

GOLDSMITHS, UNIVERSITY OF LONDON

SOFTWARE PROJECT FINAL REPORT

iLost

*Ahmed, Muhammad
Chowdhury, Thairan
Davies Minta, Dylan
Fakrul, Mahmudul
Farkhani, Hussein
Jheng-Hao, Lin
Pecorella, Mariano*

supervised by
TIM BLACKWELL

April 24, 2018

Contents

1	Introduction	3
2	Development Record	3
2.1	Teams	3
2.2	Technology Selection	4
2.3	Agile Development	5
2.3.1	Scrum Development	5
2.3.2	Kanban	7
2.3.3	Behavior Driven Development	8
2.4	Development Process	9
2.4.1	Backlogs	9
2.4.2	Progress Tracking	10
3	Formative Evaluation	11
3.1	iOS App Evaluation	11
3.1.1	Objectives and Questions	11
3.1.2	Participants, Location and Setup	12
3.1.3	Methodology and Measures	12
3.1.4	Evaluation	13
3.2	Conclusion	14
4	Design and Implementation	14
4.1	Design Stage	14
4.2	Implementation Stage	17
5	Quality Assurance	29
5.1	Process	29
5.2	Black-box Tests	29
5.2.1	Method and Environment	29
5.2.2	Unit Tests	30
5.3	White-Box Tests	31
5.3.1	Method and Environment	31
5.3.2	Test Cases	32
5.4	Evaluation	34
6	Summative Evaluation	34
6.1	Development Methodologies	34
6.2	Formative Evaluation	35
6.3	Design and Implementation	36
6.4	Quality Assurance	36
A	Appendices	38
A	Development Records	38
A.1	Backlogs	38
A.2	Architecture Decision Records	41
A.2.1	Mobile Development	41

A.2.2	iOS Development	41
A.2.3	Android Application Development	41
A.2.4	Transfer iOS Development from Swift to React Native	42
A.3	Tasks Divided	42
A.4	Progress Tracking Form	42
B	Formative Evaluation	46
B.1	Consent Form	46
C	Design and Implementation	48
C.1	Computer Fair Posters	48
D	Quality Assurance	52
D.1	Black-box Test Cases	52

1 Introduction

Losing belongings is a common problem, roughly 70 worth of items are lost in the public per year. [1] We conducted a survey about lost items to find out if we could create a useful application based on this concern. Following our survey, this statement was found to relate to university students as 89% of our participants claimed to be susceptible to losing items. This information motivated us to reduce the loss of personal belongings; leading to the development of an item tracking application and a portable tracking device. Our app aims to circumvent the loss of bags and other valuables. The user will attach the portable tracker to their bag or belonging, allowing it to be tracked using our app. The app will notify the user when the distance between the app and tracker reaches 30m.

This report will cover five main categories:

- Development Record: We explain the methodology used by our group to decide the technology with which to implement the apps functionality alongside the development methods that were used. We also describe the way that the development was handled by the team and provide a reflection on its efficiency.
- Formative Evaluation: This section of the report describes our assessment with users during development. The formative evaluation details what we did during user testing and its outcomes.
- Design and Implementation: An overview of our final design and implementation, including any changes from our initial ideas with justifications for the significant decisions we made.
- Quality Assurance: We detail our approach to Quality Assurance and the testing carried out, including our results. An assessment of how well our final system imitates our initial requirements with justifications of any changes we made to the requirements.
- Summative Evaluation: A descriptive evaluation of the methods, results and conclusion of our final software.

2 Development Record

2.1 Teams

During the implementation stage, we set up an organisation **GSoft** on GitHub. We were divided into three teams, **iOS app**, **Android app** and **Backend/Tracker team**, which shows in figure 1. Each team was grouped by 2 to 3 people who were more interested in that topic or technology. Some of us were interested in more than two areas, if this was the case; members could join more than one team. The list of the team members can be found in table 1.

The screenshot shows the GitHub interface for the 'GSoft' organization. At the top, there are navigation links for 'Repositories 3', 'People 7', 'Teams 3' (which is highlighted in orange), 'Projects 0', and 'Settings'. Below the navigation bar is a search bar with the placeholder 'Find a team...' and a green 'New team' button. The main content area displays three teams:

- Android-App**: Described as 'Android version app.' It has 3 members and 0 teams. The members are represented by small circular profile pictures.
- Back-end**: Described as 'testing and developing the physical tracker and integration with front-end app.' It has 4 members and 0 teams. The members are represented by small circular profile pictures.
- iOS-App**: Described as 'iOS version app.' It has 2 members and 0 teams. The members are represented by small circular profile pictures.

Figure 1: Development Teams on Github

Team	Members
iOS app	Jheng-Hao(leader), Muhammad
Android app	Dyland(leader), Mahmudul, Jheng-Hao
Backend/Tracker	Hussein(leader), Thairan, Mahmudul, Mariano

Table 1: Teams

2.2 Technology Selection

Development Teams: Each team was responsible for how and what technology to use, as long as the production could be made on time. We believed this methodology had these advantages in terms of the limited development time:

- React Agilely: Compared to having a poll with all members of our group, it was agiler to come to the decision within 2 or 3 in their designated team. Since each team could react to the situation and resolve any issues in a more efficient way.
- Specialities differ: Our group was divided by our interests and specialities, we trusted each team could make the best decision for the whole team with their research and experience. For example, the iOS app team would not interfere with how the tracker team implemented the physical components at all; And how the Android app was implemented would not be the backend/tracker team's concern. All teams were trusted in which they would make the best decisions.

Even though the team was separated, it was important to keep everyone on the same page. So each technological decision or propose was documented as an architecture decision record(ADR), which enabled us to have an overview of all the technological changes. According to Michael Nygard, each ADR contains five columns[1]:

- Title: A brief description of the idea proposed.

- Context: **Why** do we need to make the decision or change.
- Decision: **What** are the responses toward the decision.
- Status: **Current** status of the decision which, should be **accepted, rejected or deprecated**.
- Consequences: **What** has become better or worse because of the decision.

Please find more, complete architecture decision records in A.2.

To make sure that the whole group had the latest information; We used several channels on Slack to keep everyone updated. Such as the **iOS channel** was the place containing all updates related to the iOS application development, this is shown in figure 2.

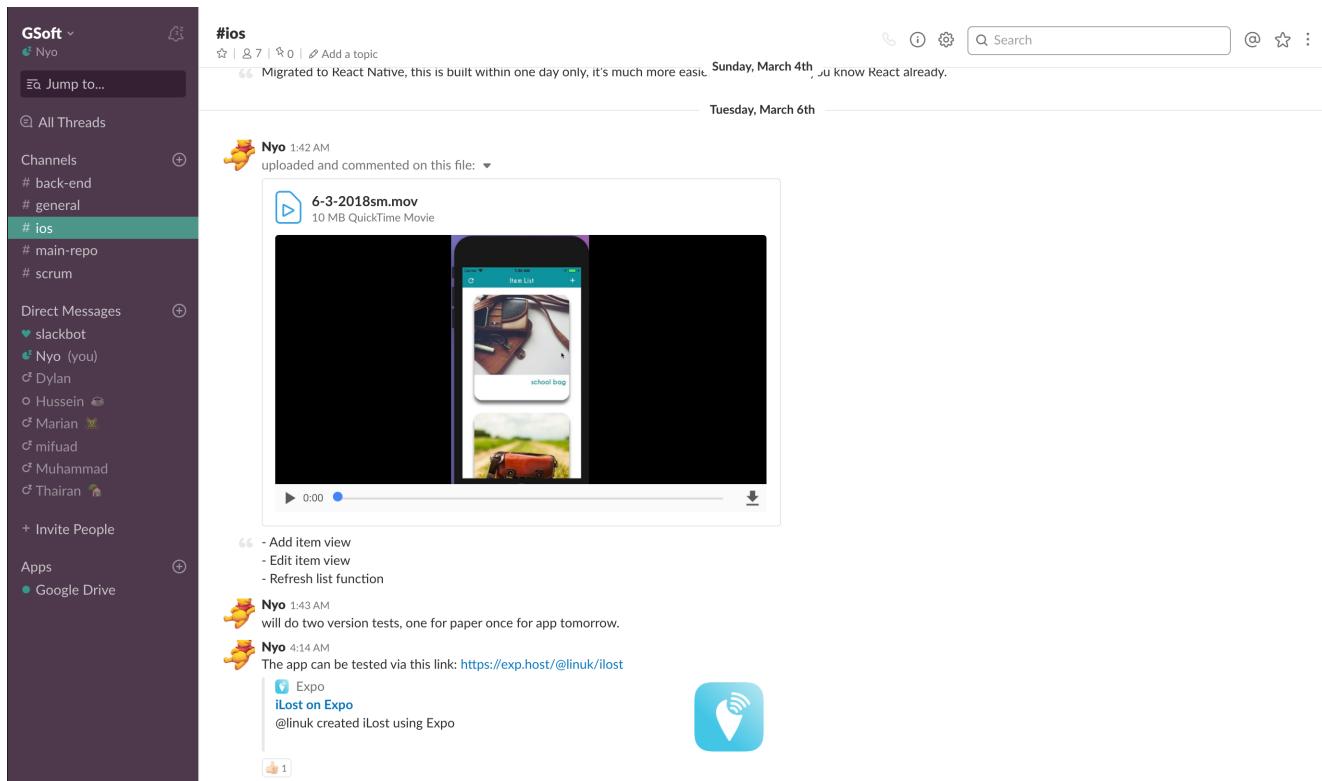


Figure 2: Slack iOS Channel

2.3 Agile Development

Our team tried out two agile development methodologies, **Scrum** and **Kanban**.

2.3.1 Scrum Development

Since we could not actually devote all of our time to develop the application, it would not be reasonable to do the daily scrum and have a meeting day-to-day for all of us. So in the beginning, we tried to set up a routine to simulate the daily scrum with Slack.

We defined 3 days per week for sprint sections to take place and whenever a sprint was finished, the team needed to answer three questions in the Scrum channel on Slack:

- What did I complete last time that it contributed to the team meeting our sprint goal?
- What do I plan to complete this time to contribute to the team meeting our sprint goal?
- Do I see any impediment that could prevent the team or me from meeting our sprint goal?

Unfortunately, not every team could follow the sprint process and make any progress every three days. Sometimes a team might work for a sprint then stopped for two sprints since other modules might have a deadline or coursework. Hence, this scrum process was not actually conducted properly and stopped after few weeks after started. The example records show in figure 3.

 **Nyo** 12:27 AM
18/1
1. Refactor and divide the project tasks.
2. Complete the iOS app UI design.
3. The design have some customised UI components, which might need more time to develop on. (edited)

 **Nyo** 12:17 AM
23/1
1. Complete the iOS app UI design
2. Write the user stories, backlogs and acceptance criteria.
3. Not sure how are we going to implement the sprint thing.

Figure 3: Slack Scrum Channel

At the middle of the development stage, our supervisor suggested us to meet 2 hours a day and 2 to 4 days a week, work and team and engage the team building, since the progress tracking form showed that the working hours were really unbalanced and some people apparently did not put enough effort into the project.

We made a daily sprint schedule, shown in figure 4 and followed by the time to do work together. Everyone should follow the schedule and spend at least two days a week to work together as a team. It went well and most of us were able to follow the schedule. We worked in RHB306a, 35 Cafe or the Whitehead building lab. But since the strike started, the daily sprint stopped again because not all of us were coming to the campus due to the long distance between home and the university.

	START	END	Dylan	Marian	Mifud	Hussein	Thairan	Jheng-Hao	
Monday	14:00	16:00	X	X	Δ	X	O	X	
Tuesday	17:00	19:00	O	O	O	O	X	O	whitehead building lab: 18:00 to 19:00
Wednesday	14:00	16:00	Δ	X	Δ	X	O	Δ	
Thursday	11:00	13:00	X	X	X	O	O	O	
Friday	17:00	19:00	O	O	O	O	X	Δ	

Legends		Availability
O		Sure
Δ		Either one
X		No

Figure 4: Daily Sprint

2.3.2 Kanban

Apart from the Scrum, Kanban was also implemented with the backlogs in our development to demonstrate the development progress. We tried to keep our project management tool simple and easy to approach, so we not only used GitHub for version control but also for the project feature, which we used it as the Kanban. The iOS development Kanban was shown in 5.

The screenshot shows a GitHub project page for 'iLost iOS / iLost-main'. The main navigation bar includes 'Code', 'Issues 11', 'Pull requests 0', 'Projects 6' (which is highlighted), 'Wiki', 'Insights', and 'Settings'. The 'Projects' section displays a Kanban board with five columns:

- Backlogs**: Contains cards like '7-1-1 Specify which data produced by the system is to be considered sensitive' and '6-3-2 The case architecture should be made with harmless in mind'.
- To Do**: Contains cards like '5-3-1 User can turn off password protecting feature in Setting page.'
- In progress**: Contains cards like 'Implement User interface #63 opened by linuk iOS App Develop' and 'Integrate UI. #84 opened by linuk iOS App Develop'.
- Done**: Contains cards like '5-5-1 User can use 4-digit or Touch-ID as input to login.' and '5-5-3 App redirect to item list if the login authentication succeed.'
- Deprecated**: Contains cards like '3-3-2 User should be able to navigate to from item list to the item history.'

Each card includes a description, a link to the GitHub issue, and the user who added it ('linuk'). A search bar at the top right allows filtering cards.

Figure 5: iOS app Kanban

Our Kanban contained five columns which shows in Table 2:

Column	Description
Backlogs	The task is scheduled to be developed, but it is possible to be assigned deprecated if it is no longer needed.
To Do	The task is going to be developed.
In Progress	The task is currently developing.
Done	The task has been completed.
Deprecated	The task no longer needs to be developed.

Table 2: Kanban Columns

Apart from the normal columns: To Do, In Progress and Done, we also added **Deprecated** to store the features or tasks that did not fit the needs anymore. Kanban gave a nice and clean overview of the current developing process, especially for people from other teams.

2.3.3 Behavior Driven Development

In order to produce the minimum viable product(MVP), we followed the Behavior Driven Development(BDD) to develop our product. To implement this methodology, we wrote the backlogs of requirements specification at the beginning of the development stage. These requirements provided us with an overview of what was more urgent in what stage.

For example, displaying the item's position and historical locations was one of the most important functionality, so its' priority was **Must** and needed to be finished at the **MVP** stage. On the other hand, navigating to the item by triggering Google Maps was helpful but might take longer to develop, so the priority was **Should** and it should be developed at the **Final** stage.

Here was our development process of BDD:

1. Requirements specification Review the backlogs and pick one from them to work on based on the priority and the estimated development time. You can find the details of the backlogs in section 2.4.1, Development Process - Backlogs.

2. Design Review the how was the data storage, data flow and the user interface design, then design how should this requirement be fulfilled. Such as displaying the item list in the item list view, we would need to implement two view components, an item list cell view and an item list table view. The item's properties data, including items id, name and photo, would be stored in the table view's state and the list cell would receive the props from the table view and render the item's properties.

3. Development Put the design into actual codes.

4. Integration Integrate the new view component with previously developed view components. Such as connecting the item list view with the item details view, where users could see its' location with a map.

5. QA reviews Conduct the quality assurance of the functionalities of latest and previously developed components. Please find more details in chapter 5, Quality Assurance.

6. Usability tests After several developments, we would bundle them as a new version of our mobile application. Then do the usability tests to test if any of the components, user interface or user experience could be improved. Please find more details in chapter 3, Formative Evaluation.

7. Maintenance After the usability tests, we did some improvements based on the usability testing results.

This was a cycle process, after step 7 we went back to step 1 and kept going.

2.4 Development Process

2.4.1 Backlogs

Based on the user stories we made in the proposal, not only did we add more but also wrote sub-user stories and acceptance criteria's for each of them. We had a list contain backlogs stored in Google SpreadSheet, which contained these columns:

Column	Description
User Story ID	The user story ID or sub-user story ID, a user story ID should be 1, 2, 3...n, and sub-user story should be 1-1, 1-2, 1-3, ... 1-m where 1 is the parent user story.
User Story	A brief description of the user story, starting with "As a user I want to ..." to describe what are the users' needs. Then followed by "so that ..." to explain why users need it.
Backlog ID	An ID number for the backlog, a sub-user story may contain more than one backlog to satisfy the sub-user story. If a sub-user story ID is 3-5, then the task ID will be 3-5-1, 3-5-2, 3-5-3, ... 3-5-p.
Acceptance Criteria	A description of the backlog which describes how the mobile application or the tracker would satisfy the user stories.
Priority	The priority of this backlog, should be Must , Should or Could .
Dev Days	The estimated developing times in days, 8 hours counted as one day.
Phase	In which phase should the backlog be finished, in our case was either the MVP or Final .
Process	The current developing process, which should be To Do , In Progress , Deprecated or Done .

Table 3: Backlogs Columns

For example, the user story 3, **Track Item: User uses App to see the tracking list and track Item.** This section has 7 sub-user stories, which shows in table 4:

User Story ID	User Story
3-1	As a user, I want to view the tracking item list so that I can view my item and look up more details if I want.
3-2	As a user, I want to activate or deactivate the tracker so that I can save my mobile and the tracker's batteries.
3-3	As a user, I want to see the location history in a map format of the item so that I can track/look for it, if it's lost.
3-4	As a user, I want to navigate me to my item so that find it quicker.
3-5	As a user, I want to edit my item's details so that I can change the tracking item.
3-6	As a user, I want to delete the item so that I can stop tracking the item for good.

Table 4: Backlog Example: User Story 3

Take sub-user story 3-1 as an example. In order to satisfy this user story, we had 4 backlog and acceptance criteria's to meet the requirements. Each of them had their own priority, estimated development days, phase and process. So this backlog will look like figure 6:

User Story ID	User Story	Backlogs					
		Backlog ID	Acceptance Criteria	Priority	Dev Days	Phase	Process
3-1	As a user I want to view the tracking item list, so that I can view my item and look up more detail if I want.	3-1-1	Item list user interface should be simple, clear and easy to navigate.	Must	2	MVP	Done
		3-1-2	App should get item data from mobile database	Must	1	MVP	Done
		3-1-3	App should fetch lastest locations of all the tracking items when User visit item list page.	Should	1	MVP	Todo
		3-1-4	App should display if any of Items is lost clearly.	Must	2	MVP	In Progress

Figure 6: Backlog Example: Sub-user story 3-1

The full backlog can be found in the appendix A.1 and it was used to create our Kanban showed in figure 5, which kept the developing process recorded easily to follow.

