KEEPER

CS 123 PROJECT

Brent Anonas, Harris Lao, Jacqueline Dinglasan, Paulo de Leon

12/8/2016

The Keeper Company

-established on August 2016

Table of Contents

I. Introduction

KEEPER

Problem

Losing things is inevitable in people's lives. Given the nature that sometimes people are forgetful, they lose their things and sometimes do not have a clear picture as to where they lose them; or if they do, sometimes somebody gets them and surrenders it to a random (usually the nearest) lost item keeper such as police stations, lost and found offices, etc. which the owner of the lost item is unaware of. Sometimes the person who founds the item reports it to television stations and radio shows, to which the owner might also not be aware. The problem is that there is a lack of a convenient medium or entity to help lost things be returned to their respective owners.

Furthermore, in lost and found offices, storage can be a problem when unclaimed items pile up and accumulate space as well as dust in the office. Although there are rules and standards in some offices where unclaimed items in certain periods of time can be disposed, donated, kept or sold. However, these rules and standards vary.

Project Description

The Keeper company introduces a software application for Android mobile phones. This aims to address the problem of losing items. The *Keeper* application connects people who lost an item and finds it (Finder) and someone who found the item (Founder). It helps people notify a found item as well as report a lost item at real time.

Requirements to Use the Application

- Android Phone (Jellybean)
- Internet

II. Language, Constraints, Inspiration and Aesthetic, Non-functional Requirements, and Design

Language

The programming language used to build the software application is Java using Android Studio Framework. We chose as such because of our team's lack of training for iOS development or other frameworks such as Ionic Framework that facilitate both iOS and Android Development. Our team leader knows how to develop Android Applications because he once interned in a company and was one of the developers of their app. Aside from this, Java is a language that all of the team members are familiar with. The problem of this choice, however, is that it is unusable for iPhone users. This makes our app's portability score low.

Constraints

Programmer training

Due to the lack of training, we were forced to use Android Studio instead of a more portable framework. Aside from that, our team is not refined with backend knowledge.

Time constraints

Although the time allotted for the project is an entire semester, our team leader was busy with a lot of things, and thus delayed the implementation of the application.

Tools

The phone used for testing is slow and outdated, making testing more difficult. We could have used an emulator, but the laptop of our developers could not handle it.

Money

We wanted to be as cost-efficient as possible and opted for open-source and free webhosting sites. Problem with this is an increase in difficulty and quality of the services we aimed to use. An example would be 000webhost.com. The server cannot handle a lot of load.

Inspiration and Aesthetic

Our design focused on simplicity as our app is in itself simple. This meant that navigation-wise, the buttons and all other clickables are directed to specific activities drawn to only their single purpose; which means there are rarely overlapping navigations. We've also drawn inspiration from other application's background design from PayMaya, while we've gotten some on even the simplest messenger application right on the phone.

Aside from the uniqueness of our application, we drew inspiration from Google's material design standards, although imperfectly implemented since we had no User Experience expert. Our intention is to make the app similar in some regards to the present successful apps of today while also blending in some zest in our ideas.

The theme and color of blue and purple also made our app symbolize not only the alma mater but also the color of royalty that connects the trait of honesty. We view the combination of these colors as an accentuation of integrity.

Overall, our design and function operated in our viewed ideal aesthetic visual with the intention of synergizing the two. Even with our shortcomings, we still have a clairvoyant outlook on the potential look and feel of the app once completion.

