THE UNIVERSITY OF MELBOURNE

Mobile App for Bird Song (Android)

COMP90019 Distributed Computing Project

Software Development Project

25 Credits Project

Name: Xiaoyu Zhang

Student Number: 712379

Supervisor: Prof. Richard Sinnott

I certify that
- this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.
- where necessary I have received clearance for this research from the University's Ethics Committee and have submitted all required data to the Department
- the thesis is 6516 words in length (excluding text in images, table, bibliographies and appendices).

Acknowledgements

I would like to express my greatest gratitude to Professor Richard Sinnott for offering me great supervision and support during the whole project.

I also want to appreciate Ms Yun Wang for creating the first version of server recognition program and my teammates for great assistant during the developing phrase.

Last but not least, I would like to thank my parents for their financial support and encourage.

Abstract

This client for bird species recognition targets for Android device. By utilizing the portability and compact hardware for smart phone, a mobile client will provide convenience for accessing the bird species recognition algorithm. This project formulate a client-server model with recognition algorithm on server and using mobile client to interact with users. This report will focus on the development of client on Android. We used AudioRecord class to capture bird songs with android device and utilize libs under AudioEffect class to achieve Double MIC Noise Reduction and enhance the audio quality. Furthermore, Google Map API has been imported to the project to realize visualization of bird distribution in Australia.

Keywords: Android, Noise reduction, Client-server, Google Map API

Table of Contents

Introduction 1

1. Objective and Platform 1

1.1 Objective 1

1.2 Android 1

2. Application Structure 2

2.1 Web Service 2

2.2 Android Client 2

3. Application Analysis 3

3.1 Feasibility Analysis 3

3.2 Use Case Analysis 4

3.3 Class Design Analysis 4

4. Functions and Implementation 6

4.1 Login and Register 6

4.2 Bird Song Collection and Upload 7

4.2.1 Audio Collection 8

4.2.2 Location Service 10

4.2.3 HTTP REQUEST 11

4.3 Result Display 12

4.4 Map Visualization 13

4.4.1 Google Map API 13

4.4.2 Map Drawing 13

4.5 User Information 14

4.5.1 User Management 14

4.5.2 Uploading History 14

5. Challenges 15

5.1 Noise Reduction 15

5.1.1 Research for Noise Reduction 15

5.1.2 Implementation 17

5.2 Network Issue 18

5.2.1 Audio Size Limitation 18

5.2.2 Information Localization 18

6. Error handling 18
6.1 Null Pointer Exception 19
6.2 Thread Collision 19
7. Future Works 19
7.1 Complete Expert System 19
7.2 Multi Bird Recognition 19
Conclusion 20

Introduction

Bird research is crucial for biodiversity, as it will estimate bird species, distribution and habitat with the changing of season and climate. Bird protection can be implemented according to these data. For bird researchers, recognition bird species can always be a great challenge. The traditional way to recognize bird species is mainly based on visual, however, this approach can always be hindered by the remote data capture (Brigges et al. 2012). This project will solve this problem by recognizing bird species according to audio. Compare with image data, collecting acoustic data from bird is more convenient for researchers.

This work mainly address the client development on Android platform. Android is a light weight operating system has been used wildly in the market for mobile devices. The software development can be divided into three stage: analysis, implementation and testing. In this report, we will firstly introduce the objective and the platform of the application. Then we will analysis the application in aspects of feasibility, use case and class. After that the detailed functions and implantation will be illustrated, which will be followed by the challenges encountered in the project and related solutions. Also, the error handling will be addressed to explain the error handling strategies for this application. Finally, a conclusion will be done to summarize the report.

1. Objective and Platform

Bird species recognition Android client solve the problem of the interaction between user and servers. In this section, we will explain the objective of Android client and briefly introduce the platform we develop the application.

1.1 Objective

This application aims at developing an Android client for bird species recognition. The recognition program has been deployed on server, which accept audio data as predicting basis. The objective of client is to implement audio collection with Android devices and achieve automatic interaction with server. Server replies are all JSON format files, which can hardly be understand by ordinary users, so the client should translate these pale texts into a user friendly format. Also, this application will let users fully participant into the application.

Moreover, this application aims at providing both ordinary user and researchers a convenient and acceptable way to study birds. Hence, visualization of bird distribution in Australia will also be implemented.

1.2 Android

Android as an open-source platform, which is designed especially for mobile device. Nowadays, Android has generated great revolution for the smartphone industry as Android devices have occupied more than half of current mobile phone market. Android is based on Linus operating system and developed by Google and the Open Handset Alliance. As a light- weight operating system, Android has become one of the most popular mobile operating system for smart phones (Developers 2011). Based on Java, Android has perfectly extends its APIs, which attracts plenty of programmers.

For this bird species recognition project, Android devices will be considered as a crucial component for the system. Having developed and progressed for so many years in both hardware and software for Android devices, utilizing Android gadgets to achieve audio collection, location requests and map visualization is reliable and brilliant. Also, with the prevalent using of mobile device between ordinary peoples, it will provide great convenience for researchers in conducting bird research.

2. Application Structure

Bird species recognition App is highly relay on the Internet to communicate with server where bird recognition algorithms and web service are settled. As Android client needs to achieve smooth interaction with server and to formulate a user friendly visualization for server replies. In this section, we will discuss the application structure on both server and Android client.

2.1 Web Service

The Web service is built on Nectar Cloud. As shown in figure 2.1, there are two servers on cloud. One is static server, which is mainly used for storing audios. User can reach audio data on this server with URLs and be able to pull these data to local mobile device. Another server is designed for handling requests form clients (including Android, IOS and Web). Client uses HTTP protocols to achieve message exchanging and file uploading with server. The web service, however, will return results in JSON format according to clients' request.

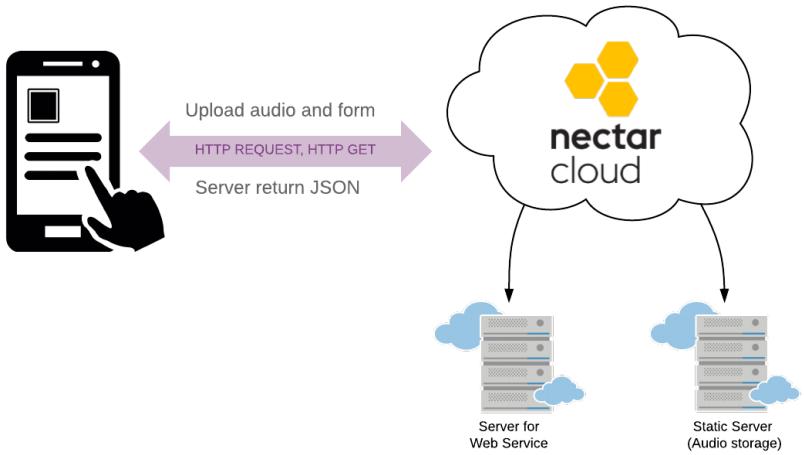

Figure 2.1 Web service and interaction with client

2.2 Android Client

Figure 2.2 demonstrates the application structure on Android. The App is launching on Main Activity, which contains three fragments: Recorder, Map and User Info. These three fragment can flip to another by user sliding the screen. Result Display activity is connect with Recorder Fragment, which focuses on visualizing server reply. Furthermore, to use all functions on this App, users are forced to login first. The Main Activity will check the identity of current user first when user launch the app. For unauthorized user, Main activity will guide user automatically to Login Activity and Register Activity.

