A Mobile Application for Preventing Telephone Deception

Supervisor
Dr. T.W. Chim

Member
Cheng Tsz Fung 3035684477

Date of Submission: 18/04/2023
Abstract

The pecuniary loss of telephone deception has been rising since 2014. There are existing apps to tackle this problem, but their UI and features have room for improvement. Thus, this project is to develop an Android app with a view to safeguarding Hong Kong people's assets. Regarding features, it executes call screening by querying the Firebase Realtime Database and web-scraping JunkCall HK. Then, it displays the caller information before users accept the call. User’s trusted person will be notified by a SMS message about that call. It also provides a simplified UI for the elderly. Moreover, it will show the latest news about telephone deception. After eight months, Call Guardian and Call Guardian are completed. Most of the planned features are completed, except for call analysis. This report focusses on the implementation, limitation and challenges of the project.
Acknowledgement

I would like to express my deepest appreciation to Dr. T.W. Chim for supervising my final year project, answering my questions and providing guidance.
# Table of Contents

Abstract ...................................................................................................................................... i
Acknowledgement .................................................................................................................... ii
Table of Contents ...................................................................................................................... iii
List of Figures ........................................................................................................................... v
List of Programs ......................................................................................................................... vi
List of Tables ............................................................................................................................. vii
Abbreviation ............................................................................................................................. viii
1. Introduction .......................................................................................................................... 1
   1.1. Background .................................................................................................................... 1
       1.1.1 Types of Telephone deception in Hong Kong ......................................................... 1
       1.1.2 Current Situation of Telephone Deception in Hong Kong ................................. 1
       1.1.3 Existing Mobile App ............................................................................................ 3
   1.2. Project Overview ........................................................................................................... 3
   1.3. Objectives ..................................................................................................................... 4
   1.4. Report Outline .............................................................................................................. 4
2. Methodology .......................................................................................................................... 5
   2.1. IDE and Language ....................................................................................................... 5
   2.2. App Architecture ......................................................................................................... 5
   2.3. App Workflow ............................................................................................................. 7
   2.4. UI and UX ................................................................................................................... 9
   2.5. Web Scraping ............................................................................................................. 11
   2.6. Phone Call Screening ............................................................................................... 13
   2.7. Phone Conversation Analysis .................................................................................. 17
   2.8. Database .................................................................................................................... 18
   2.9. News .......................................................................................................................... 19
   2.10. Report ....................................................................................................................... 23
3. Result ..................................................................................................................................... 26
   3.1. Navigation ................................................................................................................... 26
   3.2. Main Fragment ............................................................................................................ 27
   3.3. News Fragment ........................................................................................................... 28
   3.4. Saved News Fragment .............................................................................................. 29
   3.5. Query Fragment ......................................................................................................... 31
   3.6. Report Fragment ........................................................................................................ 32
   3.7. Article Fragment ........................................................................................................ 34
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8</td>
<td>Display Report Fragment</td>
<td>36</td>
</tr>
<tr>
<td>3.9</td>
<td>View Report Fragment</td>
<td>37</td>
</tr>
<tr>
<td>3.10</td>
<td>Setting Fragment</td>
<td>38</td>
</tr>
<tr>
<td>3.11</td>
<td>Call Screening</td>
<td>39</td>
</tr>
<tr>
<td>3.12</td>
<td>SMS Warning</td>
<td>40</td>
</tr>
<tr>
<td>3.13</td>
<td>Limitation and Mitigation</td>
<td>41</td>
</tr>
<tr>
<td>3.14</td>
<td>Challenges during development</td>
<td>41</td>
</tr>
<tr>
<td>4.</td>
<td>Future Plan</td>
<td>42</td>
</tr>
<tr>
<td>5.</td>
<td>Conclusion</td>
<td>43</td>
</tr>
<tr>
<td>6.</td>
<td>References</td>
<td>44</td>
</tr>
<tr>
<td>7.</td>
<td>Appendices</td>
<td>48</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1.1 Year-to-year comparison in topical frauds figures in Hong Kong [4] ........................................ 1
Figure 1.2 The number of telephone deception in Hong Kong [5-12].......................................................... 2
Figure 1.3 The pecuniary loss of telephone deception in Hong Kong [5-12].................................................. 2
Figure 1.4 Icons of Call Guardian (a) and Call Guardian Elderly (b) ......................................................... 3
Figure 2.1 App Architecture Diagram ........................................................................................................ 5
Figure 2.2 Android Project Folder Structure ............................................................................................... 6
Figure 2.3 Call screening flowchart ............................................................................................................ 8
Figure 2.4 Report query flowchart .............................................................................................................. 8
Figure 2.5 Top App Bar .................................................................................................................................. 9
Figure 2.6 Call Guardian UI ....................................................................................................................... 9
Figure 2.7 Call Guardian Elderly UI ......................................................................................................... 10
Figure 2.8 HTML of a report list on JunkCall HK ....................................................................................... 11
Figure 2.9 HTML and content of a report on JunkCall HK ........................................................................ 11
Figure 2.10 Permission and default app request ......................................................................................... 14
Figure 2.12 Entity Diagram ....................................................................................................................... 18
Figure 3.1 Call Guardian's fragment flow ................................................................................................. 26
Figure 3.2 Call Guardian Elderly’s fragment flow .................................................................................... 26
Figure 3.3 Main Fragment .......................................................................................................................... 27
Figure 3.4 News Fragment ......................................................................................................................... 28
Figure 3.5 Saved News Fragment ............................................................................................................. 29
Figure 3.6 Snack bar to undo remove ....................................................................................................... 30
Figure 3.7 Query Fragment ....................................................................................................................... 31
Figure 3.8 Keyboard for typing phone number ......................................................................................... 31
Figure 3.9 Report Fragment ....................................................................................................................... 32
Figure 3.10 Error when required fields are empty ................................................................................... 33
Figure 3.11 Article Fragment with a saved article .................................................................................... 34
Figure 3.12 Article Fragment with an unsaved article .............................................................................. 35
Figure 3.13 Display Report Fragment .................................................................................................... 36
Figure 3.14 View Report Fragment .......................................................................................................... 37
Figure 3.15 Setting Fragment .................................................................................................................. 38
Figure 3.16 Notification of call screening ................................................................................................. 39
List of Programs

