Requirements Specification
for
Ineed App

October 2016
Version 0.1

Prepared By
Rabia KIRIM
Büşra GÜL

Change History

Date	Version	Description	Updated By
30 OCT 2016	0.1	Initial Draft	Rabia KIRIM Büşra GÜL

Document Approvals

Name	Role	Signature
Rabia Kırım	Developer	
Büşra Gül	Developer	
Rahime Belen Sağlam	Advisor	
Çağrı Aslan	Advisor	

Table of Contents

1 Introduction 4
1.1 Purpose 4
1.2 Document Conventions 4
1.3 Project Scope 4
1.4 References 4
2 System Description 4
3 Functional Requirements 5
3.1 System Features 5
3.1.1 System Feature 1 6
3.1.2 System Feature 2 7
3.1.3 System Feature 3 7
3.2 Use Cases 8
3.2.1 Use Case Diagrams 8
3.2.2 Use Case 1 8
3.2.3 Use Case 2 9
3.2.4 Use Case 3 10
3.2.5 Use Case 4 11
3.3 Entity Relationship Diagrams 12
3.4 Data Dictionary 13
3.4.1 Entity 1 13
3.4.2 Entity 2 13
4 External Interface Requirements 13
5 Technical Requirements (Non functional) 14
5.1 Performance 14
5.2 Scalability 14
5.3 Security 14
5.4 Maintainability 14
5.5 Usability 14
5.6 Multi lingual Support 14
5.7 Auditing and Logging 14
5.8 Availability 14
6 Open Issues 14

1 Introduction

1.1 Purpose

We aim to provide information about Ineed App.

1.2 Document Conventions

The aim of SRS document is to specify the software requirements of the project "General Purpose Ineed App". Basically, it gives detailed description of the project. Moreover, the goal and the qualification of the system, the constraints and interfaces of the system and what the system will do are clarified in this document. The explanation of the functions that the software will perform helps to see whether the software meets its requirements. This document specifies all the requirements pre-design, during design, programming and testing.

