A Mobile Application to help the Elderly to avoid Telephone Deception
Abstract

This report researches the mechanism to screen out scam calls for the elderly by the mobile application. In Hong Kong, telephone deception is discovered to be the most common scam for elderly. Although there are existing applications in dealing with the problem, great amount of financial losses has been caused throughout these years. Therefore, the project provides a solution to avoid the elderly from telephone scams. An elderly-friendly android application with the features of incoming call screening with the help of the data from JunkCall HK, different functions to arouse awareness and keep track on the call history and information will be developed to screen out suspicious calls. The project is now coming to the end, synchronization of the database with the data from JunkCall HK is also done. An Android application with different functionalities has been developed, along with the potential challenges that will be demonstrated in the report.
Acknowledgement

I would like to express my sincere gratitude to Dr. T.W. Chim from the Department of Computer Science for supervising me throughout the final year project.

I would also like to acknowledge JunkCall HK and HKJunkCall for allowing access to a great amount of phone call data.

Lastly, I would like to thank Mr. Cezar Cazan, who has given me advices and guidances on the writing the Final Report.
Table of Content

Abstract I
Acknowledgements II
List of Figures V
List of Tables V
List of Programmes V
Abbreviations VIII

1. Introduction 1
  1.1 Background 1
    1.1.1 Current Situation 1
    1.1.2 Review on Existing Approaches 1
    1.1.3 Project Overview 2
  1.2 Motivation 3
    1.2.1 Safeguard the Interests of the Elderly 3
    1.2.2 Offer Technical Supports for the Elderly to Adapt the E-generation 3
    1.2.3 Work as Deterrent to the Malpractice 3
  1.3 Objectives 4
  1.4 Report Outline 4

2. Methodology 5
  2.1 IDE and Language 5
  2.2 UI/UX design 5
  2.3 Database of Suspicious Calls 7
  2.4 System Design 10
  2.5 Incoming Call Screening 12
  2.6 Phone Call Reporting System 16
2.7 Call Searching 21
2.8 Call Log 26
2.9 Regular Reminder to Avoid Telephone Deception 28
2.10 Simple Questionnaire and Reminder after the Call 31
2.11 Summary 33

3. Progress and Discussion 34
3.1 Search Screen 34
3.2 History Screen 35
3.3 Detail Screen 36
3.4 Report Screen 36
3.5 Incoming Call Screening 38
3.6 Simple Questionnaire and Reminder after Call 39
3.7 Regular Reminder 41
3.8 Call Data Scrapping and Database Synchronisation 42
3.9 Potential Challenges and Mitigations 43
  3.9.1 Unable to Scrap Data from HKJunkCall 43
  3.9.2 Long Running Time in Incoming Call Screening 45
  3.9.1 Restriction on Implementing Speech Recognition on Phone Conversation 45
3.10 Summary 47

4. Future Plan 48

5. Conclusion 51

References 52
List of Figures

Figure 1.1 – Icon of Sieve 3
Figure 2.1 – Example of UI with large font size 6
Figure 2.2 – Example of different contrast ratio 6
Figure 2.3 – Current UI of Sieve 6
Figure 2.4 – The entity-relationship diagram of database 7
Figure 2.5 – Information extracted by web scraping 9
Figure 2.6 – Major workflow of the application 11
Figure 2.7 – Alert with call information when there is incoming call 16
Figure 3.1 – Call searching function in Search screen 34
Figure 3.2 – Call log in History screen 35
Figure 3.3 – Redirecting to Report screen when the report button is clicked 35
Figure 3.4 – Detail screen of call without remarks 36
Figure 3.5 – Detail screen of call with remarks 36
Figure 3.6 – Call reporting function in Report screen 37
Figure 3.7 – Additional field for “Suspected scam” 38
Figure 3.8 – Warning for missing field 38
Figure 3.9 – Alert to display the information of the incoming call 39
Figure 3.10 – Simple questionnaire to ask if the call contains suspicious information 40
Figure 3.11 – Redirecting to the Report screen with the corresponding phone number 40
Figure 3.12 – A reminder to remind user with information of the call 41
Figure 3.13 – Daily reminder to educate user in preventing telephone deception 42
Figure 3.14 – Flow chart of screening if there aren’t any records in the database 45
Figure 3.15 – The original system design of the application 46

List of Tables

Table 4.1 – Project schedule 48

List of Programmes

Programme 2.1 – Python GET API for data parsing and scrapping on JunkCall HK 8
Programme 2.2 – The configuration of Firebase in config/firebase.js 10
Programme 2.3(a) – Module import for setting up Firebase 10
Programme 2.3(b) – Functions to connect and retrieve the data from database
Programme 2.4(a) – Adding permission in AndroidManifest.xml
Programme 2.4(b) – Module import to use CallLogs and CallDetectorManager in App.js
Programme 2.4(c) – Permission granting for READ_CALL_LOG in App.js
Programme 2.5 – Function to detect the incoming call with CallDetection in App.js
Programme 2.6 – Function to display the customised notification for incoming call in App.js
Programme 2.7 – Function to get the information of the call from the database in App.js
Programme 2.8(a) – Module import to change the ring mode in App.js
Programme 2.8(b) – Function to change the ring mode in App.js
Programme 2.9(a) – Adding permission to read call log in AndroidManifest.xml
Programme 2.9(b) – Adding service ConnectionService for CallKeep in AndroidManifest.xml
Programme 2.9(c) – Module import to use CallKeep in App.js
Programme 2.9(d) – Set up for RNCallKeep in App.js
Programme 2.9(e) – Display the customised notification for incoming call in App.js
Programme 2.10(a) – Packing import to use DropDownListPicker in Report.js
Programme 2.10(b) – Module import to use RadioButton in Report.js
Programme 2.10(c) – Module import to use TextInput and TouchableOpacity in Report.js
Programme 2.10(d) – Form of phone call reporting in Report.js
Programme 2.11 – Function to handle the form submission in Report.js
Programme 2.12(a) – Dialog confirmation of form submission in Report.js
Programme 2.12(b) – Dialog warning if the input is invalid in Report.js
Programme 2.13 – <InputField> component in components/InputFields.js
Programme 2.14(a) – Module import for InputField in Welcome.js
Programme 2.14(b) – Search bar in Welcome.js
Programme 2.15 – Function to handle the search in Welcome.js
Programme 2.16 – Parameter numberList for the matched numbers in Welcome.js
Programme 2.17(a) – Route prop in Detail component in Detail.js
Programme 2.17(b) – Set the value of phone be the number pass from Welcome component in Detail.js

Programme 2.17(c) – The detail of call to be displayed in Detail.js

Programme 2.18 – Welcome component with navigation prop in Welcome.js

Programme 2.19(a) – Adding Permission to read call log in AndroidManifest.xml

Programme 2.19(b) – Module import for call log reading in History.js

Programme 2.19(c) – Permission granting when accessing the History component at first render in History.js

Programme 2.20(a) – Parameter to store the log in History.js

Programme 2.20(b) – ScrollView to display the log callLogList in History.js

Programme 2.21(a) – Module import for PushNotification in App.js

Programme 2.21(b) – Adding permissions for PushNotification in AndroidManifest.xml

Programme 2.21(c) – Adding receiver and service for PushNotification in AndroidManifest.xml

Programme 2.21(d) – The configuration of PushNotification in index.js

Programme 2.22 – Function to handle the regular notification for arousing user’s awareness on telephone deception in App.js

Programme 2.23 – Functions to end a call in App.js

Programme 2.24 – Simple questionnaire by PushNotification after the unknown call in App.js

Programme 2.25 – Action declared to navigate to Report screen at PushNotification.configure in index.js

Programme 2.26 – Dialog box to double remind the user in Report.js

Programme 2.27 – Function to handle the notification for call that contains information in the database after the end of the call in App.js

Programme 3.1 – Original Python function to perform data parsing and scrapping on HKJunkCall with Requests and BeautifulSoup4

Programme 3.2 – Python function to perform data parsing and scrapping on HKJunkCall with Selenium and BeautifulSoup
**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated development environment</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
</tbody>
</table>
1. Introduction

The following chapter introduces the background, motivation as well as objectives of the project, followed by the outline of the remaining part of the report.

1.1 Background

1.1.1 Current Situation

Under the coronavirus pandemic, phone scams in Hong Kong have been on an upward trajectory. There were 1,193 cases in 2020, which has been increased by 80 percent from 2019. It has resulted in a cumulative loss of $93 million in 2019 and 2020. In the first quarter of 2021, 200 phone scams were recorded and have resulted in a total of $45 million of financial losses [1]. While the elderly are always the most common targets for telephone deception, cumulative financial losses are reported nearly $100 million, with each losing about $380,000 on average as at 2020 [2]. Based on the above statistics, it is found that the problem of telephone deception is serious among the elderly in Hong Kong.

According to the data of 2019 and 2020, there are a few popular approaches in telephone deception. For example, scammers may pretend officials and call the victims with pre-recorded voice messages, and ask them to give out the login detail of the bank account [3]. In addition, phone number starting with “+852” are most likely to be scam call as the Office of the Communications Authority has indicated that the calls originate from outside Hong Kong may be masqueraded as Hong Kong phone numbers by displaying Hong Kong’s area code ”852” [4]. The most common methods of operation utilized by scammers were to pretend officials, asking victims to guess who he or she is and transfer them money [5].

