Final Report

AppTrackWildlifeDiseases

Instructor: Dr. Edward A. Fox

Client: Luis E. Escobar

Authors:
Justin Vu
Shangzheng Ji
Tenghui Zhang
Jiarui Lyu

May 7, 2021
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ask current location permission</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Project Description</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Leader Profile</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Mange description</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Upload task</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Image picker</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>Request Camera Permission</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>Other Sections</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>Select a location on the map</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>Successful Uploaded</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>Upload Fail</td>
<td>23</td>
</tr>
<tr>
<td>12</td>
<td>Empty List</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>Data Records</td>
<td>25</td>
</tr>
<tr>
<td>14</td>
<td>User website view</td>
<td>26</td>
</tr>
<tr>
<td>15</td>
<td>Website Marker Popup View</td>
<td>27</td>
</tr>
<tr>
<td>16</td>
<td>Admin Website View</td>
<td>27</td>
</tr>
<tr>
<td>17</td>
<td>Admin Map Download Function</td>
<td>28</td>
</tr>
<tr>
<td>18</td>
<td>Firebase Data.CSV File Detail</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>Clone the project from GitHub</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>CocoaPods installation finished</td>
<td>31</td>
</tr>
<tr>
<td>21</td>
<td>Podfile content</td>
<td>32</td>
</tr>
<tr>
<td>22</td>
<td>Pod update Command</td>
<td>33</td>
</tr>
<tr>
<td>23</td>
<td>The TrackDiseases.xcworkspace file</td>
<td>34</td>
</tr>
<tr>
<td>24</td>
<td>Firebase Storage</td>
<td>34</td>
</tr>
<tr>
<td>25</td>
<td>Firebase Firestore</td>
<td>35</td>
</tr>
<tr>
<td>26</td>
<td>Firebase Hosting JSON file</td>
<td>36</td>
</tr>
<tr>
<td>27</td>
<td>Firebase Hosting</td>
<td>36</td>
</tr>
<tr>
<td>28</td>
<td>Mapbox Implement</td>
<td>37</td>
</tr>
<tr>
<td>29</td>
<td>MapboxGL Object Creation</td>
<td>37</td>
</tr>
<tr>
<td>30</td>
<td>Firebase Upload Website</td>
<td>38</td>
</tr>
<tr>
<td>31</td>
<td>Process of taking photo for the wildlife disease</td>
<td>39</td>
</tr>
<tr>
<td>32</td>
<td>Process of picking up a photo from the photo library</td>
<td>40</td>
</tr>
<tr>
<td>33</td>
<td>Process of recording the geographic information</td>
<td>41</td>
</tr>
<tr>
<td>34</td>
<td>Process of reviewing uploaded histories</td>
<td>42</td>
</tr>
<tr>
<td>35</td>
<td>Process of uploading the information</td>
<td>43</td>
</tr>
<tr>
<td>36</td>
<td>Download all information from the cloud database</td>
<td>44</td>
</tr>
<tr>
<td>37</td>
<td>Zipped Web applications</td>
<td>48</td>
</tr>
<tr>
<td>38</td>
<td>Timeline</td>
<td>49</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Functionality Test</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>Users Experience Test</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Problems and Solutions</td>
<td>51</td>
</tr>
</tbody>
</table>
1 Executive Summary / Abstract

The AppTrackWildlifeDiseases is a project originated by Luis E. Escobar who focuses on the study of infectious diseases. The goal of our project is to provide a free-of-cost smartphone application and a website to achieve the purpose of early detection of the wildlife disease called mange. As our client Luis E. Escobar expects, our smartphone application should be able to obtain information from different users, such as the picture uploaded by the user about the mange, the date the picture was uploaded, the geographic location where the picture was taken, etc. Also, we are supposed to display the information on our website.

This project is new, so we started it from the design part. We start to design the application first. We aim to satisfy the client requirement, collect the information, and at the same time, be operation friendly to the user. We put a lot of time to design the user interface in the application. In the application, we design many mechanisms to handle the errors from the user or environment.

To be easily built, we use the SwiftUI to implement the iOS application in Xcode. In our iOS app, we will give the introduction of our project and will also provide a questionnaire for the users which will ask them to upload the image of mange, provide related information of the date and the geographic location, and indicate their knowledge level regarding the disease.

Then we use the cloud service, Firebase, as the data center. Firebase is easy to manage for people who do not have much knowledge about the database, and it saves the local computational resources for the client. The information obtained from the iOS application will be uploaded to Firebase Storage and Firebase Firestore (database).

The last part is the website. We use the Firebase Hosting service to host our website. Therefore, the client does not need to get their own domain for the website. We design two versions of the website for different types of users, the administrator and the normal user. From the administrator website, the admins can save all information into local storage in the CSV file format. The website can access and retrieve all the data from the database and visualize the distribution of users’ records by geographic information.

During the project time period, we were able to design and implement the application and website, so all products pass tests and satisfy the client requirements. We could test the application in good and bad Internet conditions and introduce some mistake operations to test the stability of the application. The application passes the test cases. The website can successfully read the data and display the distribution accuracy on the map.