Program 2.1 Function to scrape reports of a given phone number from JunkCall HK .......................... 13
Program 2.2 Necessary permissions for this project features .......................................................... 13
Program 2.3 Function to handle incoming call .................................................................................. 14
Program 2.4 Function to get caller information ............................................................................... 15
Program 2.5 Function to show a pop-up notification ...................................................................... 16
Program 2.6 Function to send a SMS warning ................................................................................. 16
Program 2.7 Function to get news articles from Firebase ................................................................. 19
Program 2.8 Function to collect news articles from getNews() ......................................................... 20
Program 2.9 Observer to handle updated news articles ................................................................. 20
Program 2.10 Functions to interact with local database in UI layer ................................................. 21
Program 2.11 Functions to interact with local database in Data layer ............................................. 21
Program 2.12 Functions to interact with local database in an interface ......................................... 22
Program 2.13 Observer to handle updated saved news articles ...................................................... 22
Program 2.14 Function to get reports from Firebase ..................................................................... 23
Program 2.15 Function to collect reports from getReports() ......................................................... 24
Program 2.16 Observer to handle updated reports ......................................................................... 24
Program 2.17 Function to submit a report in the UI layer ............................................................... 25
Program 2.18 Function to submit a report in the data layer ............................................................ 25
Program 2.19 Observer to handle submission status ...................................................................... 25
List of Tables

Table 7.1 Problem, Difficulty and Solution.................................................................48
Table 7.2 Project Schedule .........................................................................................49
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</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>MVVM</td>
<td>Model – View – ViewModel</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Background

1.1.1 Types of Telephone deception in Hong Kong

In general, scammers fabricate reasons and identities to defraud victims’ money and personal information. The two major types of phone frauds in Hong Kong are “Pretend Officials” and “Guess Who” [1]. In the first case, scammers claim to work for the government and accuse victims of a crime [2]. For instance, they impersonate Mainland public security officers and deceive victims into providing crucial personal information to prove innocence, such as bank account details [2]. In the second case, the scammer asks the victim to guess his/her identity [3]. If the victims reply with a name, the scammer will impersonate and state they had committed crimes in Mainland China [3]. Lastly, they deceive victims into transferring money to them [3].

1.1.2 Current Situation of Telephone Deception in Hong Kong

In figure 1.1, the number of telephone deception was 152, which increased by around 1.6 times. The monetary loss was 41.29 million HKD, which increased by roughly HK$ 6 million. In comparison with other types of frauds, the monetary loss of telephone deception is second-to-last. However, the number of cases is the second highest. By observing the past eight years, the problem of telephone deception is more apparent to the eye.

Figure 1.1 Year-to-year comparison in topical scams’ figures in Hong Kong [4]

The x-axis and y-axis represent the number of cases and types of frauds respectively. This graph displays the number of cases and monetary loss difference of each fraud between June 2021 and June 2022. The dark grey and light grey bar represent the number of cases and monetary loss in June 2021 respectively. The blue and red bar represent the number of cases and monetary loss in June 2022 respectively.
The number of cases has fluctuated during the past eight years (see figure 1.2). Although it is decreasing overall, the pecuniary loss has escalated sharply since 2018 and reached 810 million HKD (see figure 1.3). Thus, telephone deception is still a pressing problem in Hong Kong.

![Number of Telephone Deception](image1.png)

**Figure 1.2** The number of telephone deception in Hong Kong [5-12]

The x-axis and y-axis represent the number of cases and year respectively. This graph display how the number of telephone deception in Hong Kong changed from 2014 to 2021.

![Pecuniary loss of Telephone Deception](image2.png)

**Figure 1.3** The pecuniary loss of telephone deception in Hong Kong [5-12]

The x-axis and y-axis represent the number of cases and year respectively. This graph displays how the pecuniary loss of telephone deception in Hong Kong changed from 2014 to 2021.
1.1.3 Existing Mobile App
The top three phone call filter apps are Call Defender (小熊來電), Headuck Call Blocker (小鴨幹線) and Jima Caller ID (芝麻來電) in Hong Kong [13]. They have more than four pages and each page contains different components. Consequently, it requires a longer time to be familiar with apps, which is problematic for the elderly. Furthermore, Call Defender has advertisements, which can worsen the user experience. They also lack phone conversation analysis and a monitoring feature.

1.2. Project Overview
This project is to develop an Android app (Call Guardian) with a view to safeguarding Hong Kong people asset and ameliorating existing apps. The app only supports phones with Android 10 or above. It was planned to have five major features.

1. Call screening on unknown phone numbers. It will further send an SMS message to notify user trusted person(s).
2. Call analysis on the conversation by speech-to-text to determine whether it is suspicious.
3. Latest news about phone frauds.
4. Making and viewing reports of a given phone number.
5. Two apps with different UI for normal and elderly users respectively.

The third feature is cancelled, due to technical constraints. It will be further explained in chapter 3.

(a) 
(b) 

Figure 1.4 Icons of Call Guardian (a) and Call Guardian Elderly (b)
1.3. Objectives

- To raise users’ alertness when receiving an unknown phone call
- To simplify phone number information checking
- To protect users who are vulnerable to phone scams, such as the elderly
- To educate users what to do when receiving an unknown phone call
- To provide latest news about telephone deception, such as the content and type
- To provide a simple and straight-forward UI for users

1.4. Report Outline

This report has four remaining chapters. Chapter 2 explains approaches and designs for implementing the project. Chapter 3 reports the result of the project, limitation and challenges during development. Chapter 4 outlines the future plan. Chapter 5 summarizes this final report.
2. Methodology

This chapter explains the implementation and technique for this project, such as IDE, UI, and call screening and database. In addition, it gives reasons for choosing specific techniques or approaches.

2.1. IDE and Language

Android Studio is used for the app development, which is the official IDE for Android development. Kotlin is used for the frontend development. In comparison with Java, it is more modern and recommended by Google [14]. Furthermore, Kotlin coroutine is good for managing and executing long-running tasks, such as web scraping [14]. XML is used for building the UI.

2.2. App Architecture

MVVM is adopted for this app. Model holds the app data. View is the app UI. View Model acts as a bridge between view and model and transforms data from model and exposes it to view [15]. Then, views can observe those data and update UI elements if there are changes. This separation enhances the app’s understandability, extensibility and testability.

![Figure 2.1 App Architecture Diagram](image)

The UI layer is responsible for displaying data on the screen. View models hold and provide those data to the UI. Those data come from the data layer. Repository handles the reading and writing of data from Firebase Realtime Database, web scraping from JunkCall HK and local Room database. Then, it exposes data to UI layer.
The adapters code files manage a list of UI elements, namely news articles and reports. The database code files manage the operations with the local database. The models code files are data models. The repository code file manage the operations with data sources. The services code file manages the call screening foreground service. The ui code files are UI logic. The utils code files provide public static functions.