1.1.2 Review on Existing Approaches

Regarding the problem of phone scamming, different mobile applications have been launched to address this problem, including Call Defender (小熊來電), Whoscall, Jima CallerID (芝麻來電), Headuck (小鴨幹線) and Call Advisor [6]. Android application Call Defender and iOS application Jima Caller ID are popular mobile applications to filter out, identify and block the scam calls. Functions like displaying a specific incoming call UI with alert and information of the suspicious call, as well as hanging up or muting the suspicious incoming call automatically are available in these existing applications [7][8]. To perform the above functions, these
applications has been relied on the database of HKJunkCall and JunkCall HK. They are the database that contains data of scam call in Hong Kong, has provided information of the suspicious call as well as an API for the applications in accessing and updating the call data. The five applications have been connected to HKJunkCall while only Whocalls, Call Defender, Jima CallerID and Headuck have connected to JunkCall HK, and they may be able to get the information of the scam call in Hong Kong by calling the HKJunkCall API or JunkCall HK API with the suspicious phone number [9][10].

Although these applications seem to be efficient in avoiding telephone deception, the number of telephone deception cases is still rising, there are still shortcomings. The existing applications mostly performed in filtering and blocking the call, which does not arouse the awareness or educate the user in avoiding telephone deception. According to HKJunkCall, these applications used to hang up the call which causes the number of unknown incoming call rises and increases the risk on missing important call e.g. call from hospital [11]. In addition, the existing applications consist of complex UI and functionality which is not elderly-friendly. For example, Whoscall requires email registration to use the application [12]. Moreover, a high proportion of the devices’ memory is occupied in Call Defender and Jima CallerID as the database will be downloaded to the device for call screening [13].

1.1.3 Project Overview

In light of the trend of telephone deception cases that happened on the elderly and the imperfection of the existing applications, we are going to develop an elderly-friendly Android mobile application, Sieve (筛), which is designed to screen out scam calls thoroughly. Sieve will be delivered in Traditional Chinese. It is the enhanced version of Call Defender, acting as a sieve for screening the incoming call by checking the database of suspicious calls with the help of JunkCall HK [14]. Instead of hanging up the call, it is aimed to screen the call and provide the user with information of the call as well as mute the incoming notification thus reducing the chances for the user to pick up the call. According to HKJunkCall, the number of future suspicious call has dropped about 77% if the user ignore incoming call instead of hanging it up [11]. Hanging up the call may also increase the risk of missing important call e.g. call from hospital. It will also focus on educating and arousing the awareness of the elderly in avoiding the telephone deception through sending out warning and interactive notification to the users. Call reporting are also available for user to report the suspicious call. In addition,
call searching feature and call log of unsaved calls with information extracted from database will be provided, together with the regular reminders for the elderly to avoid telephone deception will be pop-up from time to time as notification.

Figure 1.1 – Icon of Sieve

1.2 Motivation

1.2.1 Safeguard the Interests of the Elderly

The major momentum to launch such an app is stemmed from the social responsibility for safeguarding the properties of the elderly. Given that seldom have some of them been promoted or educated about the tactics of phone frauds, their savings can be easily deprived due to sympathy and lack of alertness. Tragedies are not uncommon, a 90-year-old woman was deceived with the loss of HK$250 million by a phone scam, claiming himself as a mainland official [15]. It is aimed that the app can act as a barrier to protect the interests of the elderly, as well as prevent their sympathy and carelessness from being manipulated.

1.2.2 Offer Technical Supports for the Elderly to Adapt the E-generation

Another trigger to devise this app is since the elderly are losing track during the digitalized generation. Often, the majority are lacking sufficient knowledge of how outlaws utilize technological tools for scams, resulting in falling prey to the deceptions. Similar to the case with the 90-year-old victim, bulk bogus massages or recordings can be sent to the phones [15]. Thus, it is envisaged that the elderly can be equipped with more experience in using the app for combating crimes and be vigilant with the aid of advanced technologies. The app is hoped to embrace their generations with technological supports.

1.2.3 Work as Deterrent to the Malpractice

The app is also hoped to arouse awareness on the severity brought by phone deception, as well as present as a warning to deter the outlaws from committing such misconduct. With this idea,
it is motivated to design the app with the function to alert the elderly once sensitive wordings have been received during the phone call. Also, the suspected phone numbers will keep sharing with the public, which will be likely to reduce the occurrence of such frauds and maintain social harmony, as it may be harder for the criminals to persuade the elderly in the presence of the reminders from the app.

1.3 Objectives

This project aims to build a mobile application to help the elderly to avoid telephone deception. The objectives of this project are listed below.

1. To provide an elderly friendly UI UX
2. To parse and scrap data on JunkCall HK and synchronize and update the database with the scrapped data and the reported call
3. To create a database for storing suspicious calls
4. To reduce the number of handling suspicious incoming call
5. To scan the incoming calls from database
6. To ask the user if the call contains suspicious content after hanging up the call
7. To allow users to report the scam calls and update the database
8. To allow users to search and get the information from previous unknown call
9. To send regular reminder of tips in avoiding telephone deception
10. To silent the incoming scam call notification

The scope of the project involves five major parts, a function to scan the suspicious calls from the database and give out warnings, ask if the call contains suspicious content after hanging up, the phone call reporting system for users to report the scam calls as well as call searching and log tracking, as well as regular reminders to educate the user.

1.4 Report Outline

The remainder of the report is organized into three parts: Chapter 2 discusses the methodology and approaches of the project. Chapter 3 presents the progresses and results of the project, demonstrating what have been done so far and the results of the projects. Potential challenges and mitigations are also included. The future improvements with the project schedule will be illustrated in Chapter 4. Last but not least, Chapter 5 summarizes the overall picture of the project.
2. Methodology

The following chapter discusses the approaches of the project, including the implementation detail and technique to be used. The major features and the workflow of the project will be introduced.

2.1 IDE and Language

React Native will be used as the IDE for the application development as there are sufficient packages for building the application, as well as it is fast and easy to operate [16]. JavaScript will be the main language used for frontend development [17]. Python will be used for the development as it is applicable in data parsing and scrapping [18]. While Google Colab and Jupitar Notebook will be used to run the function of phone call information parsing and scrapping with JunkCall HK, since they are easy to maintain and user-friendly with different built-in features that help facilitate the development process [19][20].

Steps to run the application on emulator or Android devices

To run on Android devices, devices should be connected to the computer and check if it is successfully connected by the following code:

$ adb devices

1. To install the packages that are required to run the application, go to fyp1/level and run the following command in terminal:

$ npm install

2. To run the application, run the following command:

$ react-native run-android

2.2 UI/UX Design

Design platform Figma has been used for UI/UX design since it provides various design ideas and templates. As the application is tailor-made for the elderly, accessibility will be one of the main focuses of the UI/UX design. Research and interviews have been done to investigate the needs of the elderly user.
As shown in the example in Fig. 2.1, the text size will be large enough to make it clear to read together with a line spacing of at least 1.5 [15]. Especially when the elderly is much less eager to perform multi-finger gestures, gesture controls will be implemented with care, such as “pinch to zoom” will not be needed with the large text size [17]. At least 20 percent of the elderly in Hong Kong are suffering from vision loss each year, for the sake of visual disability, the contrast ratio will be at least 5.5:1 (see Fig. 2.2) [17] [18]. The UI and content will stay neat and tidy with simple and precise functionality. Assuming most elderly read and speak Chinese, for their convenience, content is provided in Chinese.

Figure 2.1 – Example of UI with large font size
Figure 2.2 – Example of different contrast ratio
Figure 2.3 – Current UI of Sieve
Fig. 2.3 shows the current UI of Sieve, different elderly-friendly feature such as larger font size, direct and simple navigation as well as high contrast ratio has been implemented. It consists of three screens i.e. Search (Welcome.js) (see Fig. 2.3(a)), History (History.js) (see Fig. 2.3(b)) and Report (Report.js) (see Fig. 2.3(c)). The needs of typing have also been minimized, only two mandatory field are required to fill and dropdown picker are used for call reporting to reduce the input error. The detailed functionality of the screens will be discussed and justified in the later sections.

### 2.3 Database of Suspicious Calls

A database to store the scam calls will be developed to scan if the incoming call is suspicious. The data collection of the scam calls will be conducted with the help of the HKJunkCall API in the process of regular database synchronisation, incoming call screening (see section 2.5), as well as the suspicious calls reported by the users (see section 2.6). Firebase by Google are used as the database since it is a real-time database which allow syncing the real-time data across all devices [19]. It is a database structured in JSON tree which is free of charge and easy to maintain. Fig. 2.4 shows the entity relationship of the database. There is only one object “Phone”, with child “PhoneNumber”, “Organization”, “Description” and “Category”, which stores the phone number, organization, description and the category of the call. The child “PhoneNumber” is the primary key of the database.