2.4.2 Progress Tracking

To record our progress, we kept using the same progress tracking form as the last term, and this term we did some improvements:

- Lock every week: To prevent any team members from modifying the resource hours dishonestly, a range of the form was locked every week. Only records within recent two weeks were allowed to be added.
- Status documented in more detail: To increase the traceability and convincing evidence, each resource hours commitment was asked to provide a more detailed description. Instead of *reporting writing*, it was improved to *Final report: Formative Evaluation - mobile application test*.

Please find the full progress tracking form in appendix A.4.

Until March 16, the total working time during these two terms is **581.9 hours**. All these hours contributed includes all lab sessions, supervisor meetings, daily sprint and self-independent

working time. 58.7% was contributed by Hussein and Jheng-Hao as shown in figure 7. The top contributor is Jheng-Hao with 222.95 hours contributed, while the last is Chin with 26.5 hours recorded. The line chart displays that the commitment became dramatically different since the start of the second term. Most of the team members stopped to work in the second term, excluding Hussein and Jheng-Hao who still kept working continuously.

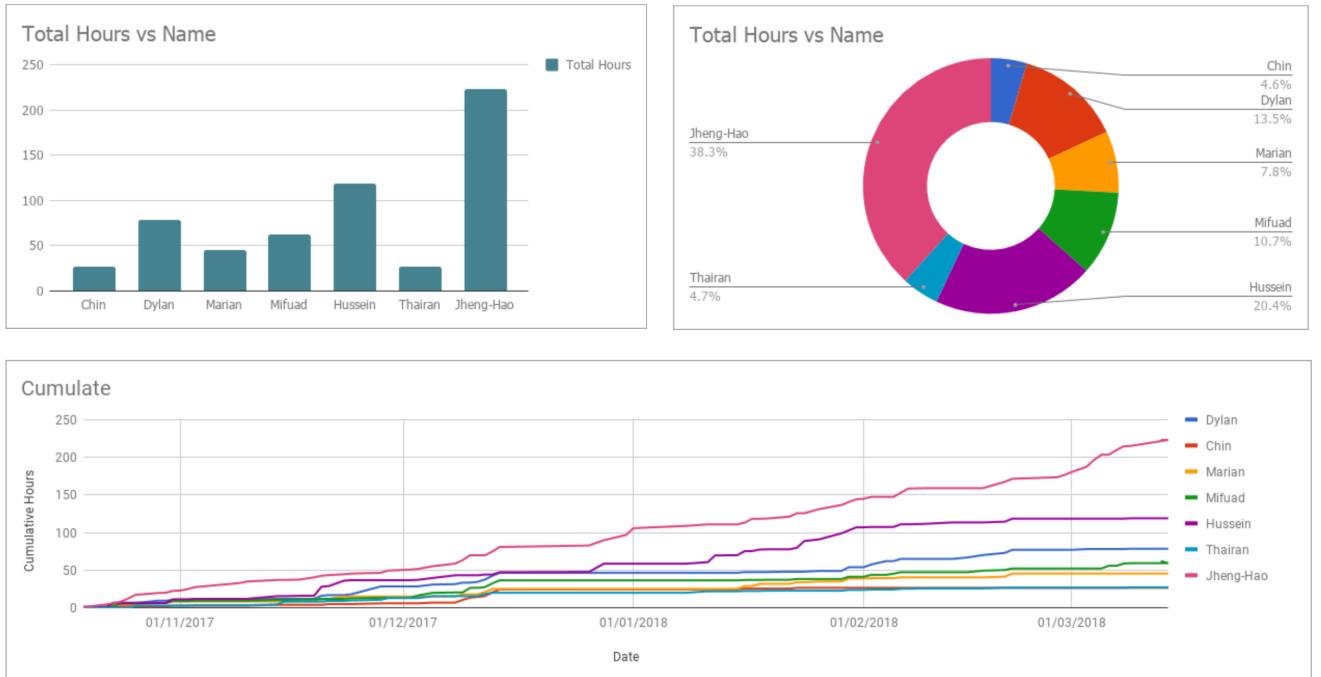


Figure 7: Progress Tracking Diagram (until March 16th)

3 Formative Evaluation

3.1 iOS App Evaluation

3.1.1 Objectives and Questions

We wrote quantitative tasks to test the usability of our mobile application[3]. The purpose of the tests was to test if the participants can actually finish the task with the application, and we could learn from the process and observe how users used our application and improve it if any issues or confusion were raised.

Task ID	User Goal	Task	Scenario	Developing Status	Can be tested		
					v0.10	v0.11	v0.12
1	Log item in the app	Add my bag as a new item and pair a tracker in the mobile app	You are planning to find a solution to prevent losing your bag in public again, so you buy a iLost tracker online now and download the free mobile app in your phone. Go to the app to add your bag in the tracking list.	Done	Y	Y	Y
2	Check items status	See if all my bag item is safe.	You are successfully log your item and pair the tracker with the mobile app. Go to the app to check if your bag is safe.	Done	Y	Y	Y
3	Check item position	See my bag's current position	You just come back from a chill night in a bar, you realise that you forget your bag at somewhere until arriving home. Check where is your bag now.	Done	Y	Y	Y
4	Check item position records	See my bag's 10 minutes before's position	You have checked your bag's current position and feel weird why is it in a place you have never been before, so you decided to see how it end up there. Check your bag's location ten minutes ago.	Done	Y	Y	Y
5	Trigger native maps app navigation with current position	Navigate me to the my bag's current position in a public transportation way.	You are really worry about where is your bag, because it seems like someone has taken it to somewhere, and inside your bag are all your important belongings. Go find your bag with the app's navigation feature.	Done	Y	Y	Y
6	Trigger native maps app navigation with history position record	Navigate me to the my bag's 10 minutes before's position on foot.	There is something missing from your bag, it might be left outside of the bag. Go and find your bag's previous location 35 minutes ago.	Done	Y	Y	Y
7	Edit item name or photo.	Attach the tracker to other item.	You just bought a new bag and you would like to attach the original tracker to it. Go to one of the tracker view to edit the item's name and photo.	Done	Y	Y	Y
8	Receive notification	Receive notification once my bag is lost.	You receive an notification since the app detecting your item is too far away from you. Check the notification and find where is you lost item.	Done	N	N	Y
9	Setup password	Setup password for the app	Now you know how iLost can help you find your lost precious, but the app need to be kept in private otherwise other people can the app to track you! So you decide to prevent someone else to use the app by setup a password in the app.	Done	N	N	Y

Figure 8: Usability test task list

Column	Description
Task ID	Identify the number of the task.
User Goal	What is the objective we need users to perform.
Task	The process in terms of completing the goal.
Scenario	A setup scenario for engaging testers to use the application in a real-life case.
Developing Status	Current latest developing status of the functionality of this task, it should be To do , In Progress and Done .
Can be tested	Whether the functionality of the application is ready to be tested in each version that was created.

Table 5: Usability test task list columns

Figure 8 demonstrates our tasks and goals. Table 5 shows what the columns stand for.

3.1.2 Participants, Location and Setup

According to Jakob Nielsen, testing 5 users in a usability study could find almost as many usability problems as testing more participants[2]. Following this theory, our application was tested with 5 participants for each version that was created. The study was taken place in the library of Goldsmiths, University of London, and the participants were the students who used to bring a bag to the campus daily. In total we have conducted three versions of the application.

3.1.3 Methodology and Measures

Firstly, we explained what was our project about and asked participants to sign up the consent form which can be found in appendices B.1. During the test, the participants were provided an iPhone with the application built-in to test. An observer would guide them through the tasks and the scenarios, then took notes of how the participants reacted to the application. For each task, the observer would record if it was successfully completed or failed. The records would help us to build the success rate diagrams which helped us to understand which usability needed to be improved[4].

3.1.4 Evaluation

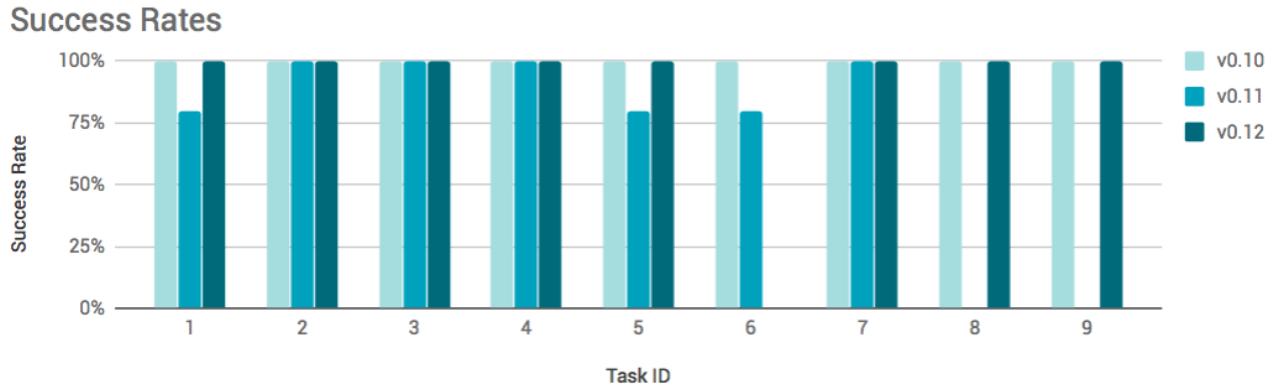


Figure 9: iOS App Success Rates Diagram

v0.10 The tested subject was the application prototype built in **Adobe XD**, which was one of the best design tools to test the prototype. Thanks to the well-designed user interfaces, the goals were easily achieved and the tasks were all successful. The results only proved that the user interfaces guided the users to the right view, however it could not actually reflect on the usability of the real application. After all, Adobe XD could only let users walk through each view by clicking, while an actual iOS application would support swipe or other gestures. It was more like a paper prototype usability test.

v0.11 We benefited most from this test since this test was the native mobile application, where the participants can actually use it like other applications. This version was built in React Native and tested in Expo which is a tool and service which we used to build the mobile native application with React Native. During this test, we received several comments towards the **add item view**. For example, the camera icon in that view was originally used as a button. But some of the users could not really regard it as a button but a decoration since it was colourful. Also, the placeholder of the item name field was **Item Name** instead of a prompt message which confused some of the participants as well. So after this test, we resolved the issues and the differences show in figure 10. Task 8 and 9 were not developed completely at the moment so there was not any record.

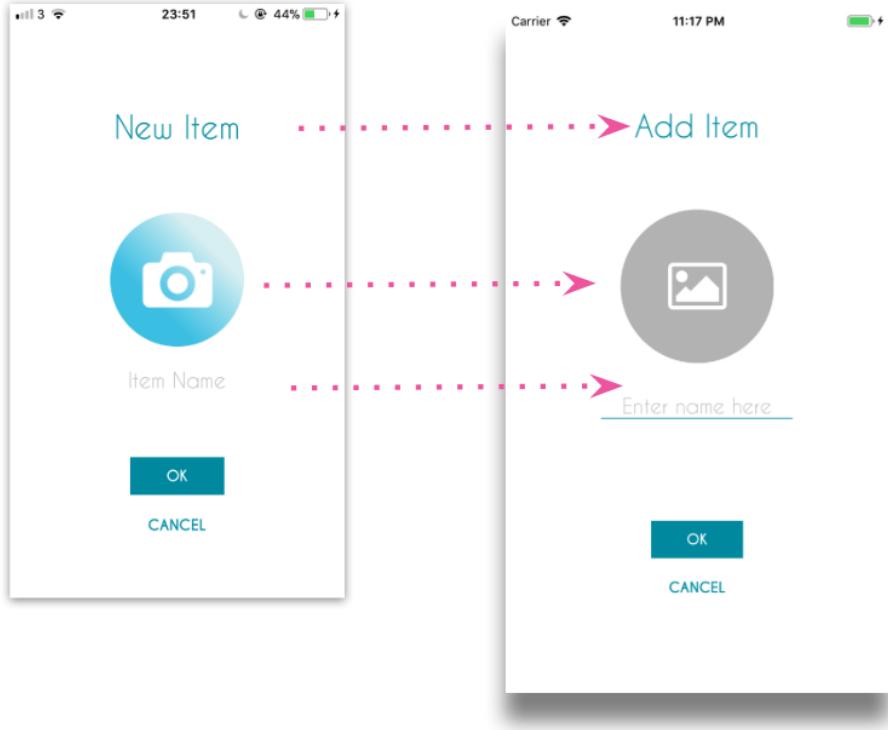


Figure 10: Improvement of the add item view

v0.12 After the previous test, we not only improved some of the user interfaces but also removed some of the features. It is worth notice that there is no record of task 6, which is **navigating to previous locations of the item's location history**. The reason we removed this task was that this goal was not really helpful. The participants commented that it was not beneficial to navigate to the previous locations, they cared about the current location of the item more. So thanks to the feedback, we removed task 6 and eliminated the functionality of navigating through past location history which made the application more simple.

3.2 Conclusion

We found it really helpful to do the testing regularly, because what we thought the users' needs might not be true, it was quicker to hear their actual voice by conducting tests with them. In terms of the cycle of agile development, it might be really practical to conduct the tests every half month, to make sure we are on the right track and meet the users' needs. Due to the time limit, we were not able to conduct the tests for the tracker unfortunately.

4 Design and Implementation

4.1 Design Stage

Our final design for the iLost tracker consists of an Android app and a Raspberry pi with the attached Hologram Nova. The User interacts with the Android app, whereas the Raspberry pi provides the short-range capability of the Bluetooth and the Hologram Nova cellular modem attached to the Raspberry pi provides the long-range capability.

Short range (figure 12) is defined as less than 30 metres from the target and long-range (figure 11) is defined as above 30 metres to 400 km and beyond.

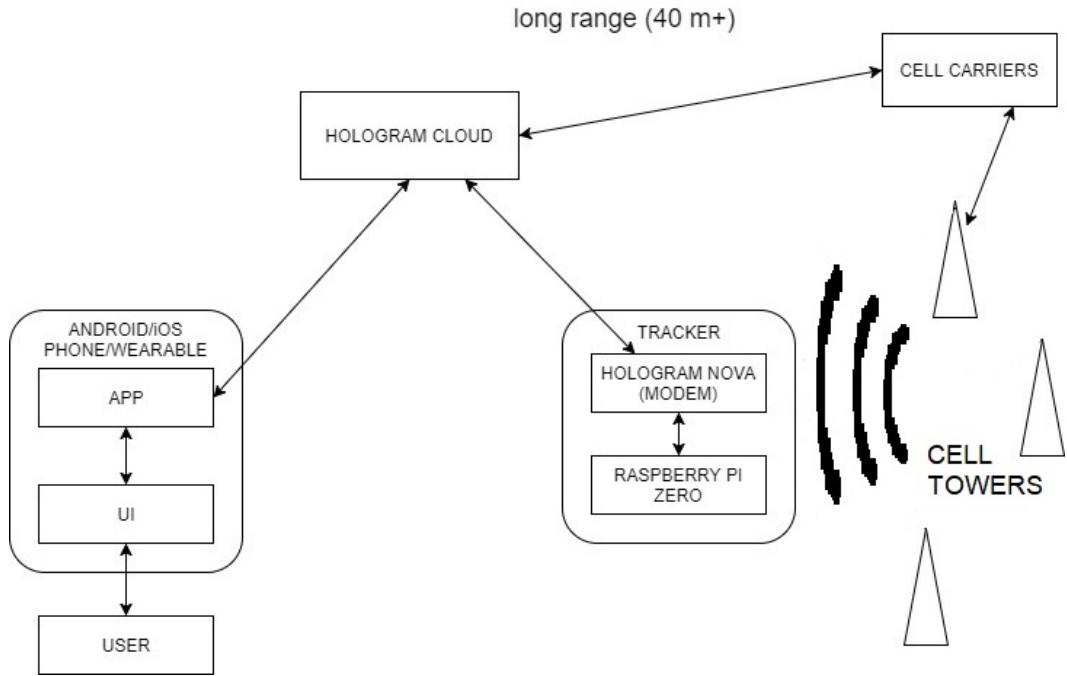


Figure 11: Tracker concept 2.0 at January - long range

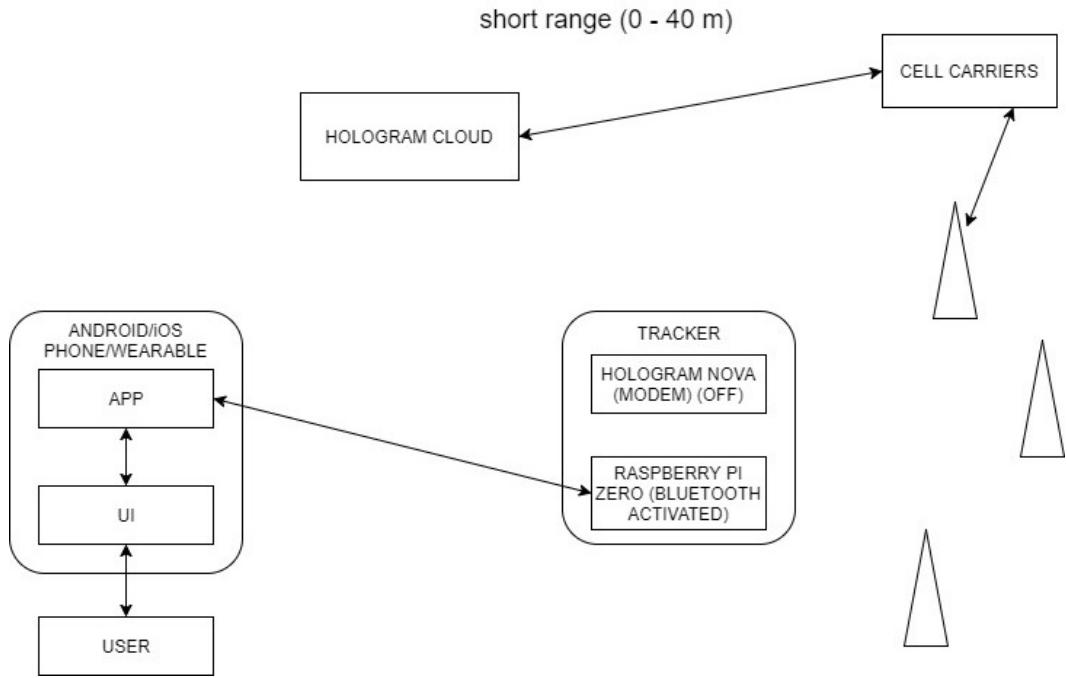


Figure 12: Tracker concept 2.0 at January - short range

This Hybrid approach, compensates for the weaknesses of the Bluetooth range and the weaknesses of the accuracy of the Hologram Cellular modem, by leveraging the short-range accuracy of the Bluetooth and extremely long range of the cellular modem. (There are cell towers in nearly every part of the world.)