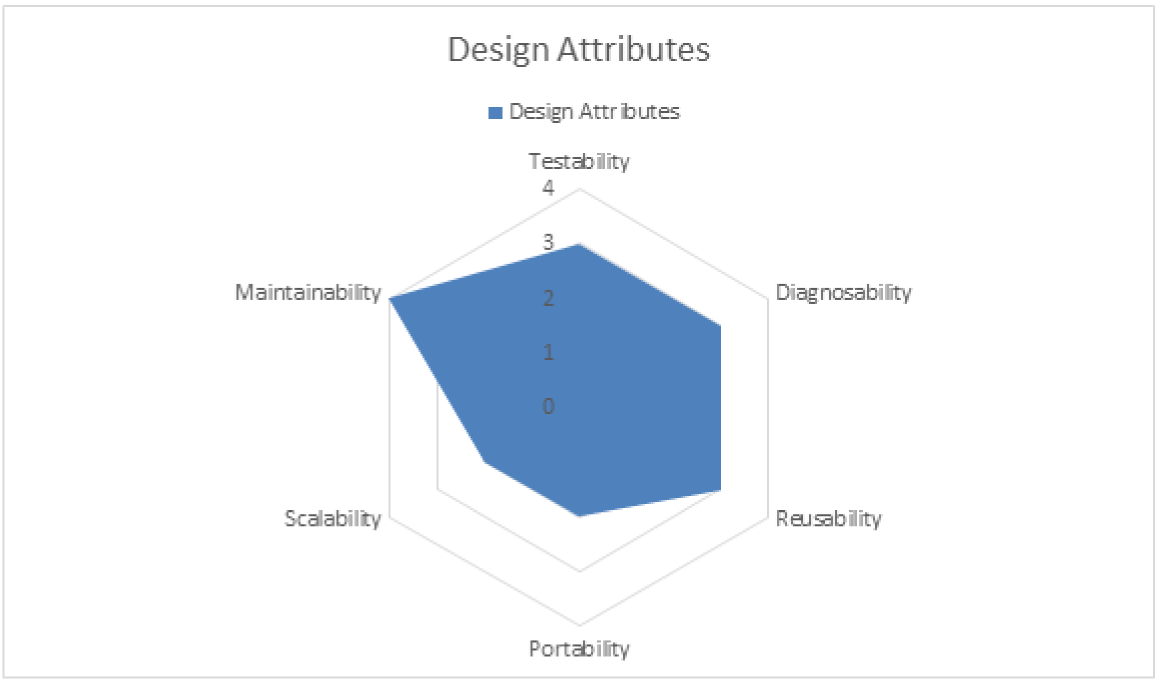
Non-Functional Requirements

Product Requirements
The application needs a stable internet so that the application is in real time when posting and seeing the items posted by other users.

Organisational Requirements
The application needs a person or maybe more, to serve as the maintenance person/people of the application itself. Not only are these maintenance to apply corrective maintenance which are fixes to certain errors, but also additional features such as data gathering. This means the application needs a statistician as well to collect and analyze the data that can be gathered from the application. Data such as what items are usually lost or what time do people usually lose their items, etc.

External Requirements
The application requires the user to have an Android Jellybean version as well as internet. Without these two, the application will not work. However, for future implementation, the scope will be more diverse so that the limitations of the application will be reduced. The application can also be partnered with other lost and found offices.

Design



The radar chart above is a rating from 0 to 5, where 0 is the lowest and 5 the highest. The radar chart only displays until 4 here because it is the highest among all of the data.

Testability: 3

It can be tested and errors can be seen. However, testing takes more time than normal programs especially in our setup because every time we test our app, it needs to reinstall the app in our phone. This rating could have been higher since newer versions of phones could test without reinstall, but we had an old and slow phone to test to.

Diagnosability: 3

There are times when unknown errors surface, like the URI not found exception or crunching error during image compression. This is internal to Android Studio and not our code.

Reusability: 3

We were able to reuse some modules of our code for other modules. For example, since Lost and Found item are similar, instead of having separate views, we coded it in such a way that we would be able to instantiate two separate objects.

Portability: 2

The application is portable to other Android Phones, even older ones since our minimum SDK is Jellybean. However, it is still undeniable that iPhones cannot use our app, thus crippling our portability score.

Scalability: 2

It worked perfectly when it was tested on only one phone. When we added more phones however, accessing the server was impossible due to our constraints.

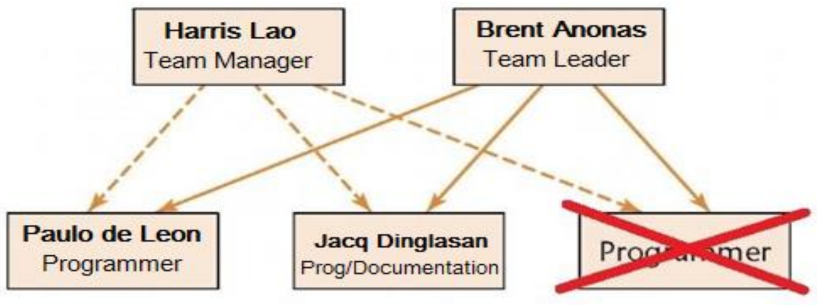
Maintainability: 4

Maintaining the system is simple since the program is well-documented not only on paper but in the code as well.