<table>
<thead>
<tr>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhoneNumber</td>
</tr>
<tr>
<td>Organization</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Category</td>
</tr>
</tbody>
</table>

**Figure 2.4** – The entity-relationship diagram of database

A GET API written in Python for parsing and scrapping the information of the incoming call from JunkCall HK is developed. JunkCall HK is a website that provides the list of call information according to their categories with connection to Whocalls, Call Defender, Jima CallerID and Headuck, which allows backtracking on the scam calls [10]. Requests together with Python library BeautifulSoup4 has been used to perform web parsing and scrapping on the data on the website [20]. As shown in the code below:
from bs4 import BeautifulSoup
import urllib3
import requests.packages.urllib3
requests.packages.urllib3.disable_warnings()

import requests
from firebase import firebase
key = '0pB9SnU1pgyRvnG9dupf8feVudk4YGU50HJpy0qA'
authentication = firebase.FirebaseAuthentication(key, '1lpy1114@connect.hku.hk')
firebase.authentication = authentication
user = authentication.get_user()

proxy = '169.57.1.85:8123'

http = requests.packages.urllib3.PoolManager()
url = "https://www.junk-call.com/hk/詐騙"
response = http.request('GET', url)

soup = BeautifulSoup(response.data)

h2 = soup.find("h2")
print(h2)

ols = []
for nextSibling in h2.findNextSiblings():
    if nextSibling.name == 'ol':
        ols.append(nextSibling)

lis=[]
for li in ols[0].findAll('li'):
    lis.append(li)

dataAll=[]
k = 0
for li in lis:
    if li.find("div", {"class": "right"}):
        remark = li.find("div", {"class": "remark"})
        i = 0
        dataAll.append([])
        for a in li.find_all('a', href=True):
            if i > 1:
                dataAll[k].append(li.find('a', {'href': a['href']})).text
            i = i+1
        if(remark != None):
            dataAll[k].append(remark.text)
            k = k+1

print(dataAll)
for entry in dataAll:
    phone = entry[0]
#phone = datak[0][0]
dbphone = "number/" + phone
firebase.put(dbphone,"organisation",entry[1])
firebase.put(dbphone,"category",entry[2])
if(len(entry)==4):
    firebase.put(dbphone,"description",entry[3])

Programme 2.1 – Python GET API for data parsing and scrapping on JunkCall HK

By calling the Python function with url https://www.junk-call.com/hk/詐騙, the source code of the html will be parsed with Requests. Since the tags and attributes of the site are not well developed e.g. class attributes are not available for some description of the call, patterns are observed to scrap the right data. Through identifying the pattern of hyperlink i.e. href in the list <li> of the first order list <ol> under the first <h2>, the list of scam call including the number with corresponding organisation category and description are scrapped with BeautifulSoup4 and added to the real-time database in Firebase (see Fig. 2.5). To prevent from outdating of the database, synchronisation will be performed every three months.

Figure 2.5 – Information extracted by web scrapping

In the application, firebase configuration is also required (see Programme 2.2). Detail steps can be found in the website of Firebase and the project setting in the Firebase account [21]. After configuration, getDatabase() can be called to connect to the database, and getNumber() can be called to retrieve the data from the database (see Programme 2.3).
The project aims to develop an end-to-end solution, which the application will help the elderly users to screen out suspicious calls and give out warnings to the elderly. Fig. 2.6 has demonstrated the major workflow of the mobile application. The solution consists of four main parts: (a) incoming call screening, (b) actions and questionnaire for calls from unsaved contact without information found in database, and (c) actions and alert for calls from unsaved contact with information found in database. They are represented in orange, purple, blue and green respectively in Fig. 2.6. As shown in Fig. 2.6(a), when an incoming call is received, the number will be checked if it is a saved contact in the user’s contact list, if yes, the database of scam calls will be inspected to see if the number has been recorded previously. If the number has been found suspicious from the database, a customized incoming call notification with the information found will be given and the ringtone will be disabled and changed to silent mode. However, if there are no records of the number found in the database, the application will just return and display a customized incoming call notification of “Unknown Call” (不明來電), the detail technology used will be discussed in section 2.5.
Figure 2.6 – Major workflow of the application

When the unknown call (without information found in the database) ends, there will be a “Yes” or “No” questionnaire appeared as a notification to ask if the call contains suspicious content thus performing post-tracking on incoming suspicious call as shown in Fig. 2.6(b). If the user clicks “Yes”, it will be directed to the reporting system with the corresponding phone number filled in in the reporting form. A dialog box will also be appeared in the Report screen to encourage to user to seek help if it is a suspicious call. While if the call is not an unknown call which there is information found in the database, when the call ends, an alert with the
information of the call will be displayed to remind the user (see Fig 2.6(c)). The detail will be explained in section 2.10.

Lastly, users are allowed to report for the number if they think that the number is suspicious after the call. As shown in Fig. 2.6(d), the database will be updated with suspicious calls reported by the users. The techniques used will be discussed in detail in section 2.7. Apart from the above, regular reminders will be delivered to the user to raise their awareness in preventing telephone deception, which will be introduced with the technology used in section 2.9. Also, call searching function and call log tracking are also available in the application for the user to keep track on the information of the other suspicious or previous calls. The detail of implementation will be discussed in section 2.7 and 2.8.

### 2.5 Incoming Call Screening

Screening will be performed and alert including the information of the call will be pop up whenever there is any incoming call. The function will first extract the number of the call and event e.g. incoming by CallDetection module from ‘react-native-call-detection’ with permission READ_PHONE_STATE [22]. Module ‘react-native-call-log’ will also be used to identify whether the call is from a saved contact, no alert will be displayed if it is from a saved contact, permission READ_CALL_LOG is required to implement the function [23]. The permission granting and module adding can be referred to Programme 2.4. Incoming call will be detected with the corresponding ‘phoneNumber’ and incoming ‘event’ by CallDetectorManager as shown in Programme 2.5. By using the API CallLogs.load(1), the most recent call information including the name of the call in the contact list can be loaded, if any. The incoming call can be identified if it is a saved contact, a flag flagContact has been set to distinguish whether the call is a saved contact.

```jsx
import CallLogs from "react-native-call-log";
import CallDetectorManager from "react-native-call-detection";

const result = await PermissionsAndroid.check(
  
Programme 2.4(a) – Adding permission in AndroidManifest.xml

<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.READ_CALL_LOG" />

Programme 2.4(b) – Module import to use CallLogs and CallDetectorManager in App.js
```
PermissionsAndroid.PERMISSIONS.READ_CALL_LOG

const granted = await PermissionsAndroid.request(
    PermissionsAndroid.PERMISSIONS.READ_CALL_LOG,
    {
        title: "Call Log Example",
        message: "Access your call logs",
        buttonNeutral: "Ask Me Later",
        buttonNegative: "Cancel",
        buttonPositive: "OK",
    }
);

Programme 2.4(c) – Permission granting for READ_CALL_LOG in App.js

... callDetector = new CallDetectorManager(
    event, phoneNumber) => {
    if (event === "Incoming") {
        var flagContact = 0;
        CallLogs.load(1).then((c) => setName(c[0].name));
        console.log(setName);
        if (callName === "" || callName == null){
            flagContact = 1;
        }
    if (flagContact == 1){
        setName(phoneNumber);
        console.log(phoneNumber);
        displayIncomingCall(phoneNumber);
        stopListenerTapped();
    }
    ...
    stopListenerTapped = () => {
        callDetector & callDetector.dispose();
    }
    ...

Programme 2.5 – Function to detect the incoming call with CallDetection in App.js

If it is a call from unsaved contact, displayIncomingCall(number) will be called and checking will be performed to see if the call has been recorded in the database (see Programme 2.6), data will be extracted from the database if there is any, it will be performed in the getInfo(phoneNumber) (see Programme 2.7).

const displayIncomingCall = (number) => {
    var warn = getInfo(number);
    console.log(warn);
    const callUUID = getNewUuid();
    if (warn != "不明來電") {
        changeMode(RINGER_MODE.silent);
    }...
warn = "("+number+")"+warn;
addCall(callUUID, number);
console.log(calls);
log(['displayIncomingCall] ${format(callUUID)}, number: ${number}');
RNCallKeep.displayIncomingCall(callUUID, number, warn, 'number', false);

Programme 2.6 – Function to display the customised notification for incoming call in App.js

const getInfo = (phoneNumber) =>{
  var category = "";
  var name = "";
  var flaga = 0;
  Object.entries(result).map((entry, index) => {
    if (entry[0] == phoneNumber) {
      name = entry[1].organisation;
      category = entry[1].category;
      flaga = 1;
    }
  })
  if(flaga==1){
    var warn = name+"|"+category;
    console.log(warn);
    return warn;
  }else{
    return "不明來電";
  }
};

Programme 2.7 – Function to get the information of the call from the database in App.js

With getNumber() that mentioned in section 2.3 and Programme 2.3(b), the application is linked to the real-time database in Firebase, the result has stored the data of the calls. The function getInfo(number) may return “不明來電” if there is no information found in the database, or it will return the information of the call if there is any. The returned value will be stored in warn. Referring to Programme 2.8 below, if it is not an unknown call, the ringtone will change to silence by changeMode(RINGER_MODE.silent) with the help of module “react-native-ringer-mode” [24], as silence may reduce the number of future incoming suspicious call according to the statistic from HKJunkCall [11].

import {useRingerMode, RINGER_MODE, checkDndAccess, requestDndAccess} from 'react-native-ringer-mode';

Programme 2.8(a) – Module import to change the ring mode in App.js

const { mode, setMode } = useRingerMode();
const changeMode = async (newMode) => {
    if (newMode === RINGER_MODE.silent || mode === RINGER_MODE.silent) {
        const hasDndAccess = await checkDndAccess();

        if (hasDndAccess === false) {
            // This function opens the DND settings.
            // You can ask user to give the permission with a modal before calling this function.
            requestDndAccess();
            return;
        }
    }

    setMode(newMode);
};

Programme 2.8(b) – Function to change the ring mode in App.js

After all, as shown in Programme 2.9, by ConnectionService on Android and ‘react-native-callkeep’ module, RNCallKeep.displayInocmingCall(CallUUID, handle, localizedDescription, handleType, hasVideo) function can be called, which the CallUUID is a unique uuid for each call, handle is the number of the call, the localizedDescription is the message that will be displayed when there is incoming call, and the handleType will be ‘number’ and hasVideo will be false. An alert of incoming call with warn “不明來電” or the information of the call will be pop up immediately as shown in Fig. 2.7 [25][26].