Figure 2.2 Android Project Folder Structure
2.3. App Workflow

Call screening and SMS warning are the primary features to avoid telephone deceptions. Call screening displays caller information and necessary warning to users, so that they can refuse suspicious calls. If they decide to accept the call, they will take extra caution during the conversation. Furthermore, SMS warning notified their trusted person to stop users falling into frauds. Figure 2.3 illustrates the workflow of call screening.

Call screening is only triggered if the incoming phone number is not saved contact. The app will firstly lookup Firebase Realtime Database. If corresponding records exist, the app will get the latest record and display its caller information through a pop-up notification. It will further send an SMS message to notify users’ trusted person.

If there are not any records in Firebase, the app will get caller information from JunkCall HK through web scraping. If JunkCall HK also does not have records, the app will only make a pop-up notification to alert users. After the call, users can make a report regarding the phone number. Then, other users can view a more updated record.

Apart from the above features, this project also allows users to query reports of a phone number (see figure 2.4). After users input a phone number, the app will firstly lookup the Firebase Realtime Database. If records do not exist, it will get reports from JunkCall HK through web scraping. Users can also press open in browser button to view reports of same phone number in JunkCall HK. If there are not any reports, it will tell users.
Figure 2.3 Call screening flowchart

Figure 2.4 Report query flowchart
2.4. UI and UX

Figma is used for UI and app icon design, which is a free web application for designing app UI and UX [16]. Furthermore, Call Guardian features the latest material design: Material 3 to provide better UI and UX.

Call Guardian has four main fragments, namely News, Saved News, Query and Report. Bottom navigation bars are implemented for navigating between them quickly (see figure 2.6). Each destination is represented by an icon and a text label, which help users find their desired destination quickly. The bold label and an orange pill indicate the current location. It is also ergonomic because users can easily switch destinations with their thumbs.

Furthermore, there are four more fragments, namely article, display report, view report and setting. The top app bar’s back button and Android native navigation bar enable users to go back to the previous fragment (see figure 2.5). The top app bar displays the current fragment name as title. Moreover, it provides relevant functions based on the current fragment (see figure 2.5).

![Figure 2.5 Top App Bar](image)

This is article fragment’s app bar, which has a back button, a title, open in browser and share button.

![Figure 2.6 Call Guardian UI](image)

(a) (b) (c) (d)
Call Guardian Elderly is for elderly users (see figure 2.7). Research has been done to design an elderly-friendly UI. Font size and UI components are bigger to cater for their weakened vision. Font size is at least 16 px [17, 18]. The line spacing is 1.5 [19]. Furthermore, the color contrast between text and background is no less than 4.5:1 [19]. Apart from text, clickable components are at least 11 mm diagonally long and 2 mm away of other components [20].

![Figure 2.7 Call Guardian Elderly UI](image)
2.5. Web Scraping

Skrape{it} is used for web scraping and data parsing, which is a Kotlin-based web scraping library [21]. The HK Junk Call database is no longer open to the public. Therefore, web scraping is chosen for retrieving phone number information and reports. JunkCall HK is simpler than the HK Junk Call webpage in terms of structure and number of elements. Hence, it reduces web scraping time cost. Since Firebase Realtime Database lacks records, the app relies on JunkCall HK for obtaining information about phone numbers.

Reports of a given phone number is at https://www.junk-call.com/hk/[phone_no]. Those reports are inside HTML ol tag with class name “junkcall” (see figure 2.8). The report content is inside several div elements (see figure 2.8). Luckily, most of the div elements have a class name, which makes web scraping easier.

Figure 2.8 HTML of a report list on JunkCall HK

Figure 2.9 HTML and content of a report on JunkCall HK

Program 2.1 makes a HTTP request to that URL and analysis response’s HTML document. HTML elements can be picked inside htmlDocument{} by their tags and class names. If there are not any reports on that URL, this function will return an empty list.
fun scrapeReports(phone_no: String): MutableList<RetrievedReport> {
    val reportList = mutableListOf<RetrievedReport>()
    var id = 1
    var noData = false

    skrape(HttpFetcher) {
        request {
            url = "$JUNKCALLHK_URL/$phone_no"
            userAgent = USER_AGENT
        }

        response {
            htmlDocument {
                relaxed = true
                div {
                    withClass = "intro"
                    findAll {
                        if (isNotEmpty()) {
                            noData = true
                        }
                    }
                }
                if (noData) {
                    return@htmlDocument
                }
                ol {
                    withClass = "junkcall"
                    findFirst {
                        li {
                            findAll {
                                forEach { li ->
                                    var type = ""
                                    var caller = ""
                                    var description = ""
                                    li.div {
                                        withClass = "cat2"
                                        findFirst {
                                            type = text
                                        }
                                    }
                                    li.div {
                                        withClass = "remark"
                                        findFirst {
                                            description = text
                                        }
                                    }
                                    li.a {
                                        findAll {
                                            val len = eachText.size
                                            if (len > 3) {
                                                caller = eachText[len - 2] + "(" + eachText[len - 1] + ")"
                                            } else {
                                                caller = ""
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
2.6. Phone Call Screening

It is a real-time mechanism which displays the caller information before accepting the call [22]. When a user receives an unknown phone call, this app will conduct caller identification by querying the database and JunkCall HK with the given phone number. Next, it will display the caller information. However, this app relies on JunkCall HK as mentioned in chapter 2.3. Internet connection is required to do the call screening.

When the app launches, it will request permission and being default caller ID and spam app (see figure 2.10). Users must accept them in order to enable features in chapter 1.2.

