Faculty of Engineering and Digital Technologies
Department of Computer Science

Lost/Found Mobile Application

Mobile Application Development Coursework Report

Group Name: dvicente_ebgibson_nlewando

Written by:

- David Vital (UoB: 20008506) | Score: 5
- Elliot Gibson (UoB: 20006474) | Score: 4
- Natalia Lewandowska (UoB: 20015091) | Score: 5

3 January 2024

Declaration

David Vital, Elliot Gibson and Natalia Lewandowska

Abstract

Once complete, this project can be implemented in any academic, or professional environment and it will let any student, or staff member to report any lost items and more importantly, guide anyone who might have lost an item to a place where they can easily find it. This will be achieved by creating a mobile application using Android Studio, a Firebase Database instance and the Java programming language. We are hoping that this solution can work in helping any system who uses it to reduce the amount of important items lost and speed up the process of recovering said lost items, generally speaking.

Table of Contents

1. Group Contribution

1.1 Elliot Gibson – 20006474

My teammates David and Natalia were involved in most aspects of the projects and carried out all programming of the solution. I was involved in both the design phase and testing phase of development, but I think I could have contributed more than I did.

- David: 5
- Natalia: 5
- Elliot: 4

1.2 Natalia Lewandowska – 20015091

The project idea was mine, so I designed the app's UI in cooperation with David and Elliot (he also tested the app). David and I designed the database, took care of coding the app and documented it all in this report. I also took care of the app's security side.

- David: 5
- Elliot: 4
- Natalia: 5

1.3 David Vital – 20008506

I was allocated the role of team leader and was therefore mainly in charge of the coding and designing the app with Natalia (she also worked on cyber-security tasks) and Elliot took care of any testing, helping out with design whenever possible. Everyone did a solid job.

- Elliot: 4
- Natalia: 5
- David: 5

2. Introduction

University students and staff members have always led quick paced and busy lives. This, unfortunately, means that sometimes we are bound to lose all kinds of things as we go on through with our days. Most of the time, those items never get found and when they do, there is usually no guarantee that whoever finds it knows how to give it back.

For this reason, we came up with the idea of building a lost/found app for this project. An application that would let anyone who uses it, post about items they might have found lying around campus, mention where they have left it and hopefully its rightful owner would come and pick it up.

When developing this application we will be following along with the requirements provided to us in the coursework description, these are:

- Comprise of good quality UI with at least two fragments separating all the components.
- Use the activity life cycle states through its various methods (onStart(), onPause()).

- Utilise at least one of Device Location / Google Maps / Camera / Microphone.
- Use persistence data storage (Firebase).
- Use MVC or similar framework.

3. Case Study Brief

Case Study: "The practicality of a lost/found app in an academic environment"

This lost/found application can provide the university with a place where both teachers and students can report items that were lost during the day, or see if any items they might have lost were found and if so, where can said items be found. The main priority of this application will always be to decrease the amount of items with high importance that are lost, like wallets, student cards and even pen drives. For example, with this piece of software implemented, a staff member could find a lost wallet, laying around in our lab facilities, easily create a post through the form system, mention who they left it with and hopefully the person who lost the item could easily go and pick it up.

4. Problem Definition and Analysis

While thinking on what kind of mobile application our team could develop, we knew we are aiming for a project, which would be useful for three of us and also our University colleagues. The Lost/Found App is a perfect example of how anyone on the campus could safely use it for themselves. The significant problem picked up by us was the hassle any of us had to go through in the case if we lost something while being on the campus. The only way of receiving our items back was to physically search the labs rooms or check the reception with hope someone found it and brought it there. We could also ask around if anyone saw the missing item, but in most of the cases we would have to accept the fact that we lost it forever.

What in the case if the missing belonging is very expensive phone, wallet or pen-drive containing sensitive data? We believe that our mobile application would not solve the problem in 100%, since not everyone who finds someone's lost items bother to report it anywhere. However, with the Found/Lost app the hassle becomes a one minute process of posting it in our system. This way the items' owner will have easier and less stressful time of looking for it and know at least if there is any point of even bother to go to the University reception.

As we noticed, in fact, there is no similar system created for the University Of Bradford. Since it is way common problem than we think, we tried to make the UI as easy to use as possible, on some sort of social media posts we create every day. The app design allow the user to provide various of found item details like time, date, its description, picture and even exact location by used google maps. We believe the Found/Lost app is a very practical tool, which could be used by anyone as long as they belong to the University of Bradford community.

5. Requirements

In this chapter there will be a conversation about the app requirements (may they be mandatory, or optional) that we, as a group, came up with, in order to improve project.

5.1 Mandatory requirements

In the following segment, we will be focusing our attention to the requirements that not only need to be accomplished by the end of our implementation, but are also responsible for having most of the application´s functionality related to them. These are presented in the table below:

Mandatory Features	
Feature Number	**Feature Description**
1	Login and Sign Up Activity that link back to Firebase Authentication.
2	Create a way to make sure that only University of Bradford students/staff members can access this application.
3	Manage error handling when logging in and signing up.
4	Have the list of found items show up in its entirety in one of the panels.
5	Have a form that lets users create their own posts, by filling in all the information required and then sending it to the Firebase.
6	When creating the form make use of the Google Maps and Camera APIs.
7	Manage all the different errors that can possibly show up when filling in the create a post form.
8	Make sure that all the buttons/links in the user interface have a purpose and fulfil it correctly (including an option to logout).

Table 1: Mandatory project features (Source: Authors)

5.2 *Optional requirements*

On the opposite side of things, this next section, will be more focused on presenting what optional ways we came up with that can eventually help us with improving the user experience of the app, but are not exactly necessary for it to reach the desired functionality. These are presented below:

Optional Features	
Feature Number	**Feature Description**
1	Make use of the email verification functionality provided to us by Firebase.
2	Implement forgot password functionality also provided to us by Firebase.
3	Add some extra checks, in the foundItems fragment in order to let users delete their own posts.
4	Create an admin panel to make it easier to delete users from the app once they are no longer part of the university and approving posts.
5	Have an extra fragment or activity that users can use to report issues.
6	Find a way to highlight lost items with higher importance.

Table 2: Optional project features (Source: Authors)

Most of these features might never get to be implemented and this would be due to not only a lack of time, but to the fact that this app is not meant to be very complex either. This, however, did not stop us from coming up with these ways of improving our system.

6. Design and MVC

6.1 Design

In this section we are going to talk about the initial mobile app design we thought would fulfil the project requirements. As shown below – Fig.1. Presents four phone screens:

- Login Panel – Independent Activity.
- SignUp Panel – Independent Activity.
- FoundList Fragment – Part of the MainActivity
- Form Fragment – Part of the Main Activity.