III. Terminology

a) Keeper - a mobile application that keeps lost/found information
b) Finder - someone who is looking for an item of his own
c) Founder - someone who found an item
d) User - denotes to either a Finder or a Founder or both
e) Item - refers to an object owned by someone
f) List - collection of item information

IV. Team Organization



<u>Team Manager</u>
-in charge of leading and motivating the team
-in charge of the application design
-in charge of the necessary structures and procedures

<u>Team Leader</u>
-in charge of building the software application
-in charge of leading the other programmer(s) on the developing process
-in charge of the critical thinking aspect of the development process

<u>Programmer</u>
-follows the Team Leader’s advice on the development process
-helps build the software application
-helps addressing the necessary fixes in the application

<u>Documentation (Secretary)</u>
-in charge of the necessary documentation of the software building process
-in charge of making sure all information are updated
-makes sure each process and information are detailed and understandable

V. Sprint Plans

PROJECT	RELEASE		ITERATION		TASK	EPISODES	STATUS
S1: I want to be able to navigate easily on the application from fragments	1	S1: I want to be able to navigate easily on the applicati	1	S1: I want to be able to navigate easily on the applicati	T1: Basic User Interface Design	E1: Conceptual visualization of views	Done
S2: I want to be able to log-						E2: Create wireframes	Done

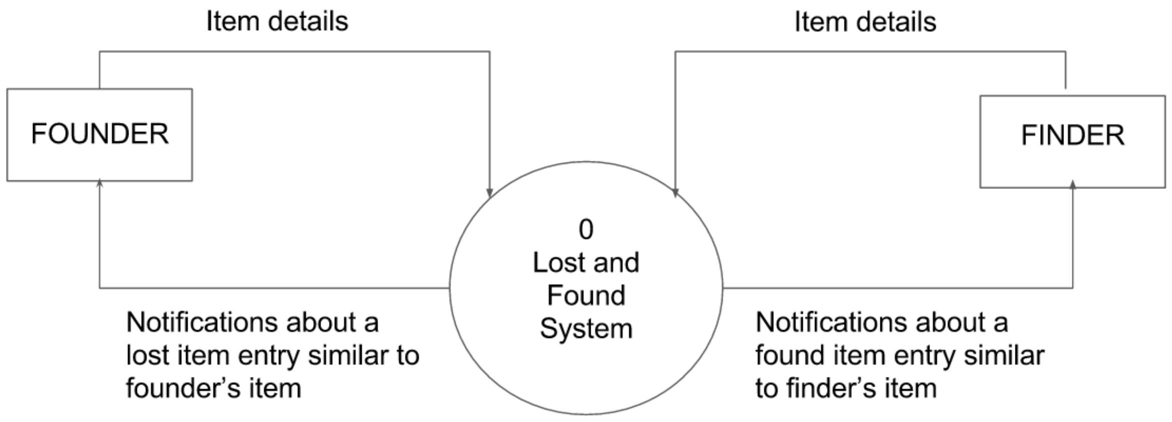
in and log-out		on from fragments		on from fragments			
S3: I want to be able to add and post a lost item to the list of all lost items					T2: Database Design	E3: Conceptual Design	Done
S4: I want to be able to add and post a found item to the list of all found items						E4: Logical Design	Done
S5: I want to be able to easily search 'Lost' and 'Found' items using keywords and filters					T3: Build Interface	E5: Build Application Fragments	Done
S6: I would like to be notified if an item similar to mine is posted		S2: I want to be able to log-in and log-out		S2: I want to be able to log-in and log-out		E6: Log-in and Log-out Page	Done
S7: I would like to contact people easily for a lost & found transaction					T4: Development of Back-end - Database	E7: Back-end coding	Done
S8: I want to keep track of my ongoing and archived transactions	2	S3: I want to be able to add and post a lost item to the list of all lost items	2	S3: I want to be able to add and post a lost item to the list of all lost items	T5: Posting functionality in 'Lost' items fragment	E8: Create "add lost item" view for posting an item I lost	Done
						E9: Create template view that properly displays relevant data regarding the item I lost	Done
						E10: Connecting database to prototype UI	Done
					T6: Whole list view of 'Lost' items fragment	E11: Make sure information flows from "add lost item" view to the template view	Done

