Konnected – Synchronize Android Users

Dodangoda D.A.P.P.
Department of Computer Science and Engineering
University of Moratuwa
Sri Lanka
pubudu.11@cse.mrt.ac.lk

***Abstract*—This paper discusses the project, "Konnected" which is an Android based application for synchronizing several Android users into a single private network. The goal is to maintain a To-Do list which allows concurrent access and modification from a group of users. This application also uses GPS technology to track the location of users and the To-Do tasks as an additional functionality. The solution is built as a decentralized system which can be launched by a group of users without any risks of privacy or information leakage.**

Keywords—Android; GPS; to-do list

I. INTRODUCTION

This paper presents on how the Konnected Android application was designed and developed, the core concepts and design measures used to assure non-functional requirements. Preceding sections explain the quality assurance measures taken to verify the stability of the application.

The key objective of this application is to provide Android users a To-Do list which has a set of attributes which can be modified by users and the changes done will be visible to a selected group of users instantly to avoid possible conflicts due issues in synchronizations.

Early researches suggested that the similar tools available in the market, primarily for managing a To-Do list have several shortcomings which restricts users from getting a complete experience of freedom. Inability to mark locations, lack of synchronization and poor use of notifications stand out most as the reasons for the need for an alternative application such as Konnected.

This application runs in devices which has an operating system newer than Android 2.2.

II. LITERATURE REVIEW

A. Insight from real world

Imagine a Pizza Restaurant which provides a delivery service. There can be several delivery boys working for the restaurant. The restaurant can get multiple requests for pizza for a single day. This will not be a problem as far as a single branch is considered. The delivery boy can mark a delivery as complete or in progress manually while visiting the branch to take pizza from there. So nobody else will try to do the same task.

But what will happen if there are multiple branches? If a user orders a pizza online, it will be shown to the delivery boys as a task to be completed. In this case two or more delivery boys can mistakenly try to deliver the same order without knowing that some other agent is also trying to the same task. Can this be prevented? Can technology be of any help? These were the main motives behind the Konnected application.

B. Key Idea

In today's world, sharing information can be considered as the most requested and used service from software applications. This can be pointed out as the reason for spreading and growing the interest towards social media websites and applications. In specific, the highest interest is towards sharing dynamic information. A very common feature that exists among almost all such sources is that these are applied only for general use and entertain. However this approach can be very useful and productive if used in the business world.

As explained in the practical scenarios, connecting and synchronizing multiple Android users can be very productive in business societies as well as in general. Yet there are no applications so far which allow a fully customizable feature of connecting users. Of course there are several applications which use this principal for their own tasks.

Many companies have embarked on initiatives that enable more demand in information sharing between retailers and their upstream suppliers [1]. The four primary information sharing design patterns are sharing information one-to-one, one-to-many, many-to-many, and many-to-one. Technologies to meet all four of these design patterns are evolving and include blogs, wikis, really simple syndication, tagging, and chat. Thus it is worthwhile to examine and analyze about popular applications which have been successful in this area.

C. Similar Products Available

Following is a list of extremely popular applications which connect multiple users. Nevertheless it can be noted that it is not their primary task. Connecting users is done in a way such that only a small requirement is fulfilled.

1) Facebook: Facebook constitutes a rich site for researchers interested in the affordances of social networks due to its heavy usage patterns and technological capacities that bridge online and offline connections [2]. Facebook

allows users to create and organize user groups. There are options to get notifications and share information. Facebook has several features related to geographical location, but no option to affiliate them with tasks.

2) WhatsApp: An instant messaging smartphone application [3]. It can share information such as texts, images and audio. Can be considered as a chat application.

3) Google Drive: Provides users, the ability to access, store, collaborate, and disseminate data [4]. Cannot share dynamic resources and get related notifications.

It is to be noted that currently none of these or other applications provide the expected functionality of the "Konnected" application.

D. Target Platform

Today, almost everything depends on time and efficiency. Even the earnings and profit of a business system depend on how efficient the system is. Therefore people value time over every other aspect. Mobile phones and similar mobile devices were invented as a solution to save money. Therefore now it is the responsibility of software developers to make those high performance hardware devices even better by creating better and efficient software to run on them.

One of the most successful mobile operating systems is the Google Android operating system which is a free to use open source OS [5]. Therefore the Android OS was selected as the platform for this application and hence the Android Software Development Kit (SDK) was also used along for the application development.