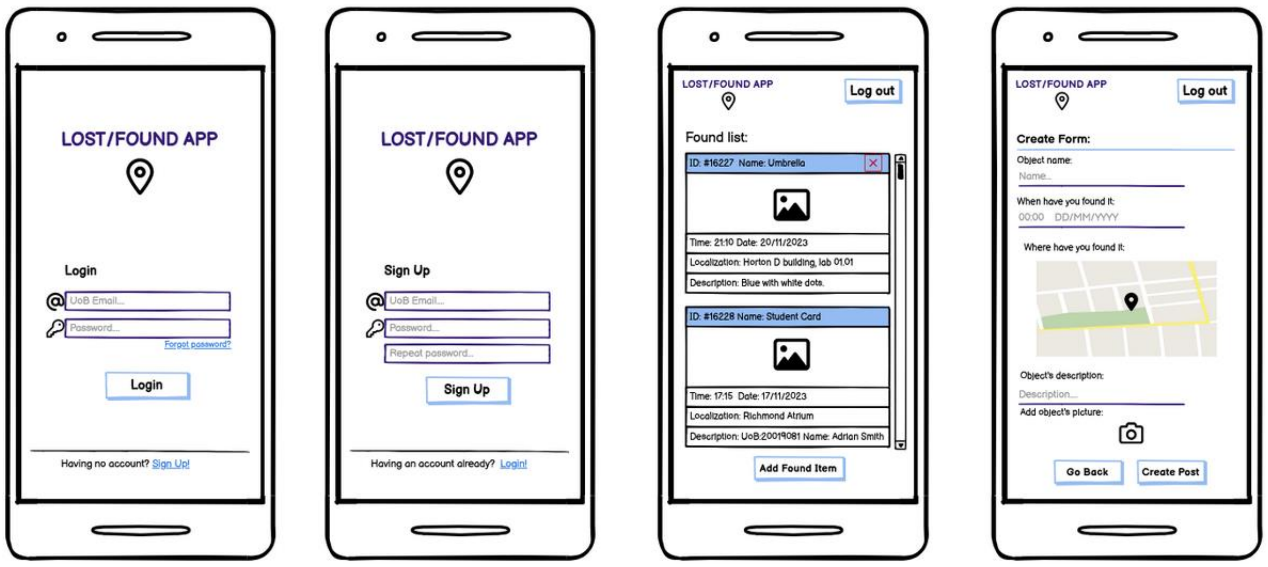


Fig. 1. The UI design sketches of four mobile Panels. Source: Authors.

Regarding project requirements about developing at least one activity and two fragments, we can safely admit that all the design requirements are met. As mentioned before, all four UI sketches were created to help visualize our project ideas and to support the XML coding part in android studio. These were planned and developed using balsamiq.com tool. Thanks to their services the planning part was easier, faster and more efficient in general.

So, regarding the earlier mentioned panels, we planned to have two independent Login/SignUp panels, which would take care of the user authentication using Firebase database. While designing the MainActivity, which would include two fragments, we met our first challenge with planning the FoundList fragment. We were planning on displaying all the found items in a list, which would contain:

- Object ID,
- Object Name,
- Object Image,
- Object Time,
- Object Date,
- Object Localization,
- Object Description.

We have decided to go with RecycleView mentioned during one of our lectures, which would help us neatly display all the data by using a CardView. We aimed to make that fragment view as easy and intuitional as possible.

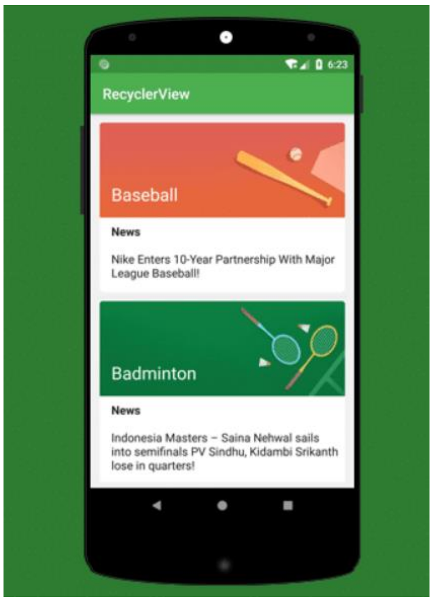

RecyclerView
Baseball
News
Nike Enters 10-Year Partnership With Major League Baseball!
Badminton
News
Indonesia Masters – Saina Nehwal sails into semifinals PV Sindhu, Kidambi Srikanth lose in quarters!

Fig. 2. An example of RecycleView Design. Source:[1]

Based on found examples, we managed to design third Panel. However, the fourth panel – FormFragment was designed to take earlier mentioned information from the user. We planned on using camera and also a google MapView as a part of our submission requirements. In the Implementation chapter we are going to go deeper into each of our panels and explaining their functionality.

6.2 *MVC vs MVVM*

In this section we are going to explain how our program logic was divided into three interconnected elements: Model, View and Control/ViewModel. As we used Firebase database, we found the MVC design quite tricky. The first reason is that the no-SQL database we decided to use for the Found/Lost mobile app takes care of lots of back-end controls. On one hand, taking into account Firebase's reactive and real-time nature, we think it would be easier to apply MVVM model, which stands for Model, View, ViewModel. On the other hand, our mobile application logic and general design is very simple, which does not require complex model.

Finally, we decided to apply MVC model, which would look as below:

- User – as a part of Firebase Authentication (Model),
- FoundItem Class (Model),
- RecycleView Class (Controller),
- Login Activity (View/Controller),
- SignUp Activity (View/Controller),
- MainActivity (View/Controller),
- FormFragment (View),
- FoundListFragment (View).