						E12: Create list view for lost items	Done
		S4: I want to be able to add and post a found item to the list of all found items	3	S4: I want to be able to add and post a found item to the list of all found items	T7: Posting functionality in 'Found' items fragment	E13: Create "add found item" view for posting an item I found	Done
						E14: Create template view that properly displays relevant data regarding the item I found	Done
					T8: Whole list view of 'Found' items fragment	E15: Make sure information flows from the "add found item" view to the template view	Done
						E16: Crete list view for found items	Done
	3	S5: I want to be able to easily search 'Lost' and 'Found' items using keywords and filters	4	S5: I want to be able to easily search 'Lost' and 'Found' items using keywords and filters	T9: Search functionality	E17: Searching algorithm	Done
						E18: Add search bar to view and asynchronous data display to list view	Done
					T10: Add year, month and date filter aspect near search bar	E19: Add year, mo nth and date filter aspect near search bar	Done
		S6: I would like to be notified if an item similar to mine is posted	5	S6: I would like to be notified if an item similar to mine is posted	T11: Notification to founder regarding a lost item entry similar to what founder posted	E20: Account to account connection	Done
						E21: Notification through the use of matching the Keywords	Done
					T12: Notification to finder regarding a found item entry similar	E22: Account to account connection	Done
						E23:	Done

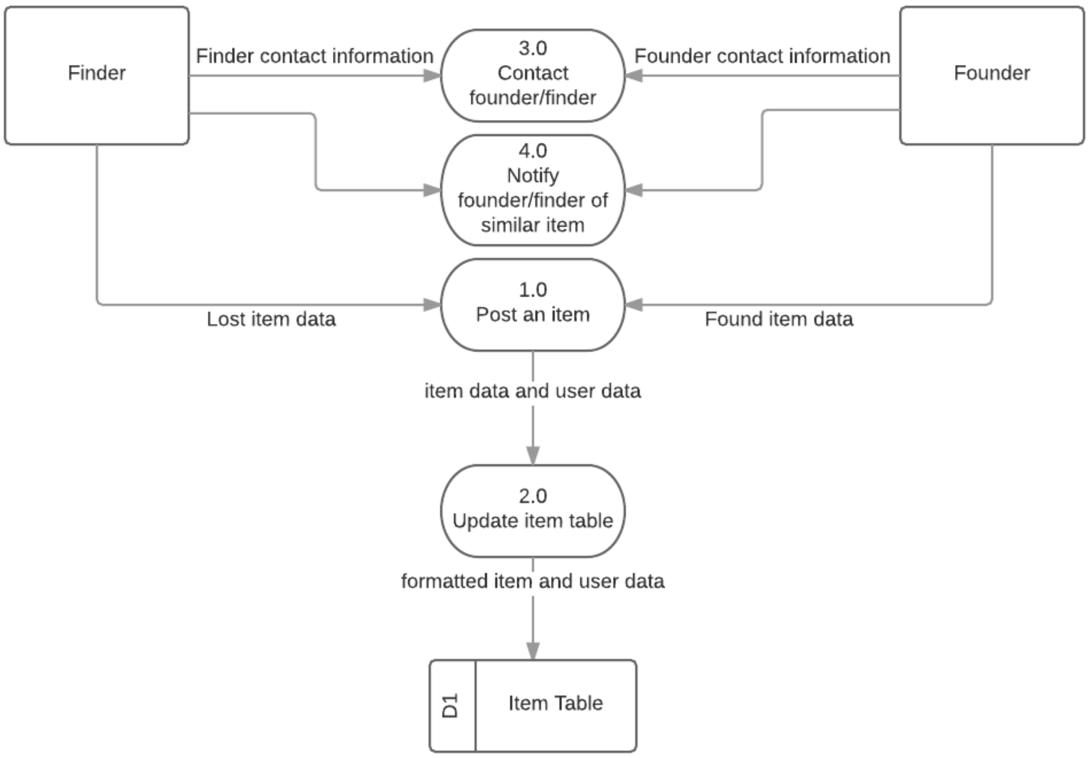
					to what finder posted	Notification through the use of matching the Keywords	
		S7: I would like to contact people easily for a lost & found transaction		S7: I would like to contact people easily for a lost & found transaction	T13: Email Notification	E24: Email to email connection	Done
						E25: Messaging interface	Done
					T14: Finder can see founder's track record	E26: Integrate founder's track records in one	Done
		S8: I want to keep track of my ongoing and archived transactions	6	S8: I want to keep track of my ongoing and archived transactions	T15: Founder/Finder can keep track activities thru email.		Done

