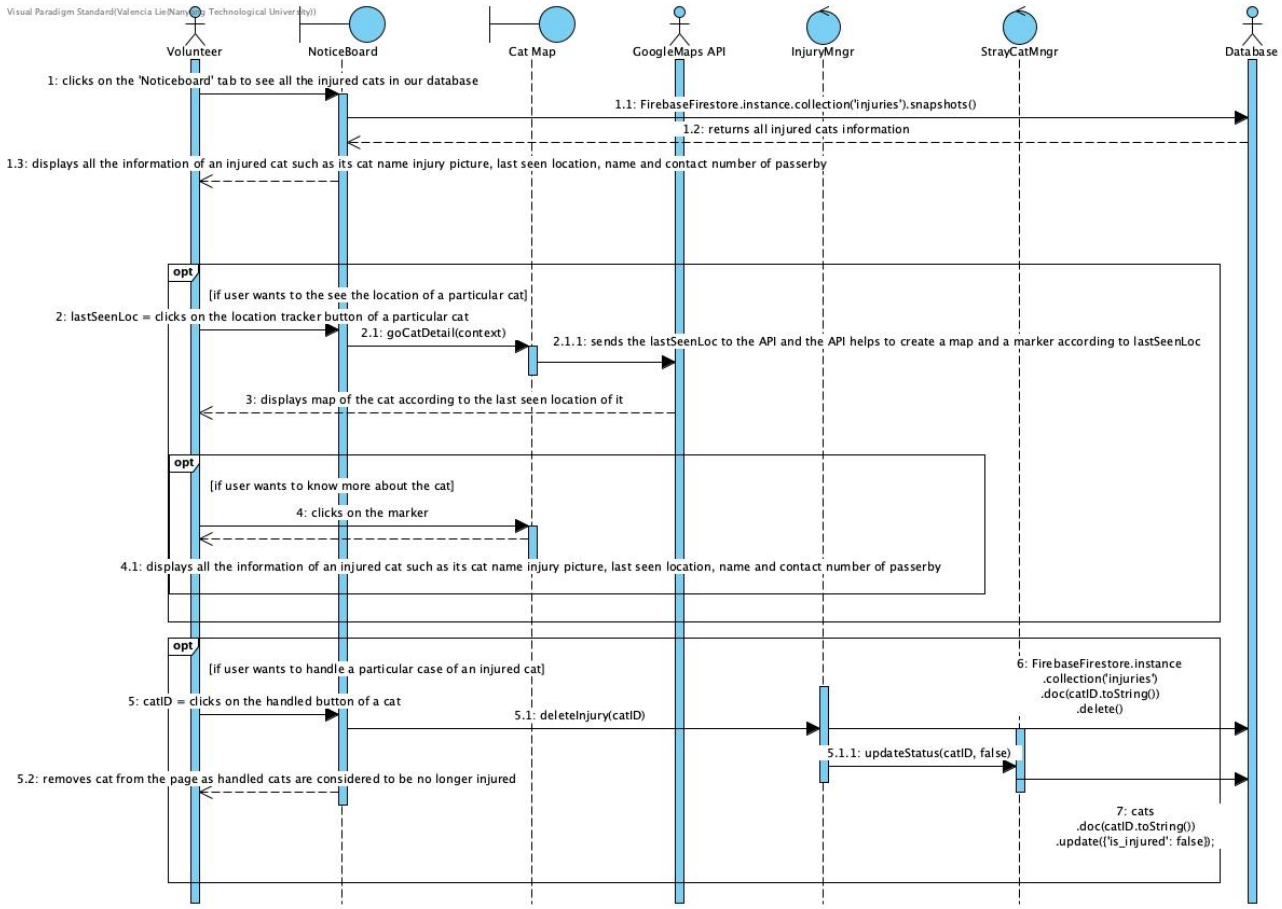




Use Case ID:	5
Use Case Name:	Edit User(volunteer) Profile Settings

Created By:	Jacintha Wee Yun Yi	Last Updated By:	Jacintha Wee Yun Yi
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	User(volunteer), Database
Description:	Users(volunteer) will be allowed to edit the organization name, address and contact number for the volunteer organization's account.
Preconditions:	1. User(volunteer) account exists and is registered in the database (Use Case 2 Account Registration). 2. An active and stable internet connection is required to access the app.
	3. User(volunteer) has successfully logged into the app.
Postconditions:	1. User(volunteer) account profile is updated and the new changes will be seen on subsequent logins.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. User(volunteer) clicks on the profile icon at the top right hand corner of the screen. 2. User(volunteer) clicks on the pen icon at the top right hand corner of the screen to begin editing. 3. Users(volunteer) can choose to edit the "Organization Name", "Contact Number" and "Address" text fields which have been automatically populated with the last updated data from the database. 4. User(volunteer) clicks on the brown "Confirm" button to save changes once he/she has edited all text fields.
Alternative Flows:	AF-S3: User(volunteer) decides not to edit profile 1. User(volunteer) clicks on the back arrow button at the top left hand corner of the screen and any unsaved edits will be discarded.
Exceptions:	EX7: Not all text fields are filled up 1. An error message prompts the user(volunteer) to fill in all text fields before proceeding to confirm updates. EX8: Contact number input is not 8 digits 1. An error message prompts the user(volunteer) to only enter 8 digits for the contact number to be updated. EX9: None of the text fields were updated 1. The app displays a message that no information was updated.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

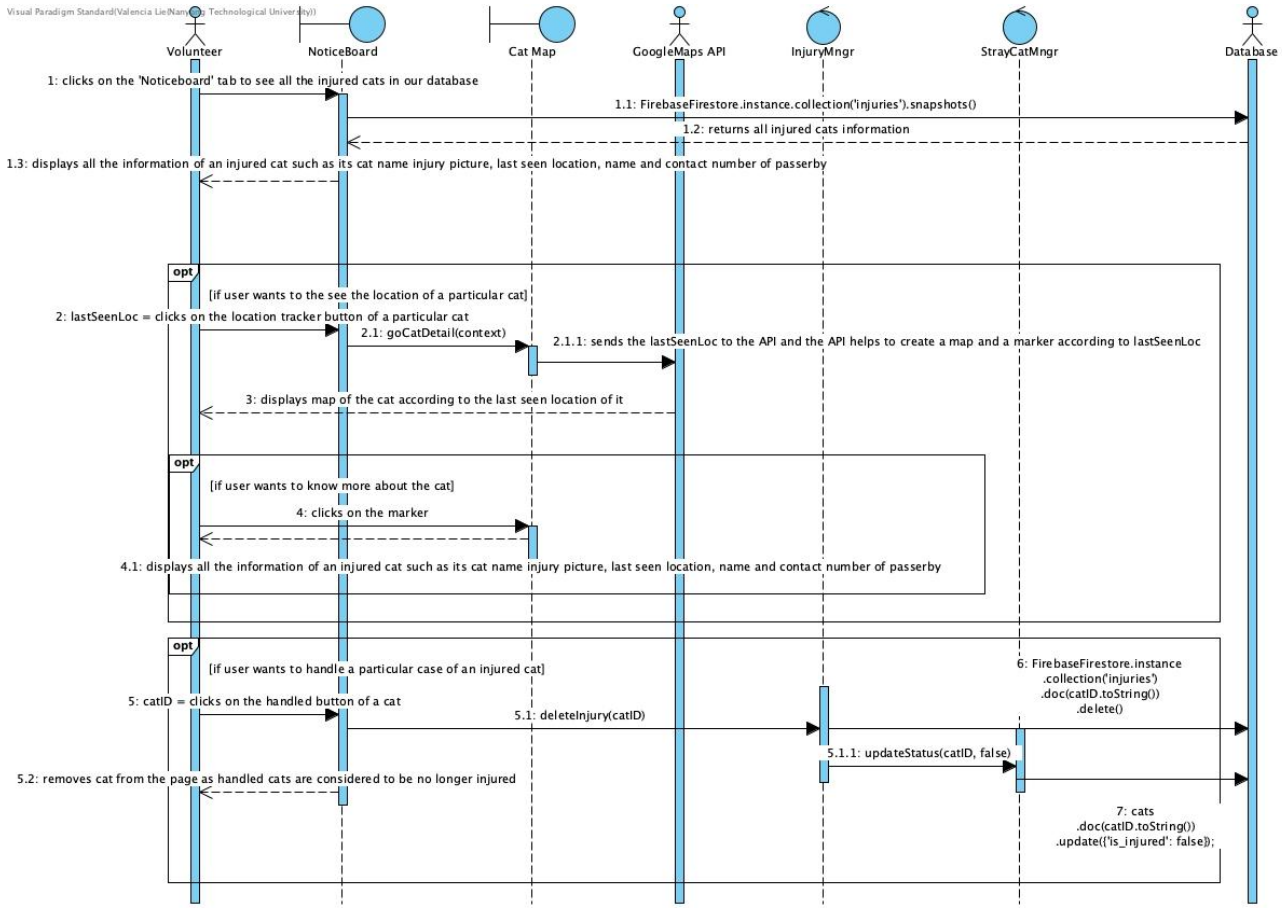


Use Case ID:	6		
Use Case Name:	Viewing Injury Noticeboard		
Created By:	Jacintha Wee Yun Yi	Last Updated By:	Jacintha Wee Yun Yi

Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	User(volunteer), Database, Google Maps API
Description:	Users(volunteers) can view the noticeboard feature which lists the injured stray cats which have been reported by other users(passerby).
Preconditions:	1. Feature is made available after the user(volunteer) has logged in successfully (Use Case 1 Verify Login Credentials). 2. There are reports made by users(passerby) on injured stray cats which were found, so that users(volunteer) can view and address the case. 3. An active and stable internet connection is available to all users so that reports made by users(passerby) can be updated and seen by users(volunteer).