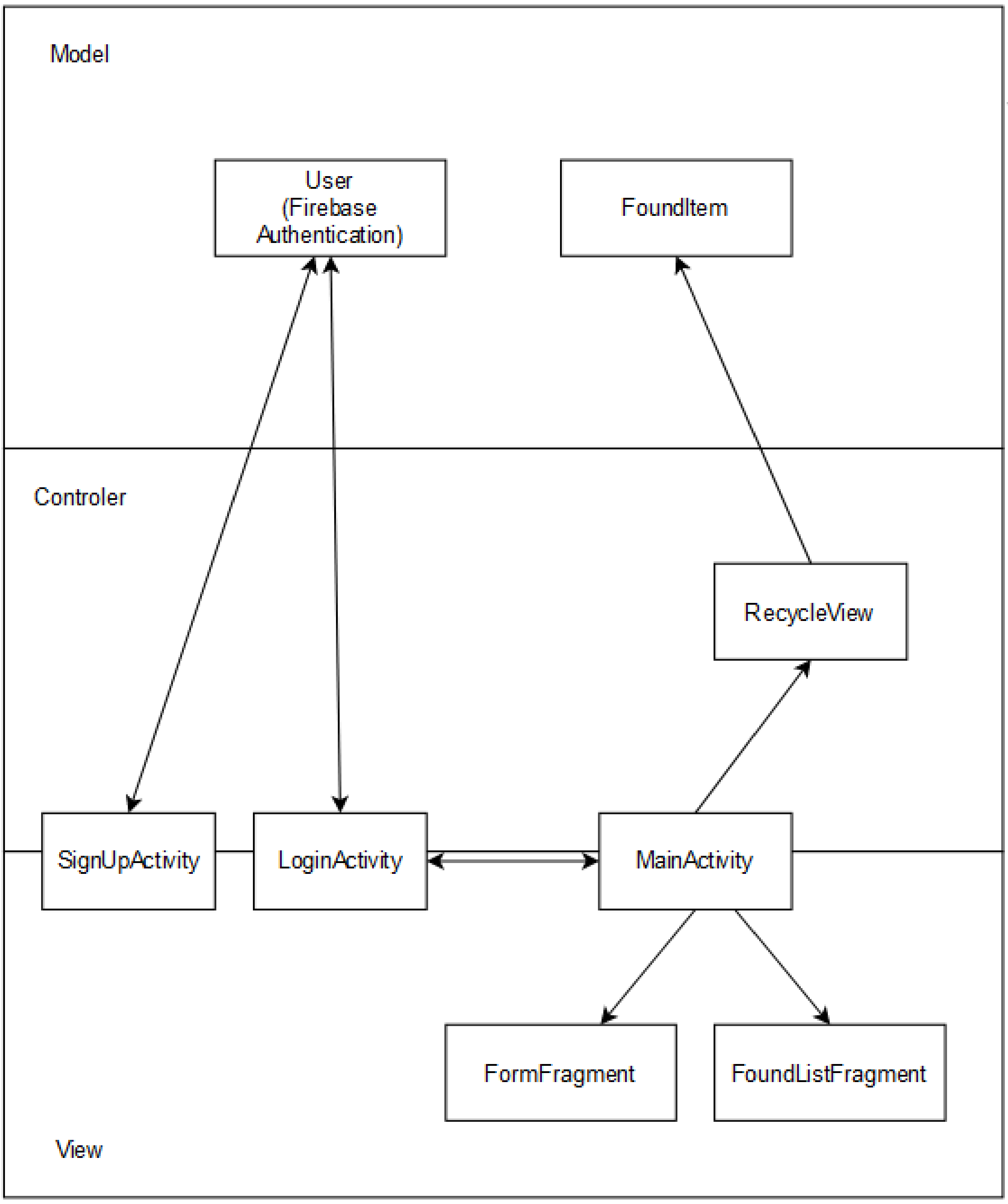


Fig. 3. MVC model based on Lost/Found mobile app's logic. Source: Authors.

7. Implementation

Implementation section of the submission is the part we certainly put the most effort in. The current chapter includes project settings, Login/SignUp Panels, MainActivity, Found List Fragment and the Form Fragment. The last part of the Implementation chapter is Firebase performance and all used services. The project Lost/Found mobile app was developed in android studio using java language with the most recent plugin versions.

Android Studio Version: Android Studio Hedgehog, 2023.1.1 November 30, 2023
Java: JavaVersion.Version_1_8
Plugins versions: 8.2.0
SDK: minSdk: 21 | targetSdk: 34

7.1 Login and Sign Up Activities

Login and SignUp panels are very classic, both based on the users email and password, as it is the chosen Firebase provider. The Login panel is the default view, so after running the application it will be the first thing to show on the user's screen. If the user has no created account yet, by clicking the „Sign Up!" link (located on the bottom of the Login Screen – Fig.4.), the system will take the user to the SingUp panel. It is where accounts are created, by providing University Of Bradford email (only accepted by the system) and strong password. If the entered details meet the system requirements, then the sign up is successful and the system sends the email verification link – Fig.6.

Fig.4, Fig.5, Fig.6, Source: Authors

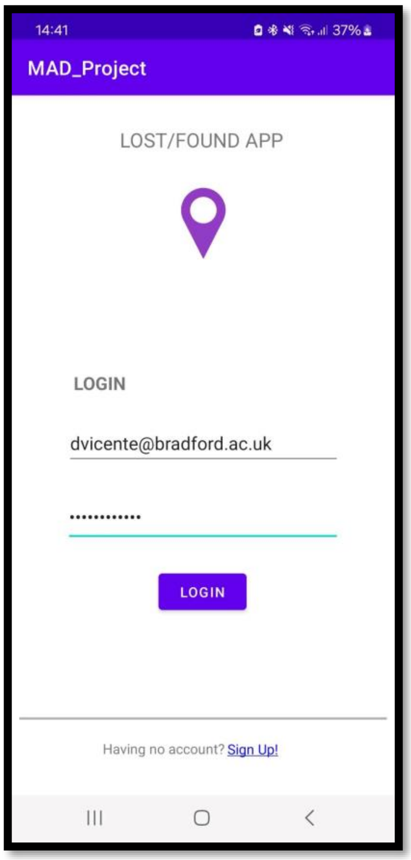


Fig. 4. Login Panel.

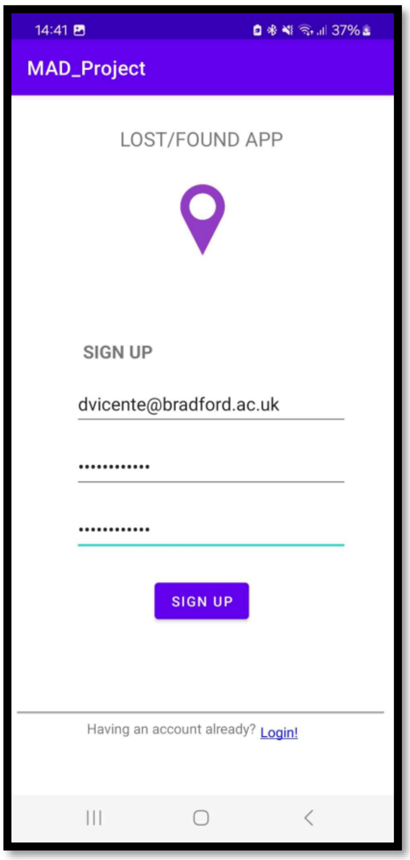
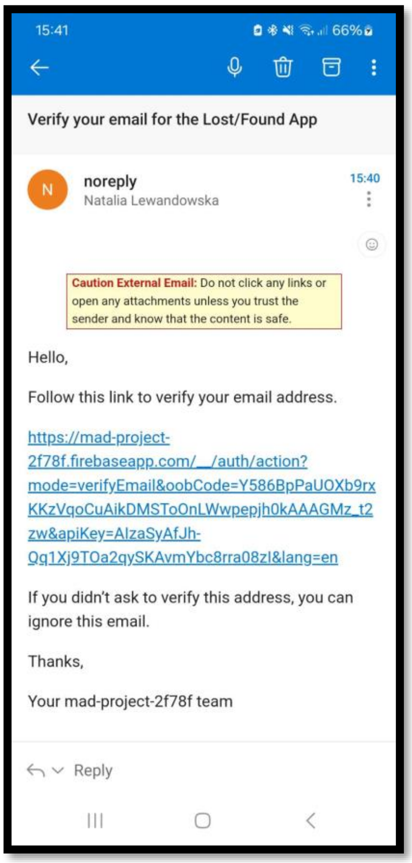


Fig.5. SignUp Panel.



Verify your email for the Lost/Found App

noreply
Natalia Lewandowska
15:40

Caution External Email: Do not click any links or open any attachments unless you trust the sender and know that the content is safe.

Hello,

Follow this link to verify your email address.

https://mad-project-2f78f.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=Y586BpPaUOXb9rxKKzVqoCuAikDMSToOnLWwpepjh0kAAAGMz_t2zw&apiKey=AIzaSyAfJh-Qq1Xj9TOa2qySKAvmYbc8rra08zI&lang=en

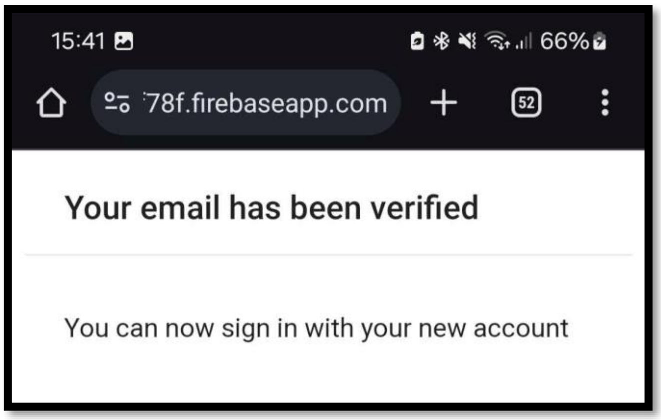
If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your mad-project-2f78f team

Fig.6. SignUp verification email.