The Android client has considered the user friendly UI design. We utilize both gesture and different clickable events to enhance user participants of the app. Also, all functions provided by the App can be triggered easily by users.

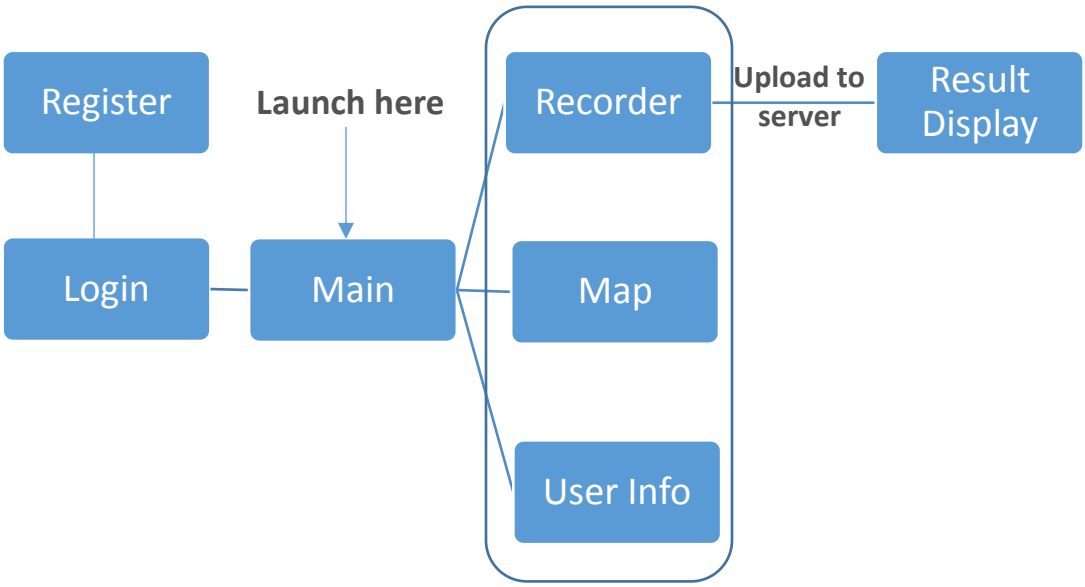

Figure 2.2 Android client structure

3. Application Analysis

This section will explain the analysis of the application, including feasible analysis, use case analysis and class analysis. Before we start programming for the application, an explicate analysis has been done to guarantee that the App developing is feasible and well guided.

3.1 Feasibility Analysis

Feasibility analysis explains whether the project is feasible. There are main two questions need to solve before exploration: how to achieve bird recognition and whether mobile device are capable of audio collection task.

Bird recognition will be done on server side, which uses supervised learning to train a model. The training processing is time consuming and needs better hardware support. Put these process on server side is feasible as the server has a better hardware than mobile device. Moreover, communication should be built between client and server. Android APIs provides an excellent HTTP requests mechanism (HttpUrlConnection), which can be used for both file uploading and form submitting (Shu, Du & Chen, 2009). Overall, the bird recognition on server is feasible.

Android clients need to collect bird songs and do a visualization of the bird distribution. Android powered mobile devices nowadays have already contain the recording module, so audio collection is feasible on Android client. Also, mobile devices has been wildly accept by citizens, which makes sure that to explore an Android clients can fully cater the requirements of users. Moreover, Android can implement visualization concisely and effectively with Google Map API. Therefore, it will be a brilliant choice to put client on Android device.

Overall, both HTTP connection and audio collection can be achieved by Android client, it is feasible to design an application on Android devices to cooperate with server prediction.

3.2 Use Case Analysis

Use case diagram describes the application on user's perspective (Warmer & Kleppe 1998). Figure 3.1 describes the interaction between user and cases. User will directly interact with three use cases: Audio Record, Map Display and User Information. The premise of users using them is they had login and registered. Audio Record provide user functions of collecting bird song with a recorder, which also contains locating information request, upload audio and result display. Map Display utilize Google Map API to visualize the bird distribution in Australia. User Information, however, will show the profile of the current login user. Also, the most crucial function of this use case is to show the user uploading history and provides audio replay service.

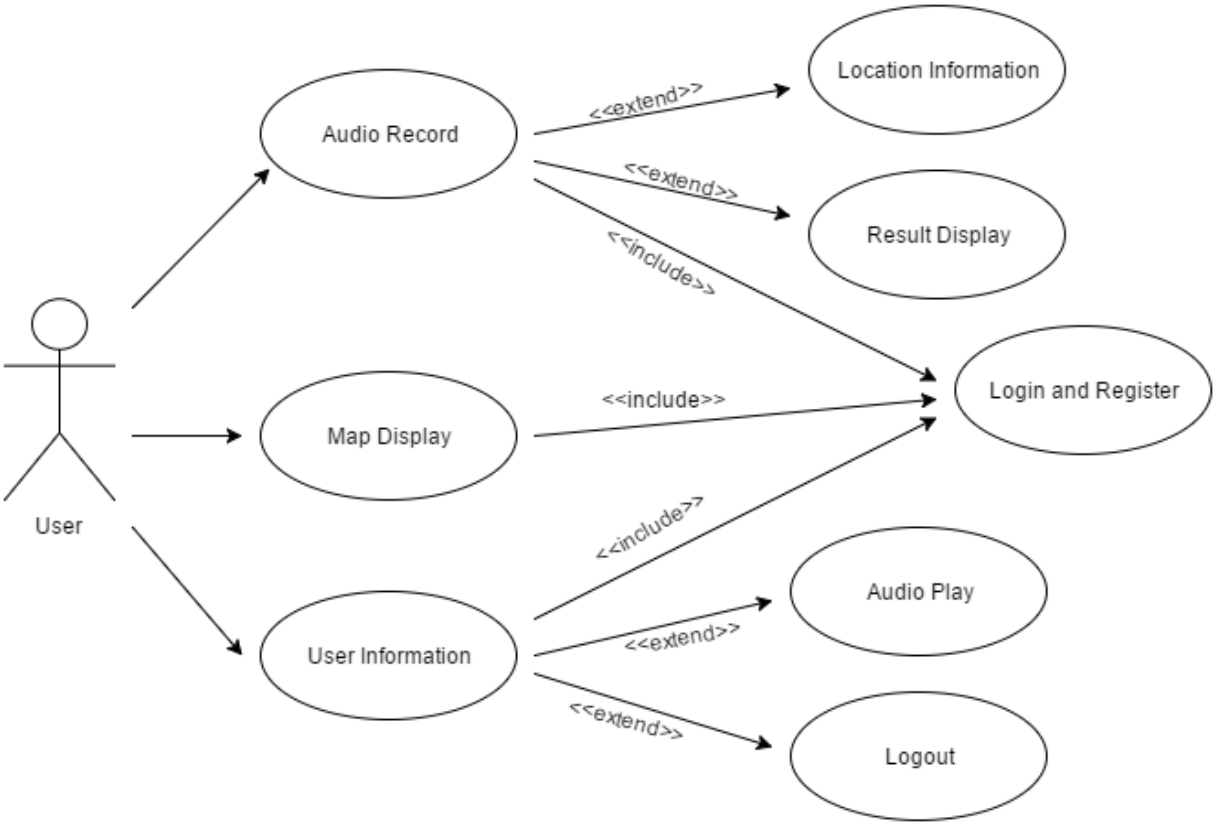

Figure 3.1 Use case diagram

3.3 Class Design Analysis

Class Diagram shows programming structure of the application (Warmer & Kleppe 1998). Figure 2.4 demonstrates all classes and their relationships. RecorderFragment, MapFragment and UserInfoFragment are the three classes that hold the main logics of this application.