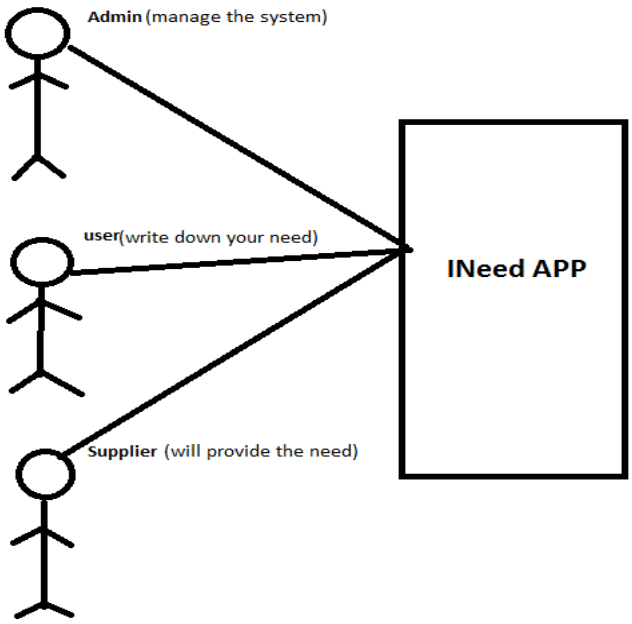
1.3 Project Scope

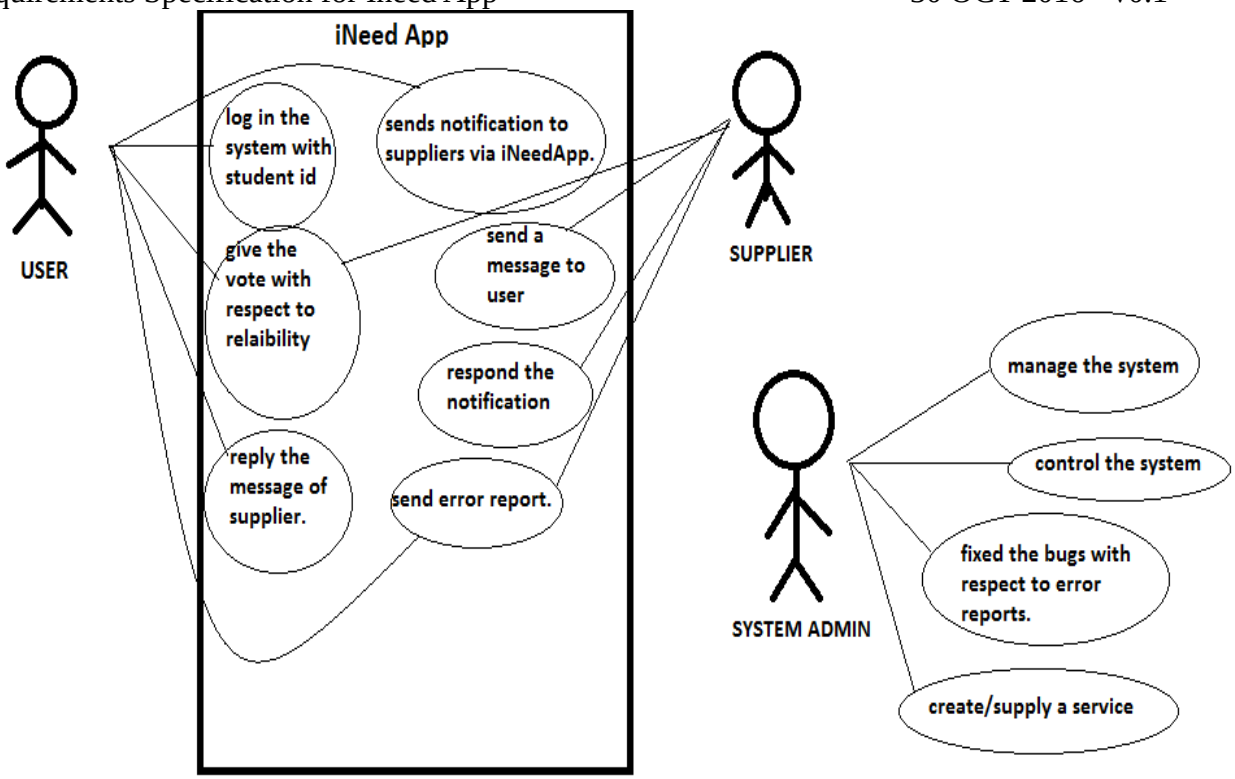
In campus life people sometimes need some things like pencil, ruler, calculator etc. Instead of going and asking for people to find needed item, a mobile application will be used for this purpose. When a person need something they will specify the need in the mobile app and notifications will be sent to nearby people. People can chat within the app to meet and resolve the need. They can also see the others' position (optional) corresponding to their location. (i.e 2 minutes to your position) Helpers will get points.

1.4 References

- *http://www.chip.com.tr/haber/nereden-baslamali-eclipse--android-studio_50226_2.html*
- *https://www.mobilhanem.com/android-studio-ide-kurulumu-ve-yeni-proje-acma-resimli/*
- *https://gelecegiyazanlar.turkcell.com.tr/konu/android/egitim/android-201/android-icin-gelistirme-ortaminin-kurulmasi*

2 System Description





3 Functional Requirements

Interface requirements

- *Screens will be native*
- *Screens will be understandable*

Business Requirements

- The next person will provide communication.
- Android media to be used.
- Data will be entered with TextBox.
- Messaging and notification system.

Regulatory/Compliance Requirements

- The database will have a functional audit trail.
- The system will limit access to authorized users.

3.1 System Features

- To be developed in android environment
- Messaging and notification.
- It will be the person finding the nearest person.

3.1.1 System Feature 1

NOTIFICATION IN ANDROID

Step 1 - Create Notification Builder

As a first step is to create a notification builder using *NotificationCompat.Builder.build()*. You will use Notification Builder to set various Notification properties like its small and large icons, title, priority etc.