In the future, we hope the collected information and the distribution on the map can contribute to VT students in related areas and the client’s research.
2 Introduction

The destruction of wildlife by disease outbreaks is enormous. Especially for rare animals and endangered species, there will be an unpredictable and possibly very serious effect. Therefore, early detection of wildlife diseases is extremely important. Citizen science provides opportunities for the early detection of wildlife diseases. Our project will focus on developing applications and websites that are free to the public for tracking and management. This kind of tracking and management of infectious diseases in wildlife is easily discovered by non-experts. Since mange is spreading globally, and an epidemic of bears is ongoing in Virginia, this disease is an excellent model for implementing this application.

Monitoring wildlife diseases is important to humans as well as to wild animals. Certain diseases (such as HIV) were Zoonoses at first, but later mutated into human-specific strains [10]. For example, in 2012, the Middle East Respiratory Syndrome was transmitted from camels to humans. Zoonoses disease is an infectious disease that spans from non-human animal origin to human origin. The pathogens of Zoonoses diseases may be bacteria and viruses, and they may also be transmitted to humans through food, water, or the environment. Among all newly discovered infectious diseases and many existing infectious diseases, most are Zoonoses diseases. Some Zoonoses diseases may cause repeated disease outbreaks, such as Ebola and COVID-19.

It is known that there are a large number of new or undocumented diseases in certain wildlife populations, so what we need to do now is to collect more wildlife cases with suspected diseases. In the past, the epidemiological monitoring of wildlife diseases usually relied on samples harvested from hunters or related experts. But the biggest problem is that although this method is cost-effective, due to the deviation of disease distribution and sampling, the samples collected by hunters cannot represent an entire population. A more effective method is “national surveillance”, that is, provide everyone a method to upload the image of potential wildlife disease and the information can be used to inform wildlife disease surveillance strategies. This is what our project wants to achieve.
3 Requirements

3.1 iOS Application

Our client asked that the iOS application contain an interface to collect the data the client expected. Following is that list of data:

- Image of the wildlife disease.
- Geographic information (e.g., longitude and latitude).
- The date.
- A small questionnaire to investigate the knowledge level of the wildlife animal.
- Contact information (optional).
- Additional information about the image (optional).

We also need to package all the information collected in the Application and upload that to the database.

3.2 Website

For our website, our client requires us to have the admin version of all valid data records of the photo displayed on the map. For normal users, only the picture and its corresponding location on the map can be seen. However, the client would like to see the complete and organized information of each uploaded photo. To meet this requirement, we create another website which is an “admin version” of the normal website, where only our client knows the URL. On this website, our client can directly see the URL, UUID (Universally unique identifier), photo’s uploaded date, the longitude and latitude of photo’s uploaded geographic location, and the users’ familiarity with mange.
4 Design

Our project is mainly divided into 3 parts: the free-of-cost smartphone application and the website that together constitute the front-end, and the database which is the back-end. We will use the SwiftUI framework and Firebase framework to write an iOS application using Xcode. Our application has three tabs – AboutUs, Upload, and History. AboutUs is designed as the home page and provides the description of our project, our client Luis E. Escobar’s profile, and an introduction to mange obtained from Wikipedia. In the Upload tab, there will be 2 sections – the questionnaire section and the photo-pick section. The questionnaire is served to collect the user’s familiarity with mange. The photo-pick section should realize the functionality of taking a photo of the wildlife disease, picking up the photo of the wildlife disease from the photo library, and recording geographic location and date of the photo. Besides, our app supports offline uploading which means even if the user is in the forest and his phone has no signal, he can still upload the photo. In the History page, the user can review their personal uploading history and check if everything is correct.

Our requirement is that the database is free, and there is enough data storage space. Thus, we use Firebase as our database. In Firebase’s storage, the latitude and longitude of the photo’s geographic location will be processed, all information that the user uploaded from the app will be dealt with, and then it will return a corresponding URL. The data prepared for the website will be stored in the database and is a composite of the longitude and latitude.

Our website is designed to be a map displaying dozens of location markers which are generated based on the longitude and latitude location stored in Firebase. The number of location markers depends on the amount of data uploaded from the iOS application. The detailed photo and the time the photo is taken can be viewed by clicking the location marker. The map has the left-side bar providing a brief introduction about the project and the information of our client Luis E. Escobar. As our client demands, our website is designed with an admin version with the functionality of reviewing the complete and organized information of all uploaded photos. The admin version can be viewed by clicking the client name in the left-side bar.
5 Implementation

5.1 iOS Application

The application we implemented focuses on two parts: (1) get information the client required and (2) package that to the website. We hope the application becomes a window to introduce the aim of the research. Because this is a new project, we implement the application from scratch. We design the introduction page for when the application is launched, so the user can know this research, and know every contribution the user makes via the application is important for the research.

We design an interface to collect the information. The information includes:

- Geographic information
- The image the user takes
- Date of the image
- Additional details of the image

The geographic information is the most complex to get. We consider this application may be used in the forests or other places where the GPS signal may not cover. Therefore, the application cannot get the latitude and longitude, geographic information, in time. To deal with this situation, the solution is that we implement an interactive map, so the user can choose the location manually in the map, and the application will convert this location into geographic information. For the other information, we also provide the interface. For the image, we provide an image picker in the application. Users can take the photo by camera in the application or pick a photo from the photo library. For the image picker, we add one limitation that the user cannot upload duplicated images to the database.