RecorderFragement mainly handle audio and location information collection. To achieve these functions, it depends with GeoLoaction class, AudioProcess class, UploadUtil class and RecorderTimerView class. Also, it associate with DisplayResultActivity class which displays predicting results. MapFragment depends on UploadUtil class as it utilize UploadUtil class to realize HTTP GET. It also contains many marker information which are holds by MarkerInfo Class. UserInfo class is a relative single class as its

main task is to show user information and uploading history. For all users predicting histories, they are hold by BirdInfo class. For each user, they may keep many items.

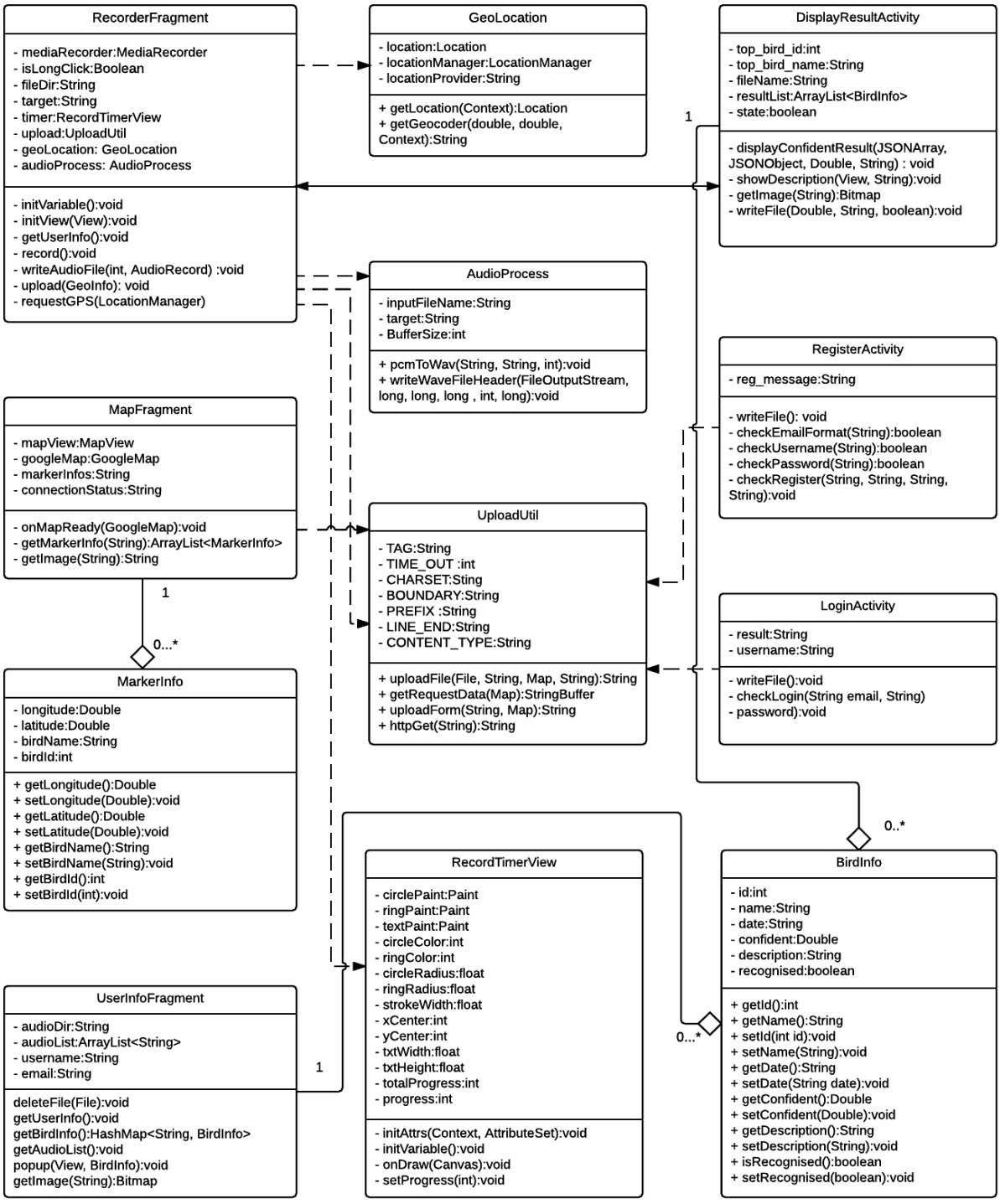

Figure 2.4 Class Diagram

4. Functions and Implementation

Bird Species recognition App was design with three tabs: recorder, map and user information. We display these page as fragment and use a ViewPager as a container to hold them. This design will generate a user interactive UI for pages flipped when user's finger sliding the screen.

ViewPager accompany with Fragment is recommended by Google to achieve pages flip, because ViewPager provide a convenient and flexible way to manage the life cycle of pages (Rogers et al. 2009). Android provide two interface FragmentPagerAdapter and FragmentStatePagerAdapter, which are two different fragment life cycle management strategies. According to Android API Guide (2016), FragmentStatePagerAdapter is design for large Fragment display as it will destroy all fragments when they lose focus and only stores the savedInstanceState of fragments. FragmentPagerAdapter, however, will load on all fragment at the initial stage. FragmentPagerAdapter may lead to memory exceed but for small fragment display, it will be more efficient. In this application, FragmentStatePagerAdapter is used because each fragment has to run complex algorithms, which includes frequent push information to server.

In this section, the detailed functions and implementation of this App will be explain in below.

4.1 Login and Register

Bird species recognition App has strict authentication control. Services provided by this App are only available for authorized user. Once user launch the application, it will automatically check the identity of user. For unauthorized users, the system will guide them to login and register page.

Figure 4.1 and Figure 4.2 demonstrate login page and register page respectively. As shown in these figures, user input should be checked carefully. The verification will be done in both client and server. Server side has a complete and strict verification system, which will check both input format and information collision with other users. It ensures the security of the system for no user can manage to skip the authentication process. The client also implements a simple version of verification, which only checks user input format. Verification solely on client cannot guarantee the security because experienced user can therefore try to dodge identity check process. Also, client do not has the ability to identify information collision. Verification did on client, aims at releasing the network burden as it will avoid useless information exchange with server.