Program 2.1 Function to scrape reports of a given phone number from JunkCall HK

```
if (type.isNotEmpty()) {
    reportList.add(
        RetrievedReport(
            id.toString(),
            type,
            caller,
            description,
            -1
        )
    )
    id += 1
}
```
Call screening runs in the background and activates when the phone receives incoming call. To achieve this, this project implements the abstract class CallScreeningService provided by Google. The onScreenCall function needs to be overridden to implement more features, namely caller identification, notification and SMS warning (see program 2.3). These features take time to finish and should be run in another thread, otherwise the system call app will be unresponsive. Therefore, coroutine is used to run those asynchronous tasks: scope.launch {}.

```kotlin
override fun onScreenCall(callDetails: Call.Details) {
    val isCallScreeningEnable = sharedPreference.getBoolean("call_screening", false)
    if (isCallScreeningEnable && callDetails.callDirection == Call.Details.DIRECTION_INCOMING) {
        val phoneNo = getPhoneNo(callDetails)
        scope.launch {
            val callerInfoResponse = CallIdentificationUtils.getCallerInfo(phoneNo)
            showNotification(phoneNo, callerInfoResponse)
            if (sharedPreference.getBoolean("monitor", false)) {
                sendSmsMessage(phoneNo, callerInfoResponse)
            }
        }
    }
    val response = CallResponse.Builder().respondToCall(callDetails, response.build())
}
```

Program 2.3 Function to handle incoming call
The `getCallerInfo()` is called for getting the latest record of incoming phone number by using `orderByChild("timestamp")` and `last()` (see program 2.4). Records are sorted from the oldest to most recent, so the latest record is the last element.

This function will firstly lookup records in Firebase Realtime Database. If there are no records, it will get records from JunkCall HK through `scrapeReport()`. If JunkCall HK also has not corresponding reports, it will set `isSuccessful` attributes of `CallerInfoResponse` to false.

```kotlin
suspend fun getCallerInfo(phoneNo: String): CallerInfoResponse {
    val database = FirebaseDatabase.getInstance().reference
    val reportRef = database.child("report").child(phoneNo)
    var callerInfo: CallerInfoResponse

    try {
        val dataSnapshot = reportRef.orderByChild("timestamp").get().await().children.last()
        val type = dataSnapshot.child("type").value as String
        val caller = dataSnapshot.child("caller").value as String
        callerInfo = CallerInfoResponse(true, CallerInfo(type, caller))
    } catch (exception: Exception) {
        callerInfo = scrapeReport(phoneNo)
    }

    return callerInfo
}
```

**Program 2.4 Function to get caller information**

After `getCallerInfo()` is finished, `showNotification()` will be called to show a pop-up notification with caller information. If caller information is unavailable, it will make a pop-up notification with text “沒有資料，請提高警覺!”. 


private fun showNotification(phoneNo: String, callerInfoResponse: CallerInfoResponse) {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.POST_NOTIFICATIONS) == PackageManager.PERMISSION_GRANTED) {
        val title = "$phoneNo 資料"
        var content = getString(R.string.no_record_error) + getString(R.string.reminder)
        if (callerInfoResponse.isSuccessful) {
            content = callerInfoResponse.callerInfo.type + "\n" + callerInfoResponse.callerInfo.caller
        }
        val notification = NotificationCompat.Builder(this, CHANNEL_ID)
            .setSmallIcon(R.drawable.icon_new_release)
            .setContentTitle(title)
            .setStyle(NotificationCompat.BigTextStyle().bigText(content))
            .setDefaults(NotificationCompat.DEFAULT_ALL)
            .setPriority(NotificationCompat.PRIORITY_MAX)
            .build()
        with(NotificationManagerCompat.from(this)) {
            notify(NOTIFICATION_ID, notification)
        }
    }
}

private fun sendSmsMessage(phoneNo: String, callerInfoResponse: CallerInfoResponse) {
    val relation = sharedPreferences.getString("relation", "")
    val receiverPhoneNo = sharedPreferences.getString("monitor_phone_number", "")
    if (relation != null && receiverPhoneNo != null) {
        if (relation.isNotEmpty() && receiverPhoneNo.isNotEmpty()) {
            val current = LocalDateTime.now()
            val formatter = DateTimeFormatter.ofPattern("HH:mm")
            val currentTime = current.format(formatter)
            val url = "https://www.junk-call.com/hk/$phoneNo"
            val msg = "你的" + relation + "在" + currentTime + "收到了" + phoneNo + "\n" + "類型:" + callerInfoResponse.callerInfo.type + "\n" + "致電者:" + callerInfoResponse.callerInfo.caller + "\n" + "更多資訊:" + url
            val smsManager = this.getSystemService(SmsManager::class.java)
            val smsArray = smsManager.divideMessage(msg)
            smsManager.sendMultipartTextMessage(receiverPhoneNo, null, smsArray, null, null)
        }
    }
}

Program 2.5 Function to show a pop-up notification

Program 2.6 Function to send a SMS warning
2.7. Phone Conversation Analysis

After the speech-to-text conversion, this app will identify any suspicious keywords, such as claiming to be a government officer. It will warn users if it is suspicious.

There are two methods. First, a real-time speech-to-text by Speech Recognizer, which enables the app access to the Android built-in speech recognition service [23]. Another method is conversation recording and transcription. For instance, Google speech-to-text API transcribes mp3 file and returns a text file back to the app [24].

The first method is free and already available on Android devices. Furthermore, it can be executed without the Internet connection if users downloaded the offline speech recognition package.

However, Speech Recognizer is tested and cannot read audio input during phone call. Besides, Google announced third-party apps cannot record phone calls by using its Accessibility API. Both, first and second method are unfeasible. Consequently, this feature is cancelled.
2.8. Database

The Firebase Realtime Database is used for storing users' reports and news about telephone deception. It is a cloud database and stores data as JSON [25]. Furthermore, data is synchronized in real time, which means users automatically receive new phone number reports and news [25]. News articles are input by the developer, while users or developers input reports. There are two paths for reading and writing data: news and report.

There are two entity sets, news and report. Figure 2.4 illustrates attributes of news and report respectively. All attributes are string except for timestamp being long. Firebase Realtime Database only supports string, long, double, Boolean, map and list, so datetime object cannot be used. Although time can be saved as string, records cannot be sorted chronologically by using orderByChild(). Therefore, a timestamp is chosen for storing datetime, which counts number of seconds that have elapsed since 00:00:00 UTC on 1 January 1970.

<table>
<thead>
<tr>
<th>news</th>
<th>report</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>phone_no</td>
</tr>
<tr>
<td>title</td>
<td>type</td>
</tr>
<tr>
<td>page_url</td>
<td>caller</td>
</tr>
<tr>
<td>image_url</td>
<td>description</td>
</tr>
<tr>
<td>description</td>
<td>timestamp</td>
</tr>
<tr>
<td>advice</td>
<td></td>
</tr>
<tr>
<td>timestamp</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.11 Entity Diagram

The primary keys of news and report are id and phone_no respectively. All attributes are string except for timestamp being long.