VI. Data Flow Diagram

Context Level Diagram

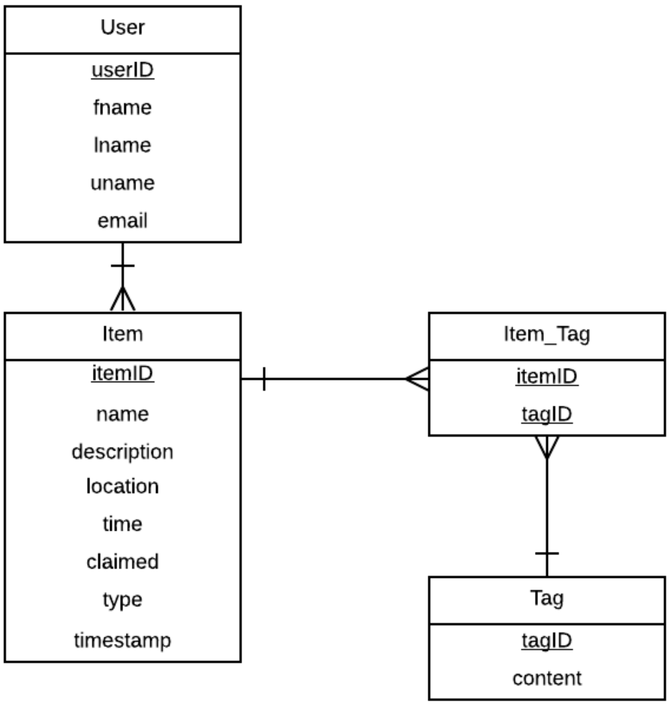

Item details
Item details
FOUNDER
FINDER
0
Lost and Found System
Notifications about a lost item entry similar to founder's item
Notifications about a found item entry similar to finder's item

LEVEL 0 DIAGRAM


Finder
Finder contact information
3.0 Contact founder/finder
Founder contact information
Founder
4.0 Notify founder/finder of similar item
1.0 Post an item
Lost item data
Found item data
item data and user data
2.0 Update item table
formatted item and user data
D1
Item Table

Just to clarify, a user can either be a founder or a finder. We just wanted to delineate the data that flows in between founder and finder.

VII. **Entity Relationship Diagram**



VIII. Data Dictionary

TABLE NAME: Item

FIELD NAME	DATA TYPE	LENGTH	PK?	FK
itemID	Int	100	Y	N
name	varchar	100	N	N
description	varchar	255	N	N
location	varchar	100	N	N
time	varchar	10	N	N
claimed	Tinyint	1	N	N
type	Smallint	6	N	N
timestamp	Timestamp		N	N
userID	Int	100	N	Y