Figure 4.1 Login page

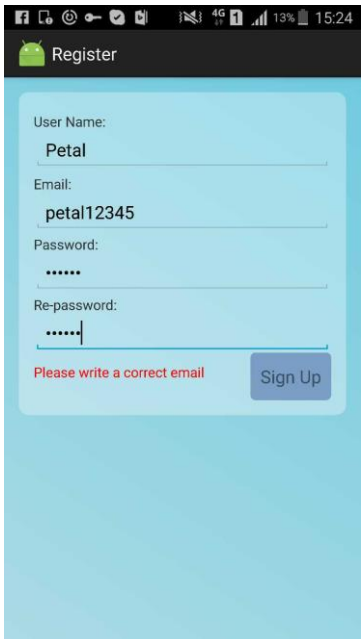

Figure 4.2 Register page

4.2 Bird Song Collection and Upload

Recording Fragment is the most crucial part of this application. In that fragment, user can invoke a recorder to gather bird tweets. Combine with location information, an HTTP REQUEST will be used to send audio to server.

Figure 4.3 and Figure 4.4 show recording process and uploading respectively. The cycle in the middle of the screen is a timer, which can not only interact with user for informing recording time, but also limit the longest recording time to 30 seconds. The time we set as 30s mainly consider the network issue, which will be discuss later in this report. The design of the recording module, we use a user participating way for the recorder will be triggered while user hold the button. Recording will be stop automatically once the user remove the finger from screen. The reason we do this design is hoping that user can fully participant in the application and can have a clear sense of time elapse. Uploading function will be triggered automatically whenever it detect the recording finished.

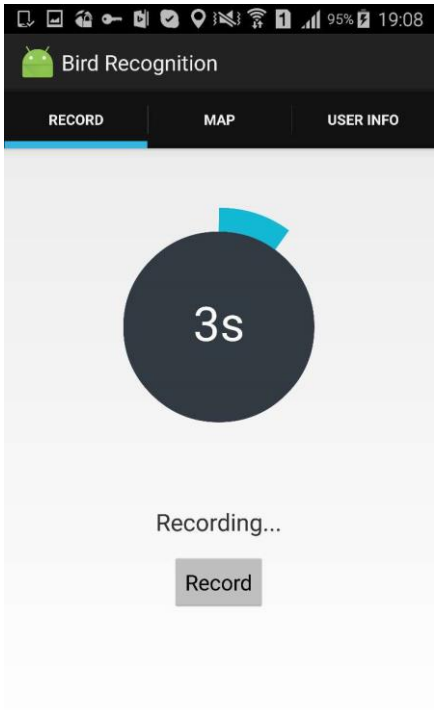

Figure 4.3 Recorder with a timer

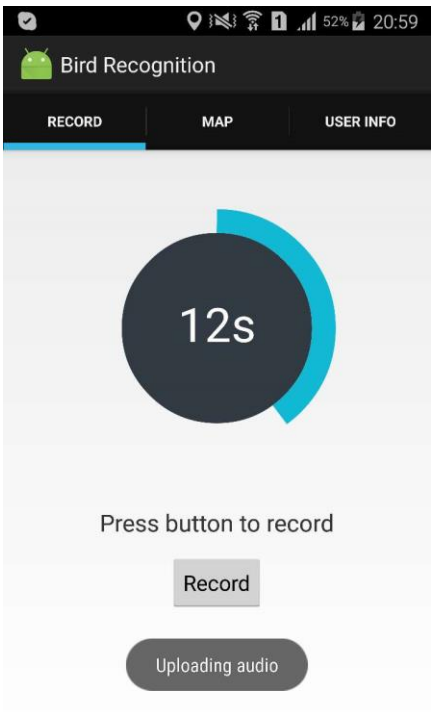

Figure 4.4 Audio uploading

4.2.1 Audio Collection

Android APIs provide two ways to implementing audio recording: MediaRecorder Class and AudioRecord Class (Meier 2012). The audio format request by server is WAV, but both of the class cannot generate WAV audio directly. In this part, we will first discuss different audio formats. Then, we will briefly introduce the MediaRecord Class and AudioRecord Class respectively. Finally, we will compare results produced by these two class and find the best solution for this project.

Output Audio Format

Table 4.1 illustrates three audio formats WAV, PCM and AMR. WAV is the target format that requested by server. PCM and AMR, however, is the audio format produced by AudioRecord and MediaRecorder respectively. According to the table, for the characters of compression, file size and quality, WAV and PCM is completely same. PCM, as a matter of fact, is a codding format for lossless WAV file, in other words, transform PCM into WAV can be implemented by simply adding a WAV header. PCM, as a coding method, cannot be played directly on mobile device. Compare with WAV and PCM, AMR is a compressed audio format and accordingly the file size is smaller than WAV and PCM. The audio quality of AMR is good but still inferior to those two.

	WAV	**PCM**	**AMR**
Compression	No	No	Yes
File size	Large	Large	Medium
Quality	High	High	Medium
Play on phone	Yes	No	Yes

Table 4.1 Comparison for different audio format

MediaRecorder Class

MediaRecorder class can be used for capturing both audio and video. Program with MediaRecorder is quite simple as it is a higher level class which encapsulates most of recording process. The format of audio it capture is ARM, which is a compressed audio with a medium file size (Chin et al. 2011). However, just because this audio format is compressed, it can hardly be modified, which will hinder the next step of noise cancellation. Moreover, MediaRecord as a well encapsulated class, it does not permit user to access the file output stream.

AudioRecord Class

AudioRecord Class is a base class in Android to handle audio processing and it provides a compact audio recording solution. AudioRecord can be divided into two parts: audio recorder and file output stream. File output process will be done simultaneously with audio recording. The file writing thread should listen the audio recording thread and whenever new data produced, the file writing thread will write it to file immediately. The output format of AudioRecord is PCM, which is an uncompressed audio format with all audio data recorded. As AudioRedord class allows developers to manipulate file writing process, modification such as noise suppression can be done while the data is writing to file.

Comparison and Implementation

Table 4.2 is a brief comparison between MediaRecorder Class and AudioRecord Class. MediaRecorder Class has the advantage that the output file can be played directly on the phone, however, by testing the audio quality, it can be found that even though we set a relatively high sample rate with 44100 Hz, the noise is easy to detect and has seriously affected the hearing of content. MediaRecorder Class cannot achieve active noise suppression which aims at canceling the noise while recording. Although we can still do noise reduction after recording finished, the efficiency and quality of noise reduction will be discounted. In comparison, the audio file produced by AudioRecord Class cannot be played on phone directly, but it has the advantage that this uncompressed audio contain very detailed information and suitable for signal processing. Furthermore, noise suppression process can be done simultaneously with recording proceed.