By clicking on the link provided inside of the verification email, the system will display web notification – Fig.7, which effectively will eliminate any credential thefts and unauthorised system breaches. After completing all mentioned steps, the system will let the user log in in the Login panel.


15:41

78f.firebaseapp.com

Your email has been verified

You can now sign in with your new account

Fig. 7. Email verified notification. Source: Authors.

Limitations: While we were working on these two panels, we did not take into consideration developing „Forgot Password" Panel. Certainly, we are thinking of adding this app function to the Future Work section.

7.2 Main Activity

The main activity does not present in the application as a panel itself. Its role is to let the user logout and it also holds the Fragment Container for: Found List and Form fragments. Once the user (teacher or student) login to the system, the MainActivity appears containing Found List Fragment. After clicking the „Add Found Item" button, the fragment container exchange it for the Form Fragment. We could achieve that using Fragment Manager and Fragment transaction presented below – Fig.8.

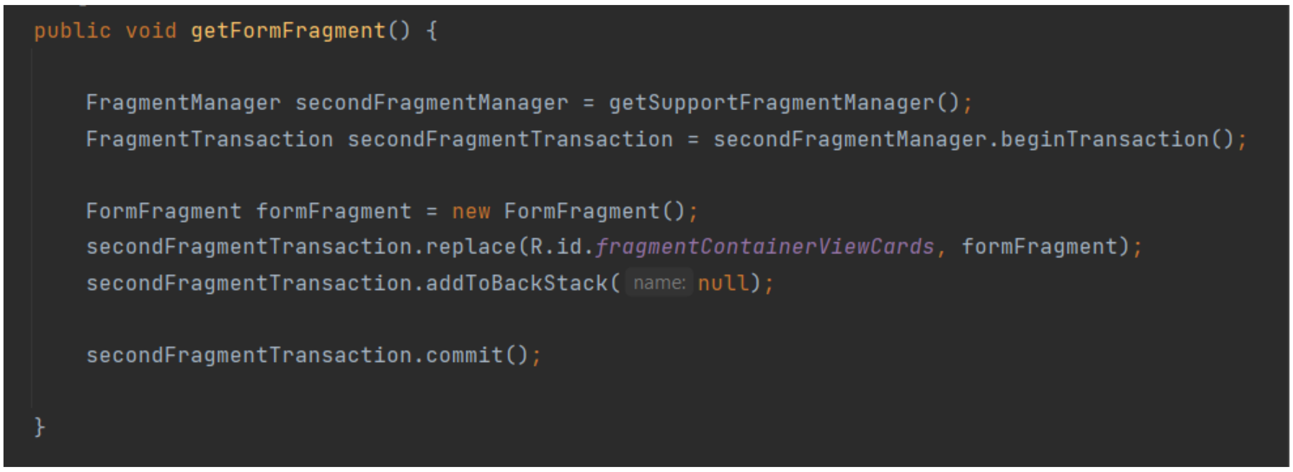
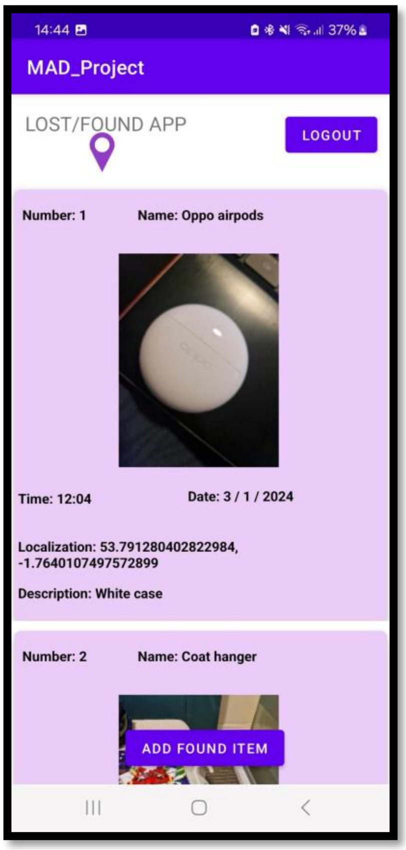

```
public void getFormFragment() {

    FragmentManager secondFragmentManager = getSupportFragmentManager();
    FragmentTransaction secondFragmentTransaction = secondFragmentManager.beginTransaction();

    FormFragment formFragment = new FormFragment();
    secondFragmentTransaction.replace(R.id.fragmentContainerViewCards, formFragment);
    secondFragmentTransaction.addToBackStack( name: null);

    secondFragmentTransaction.commit();

}
```

Fig.8. Fragment transaction to get formFragment called from MainActivity. Source: Authors.

7.3 *Found List Fragment*

As mentioned, Found List Fragment is the default user's view after login to the system. That screen displays all the data about all Lost/Found items previously inserted in the Form Fragment. For displaying the data we decided to use RecycleView. Especially, taking into account the kind of data we are going to store: images and geo-points. As mentioned in the design section, we had quite a lot of data we were going to display, so it was very important to careful plan the CardView. At the end we came up with results shown below:



Number: 1 Name: Oppo airpods

Time: 12:04 Date: 3 / 1 / 2024

Localization: 53.791280402822984, -1.7640107497572899

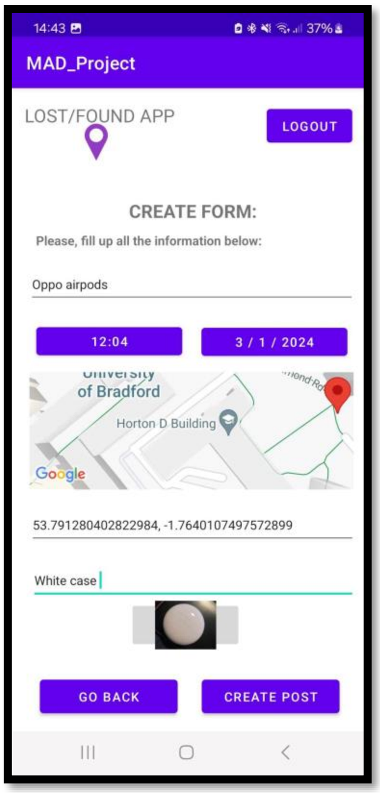
Description: White case

Number: 2 Name: Coat hanger

Fig. 9. The foundList fragment. Source: Authors.

7.4 *Form Fragment*

Even though we knew already how we are going to display our data, the most challenging part was to figure out how to receive it from the user and store in the database. There was no problem with getting Strings, since we did it before with users credentials. It relates to the plan on adding a google map, so the user could find the place of the Found item on the map and by clicking on it reads its geo-location. The image storage problem was solved by using Firestore service instead of Real-Time database dedicated for rest of the data storage. The last concern was to make our Time and Date fields to accept only the time and date formats. So, to ensure it, we transformed our fields into spinner buttons – Fig. 11.