For packaging the collected information to the website, we choose to set up a database and storage between the application and website. Therefore, the application can upload the packaged information to the storage and database, and the application can read all the information from the database and storage. We choose to use the Firebase service to be the database and storage because the service is on the cloud and due to the limited resources we have in this project. The application uses the Firebase SDK to upload data to the storage and database. Because the application may be used where signals are not available, the Firebase SDK is supporting the upload asynchronously. When the signal condition becomes good, the upload task can resume uploading to the storage and database.

5.2 Web Application

The application we implemented focuses on three parts: get JSON information from the Firestore, display the given information on the map based on the
geometry information, and set up the download function for the admin user. Because this is a new project, we implement the application from scratch. We design the UI of the website to operate when either of the two different website URLs is launched. By entering different URLs, the users can get into different websites.

Displaying the map with popups is the most complex and main function of the website. It requires two steps for reaching this goal and we decide to use the Mapbox and Firebase SDK. After the iOS application uploads all the necessary information to the Firebase server, we can read the JSON array from Firebase and reorganize the information to save in local storage for the Mapbox to implement. By using the Mapbox SDK, we are able to create a map with popups to display the necessary information. We consider this web application may view, in a different browser, all of the text, image, and map. The display must be flexible based on the size of the screen.

For the hosting of the two websites, both of them are hosted on the Firebase server. We define the basic roles of the Firebase server to be dictionary, port, directory, and so on. After deploying our web application to the Firebase Hosting, the hosting server begins and both of the websites are sharing the same Firestore database. The admin website has the right to download all the JSON array which is saved in a well-organized CSV file.
6 Testing/Evaluation/Assessment

Test group:

- one professional tester: Mariana Castaneda Guzman
- 20 normal users: friends of our group members.

We ask two different groups of people to test the App and website. Our tester not only tested whether our app and website met all the standards, but also provided us with suggestions for improvement and optimization advice in terms of details and content. Normal users tested our app and website regarding functionality and user experience.

6.1 Professional Tester

6.1.1 Application

Our app’s functionality satisfies all the requirements and each Tab is coming along nicely. However, we have to change typos and format errors, such as change “flue” to “flu”, add missing spaces after some periods, etc. Also, in order to optimize our app, we are suggested to add a section of “Additional Comments” so that the users can freely leave their thoughts and have a better user experience.

6.1.2 Website

Our website successfully displays all the information obtained from the app and allows either the admin or the general users to have their own website version, but we still need to beautify the website’s appearance, such as find a higher resolution VT logo, increase font size and add more spaces. Moreover, our website should be able to show the information uploaded by the user based on the “Additional Comments” of the app.

6.2 Normal Users

6.2.1 Functionality Test

In order to test each Tab of the app, normal users are provided with preconditions and expected outcomes according to the given test steps. Similarly for testing the website, since normal users are only available for the user website, they are only provided with the user website view of the given test steps and the corresponding expected outcomes. We will collect answers (Yes/No) from every normal user whether the test results meet the expectations. Table 1 shows the number of users who obtained the same results as expectations.
6.2.2 Users Experience Test

After finished the functionality test, 20 normal users are asked to give a score from 1-100 for each aspect based on their using experience. The numbers in Table 2 shows the average score of 20 normal users. Judging from the above scores, users generally can quickly accept and use our app and website, and our design seems to be in line with users’ habits. The only downside is that it takes a little time for users to wait to receive the app responses.

<table>
<thead>
<tr>
<th></th>
<th>Application</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Design</td>
<td>96.4</td>
<td>95.2</td>
</tr>
<tr>
<td>Functional ease of use</td>
<td>97.8</td>
<td>98.3</td>
</tr>
<tr>
<td>System function response (interface, fluency, startup time)</td>
<td>92.6</td>
<td>97.4</td>
</tr>
</tbody>
</table>

Table 2: Users Experience Test
7 Users’ Manual

7.1 iOS Application

For the iOS Application, the main function is to let the user report the wildlife disease to the database and provide project description and disease information to make the user have a better understanding of mange and our project.

After downloading and installing the application on the phone, the application will ask for permission to access the current location.

As shown in Figure 1, a message box shows up to ask the location permission. Giving location access permission is required for the application to work properly. The Allowed Once option only gives a one-time access grant to the application, so when you next launch this application, the message box will show up again to request access permission.

7.1.1 About Us

The application will display the description of the project and wildlife disease knowledge when the application is launched. We have three sections to implement this function.

As shown in Figure 2, this section displays the project introduction. You can click the blue icon beside the title to access the project homepage. In Figure 3, the second section is the leader profile; you can tap the blue words and then jump to the web page to learn more details. As shown in Figure 4, the last section gives a basic introduction to mange; you can see more details by tapping the blue icon next to the title.