In our final design, considering the strong demand from users to have the tracker as small as possible (similar to the Tile Mate in figure 13). We endeavoured to ensure the tracker is compact as possible.

We also initially designed the Tracker to be server-less. In this way, the tracker would communicate and share information directly with the users iLost app.

Moreover, we set out to have the tracker adapt to the users need. The iLost tracker should be able to be smart enough to understand that a user has left behind an item regardless of the item type (luggage, handbag, laptop etc.) However, our design plan changed rapidly as we started implementing the tracker and as we consulted with more users and professionals (both in marketing and in engineering).



Figure 13: Tile Mate

At the computer fair, in which we consulted with various industry professionals, we realised that we should focus on a minimum viable product for the duration of this project. Certain product features such as having the tracker as physically small as a sticker (similar to the Tile Mate) is not possible in the scope of the project as we need specialist manufacturers, engineers and funding (we already looked at all the possible routes to making the tracker smaller). Most users clearly want a tracker that is as small as Tile Mate, so we looked at the next viable use case and the minimum physical size of the tracker that can fit that viable use case, this was baggage use case.

The posters for the computer fair can be found in appendix C.1.

It is feasible in the time frame and available resources in this project, to build a physical tracker that is small enough (insert dimensions here) to fit in the users handbag, rucksack and travel luggage (the baggage use case). While also available to fulfil the user criteria in terms of accurate, long range tracking (over 30 metres).

At the design stage, we initially designed the tracker to be server-less i.e. The physical tracker solely communicates via cellular data to the app on the users phone, reducing cost and increasing security for the user. However, for the cellular modem to function properly, it must communicate with the Hologram Nova cloud servers, this is so that the Hologram can negotiate with multiple phone carriers, this functionality allows the user with one sim card to be able to have a tracker that can triangulate and transmit data anywhere in the world which has cell towers.

Currently the tracker communicates with the Hologram Cloud servers, which in turn communicates with the users app.

Also, in the design stage we realised that our tracker can be developed into an enterprise framework. If our product is successful at item tracking for consumers and is fully tried and tested, this same technology can be used for enterprise applications, such as asset tracking, shipment tracking, even livestock tracking. However, this is not in the scope of this project.

4.2 Implementation Stage

We initially designed the user app to be hosted on the Android platform, instead we implemented the user app on both the iOS and the Android platform. We did this for two reasons. Firstly, from market research shown in figure ??, roughly 53% of users do not have an Android smartphone. Secondly, developing the Android app allows us to have a plan B, in case we cannot develop the iOS app in time or if there are compatibility issues with the tracker. The Android and iOS app both have different programming languages and paradigms.

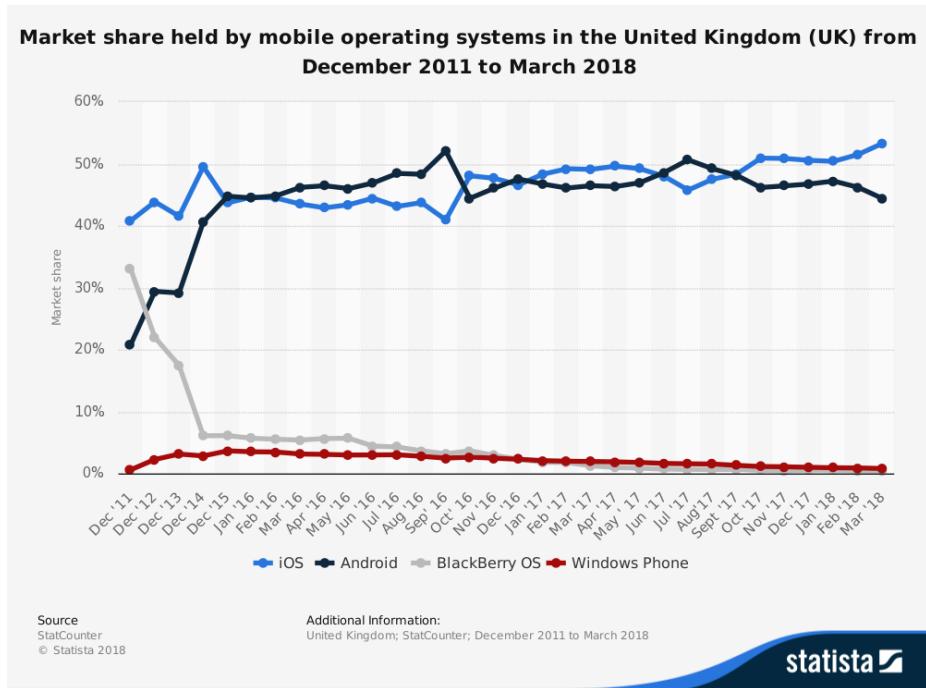


Figure 14: Mobile OS Market Research in UK [5]

The reason why we developed an Android/iOS app instead of a web app, is that as part of the iLost design, the user needs notifications in real-time (when an item has been left behind). Real-time notifications are not currently possible with web apps.

Another significant change to the iLost tracker was the discarding of the Bluetooth function. Initially, the iLost tracker was supposed to use the Generic Bluetooth adapter on the Raspberry pi and the users smartphone.

The Bluetooth function is needed for the tracker at short range (less than 30 metres) to remind the user if they have left an item behind, if an individual recently just stole their item or for tracking down the location of an item (to a range +- 5 metres). This is after the long-range function (i.e. the cellular modem) locates a rough location of the item (30-40 metres from actual location)

In a situation whereby, the user is greater than 30 metres away from the physical tracker but requires a more precise location of the tracker, the Bluetooth function was supposed to fulfil this use case. By measuring Bluetooth signal strength coming out of the tracker, (2.4 to 2.485 GHz radio frequencies) the user can determine how near they are to the tracker and thereby their lost/stolen item.

However, although the Bluetooth concept is simple, as many developers (we talked with one personally in the department) implementing reliable Bluetooth functionality is problematic (looking at most Bluetooth products on the market, many products have connectivity problems).

The main disadvantage that led us to cancelling the Bluetooth function, is that in order to support the iOS platform for the users, the iLost tracker must comply with MFi (Made For iPhone) program. The MFi program restricts the types of devices that can connect to Apple products. The iLost tracker is a Raspberry pi with an attached cellular modem, the Raspberry pi is not certified under the MFi program. This means the iLost tracker Bluetooth function (its a Raspberry pi zero, which has a Bluetooth module by default) cannot interact with the user app and we cannot release the iLost tracker for the iOS platform.

Subsequently, we tried to implement the iLost Bluetooth feature with the Android app. Although, the Android platform did not have a compliance program like the iOS, there were other serious issues. We simply could not find the functionality to be able to measure the Bluetooth signal strength emitting from the iLost tracker. We assumed that the mac address of the iLost tracker would always be the same, but this is not correct. After Bluetooth 4.2 was released, the mac addresses of any Bluetooth device is constantly randomised. Because the mac addresses (we need this address to communicate with the iLost tracker) is randomised we cannot from Android app side be certain which Bluetooth device is the iLost tracker.

We also looked at other Bluetooth products on the market, such as most headphones. They were also plagued with stability issues (such as signal not receiving or pairing breaking), if we were to implement the Bluetooth feature we would need experienced developers who specialise in implementing reliable Bluetooth communications between user apps and physical computing devices. Unfortunately, there is also no library or service (like Hologram) which simplifies the Bluetooth stack.

Moreover, we experimented with other Bluetooth based technologies, to find a replacement. We tested and built mock-ups using the estimote beacon (a Bluetooth tracker, used in upcoming retail and asset tracking), this did not fit our use case, as it increased unnecessary cost and complications to the user. Another novel Bluetooth tracker which failed the use case was the puck.js (a small Bluetooth tracker that can programmed using JavaScript) although a novel concept, we did not possess enough technical skill to be able to program the Puck to detect it to read Bluetooth signals specifically emitting from the raspberry Pi.

Our next step, was to implement the Cellular functionality of the iLost tracker. The cellular functionality consists of a cellular modem (called Hologram Nova) provided by the company Hologram. The Hologram Nova, via USB is attached to the Raspberry pi (for testing and early prototyping we used the Raspberry pi 3 model B) and this is what we refer to as the iLost tracker. We managed to establish a protocol to communicate with the iLost tracker.

This protocol consists of sending HTTP requests to the tracker, the tracker then responds via HTTP post with the data back to the sender.

At first, we sent HTTP requests via our computers (using the Postman application) to the iLost tracker, we received responses that showed that the communication method functioned.

After, we checked that the cellular modem method of location worked and that it could return useful data. The Hologram Nova uses nearby cell towers to triangulate its location. It did work, but the accuracy was not satisfactory, on average location was 40 metres away from the bag we were tracking.

Even though we could send HTTP requests to the tracker as well as sending simple worded messages (such as RL for request location), we could not request the tracker to send its location back to the sender.

So, we needed to solve this issue. At first, we tried to set up an apache web server on the Raspberry pi, open a specified port on the server and send messages to that port. We built a python script that would reply to any specified message we sent to the server. This did not work, any messages we sent were not received by the server. What we didnt realise was that when we were sending a message to the tracker, the payload was being sent to the Hologram Cloud servers, these cloud servers would then forward the payload directly to the tracker.

We could not find a way around the Hologram cloud servers, in fact the Hologram cloud servers were essential. They reason why their cloud servers are needed is that in order for us to communicate with the Hologram nova (cellular modem) and be able to triangulate the location of the tracker (i.e. Raspberry pi), the Hologram service as a whole needs to negotiate with multiple mobile network carriers. Being able to triangulate location using cell towers and communicate with a cellular modem internationally at a regular price (it currently costs us \$1 a month) is only possible currently using the Hologram service. Previously developers had overcome many administrative, legal and economic issue in order to work with cellular networks.

Through all our testing and use the Cloud servers did not fail and reliably delivered any payloads we sent to the tracker. Without the Apache server, we then used Hologram SDK (which includes a Command Line interface) to open a server at port 4010 (on the Raspberry pi), any messages we sent via HTTP were now received (we could view the message on a terminal) we then used Bash scripts to automate useful responses. For example, if the message RL was sent (Request Location) then the tracker via the scripts would triangulate its location, store this data (longitude, latitude, altitude) as string data and send this back to the sender. Scripts were also created for the tracker

to shut itself down, power itself up, disconnect from cellular networks (use cases such as the user traveling on an airline).

We also tried to host a server on Igor instead of the Raspberry pi, however we found a more efficient solution, this was to leverage the Hologram Cloud servers. The Igor servers are not available worldwide and offered limited scalability.

The reason why Bash scripts were used instead of python is that there is native support for using CLI commands provided by the Hologram SDK. Another advantage of using Bash scripts were that we were able turn these scripts into daemons. Daemons are process that run without user supervision and in the background on Unix/Unix-like operating systems. The cron (specifically crontab) program is used and whenever the tracker is turned on, all of the Bash scripts that enable tracker functionality run as background processes, lowering the power consumption of the tracker.

Using a Bash script, and using inbuilt Linux programs such as grep, cron, is in line with the Unix philosophy of small programs which do one specific function efficiently, being chained together to build a larger program.

Even though we could request the tracker to send its location back successfully, we still needed to be able to store this data, so the user app could access it and use it. We researched into the possibility of running a small server on the user app, which could receive this data. This is not possible; Any server must operate outside of iOS and Android platform.

We then decided to use the Amazon s3 servers which are scalable (servers can be replicated or moved to any AWS region), reliable (good track record, enterprise grade), available worldwide and inexpensive. The tracker now sends location data to our s3 server. We were then able to have the user app request location data from the s3 server and display this data/their location of the tracker to the user.

We also encrypted this data using AES-256 servers side encryption keys, each record is encrypted with a unique key, even the key itself is regularly encrypted. This service is provided by Amazon.

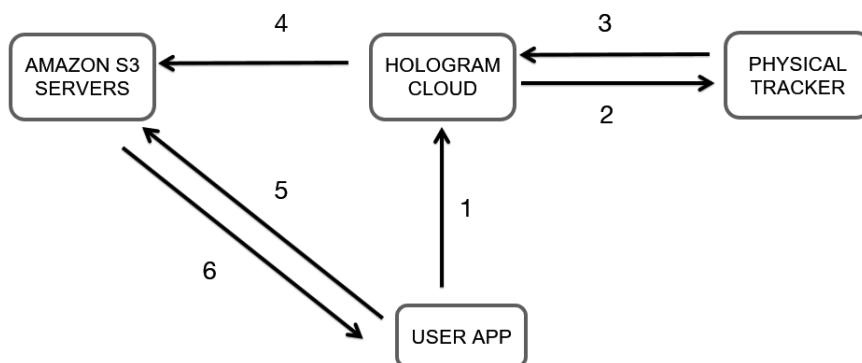


Figure 15: Current Stack

Post-Snowden age, we understand users are concerned about privacy. All data that is stored on Amazon servers are automatically encrypted by AES-256 servers side encryption keys. We only store longitude, latitude and time data, for 14 days. It saves us space and cost, to not store data permanently.

Location data such as longitude and latitude is metadata, its ambiguous not content that can identify the users personal information. The purpose of the Amazon server is only required to provide data back to the user. Also, each tracker has an unique ID number, personal information is not identifiable from this data.

We cannot guarantee compliance with UK Data protection act, due to the legal complexity of having the Amazon operating servers globally as well as the Hologram Nova having to operate with many network carriers globally. Ideally, with more funding we would be able to consult with legal professionals.

Next, we needed to create the UI for the iOS app and Android app. We approached this by keeping everything as minimal and simple as possible for the user. The user would take a picture of the item they want to track and assign a category for it. When the user wants the location of their bag for example they would tap on the [locate] button, a Google map view of their item would show along with any information such as directions and distance.

However, we migrated from the iOS app being built in the swift language to React Native. React Native is a framework that allows developers to build mobile apps for both the iOS and Android platform using JavaScript and the React framework.

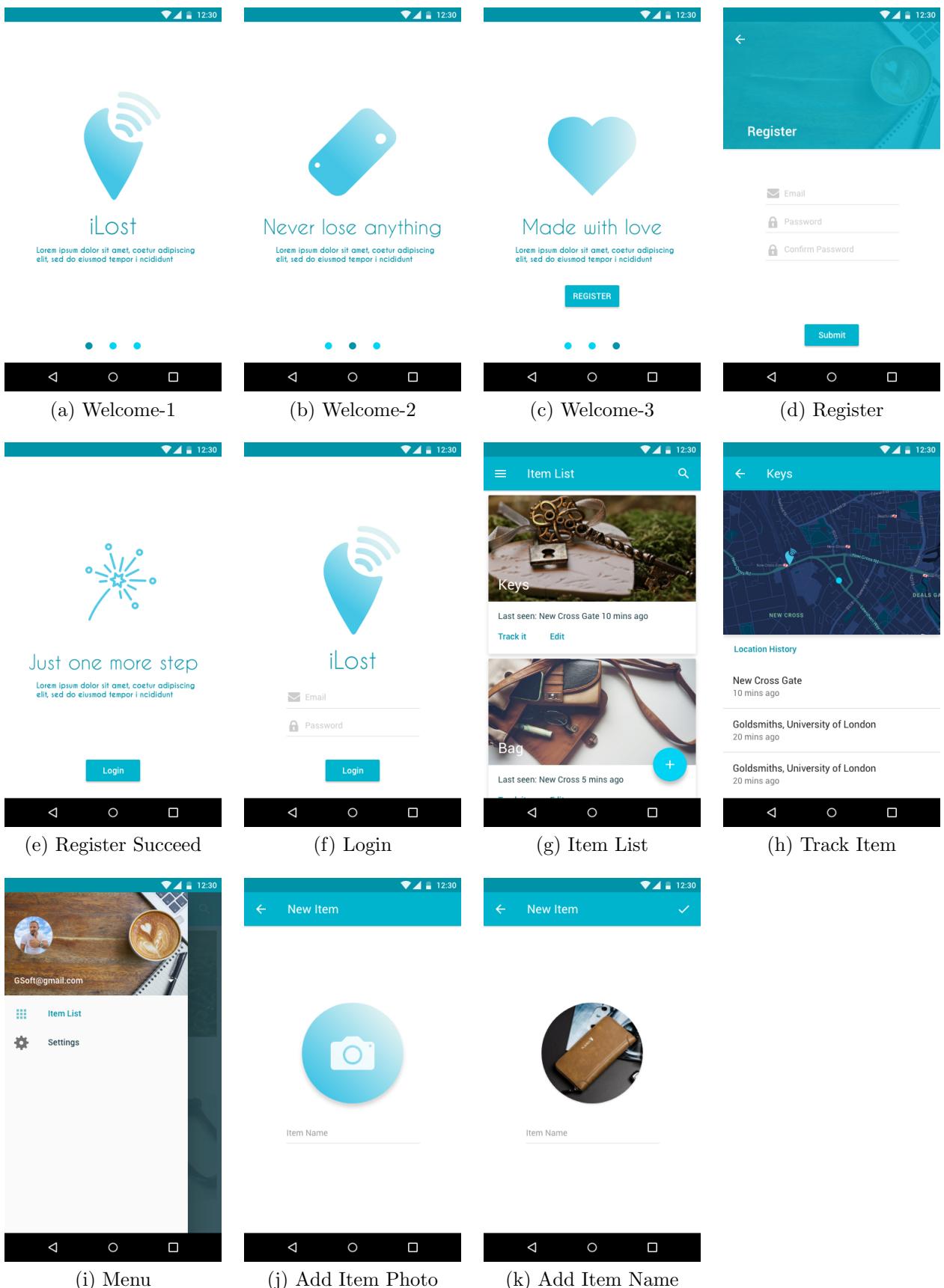
At the beginning of the project, our iOS developer team did not have experience with using the Swift language and associated frameworks. We spent 3 - 4 weeks building a basic application which only had a tab bar view to switch between pages, and only half of the user interface components were functional.

The Android app and iOS app user interface design can be found in figure 16 and figure 17.

Due to the limited development time, we searched for an alternative method that would reduce development time. We found that React Native and Flutter were good alternatives to develop native mobile applications. React Native was chosen because it was more mature than Flutter, which was relatively unstable as it had just been released. Unexpectedly, only three days were required to develop the same application, which the iOS team spent almost a month to build, in React Native. Switching to React Native was a good decision as it significantly shortened the development time providing us more time to do usability tests.

The iOS team also had previous experience in developing React web applications, where React shares the same component-based concept and reactive function programming paradigm with React Native. Also, React Native uses JavaScript which we were more familiar with, so it was relatively easier for us to develop the application.