TABLE NAME: Item_tag

FIELD NAME	DATA TYPE	LENGTH	PK?	FK
tagID	Int	25	Y	N
itemID	Int	25	Y	N

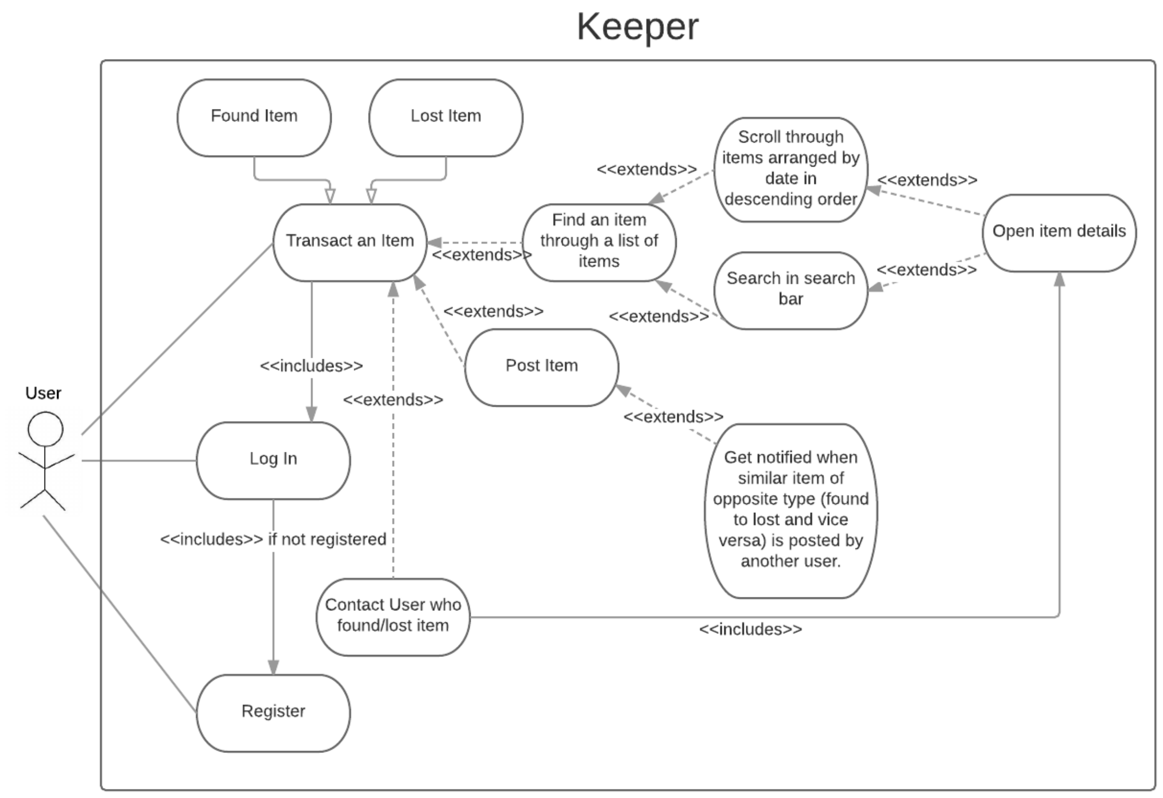
TABLE NAME: Tag

FIELD NAME	DATA TYPE	LENGTH	PK?	FK
tagID	Int	25	Y	N
Content	Varchar	100	N	N

TABLE NAME: User

FIELD NAME	DATA TYPE	LENGTH	PK?	FK
userID	Int	100	Y	N
fname	Varchar	25	N	N
lname	Varchar	25	N	N
uname	Varchar	25	N	N
pass	Varchar	25	N	N
email	Varchar	25	N	N

IX. Use Case Diagram



Keeper
Found Item
Lost Item
Transact an Item
Find an item through a list of items
Scroll through items arranged by date in descending order
Open item details
Search in search bar
Post Item
Get notified when similar item of opposite type (found to lost and vice versa) is posted by another user.
User
Log In
Register
Contact User who found/lost item
<<extends>>
<<includes>>
<<includes>> if not registered

In our diagram, the User has three ways in order to interact with the system: Register, Log In, and Transact an Item. Transact an item was used as some sort of container for the other test cases in order to reduce the complexity of our diagram. As you can see, there are two types of items: found and lost item. Without Transact an Item, each test case that has anything to do with an item has to connect to the found and lost item node. Another consideration is that every transaction with an item requires Log In. Without Transact an Item, each test case has to connect to Log In, which increases the complexity and thus higher chances of confusion.

X. Use Case Description

Use Case ID: 1
Use Case: Create profile
Author: Jacqueline Dinglasan
Purpose: To create a profile to be able to log in

Overview:
The Founder/User registers in the app by creating a profile. This enables them to perform actions such as searching and posting of item/s.
Actors: Founder/Finder
Preconditions:
Connected to the internet
Typical Course of Events:

ACTOR ACTIONS	SYSTEM ACTIONS
1. The Founder/Finder click to open the application	2. Direct to the Start Page (Not sure what is it called)
3. The Founder/Finder click the Register button	4. Opens the profile form
5. The Founder/Finder fill-up the form	6. Saves into the database
7. The Founder/Finder input log in details	8. System verifies the details