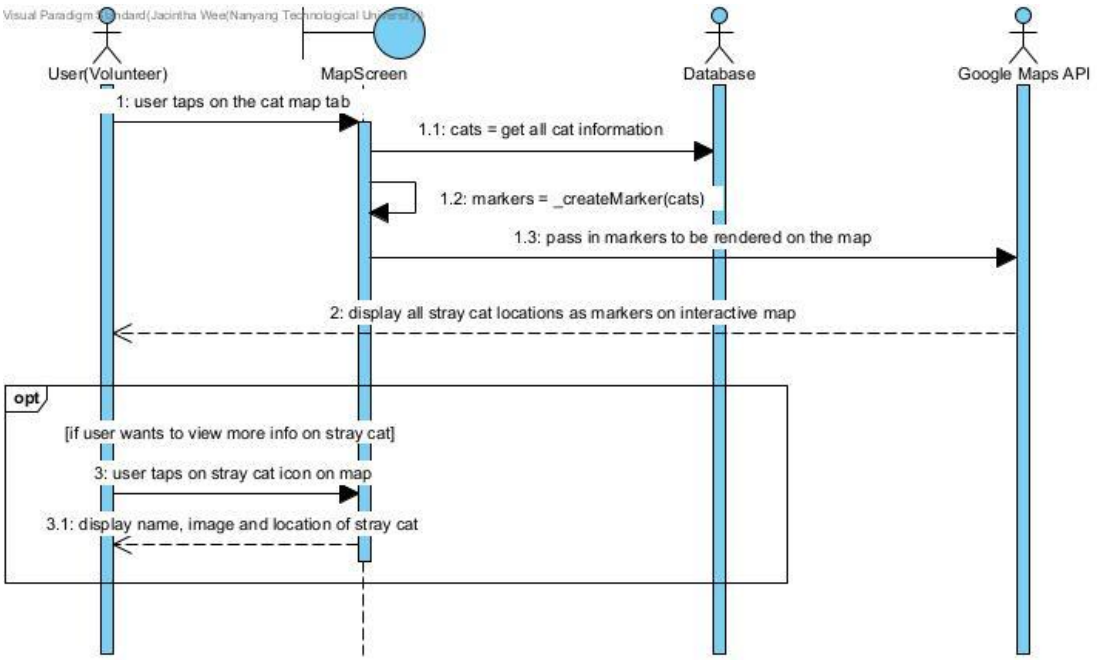
Postconditions:	1. If the user(volunteer) chooses to handle an injured stray cat, the information of the injured stray cat being handled by the user(volunteer) is removed from the list.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User(volunteer) clicks on the "Injury Noticeboard" button at the bottom of the screen. 2. The app displays a list of all injured stray cats which have been reported by other users(passerby) and these injured stray cats have not been attended to yet. 3. Information on the injured stray cats are provided such as "Location Tracker", "Contact of passerby (if any)", "Condition", along with the cat's name and the photo of the injury. 4. User(volunteer) can contact the user(passerby) who reported the injured stray cat, only if the user(passerby) chooses to send the injured stray cat to the nearest vet clinic themselves. 5. User(volunteer) can check on the location of the injured stray cat on the map by clicking "Location Tracker". (Use Case 7 Viewing Individual Cat Map). 6. User(volunteer) can click on the "handle" button to confirm that he/she will be handling the case of the corresponding injured stray cat.
Alternative Flows:	AF-S4: User(volunteer) wants to track the location of the injured stray cat 1. User(volunteer) clicks on the location hyperlink of the corresponding injured stray cat he/she wants to check on. 2. App retrieves the location of the injured stray cat and its last seen location is displayed on the interactive map as a marker. 3. When the marker is pressed, it shows the name of the cat, the passerby name and contact number (if applicable) and a button to navigate to the location of the cat via Google Maps. AF-S5: User(volunteer) wants to handle the case of the injured stray cat: 1. User(volunteer) clicks on the "handle" button. 2. The condition of the injured stray cat is changed to uninjured and its case is removed from the Injury Noticeboard.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-



Use Case ID:	7		
Use Case Name:	Viewing Individual Cat Map		
Created By:	Valencia Lie	Last Updated By:	Valencia Lie
Date Created:	16/04/2022	Date Last Updated:	16/04/2022

Actor:	User(volunteer), Google Maps API, Database
Description	Users(volunteer) will be shown a map of Singapore on which the last seen location of the injured stray cat they specified
Preconditions:	1. Feature will be made available as a button on the bottom navigation bar upon successful login in "Use Case 1: Verify Login Details" 2. An active and stable internet connection is required. 3. Invocation of "Use Case 6: Viewing Injury Noticeboard" if the user(volunteer) decides to click the "Location Tracker" button.
Postconditions:	1. User(volunteer) is able to identify the last seen location of the injured stray cat specified on a map
Priority:	High
Frequency of Use:	High

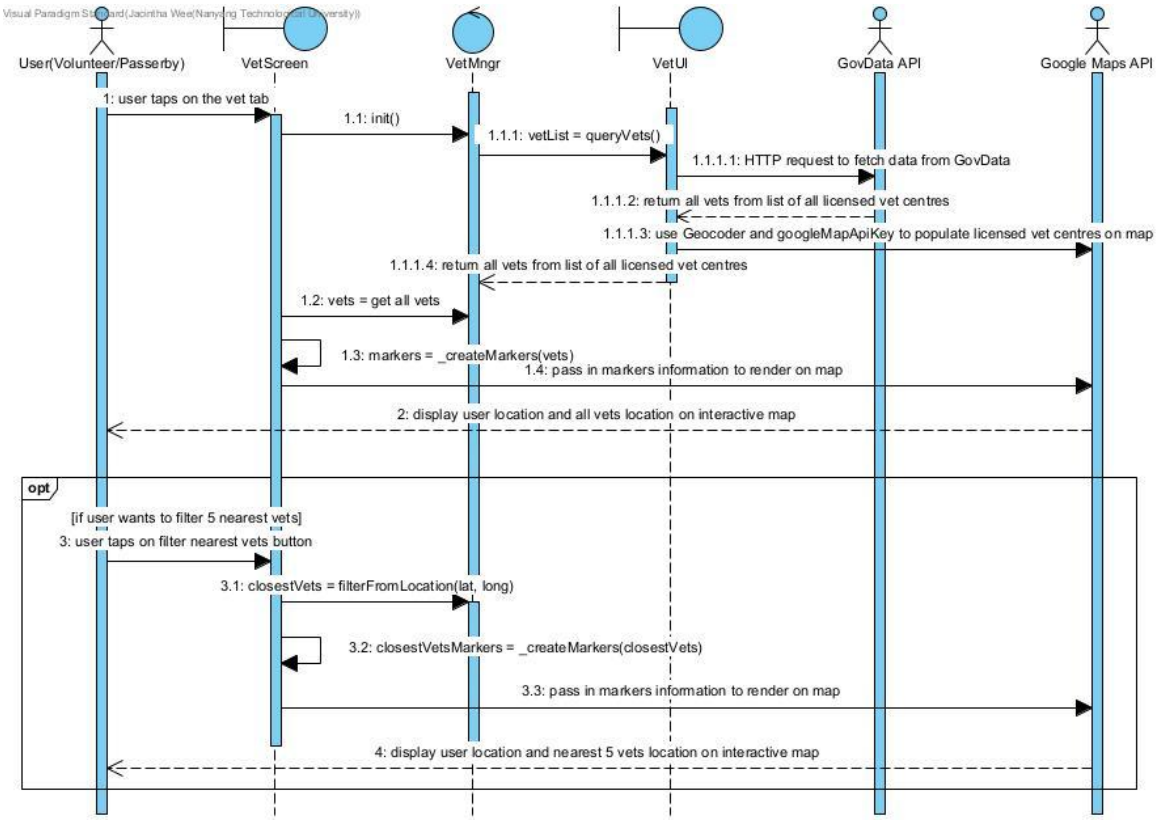
Flow of Events:	1. User(volunteer) clicks on the "Location Tracker" button on the Noticeboard page.. 2. The application will automatically display the map containing the marker of the specified injured stray cat in accordance to its last reported location.
Alternative Flows:	AF-S6: User(volunteer) wants to view more information on the injured stray cat 1. When the marker of the cat is pressed, the name of the cat, the passerby's name and contact number (if applicable) and a button to navigate to the location of the cat via Google Maps will be shown.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-



Use Case ID:	8		
Use Case Name:	Viewing Cat Map		
Created By:	Hrishikesh Harish Pai	Last Updated By:	Valencia Lie
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	User(volunteer), Google Maps API, Database
Description	Users(volunteer) will be shown a map of Singapore on which the last seen locations of all stray cats existing within the database will be displayed on the map using markers.
Preconditions:	1. Feature will be made available as a button on the bottom navigation bar upon successful login in "Use Case 1: Verify Login Credentials" 2. An active and stable internet connection is required.
Postconditions:	1. User(volunteer) is able to identify the last seen locations of all stray cats in the database, and is able to clearly identify the injured cats through the different markers indicating the different injury status
Priority:	High
Frequency of Use:	High

Flow of Events:	1. User(volunteer) clicks on the "Cat Map" button on the bottom navigation bar. 2. The application will automatically display the map containing the markers of all last seen cats stored within the database. 3. The stray cats will be displayed on the cat map, represented by markers that denote their injury status: purple for injured and orange for uninjured cats.
Alternative Flows:	AF-S7: User(volunteer) wants to view more information on the stray cat 1. Upon pressing on the marker, information regarding the cat will be displayed in an information window, namely the name of the cat, the photo of the cat and a button to navigate to the location of the cat via Google Maps
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