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
```

Step 2 - Setting Notification Properties

Once you have **Builder** object, you can set its Notification properties using Builder object as per your requirement. But this is mandatory to set at least following −

- A small icon, set by **setSmallIcon()**
- A title, set by **setContentTitle()**
- Detail text, set by **setContentText()**

```
mBuilder.setSmallIcon(R.drawable.notification_icon);
mBuilder.setContentTitle("Notification Alert, Click Me!");
mBuilder.setContentText("Hi, This is Android Notification Detail!");
```

You have plenty of optional properties which you can set for your notification. To learn more about them, see the reference documentation for NotificationCompat.Builder.

Step 3 - Attach Actions

This is an optional part and required if you want to attach an action with the notification. An action allows users to go directly from the notification to an **Activity** in your application, where they can look at one or more events or do further work.

The action is defined by a **PendingIntent** containing an **Intent** that starts an Activity in your application. To associate the PendingIntent with a gesture, call the appropriate method of *NotificationCompat.Builder*. For example, if you want to start Activity when the user clicks the notification text in the notification drawer, you add the PendingIntent by calling **setContentIntent()**.

A PendingIntent object helps you to perform an action on your applications behalf, often at a later time, without caring of whether or not your application is running.

We take help of stack builder object which will contain an artificial back stack for the started Activity. This ensures that navigating backward from the Activity leads out of your application to the Home screen.

```
Intent resultIntent = new Intent(this, ResultActivity.class);
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(ResultActivity.class);

// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
stackBuilder.getPendingIntent(0,PendingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(resultPendingIntent);
```

Step 4 - Issue the notification

Finally, you pass the Notification object to the system by calling NotificationManager.notify() to send your notification. Make sure you call **NotificationCompat.Builder.build()** method on builder object before notifying it. This method combines all of the options that have been set and return a new **Notification** object.

```
NotificationManager mNotificationManager = (NotificationManager)
```

```
getSystemService(Context.NOTIFICATION_SERVICE);