7.1.2 Upload Task

For the Upload task, this page in the application can collect a user’s data and verify the input from the user to reject the invalid user behaviors via popping out alert information.
Figure 1: Ask current location permission
COVID-19 and many other emerging diseases, such as Ebola and the avian flu, have origins in wildlife. Thus, it is crucial to monitor wildlife diseases. However, this is expensive and time consuming, and the diseases could emerge in areas without monitoring. Citizen science bring an opportunity for early detection of wildlife diseases. This project will focus on the development of an app, freely available to the public, to track mange, an infectious disease in wildlife that is easy to detect by non-experts. Mange is spreading globally and Virginia has an ongoing epidemic in bears. Thus, this disease is a great model to implement the app, which can be expanded to other diseases in the future.

Figure 2: Project Description

Profile

Luis E. Escobar
Asst Professor AY
✉️ escobar1@vt.edu
🌐 More Details

D.V.M., Universidad de San Carlos, Guatemala City, Guatemala (Avian Flu in Wild Birds) (2009)

M.Sc., Wildlife Management, Universidad de San Carlos, Guatemala City, Guatemala (Community Ecology of Fleas in Guatemala) (2011)

M.Sc., Veterinary Sciences, Universidad Andres Bello, College of Ecology and Natural Resources, Chile (2012)

Ph.D., Conservation Medicine (Summa Cum Laude), Universidad Andres Bello, College of Ecology and Natural Resources, Chile (Ecology and Biogeography of Bat-borne Rabies) (2014)

Figure 3: Leader Profile
Mange is a type of skin disease caused by parasitic mites. Because mites also infect plants, birds, and reptiles, the term “mange” or colloquially “the mange”, suggesting poor condition of the hairy coat due to the infection, is sometimes reserved only for pathological mite-infestation of nonhuman mammals. Thus, mange includes mite-associated skin disease in domestic animals (cats and dogs), in livestock (such as sheep scab), and in wild animals (for example, coyotes, cougars, and bears). Since mites belong to the arachnid subclass Acari (also called Acarina), another term for mite infestation is acariasis. Parasitic mites that cause mange in mammals embed themselves in either skin or hair follicles in the animal, depending upon their genus. Sarcoptes spp. burrow...

Figure 4: Mange description
When for the first time, a user opens the upload task view, as shown as Figure 5, the screen shows two sections. The first section is a questionnaire to collect a knowledge level from the user. The form is a multiple choice. The second section is an image picker. This section can take the photo from the camera or photo library.

As shown in Figure 6, an action sheet will show up, and the user can choose a way to pick the photo.

If the user wants the camera, as shown in Figure 7, the application will request the permission from the user to access the camera, since the camera permission is required. After the photo is selected, other selections show up as is illustrated in Figure 8. The date picker allows the user to choose the date; the default date is the current date. The next section is a text entry box. Users can type contact information for the current photo, which is optional.

The last section can get the geographic information where the photo was
Figure 6: Image picker
Figure 7: Request Camera Permission
Figure 8: Other Sections

Select date when picture was taken

**select a date**  **Apr 27, 2021**

The default date is today

Contact Info (Optional)

email and full name

Additional Comments (Optional)

Select location where image was taken

**Select location**  

Long: 0  Lat: 0

To submit your observation click on “Upload” in the right top corner of your screen.
taken. This section will only show up when the user picks up the photo from the photo library. If we click this section, the application spans a map and a pin annotation is on the map as shown in Figure 9. The user can drag the red pin annotation through the map to choose the location.

![Map with pin annotation](image)

Figure 9: Select a location on the map

After the user fills in all of the information, the user can use the upload button on the right-top corner shown in Figure 9. If the uploading process succeeds, a successful message will show up which is shown in Figure 10. A bad internet connection will lead to a failure of uploading which is shown in Figure 11.
Figure 10: Successful Uploaded
7.1.3 History

The History function records all uploaded records. The user can see all data records in the local database. When this application is opened for the first time, as shown in Figure 12, the list is empty.
Figure 12: Empty List
As shown in Figure 13 when the user has successfully uploaded the data to the cloud database, the list will contain all of the data records. For each item in this list, it shows the date and time of the uploaded record, and tells the geographic information and the detailed address.

7.2 Website

For the website, the main function is to let the users view the picture which is taken by the photographer and the time that the picture was taken. The website will display the location markers based on the longitude and latitude location which the iOS application uploaded and stored in the Firebase Firestore database.

There are two kinds of website URL. The first one “trackwildlifeanimal-58tz.web.app” is for the user and the second one “trackwildlifeanimaladmin-58tz.web.app” is for the admin. Both of the websites have a big school icon and website title of “Worldwide Animal Disease Distribution Map.”
side, there is a brief introduction about the project and the information of the project manager – Luis E. Escobar. The users can view his personal website by clicking his name in the description portion. On the right side, there is the map view. Hundreds of markers based on the iOS application upload can be shown on the map.

![Map View](image)

Figure 14: User website view

As shown in Figure 14, each blue underlined word is a hyperlink. When the user clicks on “Luis E. Escobar”, it will jump to Luis E. Escobar’s personal website. When the apptrackwildlifediseases@gmail.com or escobar1@vt.edu is clicked, the website will jump to the default mail application and create a new draft email to apptrackwildlifediseases@gmail.com or escobar1@vt.edu.