Alternative Flow of Events:
Step 8: Once the details are verified, the founder/finder will be directed to Startup tab.
Exceptional Flow of Events:
Step 7: If input of details are incorrect, the system will notify an error. Use Case: Create new Profile

User Case ID: 2
Use Case: Search an Item
Author: Jacqueline Dinglasan
Purpose: To search a lost or find item
Overview:

This is to locate/verify an item

Actor/s: Founder/Finder
Preconditions:

Connected to the internet
Founder/Finder is logged in

Typical Course of Events:

ACTOR ACTIONS	SYSTEM ACTIONS
1. Founder/Finder search the database of an item	2. App verifies similar item

Alternative Flow of Events:

Step 2: If the system found a similar item, Founder/Finder get in contact with the contact person

Exceptional Flow of Events:

Use Case: Search new item

User Case ID: 3
Use Case: Post a Found Item
Author: Jacqueline Dinglasan
Purpose: To publish an item that has been found
Overview:

The Founder posts a found item with details. Finder keeps in touch thru contact details, otherwise if the post is overdue the founder will be notified on what will be the next step.

Actor/s: Founder

Preconditions:

Connected to the internet
Founder is Logged in

Typical Course of Events:

ACTOR ACTIONS	SYSTEM ACTIONS
1. Founder post a found item	2. Item information saves into the database
	3. App notifies the Finder an overdue post

Alternative Flow of Events:

Step 3: The system suggest an option to retain, keep or donate the item.

Exceptional Flow of Events:

Use Case: Post another found item

User Case ID: 4
Use Case: Post a Lost Item
Author: Jacqueline Dinglasan
Purpose: To announce an item is lost
Overview:

This use case is performed once the Finder has logged in the application and searched for a similar item, he then can post for a lost item.

Actor/s: Finder
Preconditions:

Finder is logged in
Connected to the internet

Typical Course of Events:

ACTOR ACTIONS	SYSTEM ACTIONS
1. Founder post a lost item	2. Item information saves into the database

Use Case: Post new Lost Item

User Case ID: 5
Use Case: Validate an item
Author: Harris Lao
Purpose: To contact, ask and check if the item belongs to the finder
Overview:

This use case is performed when finder has searched for an item he thinks belongs to him or when founder has searched a finder finding an item similar to what founder found.

Actor/s: Finder
Preconditions:

Finder is logged in and clicked an item in the list of searches
Connected to the internet

Post conditions:

Typical Course of Events:

ACTOR ACTIONS	SYSTEM ACTIONS
1. Founder contacts finder 2. Finder contacts founder	2. System provides email for user to contact

Alternative Flow of Events:

Exceptional Flow of Events:

Use Case: Validate another item

User Case ID: 6
Use Case: Notifies user
Author: Harris Lai
Purpose: To notify a finder and a founder when a similar item to his post comes up
Overview:

This use case is performed when a similar post is posted in the application and notifies the finder of similar found items or the founder with similar lost items.

Actor/s: Finder
Preconditions:

Connected to the internet
Finder or Founder posted an item.

Post conditions:

Typical Course of Events:

ACTOR ACTIONS	SYSTEM ACTIONS
	1. System pops up a notification and displays the similar items. 2. System sends and email to the user

Alternative Flow of Events:

Exceptional Flow of Events:

Use Case:

XI. Lessons Learned

- Assigning and utilizing each member’s strengths as early as the initial phase does pay off in a software engineering project and can be applied in any other project.

- It is important to hear opinions and suggestions from other members even though they are not assigned to different tasks.

- Sometimes adjustments to schedules are necessary due to unforeseen happenings which might delay a project and so it is important as well to have attainable sprints and goals. Constant communication is a must.

- Google drive, Facebook, and Github are our friends in developing.
- Face-to-Face brainstorming with group mates are very productive in letting out all the ideas of the group members.
- Persistent errors like crunching cruncher and URI not registered are usual errors that randomly appear that sometimes fixes itself and leaves no explanation as to why.

XII. Future Implementation

Perfective Maintenance

Animations and the application of User Experience Design

To make the application more interactive and pleasing to the user's eye, various animations can be incorporated in the design and buttons. This is to also make the application less boring and make a more friendly and fun interface.

Security

To prevent erratic transactions in the application, standards and security measures such as security questions can be added in the critical features of the application, and it also adds to the credibility and safety of the application. This adds to the potential of the reliability of the application.