Android SDK is a software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use which runs on top of a Linux kernel [6].

E. Approach

The approach for developing this application is pretty much similar to the incremental model. Incremental model in software engineering is a one which combines the elements of waterfall model which are then applied in an iterative manner. It basically delivers a series of releases called increments which provide progressively more functionality for the client as each increment is delivered [7].

F. Testing and Quality assuarance

Robotium solo [8] 5.2.1 was used to test this Android application. Robotium is a tool which uses automated testing feature to test the functionalities of different functions and activities.

Robotium is well known as an Android test automation framework that has full support for native and hybrid applications. Robotium makes it easy to write powerful and robust automatic black-box UI tests for Android applications. With the support of Robotium, test case developers can write function, system and user acceptance test scenarios, spanning multiple Android activities [8].

III. Requirement Specification

This section describes the Requirement Specification which was launched at the initial phase of the project. Please note that there can be a few differences from the following requirements in the final product which was released recently.

A. Scope

This is an Android application which allows several smartphone users to join a personal network and synchronize details about their geographical position and some other related features.

B. Functional Requirements

1) Login and Signup: Users can log in to the application by giving valid a user name and a password, or can create an account if it does not exist. Users can join to a specific network by giving the required credentials.

2) Shared To-Do List: Users can add a To-Do task by giving the required parameters (Title, Description, Priority, Location on the Map, etc.). It will be updated on every other user's device. Then another user can mark the task as "acquired" and literally start doing the intended task. After completion he or she can mark the task as "Accomplished". If anything goes wrong, it can be marked as "Failed" so that another user can acquire the task.

3) Location based notifications: Users can get location based notifications about the to-do tasks when he or she is closer to a location which is marked on any of the to-do tasks. The notification message may include the following details:

- task name and description
- location details and distance from current location
- the user who initiated the task
- current progress of the task

4) Shortest Path: Shortest path and other related details such as shortest path distance and time to travel can be calculated and/or displayed on a map. This can be related to shortest path to a to-do task or the shortest path to some other member of the group.

C. Non-Functional Requirements

1) Usability: Any smart phone user with Android operating system can easily operate the application. Additionally, the application will be designed in such a way that unnecessary interaction by the users is not expected. An internet connection is a must for this application to perform

correctly. A functionality to retrieve data from the central database and update a local database can be added later.

2) Performance: Continuous internet connectivity is required as the users are synced through the internet. We can expect latencies in cases where the internet connection is not speed enough. An Android device has many methods to connect internet such as Wireless Fidelity (Wi-Fi), General packet radio service (GPRS), and 3rd Generation (3G). This application must support all these connecting methods in order to function in every case possible.

3) Security: Possible ways of information leakage must be identified and steps must be taken in order to prevent any sorts of security threats to the system.

D. Interfaces

1) User Interface: A user can log into the app via the login screen. Then the user will be forwarded to the main screen where the user can join a team. Also, there will be a user friendly icon set to perform the functionalities provided by the application.

2) Hardware Interface: Android device which supports internet connectivity and Secure Digital Card (SD) storage card is sufficient.

3) Communication Interface: Hypertext Transfer Protocol (HTTP) is used for feed downloads. For establishing internet connection Wi-Fi, 3G or GPRS must be used.

IV. System Design Process

This section describes the requirements and objectives that have some significant impact on the architecture of the application.

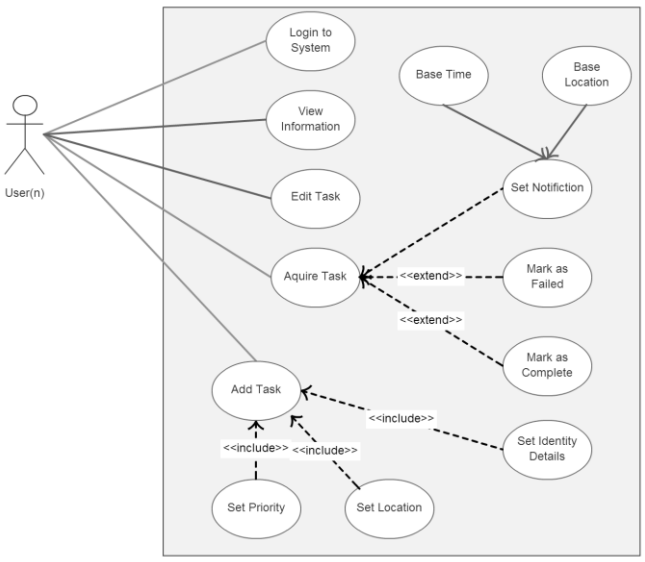

Fig. 1. Use case Diagram for the application.