// notificationID allows you to update the notification later on.
mNotificationManager.notify(notificationID, mBuilder.build());
```

3.1.2 System Feature 2

MESSAGING IN ANDROID

Messaging apps do not run directly on the Android dashboard hardware. They are installed on a separate Android mobile device. When the mobile device is plugged into a dashboard, the installed messaging apps can offer services for viewing and responding to messages through the Auto user interface.

To enable your app to provide messaging services for Auto devices:

- Configure your app manifest to indicate that your app provides messaging services which are compatible with Android Auto dashboard devices.
- Build and send a specific type of notification for display on Auto devices.
- Configure your app to receive `Intent` objects that indicate a user has read or replied to a message.

3.1.3 System Feature 3

IBEACON

Is a protocol developed by Apple and introduced at the Apple Worldwide Developers Conference in 2013. Various vendors have since made iBeacon-compatible hardware transmitters - typically called beacons - a class of Bluetooth low energy(BLE) devices that broadcast their identifier to nearby portable electronic devices. The technology enables smartphones, tablets and other devices to perform actions when in close proximity to an iBeacon.

iBeacon use Bluetooth low energy proximity sensing to transmit a universally unique identifier picked up by a compatible app or operating system. The identifier and several bytes sent with it can be used to determine the device's physical location, track customers, or trigger a location-based action on the device such as a check-in on social media or a push notification.