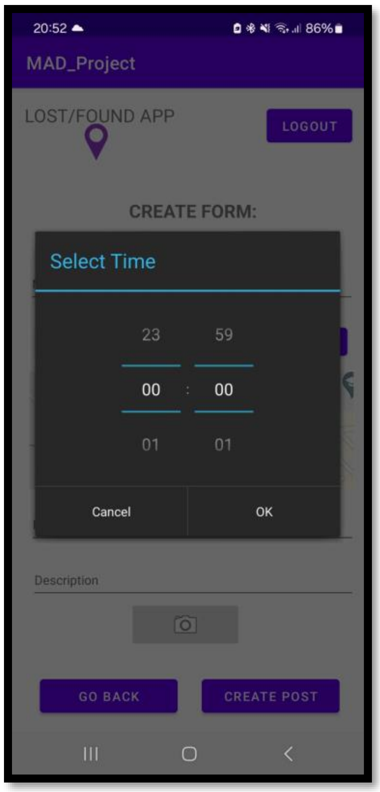


Fig. 10. The formFragment. Source: Authors.

Fig. 11. The Time spinner. Source: Authors.

Another part of our submission was applying the activity life cycle: OnStart(), OnPause(), OnDestroy() etc. Thanks to this solution our mobile application benefits e.g. by saving and restoring previously entered data, even if the user will leave the app without destroying it (onPause()). The user experience is better in general, in terms of reguarly updated UI and well-handled background processes. The last significant benefit is statisticaly less appeared errors due to higher testing performance by triggering various life cycle events. Relating to our application, we used the Activity Life Cycles as shown below – Fig. 12.

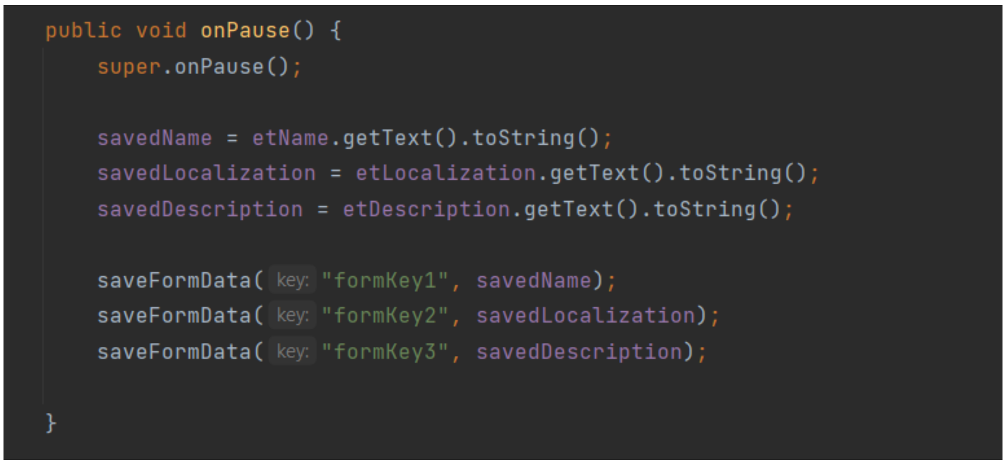
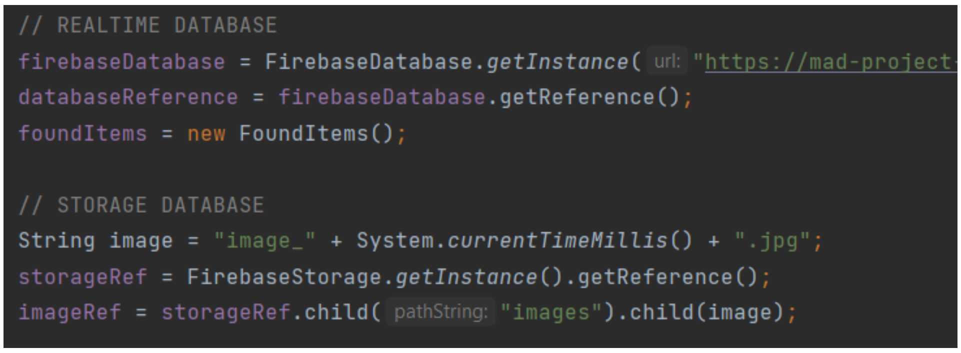

```
public void onPause() {
    super.onPause();

    savedName = etName.getText().toString();
    savedLocalization = etLocalization.getText().toString();
    savedDescription = etDescription.getText().toString();

    saveFormData( key: "formKey1", savedName);
    saveFormData( key: "formKey2", savedLocalization);
    saveFormData( key: "formKey3", savedDescription);

}
```

Fig.12. An example of onPause() life cycle usage. Source: Authors.

7.5 *Firebase database*

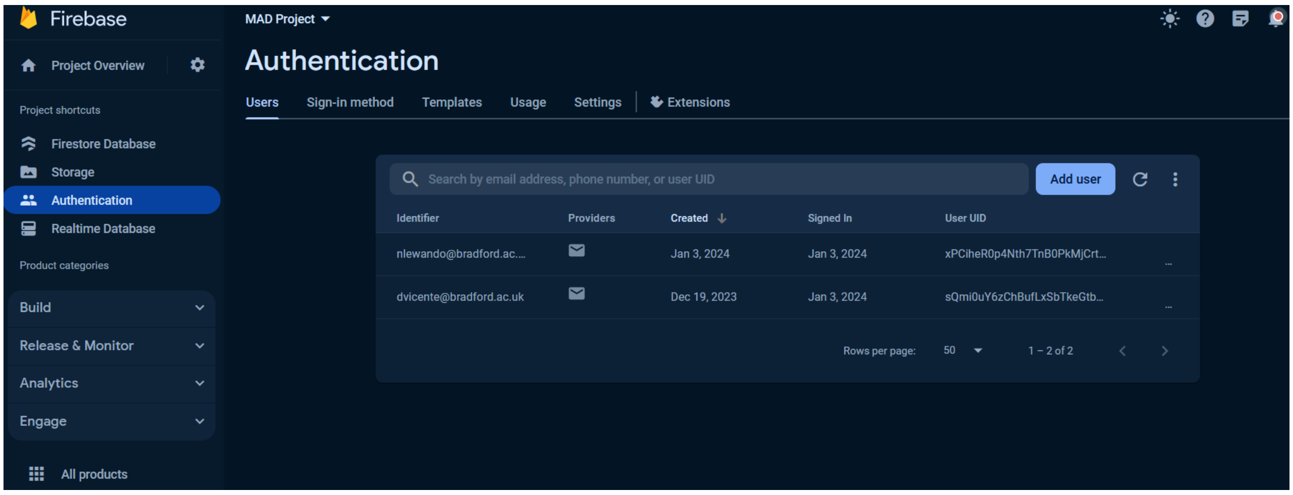
Last part of the chapter is the Firebase database. Our choice about using no-SQL cloud storage method was mostly based on the need of Real-Time storage and the general ease of using Firebases services. As mentioned before, we also used the Firestore for our items' images and the Firebase authentication service. To connect with the Real-Time and Storages databases we used:



```
// REALTIME DATABASE
firebaseDatabase = FirebaseDatabase.getInstance( url: "https://mad-project
databaseReference = firebaseDatabase.getReference();
foundItems = new FoundItems();

// STORAGE DATABASE
String image = "image_" + System.currentTimeMillis() + ".jpg";
storageRef = FirebaseStorage.getInstance().getReference();
imageRef = storageRef.child( pathString: "images").child(image);
```

Fig. 13. Firebase Real-Time and Firestore connection code in Android Studio. Source: Authors.

Our users sign up to the system by using Email/Password Firebase authentication method. Below we presented a sample of our user test database entries – Fig. 14.