The Swift API had many noteworthy changes from Swift 2.0 to 4.0 which is the latest version (we were developing with latest version), so whenever we encountered an issue, it was difficult to search for the solution for the latest version of Swift. The official documentation of Swift was also quite ambiguous, it lacked working examples. This led to a steep of learning curve when developing in Swift. On the other hand, React Native had better documentations and community support. The official documentation was explained clearly with good working examples, there were more community-built components that can be approached easily as well, this improved the developing experience and made it easier.



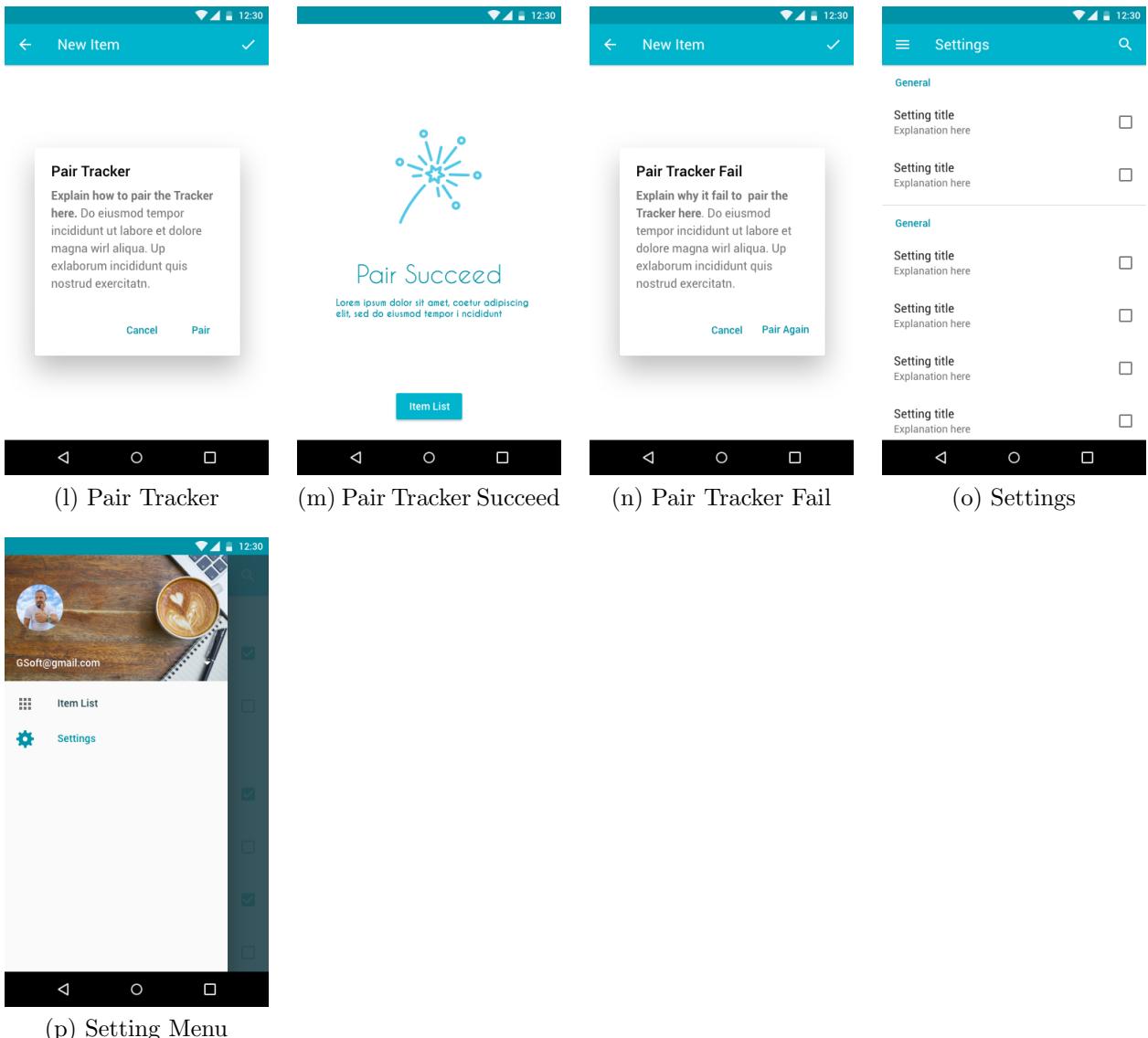


Figure 16: Android App User interface Design



iLost

Lorem ipsum dolor sit amet, coetur adipiscing
elit, sed do eiusmod tempor i incidunt

• • •

(a) Welcome-1



Never lose anything

Lorem ipsum dolor sit amet, coetur adipiscing
elit, sed do eiusmod tempor i incidunt

• • •



Made with love

Lorem ipsum dolor sit amet, coetur adipiscing
elit, sed do eiusmod tempor i incidunt

START

• • •



Register 4-digit passcode

We are protecting your privacy by
preventing someone use the app without
your permission, the code will be used to
launch the app in the future.

• • •

1	2	3
4	5	6
7	8	9
0		✖

(d) Register



Re-enter 4-digit passcode
Please enter the same passcode again.

• • •

1	2	3
4	5	6
7	8	9
0		✖

(e) Register Passcode



Just one more step

If you forget the passcode, you can always
use touch ID to unlock it, but first we need
your permission to set it up.

OK

SKIP



Congratulations!

Everything is setting up now. Let's start!

START



Enter 4-digit passcode

• • •

1	2	3
4	5	6
7	8	9
0		✖

(h) Login

9:41 AM * 100% ■■■■■

TRACK

Your items are safe


Bag
ON


Another Bag
ON

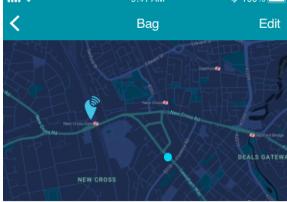
NEW
TRACK
SETTINGS

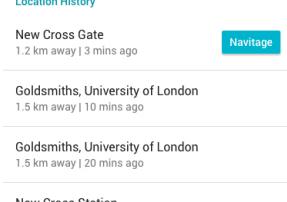
(i) Item List

9:41 AM * 100% ■■■■■

Bag

Edit


New Cross Gate
1.2 km away | 3 mins ago
Navitage


Goldsmiths, University of London
1.5 km away | 10 mins ago


Goldsmiths, University of London
1.5 km away | 20 mins ago


New Cross Station
2.3 km away | 25 mins ago

OK
DELETE

(j) Add a new Item - Track Item

(g) Register Touch ID Succeed



Bag

OK

New Item

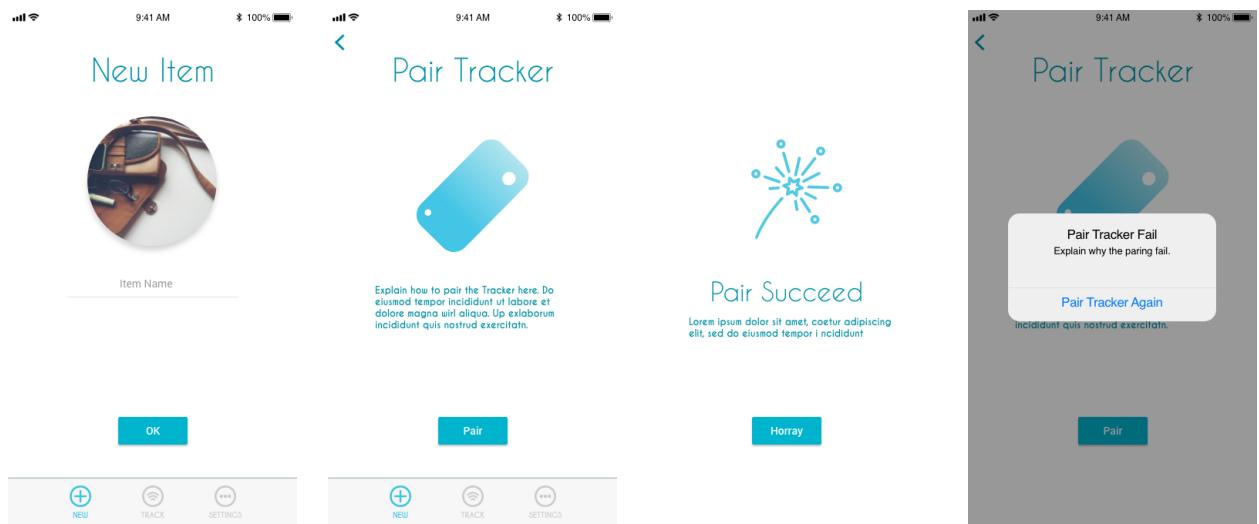


Item Name

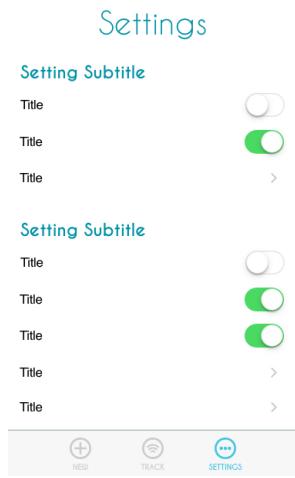
OK

+	NEW	TRACK	SETTINGS
---	-----	-------	----------

(k) Add a new Item - Edit Item



- (m) Add a new Item - Name (n) Add a new Item - Pair Tracker (o) Add a new Item - Pair Tracker Succeed (p) Add a new Item - Pair Tracker Fail



(q) Setting Menu

Figure 17: iOS App User interface Design

For our tracker to be useful and satisfy its use cases it had to be portable. When we were testing our tracker, which was a Raspberry pi 3 model b, it had to be plugged into the mains. We replaced the Raspberry pi 3 model b with the Raspberry pi zero. The Raspberry pi zero is 2x smaller than the Raspberry pi 3 and uses much less power and still uses the same operating system (so all the Bash scripts and configurations still worked).

We first tried to use small power banks to power the tracker (normally used for mobile phones) to test the average power consumption of the tracker. The tracker requires 130 mA. Using power banks is not practical for the user, so instead we tested lithium-ion batteries, these were slim and provided more power.

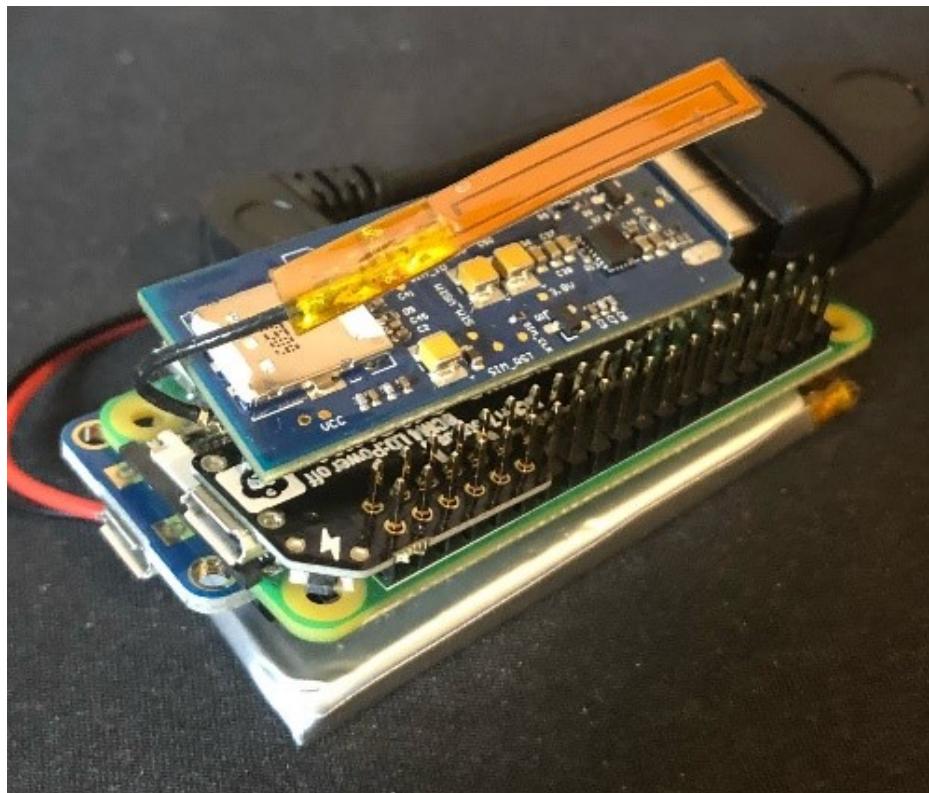


Figure 18: Tracker without case

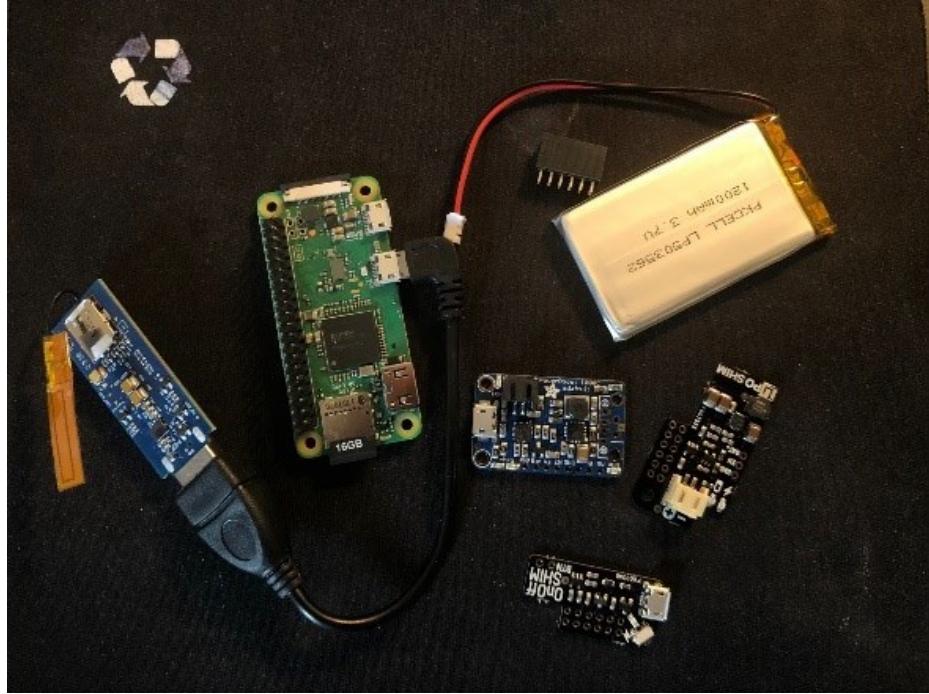


Figure 19: Tracker with all components

Moreover, a physical case which would fulfil the user use cases was needed for the tracker. Namely, it needed to be reasonably drop proof, strong enough to withstand day to day life, durable, simple and functional enough to be attached to a bag or another belonging. A prototype

was created using laser-cut cardboard, which was interlocking. As a first iteration, it was useful as it allowed us to show to the users how our Physical tracker would look, in terms of size (how it compared to other items an user may carry), this iteration was also low cost as it was made from cardboard. By creating the first iteration, we could also see dimensions required for the internal components of the tracker and how these components would behave inside an enclosed space.

Some members of team also undertook 3D printing training, the 3D model design shows in figure 20 and figure 21. Ideally, we wanted the case to be 3D printed, unfortunately we did not have enough time to progress beyond the cardboard prototype, which shows in figure 22.



Figure 20: Concept Tracker Case 1

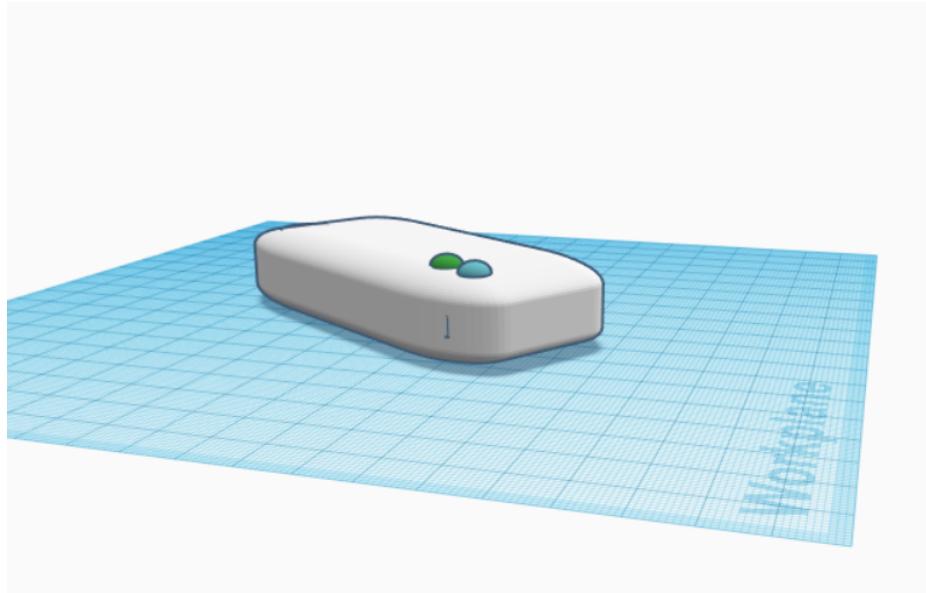


Figure 21: Concept Tracker Case 2

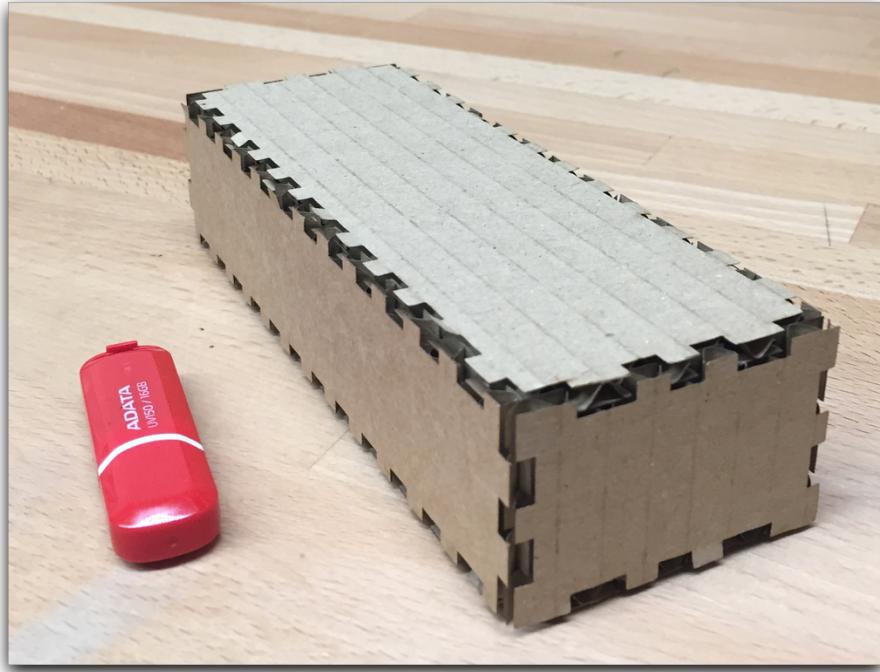


Figure 22: Concept Tracker Demo (Dimensions 5.6 x 3.6 x 15.5 cm)

5 Quality Assurance

5.1 Process

We conducted two types of quality assurance to ensure our mobile application worked as we expected, including black-box testing and white-box testing. The white-box testing focused on the functionalities, such as how the item list view should react when an add new item button was clicked, the black-box testing covered how a view component or a function should be rendered or implemented, such as a button should render a text on it based on what text state passed to it. Usually. The black-box tests were conducted after the white-box tests, since the functionalities could be functional only if the component worked properly.