<uses-permission android:name="android.permission.READ_CALL_LOG" />

Programme 2.9(a) – Adding permission to read call log in AndroidManifest.xml

<service android:name="io.wazo.callkeep.VoiceConnectionService"
    android:permission="android.permission.BIND_TELECOM_CONNECTION_SERVICE"
    android:foregroundServiceType="phoneCall">
    <intent-filter>
        <action android:name="android.telecom.ConnectionService" />
    </intent-filter>
</service>

Programme 2.9(b) – Adding service ConnectionService for CallKeep in AndroidManifest.xml

import RNCallKeep from "react-native-callkeep";

Programme 2.9(c) – Module import to use CallKeep in App.js

const options = {
    ios: {

Programme 2.9(d) – Set up for RNCallKeep in *App.js*

```javascript
RNCallKeep.setup(options).then(({accepted} => {}));
RNCallKeep.setAvailable(true);
RNCallKeep.backToForeground();
```

Programme 2.9(e) – Display the customised notification for incoming call in *App.js*

```javascript
RNCallKeep.displayIncomingCall(callUUID, number, warn, 'number', false);
```

Figure 2.7 – Alert with call information when there is incoming call

### 2.6 Phone Call Reporting System

Users are also encouraged to report the call if they think that is suspicious after the call. There will be a form with different field such as phone number, category, organisation, description for user to report. As interviewed with the elderly, most of them prefer knowing only whether the number is suspicious or not, relatively less interest on knowing the organisation and the reasons why it is a scam. Therefore, in order to reduce the complexity, there are only two
mandatory fields for the user to input, which is “Phone Number” and “Category”. Numeric keyboard is used for inputting “Phone Number”. In addition, to reduce the input error, there will be a dropdown list i.e. `<DropDownPicker>` from “react-native-dropdown-picker” for user to pick the corresponding category i.e. telemarketing (廣告電話) or suspected scam (疑似詐騙), under the “Category” field [27]. As it will be demanding for the elderly to type the reasons why it is suspicious, therefore, if “suspected scam” is chosen in the “Category” field, fields i.e. dropdown list for scam reasons (詐騙內容) like “Fake bank” or “Fake government”, as well as radio button `<RadioButton>` from “react-native-paper” for user to choose whether the call has asked for money transfer (有沒有要求匯款?), will be displayed dynamically and available for choosing [28]. The code of the form can be referred in Programme 2.10 below. The UI of the form may be referred in Fig. 2.3(c) in section 2.2. The choices will also be listed in a drop-down list for users to choose and submit. Users may also input the additional information or the reasons that are not listed, in the “Description” field if necessary.

```javascript
import DropDownPicker from "react-native-dropdown-picker";
Programme 2.10(a) – Packing import to use DropDownPicker in Report.js

import { RadioButton } from "react-native-paper";
Programme 2.10(b) – Module import to use RadioButton in Report.js

import { ScrollView, TextInput, Keyboard, TouchableOpacity} from "react-native";
Programme 2.10(c) – Module import to use TextInput and TouchableOpacity in Report.js

<ScrollView style={styles.dropdownview}>
    <View style={styles.formline}>
        <Text style={styles.formtitle}>電話號碼:</Text>
        <View style={{ width: "62%" }}>
            <TextInput
                placeholderTextColor="#b3b1b1"
                style={styles.textinput}
                placeholder="輸入電話號碼"
                maxLength={20}
                onBlur={Keyboard.dismiss}
                value={phone}
                keyboardType="numeric"
                onChangeText={(text) => {
                    setPhone(text);
                }}
            />
        </View>
    </View>
</ScrollView>
```
<View style={styles.formlinecat}>
  <Text style={styles.formtitle}>類別:</Text>
  <View style={{ width: "62%" }}>
    <DropDownPicker
      textStyle={{
        fontSize: 21,
        marginRight: 10,
        marginLeft: 10,
      }}
      placeholderStyle={{
        color: "#b3b1b1",
      }}
      open={open}
      items={items}
      setOpen={setOpen}
      setValue={setValue}
      setItems={setItems}
      itemStyle={{ justifyContent: "center" }}
      placeholder="選擇類別"
      value={value}
    />
  </View>
</View>

{value == "疑似詐騙" ? (}
  
  <View style={styles.formline}>
    <Text style={styles.formtitle}>詐騙內容:</Text>
    <View style={{ width: "62%" }}>
      <DropDownPicker
        textStyle={{
          fontSize: 21,
          marginRight: 10,
          marginLeft: 10,
        }}
        placeholderStyle={{
          color: "#b3b1b1",
        }}
        open={open2}
        items={scamItems}
        setOpen={setOpen2}
        setValue={setScamValue}
        setItems={setScamItems}
        itemStyle={{ justifyContent: "center" }}
        placeholder="選擇詐騙內容"
        value={scamValue}
      />
    </View>
  </View>
</View>

<View style={styles.formline}>
  <Text style={styles.formtitle2}>有沒有要求匯款？</Text>
</View>
<View style={{ fontSize: 10 }}>
    <RadioButton.Group
        onValueChange={(value) => setChosenOption(value)}
        value={chosenOption}
    >
        <RadioButton.Item
            label="有"
            value="有"
            color="#1E6DAD"
            style={{ fontSize: 30 }}
        />
        <RadioButton.Item
            label="沒有"
            value="沒有"
            color="#1E6DAD"
            style={{ fontSize: 30 }}
        />
    </RadioButton.Group>
</View>

<Text style={styles.formtitle2}>機構（選填）:</Text>
<TextInput
    placeholderTextColor="#b3b1b1"
    style={styles.textInput}
    placeholder="輸入機構名稱"
    maxLength={20}
    onBlur={Keyboard.dismiss}
    value={name}
    onChangeText={(text) => {
        setName(text);
    }}
/>

<Text style={styles.formtitle2}>備註（選填）:</Text>
<TextInput
    placeholderTextColor="#b3b1b1"
    style={styles.textInput}
    placeholder="輸入備註"
    maxLength={20}
    onBlur={Keyboard.dismiss}
    value={description}
    onChangeText={(text) => {
        setDescription(text);
    }}
/>
Programme 2.10(d) – Form of phone call reporting in Report.js

<TouchableOpacity> is used as a button to submit will also be available [29]. Database will be updated under the handleSubmit() when the form is submitted. Dialog box <Dialog> is used to display warning if the input is invalid or the form has been successfully submission [30]. The function of handling the submission as well as the dialog box can be referred in Programme 2.11 and Programme 2.12.

```javascript
function handleSubmit() {
  if (phone !== null && value !== null) {
    const reference = ref(db, `/number/` + phone);
    var organisation;
    var tempdescription;
    if (name !== null && scamValue !== null) {
      organisation = scamValue + `|` + name;
    } else if (name === null && scamValue !== null) {
      organisation = scamValue;
    } else {
      organisation = name;
    }
    if (chosenOption === '有' && description !== null) {
      tempdescription = `注意！此碼有要求匯款; `' + description;
    } else if (chosenOption === '有' && description === null) {
      tempdescription = `注意！此碼有要求匯款`;
    } else {
      tempdescription = description;
    }
    set(reference, {
      category: value,
      organisation: organisation,
      description: tempdescription,
    });
    setValue(null);
    setScamValue(null);
  }
```
```javascript
setPhone(null);
setName(null);
setDescription(null);
setChosenOption(null);
showDialog();
}
else{
  setVisibleVerify(true);
}
}
```

**Programme 2.11** – Function to handle the form submission in *Report.js*

```javascript
(Dialog.Container visible={visible})
  <Dialog.Description>報復成功！</Dialog.Description>
  <Dialog.Button label="知道" onPress={handleCancel} />
</Dialog.Container>
```

**Programme 2.12(a)** – Dialog confirmation of form submission in *Report.js*

```javascript
(Dialog.Container visible={visibleVerify})
  <Dialog.Description>請輸入電話號碼及選擇類別！</Dialog.Description>
  <Dialog.Button label="知道" onPress={handleCancel} />
</Dialog.Container>
```