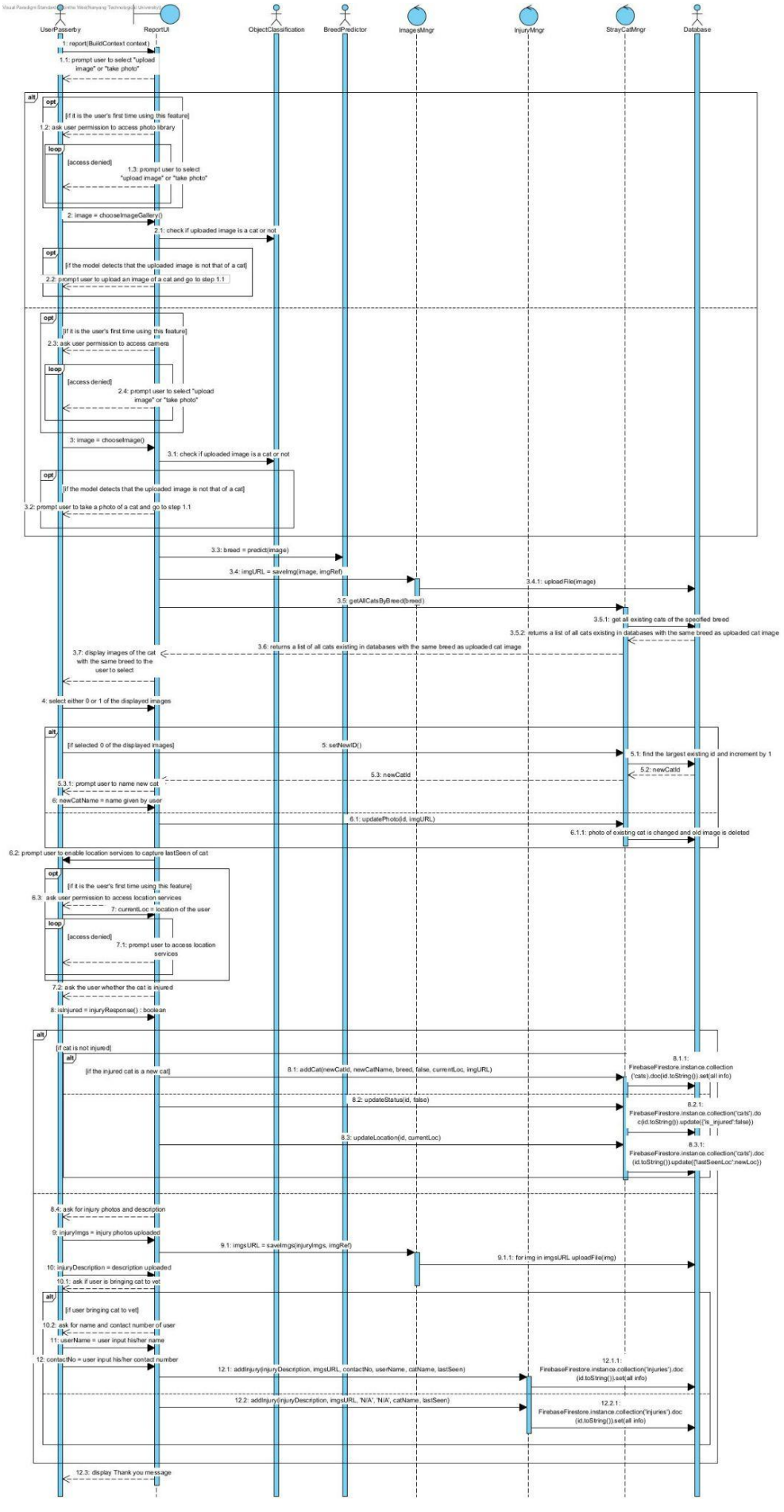


Use Case ID:	9		
Use Case Name:	Viewing of All and Nearest Vet Clinics		
Created By:	Hu Zhuangyu	Last Updated By:	Valencia Lie
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	User (volunteer, passerby), Google Map API, Gov Data API
Description:	Users(volunteer,passerby) can view all vet clinics and the 5 nearest vet clinics according to the user's location.
Preconditions:	1. Feature is made available after the user(volunteer) has logged in successfully (Use Case 1 Verify Login Credentials). 2. Feature is also available for users(passerby) to access from the bottom navigation bar. 2. Mobile must be connected to the Internet. 3. Mobile must grant the application access to the current location.

Postconditions:	-
Priority:	Medium
Frequency of Use:	Medium

Flow of Events:	1. User (volunteer) clicks on the “vets” button at the bottom of the screen. 2. A map displaying drop pins of all vets within the database from the list of licensed vet centers from the GovData API are shown.
Alternative Flows:	AF-S8: 1. Upon clicking the floating action button at the bottom right corner of the screen, two buttons titled “Filter Nearest 5 Vets’ and “Show All Vets” will be shown.The user can then display the top 5 nearest vet clinics as markers on the map, with respect to the user's (passerby, volunteer) current location, upon clicking the first button. 2. When one of these markers is clicked, an information box containing details of the vet will be displayed above the marker, such as the name, the type, the address and the telephone number of the vet. 3. By clicking “Show All Vets”, the user is, again, shown all the vets present within the Govdata API.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-


UserPasserby
ReportUI
ObjectClassification
BreedPredictor
ImagesMngr
InjuryMngr
StrayCatMngr
Database
1: report(BuildContext context)
1.1: prompt user to select "upload image" or "take photo"
alt
opt
[if it is the user's first time using this feature]
1.2: ask user permission to access photo library
loop
[access denied]
1.3: prompt user to select "upload image" or "take photo"
2: image = chooseImageGallery()
2.1: check if uploaded image is a cat or not
opt
[if the model detects that the uploaded image is not that of a cat]
2.2: prompt user to upload an image of a cat and go to step 1.1
opt
[if it is the user's first time using this feature]
2.3: ask user permission to access camera
loop
[access denied]
2.4: prompt user to select "upload image" or "take photo"
3: image = chooseImage()
3.1: check if uploaded image is a cat or not
opt
[if the model detects that the uploaded image is not that of a cat]
3.2: prompt user to take a photo of a cat and go to step 1.1
3.3: breed = predict(image)
3.4: imgURL = saveImg(image, imgRef)
3.4.1: uploadFile(image)
3.5: getAllCatsByBreed(breed)
3.5.1: get all existing cats of the specified breed
3.5.2: returns a list of all cats existing in databases with the same breed as uploaded cat image
3.6: returns a list of all cats existing in databases with the same breed as uploaded cat image
3.7: display images of the cat with the same breed to the user to select
4: select either 0 or 1 of the displayed images
alt
[if selected 0 of the displayed images]
5: setNewID()
5.1: find the largest existing id and increment by 1
5.2: newCatId
5.3: newCatId
5.3.1: prompt user to name new cat
6: newCatName = name given by user
6.1: updatePhoto(id, imgURL)
6.1.1: photo of existing cat is changed and old image is deleted
6.2: prompt user to enable location services to capture lastSeen of cat
opt
[if it is the user's first time using this feature]
6.3: ask user permission to access location services
7: currentLoc = location of the user
loop
[access denied]
7.1: prompt user to access location services
7.2: ask the user whether the cat is injured
8: isInjured = injuryResponse() : boolean
alt
[if cat is not injured]
alt
[if the injured cat is a new cat]
8.1: addCat(newCatId, newCatName, breed, false, currentLoc, imgURL)
8.1.1: FirebaseFirestore.instance.collection ('cats').doc(id.toString()).set(all info)
8.2: updateStatus(id, false)
8.2.1: FirebaseFirestore.instance.collection('cats').doc(id.toString()).update({'is_injured':false})
8.3: updateLocation(id, currentLoc)
8.3.1: FirebaseFirestore.instance.collection('cats').doc(id.toString()).update({'lastSeenLoc':newLoc})
8.4: ask for injury photos and description
9: injuryImgs = injury photos uploaded
9.1: imgsURL = saveImgs(injuryImgs, imgRef)
9.1.1: for img in imgsURL uploadFile(img)
10: injuryDescription = description uploaded
10.1: ask if user is bringing cat to vet
alt
[if user bringing cat to vet]
10.2: ask for name and contact number of user
11: userName = user input his/her name
12: contactNo = user input his/her contact number
12.1: addInjury(injuryDescription, imgsURL, contactNo, userName, catName, lastSeen)
12.1.1: FirebaseFirestore.instance.collection('injuries').doc(id.toString()).set(all info)
12.2: addInjury(injuryDescription, imgsURL, 'N/A', 'N/A', catName, lastSeen)
12.2.1: FirebaseFirestore.instance.collection('injuries').doc(id.toString()).set(all info)
12.3: display Thank you message