When the user clicks the markers, it will show the information as Figure 15. Out of respect for privacy, the website only displays the time and date of the picture taken and the diseased animal picture which is uploaded by the iOS application. Users are allowed to drag, spin, zoom in, and zoom out by using the icon on the right top corner side.

The admin has a unique website. The admin can do all of the operations which a user does, and the admin is allowed to download all of the pictures and JSON information which is saved in the CSV file. The view of the admin website is shown as Figure 16.

In Figure 17, when the blue Download button is clicked, all of the pictures’ URL and the JSON data which is stored in the Firebase Firestore will be converted into CSV format and downloaded as a CSV file which is called FirebaseData.csv.

The format of the FirebaseData.csv file is shown in Figure 18. It includes the contact information which makes it easy for the professor to contact the photographer.

Also provided is the URL which is the location of each picture stored, the
date which is the time the picture was taken, the UUID which is the unique ID for the data stored in the Firestore database, the choice which is the knowledge level of the photographer relative to the disease, and the geo longitude/latitude which is the location where the photographer found the animal with disease. By using this organized CSV file, the admin can find the information which he wants, immediately.
Wildlife animal disease tracker app is an application with the purpose of gathering, analyzing, and freely sharing information about wildlife diseases around the world to power new data-driven approaches to science, conservation, and education.

To download all current observations, click on the button below.

If you would like to contribute to the database, you can record your observation by downloading the app. "TrackDiseases."

If you encounter any errors, please contact Dr. Michael E. Movie at mmovie@vims.edu or

Figure 17: Admin Map Download Function

<table>
<thead>
<tr>
<th>Date</th>
<th>Image URL</th>
<th>Contact Information</th>
<th>Disease Identification</th>
<th>Additional Information</th>
<th>Latitude</th>
<th>Longitude</th>
<th>U.S.</th>
<th>Canada</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013/4/12 11:35</td>
<td><a href="https://firebasestorage.googleapis.com/">https://firebasestorage.googleapis.com/</a>....</td>
<td>NC</td>
<td>Competent</td>
<td>Photo of disease, found on May 12, 2013</td>
<td>34.478745</td>
<td>-78.9445</td>
<td>45</td>
<td>90</td>
</tr>
<tr>
<td>2013/4/12 12:35</td>
<td><a href="https://firebasestorage.googleapis.com/">https://firebasestorage.googleapis.com/</a>....</td>
<td>NC</td>
<td>Competent</td>
<td>Photo of disease, found on June 12, 2013</td>
<td>44.578735</td>
<td>-83.9545</td>
<td>45</td>
<td>90</td>
</tr>
</tbody>
</table>

Figure 18: FirebaseData.CSV File Detail
8 Developer’s Manual

8.1 Environment

- iOS application
  - Xcode version: 12.2 or later
  - iOS target version: 14.4 or later
  - Platform: iPhone
  - CocoaPods version: 1.10 or later
  - Firebase Framework [3]
    * Version: 7.5.0 or later
  - Sep up a physical iOS device (iPhone) or use the iOS simulator (build in the Xcode) to run the app [6]

- Web application
  - Any JSON IDE
    * Recommendation: WebStorm 2020.2.4 or later
  - Platform: Web Browser
  - Firebase Framework [3]
    * Version: 7.5.0 or later
  - Open any web browser and type the website URL for entering the website

8.2 Xcode Installation

The Xcode only works in the MacOS system. You can download the Xcode via the App Store using your Apple ID. Xcode is mandatory for iOS developing.
8.3 Clone the Project

Before the project runs on the Mac the first time, we need to clone the code from GitHub [7]. The Xcode has integrated the Git component, so you do not need any installations in the computer. In the Xcode, we can directly clone the code from the remote repository (the URL in GitHub). This is shown in Figure 19 with the red box.

8.4 CocoaPods

After we cloned the project, we need to configure the project dependence using CocoaPods. CocoaPods is a dependency manager for the Swift language project, which helps us set the programming environment efficiently to satisfy the project requirements.

8.4.1 Installation

For CocoaPods, we recommend using the command-line tool to install CocoaPods. We can use the command:

```
$ sudo gem install cocoapods
```

After the command execution has finished, the terminal shows information that Successfully installed cocoapods-1.10.1. This is shown in Figure 20.
8.5 Firebase

Firebase is a platform developed by Google for creating mobile and web applications [11]. In the project, we use the Firebase Firestore service and Firebase Storage service as the cloud database and the cloud drive for saving the information from users. Then the website is also hosted by Firebase under the Firebase Hosting service. In this section, we will install the Firebase iOS SDK using CocoaPods. Please confirm that the version of CocoaPods is 1.10.1, otherwise, the SDK cannot install successfully.

Firebase SDK is the prerequisite to using other Firebase services. In the project directory, we already have generated the Podfile using CocoaPods. Regarding the Podfile, the Firebase pods that we needed in the app are listed in the Podfile by $ pod init command [1]. This shown in Figure 21.
The next step is to open the terminal and switch to the project directory, and use the script: $ pod update to build up the environment and configure the dependencies, which is shown in Figure 22.
After the command execution has finished, the project directory shows three new items: Pods, Podfile.lock, and TrackDiseases.xcworkspace. The most important one is the file with the extension named .xcworkspace, which is shown in Figure 23. Please pay attention to the difference between TrackDiseases.xcworkspace and TrackDiseases.xcworkspace. Opening the project via the TrackDiseases.xcworkspace can correctly compile the code and run the code properly on the device or the simulator.
Here you have finished the installation of Firebase iOS SDK! Next are several descriptions of services we used in the Firebase.