Identifier	Providers	Created	Signed In	User UID
nlewando@bradford.ac....	✉	Jan 3, 2024	Jan 3, 2024	xPCiheR0p4Nth7TnB0PkMjCrt...
dvicente@bradford.ac.uk	✉	Dec 19, 2023	Jan 3, 2024	sQmi0uY6zChBufLxSbTkeGtb...

Rows per page: 50 1 – 2 of 2

Fig.14 The sample of the user test authentication entries. Source: Authors.

8. Testing

This section covers the testing process including both functionality and usability aspects of the solution.

8.1 Functionality Testing

Before engaging in usability testing, it is important that the functional requirements of the case study have been reached. This will allow the user journey to be completed start to finish as originally intended in the design stage. Here we will list each of the requirements from sections 5.1 and 5.2 with an assessment of how they have or have not been effectively implemented in the final product build.

Test Reference	FTS01
Tested Requirement	FT01
Test Content	Checks user can sign up and log in.
Input	An email and user-defined password.
Pass Criteria	Program allows sign up and login with email and user-defined password.
Result	PASS

Test Reference	FTS02
Tested Requirement	FT02
Test Content	Checks user can only sign up with a UoB email (example@bradford.ac.uk).
Input	A UoB email and user-defined password.
Pass Criteria	Program allows sign up and login with UoB email, and rejects sign up with any other email suffix (e.g. @hotmail.co.uk).
Result	PASS

Test Reference	FTS03
Tested Requirement	FT03
Test Content	Checks user cannot enter nebulous or non-existent email and password for sign up or login.
Input	Combinations of empty email field or incorrectly formatted email, and empty password field.
Pass Criteria	Program does not allow sign up or login with incorrect input.
Result	PASS

Test Reference	FTS04
Tested Requirement	FT04
Test Content	Ensure lost items can be viewed as an entire list in the related fragment.
Input	N/A
Pass Criteria	Application correctly lists lost items stored in Firebase in their entirety, including image attachments.
Result	PASS

Test Reference	FTS05
Tested Requirement	FT05
Test Content	Ensure lost item submission form functions as expected.
Input	Required fields for making an item submission.
Pass Criteria	Item details stored successfully in Firebase with all required fields filled correctly.
Result	PASS

Test Reference	FTS06
Tested Requirement	FT06
Test Content	Included Android APIs are integrated correctly and function as expected.
Input	Google Maps co-ordinates and Camera-assisted image attachment for lost item submission.
Pass Criteria	Lost item form submits successfully, optional fields (co-ordinates and image attachments) are stored correctly in Firebase.
Result	PASS

Test Reference	FTS07
Tested Requirement	FT07
Test Content	Ensure robustness of submissions form error-handling.
Input	Nebulous input for form fields, e.g. empty required fields.
Pass Criteria	Form handles any nebulous input with graceful catches and user notifications.
Result	PASS

Test Reference	FTS08
Tested Requirement	FT08
Test Content	All interactive parts of the UI are functional and succeed in carrying out their expected jobs.
Input	N/A
Pass Criteria	UI is fully interactive and usable.
Result	PASS

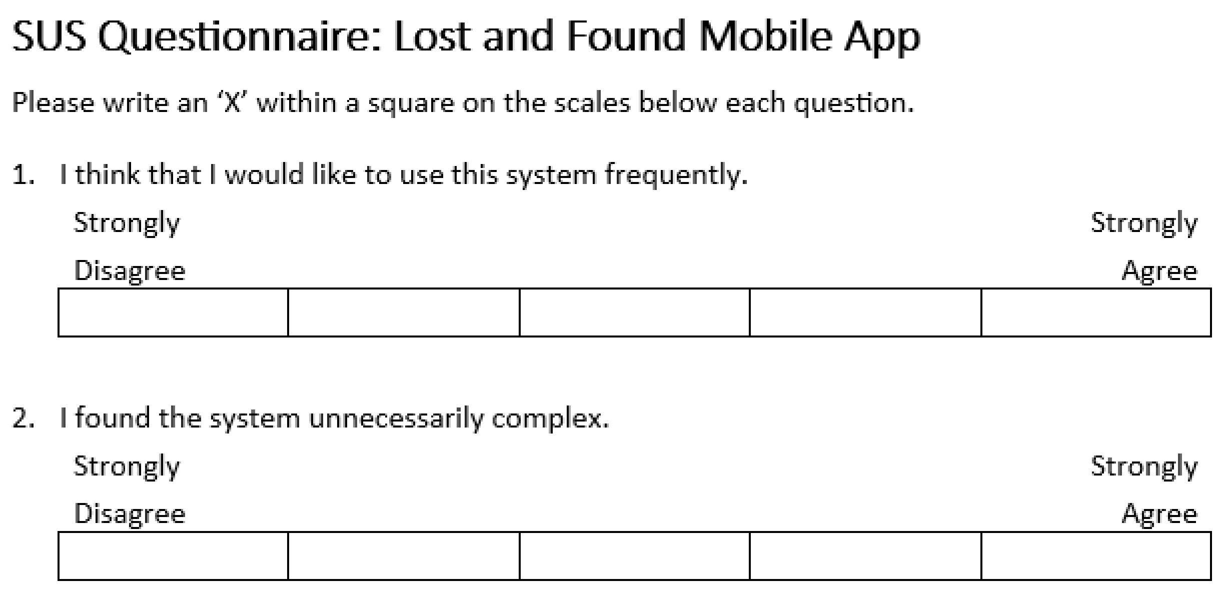
Table 3: Tables containing all of our functionality tests (Source: Authors)

8.2 *Usability Testing/Surveying*

Usability testing will cover both navigation and content of the current project build. The .apk file and a goal-based scenario is given to the users. The scenario is shown below:

"This app facilitates submission of lost items found by you - the user. Your goal is to use the app to submit a lost item form. Optional: include an image and/or a Google Maps tag with your form submission through the app."

After the users complete the scenario they are given an SUS questionnaire to rate their experience with the app. The questionnaire consists of 10 questions to be answered via a 1-5 scale.



SUS Questionnaire: Lost and Found Mobile App

Please write an 'X' within a square on the scales below each question.

1. I think that I would like to use this system frequently.

Strongly Disagree — Strongly Agree

2. I found the system unnecessarily complex.

Strongly Disagree — Strongly Agree

Fig.15 Example questions used in our SUS Questionnaire. Source: Authors.

After the users have answered each question the SUS formula is applied, giving a final scale of usability for the app.

User One's SUS results:

$$(1 + 4 + 0 + 0 + 4 + 3 + 4 + 1 + 0 + 1) * 2.5 = 45$$

User Two's SUS results:

$$(3 + 3 + 3 + 4 + 3 + 3 + 4 + 4 + 3 + 4) * 2.5 = 85$$

According to researchers, a score of around 68 is considered the average. From these results we can see that some areas of the app could use re-evaluating to make usability more consistent among a wider range of users. Screenshots of the full completed questionnaires can be found in the Appendix.

9. Conclusions

When looking back at the requirements mentioned previously, we can see that 9 of them where met, (including all the mandatory ones and the email verification one in the optional features) and the other 5 where not. Thankfully, this does not affect how satisfied we are with the final product, it just means that there are some opportunities left for improvement, in the future.

In conclusion, a lot of work and effort was put into this project, most noticeably into its implementation and hopefully in can be turned into something more and be useful to someone. Like mentioned before, if this project ever gets picked up again, we are hoping that some development can be done towards implementing most of the optional features that were left, if not all of them. Optional features like the reporting issues and the admin panel should always take priority, as if this kind of app is left unsupervised, it can lead to some unfortunate situations.