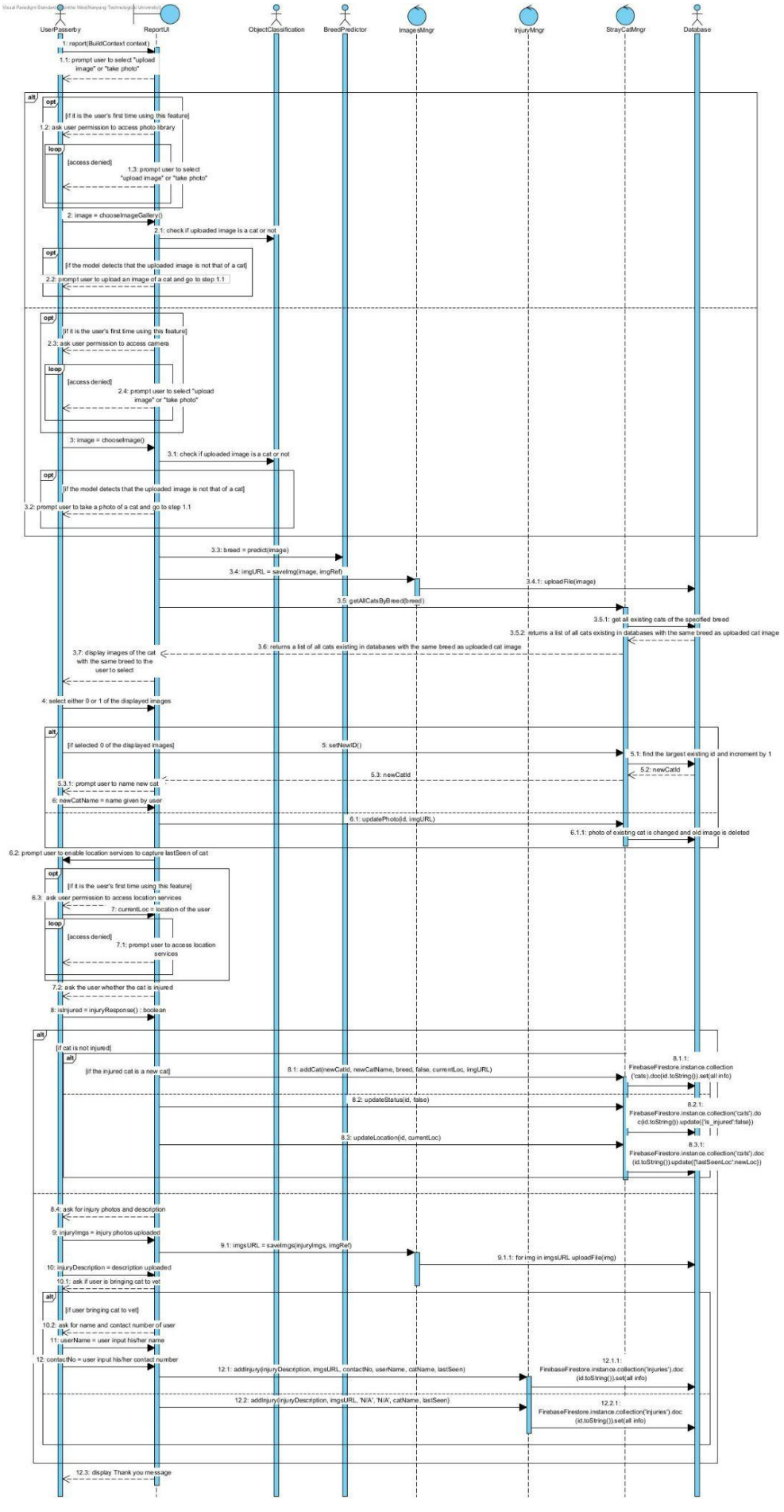
Use Case ID:	10		
Use Case Name:	Report Stray Cat Found		
Created By:	Jacintha Wee Yun Yi	Last Updated By:	Jacintha Wee Yun Yi
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	User(passerby), Database
Description:	When a user(passerby) comes across a stray cat, he/she may choose to upload a photo of it and check if its identity has been established in the database.
Preconditions:	1. Feature is made available after the user(passerby) clicks on the "I am a passerby" button on the main landing page of the app. 2. User(passerby) grants the app access to the mobile phone's camera and photo library. 3. An active and stable internet connection is required to access the app.
Postconditions:	1. A new entry for the reported stray cat will be created in the database if the reported stray cat does not exist in the database prior to reporting.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User(passerby) clicks on the "Report" button on the bottom left hand corner of the screen. 2. User(passerby) is shown two buttons, "Take a photo" and "Upload a photo" which represents the two modes that the user(passerby) can upload their photo of the stray cat in the app. Clicking on the former button opens the user's(passerby) camera while clicking on the latter button opens the user's(passerby) photo library. 3. If it is the user's(passerby) first time using this feature, he/she will be prompted by the app to allow access to his/her phone camera or photo library. Subsequently, as his/her preferences are saved by the app, he/she will not be prompted again. 4.The photo is first processed by an object classification model in order to verify that the uploaded image is indeed that of a cat. 4. The photo of the stray cat is then analysed, and a collection of photos of stray cats existing within the database is displayed to the user(passerby). These stray cats have the same breed to the cat which was reported and are matched by a pre-trained machine learning algorithm within the app. 5. User(passerby) can select one of the cat photos displayed if he/she thinks that is the cat he/she is currently reporting on. Once selected, a tick icon will appear on the selected photo to indicate selection. Clicking on the "continue" button on the bottom right of the screen will lead to step 7. 6. If the user(passerby) is unable to identify any of the cats displayed as the cat he/she is reporting on, he/she does not need to select any of the cat photos and proceed to click on "continue". He/she will be led to a page to name the cat he/she found. After naming the cat and clicking "confirm", the user(passerby) is led to step 7. 7. The user(passerby) is prompted to confirm if the stray cat is injured.

Alternative Flows:	AF-S9: The reported stray cat is injured 1. Proceed to Use Case 11 Report Injured Stray Cat Found. AF-S10: The reported stray cat is not injured 1. A thank you message is displayed (Use Case 12: Confirmatory Page After Responding)

Exceptions:	EX10: If picture uploaded is not a cat picture: 1. An object classification model will detect if the photo uploaded is a cat or not. If it is not a cat, an error message will be displayed
Includes:	1. Report Injured Stray Cat Found (Use Case 11) 2. Confirmatory Page After Responding (Use Case 12)
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

For diagrams in higher resolution: shorturl.at/jkoBC

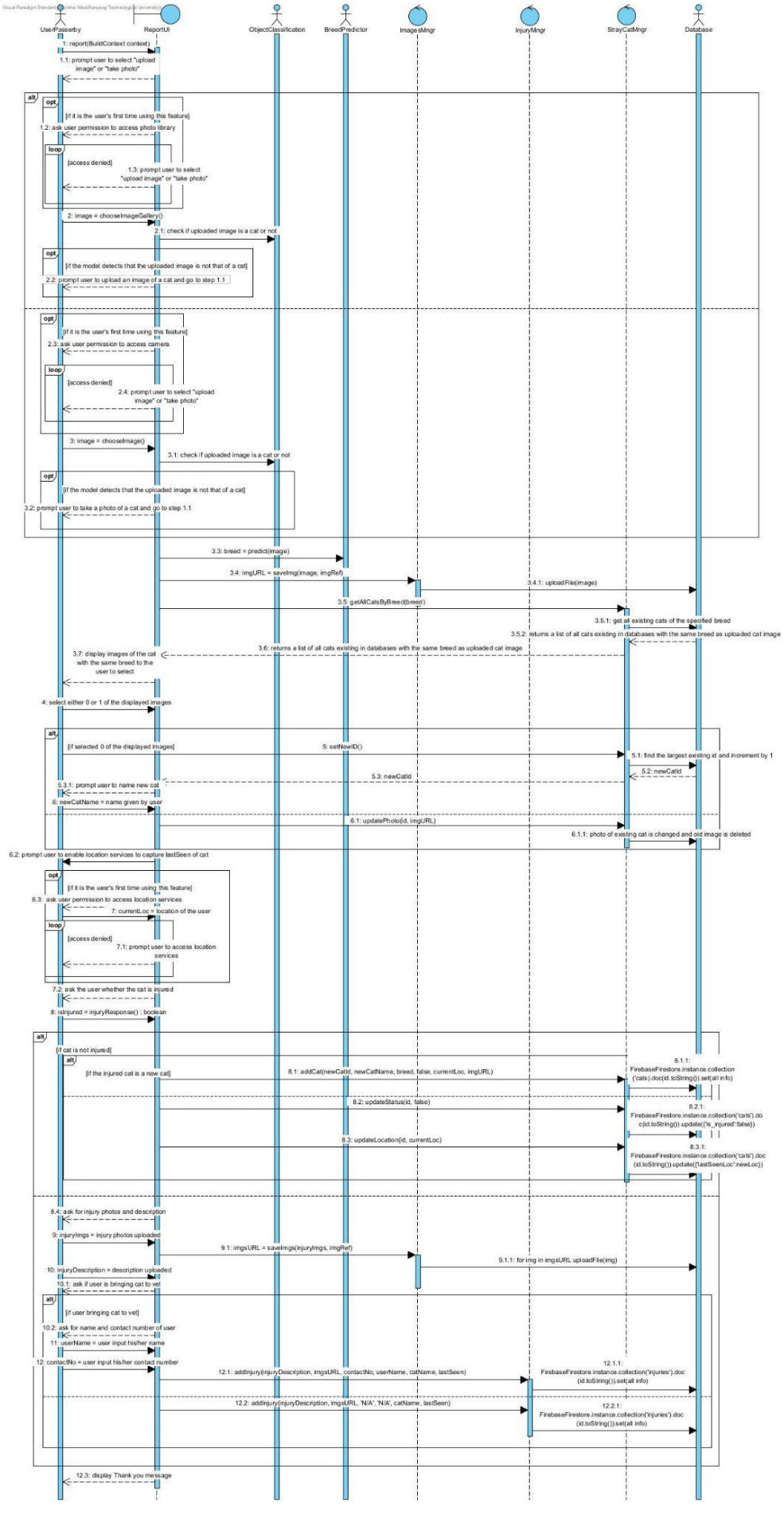


Use Case ID:	11		
Use Case Name:	Report Injured Stray Cat Found		
Created By:	Hrishikesh Harish Pai	Last Updated By:	Valencia Lie
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	User(Passerby), Database
Description:	When the stray cat that a user(passerby) has come across is injured, this information will be updated in the database and the user(passerby) will be prompted on their decision to take the cat to a nearby vet themselves.
Preconditions:	1. Feature is invoked as an included case from Use Case 10: Report Stray Cat Found 2. Mobile must grant the application access to the current location. 3.An active and stable internet connection is required to access the app.
Postconditions:	1. Information pertaining to the cat's injuries will be updated within the application database and will assign the condition of the cat as "Injured"
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The user(passerby) must allow the app to gain access to their current location before proceeding. 2. The user(passerby) is prompted to confirm if the cat is injured, and two options - a green circle with a tick indicating yes, and a red circle with an 'X' indicating no - will be displayed. 3. The user(passerby) clicks the green circle, redirecting them to a page prompting them to take a picture of the injury as well as a text box (optional fill-in) requesting for further information to describe the injury.
Alternative Flows:	AF-S11: User(passerby) does not allow for app to gain access to their current location 1. User(passerby) will be able to provide an address as their location manually AF- S12: User(passerby) DOES NOT take the cat to a nearby vet themselves. 1. User(passerby) is directed to Use Case 12 (Confirmatory Page After Responding). AF- S13: User(passerby) DOES take the cat to a nearby vet themselves. 1. User(passerby) is directed to Use Case 13 (Request for Particulars of Passerby).
Exceptions:	EX11: User(passerby) does not include a picture and/or description of injury 1. Users(passerby) will be prompted to take a picture of the injury before being allowed to continue.