	MediaRecorder Class	**AudioRecord Class**
Program	Easy	Complex
Audio quality	Low	Adjustable
Output format	AMR	PCM
Noise suppression	No	Yes
Real-time modify	No	Yes

Table 4.2 Comparison for MediaRecorder and AudioRecord

Therefore, although implementing a recorder with AudioRecord Class is more complex to program, the result produce by AudioRecord Class is way better than MediaRecorder. In this application, we finally utilize AudioRecord to achieve a recorder and parameters for audio collecting are set as Table 4.3.

Parameter	**Value**
Sample rate	44100 Hz
Channel	STEREO
Encoding Format	PCM_16BIT

Table 4.3 Parameter setting for recorder

4.2.2 Location Service

Almost all smartphones nowadays contain the GPS module and network module, which guarantees location information can be acquired by mobile device easily. In this project, the location information is requested by server, which is a crucial part in improving predicting algorithm. Therefore, with users' permission, the application will automatically acquire users' location information.

For Android, Location Based Service (LBS) are all contained in android.location package. This package mainly has two libs, LocationManager and Geocoder. LocationManager mainly provide location service by utilizing GPS network or AGPS to acquire information such as current latitude, longitude and altitude. Geocoder main handle the translation between location name and coordinates (Kushwaha & Kushwaha 2011). The detailed explanations of them are in below.

Android GPS

Android GPS needs hardware support of the phone to communicate directly with satellite and most mobile devices nowadays have contained this module. The GPS does not need the support of Internet. The advantage of GPS its high location accuracy, but the drawback is also inevitable. First, this module is energy consuming. As the battery life is one of the biggest issue for mobile device, it is less likely to open GPS service all the time. Second, most of mobile owners choose to shuttle down GPS service by considering security issue. For every time we need the GPS, we should have to guide user to open this service manually. Without the service permitted, it is unlikely to get the information. Third, the setup time for GPS module is longer than expected, which leads to the problem that the first time location request will be extremely long and will interfere user experience. Fourth, GPS can be barely used inside building (Kumar, Qadeer & Gupta 2009).

Network

Android can utilize the base station to realize getting location information. There are mainly two different ways for getting location information via base stations. First, utilize the nearby three base stations around the phone to calculate the coordinate of mobile device. As the position of base stations are fixed, so these data is reliable, however, the calculating usually time consuming, but still in acceptable range. Second, using the nearest base station information directly. These information conation base station ID, location area code, mobile country code, mobile network code and signal intensity. By sending these information to Google service, we can get the current location (Kumar, Qadeer & Gupta 2009). This approach is the most efficiently way, however, it has the deviations from ten to hundreds miles. The intensity of signal is a significant parameter, which can directly decide the range of deviations.

Another approach is to utilize the WIFI. Nowadays, the WIFI has been wildly used, so the WIFI location service is still considerable for this project. The principle of WIFI is similar with the base station, as the WIFI will find the nearby WIFI Access Point and MAC Address. Check the location of the MAC Address, users can thereby getting the location information. This approach is highly relay on the Internet because the query of data bank should be done online.

AGPS (Assisted GPS)

AGPS is the combination of GSM and GPRS, which utilize Access Point to assist sending satellite information. This approach will cut down the delay by directly using GPS module getting satellite information. The idea of AGPS is utilizing GMS to get the location of current base station and using this location to do a search for available satellite in this area (hebedich 2014). Compare with GPS, AGPS needs the assistant of Internet, however, the initial time for AGPS is way shorter than GPS. AGPS can compensate the indoor weak satellite signal in some degree, but still cannot solve this problem perfectly.

Geocoder

Geocoder provides transmission between coordinate and location name. Geocoder contains two Geology coding approaches: Forward Geocoding and Backward Geocoding. The forward geocoding provide service of getting coordinates according to location name, while the backward coding is to decode location name with coordinates(hebedich 2014) .

Comparison and Implementation

Table 4.4 demonstrates the Comparison between GPS, Network and AGPS. According to the table, GPS is the only one that do not relay on the Internet and has high location accuracy. However, it has the drawback that the initial time is extremely long and has bad performance inside buildings. For network approach, the initial time and indoor performance are all good but the accuracy is quite low. AGPS seems the best one in these three, because it is the combination of these two, however, it has not been used wildly in all mobile devices. Therefore, we cannot choose it as the location provider. In this project, considering the Internet nowadays are accepted wildly by users and easy to access, we set the Network for the first choice location provider. Although the accuracy is lower than GPS, this project can actually accept this deviation and compare with long initial time sacrificing the accuracy is deserved. However, under the condition that no network accessible by users, the GPS is set the backup plan for getting location information.

	GPS	**Network**	**AGPS**
Internet	No	Yes	Yes
Accuracy	High	Low	High
Initial time	Long	Short	Short
Indoor Performance	Bad	Good	Average
Usability	Wild	Wild	Limited

Table 4.4 Comparison of three location services

4.2.3 HTTP REQUEST

While the audio recording finished, the app will upload the audio with other parameters including location information to server. We need server to reply the computation result of possible birds, so the HTTP REQUEST should be choose to do the uploading. By decoding the response, we can get the content of server reply.

Figure 4.5 is a piece of code which shows the parameters we need to submit to server accompany with the audio. We provide server the location information of longitude, latitude, altitude and the decoded location name. The email, which is the unique identification of users on server database, will also be send to server. Also, in continence of web service to handle request from different device, the client which represent the platform (Android, IOS or Web) will also be addressed.

```
//set parameters of uploading
Map<String, String> params = new HashMap<~>();
params.put("longitude", longitude);
params.put("latitude",latitude);
params.put("evaluation",altitude);
params.put("submitted_by",email);
params.put("date",formater1.format(date));
params.put("time",formater2.format(date));
params.put("client","android");
params.put("user_comment","This is recored in School in the spring.");
params.put("location",address);
```

Figure 4.5 Parameters accompany with audio uploading

4.3 Result Display

After user uploading audio, the server will reply a JSON which contains a rank list of possible bird species. For the top one bird, a percentage of confident will also be provided. Figure 4.8 is a sample JSON format (partial) which contains bird ranking list, confidence, picture URL and a short Wiki description.

Figure 4.6 shows the result display page. For each possible bird, we give an image which provides another way for user to check the result. To guarantee user could fully understand the bird, a popup window with short Wiki description will appear when user clicks the bird image. For other possible birds, we put them on a ListView for user to reference. The percentage of confident tells user how much trustable the result is. Theoretically, all result we display for user is quite confident in correction. We set a threshold of 50.5% and result with confidence lower than that will not be displayed for users. Instead, find expert page will appear, which will guide user to request for experts' help (Figure 4.9). Every piece of predict result, we will extract most valuable information such as top bird name, confidence and Wiki description and write them to a file at the local of the phone. This will be used for further step of showing user uploading history in UserInfo page.