Since the time limitation, only the iOS version was fully tested, but the test cases could be implemented in both version of the application.

5.2 Black-box Tests

5.2.1 Method and Environment

The black-box tests covered the functional testing of the use scenarios, they were developed by the acceptance criteria of the backlogs. Since our mobile application was not really complicated and complex, we did not implement any automated way to conduct the tests but manually merely. We chose **use case testing** as our approach to conduct the static tests, since it was the quickest way we could conduct the tests with existing resources.

The iOS application was written in React Native, it was imported and tested in the Expo [6] application on an iPhone 6. Expo is a mobile application where everyone can download others'

React Native codes and run on their iPhone or iPad, rather than paying USD 3000/year to subscribe to the Apple Developer Programme to use TestFlight [7], which is the official testing application owned by Apple Inc.

5.2.2 Unit Tests

Our use cases table was adapted from the format written by a software consultant company, IntexSoft [8]. The test cases contained the columns shown in table 6:

Column	Description
ID	Unique ID of the test case.
Backlog ID	Cross-reference of the Backlog ID.
Name	Name of the test case.
Goal	The detailed description of the purpose of the test goal.
Precondition	What need to be satisfied before the test case can be tested.
Success End Condition	The definition of how the test case considered as passed.
Field End Condition	The definition of how the test case considered as failed.
Trigger	How this use scenario is triggered.
Normal Flow	The flow of this test case, such as how the stakeholders reacted with the application.
Alternative Flows	The details of alternative ways to different successful ends.
Frequency of Use	How often has this uses cases been triggered, we use Rare , Normal and Often to represent various levels of the frequency since we did not have proper statics for each use case.
Assumptions	Any assumptions that were made during the analysis of the use case.
PassFail	The result of test, it should be Pass or Fail .

Table 6: Black-box Test Cases Columns

So for example, one of the test cases is to trigger the Apple map and navigate the user to the item's location. The field look's like table 7:

Column	Description
ID	30
Blacklog ID	3-4-1
Name	Item details view: trigger Apple Map.
Goal	The application should be able to use the item's location data, pass it to the Apple Map application, and trigger the navigation function and lead the user to the item's location.
Precondition	User is in the item details view.
Success End Condition	The Apple Map application is triggered after the user taps the navigate button. It gets into the navigation mode and leads the user to the location based on the distance. If the user is close to the item, then it will navigate the user to the item in "walking" mode. If the item is far from the user it will be in the "transit" mode.
Failed End Condition	The Apple Map application is not triggered after the navigate button is tapped. Or the Apple Map application is triggered but it does not receive the correct location data of the item and does not activate the navigate mode either.
Trigger	User taps the navigate button.
Normal Flow	1. The user taps the navigate button. 2. The application triggers the Apple Map application and passes the geolocation of the item to it. 3. Apple Map automatically activates navigation mode and leads the user to the item.
Alternative Flows	N/A
Frequency of Use	Often
Assumptions	The item's location data is stored in the application already.
PassFail	Pass

Table 7: Black-box Test Case: Trigger Apple Map

Please find more details of the black-box test cases in the appendix D.1.

5.3 White-Box Tests

5.3.1 Method and Environment

The iOS application was developed in a component-based way, and we tested most of the elements whenever a new one was built. In React Native, there are two types of components, pure components and containers. Pure components do not store any data but render whatever data are passed to them only. The passed data in React are called "props". On the other hand, containers are the components that store the data, which is named "state", and render the pure components. In an MVC model, the pure components are more the **View** and the containers are the **Controller**. We were only able to test the pure components at the time limitation. Testing the pure components was relatively easy and quick, and it helped us to debug and boost the developing efficiency.

The testing framework we chose was Jest[9] and Enzyme[10], a JavaScript testing framework developed by Facebook. The main reason we chose Jest rather than other testing frameworks was that it was maintained by the same company, React Native. There were not only resourceful

community supports but also the official documents were really well documents. These reasons made it one of the trendy testing framework in the React ecosystem. Mocha, another popular JavaScript unit testing framework, was considered too. However, since the iOS team did not have any previous experience with it, so to meet the deadline we chose to use the most familiar tool.

The iOS application was developed in a component-based way, and we tested most of the elements whenever a new one was built. In React Native, there are two types of components, pure components and containers. Pure components do not store any data but render whatever data are passed to them only, The passed data in React are called "props". On the other hand, containers are the components store the data, which is named "state", and render the pure components. In an MVC model, the pure components are more the **View** and the containers are the **Controller**.

We were only able to test the pure components at the time limitation. Testing the pure components was relatively easy and quick, and it helped us to debug and boost the developing efficiency.

5.3.2 Test Cases

The testing framework we chose were Jest[9] and [10], Jest was a JavaScript testing framework developed by Facebook and Enzyme was the React component testing utility. There were several reasons made us choose these tools rather than other testing frameworks. Firstly, Jest and Enzyme were developed and maintained by Facebook and the Airbnb developers team. The resourceful community supports and the well-written official documents were all really helpful. These reasons made these tools became the trendy testing frameworks in the React ecosystem. Mocha, another popular JavaScript unit testing framework, was considered too. However, the iOS team did not have any previous experience with it, so we chose to use the most familiar tools in order to meet the deadline.

Jest and Enzyme were combined together into one testing file and this tested if a component was rendered properly. Button was one of the pure components which has been unit-tested, button contained two props: **text** and **handler**. The **text** was the text rendered on the button, and the **handler** was the function which would be called whenever the button was pressed. The button code shows in Listing 1:

```
import React from 'react';
import { mainFontBold, style } from '../styles/variables';
import { Text, TouchableHighlight } from 'react-native';

export default ({ text, handler }) => (
  <TouchableHighlight
    style={style.button}
    onPress={() => handler()}
  >
    <Text style={{ color: '#fff', fontFamily: mainFontBold }}>{text}</Text>
  </TouchableHighlight>
);
```

Listing 1: Button.jsx

In terms of the button component, we tested it in four aspects:

- **Rendered without crashing:** We used the snapshot API from Jest. A snapshot was JSON string of the component, during the test the newly built component would be converted to JSON and compared with the previously built snapshot. The test would be passed if both snapshots were the same.
- **Rendered with correct elements:** Text and TouchableHighlight should be rendered in the button component, we tested if they existed.
- **Rendered with correct props:** The text field was rendered with a text prop, which should be rendered in the Text component inside. So we tested if the Text component was rendered with the correct text.
- **The button handler should work if the button is pressed:** Tested if the handler was functional if the button was pressed simulated.

The actual code shows in Listing 2:

```

import React from 'react';
import { Text, TouchableHighlight } from 'react-native';
import { shallow, configure } from 'enzyme';
import renderer from 'react-test-renderer';
import Adapter from 'enzyme-adapter-react-16';

import Button from '../../src/components/Button';

configure({ adapter: new Adapter() });

describe('<Button/>', ()=>{
  it('renders without crashing', () => {
    const rendered = renderer.create(<Button text="title" handler={() => {}}/>).
      toJSON();
    expect(rendered).toMatchSnapshot();
  });

  it('renders with correct elements', () => {
    const wrapper = shallow(<Button text="title" handler={() => {}}/>);
    expect(wrapper.find(Text).exists()).toBeTrue();
    expect(wrapper.find(TouchableHighlight).exists()).toBeTrue();
  });

  it('renders with correct props', () => {
    const wrapper = shallow(<Button text="title" handler={() => { testData =
      "button_is_pressed"; }}/>);
    const text = wrapper.find(Text);
    expect(text.props().children).toEqual("title");
  });

  it('the button handler should work if the button is pressed', () => {
    let testData = "before_button_is_pressed";
    const wrapper = shallow(<Button text="title" handler={() => { testData =
      "button_is_pressed"; }}/>);
    wrapper.find(TouchableHighlight).simulate('press');
    expect(testData).toEqual("button_is_pressed");
  });
});

```

Listing 2: Button.test.jsx

This is how we unit tested the pure components. Our application used most of the native components provided by Expo SDK and React Native, so only 4 components were needed to be tested: **Button**, **FullPageView**, **ItemListCell** and **PasswordInput**. The other 9 container components were tested if they were rendered properly without crashing only. The source code for unit testing could be found in our public Github repository: <https://github.com/GSoft-Goldsmiths/iLost-main/tree/master/src/react-native-app/tests>

5.4 Evaluation

Both types of tests helped us to maintain the quality of our software before actually conducting the usability tests with the participants, and here are some points that could be improved in terms of our quality control:

- **Add extreme cases:** The test cases aimed to test the normal use cases since we did not have much time to test the extreme cases. But in real life, the boundary value analysis is relatively important in terms of the software testing as well apart from the normal use cases.
- **Automated the tests:** During the black-box tests we conducted the tests manually, but in fact, we could automate the process and this would save us a lot of time. In fact, Jest, the tool we used for automated unit testing, supports the automated application testing, it is worthy to do more research on it.
- **More tests related to the tracker:** Due to limited time, we did not have much time left to do the tests for the physical tracker. It is important to test any part of our product, especially the integration between the tracker and the mobile application should be well tested.
- **Follow Test Driven Development(TDD):** TDD was one of the development methodologies we intended to implement since it controlled the code quality. But the trade off was slowing down the developing speed. For building an MVP, TDD might not be a good option, but in future development, it would be essential.

6 Summative Evaluation

6.1 Development Methodologies

Outcomes During the development we used three development methodologies: **Scrum**, **Kanban** and **Backlogs**, also we used **Progress Tracking form** to record the team's time contributions.

- **Scrum** This methodology is suitable for a team where they tend to work together in a fixed time, day-to-day. Even though we had scheduled the daily sprint time, not everyone would attend since it was not compulsory and registered. Only few sprint sessions at the beginning went well. In terms of time management, not every one of us were able to contribute fixed time every one or two days, so we could not have a daily sprint or even weekly sprints. Unless we could work full time on this project and everyone is willing to meet at a place and work together, or it would be better not to implement Scrum in our team.

- **Kanban** Regards of project management, Kanban went well, but it would be better if we could have a workspace where we could combine our backlogs and Kanban. We found that there are two web applications that could provide this kind of services: Clubhouse and Jira. They are the web applications where we can add user stories which link to Kanbans at the same time. It would be nice to use these kind of tools to save time for the team .However, this was not possible as we are students that are currently studying and therefore do not have the budget to use these services.
- **Backlogs** The backlogs were the most important things we could improve on. We only listed the basic cases while the extreme use stories were not in our backlogs. For example, "**As a user I want my tracker to keep working in the snowy seasons**" could be in our user story as our product should still be functional in the winter season.
- **Progress Tracking** In terms of the progress tracking form, it should add one column which links to the contributions, either their GitHub commit or documentation. It was not convenient to record the hours each individual worked without any evidence but this did happen. Also, it was not really feasible to use working hours to represent the contributions. For example, with the same task, people spend less time to finish the same job should be rewarded higher compared to who spend longer period to do so. But we were not able to do this since everyone worked on different tasks. It would be fairer if each task could be evaluated by the importance, whoever completes the higher importance means contributing more. We should have a contribution list which records everyone's participation. Then with the list, we could write the peer access form in a more objective point of view.

Overall Not every development methodology fits our team, in terms of our team size and working time period, Scrum is definitely did not suit us, the progress tracking was not really helpful, while Kanban and the backlogs went well and helped us to monitor the developing process. After having a taste of each methodology, then we are able to select the suitable developing methodology for our next project.

6.2 Formative Evaluation

Outcomes Our formative evaluation was focused on the functionalities of our iOS application, we conducted the tests after a certain mount of functionalities were developed. We found it really helpful to observe how the participants interacted with our application, and learned the fact that some of the functionalities might not be that useful, or the user interface might not be very intuitive but rather confusing. The success rate diagrams show that most of the users could complete the task we assigned them to do, and we needed to work on the ones with lower success rates to improve the usability.

Overall The tests were conducted with totally 15 participants with three versions of our iOS application within one month, what we could do better is to conduct the tests more regularly, such as every half month. We found that some of the functionalities might not be that useful at all, which might already take us half month to develop. We would have saved time if we did the tests before developing it. In fact, the tests should also be scheduled in the Kanban as well and considered as important as the actual development.

6.3 Design and Implementation

Outcomes During the stage, both hardware and software technologies were chosen with a lot of research that was carried out in the first term. But still, not every development step went smoothly.

- **Bluetooth** It turned out that the Bluetooth technologies were not that easy to use and we were not able to integrate it with the application.
- **iOS Development** Initially we thought it was possible to learn how to build the iOS application from zero experience and develop it within the time frame, but in fact it was a very steep learning curve.
- **Tracker Server** The tracker was designed to host a server and the application could talk to it directly, but apparently it did not work like this.
- **Tracker Case** The tracker case was designed with a dedicated 3D model, but we did not have time to actually print it out. Only a cut-board box was made to show the size of the case.

Overall By encountering various problems and issues during the development, we learned to tackle the issues by finding an alternative way towards the same goal, rather than being stuck and trying to work with the same method. For example, the iOS development process was dramatically slowed down since the iOS team was inexperienced in Swift, so we managed to find an alternative development way, to use React Native, to build it within a short period of time. Be flexible with the confront problems was the most valuable lesson we learnt during the development.

6.4 Quality Assurance

Outcomes The quality assurance was divided into two parts: **white-box tests** and **black-box tests**, the first one was conducted before the second one.

- **White-box tests** We conducted the unit-tests for every user interface components to make sure that each of them renders correctly. Before each commit of the source code, the components need to pass the unit-tests to ensure that all components would not crash. In fact, the tests could be improved by adding more tests in terms of the functionalities of an component. For now the tests were only aimed for testing rendering but not the functionalities, how a button looks like would be tested but not what would happen if it was clicked.
- **Black-box tests** The black-box tests were cross reference to the backlogs and tested by us instead of the actual users. We wrote all the test cases for each backlog, we defined what was the precondition before a test, what triggered the tests, how to determine whether a test passed or failed, and so on.

Overall In terms of the white-box tests, we were able to test the rendering, and it helped us to prevent someone accidentally putting a bug in the program during the development. It saved us time to not worry about the safety of the codes. It was really helpful and we learnt the importance of TDD and why developers need it. In terms of the black-box tests, unfortunately we were not able to pass all the tests because not all the functionalities were fully developed. During writing the black-box test cases, we needed to be really clear about how the application should work and how they interacted with the tracker or Hologram Cloud server. It was really important for the person who wrote the test cases to know the whole architect of the service. It might be better to write the tests even before any implementation, because we might end up finding some issues within the program or architect.

References

- [1] M. Nygard, "Blog — Documenting Architecture Decisions — Relevance", Thinkrelevance.com, 2011. [Online]. Available: <http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions>. [Accessed: 15- Mar- 2018].
- [2] "How Many Test Users in a Usability Study?", Nielsen Norman Group, 2012. [Online]. Available: <https://www.nngroup.com/articles/how-many-test-users/>. [Accessed: 01- Mar- 2018].
- [3] "Writing Tasks for Quantitative and Qualitative Usability Studies", Nielsen Norman Group, 2018. [Online]. Available: <https://www.nngroup.com/articles/test-tasks-quant-qualitative/>. [Accessed: 14- Mar- 2018].
- [4] "Success Rate: The Simplest Usability Metric", Nielsen Norman Group, 2001. [Online]. Available: <https://www.nngroup.com/articles/success-rate-the-simplest-usability-metric/>. [Accessed: 14- Mar- 2018].
- [5] M. 2018, "Mobile OS: market share in the United Kingdom 2011-2018 — Statistic", Statista, 2018. [Online]. Available: <https://www.statista.com/statistics/262179/market-share-held-by-mobile-operating-systems-in-the-united-kingdom/>. [Accessed: 20- Apr- 2018].
- [6] "Expo", Expo, 2018. [Online]. Available: <https://expo.io/>. [Accessed: 18- Mar- 2018].
- [7] "TestFlight - Apple Developer", Developer.apple.com, 2018. [Online]. Available: <https://developer.apple.com/testflight/>. [Accessed: 18- Mar- 2018].
- [8] "use case testing - IntexSoft", Intexsoft.com, 2015. [Online]. Available: <http://www.intexsoft.com/blog/item/117-use-case-testing.html>. [Accessed: 18- Mar- 2018].
- [9] "Jest Delightful JavaScript Testing", Facebook.github.io, 2018. [Online]. Available: <https://facebook.github.io/jest/>. [Accessed: 18- Mar- 2018].
- [10] "Introduction Enzyme", Airbnb.io, 2018. [Online]. Available: <http://airbnb.io/enzyme/>. [Accessed: 19- Mar- 2018].