Includes:	1. Report Stray Cat Found (Use Case 10)
Special Requirements:	-
Assumptions:	The stray cat found is injured.
Notes and Issues:	-

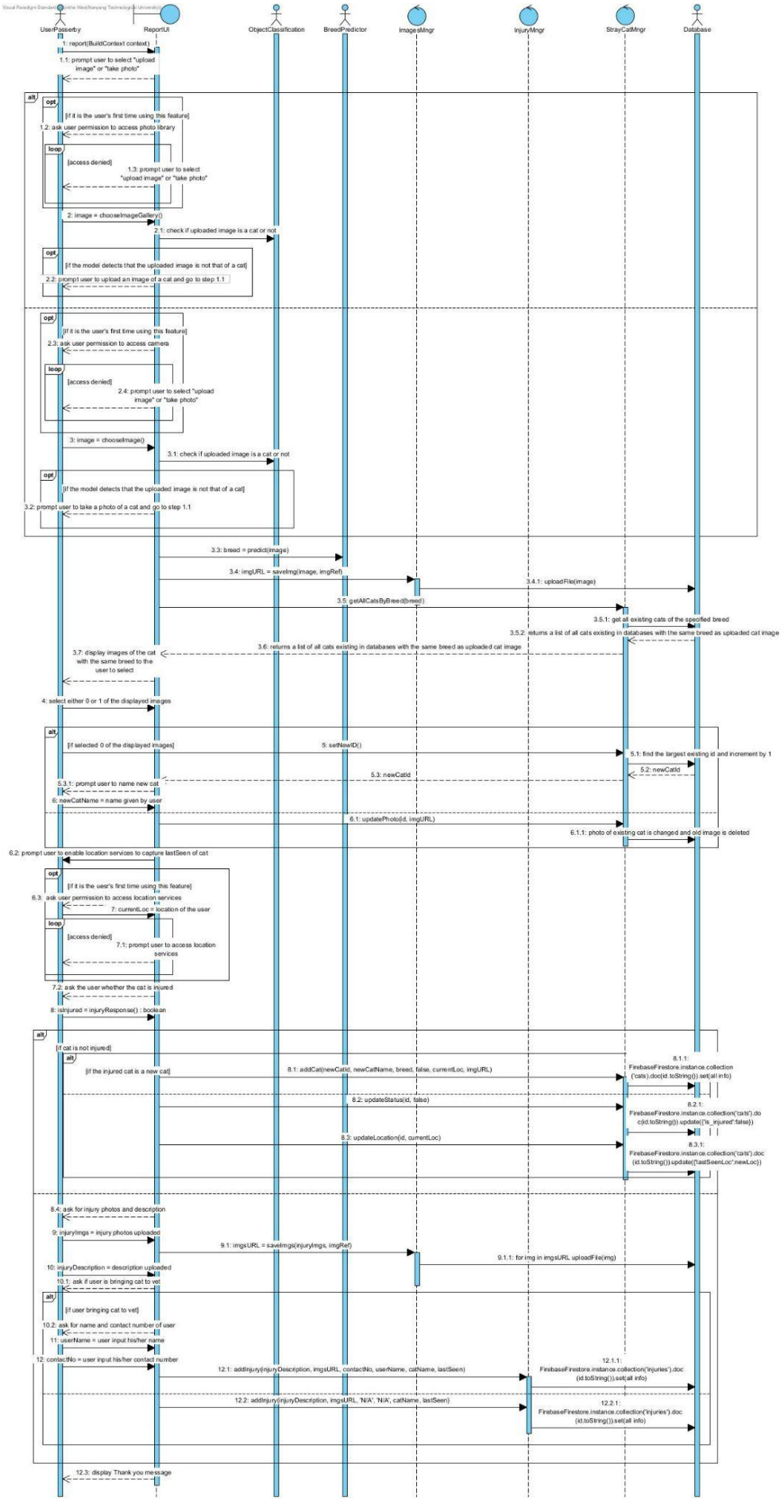
For diagrams in higher resolution: shorturl.at/jkoBC



Use Case ID:	12		
Use Case Name:	Confirmatory Page After Responding		
Created By:	Hrishikesh Harish Pai	Last Updated By:	Jacintha Wee Yun Yi
Date Created:	05/02/2022	Date Last Updated:	05/02/2022

Actor:	User(Passerby)
Description:	User is displayed a confirmatory message thanking them for locating a stray cat
Preconditions:	1. Feature is invoked as an alternate case from Use Case 10: Report Stray Cat Found and Use Case 11: Report Injured Stray Cat Found
Postconditions:	-
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User has entered the relevant details about the stray cat and has chosen not to take it to a nearby vet themselves. 2. A message thanking the user for having found that particular stray cat will be displayed on this page. OR 1. User(passerby) found a stray cat which may or may not exist in the database and the stray cat is not injured. 2. A message thanking the user for having found that particular stray cat will be displayed on this page.
Alternative Flows:	AF-S14: If the user(passerby) presses on the ‘report more cats’ button 1. The user(passerby) is directed to the start of the report page
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

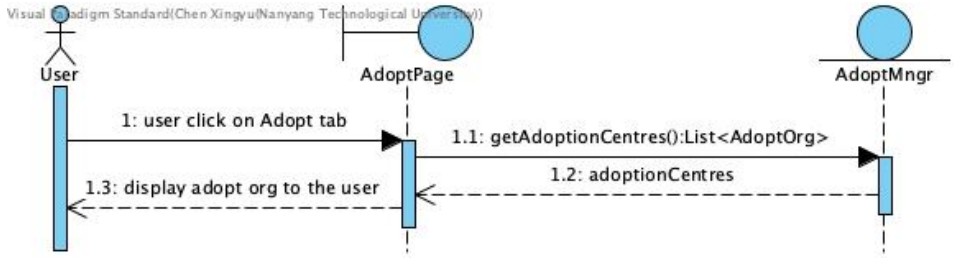
For diagrams in higher resolution: shorturl.at/jkoBC



Use Case ID:	13		
Use Case Name:	Request for Particulars of Passerby		
Created By:	Hrishikesh Harish Pai	Last Updated By:	Jacintha Wee Yun Yi
Date Created:	05/02/2022	Date Last Updated:	05/02/2022

Actor:	User(Passerby), Database
Description:	Users(Passerby) are prompted to enter their particulars in the event that they take an injured stray cat to the vet so as to allow a user(volunteer) to follow up on the stray cat.
Preconditions:	1. This feature is invoked as an alternative case from Use Case 11: Report Injured Stray Cat Found.
Postconditions:	1. Information submitted by the user(passerby) will be uploaded onto the database for the user(volunteer) to retrieve and follow up upon.
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. User(passerby) is prompted to input his/her name and contact number so that the user(volunteer) taking over the case of the injured stray cat can contact the user(passerby) for more information on the injured stray cat. 2. After inputting name and contact number, the user(passerby) clicks on the “Confirm” button to save the information to the database. 3. A thank you message is displayed to the user(passerby) (Use Case 12: Confirmatory Page After Responding).
Alternative Flows:	AF-S15: The user(passerby) decides not to take the injured cat to the vet 1. User(passerby) clicks on the back arrow and any information input into the data fields for “Name” and “Contact number” will not be saved.
Exceptions:	EX12: User(passerby) does not enter a valid contact number 1. The “Confirm” button will be disabled until the user(passerby) inputs a valid contact number and has to submit his/her information again.
Includes:	-
Special Requirements:	-
Assumptions:	-

Notes and Issues:	-

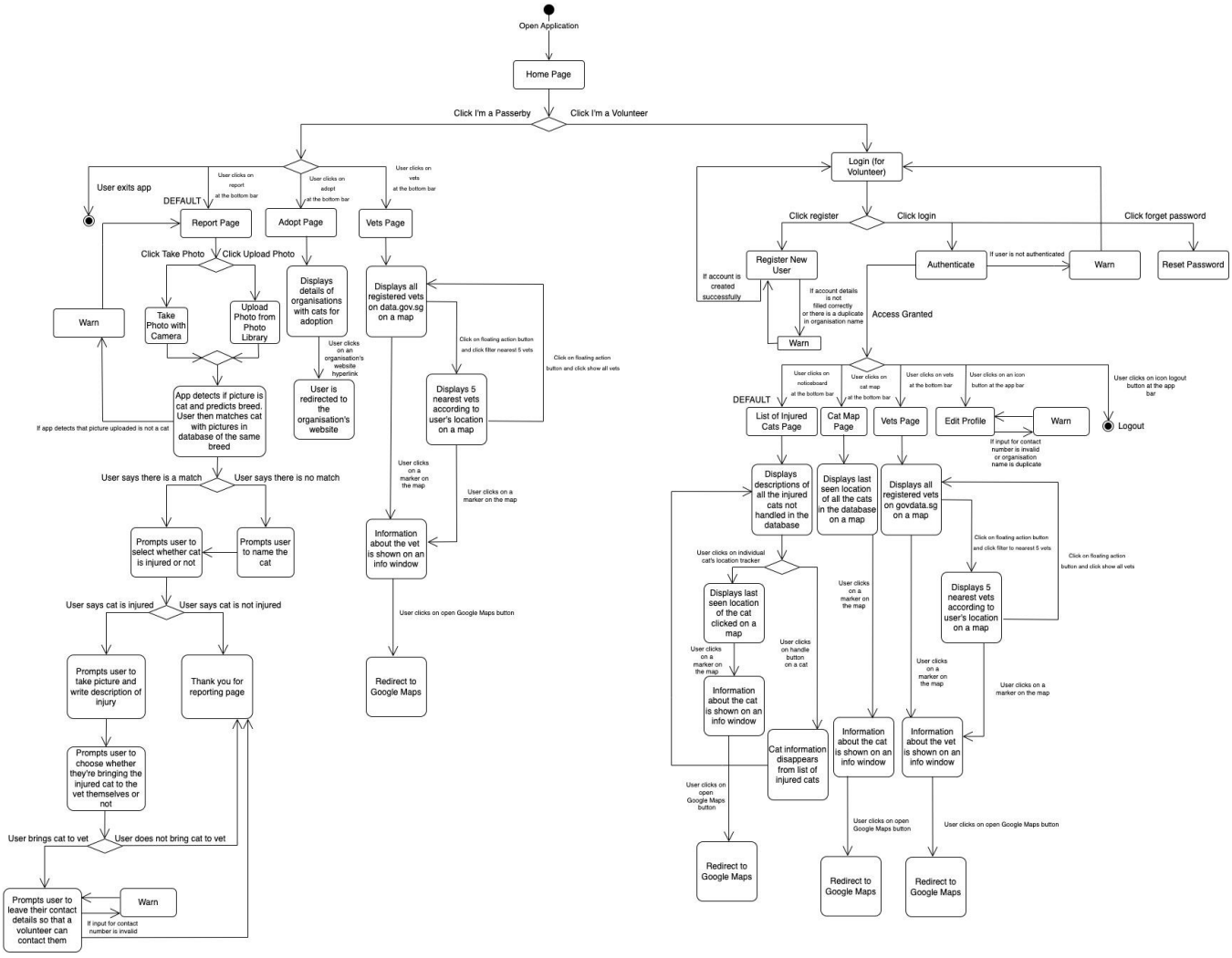


Use Case ID:	14		
Use Case Name:	Adopt a Stray Cat		
Created By:	Hu Zhuangyu	Last Updated By:	Valencia Lie
Date Created:	05/02/2022	Date Last Updated:	16/04/2022