A. Use Case View

The users can log into an account and once they have, they can choose any from several available options. The main 3 functions as per described in the above diagram are, Add Task, Acquire Task and update ToDo list. Other general functions such as leaving the account and set visibility are also possible. The main 3 use cases are complex use cases which include and extend several other use cases (*See* Fig. 1).

B. Logical View

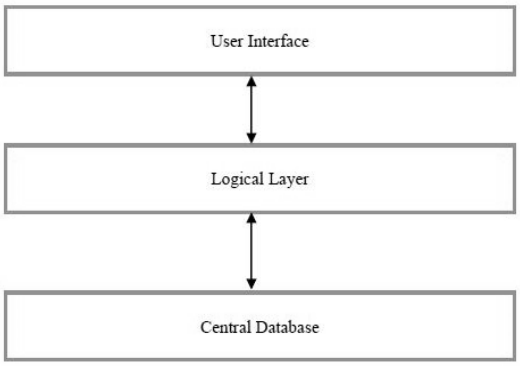

Fig. 2. Layered View of the application.

This application consists of 3 main layers (*See* Fig. 2). They are User Interface, Logical Layer and Central Database Top layer is the User Interface layer which is used by the user to interact with the system. Middle layer is the logical layer. The logical layer manages communication between users and the system. The database layer stores data about To-Do tasks, Notifications, the user accounts, and user behaviors.

C. Architecturally significant Design Packages

1. UI Layer

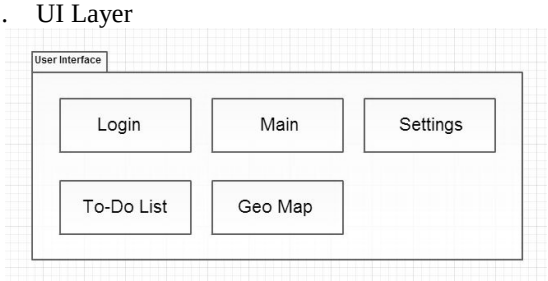

Fig. 3. Package Diagram of the UI Layer.

2. Logical Layer

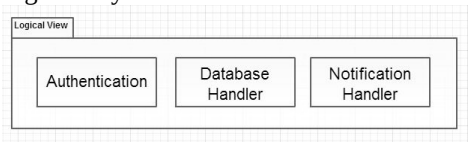

Fig. 4. Package Diagram of the Logical Layer.

D. Process View

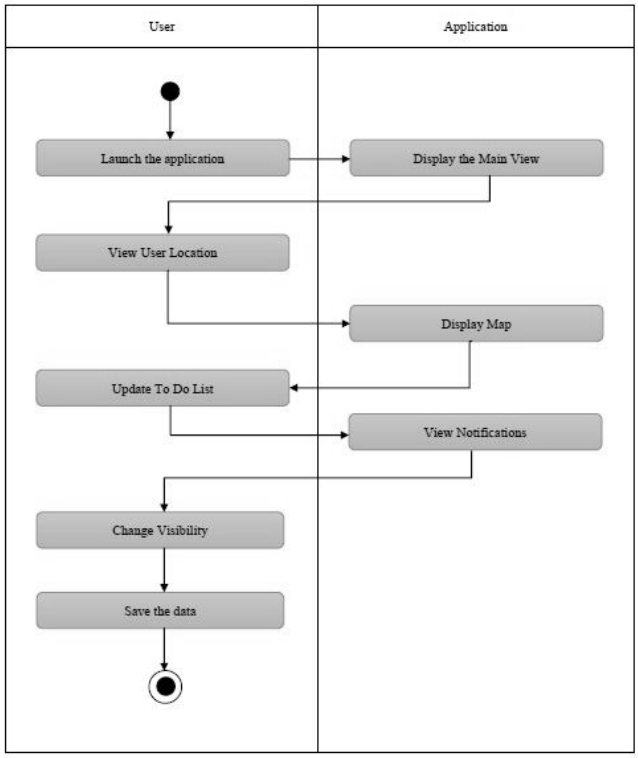

Fig. 5. Activity Diagram showing the Workflow of activities.