Room is chosen for building a local database to save news articles. The Room persistence library provides an interface to access SQLite database and enable different database operations [26].
2.9. News

The source of news article is Anti-Deception Coordination Centre, which is set up by the Hong Kong Police Force, aims to fight against deception and increase public awareness of various frauds [27]. Hence, its news articles are credible and up to date. News articles are input by developers on Firebase Realtime Database.

Program 2.7 shows the definition of getNews() in mainRepository.kt. It firstly gets a reference to the news table in which records are ordered by timestamp ascendingly. It needs to be reversed later to display the latest news article first. The ValueEventListener is added to listen for updates to the news table. When data is updated, it will return a flow of NewsResponse objects through trySend() statements. awaitClose {} is for detaching the listener when the flow is canceled. getNews() is a data emitter, there is another function to collect its emitted data.

```
fun getNews(): Flow<NewsResponse> {
    return callbackFlow {
        val newsRef = database.child("news").orderByChild("timestamp")
        var articleList: List<Article> = 
        val listener = newsRef.addValueEventListener(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    articleList = snapshot.children.map { dataSnapshot -> dataSnapshot.getValue(Article::class.java)!! } reversed()
                    trySend(NewsResponse(true, "", articleList))
                } else {
                    trySend(NewsResponse(false, "not exist"))
                }
            }

            override fun onCancelled(error: DatabaseError) {
                trySend(NewsResponse(false, error.toString()))
            }
        })
        awaitClose {
            newsRef.removeEventListener(listener)
        }
    }
}
```

Program 2.7 Function to get news articles from Firebase
Program 2.8 shows the definition of `getNews()` in `MainViewModel.kt`, which collects emitted `NewsResponse` and updates the value of `news`. `news` is a `MutableLiveData` object. There is an observer in the fragment to listen for changes of `news`. When `news` is updated, it will update the UI.

```
val news: MutableLiveData<Resource<NewsResponse>> = MutableLiveData()

private fun getNews() {
    viewModelScope.launch {
        news.postValue(Resource.Loading())
        withContext(Dispatchers.IO) {
            mainRepository.getNews().collect { newsResponse ->
                news.postValue(handleNewsResponse(newsResponse))
            }
        }
    }
}
```

**Program 2.8 Function to collect news articles from getNews()**

Program 2.9 shows the logic of observer in `NewsFragment.kt`. If data is not null, it extracts a news article list and passes it to the `newsAdapter`. The `newsAdapter` will update the recycler view to display a new article list.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    initViews()

    viewModel.news.observe(viewLifecycleOwner) { response ->
        if (response is Resource.Success) {
            response.data?.let { newsResponse ->
                newsAdapter.differ.submitList(newsResponse.articleList)
            }
        } else if (response is Resource.Failure) {
            response.message?.let { message ->
                Log.e(TAG, message)
            }
        }
    }
}
```

**Program 2.9 Observer to handle updated news articles**

In summary, `MainRepository` communicates with `MainViewModel` through Kotlin Flow. `MainViewModel` communicates with `NewsFragment` through LiveData. Data can be transferred from data layer to UI layer smartly.
Furthermore, users can save or remove saved news articles. The reading and writing operations are synchronous tasks, so coroutine is used again (see program 2.10). Actual SQL is written in program 2.12.

```kotlin
val savedNews = MutableLiveData<List<Article>>()

fun saveArticle(article: Article) {
    viewModelScope.launch {
        mainRepository.insertArticle(article)
    }
}

fun checkArticleSaved(article: Article) {
    viewModelScope.launch {
        val isSaved = mainRepository.checkArticleSaved(article)
        isArticleSaved.postValue(isSaved)
    }
}

fun getSavedNews() {
    viewModelScope.launch {
        savedNews.postValue(mainRepository.getSavedNews())
    }
}

fun unsaveArticle(article: Article) {
    viewModelScope.launch {
        mainRepository.deleteArticle(article)
    }
}
```

Program 2.10 Functions to interact with local database in UI layer

```kotlin
suspend fun insertArticle(article: Article) {
    localDB.getArticleDao().insertArticle(article)
}

suspend fun checkArticleSaved(article: Article): Boolean {
    var isSaved = false
    article.page_url?.let {
        val result = localDB.getArticleDao().isArticleSaved(article.page_url)
        if (result.isNotEmpty()) {
            isSaved = true
        }
    }
    return isSaved
}

suspend fun getSavedNews() =
    localDB.getArticleDao().getAllArticles()

suspend fun deleteArticle(article: Article) {
    article.page_url?.let {
        localDB.getArticleDao().deleteArticle(article.page_url)
    }
}
```

Program 2.11 Functions to interact with local database in Data layer
The `onSwiped()` function enables users to remove saved new articles by swiping that card from left to right (see program 2.13). The observer will submit the updated saved news article list through `submitList()`. Then, the Saved News Fragment will display a new list of saved news articles.

```kotlin
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    initViews()

    val itemTouchHelperCallback = object : ItemTouchHelper.SimpleCallback{
        ItemTouchHelper.UP or ItemTouchHelper.DOWN,
        ItemTouchHelper.LEFT or ItemTouchHelper.RIGHT
    }

    override fun onMove(recyclerView: RecyclerView,
        viewHolder: RecyclerView.ViewHolder,
        target: RecyclerView.ViewHolder
    ): Boolean {
        return true
    }

    override fun onSwiped(viewHolder: RecyclerView.ViewHolder,
        direction: Int) {
        val position = viewHolder.adapterPosition
        val article = newsAdapter.differ.currentList[position]

        viewModel.unsaveArticle(article)
        showSnackBar(article)
    }

    ItemTouchHelper(itemTouchHelperCallback).apply {
        attachToRecyclerView(binding.newsRecView)
    }

    viewModel.getSavedNews()
    viewModel.savedNews.observe(viewLifecycleOwner) { articles ->
        newsAdapter.differ.submitList(articles)
    }
}
```

Program 2.13 Observer to handle updated saved news articles
2.10. Report

Users can view and create reports. Fetching reports is like fetching news articles. The `getReports()` function is defined in `MainRepository.kt`. The `trySend()` function emits updated report lists.