Actor:	User (passerby), Database
Description:	User (passerby) can view the list of adoption centers if he or she wants to adopt a cat.
Preconditions:	1. Feature is made available after the user (passerby) clicks on the “I am a passerby” button on the main landing page of the app.
Postconditions:	-
Priority:	Low

Frequency of Use:	Low
Flow of Events:	1. User (passerby) clicks on the “Adopt” button at the bottom of the screen. 2. The list of adoption centers will be displayed with information such as its full name, telephone number, email address, physical address, opening hours and adoption fee(if applicable) 3. The hyperlink to the adoption centers’ website will be available.
Alternative Flows:	AF-S16: User(passerby) clicked on the hyperlink: 1. User(passerby) will be directed to the adoption center’s website
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

6.4 Dialog Map

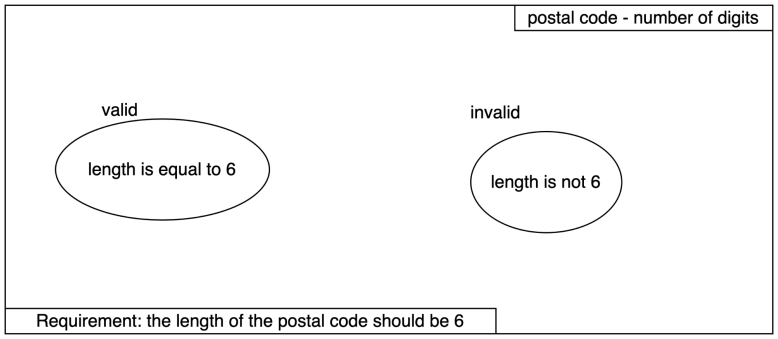


6.5 Testing

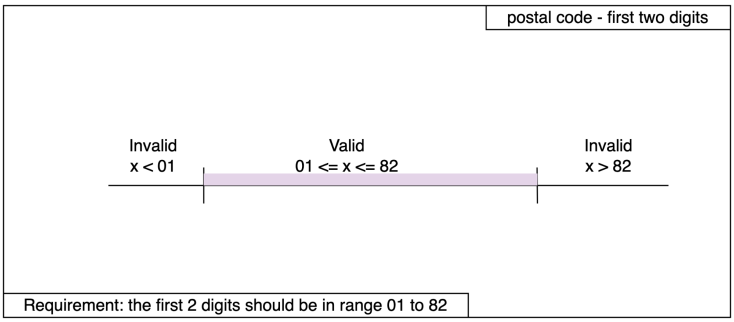
6.5.1 Black box

Black box testing is used to test our application so that software functionality can be tested without necessarily knowing the internal structure of the system, in order to better analyze app requirements and specifications. Both equivalence class and boundary value testing methods were used.

- User input location:
    - Equivalence class testing



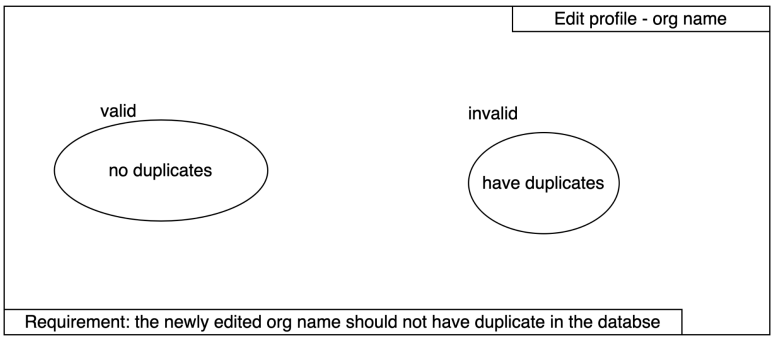
    - Boundary value testing

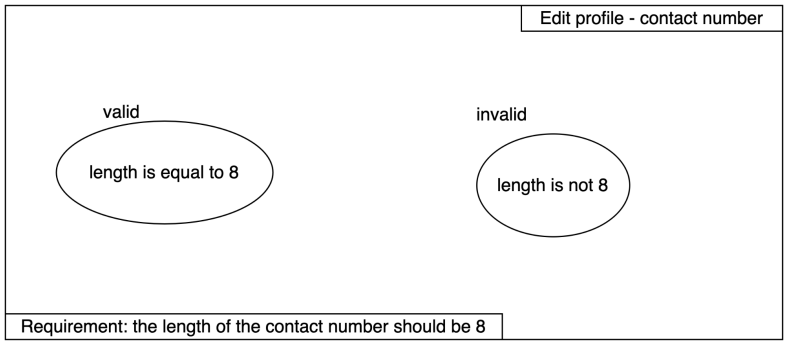


    - Combined test cases

Postal Code	Length	First two digits	Expected result	Test result
000000	6	00	Error message	Error message
1234567	7	12	Error message	Error message
637659	6	63	1.3507, 103.6810	1.3507, 103.6810
050335	6	05	1.2826, 103.8431	1.2826, 103.8431
278447	6	27	1.3090, 103.7992	1.3090, 103.7992

- Edit profile:
    - Equivalence class testing





    - Combined test cases

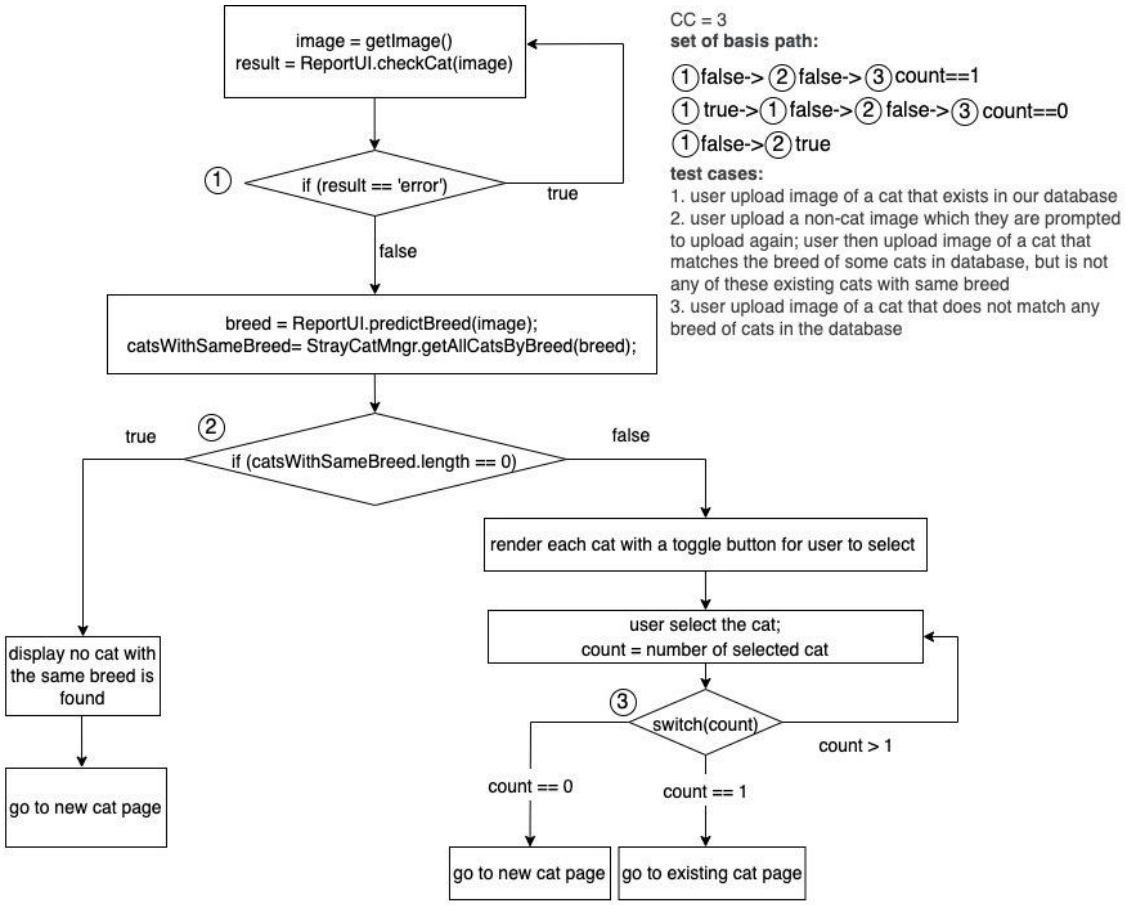
Type of input	Test ID	Input Description	Expected Output	Actual Result
Organisation name	1	spca (spca is already registered in our database)	Dialog Box showing that there is a duplicate organization name after register button is pressed	Dialog Box showing that there is a duplicate organization name after register button is pressed
	2	cat welfare society (it has not been registered in our database yet)	Dialog box showing information is successfully updated and screen changes to display state after confirm button is pressed	Dialog box showing information is successfully updated and screen changes to display state after confirm button is pressed
Contact number	3	123456	Dialog box showing that contact number can only be of 8 digits after confirm button is pressed	Dialog box showing that contact number can only be of 8 digits after confirm button is pressed