**Programme 2.12(b)** – Dialog warning if the input is invalid in *Report.js*

### 2.7 Call Searching

Phone number searching feature is available for user to search and get the information of a corresponding number. A search bar will be provided for user to input the number, result will be displayed in a list. The search function is developed in the search screen i.e. *Welcome.js*. It will be connected and retrieved data from the real-time database in Firebase by using `getNumber()` mentioned in section 2.3. `<InputField>` are developed as a search bar with the help of `<TextInput>` as shown Programme 2.13, it is stored in the “components” folder in the project.

```javascript
import React from 'react';
import { View, StyleSheet, TextInput, TouchableOpacity } from 'react-native';
import { MaterialCommunityIcons } from '@expo/vector-icons';

const InputField = ({
  leftIcon,
  iconColor = '#000',
  rightIcon,
  InputStyle,
  containerStyle,
  placeholderText = '#444',
  handlePasswordVisibility,
  ...rest
```
```javascript
const styles = StyleSheet.create({
  container: {
    borderRadius: 10,
    flexDirection: 'row',
    padding: 12,
  },
  leftIcon: {
    marginRight: 10
  },
  input: {
    flex: 1,
    width: '100%',
    fontSize: 18,
  },
  rightIcon: {
    alignItems: 'center',
    marginLeft: 10
  }
});

export default InputField;
```

Programme 2.13 – <InputField> component in components/InputFields.js
The `<InputField>` allows the user to input the number with a numeric keyboard by defining `keyboardType = "number-pad"`, which reduces the input complexity (see Programme 2.14). The function `handleSearch(text)` may return results that matches or includes the input number `text`, with the corresponding organization and category from the database at real-time, i.e. inputting “23” may display information of numbers such as “34323888”, “23999999” immediately without clicking a button to trigger the search (see Programme 2.15).

```javascript
import { InputField } from "../components";

Programme 2.14(a) – Module import for InputField in Welcome.js

```<View style={styles.searchbar}>
  <InputField
    inputStyle={{
      fontSize: 22,
    }}
    containerStyle={{
      backgroundColor: "#E8E8E8",
    }}
    leftIcon="magnify"
    placeholder="搜尋電話"
    autoCapitalize="characters"
    autoCorrect={false}
    textContentType="password"
    value={number}
    keyboardType="number-pad"
    //enablesReturnKeyAutomatically={false}
    onChangeText={(text) => {
      setNumber(text);
      handleSearch(text);
    }}
  />
  <ScrollView
    keyboardDismissMode="none"
    style={styles.scrollView}
    nestedScrollEnabled={true}
  >
    {numberList}
  </ScrollView>
</View>

Programme 2.14(b) – Search bar in Welcome.js

```javascript
function handleSearch(text) {
  temp = {};
  if (text !== "") {
    Object.entries(result).map((entry, index) => {
      if (entry[0].includes(text)) {
```
temp[entry[0]] = entry[1];

});

setAllNumber(temp);

Programme 2.15 – Function to handle the search in Welcome.js

The <ScrollView> may display the numberList, that includes allNumber that set in the handleSearch(text), and returning <TouchableOpacity> with the number, organisation and category for each entry in allNumber (see Programme 2.16). Entries are available for clicking and will be redirected to the Detail screen i.e. Detail.js by navigation.navigate("Detail", { phone: numm[0] }) with the corresponding number, the number will be available in route.params.phone in Detail component and the value will be set into phone (see Programme 2.17(b)), the information of the call can be extracted and display with parameter phoneNumberDetail (see Programme 2.17(c)). To do this, the navigation prop is passed down to the Welcome screen component as shown in Programme 2.18 [31]. The route prop is also passed down to the Detail screen component as shown in Programme 2.17(a).

```
const numberList = Object.entries(allNumber).map((number, index) => {
  return (
    <View key={index} style={styles.calllist}>
      <TouchableOpacity onPress={() => navigation.navigate("Detail", { phone: number[0] })}>
        <Text style={styles.NumberTitle}>{number[0]} {number[1].category}</Text>
        <Text style={styles.NumberSubTitle}>{number[1].organisation}</Text>
      </TouchableOpacity>
    </View>
  );
});
```

Programme 2.16 – Parameter numberList for the matched numbers in Welcome.js

```
export const Detail={({navigation, route})=>{...
```

Programme 2.17(a) – Route prop in Detail component in Detail.js

```
const [phone, setPhone] = useState(route.params.phone);
```

Programme 2.17(b) – Set the value of phone be the number pass from Welcome component in Detail.js
const phoneNumberDetail = Object.entries(result).map((number, index) => {
  var category = "";
  var name = "";
  var description = "";
  if (number[0] === phone) {
    category = number[1].category;
    if (number[1].description !== null) {
      description = number[1].description;
    } else{
      description = "/";
    }
    if (number[1].organisation !== null) {
      name = number[1].organisation;
    } else{
      name = "/";
    }
  return ({
    <View style={{padding:9}}>
      <View style={styles.formline}>
        <View style={styles.formline2}>類別: </View>
        <Text style={styles.formtitle2}>{category}</Text>
      </View>
    </View>
    <View style={styles.formline}>
      <View style={styles.formline2}>機構: </View>
      <Text style={styles.formtitle2}>{name}</Text>
    </View>
    <View>({description === '/' ? (<View><View style={styles.formline}>
          <Text style={styles.formtitle2}>備註: </Text>
          <Text style={styles.formtitle2}>{description}</Text>
        </View>) : <View><View style={styles.formline}>
          <Text style={styles.formtitle2}>備註: </Text>
        </View></View>})
    </View>
    </View>
  });
});

Programme 2.17(c) – The detail of call to be displayed in Detail.js

export const Welcome = ({ navigation, route }) => {
  ...

Programme 2.18 – Welcome component with navigation prop in Welcome.js
2.8 Call Log

Besides searching the number, the users are allowed to view log and history of unknown call together with the information from Firebase. This function is available in the History component. By granting the READ_CALL_LOG permission and using the CallLogs module from ‘react-native-call-log’, the log of previous call can be extracted (see Programme 2.19) [32].

```xml
<uses-permission android:name="android.permission.READ_CALL_LOG" />
```

Programme 2.19(a) – Adding Permission to read call log in AndroidManifest.xml

```javascript
import { PermissionsAndroid } from "react-native";
import CallLogs from 'react-native-call-log';
```

Programme 2.19(b) – Module import for call log reading in History.js

```javascript
useEffect(() => {
  (async () => {
    try {
      CallLogs.load(30).then(c => console.log(c))
      const result = await PermissionsAndroid.check(PermissionsAndroid.PERMISSIONS.READ_CALL_LOG);
      const granted = await PermissionsAndroid.request(PermissionsAndroid.PERMISSIONS.READ_CALL_LOG,
      {
        title: 'Call Log Example',
        message: 'Access your call logs',
        buttonNeutral: 'Ask Me Later',
        buttonNegative: 'Cancel',
        buttonPositive: 'OK',
      });
      if (granted === PermissionsAndroid.RESULTS.GRANTED) {
        CallLogs.load(-1).then(c => setCallLog(c));
      } else {
        console.log('Call Log permission denied');
      }
    } catch (e) {
      console.log(e);
    }
  })();
}, []);
```

Programme 2.19(c) – Permission granting when accessing the History component at first render in History.js
Only unknown calls will be filtered out and displayed, since user may only want to know about the information of the unknown calls. The parameter callLogList will be storing the log in the form of `<View>` (see Programme 2.20(a)). It will be displayed in a `<ScrollView>` in the History component (see Programme 2.20(b)). By checking if `number[1].name == null` as `number[1].name` will be the name of contact if it is a saved contact. If the call is not from a saved contact, it will extract the information including the organisation and category of the call from the database if there is any. A flag `flaga` is used to distinguish whether the call has information in the database, it will be turned into 1 if there is entry found in the database, otherwise 0. Thus, if the call has information in the database which `flaga` is equal to 1, the log with the information i.e. number, category and organisation of the call will be displayed. Similar to the list of searching result mentioned in section 2.7, user may click on the entry and it will be redirected to the Detail component with the detail information of the call. While if the call has no information found in the database which `flaga` is equal to 0, a log with the number and description “不明来電” together with a button to report will be displayed. Number will be displayed in all logs along with the time when the call is made. The button is `<TouchableOpacity>` and it will be redirected to the Report screen with the corresponding number when it is clicked, the “Phone Number” field in form will be filled in with the number automatically. Thus, simplifying the steps for reporting the call.

```javascript
const callLogList = Object.entries(callLog).map((number, index) => {
  var category = "";
  var name = "";
  var flaga = 0;
  if (number[1].name == null) {
    Object.entries(result).map((h, index) => {
      if (h[0] == number[1].phoneNumber) {
        name = h[1].organisation;
        category = h[1].category;
        flaga = 1;
      }
    });
  }
  if (flaga == 1) {
    return {
      <View key={index} style={styles.calllist}>
        <View style={{ width: "80%" }}>
          <TouchableOpacity
            onPress={() =>
              navigation.navigate("Detail", { phone: number[1].phoneNumber })
            }
          >
            <Text style={styles.NumberTitle}>{category}</Text>
            <Text style={styles.NumberTitle2}>
          </View>
        </View>
      </View>
    }
  }
})
```
Programme 2.20(a) – Parameter to store the log in History.js