Appendices

A Development Records

A.1 Backlogs

User Story ID	User Story	Backlogs					
		Backlog ID	Acceptance Criteria	Priority	Dev Days	Phase	Process
1	Log Tracker: User logs the Tracker into App			Total Dev Days	11.5	Completion Rates	50.00%
1-1	As a user I want to have a nice item logging user interface, so that I can easily log items in App.	1-1-1	Log item user interface should be simple, clear and easy to navigate.	Must	2	MVP	Done
1-2	As a user I want to add a picture of the item as a reminder of what it looks like.	1-2-1	User can decide to use a picture from the camera rolls or take a new picture of the item	Must	2	MVP	Todo
		1-2-2	User can take a picture of their item within App.	Must	1	MVP	Todo
		1-2-3	User can decide to retake the picture if the photo is not good enough.	Must	0.5	MVP	Todo
		1-2-4	App can store the picture in the mobile.	Must	0.5	MVP	Todo
		1-2-5	User can choose a photo from the album.	Must	0.5	MVP	Done
		1-2-6	User can choose to skip adding a picture.	Must	0.5	Final	Done
1-3	As a user I want to name my item, so that I'll know what the item is if I don't want to take a picture of it.	1-3-1	User can add the item's name.	Should	0.5	MVP	Done
		1-3-2	App can store the item's name.	Must	0.5	MVP	Done
1-4	As a user I want the app to tell me if my inputs are correct automatically, so that I don't need to revise them.	1-4-1	App should validate the item inputs.	Should	0.5	Final	Done
1-5	As a user I want to know app can actually detect the tracker, so that I will feel safer that it actually works.	1-5-1	App should send a request to register Tracker.	Must	1	MVP	Todo
		1-5-2	Tracker should receive request and register App.	Must	1	MVP	Todo
		1-5-3	App should be able to store Tracker data.	Must	0.5	MVP	Done
		1-5-4	App should activate Tracker and make it send the location to App.	Must	0.5	MVP	Todo
2	Monitor Tracker: App listens to the signal of Tracker			Total Dev Days	22	Completion Rates	0.00%
2-1	As a user I want the app to track the item's location all the time, so that I have a record of its location once it's lost.	2-1-1	App listens for signal at regular intervals.	Must	3	MVP	Todo
		2-1-2	Tracker send position data to App at regular intervals.	Must	3	MVP	Todo
		2-1-3	App saves the history of the position of Tracker at regular intervals.	Must	1	MVP	Todo
		2-1-4	App should be able to upload the history of the Tracker's position to server at regular intervals.	Must	1	MVP	Todo
2-2	As a user I want the tracker to send the accurate location data to app, so that I can find it easily if it's correct..	2-2-1	Tracker uses bluetooth mode while close to User.	Must	7	Final	Todo
		2-2-2	Tracker uses cellular mode while is far from User.	Must	7	Final	Todo
3	Track Item: User uses App to see the tracking list and track Item.			Total Dev Days	29.5	Completion Rates	50.00%
3-1	As a user I want to view the tracking item list, so that I can view my item and look up more detail if I want.	3-1-1	Item list user interface should be simple, clear and easy to navigate.	Must	2	MVP	Done
		3-1-2	App should get item data from mobile database	Must	1	MVP	Done
		3-1-3	App should fetch lastest locations of all the tracking items when User visit item list page.	Should	1	MVP	Todo
		3-1-4	App should display if any of Items is lost clearly.	Must	2	MVP	In Progress
3-2	As a user I want to activate or deactivate Tracker, so that I can save my mobile and the tracker's batteries.	3-2-1	User should be able to activate/deactivate Tracker to stop tracking.	Should	2	Final	Todo
3-3	As a user I want to see the location history in a map format of the item, so that I can track/look for it if it's lost.	3-3-1	Item history user interface should be simple, clear and easy to navigate.	Must	2	MVP	Done
		3-3-2	User should be able to navigate from item list to the item history.	Must	0.5	MVP	Deprecated
		3-3-3	App should display the Item and User's location in a map format.	Should	4	Final	Done
		3-3-4	User should be able to see Item's location on the map in at different time.	Should	2	Final	Done
3-4	As a user I want to navigate me to my item, so that find it quicker.	3-4-1	App should link to Apple Map to navigate User to Item's last location.	Should	3	Final	Done
		3-4-2	App should link to Google Map App to navigate User to Item's last location.	Should	1	Final	Todo
		3-4-3	App could point in direction of item using a compass and distance in meters for close range situations in which the item isn't clearly visible.	Could	2	Final	Todo
3-5	As a user I want to edit my item detail, so that I can change the tracking item.	3-5-1	User can rename item.	Should	1	Final	Done
		3-5-2	User can retake the photo of the item.	Should	1	Final	Done
		3-6-1	User can delete the item.	Should	1	Final	Todo

An user I want to delete the item on

User Story ID	User Story	Backlogs					
		Backlog ID	Acceptance Criteria	Priority	Dev Days	Phase	Process
3-6	As a user I want to delete the item, so that I can stop tracking the item for good.	3-6-2	App should delete all the data related to the item.	Should	2	Final	Todo
		3-6-3	App should deactivate the Tracker.	Should	2	Final	Todo
4	Notification: User should receive notification if the item is lost			Total Dev Days	4	Completion Rates	0.00%
4-1	As a user I want to get notification if the item is far from me, so that I can retrieve before it's taken by someone else.	4-1-1	App should activate the OS notification function while it detects Item is too far. The default "too far" definition is 10 meters away.	Must	3	MVP	Todo
		4-1-2	Notification should be able to link to App.	Should	1	Final	Todo
5	Onboarding: User registers/logins a password for logging the app			Total Dev Days	14.5	Completion Rates	88.89%
5-1	As a user I want to set password for the app, so that others can't see where's my item.	5-1-1	Password register user interface should be simple, clear and easy to navigate.	Must	2	MVP	Done
		5-1-2	User can register 4-digit password in App	Should	3	Final	Done
		5-1-3	User can use Touch-ID as login method.	Should	1	Final	Done
5-2	As a user I want to reset my password, so that I can prevent if someone has seen my password.	5-2-1	Setting page user interface should be simple, clear and easy to navigate.	Must	1	MVP	Done
		5-2-2	User should be able to reset the password in Setting page.	Should	3	Final	Done
5-3	As a user I want to turn off password, so that I can login quicker.	5-3-1	User can turn off password protecting feature in Setting page.	Should	1	Final	Todo
5-4	As a user I want to access the app, so that I can use it.	5-5-1	User can use 4-digit or Touch-ID as input to login.	Should	2	Final	Done
		5-5-2	App prevents User from accessing App if the login authentication fail.	Should	0.5	Final	Done
		5-5-3	App redirect to item list if the login authentication succeed.	Should	1	Final	Done
6	Physical Tracker			Total Dev Days	13	Completion Rates	0.00%
6.1	As a user I want to make sure that the actual hardware is working as expected	6-1-1	User should be able to switch on and off the tracker	Must	1	Final	Todo
		6-1-2	The tracker must always work when is switch on	Must	1	Final	Todo
		6-1-3	The tracker should locate itself and send the location in the Amazon Cloud Server	Must	1	Final	Todo
6.2	As a user I want that the battery to last enough long	6-2-1	The battery should to last for at least one day	Should	1	Final	Todo
		6-2-2	User should be able to charge the battery	Should	2	Final	Todo
		6-2-3	Users should be able to change the battery if it breaks	Should	2	Final	Todo
6.3	As a user I want that my tracker does not brake if I drop it.	6-3-1	The case should be break resistant and shoud not break if it drops from a meter high	Should	2	Final	Todo
		6-3-2	The case should be made with environmental harmless materials	Should	3	Final	Todo
7	Data Security			Total Dev Days	3	Completion Rates	0.00%
7.1	As a user I want that sensitive data is kept restricted to authorized to access it.	7-1-1	Specify which data produced by the system is to be considered sensitive	Should	3	Final	Todo

A.2 Architecture Decision Records

A.2.1 Mobile Development

Title	Which mobile OS to work on the mobile application.
Context	The main stream of mobile operating systems are Android and iOS, which platform should we develop for?
Decision	In the UK, iOS and Android almost share each half of the market. It is better to develop on both platforms to serve more users as possible. Also, if either of the Android or iOS team couldn't deliver the app on time, we could still have the back-up app on the other platform.
Status	Accepted.
Consequences	We formed two teams - Android and iOS team to develop on both platforms.

Table 8: ADR Mobile Application Development

A.2.2 iOS Development

Title	Which language was used for developing the iOS mobile application.
Context	There are two main languages for iOS developing, Swift and Objective-C. While Swift is the latest programming language to write iOS applications, more libraries are written in Objective-C. Which language should we choose to use?
Decision	Since we might only need the basic functionalities provided by the generic libraries, the external libraries are not necessarily needed. Besides, the iOS team has no experience in Swift or Objective-C, so we will go for Swift which has a lower learning curve.
Status	Accepted
Consequences	The iOS team therefore used Swift to develop the iOS app.

Table 9: ADR iOS Application Development

A.2.3 Android Application Development

Title	Which language was used for developing the Android mobile application.
Context	There are two main languages for Android developing, Java and Kotlin. Kotlin was released relatively new, released by Google while Java has a long history. Which language should we choose to use?
Decision	Apparently our team is more familiar with Java than Kotlin, in order to finish the application within a limited time, we will go for Java.
Status	Accepted
Consequences	The Android team therefore used Java to develop the Android app.

Table 10: ADR Android Application Development

A.2.4 Transfer iOS Development from Swift to React Native

Title	Transfer iOS app from Swift to React Native
Context	The developing process was really slow and the iOS team could not find enough support due to the Swift version issue. React Native is an alternative way to develop the iOS application but it is written with React framework in JavaScript, which is more friendly to the team who is experienced in React. Should we migrate the codes to React Native from Swift?
Decision	Migrating would not take too much time and it would be way more easier for us to develop in JavaScript, so it would be a great chance for us to catch up on the developing schedule.
Status	Accepted.
Consequences	The app was migrated from Swift to React Native within a half week, and the same functionality could be built in one day in React Native while 4 days in Swift. The developing efficiency increased dramatically.

Table 11: ADR Transfer iOS Development from Swift to React Native

A.3 Tasks Divided

A.4 Progress Tracking Form

WBS Code	Date	Status	Resource Name (Hours)						
			Chin	Dylan	Marian	Mifuad	Hussein	Thairan	Jheng-Hao
<u>1-1</u>	19/10/2017	Brainstormed and decided the subject topic.	1	1	1	1	1		1
<u>1-2</u>	22/10/2017	Planned and scheduled tasks and redesigned progress tracking form							3.5
<u>1-3</u>	23/10/2017	Built online survey - refined and ready to deploy	3	0.5			4		
<u>1-3</u>	23/10/2017	Deployed online survey, collecting data	3	3	1	1			
<u>1-2</u>	23/10/2017	Read references and drafted first persona							1.5
<u>1-2</u>	24/10/2017	Drafted the second and third of personas							0.5
<u>4-2</u>	24/10/2017	Lab session	1			1			1
<u>2-1</u>	24/10/2017	Drafted the user scenarios							0.75
<u>2-2</u>	24/10/2017	Competing Products (Market Research)				3			
<u>2-1</u>	25/10/2017	Improved the user scenarios and drew the use case diagram.							1.5
<u>2-1</u>	25/10/2017	Drew the diagrams.							2.75
<u>2-4</u>	26/10/2017	Regulation and Standards							2
<u>2-3</u>	26/10/2017	Academic Research							1.5
<u>2-10</u>	26/10/2017	Integrate all the reports together							1.5
<u>1-2</u>	26/10/2017	Stakeholders Report						2.5	
<u>4-3</u>	27/10/2017	Meeting with supervisor	0.5	0.5	0.5	0.5	0.5	0.5	
<u>4-4</u>	29/10/2017	Recorded meeting notes							0.25
<u>2-7</u>	29/10/2017	Emailed Pete to ask for a meetup							0.5
<u>2-7</u>	29/10/2017	Research on Hardware to use for the sensors	2	1					
<u>2-7</u>	29/10/2017	RFID and NFC research							1
<u>2-7</u>	30/10/2017	Schedule a meeting with Pete							0.25
<u>4-2</u>	31/10/2017	Lab session	1	1	1	1	1		1
<u>2-7</u>	31/10/2017	Meeting with Pete		1	1				1
<u>2-7</u>	31/10/2017	Estimote research				3			
<u>2-10</u>	31/10/2017	Product name survey					1		
<u>4-1</u>	31/10/2017	Divided and assigned the tasks	0.5	0.5					0.5
<u>4-1</u>	01/11/2017	Tracking tasks progress.							0.5
<u>4-1</u>	02/11/2017	Tracking tasks progress.							0.5
<u>2-10</u>	02/11/2017	Integrate all the reports together							0.5
<u>2-5</u>	02/11/2017	Search and select project management tool							1
<u>4-3</u>	03/11/2017	Meeting with supervisor	0.5	0.5	0.5	0.5	0.5	0.5	
<u>2-7</u>	03/11/2017	Physical lab induction							2
<u>2-6</u>	09/11/2017	Conceptual Prototype Interviews							4
<u>2-6</u>	09/11/2017	Conceptual Prototype Report							1
<u>4-1</u>	09/11/2017	Set up Github Repository							0.5
<u>4-1</u>	10/11/2017	Move reports to Github Repository							2
<u>4-2</u>	14/11/2017	Lab session	1	1		1	1	1	
<u>2-7</u>	14/11/2017	Prepare Arduino tutorial kits				2.5			
<u>4-1</u>	14/11/2017	Updated repository and readme							1
<u>2-5</u>	15/11/2017	Planning the project (Sub tasks, Major tasks and Milestones)						3.75	
<u>4-3</u>	17/11/2017	Meeting with supervisor	0.5	0.5	0.5	0.5	0.5	0.5	
<u>2-5</u>	19/11/2017	Drew gantt chart and improved the project plan							3
<u>2-7</u>	20/11/2017	Required Training Project 0 and 1.	2	2.5	1				
<u>2-9</u>	20/11/2017	Project Prototype	2			12			
<u>2-5</u>	20/11/2017	Project Sequence Diagram							2
<u>4-2</u>	21/11/2017	Lab session	1	1	0.5		1	1	1
<u>2-9</u>	23/11/2017	Project Prototype Report		3		7			
<u>2-1</u>	23/11/2017	Technical Architecture Research							1
<u>4-3</u>	24/11/2017	Meeting with supervisor	0.5			0.3			0.5
<u>4-2</u>	28/11/2017	Lab session	1	1		1		1	1
<u>2-10</u>	28/11/2017	User Need Overview & Concept Introduction Draft		9.5					
<u>2-7</u>	29/11/2017	Arduino GPS						2.25	
<u>2-1</u>	29/11/2017	Architecture Diagrams							3
<u>4-3</u>	24/11/2017	Meeting with supervisor	0.5	0.5	0.5	0.5	0.5	0.5	
<u>4-1</u>	02/12/2017	Updated github work flow & convention							1.25
<u>2-7</u>	03/12/2017	Arduino - RFID			3				
<u>2-10</u>	03/12/2017	Prepare presentation				0.4			1
<u>4-2</u>	05/12/2017	Lab session	1	1		1	1	1	1
-	05/12/2017	Resource request meeting		1.5		1.5	1.5	1.5	
<u>2-10</u>	05/12/2017	Final Proposal			1				2.5
<u>4-3</u>	08/12/2017	Meeting with supervisor	0.5	0.5	0.5	0.5			0.5
<u>2-1</u>	08/12/2017	Update use cases				3			3

WBS Code	Date	Status	Resource Name (Hours)						
			Chin	Dylan	Marian	Mifuad	Hussein	Thairan	Jheng-Hao
<u>2-10</u>	09/12/2017	Evaluation Plan - Proposal	3.5						
<u>2-10</u>	09/12/2017	Proposal structural research		2					
<u>2-10</u>	09/12/2017	Final Proposal							4.5
<u>2-7</u>	09/12/2017	Estimote research			3				
<u>2-10</u>	10/12/2017	Final Proposal	3						6.5
<u>2-10</u>	10/12/2017	Final Proposal			6				
<u>2-10</u>	11/12/2017	Proposal reviews	1	1					
<u>2-10</u>	12/12/2017	Final Proposal		2	2			2	
<u>4-2</u>	12/12/2017	Lab session	1	1	1	1	1	1	
<u>2-10</u>	13/12/2017	Final Proposal	5	5	4	5		1.5	5
<u>2-10</u>	14/12/2017	Final Proposal	4	4		4	3		6
<u>4-1</u>	26/12/2017	Project cleanup						0.4	2
<u>3-7</u>	28/12/2017	iLost logo						2	2
-	28/12/2017	Computer fair prep - posters						9	5
<u>3-7</u>	31/12/2017	(ongoing) iLost app user interfaces redesign							7
<u>3-9</u>	01/01/2018	(ongoing) iLost app front-end development							9
-	08/01/2018	Computer fair prep - posters retouching							3
-	11/01/2018	Computer fair (severe travel issues)						2	2
<u>3-9</u>	12/01/2018	(ongoing) iLost back-end development						9	
<u>4-1</u>	15/01/2018	Set up back-end repo and team			0.2		0.3		
<u>4-2</u>	16/01/2018	Lab session	1	1	1	0.5	0.5	0.5	1
<u>4-1</u>	16/01/2018	Refactor project tasks and retouch use cases.							1.5
<u>3-7</u>	16/01/2018	(ongoing) iLost back-end development			2.75		5		
<u>3-7</u>	17/01/2018	iLost iOS app user interfaces design							5
<u>3-8</u>	18/01/2018	iLost back-end Set up with Rapsberry and Nova			3		2		
<u>4-3</u>	19/01/2018	Meeting with supervisor			0.3	0.3	0.3	0.3	
<u>4-1</u>	22/01/2018	Write the user stories, backlogs and acceptance criteria.	0.5						2.5
-	23/01/2018	3D printing induction							3
<u>4-2</u>	23/01/2018	Lab session	1		1	1	1		
-	23/01/2018	Meeting with tech support			1		1		
<u>4-1</u>	23/01/2018	Setup Sprint structure							1.5
<u>3-8</u>	24/01/2018	Short-range devlopment (puck.js)					9		
<u>4-3</u>	26/01/2018	Meeting with supervisor		1	1		1		1
<u>3-8</u>	26/01/2018	iLost long range						1.2	
<u>3-8</u>	26/01/2018	iOS app develop - add a new Item view							4.5
<u>3-8</u>	29/01/2018	iOS app develop - item list view							5.5
<u>3-8</u>	29/01/2018	iLost long range					8		
<u>4-2</u>	30/01/2018	Lab session	1	1	1	1	1	1	
<u>3-8</u>	30/01/2018	Android app development - Welcome screens	4						
30/01/2018				3					
30/01/2018					2				
<u>3-8</u>	30/01/2018	app endpoint fixed					3		
<u>3-8</u>	30/01/2018	iOS app develop - item list view						3	
<u>3-8</u>	31/01/2018	iOS app develop - item list view							3.5
<u>3-8</u>	31/01/2018	fixed ilost-long range					4		
<u>3-8</u>	01/02/2018	iOS app develop - item detail view							0.75
<u>4-3</u>	02/02/2018	Meeting with supervisor	0.5	0.5	0.5	0.5	0.5	0.5	
<u>3-8</u>	02/02/2018	iOS app develop - item detail view & discuss extending user stories							2
<u>3-8</u>	02/02/2018	Android app development - Main activity navigation drawer	3		2				
<u>3-8</u>	04/02/2018	Android app development - Login screen lockpattern		4.5					
<u>3-8</u>	05/02/2018	Android app development - Learning android studio				1.5			
<u>3-8</u>	06/02/2018	Android app development - item list	2		1				
<u>3-8</u>	06/02/2018	update API & back-end script					2.5		
<u>4-2</u>	06/02/2018	Lab session	1	1	1	1	1	1	
<u>3-8</u>	06/02/2018	iOS app develop - Amazon S3 API intergrate							4.5
<u>3-8</u>	07/02/2018	iOS app develop - Amazon S3 API intergrate							1
<u>4-3</u>	09/02/2018	Meeting with supervisor				0.5	0.5	0.5	
<u>3-8</u>	07/02/2018	iOS app develop - Amazon S3 API intergrate							4.5
<u>5-5</u>	13/02/2018	Final Report ch4 plan					2		
<u>3-8</u>	15/02/2018	Android app - item list	2						
<u>3-8</u>	17/02/2018	Android app - itemlist screen development & testing	3		2				
<u>3-8</u>	20/02/2018	iOS app develop - item detail view, load location							4.5