	4	123456789	Dialog box showing that contact number can only be of 8 digits after confirm button is pressed	Dialog box showing that contact number can only be of 8 digits after confirm button is pressed
	5	12345678	Dialog box showing information is successfully updated and screen changes to display state after confirm button is pressed	Dialog box showing information is successfully updated and screen changes to display state after confirm button is pressed
Address	6	Singapore 123456	Dialog box showing information is successfully updated and screen changes to display state after confirm button is pressed	Dialog box showing information is successfully updated and screen changes to display state after confirm button is pressed
All	7	If any text field is not filled	Dialog box showing that all text fields should be filled in after confirm button is pressed	Dialog box showing that all text fields should be filled in after confirm button is pressed

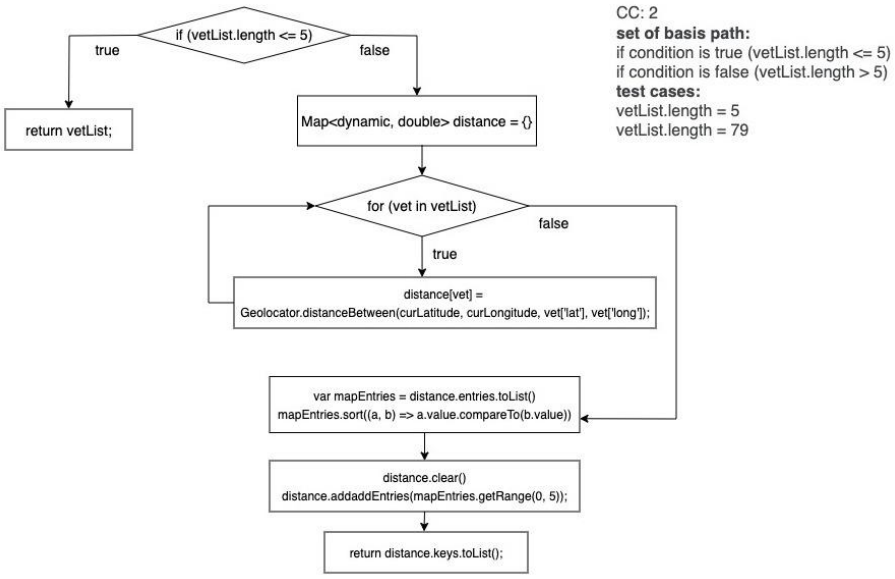
6.5.2 White Box

In contrast to Black Box Testing, White Box Testing is a testing technique we use to verify flow of input-output and to improve our app's design, usability and security. Code is also made available during testing.

- Report function
    - Control Flow Testing



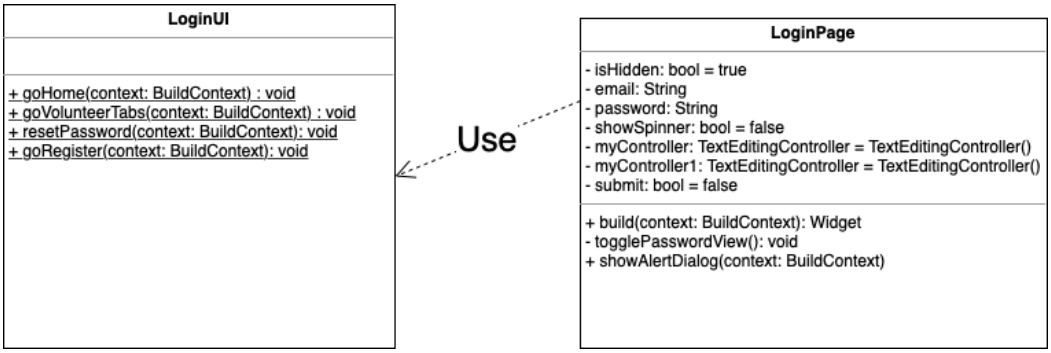
- Filter from location
    - Control Flow Testing