Better notification system

Currently the application has a plain notification message when it detects similar items in the application, and so to make it better, we can have an automatic redirecting of the application into the list of items which can be filtered and sorted in the user's liking. Also, for the transaction of the finder and founder, the application currently supports email notification only as to conform to security measures but more complex features such as in-app messaging can be added to make notification system more accessible in different mediums.

More refined searching

For the convenience of the user, it is important to have a more diverse filtering of search items such as most recent, oldest, A to Z, Z to A, by location, etc. This can be added in the searching algorithm of the application. Aside from that, there is also a need to refine the search algorithm that is O(n)^3. Very inefficient! It is advised that a 3rd party software, preferably open source, would be integrated into our app's search functionality, because building a search engine from the ground while using esoteric search algorithms is difficult and time constraints.

Domain Name

Since this is a no budget project, free domain was use. Crashes due to data overload is sometimes encountered. It is best to acquire in the future .

<u>Registration</u>

Although email for registration is enough for security purposes, it is still a good marketing tool to include social network linking to better promote the app.

<u>Data Collection</u>

Archives of item information is better to track history.

Adaptive Maintenance

Since the application is only running in Android, it is definitely a must to have it on iOS as well in the immediate future. The reason for not doing both is due to limited capabilities and time constraints and so the application is only available currently in Android.

The company can add another member who can program in iOS to have it available for Apple users.

Corrective Maintenance

<u>Notification System</u>

The notification system is simple and buggy.

<u>XML File Image Import</u>

Some parts of the app require a formal and standard way for image transfer to the drawable folder instead of simply copy-pasting them. This causes a hard to solve error called the Crunching Cruncher that sometimes fixes itself but sometimes persistently prevents us from running the app. This error has caused us to abandon three XML files already working (intermittently) since the image we used may have not been properly imported. More research on the 'formal' way should be necessitated.

<u>XML File Unknown Errors</u>

There are some portions of the xml file getting an unknown error called "URI not registered" that has multiple solutions to be found online but rarely confirms a fix. Sometimes it fixes itself but most of the time disrupts the overall work.

<u>Backend is Unstable</u>

Due to the lack of knowledge of our team in terms of backend programming, the system is unstable. One big effect of this is the server and the internet connection. We only used a free webhosting site, 000webhost.com. At first it was working perfectly: the JSON data was being sent to the client, and the client can send data to the server. However, when tested on more phones, the server could not handle the load and there were Socket exceptions occurring, causing our application to crash. It is advised that a better hosting site will be used.

XIII. Individual Comments and Insights

Initially we had a different plan and a much higher goal and wider scope. However along the way, due to time constraints as well as skill constraints, Professor Marlene was able to foresee what is the attainable goal of the development of the application as well as what is more realistic in the current situation and so we are directed to the right path which is the more realistic development of the Keeper application. Now, the application is focused on only the essential function of the application. Furthermore, sticking to the releases can actually get challenging especially when there are other deliverables that are needed to be met as well as various changes being made in the application. Therefore adjustments are inevitable and determination towards the project is needed. I also thank my group mates for exhausting their efforts to try to finish the project, they are awesome!
-Harris Brent Lao

The app’s predilection towards its belief on Rousseau’s theory of humans being innately good has made our theme much easier to convey. We believe that it is just right that we do the simplest and least technical mechanisms as well as reflect that all from the user’s standpoint to be able to grasp what truly makes life easier to live; and that is by the noble virtue of honesty and integrity within ourselves and each other.
-Pohl de Leon

Simple and direct is what this app is about. Taking into consideration the future is to better start a little rather than finish nothing. This app has the potential and will be useful as well. Nevertheless, I salute the group given the short time, Keeper was born.
-Jacq Dinglasan

Being the head programmer of this app was an interesting experience. It was a roller coaster ride. In principle, the logic was simple. It was a simple log-in-log-out type of app. However, the implementation was far from it. It made me realize my strengths and weaknesses, namely my lack in backend programming and the knowledge hitherto. However, it did make me grow as a person as it was my first time to be some sort of the moderator for a Github repository. I was able to work with a team. I also was able to experience more aspects of Software Engineering. Truly, Software Engineering is more than just programming and Computer Science.
-Brent Carl Anonas