One application is distributing messages at a specific Point of Interest, for example a store, a bus stop, a room or a more specific location like a piece of furniture or a vending machine. This is similar to previously used geopush technology based on GPS, but with a much reduced impact on battery life and better precision.

Another application is an indoor positioning system,which helps smartphones determine their approximate location or context. With the help of an iBeacon, a smartphone's software can approximately find its relative location to an iBeacon in a store. Brick and mortarretail stores use the beacons for mobile commerce, offering customers special deals through mobile marketing, and can enable mobile payments through point of sale systems.

The first commercial implementation of bluetooth beacons was by an Australian company called DKTOB (trading as Daelibs), who leveraged Bluetooth for indoor proximity sensing in its Seeknfind location attendance solution. Daelibs designed and manufactured a bluetooth beacon for use in shopping centres based on the Bluegiga chipset. In 2012 Daelibs filed its Bluetooth beacon patent.

iBeacon differs from some other location-based technologies as the broadcasting device (beacon) is only a 1-way transmitter to the receiving smartphone or receiving device, and necessitates a specific app installed on the device to interact with the beacons. This ensures that only the installed app (not the iBeacon transmitter) can track users, potentially against their will, as they passively walk around the transmitters.iBeacon compatible transmitters come in a variety of form factors, including small coin cell devices, USB sticks, and generic Bluetooth 4.0 capable USB dongles.

3.2 Use Cases

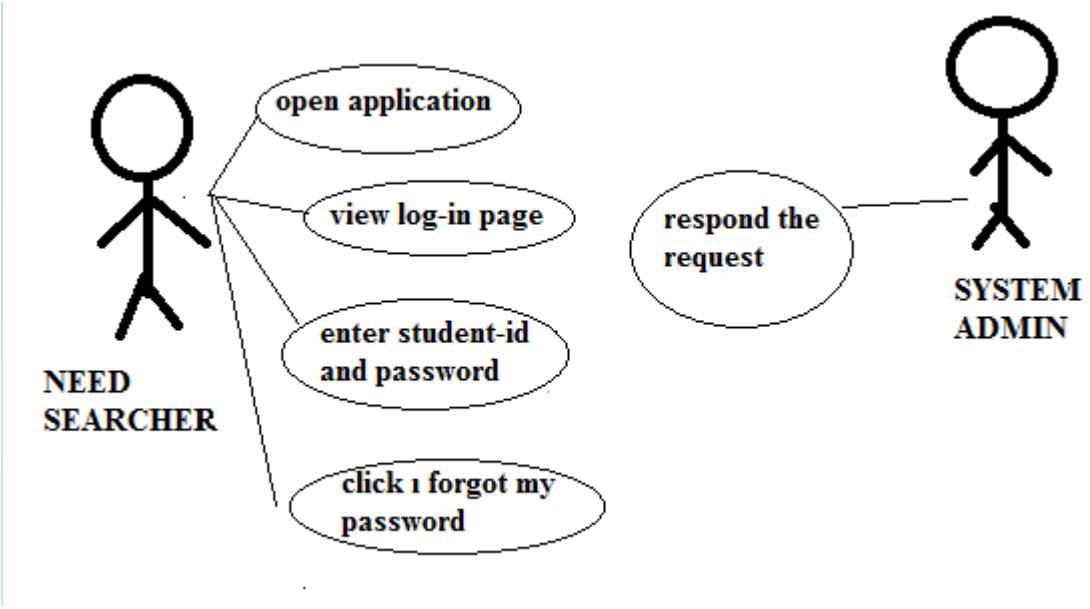
3.2.1 Use Case Diagrams

The first digram is the main success diagram.

3.2.2 Use Case 1

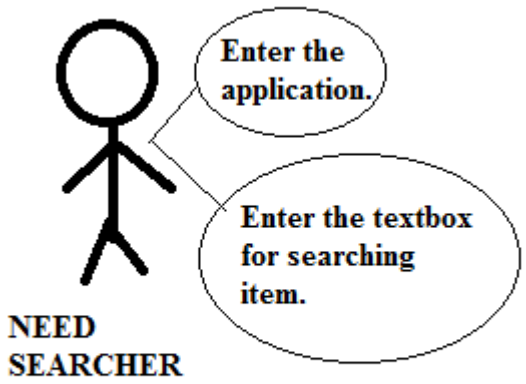
ID	UC 3.2.2.1