### 8.5.1 Firebase Storage

The Firebase Storage is a cloud storage for app developers who need to store and serve the user content. In this project, the image collected from the user will be uploaded into this Storage. In the Storage, as we designed, each wildlife disease have a folder. Currently, we only collect for one type of disease, so only one folder exists in the Storage, which is shown in Figure 24.

### 8.5.2 Firebase Firestore

The Firestore is a No-SQL and flexible cloud database. It keeps the data in sync between the application and the real-time listener, in other words, website. Please pay attention that the Firestore and Firebase Storage are different services, although both names contain “store”. In the Firestore, the architecture,
which is shown in Figure 25, is that the first level is the collection which is similar to the folder of a typical computer system. The next level is the document. Each document contains the record the user uploaded. The format of the content is JSON-like format, which is of key-value pairs.

**Figure 25: Firebase Firestore**

8.5.3 **Firebase Hosting**

Firebase Hosting provides fast and secure hosting for your web app, static and dynamic content, and microservices 5.

Firebase Hosting is production-grade web content hosting for developers. With a single command, you can quickly deploy web apps and serve both static and dynamic content to a global CDN (content delivery network). You can also pair Firebase Hosting with Cloud Functions or Cloud Run to build and host microservices on Firebase 5.

As Figure 27 shows, there are three websites hosted under the Google account: apptrackwildlifediseases@gmail.com. The first URL is the main domain. Since we need two URLs for admin and student use and consistency with the domain name, we create two separate URLs: https://trackwildlifeanimal-58tz.web.app and https://trackwildlifeanimaladmin-58tz.web.app. The first URL is for admin users, and the other one is for normal users. The back-end hosting structure is well written in the firebase.json file in the main directory as Figure 26. A user can directly use the `$firebase deploy— command in the terminal to upload the HTML file which is under the public directory.
Figure 26: Firebase Hosting JSON file

Figure 27: Firebase Hosting
8.6 Mapbox Instruction

Mapbox provides many tools to build maps into the website or web-based application. Mapbox GL JS \(^8\) is the JavaScript library for displaying Mapbox maps, which adds interactivity, and customizes the map experience in the web application \(^9\).

For adding the Mapbox component, we need to link the library in the HTML file as shown in Figure 28.

![Figure 28: Mapbox Implement](image)

After importing the Mapbox library, the website can create the map object with a unique token for identifying the user and using the Mapbox built-in function as shown in Figure 29.

![Figure 29: MapboxGL Object Creation](image)

8.7 Website Project Instruction

The first step is installing Node.js and npm which is included in Node.js component. If you have installed them, please go to the second step. For the details of installation, the Node.js \(^2\) official website gives detailed instructions.

The second step is installing the Firebase command tools by using the command `$ sudo npm install -g firebase-tools`. We can use the `firebase --version` command to check whether the installation is successful.

The third step is the Firebase login which will jump to the web browser for login. We have a Google account only to be used for this project which is AppTrackWildlifeDiseases@gmail.com. After login success on the web browser, our project connects successfully to the Firebase server.
All of the back-end hosting setting was written in the JSON file and the main function of the website was written under the public directory. As **Figure 30** shows, we can use the “firebase deploy” command to upload the website to the Firebase Hosting service.

![Firebase Upload Website](image)

**Figure 30: Firebase Upload Website**

### 8.8 Methodology

#### 8.8.1 User Goals

Our applications are made up of a free-of-cost smartphone application and a web application. Our applications have two types of users, the general user and the admin. The General User’s Goals are as follows: The free-of-cost smartphone application needs to support 6 General User goals:

- Take photo for the wildlife disease.
- Pick up the photo of the wildlife disease from the photo library.
- Record geographic location of the photo.
- Record date of the photo.
- Record users’ knowledge level with mange.
- Review personal uploading history.

The web application needs to support 1 goal:

- Get distribution of records on a map.

The Admin’s Goals are as follows: The free-of-cost smartphone application needs to support one Admin with 2 goals:
• Provide project description and information about the disease called mange to make the users have a better understanding of mange and our project.

• Upload the information to the cloud database.

The web application needs to support 2 goals:

• Retrieve all data from Google Firebase.

• Change each data’s information into landmark points and show it on the map.

8.8.2 Tasks and Subtasks

1. Take a photo for the wildlife disease. The task is separated into subtasks shown in Figure [31]

Figure 31: Process of taking photo for the wildlife disease
2. Pick up a photo from the photo library. The task is separated into subtasks shown in Figure 32.
3. Record geographic information of the photo. The process is shown in Figure 33.

Figure 33: Process of recording the geographic information
4. Review personal uploaded history. The uploaded history is separated into subtasks shown in Figure 34.

Figure 34: Process of reviewing uploaded histories
5. Upload the information to the cloud database. The process is shown in the Figure 35

Figure 35: Process of uploading the information
6. Download all information from the cloud database. The process is broken down which is shown in Figure 36.