Figure 4.6 Predict result display

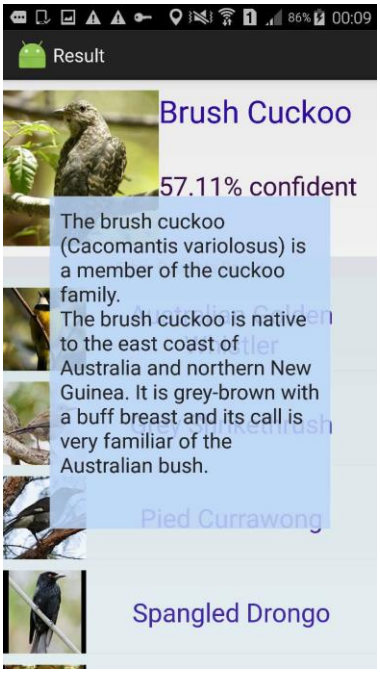

Figure 4.7 Wiki description of top bird

```
{
  "bird_code_dictionary": {
    "0": "Brush Cuckoo",
    "1": "Australian Golden Whistler",
    "2": "Eastern Whipbird",
    "3": "Grey Shrikethrush",
    "4": "Pied Currawong",
    "5": "Southern Boobook",
    "6": "Spangled Drongo",
    "7": "Willie Wagtail"
  },
  "confidence": 0.5089469404758693,
  "estimation_rank": [
    [
      1,
      -35884.33444337663
    ],
    [
      4,
      -37191.95282820944
    ],
    [
      6,
      -38311.954040815006
    ],
    [
      0,
      -42691.287023753954
    ],
    [
      3,
      -46313.69623441898
    ],
    [
      5,
      -53630.29167457494
    ]
  ],
  "status": "uploaded"
}
```


Figure 4.8 Sample JSON

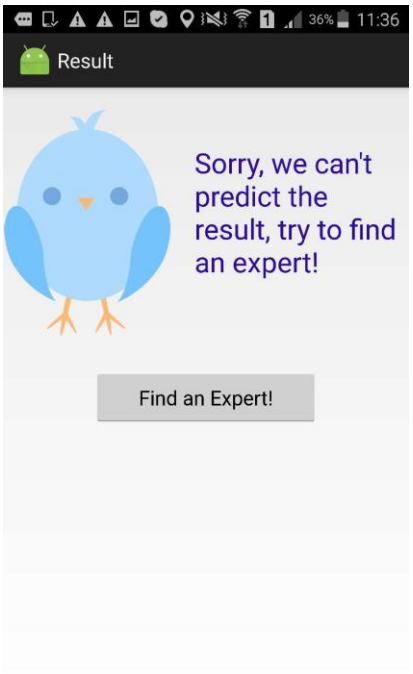

Figure 4.9 Find an Expert

4.4 Map Visualization

In this page, we realize visualization of bird distribution in Australia with Google Map API. The objective of this page is to provide researchers an intuitive way to observe bird species and their distributions.

4.4.1 Google Map API

Google Map API for Android is design for developers to realize map visualization easily with Google Map. To explore the mystery of this API, developer should get familiar with Google Map in both user and developer aspect. Google Map API will handle the user request on server, data downloading, map visualization and user gesture automatically, which provides developer great convenience on programming. Moreover, the API provide functions for user to add markers, shapes and layers to the basic map. These extra information will be helpful in realizing user interaction with map.

4.4.2 Map Drawing

When we launch the MAP fragment, a HTTP GET will be generate by client automatically to pull all confident predicted uploading results from server. Figure 4.10 is a sample JSON format which contains documents of record information. Useful information can be extracted from each document and represents as a marker on the map.

To make the different species distribution more intuitive, pictures of birds will be set as the logo of markers. Also, the bird name will be set as the title of the marker, which will appear whenever user clicks the marker.

```
"record_list": [
  {
    "_id": "80f611424174af5baa79e0160406e123",
    "_rev": "2-1ed1a4dda324b44bca3972e89f740d78",
    "client": "browser",
    "confidence": "1.0",
    "date": "2007-07-01",
    "estimation_rank": [],
    "evaluation": "",
    "expert_comment": "This audio file is downloaded from Xeno-Canto.com!",
    "expert_request_status": "not",
    "file_path": "/home/ubuntu/html/audio/training/Spangled Drongo_19471.mp3",
    "latitude": "-9.4667",
    "location": "Varirata National Park, National Capital District",
    "longitude": "147.3834",
    "md5": "1856a2545b8be1e68e59126acdacb001",
    "submitted_by": "rocklee.au@foxmail.com",
    "time": "15:24",
    "top_estimation_bird": "Spangled Drongo",
    "top_estimation_code": 6,
    "training_data": "true",
    "type": "record",
    "url": "http://115.146.90.254/audio/training/Spangled Drongo_19471.mp3",
    "user_comment": "This audio file is downloaded from Xeno-Canto.com!"
  },
  {
    "_id": "80f611424174af5baa79e01604039cce",
    "_rev": "2-edd86538af6e5c6522fea8e8fdaade2e",
```


Figure 4.10 JSON for markers

Figure 4.11 Map visualization

4.5 User Information

User Information page displays user's personal information and audio uploading history. Figure 4.12 shows the layout of that page, in which current user can check the uploading history by replaying the audio and reviewing the predict result.

4.5.1 User Management

Once user flips to USER INFO page, current login user's information will display. The displayed user information include user name, email and user type. Also, it provides user management functions as user can choose to logout at this page.

Because this App requests user to upload location information which is quite private for users, it is extremely important to protect users' privacy. This App does not keep any user's personal information or uploading history in local device and do the utmost to avoid these information exchange via the Internet. Therefore, all user uploading histories and file writing histories will be store in local device when user stay login. All these information will be clear when user logout. To avoid frequent information exchange with server, user history will never be download from server when user login.

4.5.2 Uploading History

According to the explanation in 4.3, all user uploading history has been store in a file in local device. By scanning the file with a FileInputStream, we get information such as file name, recognition state and prediction result and put items on a ListView. According to figure 4.12, the recognized item will be marked with a blue background while the items need experts assistants are marked in green. All the audio can be replayed whenever user click the item. For the recognized item, as shown in figure 4.13, a brief review of

the predict history will be displayed on a popup window which will be triggered whenever user click the button of "Recognized".

Figure 4.12 User information page

Figure 4.13 Recognition history review

5. Challenges

The design and implementation of this bird species recognition App faces many challenges. In this section, we will focus on challenges of noise reduction and network issue and related solutions. These two challenges are the most crucial problem in this project and relates closely with the predict accuracy and user experience.

5.1 Noise Reduction

The quality of audio will directly affect the quality of feather extraction and therefore affect the prediction result. Noise Reduction should be done both in client and server. At the server side, noise reduction usually accompany with feature extraction. MFCC (Mel Frequency Cepstrum Coefficient), which considering human hearing, transform spectrum into nonlinear spectrum based on Mel and then represent it in cepstrum. MFCC has both voice recognition and noise suppression functions (Sheikhzadeh & Deng 1994). However, even though MFCC has impressive performance in filtering out noise, too much noise can still seriously affect it because noise spectrum may overlap with the useful one (Vilches et al. 2006). Therefore, a preliminary noise reduction on mobile client is significant.

5.1.1 Research for Noise Reduction

Because noise reduction on client is auxiliary, it should obey two principles: first, client should filter out background noise as much as possible, but should not affect the efficiency of the whole program. Second, overly noise reduction on client should be avoid, because the effect of missing essential spectrums is way severer than the effect of noise on the predicting process.