```kotlin
val report: MutableLiveData<Resource<ReportResponse>> = MutableLiveData()

suspend fun getReports(phone_no: String): Flow<ReportResponse> {
    return callbackFlow {
        val reportRef = database.child("report").child(phone_no).orderByChild("timestamp")
        var reportList = mutableListOf<RetrievedReport>()
        val listener = reportRef.addValueEventListener(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    for (dataSnapshot in snapshot.children) {
                        val id = dataSnapshot.key
                        val type = dataSnapshot.child("type").value as String
                        val caller = dataSnapshot.child("caller").value as String
                        val description = dataSnapshot.child("description").value as String
                        val timestamp = dataSnapshot.child("timestamp").value as Long
                        val report = RetrievedReport(id, type, caller, description, timestamp)
                        reportList.add(report)
                    }
                    reportList = reportList.reversed() as MutableList<RetrievedReport>
                    trySend(ReportResponse(true, "", reportList))
                } else {
                    reportList = scrapeReports(phone_no)
                    if (reportList.isEmpty()) {
                        trySend(ReportResponse(true, "not exist", reportList))
                    } else {
                        trySend(ReportResponse(true, "", reportList))
                    }
                }
            }
            override fun onCancelled(error: DatabaseError) {
                trySend(ReportResponse(false, error.toString()))
            }
        })
        awaitClose {
            reportRef.removeEventListener(listener)
        }
    }
}
```

Program 2.14 Function to get reports from Firebase

Program 2.15 shows the definition of `getReports()` in `MainViewModel.kt`. It collects emitted `ReportResponse` and updates the value of `report`. `report` is a `MutableLiveData`
object. There is an observer in the fragment to listen for changes of the report livedata. When the report livedata is updated, it will update the UI.

```kotlin
fun getReport(phoneNo: String) {
    viewModelScope.launch {
        report.postValue(Resource.Loading())
        withContext(Dispatchers.IO) {
            mainRepository.getReports(phoneNo).collect() { reportResponse ->
                report.postValue(handleReportResponse(reportResponse))
            }
        }
    }
}
```

**Program 2.15 Function to collect reports from getReports()**

Program 2.16 shows the logic of observer in DisplayReportFragment.kt. If data is not null, it extracts a report list and passes it to the reportAdapter. The report adapter will update the recycler view to display an updated report list.

```kotlin
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    initViews()

    viewModel.getReport(args.phoneNo)
    viewModel.report.observe(viewLifecycleOwner) { response ->
        when (response) {
            is Resource.Success -> {
                binding.loadingPI.visibility = View.GONE
                response.data?.let { reportResponse ->
                    reportResponse.reportList?.let {
                        if (it.isEmpty()) {
                            binding.noRecordTv.visibility = View.VISIBLE
                        } else {
                            binding.noRecordTv.visibility = View.GONE
                            reportAdapter.differ.submitList(it)
                        }
                    }
                }
            }
            is Resource.Failure -> {
                binding.loadingPI.visibility = View.GONE
                response.message?.let { message ->
                    Log.e(TAG, message)
                }
            }
            is Resource.Loading -> {
                binding.loadingPI.visibility = View.VISIBLE
            }
        }
    }
}
```

**Program 2.16 Observer to handle updated reports**
Apart from viewing reports, users can create and submit a report, which is like reading reports. There are four fields, namely phone number, type, caller identity and description. Users cannot submit the report if the Internet is unavailable. After the app submits the report (see program 2.17 and 2.18), the value of reportSubmission is updated. The observer in ReportFragment.kt detects the change of submission status and checks its value (see program 2.19). If it is successful, it will notify users through a snack bar. Otherwise, it will let users retry.

```kotlin
val reportSubmission = MutableLiveData<ReportSubmission>()

fun submitReport(phone_no: String, report: Report) {
    reportPhoneNo = phone_no
    val submission = mainRepository.submitReport(reportPhoneNo, report)
    reportSubmission.value = submission
}
```

**Program 2.17 Function to submit a report in the UI layer**

```kotlin
    val reportRef = database.child("report")
    try {
        reportRef.child(phone_no).push().setValue(report)
    } catch (exception: Exception) {
        return ReportSubmission(false, exception.toString(), report)
    }
    return ReportSubmission(true, "", report)
}
```

**Program 2.18 Function to submit a report in the data layer**