Description	This use case includes the enter the need for iNeed App and communıcate the supplier.
Actors	Need Searcher Supplier
Preconditions	-Need searcher be a member of the system. -Need searcher must be able to enter need to textbox part. -Supplier searcher be a member of the system.
Basic Steps	1.Need searcher enters the system. 2.Need searcher reaches the textbox on the system. 3.Need sercher will enter yhe need a textbox. 4.This requirement falls to the nearest person as a notification. 5.The nearest person reaches need searcher with a message. 6.Need Searcher reaches the product.
Alternate Steps	**1a. Need searcher may not be entered into the system successfully.** 1)The system gives the warning to try again. **4a.There may not be anyone who can provide the need.** 1)The system gives the warning "No one is nearby".
Postconditions	-Need searcher must have entered need the system. -The supplier should have sent a message.

3.2.3 Use Case 2



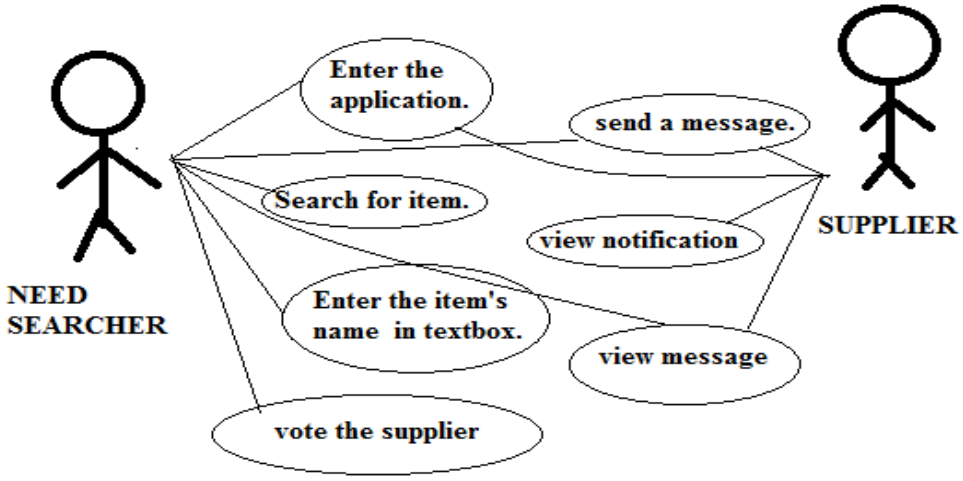
ID	UC 3.2.2.2
Description	This use case includes try to the enter the need for iNeedApp .
Actors	Need Searcher System Admin
Preconditions	-Need searcher must be able to enter the system.
Basic Steps	1.Need searcher enters the system.
Alternate Steps	**1a Need searcher may forgot his/her password** 1)Need Searcher press the 'I forgot my password.' button and sends a request for learning his/her password. 2)System admin respond the message. **1b Need Searcher may not have an account.** 1)Need Searcher press the register and register the system. 2)System admin respond the registering request and register the user.
Postconditions	-Need searcher must be entered in the system.

3.2.4 Use Case 3



ID	UC 3.2.2.3
Description	This use case includes the enter the need for iNeedApp and search it.
Actors	Need Searcher