WBS Code	Date	Status	Resource Name (Hours)						
			Chin	Dylan	Marian	Mifuad	Hussein	Thairan	Jheng-Hao
<u>3-8</u>	20/02/2018	Android app - tracking activity, google maps api implementation	2		1				
<u>3-8</u>	20/02/2018	iOS app develop - map api, data parsing							3
<u>4-2</u>	20/02/2018	Lab session	1	1		1	1	1	
<u>3-8</u>	21/02/2018	Android app	4		2				
<u>3-8</u>	21/02/2018	Tracker - Intergrate raspberry pi zero			4				
<u>3-8</u>	21/02/2018	Tracker - Intergrate raspberry pi zero					4		
<u>3-8</u>	21/02/2018	iOS app develop - UI storyboard							4
<u>5-5</u>	27/02/2018	FInal Report ch3 plan							2
<u>5-3</u>	28/02/2018	Make presentation slides							3
<u>3-10</u>	01/03/2018	Usability Tests: research							4
<u>5-5</u>	03/03/2018	FInal Report section 1 plan	1						
<u>3-8</u>	03/03/2018	iOS app develop - Migrated to React Native							7
<u>3-8</u>	04/03/2018	iOS app develop - Authenticate Views							9
<u>3-8</u>	05/03/2018	Make the tracker demo case							2
<u>3-8</u>	05/03/2018	iOS app develop - add item view, edit item view, data storage management							5
<u>5-3</u>	06/03/2018	presentation script pt. 5					1.5		
<u>5-5</u>	06/03/2018	Final report conclusions structure/plan					2		
<u>3-10</u>	07/03/2018	Usability Tests: make cosent form, structure test record table, conduct tests							5
<u>5-5</u>	07/03/2018	Final presentation: update slides - test & evaluation and production pages.							1
<u>3-8</u>	08/03/2018	iOS app develop - bugs fixed, details in commit message							2
<u>3-10</u>	08/03/2018	Usability Tests: conduct tests							1
<u>5-5</u>	08/03/2018	Final presentation: update slides - test & evaluation and production pages.							2.15
<u>5-5</u>	08/03/2018	report structure/writing			3				
<u>4-3</u>	09/03/2018	Meeting with supervisor	0.5	0.5	0.5	0.5	0.5	0.5	
<u>5-5</u>	14/03/2018	Final report: Formative Evaluation - mobile application test							4
<u>5-5</u>	14/03/2018	Final report: Development Record - Technology selection, agile development							4
<u>5-5</u>	13/03/2018	report writing/reading other points			3				

B Formative Evaluation

B.1 Consent Form

Usability Test Consent Form

Please read and sign this form.

In this usability test:

- You will be asked to perform certain tasks on a provided mobile phone.
- We will also conduct an interview with you regarding the tasks you performed.

Participation in this usability study is voluntary. All information will remain strictly confidential. The descriptions and findings may be used to help improve the iLost Project.

However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

If you have any questions after today, please contact Jheng-Hao Lin at jlin015@gold.ac.uk

I have read and understood the information on this form and had all of my questions answered

Subject's Signature

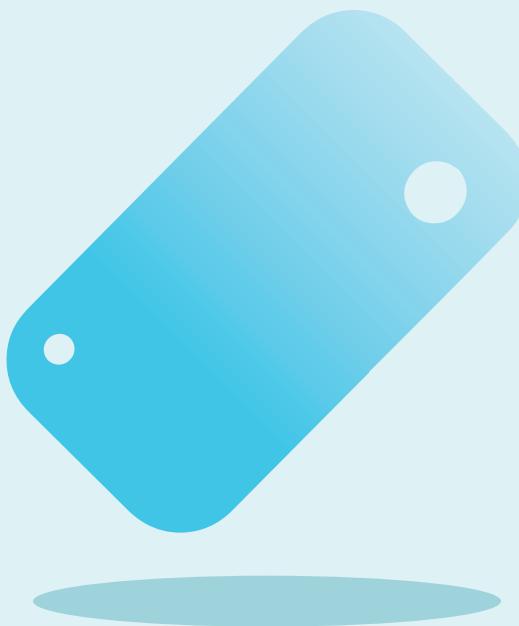
Date

Thank you!

We appreciate your participation.

C Design and Implementation

C.1 Computer Fair Posters



Track Anything Anywhere...



Attach tracker

iLost can be attached to anything, your bag, keys even your car. iLost adapts to any attached item.



Get reminder

Never lose anything again, iLost will remind you when you leave something behind.



Track item

The free iLost app will give you the precise location of the item indoors or outdoors.



Unlimited Range

iLost can locate any item accurately anywhere in the world. You can even track your airline baggage.



Long Battery Life

Approximately 3 months battery life. Battery is fully replaceable.

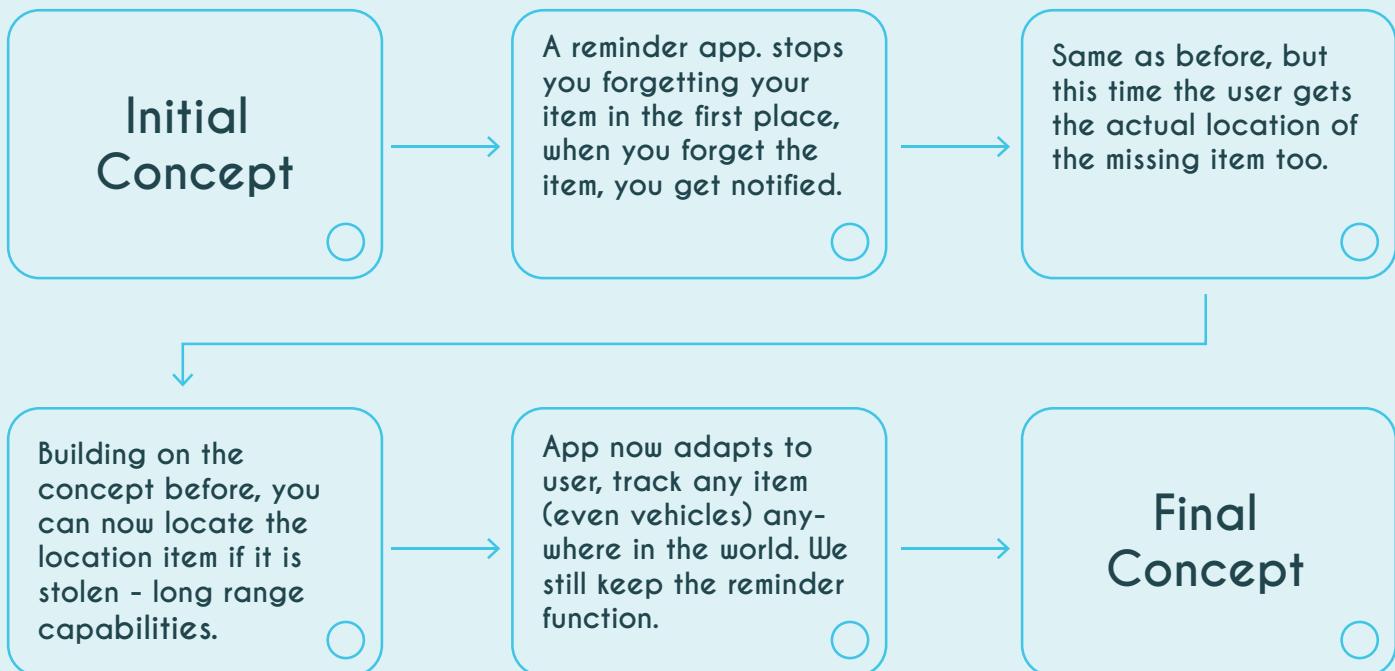


Durable

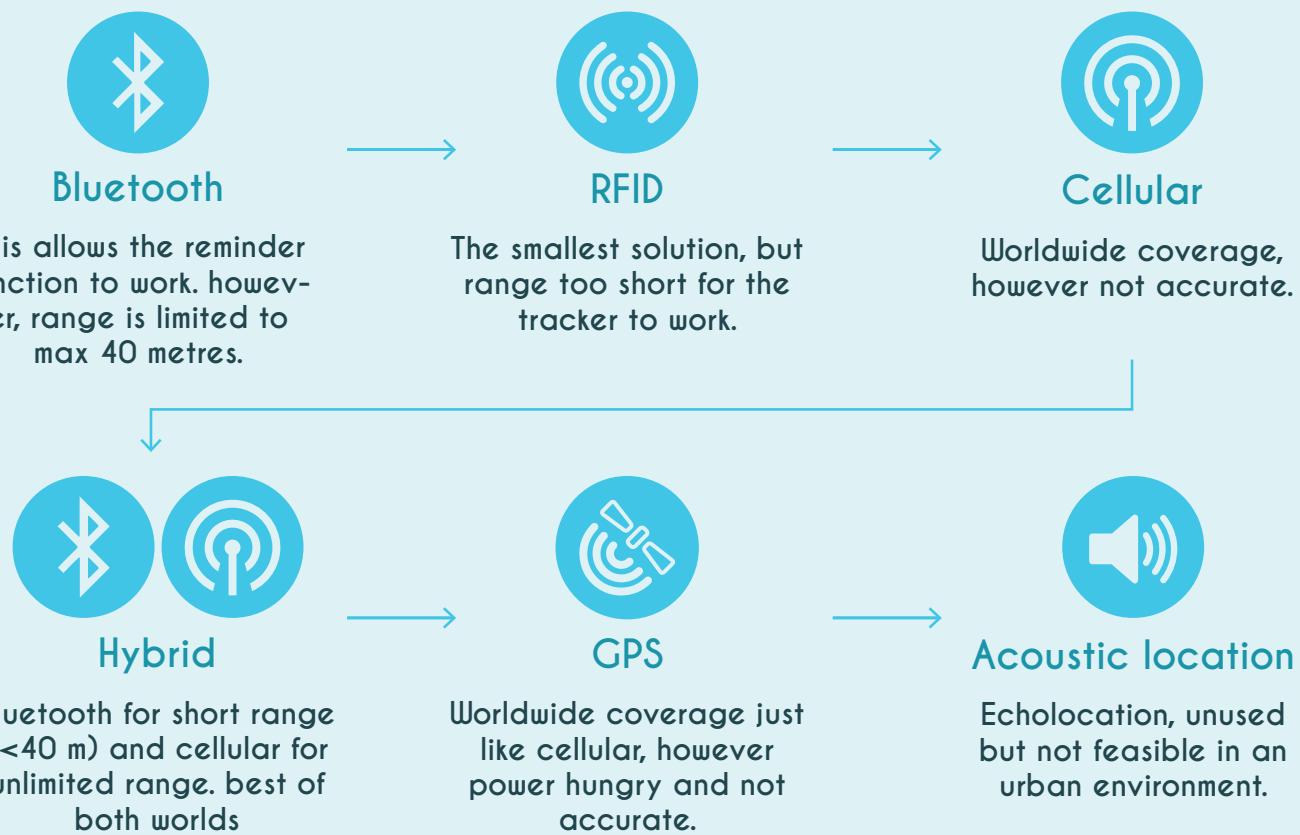
Waterproof, dust-proof, drop-proof the tracker has been designed to survive extreme conditions



Conceptual Prototyping



Functional Prototyping





Technology



Raspberry pi Zero

- Allows the iLost tracker to communicate with the Android app
- Calculates the location of the tracker
- its module module allows for short range location tracking



iLost android app

- Track your items
- Links with iLost tracker
- Adapts to the user



Battery

- Replaceable and rechargeable Li-ion battery
- Average battery life of 3 months on one charge



Hologram Nova

- Cellular modem
- Communicates with nearby cell towers, provides cellular triangulation ability
- Allows unlimited range tracking

D Quality Assurance

D.1 Black-box Test Cases

ID	Backlog ID	Name	Goal	Preconditions	Success End Condition	Failed End Condition	Trigger-	Normal Flow	Alternative Flows	Frequency of Use	Assumptions	Pass/Fail
1	1-1-1	Add item view: user interface	Display the a view title, camera icon, text field and two buttons in the view.	The user access to the application and is logged in. The user taps the plus button in the item list view.	The correct elements(a view title, camera icon, text field and two buttons) are displayed same as the UI design.	Components are not displayed properly.	Login in the application	1. User is in the item list view 2. User taps the plus button in the navigation bar 3. Application displays the add new item view.	N/A	Rare	N/A	Pass
2	1-2-1	Add item view: camera or choose pictures	Display the a view title, camera icon, text field and two buttons in the view.	The user access to the application and is logged in. The user is in the add item view.	After the camera icon is tapped, the application displays two options to select: "Select from photo library" and "take a new photo".	Nothing happens after the camera icon is tapped or nothing triggered after any of the two options is tapped.	The camer icon is tapped.	1. User is in the add item view. 2. User taps camera icon. 3. application triggers a action menu where user can have two option to select.	N/A	Rare	N/A	Pass
3	1-2-2	Add item view: trigger camera application	Trigger the native camera application once the camera icon is tapped.	The user access to the application and is logged in. The user is in the add item view.	The camera application is triggered successfully and ready to take a picture of the item.	The camera application is not triggered when the camera icon is tapped.	The camer icon is tapped then the "take a new photo" option is tapped.	1. User is in the add item view. 2. User taps camera icon. 3. application triggers a action menu where user can have two option to select. 4. User taps the "take a new photo" option.	N/A	Rare	N/A	Pass
4	1-2-3	Add item view: retake photo		The user access to the application and is logged in. The user is in the add item view and native camera application is triggered	After a photo is taken, the native camera application displays a button "retake" in the left bottom corner and another button "OK" in the right bottom corner. If the "retake" button is selected, current photo will be erased and enable user to take another photo. If the "OK" button is selected, application will pass the photo to the next view.	The native camera application does not display correct two buttons after a photo is taken.	A photo is taken in the triggered camera application.	1. If the "Retake" button is tapped, the application goes back to previous stage and let the user to take another photo. 2. If the "OK" button is tapped, the application goes to next stage which should be display the photo on the add item view.	N/A	Rare	N/A	Pass
5	1-2-4	Add item view: store photo		After taking a photo, there should be an option for using it and another option for retaking it.	The native camera application and is triggered	After a photo is taken and selected to use, the taken photo will be stored in the photo library.	The native camera application does not store the photo in the photo library.	1. User takes a photo and selects it in the native camera application triggered in the add item view. 2. The application stores the taken photo in the photo library.	N/A	Rare	N/A	Pass
6	1-2-5	Add item view: choose photo from camera roll		User decides to use the taken photo, it should store a copy in the photo library.	The user access to the application and is logged in. The user is in the add item view and native camera application is triggered	The native camera roll API is triggered and user the select any photo from the existing albums. The selected photo will be passed to the next view.	The camer icon is tapped then the "select from photo library" option is tapped.	1. application goes back the add item view if user cancel to select a photo from the photo library. 2. User taps the "select from photo library" option.	N/A	Rare	There are already taken photos in the photo libaray.	Fail
7	1-2-6	Add item view: not add photo		User can browse the native photo application to select an desired photo, rather than taking a new photo.	The user access to the application and is logged in. The user is in the add item view.	application will not prevent user to the next stage if the user save a new item without choosing a photo.	User cannot go to the next stage if the user save a new item without choosing a photo.	1. User is in the add item view. 2. User taps the save button with the validated item name input but without using any photo.	N/A	Rare	N/A	Pass

ID	Backlog ID	Name	Goal	Preconditions	Success End Condition	Failed End Condition	Trigger-	Normal Flow	Alternative Flows	Frequency of Use	Assumptions	Pass/Fail
8	1-3-1	Add item view: add item's name	application should be display display an editable text field in the add item view, which enable user to input the text item's name.	The user access to the application and is logged in. The user is in the add item view.	The text input field in the item view is editable, user can edit the text value of it.	The text input field in the add item view is not editable, user cannot edit the text value of it.	User taps the text fields in the add item view.	1. User is in the add item view. 2. The text input field displays correct placeholder text: "Enter item's name" 3. User taps the text input field. 4. The application triggers the keyboard. 5. User can input the item's name.	1. User is in the add item view. 2. The text input field displays correct placeholder text: "Enter item's name" 3. User taps the text input field. 4. The application triggers the keyboard. 5. User can input the item's name.	Rare	N/A	Pass
9	1-3-2	Add item view: store item's name	After the save button is tapped, the application will store the value of the text input field as the item's name in the application.	The user access to the application and is logged in. The user is in the add item view and has typed the item's name in the text field.	After the save button is tapped, the value of the text input field will be stored in the application successfully.	After the save button is tapped, the value of the text input field is not stored in the application/	User taps the text fields in the add item view.	1. The text input field in the add item view is edited by the user with a validated name. 2. User taps the save button. 3. The application saves the value of the text field in the application.	1. The text input field in the add item view is edited by the user with a validated name. 2. User taps the save button. 3. The application saves the value of the text field in the application.	Rare	The text input value is validated.	Pass
10	1-4-1	Add item view: validate inputs	After the save button is tapped, the application will validate the input's value. The values will only be saved if the inputs are validated.	The user access to the application and is logged in. The user is in the add item view.	After the save button is tapped, the application will validate all the inputs, including the photo and the text field. If they are all validated, then the data will be saved. Otherwise the application will display the error message to prompt the user to input the validated data.	The application is not able to validate the input, such as the input length of the item's name.	User taps the save button in the add item view.	1. User is in the add item view. 2. User taps the save button. 3. The application validates the input fields.	1. If all the data are validated, the application will save the data in the application, then return to the item list view. 2. If any of the data is invalidated, the application will display an error message to ask user to input correctly.	Rare	N/A	Fail
11	1-5-1	App registers tracker	System can store a unique id of the tracker and bound it with the item in the application	The user access to the application and is logged in. The user is in the add item view and validated inputs of the item are saved.	The system has saved the validated inputs of the item and the application sends a request to pair the tracker with this item via bluetooth.	The system is unable to send any request to the tracker.	User taps the save button in the add item view with validated inputs, and the system has saved all the input data in the application.	1. The application has saved the item's data. 2. The application send a request to register the tracker.	1. If the bluetooth sensor is turned off, prompt user to turn it on.	Rare	N/A	Fail
12	1-5-2	Tracker receives register request from App	The tracker should able to receive and read the request from the application via bluetooth.	The server hosted on the tracker is able to read and interpret the register request sent from the application	The server hosted on the tracker is not able to read and interpret the request sent from the application	The server hosted on the tracker is not able to read and interpret the register request sent from the application	The application has sent a request to register the tracker with bluetooth.	1. The application send a request to register the tracker. 2. The tracker received the request and read the data in it. 3. The tracker pairs itself with the application via the data in the request.	1. The application has sent a register request to the nearby tracker.	Rare	N/A	Fail
13	1-5-3	App stores Tracker data	The application stores the tracker's special ID and pair it with the item's data as a reference.	The tracker has received a register request from the tracker and already registered itself with the application.	The application stores an unique ID of the tracker and pair it with the item.	The application cannot store any data of the tracker.	The tracker stored the application's registered data and sends back a registering succeed response.	1. The tracker pairs itself with the application via the data in the request. 2. The application stores the tracker's unique ID in the storage.	N/A	Rare	N/A	Fail