In terms of the source of noise, there are mainly two kind of noise: device noise and environmental noise (De Moes & Valipor 2012). Device noise is the sound of hardware running of the phone. This kind of noise can be found clearly while recording without any other background sound. Figure 5.1 displays the pure device noise captured by recorder which represents in time domain. According to the picture, the sound wave is stable and no apparent change. Device noise is easy to collect and the pattern is clear, the decline of this kind of noise is quite easy (Pathak 2011). The environmental noise is unpredictable, which include sound of vehicle, wind and other animal's voice. Figure 5.2 demonstrate the recording that contain the environmental noise. According to the picture, it is hardly to detect any pattern of the noise. That is because the environmental noise will be vary at different time and place. It is not as steady as the device noise. By reference the noise reduction mechanism introduced in Noise reduction techniques and restoration effects for Audition (2015), the best way to decline the environmental noise is to capture noise print before recording. However, considering this approach will force user to collecting noise, which will disturb user experience, the noise cancellation should be done at the same time with recording.

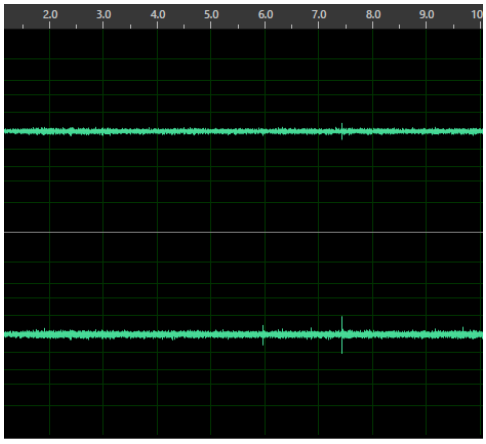

Figure 5.1 Device noise

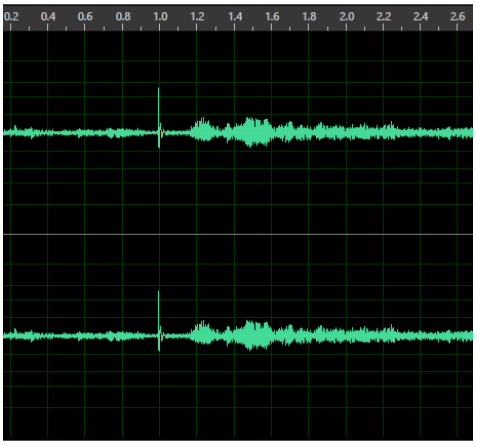

Figure 5.2 Environmental noise

The active noise cancellation is a solution for this problem. Active noise cancellation is a real-time noise cancellation approach. Mobile phone can actually achieve collecting noise and voice at the same time with double MIC, which guarantees the implementation of active noise cancellation (De Moes & Valipor 2012). For most mobile devices in the market, they usually have two microphones: one is at top of the phone and mainly collecting environmental noise; the other one is usually stay at the bottom of phone, which mainly collecting voice. As the microphone at the bottom of phone is closer to the mouse, it will collect more voice than the upper one. Usually, the two mic will have 6dB volume difference (admin 2015). We can utilize the different audio collected by the two mic to produce the anti-noise acoustic wave that to cancel the noise.

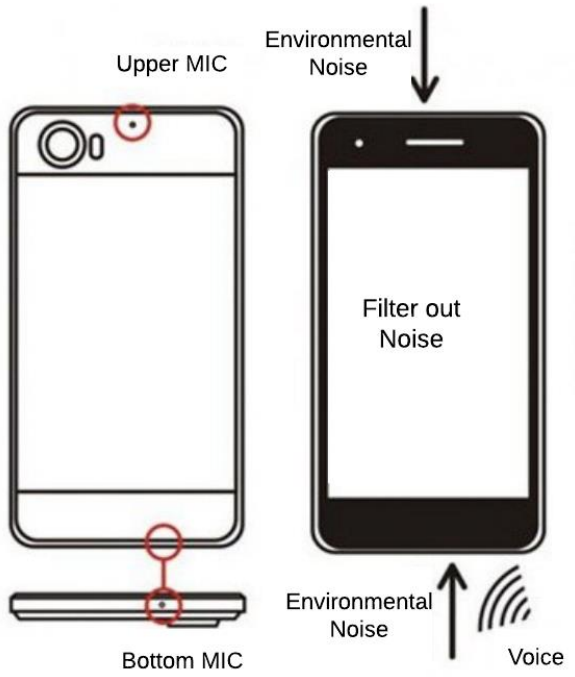

Figure 5.3 Double MIC noise reduction

Implementation of double microphone noise reduction is a great challenge. Limited by the size of the mobile phone, the distance between two microphones may not be so large, which may cause the problem that the useful part of sound may be filtered out. Therefore, an explicate algorithm should be design to achieve a high quality noise reduction.

5.1.2 Implementation

Mobile device, as a communication tool, has excellent noise reduction mechanism to ensure the quality of telecommunication. Android provide developers AudioEffect Class to improve the audio quality according to user's preference. Noise Suppressor (NS) under that class just utilize the active noise reduction to achieve a real-time noise reduction. Also, the EnvironmentalReverb, which is another user lib under AudioEffect class, can also be used to improve the audio quality.

NoiseSuppressor (NS). The component of the signal considered as noise can be either stationary (car/airplane engine, AC system) or non-stationary (other peoples' conversations, car horn) for more advanced implementations. NS is mostly used by voice communication applications (voice chat, video conferencing, SIP calls). NoiseSuppressor should be attached to a particular AudioRecord which is specified by session ID.

EnvironmentalReverb. Another useful lib under AudioEffect Class is EnvironmentalReverb. Sound travels in many directions. Sound collected by a recorder usually contains the sound directly from the source itself and the discrete echoes generated by reflections of walls or other objectives. Because the sound wave reflections arrive continuously, the individual wave can hardly be distinguished, which will cause the reverberation of audio. Reverb is caused by recording environment and effected by environment. In this project, the reverb removal is not essential but necessary. Although the wild bird tweets can barely produce

reverberation, the reverb removal can finally reach a uniform sound effect for birds live in different environment.

5.2 Network Issue

Network issue accompany with the whole developing process of this App. We set the server on Nectar, which is one of the most popular cloud platform in Australia. However, the connection with server is not that stable and the bandwidth on server is quite limited. Although the App is highly relay on server, the real implementation should cut down the frequency of communication between server and client. Also, the client should avoid large file transmitting with server. For this problem we mainly solve it in two aspects: audio size limitation and information localization.

5.2.1 Audio Size Limitation

There is a trade-off between audio quality and audio size. According to the explaination in 4.2.1, uncompressed audio format WAV carries the highest audio quality with no data loss at all. This format of audio is also perfect for signal processing and is convenient to implement both noise reduction algorithm and feature extraction. Compare with other format of audio, the only drawback of WAV is the file size is too large that 10 seconds audio may produce the file size of 1.5Mb.