Programme 2.20(b) – ScrollView to display the log callLogList in History.js

2.9 Regular Reminder to Avoid Telephone Deception

To educate the elderly and arouse the awareness in preventing phone scam, notification of tips in avoiding telephone deception will be send to the elderly in a regular basis, such as every day. The notification is created in the App.js and will be running in background even the app is killed. PushNotification module from 'react-native-push-notification' package with permission VIBRATE and RECEIVE_BOOT_COMPLETED, as well as receiver and services of RNPushNotification as shown in Programme 2.21, will be used to create regular notification [33]. Configuration is also done in the index.js. The detail installation can be found in https://github.com/zo0r/react-native-push-notification [34].
import PushNotification from "react-native-push-notification";

Programme 2.21(a) – Module import for PushNotification in App.js

<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>

Programme 2.21(b) – Adding permissions for PushNotification in AndroidManifest.xml

<receiver
    android:name="com.dieam.reactnativepushnotification.modules.RNPushNotificationActions" />
<receiver
    android:name="com.dieam.reactnativepushnotification.modules.RNPushNotificationPublisher" />
<receiver
    android:name="com.dieam.reactnativepushnotification.modules.RNPushNotificationBootEventReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.QUICKBOOT_POWERON" />
        <action android:name="com.htc.intent.action.QUICKBOOT_POWERON" />
    </intent-filter>
</receiver>

<service
    android:name="com.dieam.reactnativepushnotification.modules.RNPushNotificationListenerService"
    android:exported="false" >
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>

Programme 2.21(c) – Adding receiver and service for PushNotification in AndroidManifest.xml

PushNotification.configure{
    // (optional) Called when Token is generated (iOS and Android)
    onRegister: function (token) {
        console.log("TOKEN:", token);
    },
    // (required) Called when a remote is received or opened, or local notification is opened
    onNotification: function (notification) {
        console.log("NOTIFICATION:", notification);
        // process the notification
    }
}
function delay(n) {
  return new Promise(function (resolve) {
    setTimeout(resolve, n * 1000);
  });
}
const handleNotification = async () => {
    for (const warn in warning) {
        await delay(10);
        PushNotification.localNotificationSchedule({
            channelId: "test-channel",
            title: "温馨提示",
            priority: "high",
            message: warning[warn],
            date: new Date(Date.now()),
            allowWhileIdle: true,
            repeatType: "day",
            repeatTime: 1,
            vibrate: true,
            vibration: 300,
            actions: ["知道了"],
        });
    }
};

**Programme 2.22** – Function to handle the regular notification for arousing user’s awareness on telephone deception in *App.js*

### 2.10 Simple Questionnaire and Reminder after the Call

After finishing the call, a simple questionnaire to ask if the call contains suspicious content will be pop up as a post-tracking. It is created in a form of push notification since the user may get the message without opening the application and the questionnaire may still be showed even if the application is closed. Again, `PushNotification.localNotificationSchedule()` is used to handle the notification for questionnaire. When the call is hanged, the `endCall(callUUID)` will be performed to end the call, if the call is an unknown call, the questionnaire will be pop up as a push notification, asking the user if the call contains suspicious content, ‘yes’ (是) or ‘no’ (否) action will be available to choose. It is handled with the `handleNotificationUnknown(number)` as shown in Programme 2.23-2.24. If the user chooses ‘yes’, it will be directed to the Report screen of the application, the action is declared in the configuration of `PushNotification` in `index.js` (see Programme 2.25). When Report screen is redirected, the “Phone Number” field of the form will be filled with the number of unknown call that just ended. A dialog box will be pop up for reminding the user to be aware of the phone call (see Programme 2.26).

```javascript
const endCall = ({ callUUID }) => {
    const handle = calls[callUUID];
    log('[endCall] ${format(callUUID)}, number: ${handle}');
    var warn = getInfo(handle);
    if (warn == "不明来电") {
```
handleNotificationUnknown(handle);
} else {
    handleNotification(handle, warn);
}

hangup(callUUID);
removeCall(callUUID);

const hangup = (callUUID) => {
    RNCallKeep.endCall(callUUID);
    log(`[hangup] ${format(callUUID)}`);
}

Programme 2.23 – Functions to end a call in App.js

const handleNotificationUnknown = (number) =>{
    PushNotification.localNotificationSchedule({
        channelId: "test-channel",
        title: "剛才來電("+number+") 有以下可疑內容嗎？",
        priority: "high",
        data: number,
        message: "自稱銀行、政府機構、順豐速遞、匯款",
        date: new Date(Date.now()),
        allowWhileIdle: true,
        vibrate: true,
        vibration: 300,
        actions: ["是", "否"],
        invokeApp: false,
        autoCancel: true
    })
}

Programme 2.24 – Simple questionnaire by PushNotification after the unknown call in App.js

onAction: function (notification) {
    if(notification.action === "是"){
        RootNavigation.navigate('Report', {phone: notification.data, flag: 1});
    }
},

Programme 2.25 – Action declared to navigate to Report screen at PushNotification.configure in index.js

Programme 2.26 – Dialog box to double remind the user in Report.js
In addition, if the incoming call contains data in the database, when the user ended the call, a push notification of reminder will be pop up again, as to double remind the user to be aware of the call just now. It is handled by handleNotification(number, warn) ad shown in Programme 2.27.

```javascript
const handleNotification = (number, warn) => {
    PushNotification.localNotificationSchedule({
        channelId: "test-channel",
        title: "剛才來電("+number+")為"+warn,
        priority: "high",
        message: "請小心提防，如有可疑請盡快尋求協助！",
        date: new Date(Date.now()),
        allowWhileIdle: true,
        vibrate: true,
        vibration: 300,
        actions: ['知道了'],
        autoCancel: false
    })
}
```

**Programme 2.27** – Function to handle the notification for call that contains information in the database after the end of the call in *App.js*

### 2.11 Summary

This chapter introduced the workflow and methodology of the project. The approaches of the IDE and language, UI/UX design as well as the structure of database have been explained. System design including the major features of the project were illustrated with a flow chart (see Fig. 2.5). The implementation detail and technique to be utilized have also been investigated. The next chapter will present the progress over the first semester together with the potential challenges of the project.
3. Progress and Discussion

This chapter focuses on the results of the projects. An elderly-friendly Android application with basic styling, components and functions has been developed. There consists of three major screens with different functions i.e. Search screen, History screen, Detail and Report screen, which will be introduced one by one in section 3.1, 3.2, 3.3 and 3.4 respectively. The function of incoming call screening, simple questionnaire after call, regular reminder together with the call data scrapping and database synchronisation has also been completed, the detail will be presented in section 3.5, 3.6, 3.7 and 3.8. Finally, section 3.9 illustrates the potential challenges that have been faced in this project.

3.1 Search Screen

When the application is entered, the Search screen will first be prompted. As shown in the Fig. 3.1, call searching function which have been mentioned in section 2.7, is now available on the Search screen. It is connected to the Firebase real-time database. A search bar of number input is provided for the user to search for a specific call, the entries that consists the number in the search bar will be displayed in a list inside the scroll view at real-time with phone number together with the corresponding category and organisation.

![Figure 3.1 – Call searching function in Search screen](image-url)
3.2 History Screen

Call log of unknown calls which have been mentioned in section 2.8 is now available in the History screen, it is displayed in a list inside a scroll view (see Fig. 3.2). The call history is extracted with the help of CallLogs module and will be updated at real-time. It displayed only the call from unsaved contact. If there is information of the call in the database, it will be displayed with the corresponding number, category, organization and the time when the call is made. However, if no record is found in the database, only “Unknown Call” together with the number and the time when the call is made will be displayed. For this case, button is provided for the user to report the unknown call, screen will be redirected to the Report screen when the button is clicked, and the number in the form will automatically be passed and filled in (see Fig. 3.3).

![Figure 3.2 – Call log in History screen](image1)

![Figure 3.3 – Redirecting to Report screen when the report button is clicked](image2)
3.3 Detail Screen

When clicking into the entry of the list on the Search screen or the entry of known call in log on History Screen, the detail of the call will be shown with the number, category, and remarks i.e. description of the call. It will be directed to the Detail component in Detail.js. Fig. 3.4 shows the detail screen of number “54444861” which do not contain any remarks, while Fig. 3.5 shows the detail screen of number “29902089” which contains the description of the call, user may scroll down to view the remaining description.