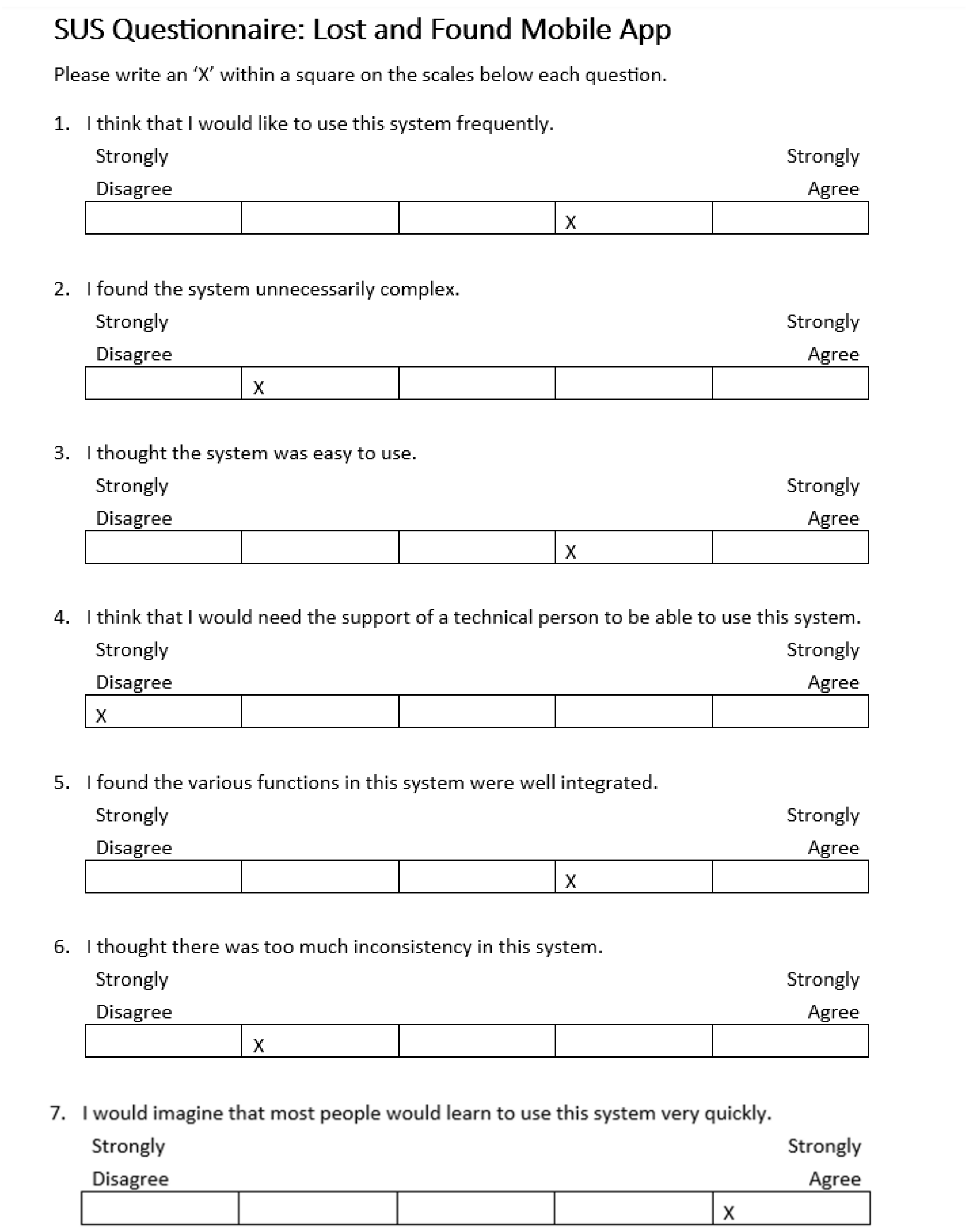
10. Bibliography

[1] Android Knowledge (2023) Login and SignUp using Firebase Authentication in Android Studio [Online]. Available at: https://www.youtube.com/watch?v=TStttJRAPhE&t=213s (Last accessed: 3rd of January 2024).

[2] AndroidWave (2020) RecyclerView in Android Example Best Practices [Online]. Available at: https://androidwave.com/android-recyclerview-example-best-practices/ (Last accessed: 3rd of January 2024).

[3] Code With Cal (2022) Pop Up Time Picker Android Studio Tutorial | Analog or Spinner Style [Online]. Available at: https://www.youtube.com/watch?v=c6c1giRekB4 (Last accessed: 3rd of January 2024).

11. Appendices



SUS Questionnaire: Lost and Found Mobile App

Please write an 'X' within a square on the scales below each question.