Preconditions	-Need searcher be a member of the system. -Need searcher must be able to enter need to textbox part. -Supplier searcher be a member of the system.
Basic Steps	1.Need searcher enters the system. 2.Need searcher reaches the textbox on the system. 3.Need searcher will enter the need a textbox. 4.Entered item which is in textbox not found.
Alternate Steps	**3a.The entering item not found nearby!** 1)Application send an error message like 'No one has this item right now!'. 2)Application will not send any notification to suppliers.
Postconditions	-Need searcher must have entered need the system.

3.2.5 Use Case 4



ID	UC 3.2.2.4
Description	This use case includes the enter the need for Ineed App and communicate the supplier.
Actors	Need Searcher Supplier
Preconditions	-Need searcher be a member of the system. -Need searcher must be able to enter need to textbox part. -Supplier be a member of the system. -Supplier be able to sending messaging.
Basic Steps	1.Need searcher enters the system. 2.Need searcher reaches the textbox on the system. 3.Need searcher will enter the need a textbox. 4.This requirement falls to the nearest person as a notification. 5.The nearest person reaches need searcher with a message. 6.Need Searcher reaches the product.
Alternate Steps	**5a Any suppliers will not messaging to need searcher.** 1)Application warns the nearest suppliers like 'Need Searcher X need your help!' and shows the 'send message' button. **6a.Need Searcher and supplier did not meet.** 1)Need Searcher votes badly this supplier and application shows the other ones again. 2)Need Searcher selects one of them and process the steps again.
Postconditions	-Need searcher must have entered need the system. -The supplier should have sent a message. -Need searcher should vote the supplier.

3.3Entity Relationship Diagrams