![Figure 3.4 – Detail screen of call without remarks](image1)

![Figure 3.5 – Detail screen of call with remarks](image2)

3.4 Report Screen

As mentioned in section 2.6, users may report a call manually. The call reporting function is available on the Report Screen. A form is provided for user to report the call (see Fig. 3.6(a)). To minimize input error, there are only two mandatory fields, “Phone number” (電話號碼) and “Category” (類別). User are only required to input the phone number, and category with a dropdown picker. Optional fields including “Organization” (機構) and “Remarks” (備註), which “Remarks” represents the additional description of the call are also available. If “Suspected Scams” (疑似詐騙) is chosen in the “Category” field, additional but optional
dropdown list for field “Scam Content” (詐騙內容), and radio button for the field “Whether the call has asked for money transfer” (有沒有要求匯款?) are also available for user to provide additional information to report the scam call, thus further enhancing the call screening in the future (see Fig. 3.7). User may decide whether to input the field other than “Phone Number” and “Category” (see Fig. 3.6(b)). If the user submits the form without inputting either in the “Phone Number” field or “Category” field, a dialog box with a warning message “Please input a phone number and choose a category” (請輸入電話號碼及選擇類別！) will be pop up and the form will not be submitted (see Fig. 3.8). After filling the form, user may click the submit button to submit the form, a dialog box with a message “Successfully reported!” (舉報成功！) will be pop up if the form is successfully submitted, (see Fig. 3.6(c)). The database will also be updated with new reported call immediately. The call information in the History screen will be updated immediately if there is any. User may also search for the new reported call in the Search screen. Additional dropdown fields for choosing the type of business and reasons of scams will be added in the later stage.

---

**Figure 3.6** – Call reporting function in Report screen
3.5 Incoming Call Screening

Incoming call is now being screened to see if it is suspicious. By calling RNCallKeep.displayIncomingCall() with ConnectionService, the function will always be available even when the app is killed [24]. As stated in section 2.5, whenever there is incoming call from unsaved contact, database will be searched to see if there contains any information of the call, the information will be extracted and displayed in the alert if there is any (see Fig. 3.9), and the silent mode will be enabled as it may reduce the risk of handling the call. “Unknown Call” (不明來電) will be displayed in the alert if there is no information found in the database (see Fig. 3.10).
3.6 Simple Questionnaire and Reminder after Call

As mentioned in section 2.10, simple questionnaire will be showed in the form of PushNotification right after the call, to ask if the call contains suspicious information such as claiming to be bank, government officials or SF Express (順豐速運), or asking for money transfer (see Fig. 3.10). These contents are known as popular scam in telephone deception [36]. There are two options in the notification, “Yes” and “No”. If “Yes” is clicked, it will immediately be redirected to Report screen with filling in the “Phone Number” field in the form with the corresponding phone number as shown in Figure 3.11. In addition, a dialog box is displayed to double remind the user to be aware of the call and encourage them to seek help if necessary (see Fig. 3.11(a)). Therefore, increasing the awareness and reducing the risk of falling into telephone deception.
Figure 3.10 – Simple questionnaire to ask if the call contains suspicious information

Figure 3.11 – Redirecting to the Report screen with the corresponding phone number
While if the information of the call is found in the database, a reminder in the form of PushNotification will be displayed instead. The information such as the category and simple description of the call will be displayed together with a note in reminding the user to be aware of the call and encourage them to seek help if necessary (see Fig. 3.12).

Figure 3.12 – A reminder to remind user with information of the call

3.7 Regular Reminder

Daily reminders to educate and raise user’s awareness in preventing telephone deception are created in the form of PushNotification as mentioned in section 2.9. As shown in Figure 3.13, the daily reminder are pop up with “Warm Tips” (溫馨提示) and different messages are delivered to educate them in how to prevent from the telephone deception as well as how to identify scam calls. Thus, avoiding from falling into the telephone deception in the future.
3.8 Call Data Scrapping and Database Synchronisation

The call data is now parsed and scrapped from the JunkCall HK (https://hkjunkcall.com/) and the real-time database in Firebase (https://fyp-slev1-default-rtdb.firebaseio.com/) is now synchronised with the JunkCall HK by the Python function introduced in section 2.3. The data is added according to the number, with the corresponding category, organisation and the description. To prevent from the outdating data, data parsing and scrapping for synchronisation of database will be performed on every three months, thus the data will be up to date to avoid the telephone deception.

Figure 3.13 – Daily reminder to educate user in preventing telephone deception
### 3.9 Potential Challenges and Mitigations

#### 3.9.1 Unable to Scrap Data from HKJunkCall

At the early stage of the project, it is expected to perform web parsing and scrapping with the help of HKJunkCall API https://hkjunkcall.com/?ft={phoneNumber}. A function was built in Python with the library Requests and BeautifulSoup4 respectively as shown in Programme 3.1 below.

```python
from bs4 import BeautifulSoup
import urllib3
import requests.packages.urllib3
requests.packages.urllib3.disable_warnings()
import requests

proxy = '169.57.1.85:8123'
http = requests.packages.urllib3.PoolManager()

from firebase import firebase
key = '{0pB9SnUIpgyRvnG9dupf8feVudk4YGU50HJpyQqA}'
authentication = firebase.FirebaseAuthentication(key, 'llpy1114@connect.hku.hk')
firebase.authentication = authentication
user = authentication.get_user()

phone = "38975362"  # phone number to be checked

url = 'https://hkjunkcall.com/?ft='+phone
response = http.request('GET', url)

soup = BeautifulSoup(response.data)
title = soup.find("meta", attrs={'property': 'og:description'})
s = title["content"]
if(s[0:8]==phone):
    s = s[10:]
    s = s[0:8]
    x = s.split(",")
    dbphone = "/number/" + phone
    firebase.put(dbphone,"category",x[0])
```

Programme 3.1 – Original Python function to perform data parsing and scrapping on HKJunkCall with Requests and BeautifulSoup4

However, it is discovered that the HKJunkCall API is blocking the automate access with captcha after twenty accesses to it. Selenium is therefore considered to replace Requests to parse data as it is parsing tool which control web browsers, perform browser automation and simulate browsing activity, the automation will be less likely to be detected and blocked by
captcha [37]. It is set to browse and parse the website with a Chrome driver, which simulates the browsing activity in Chrome. The code is shown below as Programme 3.2.

```python
import sys
sys.path.insert(0,'/usr/lib/chromium-browser/chromedriver')
from selenium import webdriver
from bs4 import BeautifulSoup

from firebase import firebase
key = '0pB9SUIpdyRvnG9dupf8feVudk4YGU50HJpyQqA'
authentication = firebase.FirebaseAuthentication(key, 'llpy1114@connect.hku.hk')
firebase.authentication = authentication
user = authentication.get_user()

options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome('chromedriver',options=options)
driver.get('https://hkjunkcall.com/')

phone = "38975511" #phone number to be checked

soup = BeautifulSoup(driver.page_source, features="html.parser")

title = soup.find("meta", attrs={'property': 'og:description'})
s = title['content']
if(s[0:8]==phone):
    s = s[10:]
    s = s[0:8]
    x = s.split(",")
    dbphone = "/number/"+ phone
    firebase.put(dbphone,"category",x[0])
driver.close()
```

**Programme 3.2** – Python function to perform data parsing and scrapping on HKJunkCall with Selenium and BeautifulSoup

Unfortunately, captcha is still appeared to block accesses after twenty trials with Selenium. Therefore, JunkCall HK is considered rather than HKJunkCall API. Requests and BeautifulSoup4 are then used to perform web parsing and scrapping on JunkCall HK as mentioned in section 2.3, which efficiently increments and synchronises the database with a huge amount of scam call data.
3.9.2 Long Running Time in Incoming Call Screening

It is originally expected to screen and return the information of the incoming calls every time whenever there is any, which requires to go through database as well as perform web parsing and scrapping from JunkCall HK, if there aren’t any records in the database (see Fig.3.14).

![Flow chart of screening if there aren’t any records in the database](image)

**Figure 3.14** – Flow chart of screening if there aren’t any records in the database

The running time will be long in this case, resulting in an unexpected time spent and delayed response for screening the incoming calls, instant alert may not be available before the incoming calls are handled. To mitigate the issue, the database will be synced with the JunkCall HKs database beforehand, with web parsing and scrapping the information with JunkCall HK by using Python library Requests and BeautifulSoup4. Hence, it is expected that only the database will be read for screening the calls most of the time, the time for running through HKJunkCall API will be minimized. In addition, to keep the database updated, the synchronization will be performed in a regular basis i.e. every three months.

3.9.3 Restriction on Implementing Speech Recognition on Phone Conversation

To enhance the features of existing applications for avoiding telephone deception, it was initially aimed not just to screen out scam calls based on the existing database i.e. HKJunkCall and JunkCall HK. A function of speech recognition for transcribing phone conversation at near real-time was expected to be implemented to screen out suspicious information during the call.

For security purposes, it was planned to be performed on the client side of the application. The phone conversation is expected to be recorded in every 30 seconds by using the SoundRecorder module from ‘react-native-sound-recorder’ package with permission RECORD_AUDIO [38]. Each audio will be passed to perform the speech recognition and transcription by ‘react-native-voice’ package, as it provides accurate transcription in Cantonese [39]. After transcribing the speech into text, conversation scanning against sensitive information e.g. bank account information, will be performed. Warnings will be delivered in the form of notification with sound and vibration when there is sensitive information detected. The entire process is expected to be performed at near real-time during the phone call. After the end of the call, an alert dialog is expected to be pop up to ask if the call just made is suspicious, thus eliminating to store
recording of unsaved known user. If the call is confirmed to be suspicious, the recording of the phone conversation is expected to be kept for three days, and available to be downloaded in the call log in History screen (see section 2.8). The database is expected to be automatically updated as suspected scam with the information scanned during the call if there is any. Otherwise, redirecting to the call reporting system of the application. Figure 3.15 shows the initial system design of the application.