1. I think that I would like to use this system frequently.

Strongly Disagree — Strongly Agree

			X	

2. I found the system unnecessarily complex.

Strongly Disagree — Strongly Agree

	X			

3. I thought the system was easy to use.

Strongly Disagree — Strongly Agree

			X	

4. I think that I would need the support of a technical person to be able to use this system.

Strongly Disagree — Strongly Agree

X				

5. I found the various functions in this system were well integrated.

Strongly Disagree — Strongly Agree

			X	

6. I thought there was too much inconsistency in this system.

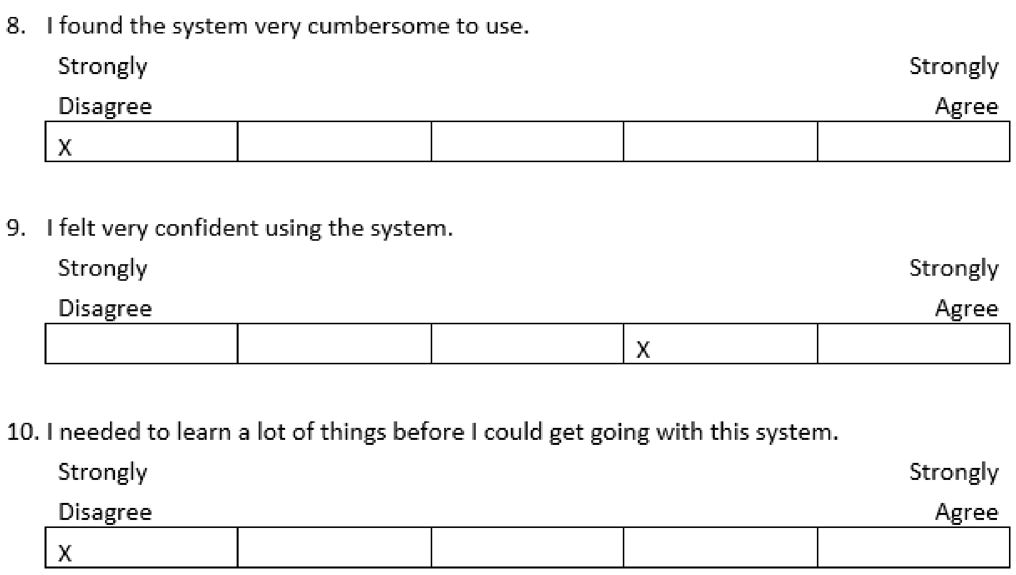
Strongly Disagree — Strongly Agree

	X			

7. I would imagine that most people would learn to use this system very quickly.

Strongly Disagree — Strongly Agree

				X



8. I found the system very cumbersome to use.

Strongly Disagree — Strongly Agree

X				

9. I felt very confident using the system.

Strongly Disagree — Strongly Agree

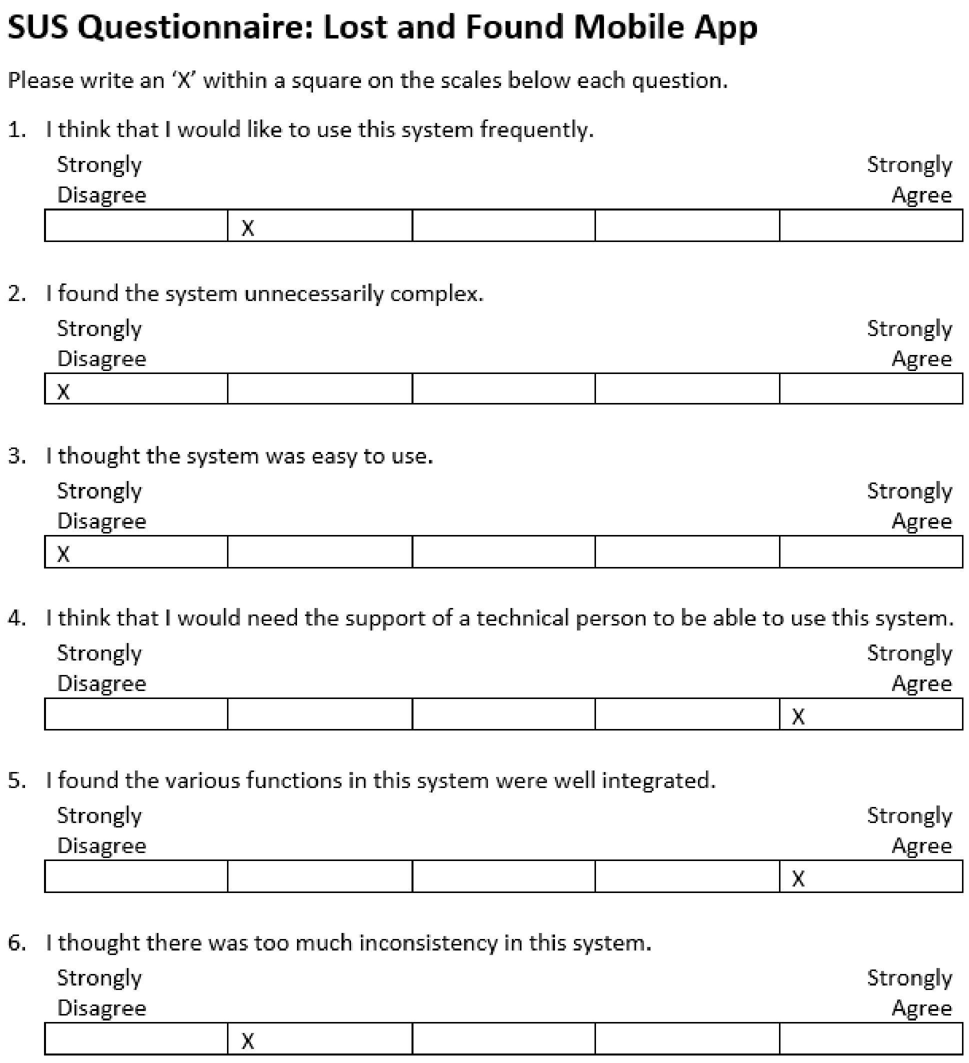
			X	

10. I needed to learn a lot of things before I could get going with this system.

Strongly Disagree — Strongly Agree

X				

SUS user 2



SUS Questionnaire: Lost and Found Mobile App

Please write an 'X' within a square on the scales below each question.

1. I think that I would like to use this system frequently.

Strongly Disagree — Strongly Agree

	X			

2. I found the system unnecessarily complex.

Strongly Disagree — Strongly Agree

X				

3. I thought the system was easy to use.

Strongly Disagree — Strongly Agree

X				

4. I think that I would need the support of a technical person to be able to use this system.

Strongly Disagree — Strongly Agree

				X

5. I found the various functions in this system were well integrated.

Strongly Disagree — Strongly Agree

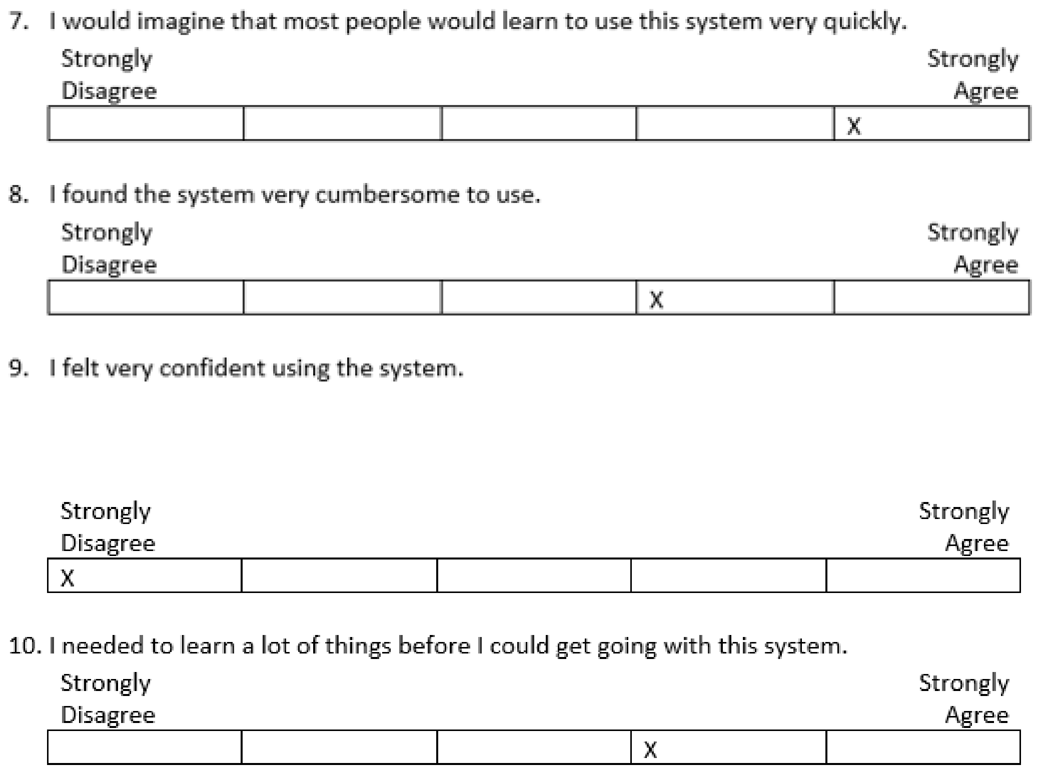
				X

6. I thought there was too much inconsistency in this system.

Strongly Disagree — Strongly Agree

	X			

SUS user 1, pt.1



7. I would imagine that most people would learn to use this system very quickly.

Strongly Disagree | Strongly Agree

				X

8. I found the system very cumbersome to use.

Strongly Disagree | Strongly Agree

			X	

9. I felt very confident using the system.

Strongly Disagree | Strongly Agree

X				

10. I needed to learn a lot of things before I could get going with this system.

Strongly Disagree | Strongly Agree

			X	

SUS user 1