E. Implementation View

Overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model of this system is same as the logic view of the system. UI layer compromises of classes extended from the activity class. The Logical layer manages the behaviors and communications among users via the To-Do list provided by the application.

F. Quality and Performance

Overall quality of the system is improved because of the layered architecture. Capabilities that are improved by the software architecture are

- Performance.
- Increased abstraction level.
- Reusability.
- Extendibility.

Performance mainly depends on the bandwidth and the signal strength of the internet connection. Memory will also be used for rendering maps. Since there is a notification system, the application will need to be running in the background. As a whole, this app will need moderate hardware support. Lowest recorded hardware specifications of a successful launch of the application are 256MB RAM and 1 GHz CPU.

V. Software Implementation Process

This section explains the implementation process of the software step by step. The application software was developed

according to the RUP model. But in this section details related to requirement analysis, designing and testing are discarded though all these were done in parallel.

A. Logging In To the Main Menu

The initial focus was on the overall structure of the application. It was decided to design separate xml activity for each important function of the software. As a first step the login activity was designed using xml and the java code was written to move from this activity to the main activity when the required credentials are given. In the main activity 4 image buttons were created to move to the main functions of the application.

After the successful completion of this task, it was decided to implement the structure of all the activities. Thus the main menu was designed with links to the 4 main activities, Todo List, Notifications, Map, and Settings activity (*See* Fig. 6).

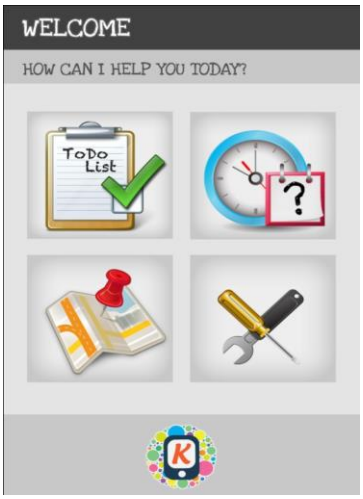

Fig. 6. Main Menu Screen.

B. To-Do List

The main function of the application was initiated at this stage, the shared to-do List. This was mostly based on the java code instead of xml designs with contrast to phase1. A class was implemented with the necessary attributes to hold all the information related to an item in the to-do list. An adapter class was used to hold the necessary information and display an entry of the To-Do list. Inside the adapter class an inner class was implemented to hold required information of an entry. The only intention of creating an inner class was the ease of access. Every object created from the main class was added to a specific ArrayList and the data was stored inside the adapter class for creating the entry. After the implementation of the above tasks, the resulting screen on the Android device is shown below in Fig. 7.

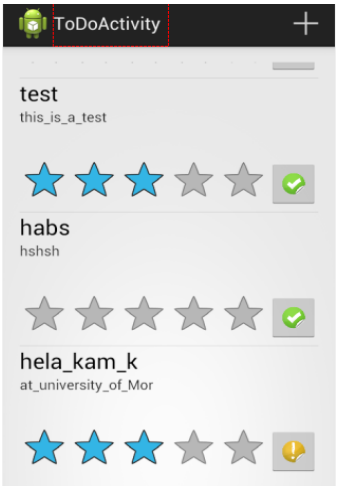

Fig. 7. To-Do Activity Screen.

Status of the task is the most important feature of this task item. When a user taps on the status button, the image as well as the database should be updated. That is the current status should be switched between the three options, In-progress (acquired), Accomplished and Failed (Not started) as shown in the images below (see Fig. 8).

In-Progress Accomplished Not Started

Fig. 8. Icons of 3 possible states.

C. Database Connection

As the next step, the database connection was setup with the application. A MySQL database was created along with the required tables. The importance of the database should be highlighted. The main function of this software application is to connect several Android users into one system. Thus a shared data center is a must have feature. This was achieved through the centralized database.

Communication of information with the database was done with php scripts. Data is sent and received as JSON objects which adds some additional structure to the data. However, this communication can become slow if done synchronously. Therefore the classes were inherited from AsyncTask class to achieve performance through asynchronous access. Writing data to the database was also done in a similar manner using JSON objects.

D. Google Map Activity

The next important task done was the configuration of the map activities. A location is added to each To-Do item as another attribute. The initial intension was to get location based notifications from the To-Do tasks added. Before that, the functionality to get user location and intended location was implemented. Some additional code was written to pan and zoom the map to focus on to the Sri Lankan map.

Handling permissions had to be done in order to let these expected functionalities flow smoothly. Permission allocation was done using the Android manifest.