```kotlin
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    viewModel.reportSubmission.observe(viewLifecycleOwner) { submissionReport ->
        if (submissionReport.isSuccessful) {
            Snackbar.make(requireView(), R.string.firebase_insert_success, Snackbar.LENGTH_SHORT).show()
        } else {
            Snackbar.make(requireView(), R.string.firebase_insert_error, Snackbar.LENGTH_SHORT)
                .setAction(R.string.retry) {
                    viewModel.submitReport(viewModel.reportPhoneNo, submissionReport.report)
                }.show()
        }
    }
}
```

**Program 2.19 Observer to handle submission status**
3. Result

This chapter reports the result of this project. Call Guardian and Call Guardian Elderly fragments will be described and compared. Then, the actual operation of call screening and SMS warning will be demonstrated. Lastly, limitation and mitigation, and challenges during development will be reported.

3.1. Navigation

Figures 3.1 and 3.2 illustrate the transition between fragments. Call Guardian enables users to switch between News, Saved News, Query and Report Fragment through bottom navigation bar. The top app bar’s setting button redirects users to the Setting Fragment.

![Figure 3.1 Call Guardian's fragment flow](image1)

![Figure 3.2 Call Guardian Elderly’s fragment flow](image2)
3.2. Main Fragment

Only Call Guardian Elderly has this fragment because bottom navigation bar’s text and icon are not big enough for the elderly. Apart from bigger icons and text, each destination is labeled with distinct colors (see figure 3.3). It helps users distinguish different destinations more quickly. It acts as a home page and allows users to navigate to other major fragments, namely News, Saved News, Query, Report and Setting fragments. These fragments have a back button at the top left corner to return to the Main Fragment.

![Main Fragment](image)

*Figure 3.3 Main Fragment*
3.3. News Fragment

This fragment displays a list of news, which is fetched from the Firebase Realtime Database (see figure 3.4). The onDataChange() function in ValueEventListener will be triggered if there is any updated news (see program 2.6). Each news article can be saved or shared through various social apps, such as WhatsApp, Instagram and Gmail.

When users press the card (orange or blue rectangle), the app will navigate to the Article Fragment to display the corresponding news article.

Call Guardian Elderly’s News Fragment has bigger text (see figure 3.4(b)). Buttons also have bigger text and distinct background to help the elderly find buttons. Besides, open in browser action is available, whereas Call Guardian only has the share function. There is not enough space to have the open in browser button in the Article Fragment (see figure 3.11), so this button is moved out from the Article Fragment.

Figure 3.4 News Fragment
3.4. Saved News Fragment

This fragment contains all the saved news articles (see figure 3.5). The top one is the most recently saved. If users save or remove news articles, the UI will display a new saved news article list. Users can remove articles by swiping the card from left to right. They can undo it by pressing the undo button in the snack bar (see figure 3.6).

![Figure 3.5 Saved News Fragment](a) ![Figure 3.5 Saved News Fragment](b)
Figure 3.6 Snack bar to undo remove

(a)  

(b)
3.5. Query Fragment

This fragment has a text field for reading phone numbers (see figure 3.7). It is set to read phone numbers only, so it can reduce typing errors and simplify data validation (see figure 3.8). After the button is clicked, the app will navigate to the Display Report Fragment. It will firstly search record in the Firebase Realtime Database. If there is no record, it will get record from JunkCall HK by web scraping.

Call Guardian Elderly has bigger text and button (see figure 3.7(b)). Furthermore, its top bar has a button to navigate back to the Main Fragment. Buttons in the top app bar are as few as possible because they only have icons without any text. The size is also small, so the elderly will find it difficult to use.

![Figure 3.7 Query Fragment](a)

![Figure 3.8 Keyboard for typing phone number](b)
3.6. Report Fragment

This fragment has four text fields for reading phone number, type, caller and description (see figure 3.9). Type is predefined: 不明來電 (unknown call), 財務借貸 (loan), 推銷 (sell), 詐騙 (scam) and 滋擾 (nuisance). Users choose one of them from the dropdown menu. Description has a word limit of 100.

Call Guardian Elderly’s Report Fragment has bigger text and button (see figure 3.9(b)).

![Figure 3.9 Report Fragment](image)
All fields are mandatory except for caller. If the required text fields are not completed, errors will pop up to remind users to input again (see figure 3.10). A snack bar will appear to inform users there is no Internet connection. If the submission is successful, a snack bar will appear and notify users. Otherwise, it will report failure and let users resend again.

Figure 3.10 Error when required fields are empty
3.7. Article Fragment

This fragment displays the content of a news article, namely title, data, image, fraud description and advice for handling it (see figure 3.11). The top app bar provides two different actions on the right side: open in browser and share. It also has a navigate back button on the left side for returning to the News Fragment.

There is a floating action button at the bottom right corner. Call Guardian Elderly’s floating action button has an extra label to show whether the current article is saved or not. Users can click it to save or unsave articles. As mentioned in chapter 3.2, open in browser and share buttons are moved to News Fragment and Saved News Fragment.

The open in browser action will open the default browser to the website of the news article. The back button lets users navigate back to News or Saved News Fragment.

(a) Figure 3.11 Article Fragment with a saved article
If the current article is unsaved, the floating action button will have a different appearance (see figure 3.12).

Figure 3.12 Article Fragment with an unsaved article
3.8. Display Report Fragment

This fragment displays reports of the phone number queried in Query fragment (see figure 3.13). Each report contains type, date, caller and description. The top app bar provides an open in browser action for visiting the JunkCall HK page of that phone number. It also provides a navigate back button on the left side for returning to Query fragment. In Call Guardian Elderly, the open in browser action is at the bottom right corner. Title and text are also bigger than normal version.

The open in browser action will open default browser to https://www.junk-call.com/hk/{phone_no}.

When users press the card, the app will navigate to View Report Fragment to see the report’s whole content. The back button lets users navigate back to the Query Fragment.

Figure 3.13 Display Report Fragment
3.9. View Report Fragment

This fragment displays the content of a report, namely time, type, caller identification and description (see figure 3.14). Call Guardian Elderly have a bigger title and time label. The back button lets users navigate back to the Display Report Fragment.

Figure 3.14 View Report Fragment
### 3.10. Setting Fragment

Users can switch on or off call screening and SMS warning in this fragment (see figure 3.15). If the monitoring feature is switched on, users can input relation with recipient and his/her phone number. Otherwise, users cannot modify those fields, which is grey.
3.11. Call Screening

Any incoming call with unsave contact will be examined. If Call Guardian is closed, the Call Screening Service will still run in the background. As explained in chapter 2.6, the service will firstly lookup the latest record of that income phone number in Firebase Realtime Database. If there is no record, it will get the report from JunkCall HK through Skrape{it} library. The caller identity and call type will be shown in a pop-up notification. Furthermore, a reminder will be given instead if both data sources have no corresponding record.

![Figure 3.16 Notification of call screening](image)
3.12. SMS Warning

If users switch on monitoring in the Setting Fragment and those two fields are not empty, sendSmsMessage() will be called to send an SMS message to users’ trusted person. It states the time, caller information and call type of that call. For instance,