ID	Backlog ID	Name	Goal	Preconditions	Success End Condition	Failed End Condition	Trigger	Normal Flow	Alternative Flows	Frequency of Use	Assumptions	Pass/Fail
14	1-5-4	App activates Tracker	After pairing the tracker, the application can activate the locating function via bluetooth.	The application sends a bluetooth request to the paired tracker, which enables the tracker starts the locating service and send data to the S3 database.	The tracker has been paired with the application.	The application cannot activate the tracker's locating service at all	For the first time that the tracker and the application have been paired with each other.	1. The tracker has finished the pairing routine with the application. 2. The application has finished the pairing routine with the tracker. 3. The application send a request to the tracker to activate its' locating service. 4. The tracker receives the request and starts the locating services.		Rare	N/A	Fail
15	2-1-1	App monitors the geolocation of Tracker.	The application sends out the location request to the tracker at regular intervals, the default interval time is 10 minutes.	The application successfully sends out an HTTP request to the tracker via Hologram cloud every five minutes.	The application fails to sends out an HTTP request to the tracker via Hologram cloud every five minutes.	The application has activated the tracker and paired with it.	The application has activated the tracker and paired with it.	1. The application sends an HTTP request to the Hologram. 2. After 10 minutes, the application repeat step 1 and step 2 again.		N/A	N/A	Fail
16	2-1-2	Tracker sends out geolocation data.	The tracker should be able to response the request from the application.	The tracker has been paired with the application.	The tracker is not able to response the geolocation data to the HologramCloud.	The tracker has been activated by the application and paired with it, then the application sends out a geolocation data request to the tracker.	The tracker has been activated by the application and paired with it, then the application sends out a geolocation data request to the tracker.	1. The tracker receives the request from the application. 2. The tracker responses the request with the geolocation data. 3. The Hologram Cloud receives the data from the tracker and stores it in the Amazon S3 server.		N/A	Often	The tracker has been activated.
17	2-1-3	App store the history of Tracker's location data at regular intervals.	The application should be able to store the location data behind the background.	The application makes a HTTP request to the Amazon S3 Server and retrieves the location data.	The application fails to make the HTTP request or retrieve the location data from the Amazon S3 server.	The application is set up to monitor the tracker.	The application is set up to monitor the tracker.	1. The application sends an HTTP request to the Amazon S3 Server. 2. After 10 minutes, the application repeat step 1 and step 2 again.		N/A	Often	N/A
18	2-1-4	[Deprecated]	-	The tracker has been paired with the application, and there are already location data stored in the Amazon S3 server.	The application fails to make the HTTP request or retrieve the location data from the Amazon S3 server.	-	-	1. The application sends an HTTP request to the Amazon S3 Server. 2. After 10 minutes, the application repeat step 1 and step 2 again.		-	-	-
19	2-2-1	Tracker activates the bluetooth mode when the paired mobile device is more than 30 meters away.	The tracker should be able to switch modes based on the distance between it and the mobile device.	The tracker has been paired with the application.	The tracker successfully actives the bluetooth mode.	The tracker fail to active the bluetooth mode.	The distance between the tracker and the mobile device is shorter than 30 meters.	1. The tracker determine if the mobile device is nearby within 30 meters, if it is true, activate the bluetooth mode.		Normal	N/A	Fail
20	2-2-2	Tracker activates the cellular mode when the paired mobile device more than 30 meters away.	The tracker should be able to switch modes based on the distance between it and the mobile device.	The tracker has been paired with the application.	The tracker successfully actives the cellular mode.	The tracker fail to active the cellular mode.	The distance between the tracker and the mobile device is longer than 30 meters.	1. If the mobile device is not 30 meters away, activate bluetooth mode.		Normal	N/A	Fail
21	3-1-1	Item list view: user interface	Display the a view title, and a item photo, a item name and a button for each item in the view within 5 seconds.	The user has added at least one item with name and photo in the item list.	The correct elements(a view title, item photos, item names and buttons) are displayed same as the UI design within 5 seconds.	Components are not displayed properly or timeout.	The user logs to the application.	1. The user logs to the application. 2.. The application renders the correct elements on the screen within 5 seconds.		Normal	N/A	Pass

ID	Backlog ID	Name	Goal	Preconditions	Success End Condition	Failed End Condition	Trigger	Normal Flow	Alternative Flows	Frequency of Use	Assumptions	Pass/Fail
22	3-1-2	Item list view: display data from storage.	The application should retrieve the data form the mobile storage and render with the correct UI elements.	The user has added at least one item with name and photo in the item list.	The correct elements(a view title, item photos, item names and buttons) are displayed same as the UI design within 5 seconds.	Components are not rendered properly with the data.	The user logs to the application.	1. The user logs to the application. 2. The application retrieves the data from the storage. 3. The application renders the data on the UI elements	N/A	Normal	N/A	Pass
23	3-1-3	[Deprecated]	-	-	-	-	-	1. The user is in the item list view. 2. The application retrieves the data from the storage. 3. The application determines if any of the item is 5 meters away from the device's current location. 4. Any of the items is 5 meters away from the device will be rendered with a red border.	-	-	-	-
24	3-1-4	Item list view: indicates the item is lost or not.	The user should be able to tell if the items are lost or not once seeing the item list..	The user has added at least one item with name andphoto in the item list.	The item is 5 meters away from the mobile device should be considered as a lost item, which will render with a red border to indicate the user it is lost.	The item is 5 meters away from the mobile device is not considered as lost or not with a red border rendered.	Any of the item is 5 meters away from the mobile device.	1. The application sends an HTTP request to the tracker via Hologram Cloud. 2. The tracker receives the request and turns on/off of itself based on the request.	N/A	Normal	N/A	Fail
25	3-2-1	Item list view: turn on and off the tracker.	The user should be able to turn on and off of the tracker in the item list view.	At least one tracker has been paired with the application.	The application successfully turns on/off of the tracker.	The application fails to turn on/off of the tracker.	The user taps the button of a tracker.	1. The application sends an HTTP request to the tracker via Hologram Cloud. 2. The tracker receives the request and turns on/off of itself based on the request.	N/A	Rare	N/A	Fail
26	3-3-1	Item details view: user interface	Display the a item name, a map and the item history location list in the view within 5 seconds.	The item has stored at least one location data.	The correct elements(a item name, a map and the item history location list) are displayed same as the UI design within 5 seconds.	Components are not displayed properly or timeout.	The user taps the item photo in the item list view.	1. The application retrieves the location data of the selected item from the storage. 2. The application renders the UI elements with the retrieved data.	N/A	Normal	N/A	Pass
27	3-3-2	Item details view: navigate from the item list view to item details view	The application is able to navigate between views.	At least one tracker has been paired with the application and that item has stored at least one location data.	The application successfully navigates from item list view to the item details view.	The application fails to switch views.	The user taps the item photo in the item list view.	1. The user is in item list view and taps the one of the item. 2. The application switches to item details view and renders the item details view UI elements.	N/A	Normal	N/A	Pass
28	3-3-3	Item details view: map	The application should display latest 10 history locations on the map.	The item has stored at least one location data.	The location data are displayed on the map in the correct location.	The location data fails to display on the map or they are in the wrong location.	The user taps the item photo in the item list view.	1. The application retrieves the location data of the selected item from the storage. 2. The application renders the geolocation of the data on the map.	N/A	Normal	N/A	Pass
29	3-3-4	[Deprecated]	-	-	-	-	-	1. User taps the navigate button. 2. The application triggers the Apple Map application and pass the geolocation of the item to it. 3. Apple Map automatically activates navigation mode and lead user to the item.	N/A	-	-	-
30	3-4-1	Item details view: trigger Apple Map	The application should be able to use the item's location data, pass it to the Apple Map application, and trigger the navigation function and lead the user to the item's location	User is in the item details view.	The Apple Map application is triggered after the user taps the navigate button. It gets into the navigation mode and lead user to the location based on the distance. If the user is closed to the item, then it will navigate the user to the item in "walking" mode. If the item is far from the user it will be in the "transit" mode.	The Apple Map application is triggered after the user taps the navigate button. Or the Apple Map application is triggered but it does not receive the correct location data of the item and activate the navigate mode either.	There are item's location data stored in the application already	Often	Often	Often	Pass	

ID	Backlog ID	Name	Goal	Preconditions	Success End Condition	Failed End Condition	Trigger	Normal Flow	Alternative Flows	Frequency of Use	Assumptions	Pass/Fail
31	3-4-2	Item details view: trigger Google Map	The application should be able to use the item's location data, pass it to the Google Map application, and trigger the navigation function and lead the user to the item's location	User is in the item details view.	The Google Map application is triggered after the user taps the navigate button. It gets into the navigation mode and lead user to the location based on the distance. If the user is closed to the item, then it will navigate the user to the item in "walking" mode. If the item is far from the user, it will be in the "transit" mode	The Google Map application is not triggered after the navigate button is tapped. Or the Google Map application is triggered but it does not receive the correct location data of the item and activate the navigate mode either.	User taps the navigate button	1. User taps the navigate button. 2. The application triggers the Google Map application and passes the geolocation of the item to it. 3. Google Map automatically activates navigation mode and lead user to the item.	N/A	Often	There are item's location data stored in the application already	Fail
32	3-4-3	[Deprecated]	-	-	-	-	-	1. The application renders the text field with the value of item's name. 2. The user edits the text field with their new name. 3. The user taps the save button. 4. The application reads the new item's name and stores it in the storage.	N/A	Rare	N/A	Pass
33	3-5-1	Item details view: rename item	The user should be able to change the item's name in the application	User is in the item details view.	The application provides an interface for user to edit the item's name, and stores the new name in the storage.	The application fails to provide an interface for user to edit the item's name or store the new name in the storage.	User taps the edit button in the item details view.	1. The application renders the photo. 2. The user taps the photo and the application triggers the camera app 3. The user takes a new photo of an item. 4. The application reads the new item's photo and stores it in the storage.	N/A	Rare	N/A	Pass
34	3-5-2	Item details view: retake photo	The user should be able to change the item's photo in the application	User is in the item details view.	The application provides an interface for user to edit the item's photo, and stores the new photo in the storage.	The application fails to provide an interface for user to edit the item's photo or store the new photo in the storage.	User taps the edit button in the item details view.	1. The application renders a delete button. 2. The user taps the delete button. 3. The application deletes all the data related to the item from the storage. 4. The application sends a turn off request to the tracker.	N/A	Rare	N/A	Pass
35	3-6-1	User deletes item	The user should be able to delete the item's all data in the application	User is in the item details view.	The application provides an interface for user to delete all the data in the storage.	The application fails to provide an interface for user to delete all the data in the storage.	User taps the edit button in the item details view.	1. The application sends a turn off request to the tracker and the tracker turns itself off after receiving the request.	N/A	Rare	N/A	Fail
36	3-6-2	App deletes item data	The application should be able to delete the item's all data in the application	User is in the item details view.	The application deletes all the data related to the deleted item.	The application fails to send a turn off request to the tracker or the tracker could not turn itself off after receiving the request.	User taps the delete button in the edit item view.	1. The application deletes all the data related to the item from the storage.	N/A	Rare	N/A	Fail
37	3-6-3	App turns off Tracker after delete the item.	The application should automatically turn off the tracker after the associated item is deleted	User is in the item details view.	The application sends a turn off request to the tracker and the tracker turns itself off after receiving the request.	The application fails to trigger the OS notification even if the mobile device is 10 meters away from the item.	User taps the delete button in the edit item view.	1. The application sends a turn off request to the tracker.	N/A	Rare	N/A	Fail
38	4-1-1	App trigger OS notification when any of the items is too far.	The application should trigger the OS notification based on the distance between the mobile device and the location data stored.	At least one tracker has been paired with the application.	If the mobile device is 10 meters away from the item, the application triggers the OS notification and warns the user that the item might be lost.	The application fails to trigger the OS notification even if the mobile device is 10 meters away from the item.	If the mobile device is 10 meters away from the item, the application triggers the OS notification and warns the user that the item might be lost.	1. The application triggers the OS notification. 2. The notification contains the warning message along with the item's name, item's last stored location.	N/A	Normal	N/A	Fail

ID	Backlog ID	Name	Goal	Preconditions	Success End Condition	Failed End Condition	Trigger-	Normal Flow	Alternative Flows	Frequency of Use	Assumptions	Pass/Fail
39	4-1-2	OS notification links to App	The OS notification should redirect the user back to the application.	At least one tracker has been paired with the application.	If the application triggers the OS notification, one of the buttons displayed with the notification can redirect the user back to the application.	If the application triggers the OS notification, none of the buttons displayed with the notification can redirect the user back to the application.	If the application triggers the OS notification triggered by the application.	If the mobile device is 10 meters away from the item and the notification is triggered.	N/A	Normal	N/A	Fail
40	5-1-1	Passcode register view: user interface	Display the an icon of lock, a text explaining why the application would be safer with a passcode setup and a inputfield let the user register the passcode.	The user has not set up the passcode.	The correct elements(a lock icon, a text description and a input field) are displayed same as the UI design within 5 seconds.	Components are not displayed properly or timeout.	The user has not setup the passcode and it is the first time login into the application	1. The application renders the correct UI element. 2. The user input the passcode.	N/A	Rare	N/A	Pass
41	5-1-2	Passcode register view: stores the passcode.	The application should let the user double input the passcode, then store the passcode in the storage both inputs are the same.	The user has not set up the passcode.	After the user inputs the same passcode two time, the application stores the passcode into the storage.	The application could not read the input data or store the input data.	The user input the passcode in the passcode field.	1. The user inputs the passcode. 2. The application asks the user to input the passcode again. 3. The user inputs the passcode again. 4. The application compares both passcode inputs. 5. If both inputs are exactly the same, stores the passcodes into the storage.	1. If both passcode inputs are not identical, the application re-asks the user to input the passcode again.	Rare	N/A	Pass
42	5-1-3	Touch ID register view: UI	Display the an icon of fingerprint, a text explaining why the application would be safer with the Touch ID setup and two buttons let the user selects whether to enable it or not.	The user has not set up the Touch ID.	The correct elements(a fingerprint icon, a text description and two buttons) are displayed same as the UI design within 5 seconds.	Components are not displayed properly or timeout.	The user has not setup whether to enable the Touch ID.	1. The application renders the correct UI element. 2. The user select whether to enable Touch ID. 3. The application stores the user's decision.	N/A	Rare	N/A	Pass
43	5-2-1	Setting view: user interface.	Display a list of setting options in the setting view.	N/A	The correct elements are displayed same as the UI design within 5 seconds.	Components are not displayed properly or timeout.	The user enters the setting view.	1. The application renders the correct UI element..	N/A	Rare	N/A	Pass
44	5-2-2	Setting view: reset password	The application should enable user to change the password	The user has set up the passcode.	The new passcode is read and stored in the storage	The new passcode could not be read or stored.	The user is in the setting view and taps the reset passcode option.	1. The application follows the process of passcode registration in backlog ID 5-1-2. 2. The user reads and stores the new passcode.	N/A	Rare	N/A	Fail
45	5-3-1	Setting view: turn off password protection	The passcode protection should be able to be turned off.	The user has set up the passcode.	The passcode is no longer needed to be entered after turning off.	The application still asks the user to input the passcode when login to the application.	The user is in the setting view and taps the turn off passcode option.	1. The application stores the user's decision of whether to turn off passcode whenever login the application.	N/A	Rare	N/A	Fail

ID	Backlog ID	Name	Goal	Preconditions	Success End Condition	Failed End Condition	Trigger	Normal Flow	Alternative Flows	Frequency of Use	Assumptions	Pass/Fail
46	5-5-1	Login: Two ways to login App	The user could either choose to use Touch ID or passcode to login the application	The user has set up the passcode and Touch ID.	The applications logs in the user if the user uses the correct fingerprint via Touch ID or enters the correct passcode.	The applications fails to login the user even if the user uses the correct fingerprint via Touch ID or enters the correct passcode.	The user has set up the passcode and Touch ID, then launches the application.	1. The application reads the user's setting of credentials. 2. The application asks the user to input the fingerprint for Touch ID. 3. If the users passes the correct fingerprint, the application logs in. 4. If the users does not pass the correct fingerprint, the application asks the user to input the passcode.	N/A	Often	N/A	Pass
47	5-5-2	Login: Authentication	The application should be protected by the credentials such as Touch ID or passcode.	The user has set up the passcode and/or Touch ID.	The application leads user to the item list view even if the user could not provide the correct credentials.	The user launches the application.	The user has set up the passcode and/or Touch ID.	1. The user is asked to provided the passcode. 2. If the user input the wrong passcode, the application displays the error message and asks the user to input again. 3. Repeat step 1 to 2 for five times if the user could not input the correct passcode. 4. The application asks the user to wait for 10 minutes to input again.	N/A	Often	The user has set up the passcode and/or Touch ID.	Pass
48	5-5-3	Login: redirect to item list page.	The application should redirect from login view to item list view.	The user has inputed the correct credentials in the login view.	The application navigates to the item list view from the login view.	The application fails to navigate to the item list view from the login view.	The user enters the correct credentials in the login view.	1. The application unmounts the login view and displays the loading animation. 2. The application reads the item data from the data storage. 3. The application renders the items list components with the item data.	N/A	Often	The user has set up the passcode and/or Touch ID.	Pass