![Diagram of call recording process](image)

**Figure 3.15** – The original system design of the application

Nonetheless, this function is not possible to be implemented as call recording has been strictly forbidden by Android [40]. The phone conversation especially for the voice of the other side are not available for record or access. According to Android, input audio recording follows the rules which the app can capture audio only if it is an accessibility service. The app can capture the voice call if it is a privileged app i.e. pre-installed by Original Equipment Manufacturer on the system, with permission CAPTURE_AUDIO_OUTPUT. Since Android does not allow user to root the devices and make the application a system app, there is no way for the function to be implemented.

To solve the problem, Call Recorder for recording the phone conversation without the other side’s voices is investigated. However, due to the differences of targetSdkVersion in app build.gradle, which targetSdkVersion 22 or below is required [47]. If changing the...
targetSdkVersion of the project, the entire project will not be working as other functions or configuration are built on top of targetSdkVersion 30. Therefore, a function to send out interactive PushNotification for asking the user if the call contains suspicious content is created to filter out the suspicious call (see section 2.10).

3.10 Summary

The chapter presented the development and result of the project, demonstrating the functionality of the application including the call searching function, call log, call reporting function, the function of incoming call screening as well as simple questionnaire after call and the regular reminder. The potential challenges and mitigations have been also covered in the chapter. The future plan together with the project schedule will be illustrated in the next chapter.
4. Future Plan

Although most functions in the application are now ready and implemented, there is still room for improvement. The way and experiments for implementing the call recording feature will still be investigated and conducted, it is hoped to successfully record the voice of the call in the future even though it would only be a one side recording without the voice of the scammer, it would be useful for the elderly to keep an evidence for reporting the phone scam to the police or to review the phone conversation in case there is any suspicious content. Also, if the call recording is available, it would be expected to perform transcription on the recording as planned with using ‘react-native-voice’ package [39]. Button for downloading the recording will be if the phone conversation can be successfully record, the audio is expected to be stored and kept in the devices for three days after the end of the call which may prevent from causing insufficiently memory. In addition, blocking button for blocking suspicious call in the call log is also considered and hoped to be implemented in the future. Researches has been done to look into this feature but there is no solution with the use of React Native, other ways of implementing this feature will be explored in the future. Furthermore, a whitelist feature is also expected to be implement in order to screen out important calls such as call from hospital. As the data that scrapped from the JunkCall HK may not be absolutely correct, a whitelist is needed to prevent elderly from missing the important call. Lastly, the preparation on final presentation and exhibition including the poster design will be done soon.

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestones</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2021</td>
<td><strong>Deliverables of Phrase 1:</strong></td>
</tr>
<tr>
<td></td>
<td>● Detailed Project Plan</td>
</tr>
<tr>
<td></td>
<td>● Project Web Page</td>
</tr>
<tr>
<td></td>
<td><strong>Research:</strong></td>
</tr>
<tr>
<td></td>
<td>● Existing applications and their technologies behind</td>
</tr>
<tr>
<td></td>
<td>● Technology can be used to develop the application</td>
</tr>
<tr>
<td>October 2021</td>
<td><strong>Research:</strong></td>
</tr>
<tr>
<td></td>
<td>● Tools and steps for implementation</td>
</tr>
<tr>
<td></td>
<td>● Usage of the HKJunkCall API</td>
</tr>
<tr>
<td></td>
<td>● UX accessibility for elderly</td>
</tr>
<tr>
<td></td>
<td><strong>Implementation:</strong></td>
</tr>
<tr>
<td></td>
<td>● System design for the app</td>
</tr>
<tr>
<td></td>
<td>● UI/UX design for the app</td>
</tr>
</tbody>
</table>

Table 4.1 - Project schedule
<table>
<thead>
<tr>
<th>Month</th>
<th>Research</th>
<th>Implementation</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 21</td>
<td><strong>Research:</strong> Way to web scrap the information from HKJunkCall API</td>
<td><strong>Implementation:</strong> Database development</td>
<td><strong>Other:</strong> Elderly interview</td>
</tr>
<tr>
<td>December 21</td>
<td><strong>Preparation of Phrase 2:</strong> Interim report, First presentation, Product Demonstration in the first presentation</td>
<td><strong>Research:</strong> The way to connect the app to the database, The way to access call log, The way to detect the incoming call number inside the function</td>
<td></td>
</tr>
<tr>
<td>January 22</td>
<td><strong>Deliverables of Phrase 2:</strong> Interim report, First presentation, Demo in the first presentation</td>
<td><strong>Implementation:</strong> Function of screening the phone call, Function of extracting and updating data to the database, Python function for web parsing and scraping the information with HKJunkCall API</td>
<td></td>
</tr>
<tr>
<td>February 22</td>
<td><strong>Implementation:</strong> Regular notification to remind elderly in avoiding scam calls, Synchronization between our database and the database of JunkCall HK, Adding additional fields e.g. business, reasons of scams to the reporting form</td>
<td><strong>Research:</strong> Performance of transcription and screening on the phone conversation, Way to record the call</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Experiment:</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Trial on call recording and transcription
- Trial to scrap and synchronise data from HKJunkCall

<table>
<thead>
<tr>
<th>Month</th>
<th>Preparations of Phrase 3:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>March 2022</td>
<td><em>Final report</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Final presentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Product Demonstration the final presentation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Testing and debugging</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Implementations:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Screen to show the detailed information of the call</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Simple questionnaire and reminder after the call</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Deploying the app</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Testing of the app</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Bug fixing</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Month</th>
<th>Deliverables of Phrase 3:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2022</td>
<td><em>Final report</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Final presentation</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Product Demonstration the final presentation</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Exhibition and poster design</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Preparation for the project competition (if selected)</em></td>
<td></td>
</tr>
</tbody>
</table>
5. Conclusion

In view of the rising trend of telephone deception happens to the elderly, it is aspired to reduce the number of victims by a mobile application. The project aims to develop an end-to-end solution that is able to screen out the suspicious calls and give out warning to the elderly. Hence, protecting the elderly from falling into the trap of telephone deception and financial loss. In the past semester, research on the technology and approaches for the application development has been done, such as the technology for detecting and screening incoming call as well as approaches to parse and scrap the data from HKJunkCall API and JunkCall HK. In addition, several research and interviews with the elderly regarding their needs on using mobile phone were also done to investigate the needs of the elderly. Essentially, an elderly-friendly Android application, Sieve has been developed with functionalities for screening the incoming call, log tracking for unknown call, call searching and reporting. Detailed technology used and implementation process throughout the entire projects has also been mentioned in previous sections in the report.

Even though most functions of the application have been implemented, improvements and enhancement are still expected to be done in the future. It is expected to figure out a way to record the conversation during the phone call. Thus, the transcription and screening feature on phone conversation can be performed as planned, and user may also download the call recording if needed. Block feature and whitelist feature will also be investigated and implement in the future as to further facilitate and protect the user. In regard to the final phrase, the final presentation with product demonstration and exhibition will be prepared soon. It is hoped that the current features of the application are meeting the objectives and have aroused the elderly’s awareness in preventing from falling into the phone scams. Therefore, helping the elderly in avoiding the telephone deception in the future.
References


[3] “有效攔截推銷電話 芝麻來電等 3 大過濾 App 推介 附識破電騙 8 招”, HK01, 2021. [Online]. Available: https://www.hk01.com/%E5%AF%A6%E6%95%99%E6%8E%A8%E6%94%95%E6%88%AA%E6%8E%A8%E9%8A%B7%E9%BB%E8%A9%B1-%E8%A9%BB-%E9%BA%BB%E4%BE%86%E9%9B%BB%E7%AD%89%E5%A4%A7%E9%81%8E%6%6%BB%BEapp%E6%8E%A8%E4%BB%8B-%E9%99%84%E8%AD%98%E7%A0%B4%E9%9B%BB%E9%A8%98%E6%8B%B9. [Accessed: 15 April 2022].


[11] “【秘技】攔截方法由「直接掛斷」改為「靜音」，可大幅減少推銷來電次數”, Ezone, 2018. [Online]. Available: https://ezone.ulifestyle.com.hk/article/2098042/%E3%80%90%E7%A7%98%E6%8A%80%E3%80%91%E6%94%94%E6%88%AA%E6%96%B9%E6%B3%95%E7%94%B1%E3%80%8C%E7%9B%B4%E6%8E%A5%E6%8E%9B%E6%96%B7%E3%80%8D%E6%94%B9%E7%82%BA%E3%80%8C%E9%9D%9C%E3%80%8D%E5%8F%AF%E5%A4%A7%E5%B9%85%E6%B8%9B%E5%B0%91%E6%8E%A8%E9%8A%B7%E4%BE%86%E9%9B%BB%E6%AC%A1%E6%95%B8. [Accessed: 15 April 2022].


[13] “拒聽急症室 攔截程式各有利弊 網絡專家教路 唔好直接拒聽”, HK01, 2017. [Online]. Available: https://www.hk01.com/%E7%A4%BE%E6%9C%83%E6%96%B0%E8%81%9E/87789/%E6%8B%92%E8%81%BD%E6%80%A5%E7%97%87%E5%AE%A4-%E6%94%94%E6%88%AA%E7%A8%8B%E5%BC%8F%E5%90%84%E6%9C%89%E5