你的兒子在 10/04/2023 13:00 收到了 95507176

類型: 詐騙

致電者: 假冒衛生署

更多資訊: https://www.junk-call.com/hk/95507176
3.13. Limitation and Mitigation

Call analysis is cancelled, so the conversation cannot be analyzed. Call analysis can prevent callee fall into frauds when the corresponding record is unavailable. Users can know whether the call is risky based on the result of call analysis. The latest news and monitoring feature can mitigate the impact of this limitation. The former teaches users notice suspicious things during the conversation. The latter can stop users falling into fraud through their trusted person.

Firebase Realtime database lacks phone number reports. Call screening and phone number query primarily rely on web-scraping JunkCall HK, which requires Internet connection. In the future, requests for accessing Junk Call HK or HK Junk Call database will be made. The app would download the database, so as to run call screening and phone number query without Internet connection.

3.14. Challenges during development

Overall, there are few challenges because of sufficient research in the first phrase. Nonetheless, there are two major challenges.

First, it is difficult to design a UI for the elderly. This project uses the latest Material Design to make the UI. The official website gives guidelines for each UI components on how to use and customize them properly. However, those guidelines are for general users. Extra research and testing must be done to ensure the app is elderly friendly. For example, Call Guardian Elderly was planned to have a bottom navigation bar. Each destination is not far enough away from the other. The size of text and icon are limited.

Second, it is difficult to design the communication between the UI and data layer. Many operations are not finished instantly, such as fetching news and reports. These operations cannot run on the main thread, otherwise the app will become unresponsive. Therefore, coroutine is used which is like threading. As explained in chapter 2.9, Live Data and Kotlin Flow are used for the communication between layers. There are also similar techniques, such as Shared Flow and State Flow. It is crucial to understand their differences to choose the proper techniques.
4. **Future Plan**

The final app has most of the planned features. However, there are still areas for continued development and improvement. Call screening could run offline, so that the phone does not need to connect to the Internet all the time. Furthermore, the implementation of call analysis will still be studied and experimented. Besides, call recording will also be considered. Recording can be used as a sample or evidence for reporting telephone deception.

Moreover, user feedback will be monitored after the app is released. They are valuable for improving UI and features. They may also provide innovative ideas for new features. Lastly, IOS version of Call Guardian will be considered based on the effectiveness and popularity of the Android app.

Tables 7.2 illustrate the project schedule from the first to final stage.
5. Conclusion

Despite the overall decline of telephone deception, it is still imperative to tackle telephone deception actively. The downsides of existing apps are complex UI and lacking conversation analysis. Therefore, this project aims to develop an app with better UI, UX and features to prevent telephone deception. Comprehensive research and prototype of web scraper are made to facilitate the development. The detailed implementation is explained with source code and chart in this report.

Two major limitations are the lack of call analysis and call screening requires Internet connection. The latest news and monitoring can mitigate the impact of limitation. Furthermore, two major challenges are designing elderly-friendly UI and communication between layers.

Call analysis and recording will be experimented again. The former is important when relevant phone number records are unavailable. The latter can be used as evidence for reporting frauds. After the app is released, user feedback will be listened to improve the app. Lastly, IOS version of Call Guardian will be considered based on the popularity and effectiveness of the Android counterpart.
6. References


### 7. Appendices

<table>
<thead>
<tr>
<th>Description</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technically Difficulty</strong></td>
<td>• Give warning instead of blocking</td>
</tr>
<tr>
<td>Distinguishing phone calls between real</td>
<td>• Query trusted website to get caller information</td>
</tr>
<tr>
<td>organization and scammers. They may share the</td>
<td>• Analyze conversation to determine whether it is suspicious</td>
</tr>
<tr>
<td>same phone number pattern, such as start with</td>
<td></td>
</tr>
<tr>
<td>number three</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Technically Difficulty</strong></td>
<td>• Study related documentation</td>
</tr>
<tr>
<td>Running Speech Recognizer in the background</td>
<td>• Find any workable solutions online</td>
</tr>
<tr>
<td>during phone call</td>
<td>• Test Speech Recognizer during phone call at the early stage</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Privacy Problem</strong></td>
<td>• Seek consent from users when they use the app initially</td>
</tr>
<tr>
<td>Accessing sensitive data such as user phone call</td>
<td>• Do not upload any call logs and phone conversations</td>
</tr>
<tr>
<td>logs and content of phone conversation</td>
<td>• Allow users to enable phone conversation analysis on unknown callers only</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Database Problem</strong></td>
<td>• Update the database when users check phone numbers which only exist in</td>
</tr>
<tr>
<td>Lacking phone number records in the Firebase</td>
<td>JunkCall HK</td>
</tr>
<tr>
<td>Realtime Database</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Table 7.1 Problem, Difficulty and Solution       |

This table not only describes difficulties and problems encountered, but also includes the corresponding solutions.
<table>
<thead>
<tr>
<th>Date</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 2022</td>
<td>Deliverables of Phrase 1:</td>
</tr>
<tr>
<td></td>
<td>• Detailed Project Plan</td>
</tr>
<tr>
<td></td>
<td>• Project Web Page</td>
</tr>
<tr>
<td></td>
<td>Research:</td>
</tr>
<tr>
<td></td>
<td>• Strength and weakness of existing apps</td>
</tr>
<tr>
<td></td>
<td>• Privacy and legal problem</td>
</tr>
<tr>
<td></td>
<td>• Web scraping</td>
</tr>
<tr>
<td>Oct 2022</td>
<td>Research:</td>
</tr>
<tr>
<td></td>
<td>• Speech Recognizer during phone call</td>
</tr>
<tr>
<td></td>
<td>• Guideline of designing UI and UX</td>
</tr>
<tr>
<td>Nov 2022</td>
<td>Implementation:</td>
</tr>
<tr>
<td></td>
<td>• UI and UX design</td>
</tr>
<tr>
<td>Dec 2022</td>
<td>Research:</td>
</tr>
<tr>
<td></td>
<td>• Pop-up warning when received suspicious phone call</td>
</tr>
<tr>
<td></td>
<td>• Background task for validating phone number</td>
</tr>
<tr>
<td></td>
<td>Implementation:</td>
</tr>
<tr>
<td></td>
<td>• Function of displaying news about telephone deception</td>
</tr>
<tr>
<td></td>
<td>• Function of querying phone number report</td>
</tr>
<tr>
<td></td>
<td>• Function of reporting phone number</td>
</tr>
<tr>
<td>Jan 2023</td>
<td>Deliverables of Phase 2:</td>
</tr>
<tr>
<td></td>
<td>• Preliminary implementation</td>
</tr>
<tr>
<td></td>
<td>• Detailed interim report</td>
</tr>
<tr>
<td></td>
<td>• First presentation</td>
</tr>
<tr>
<td></td>
<td>Implementation:</td>
</tr>
<tr>
<td></td>
<td>• Call screening</td>
</tr>
<tr>
<td></td>
<td>• Web scraping</td>
</tr>
<tr>
<td></td>
<td>• Elderly mode</td>
</tr>
<tr>
<td>Feb 2023</td>
<td>Implementation:</td>
</tr>
<tr>
<td></td>
<td>• Function of sending SMS to users' trusted people</td>
</tr>
<tr>
<td></td>
<td>• Conversation analysis</td>
</tr>
<tr>
<td>Mar 2023</td>
<td>Implementation:</td>
</tr>
<tr>
<td></td>
<td>• Testing of the app</td>
</tr>
<tr>
<td></td>
<td>• Debugging</td>
</tr>
<tr>
<td></td>
<td>• Optimization</td>
</tr>
<tr>
<td>Apr 2023</td>
<td>Deliverables of Phase 3:</td>
</tr>
<tr>
<td></td>
<td>• Finalized tested implementation</td>
</tr>
<tr>
<td></td>
<td>• Final report</td>
</tr>
<tr>
<td></td>
<td>• Final presentation</td>
</tr>
</tbody>
</table>

Table 7.2 Project Schedule

This table illustrates this project schedule from September 2022 to April 2023. It includes the tasks of each month, research or implementation.