By doing research of the feature extraction algorithm MFCC and model training algorithm Gaussian Mixture Model, we find that compare with audio length, the audio quality is more important in effecting the accuracy of predicting result. Therefore, we set a timer for recorder to limit recording time up to 30 seconds, which will ensure the audio size cannot exceed 5Mb and the waiting time will be acceptable for users.

5.2.2 Information Localization

The frequent pulling resource from server, especially image and audio, will severely interfere user experience of the App because the downloading speed is actually slow. We localize some information, such as bird images and recorded audios.

According to the explanation in section 4, bird images are used very frequently. It is impossible for user to download these image every time in need. In order to improve user experience of the application, we made the image for most frequent appeared birds in local of the App, so whenever these birds appear in prediction result, we can get the image immediately from local. For the image does not copied in local, we use Picasso, which is a convenient image downloading tool for Android, to download images in threads. Even though users will have to waiting for image downloading, it will not interfere the usage of other functions in App.

For the user uploading history, we need to replay the audio and provide user a review of recognized result. Audio size is quite large and re-uploading file will heavily increase the burden of server. Therefore, for each predict result we write it to a file on local device and we keep all uploading audio in local before user logout. With these strategies, all data for user uploading review is in local, which can be accessed very efficiently.

6. Error handling

During the programming process, we meets plenty of errors, which may cause the crush or disorder of the program. Usually these errors are hardly to detect and identified during programming but will appear while using the application. Error handling mechanism has been addressed in some place of the program to avoid application disorder.

6.1 Null Pointer Exception

Null pointer exception is the most frequent exception we meet during programming. This error main caused by the null value of variables. Under many conditions, we can hardly aware this existence of null value. For example, the null value in latitude and longitude may cause the crush of the whole program, because the Google Map functions cannot resolve a null.

We address this problem in three steps: first, go over the complete logics of the program and identify all possible place that null may occur. Second, add judgement clause to skip further steps for null parameters. Third, running the program frequently in different exploration stage. Test all the possible conditions.

6.2 Thread Collision

In developing this application, threads are frequently used to release the burden of main thread. However, threads are also sources of error because the collision of thread can be hardly detected. Also, the listener of the threads may also hinder the progress of the program and we should avoid add listener on main thread.

To address this problem, we add a listener thread separately, which will listen this state of each thread and coordinate thread to avoid collision. Also, for collision functions and variables, we set synchronized visiting limitations.

7. Future Works

The Bird species recognition application has finished at preliminary stage. Main functions, such as bird song collection, result display and map visualization has been implemented. However, some further improvements can be done to enhance the performance of the application.

7.1 Complete Expert System

Expert system is proposed at the program design stage. It will let bird experts to manually identify the bird species. Expert assistant mainly solve the problem that bird species that do not include in the training set and cannot be recognized by the model. Moreover, expert system is essential for the future improvement of the model. As the audio marked by expert is one hundred percent confident, these data can be added to training set directly.

This application has retain the interface for expert system. To cooperate with the server, we plan to add a background service for the application to listen the data change on server. Once the uncertain item is recognized by experts, client can detect it immediately and send a notification to users.

7.2 Multi Bird Recognition

In current stage, the bird recognition system can only achieve single bird recognition. However, in real using of the app, we find that bird always appear at the same time, which means that an audio may contain songs that comes from more than one bird.

The primary problem need to solve for multi bird recognition is to distinguish the spectrum that comes from different birds. According to Briggs et al. (2012), they address this problem by using multi-instance multi label (MIML) framework. MIML treats objects to represent as a “bag-of-instance”, which associated with multiple labels. To achieve this, training data with the list of species appear in recording is required. By distinguishing spectrums that belong to different bird species, it can be set as input to our current system to realize recognition.

Conclusion

In conclusion, the bird species recognition application on Android client provide convenience for user to interact with server. Users could collect bird song easily with the application and it will automatically communicate with server to get the predict result and display to users. Moreover, this application provide a visualization of bird distribution in Australia. To cooperate well with server and enhance user experience, the noise reduction and network issue are addressed on client. The application still need to improve in the future and will has better performance after complete.

Reference list

admin, 2015, Double MIC noise reduction on mobile, viewed 25th October 2016, <http://www.pc841.com/shoujizhishi/46640.html>

Briggs, F., Lakshminarayanan, B., Neal, L., Fern, X.Z., Raich, R., Hadley, S.J., Hadley, A.S. and Betts, M.G., 2012. Acoustic classification of multiple simultaneous bird species: A multi-instance multi-label approach. *The Journal of the Acoustical Society of America*, *131*(6), pp.4640-4650.

Chin, E., Felt, A.P., Greenwood, K. and Wagner, D., 2011, June. Analyzing inter-application communication in Android. In *Proceedings of the 9th international conference on Mobile systems, applications, and services* (pp. 239-252). ACM.

De Moes, A.F.A. and Valipor, S., 2012. Noise reduction application for Android smartphones.

Developers, A., 2011. What is android.

hebedich, 2014, Four approaches for getting location service for Android, viewed 23th October 2016, <http://www.jb51.net/article/52676.htm>

Google and Open Handset Alliance n.d. 2016, Android API Guide. <http://developer.android.com/guide/index.html>. Viewed 23th October, 2016.

Kumar, S., Qadeer, M.A. and Gupta, A., 2009, January. Location based services using android. In *Proceedings of the 3rd IEEE international conference on Internet multimedia services architecture and applications, ser. IMSAA* (Vol. 9, pp. 335-339).

Kushwaha, A. and Kushwaha, V., 2011. Location based services using android mobile operating system. In *International Journal of Advances in Engineering & Technology IJAET*.

Meier, R., 2012. *Professional Android 4 application development*. John Wiley & Sons.

Noise reduction techniques and restoration effects for Audition. 2015, <https://helpx.adobe.com/audition/using/noise-reduction-restoration-effects.html>. Viewed 23th October, 2016

Pathak, K., 2011. Efficient audio processing in android 2.3. *Journal of Global Research in Computer Science*, *2*(7), pp.79-82.

Rogers, R., Lombardo, J., Mednieks, Z. and Meike, B., 2009. *Android application development: Programming with the Google SDK*. O'Reilly Media, Inc..

Sheikhzadeh, H. and Deng, L., 1994. Waveform-based speech recognition using hidden filter models: Parameter selection and sensitivity to power normalization. *IEEE Transactions on Speech and Audio Processing*, *2*(1), pp.80-89.

Shu, X., Du, Z. and Chen, R., 2009, September. Research on mobile location service design based on Android. In *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*(pp. 1-4). IEEE.

Vilches, E., Escobar, I.A., Vallejo, E.E. and Taylor, C.E., 2006, August. Data mining applied to acoustic bird species recognition. In *18th International Conference on Pattern Recognition (ICPR'06)* (Vol. 3, pp. 400-403). IEEE.

Warmer, J.B. and Kleppe, A.G., 1998. The Object Constraint Language: Precise Modeling With Uml (Addison-Wesley Object Technology Series).

Appendix

Source code on GitHub: https://github.com/PetalZh/BirdRecognition