8.8.3 Implementation-based Services

1. Automatic geographic location service.
   The library we used is the CoreLocation module in the iOS Framework. This service will ask the user’s permission for accessing the location service, and then the application will use the GPS hardware and geographic information (longitude and latitude). When the application sends a request to get the geographic information, this service will retrieve the longitude and latitude and send them back.

   The requirement is that the service will work normally under good GPS signal quality and the service is activated when the user uploads the image which is from the camera.

2. Manual geographic land marking on the map
   The library we used is the MapKit in the iOS Framework. The MapKit can supply a map for the user and a pin will show on the map. The
pin is draggable. The user can drag the pin and move the pin to the location where the user wants to put the pin. And the service will get the geographic location. This service requires an Internet connection.

3. Take photo by the camera
   The library we used is PhotoUI framework in the iOS Framework. The PhotoUI will ask permission to access the camera. And then PhotoUI can call the system hardware — camera. The user can control the camera and take the photo. This service needs to grant the user’s permission to use the camera.

4. Get camera usage permission
   Ask the permission for the first time when the application wants to access the camera.

5. Date picker
   Record the date of taking picture using the SwiftUI framework.

6. Survey
   This service will retrieve the Date, Knowledge level, Contact Information, and additional information for the image.

7. Photo Library Access
   The library we used is PhotoUI framework in the iOS Framework. PhotoUI can supply a portal to pick up a picture from the photo library. This service only can access the public albums.

8. Upload the image to the database
   This service uses the Firebase iOS Framework. The Firebase iOS Framework is designed by Google. This service will create a new thread to handle the upload task, this service will package the image and other information into binary data, and then upload to the cloud database. This service needs a good Internet connection.

9. Local storage
   This service will save the photo taken by the user to the photo album.

10. Check duplicate image uploaded
    This service will check if the user uploads duplicate images to the database to prevent taking up too much bandwidth.

11. Check if the submission task to database is successful.
    This service uses the Firebase iOS Framework. This service will monitor the feedback from the uploading task.
12. Check if the Firestore database is successfully connected and initialize the website.
   Using the private key tokens to connect the Firestore in the Firebase server. If it is not connected, a 404 page will show up.

13. Check whether all data are snapshotted.
   It will loop through the Firestore database and output each record in the collections. If it catches an error, it will be directed to the 404 page.

14. Check whether the back-end data is passed to the front-end.
   It will return 6 lists which contain the total records, choice records, contact information, date, and geographic information. There is a timeout for waiting for all the data to be snapshotted and obtained from the Firestore database.

15. Mapbox for the map initialization
   Initialize the map based on the Mapbox. It will display the map and layers for all the markers and icons with specific actions.

16. Firebase hosting website
   It is the online hosting setup for the back-end server for deploying the HTML files with JSON files. A back-end server is for hosting the website and deploying the whole project.

17. Download information from the database and then generate the CSV file.
   It’s a function which will download all the pictures in the database and organize the collected information and export as CSV file.

8.8.4 Workflows
   • Implement the iOS application
     – Take photo for the wildlife disease.
       * Get camera usage permission
       * Take photo by the camera
     – Pick up the photo of the wildlife disease from the photo library.
       * Get photo library access permission
       * Photo Library Access
     – Record geographic location of the photo.
       * Automatic geographic location service
       * Manual geographic land marking on the map
     – Record date of the photo.
       * Date picker
– Record users’ knowledge level with mange.
  * Survey
– Review personal uploading history.
  * Local storage
– Upload the information to the cloud database
  * Upload the image to the database
  * Check duplicate image uploaded
  * Check if the submission task to database is successful

• Build up the website
  – Get distribution of records on a map
    * Mapbox for the map initialization
    * Check whether the back-end data is passed to front-end
    * Check whether all the data is snapshotted
  – Retrieve all data from Google Firebase
    * Check if the Firestore database is successfully connected and initialize the website.
    * Download information from the database and then generate the CSV file.

8.9 Folder and File Description
In this section, we will briefly explain the purpose of each folder or file.

8.9.1 iOS Application Directory
This section will explain the iOS application repository. For more details, please see comments in each source code file, the Apple Developer Document [4], and the Firebase Developer Document [6].

• AppTrackWildlifeDiseases Folder
  – In the folder, it contains all the source code files of the application.

• Podfile
  – This file contains the dependencies of the project. Use CocoaPods to configure the programming environment.

• TrackDiseases.xcodeproj
  – This is the configuration file used by the Xcode IDE.

• TrackDiseases.xcworkspace
This file is generated by CocoaPods using the `$ pod update` command. The `.xcworkspace` file groups the projects and other documents including the Firebase SDK and other dependencies, so the developer can work with them together.

If using the Git tool to manage the project, please do not commit the `TrackDiseases.xcworkspace` file. Use CocoaPods to generate the `TrackDiseases.xcworkspace` file locally.

### 8.9.2 Web Application Directory

- **public Folder**
  - In the folder, it contains all the source code files of the website. The `index.html` is the main HTML file which contains all the website front-end code. The `main.js` file contains all the website back-end code. The `apptrackwildlifediseases-XXX.json` file contains all of the token, private key, and ID for the identification of the Firebase server.