As the next step, the map was associated with every To-Do item added which allows tracking location corresponding to each To-Do item and required notifications.

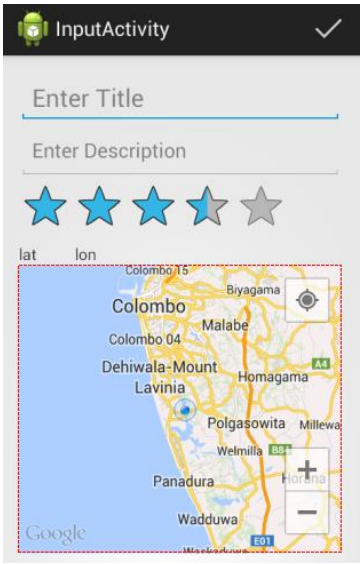

Fig. 9. Map functionality associated with adding a new item.

As shown in Fig. 9, each To-Do item finally consisted of a title, a description, a priority (out of 10), and a geographical location. Only the title was mandatory when adding a new entry.

E. Location Based Notifications

Location based notifications were added to the to-do tasks in the final phase of the application. When a user reaches a location of the marked to-do items, he or she will receive an Android notification with a reminder to this task including all the related information of the task, specially the priority of the task. The main concept behind this notification system is Interrupts.

VI. System Evaluation and Conclusions

A. Testing

In parallel to the implementation of the application, Testing was carried out as a major activity. Unit testing, Automated testing and Beta testing was carried out to assure quality of the application.

In addition to unit testing, manual testing procedures were carried out using the LOGCAT which is provided along with the Android SDK. Print streams and logs were of added help with the process.

Automated testing was carried out using Robotium Solo. Test cases for Robotium was written in a similar manner to Unit Test cases. The main target of automated testing is to test screen navigation of the application.

Beta testing was carried out with the help of several volunteer Android users from the Department of Computer Science and Engineering, University of Moratuwa. The main issue raised by several users was the screen compatibility. This was addressed and solved by configuring the related xml files which defined the layouts of the application.

B. Conclusions

Several concerns has to be addressed when developing an application to the Android mobile platform. Screen compatibility, Permission handling, User Experience, Application Stability, and Information Privacy stands out to be the most important facts which should be handled carefully. In addition to the application development process, the needs of the community should also be considered to provide the most suitable solution to the target problem.

However the application must be improved to be able to handle requests coming at a high frequency. Issues that can arise due to the concurrent access must also be taken into consideration as a possible improvement to the application.

ACKNOWLEDGMENT

The initial idea for the project was very abstract. Therefore I must thank Dr. Indika Perera for adding details and inspiring me with a lot of creative ideas. I must also thank Mr. Dumindu Buddika and Mr. Chanaka Cooray for helping me with the Google Map related work. Finally I thank all volunteer Android users who helped me out with beta testing for the "Konnected" Android application.

REFERENCES

[1] Hau L. Lee , Kut C. So, Christopher S. Tang, "The Value of Information Sharing in a Two-Level Supply Chain," *Management Science*, vol. 46, no. 5, pp. 626-643, 2000.

[2] Nicole B. Ellison, Charles Steinfield, Cliff Lampe, "The Benefits of Facebook ‘‘Friends:’’ Social Capital and College Students’ Use of Online Social Network Sites," *Journal of Computer-Mediated Communication*, vol. 12, no. 4, pp. 1143-1168, 2007.

[3] Kenton P. O'Hara, Michael Massimi, Richard Harper, Simon Rubens, Jessica Morris, "Everyday dwelling with WhatsApp," *In Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pp. 1131-1143, 2014.

[4] Darren Quick, Kim-Kwang Raymond Choo, "Google Drive: Forensic analysis of data remnants," *Journal of Network and Computer Applications*, vol. 40, pp. 179-193, 2014.

[5] "webopedia," 2014. [Online]. Available: http://www.webopedia.com/TERM/A/Android_SDK.html. [Accessed 10 June 2015].

[6] "Android Developers," Google Inc., [Online]. Available: https://developer.android.com/about/android.html. [Accessed 10 June 2015].

[7] R. Tilloo, "Technotrice," 2015. [Online]. Available: http://www.technotrice.com/incremental-model-in-software-engineering/. [Accessed 11 June 2015].

[8] "Robotium," Github, [Online]. Available: https://github.com/robotiumtech/robotium. [Accessed 11 June 2015].