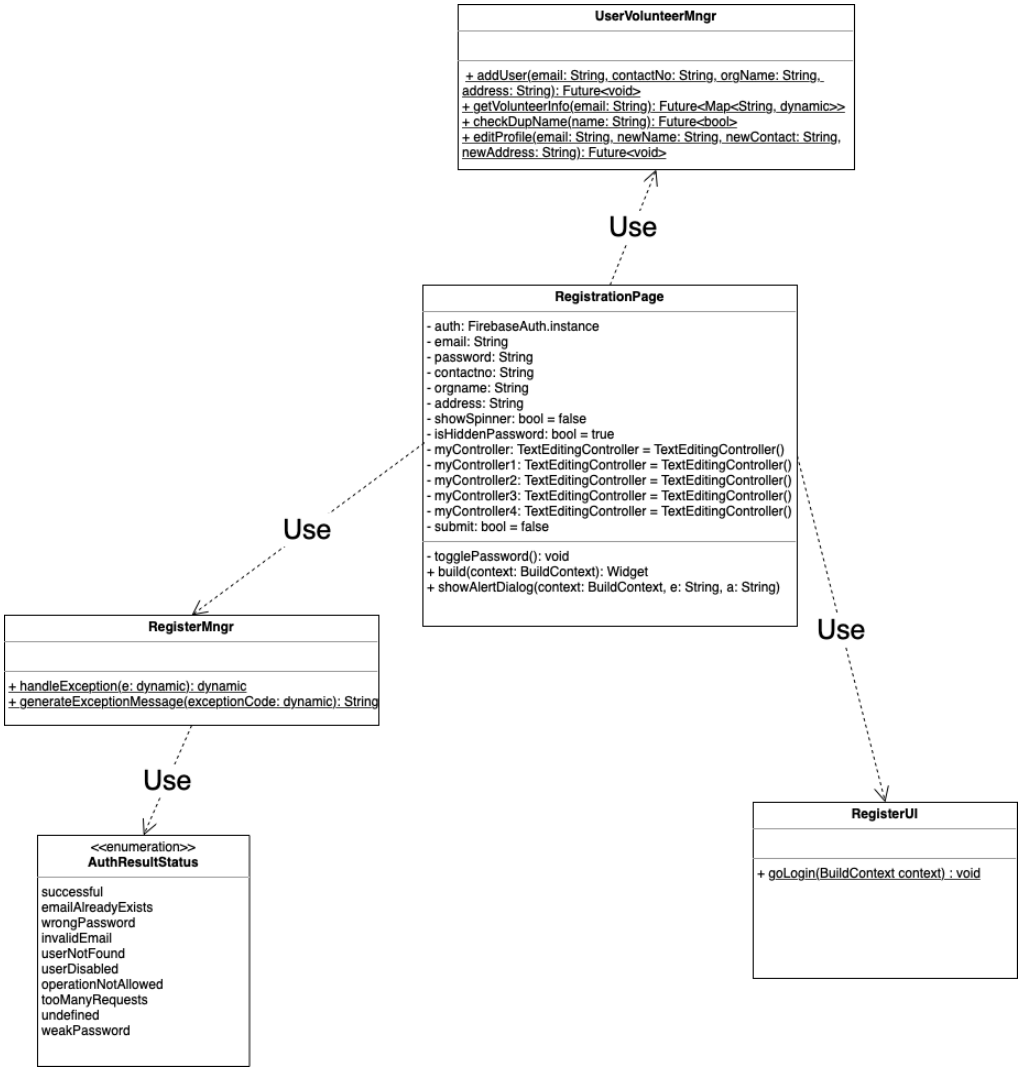
7 Design Models

7.1 Conceptual Models: Class Diagrams

1. Login function

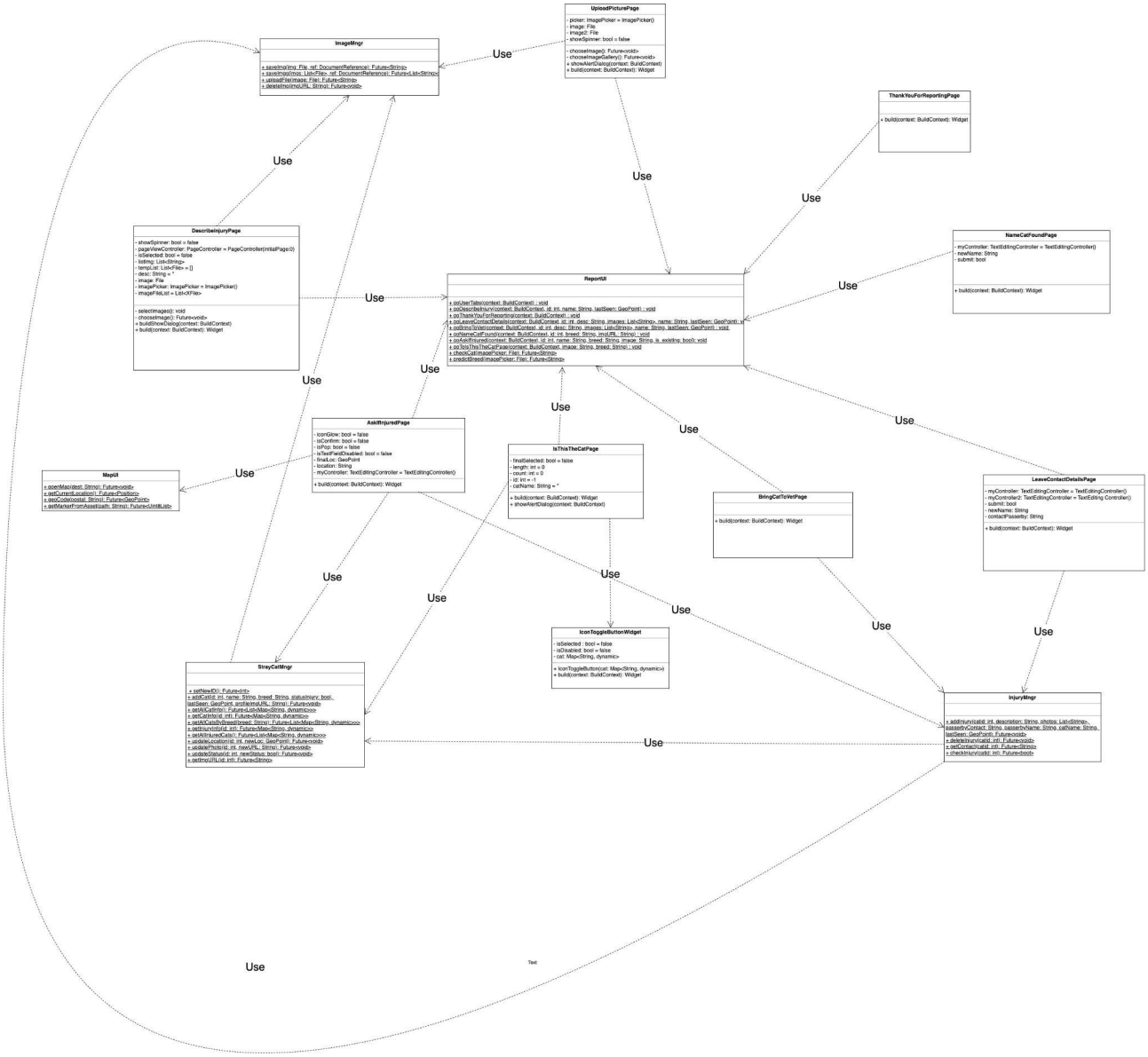


2. Register function

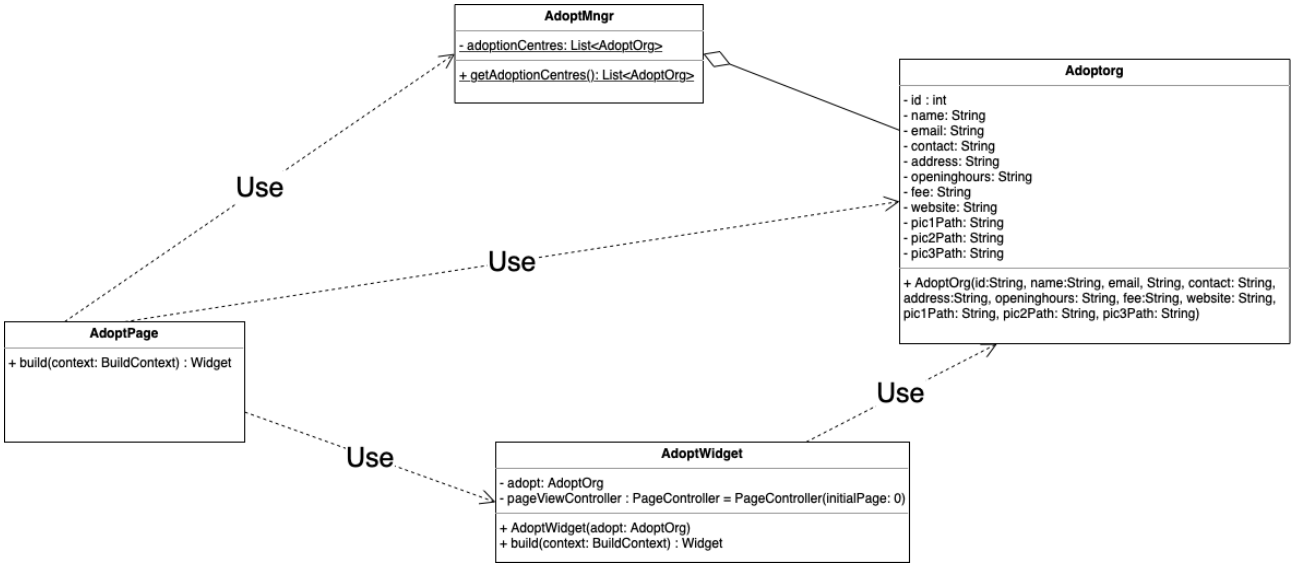


3. Edit Profile function

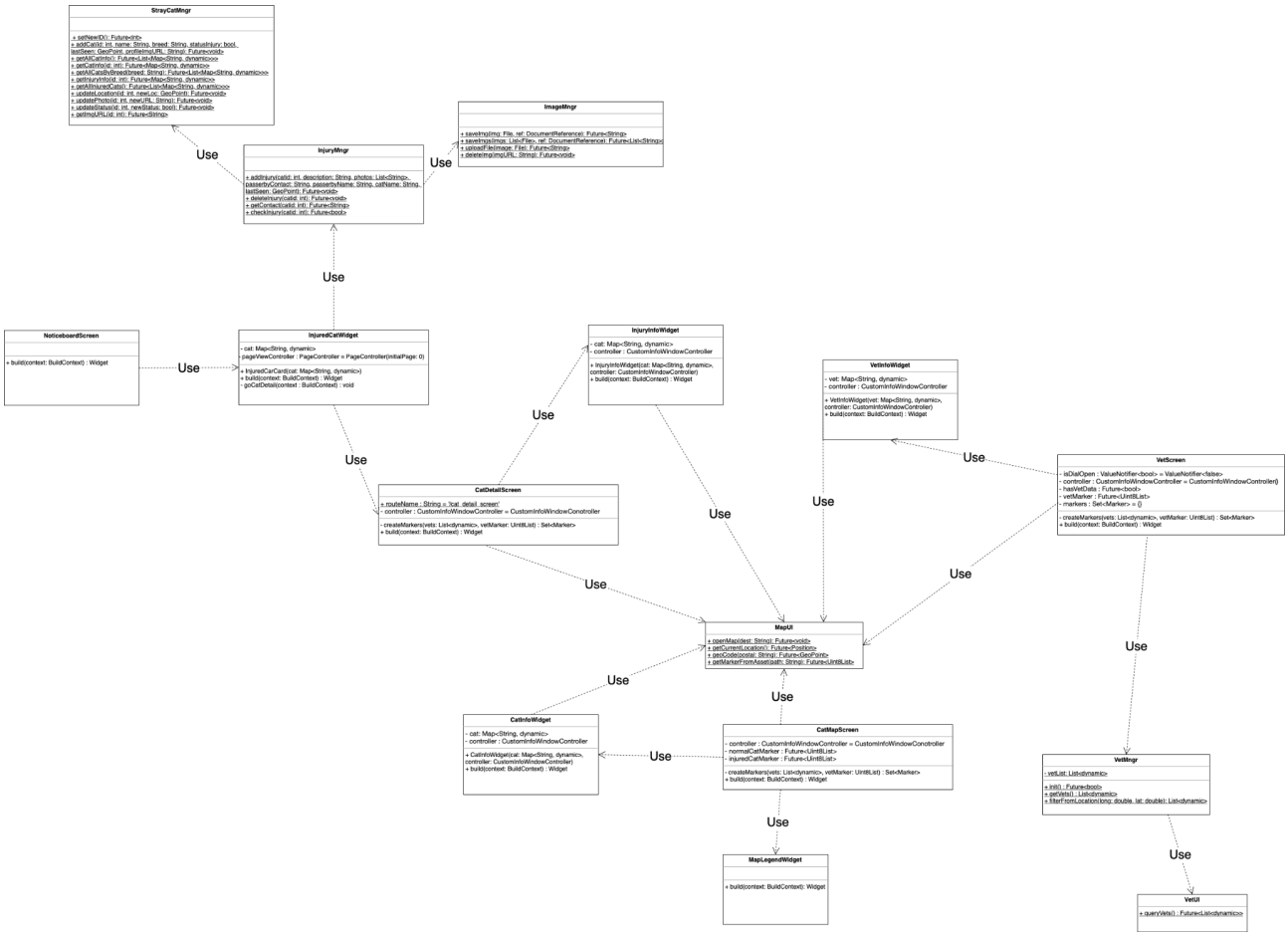
4. Report function



5. Adopt function



6. Noticeboard, cat map and vet function



7.2 Dynamic Model: Sequence Diagrams

Please refer to the sequence diagrams per use case description as shown above in section 6.3 Use Case Descriptions.

8. Software Design Principles

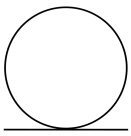
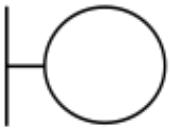
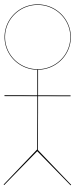
8.1 Single Responsibility Principle

This principle focuses on ensuring that a class has one, and only one, reason to be changed. Within our application, we have created various different managers to manage features respectively.

- Every Manager class has one job
    - e.g. StrayCatMngr handles get and set methods pertaining to stray cat only, while UserVolunteer handles get and set methods for user volunteer only
- Every UI class has one job
    - e.g. LoginUI and RegistrationUI are separate and handle login and registration only respectively

This means that none of the classes have multiple responsibilities and instead assumes only one important role.

8.2 Encapsulation

 Entity	 Control	 Boundary	 Actor
Represents long-lived information relevant to stakeholders • AdoptOrg	Processing and logic to coordinate and control objects involved in use case • AuthenticationMngr • StrayCatMngr • ImagesMngr • InjuryMngr • AdoptMngr	Interaction with external actors (users and external systems) • LoginUI • RegisterUI • ReportUI	An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external

	• UserVolunteerMngr etc.	• MapUI • VetUI	objects that produce or consume data. • Breed Detector API • Database • Object Classification Model • GovData API

We also demonstrated the OO principle encapsulation through sectioning our classes into 3 main class stereotypes: Boundary, Entity and Controller. By doing so, the main functionalities of our working program (what goes "under the hood") that are coded under the Controller classes will not be visible to the user, who only interacts with the Boundary class. In turn, the Boundary and Controller classes will interact with each other; the user will never directly interact with the Controller classes.

Appendix A: Glossary

Data Dictionary

Term	Description
User(passerby)	A member of the public who happened to find a stray cat and may want to report its discovery or notify volunteers of an injured stray cat
User(volunteer)	A member of the volunteer group who wants to tend to newly discovered injured stray cats. Each volunteer group shares a common account
Application	Refers to the Stray Finder mobile application
GPS	Global Positioning System that will detect the current live position of the user
Registration	User (volunteer) has to register for an account to be able to get notified for newly discovered injured stray cats
Stray cat	An un-owned domestic cat which roams the streets
Adoption	A stray cat which can be taken in by the member of the public as their own
Vet	A clinic/hospital with facilities for the treatment, cure and alleviation of disease in animals. The vet clinics used in the application are taken from a list of licensed vet clinics in Singapore provided by data.gov.sg API.

Appendix B: Analysis Models

System Architecture Diagram