<One or diagrams to depict all the entities in the system and their relationships. You can use different notations for ER diagrams. Following is a very simple ER diagram in Crow's notation.>

3.3 Data Dictionary

3.3.1 Entity 1

<A brief description of the entity followed by a table containing all its attributes as shown below. >

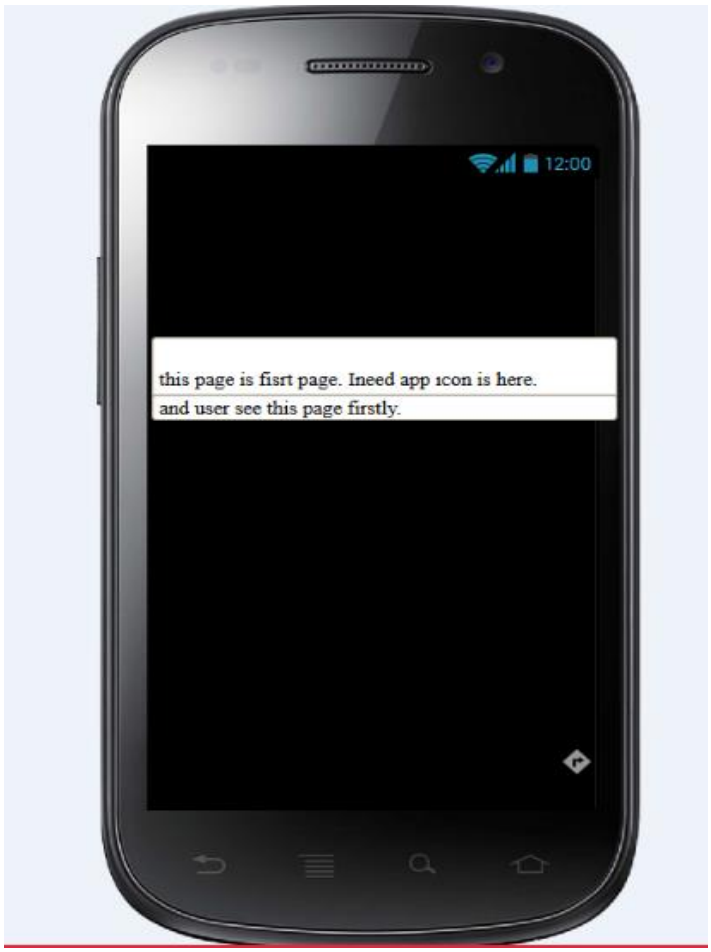
Attribute	Type	Optional?	Notes
<Attribute Name>	*<Data type of the attribute>*	*<Y or N>*	*<Explain any specific restrictions or rules applcable on this attribute>*

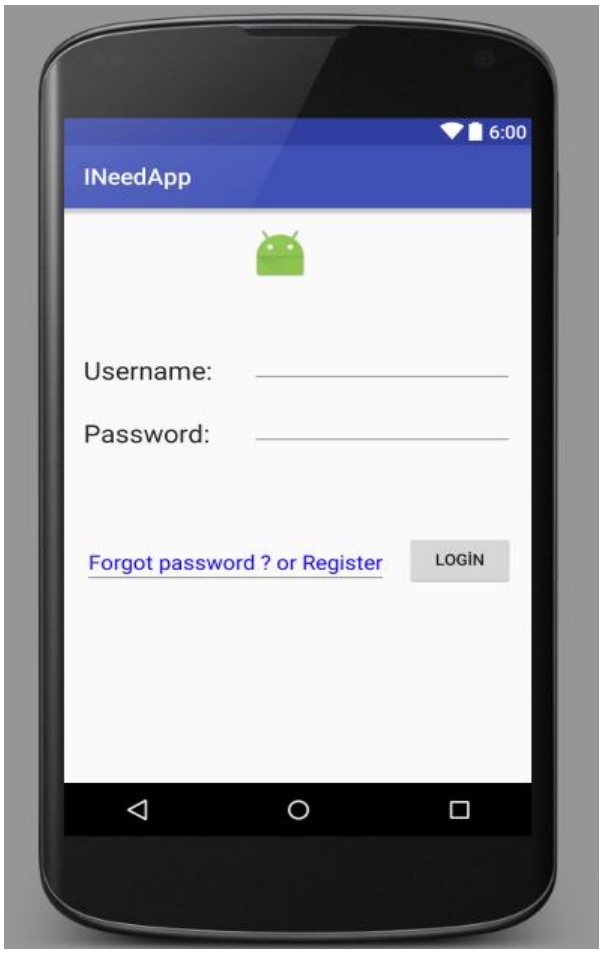
3.3.2 Entity 2

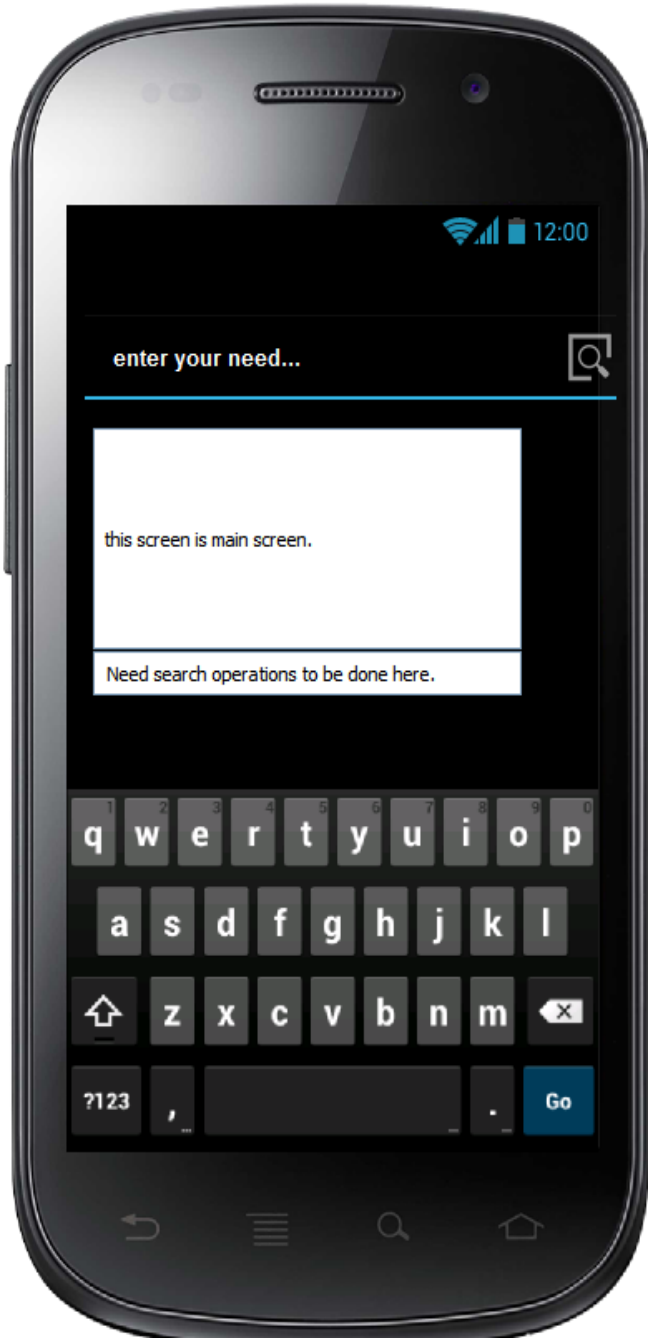
<A brief description of the entity followed by a table containing all its attributes as shown below. >

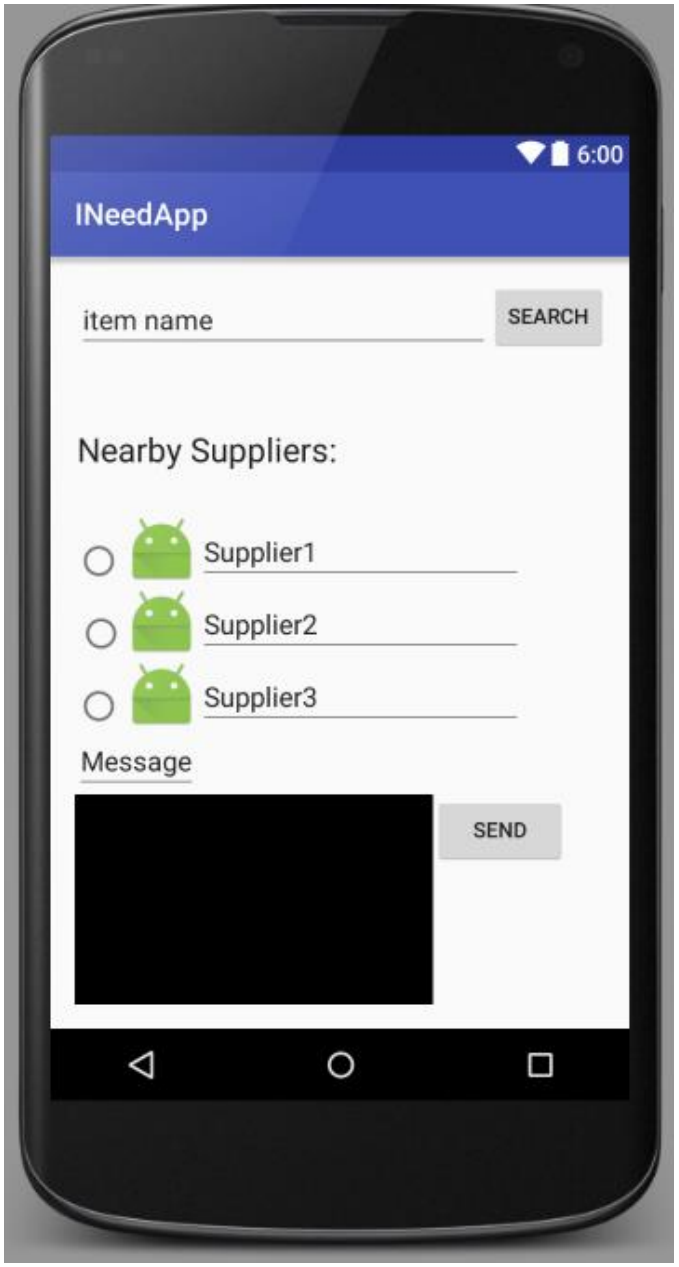
Attribute	Type	Optional?	Notes
<Attribute Name>	*<Data type of the attribute>*	*<Y or N>*	*<Explain any specific restrictions or rules applcable on this attribute>*

4 External Interface Requirements









5 Technical Requirements (Non functional)

<Please note that all the following subsections may not be applicable for a system. Sometimes you will have to add additional sections (for example, Legal requirements)>

5.1 Performance

<For example, What is the response time required?>

5.2 Scalability

<For example, how many users the system should support after two years of operation?>

5.3 Security

<For example, is data encryption required?>

5.4 Maintainability

5.5 Usability

5.6 Multi lingual Support

<What are the languages that software system should support?>

5.7 Auditing and Logging

5.8 Availability

<For example, Is any kind of downtime acceptable or required?>

6 Open Issues

<There could be open issues even at the end of the requirements elicitation. List of all of them here so it can be tracked and closed later. Some of these issues may later become project risks as well.>