- **firebase.json**
  - This file contains the dependencies of the project. The hosting directory, Firestore rules file, and the default HTML file location is defined in this file.

- **firestore.rules**
  - This is the configuration file used by the Firestore database. We enable the write and read permission for our code.

Since we directly zip the whole project files for the TrackWildlifeDiseasesUser project, and TrackWildlifeAnimalAdmin project, the later developer can directly download the zip file from the GitHub [12] and run the code in IDE or text editor. The zip files look like Figure 37.

![Figure 37: Zipped Web applications](TrackWildLifeAnimalAdmin.zip)

![Figure 37: Zipped Web applications](TrackWildlifeDiseasesUser.zip)
9 Lessons Learned

9.1 Timeline/Schedule

Figure 38: Timeline

Figure 38 shows the team’s timeline:

- 1/29
  - Assign the team works and set up the timeline.

- 2/6
  - Brainstorm the requirements from the client.
  - Starting to write the code for application.

- 2/8
  - The iOS application design review for the image capture.
  - Design review for function of making the questionnaire to record the information corresponding to the image captured.
  - Design review for the function of recording the geographic information of image

- 2/13
  - Finish the code of function on the application.
  - Code review for the functions and make improvements.
  - Design review for how to submit the images and geographic information to the database.
  - Design review for showing the information on the map in the application.

- 2/20
- Finish the first draft application and code review for it.

- **2/24**
  - The first draft application is finished and pushed to GitHub.
  - Design review for the web page.

- **2/27**
  - Start to write the first version of the website.

- **3/6**
  - Brainstorm the layout of the website.

- **3/13**
  - Create the map with data overlaid.

- **3/20**
  - The first version website is finished.
  - Start to analyze and fix application and website.

- **4/3**
  - Start to test the iOS application and website.

- **4/12**
  - Review the test results.
  - improve the application and website via test results.

- **4/25**
  - Project finished.

- **5/5**
  - Project delivered to client.
9.2 Problems and Solutions

In this project, we encountered several problems when designing and implementing the iOS application and website. The table of these is shown as Table 3.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>When we are using the API to use the camera or access the photo library, the new framework in the iOS 14 only supports to access the photo library, not support the camera.</td>
<td>The new Framework, PhotoKit, only supports to access to the photo library, so we decided to switch back to the old API; the old API supports the camera and photo library, which makes the code simple and readable.</td>
</tr>
<tr>
<td>When we design the function that lets a user select the location manually, at the beginning, we let users type in the address and convert to the geographic information. However, this way has a problem and is not friendly for the user to operate, and the accuracy is low after converting.</td>
<td>The solution is that we design an interactive map for a user to choose the location, which is easy to operate and the accuracy is high because the user selects the location directly from the map.</td>
</tr>
<tr>
<td>When we are trying to get the API data from the Firestore database, we use the callback function which is an async-await function. Therefore, the order of the JSON object is read from the Firestore in random order. Every time we try to use the data, we have to read the data from the Firestore. It costs a lot of time and we are not sure when the async function returns since it is an async-await function. Sometimes, it reads quickly the front-end will display the information. Sometimes, it will not display the information since the back-end is not finished reading the data, but the front-end finished running.</td>
<td>For increasing the speed and stability of the website operation, we use the setTimeout function for saving all the data local. Therefore, it can avoid the website front-end running faster than the back-end which makes the website not able to display any information.</td>
</tr>
</tbody>
</table>

Table 3: Problems and Solutions
9.3 Future Work

We don’t have a large testing group, although the test group has 20 people to test the project. Therefore, the user experience and feedback we have right now is not general. We decided to do the questionnaire survey and spread it through e-mail. It is a fast and convenient way to look for more people willing to help us test our app and website and complete our questionnaire.

Also, our testing rubric is not specific enough. We plan to make a more detailed testing rubric for the future tests. We only have the functionality test and the user experience survey. Maybe we will add a compatibility test, performance test, and so on. After collecting the ideas from more people, it is possible to add more functions to our app and website and therefore make our products more widespread.
10 Acknowledgements

We would like to firstly thank our client, Dr. Luis Escobar, for introducing us to this project and helping throughout the implementation of our application. We would like to also thank the tester of our application and website, Mariana Castaneda-Guzman. Finally, we would also like to acknowledge our professor, Dr. Fox, for helping answer any questions we had regarding this project. Their contact information is listed below.

- **Dr. Luis Escobar**
  Department of Fish and Wildlife Conservation
  escobar1@vt.edu
  https://fishwild.vt.edu/faculty/escobar.html

- **Mariana Castaneda-Guzman**
  Department of Computational Biology and Disease Modeling

- **Dr. Edward A. Fox**
  Office: 2160G Torgersen Hall
  Phone: (540) 231-5113
  Email: fox@vt.edu
  Address: Dept. of CS, 1160 Torgersen Hall, Mail Code 0106, Virginia Tech, Blacksburg, VA 24061
  Laboratory: Digital Library Research Laboratory - Torg 2030, x3615. DLRL facilities and support by students working there, are available to the class.
References


