Course Assignment I (Mock-Up)

Introduction

This document typifies a documentation of the conceptual design process and specific requirements for an Android Bluetooth (BT) Chat App with focus on User Experience (UX). The conceptualize design is a user experience design which would include textual description of the application wireframe and task flow display. The specific requirements would focus on the application's functional and nonfunctional requirements with brief mention of the environment's software and hardware requirements. As part of the user requirement spec, use cases would be presented and explained. And finally, a brief review of the app's User Interfaces (UIs) would be evaluated against the 10 Heuristics for User Interface Design for completeness.

User Context and Use Case Model

Communicating users with ad hoc network devices or users in a group such as PAN (personal area network) could often feel the need to communicate through free unlicensed bandwidth such as using Bluetooth technology. Another intriguing scenario is when the associated cost of network connectivity is to be completely avoided for communications among users with smart phones in close proximity to one another. It is no secret that users like it simple perhaps because as the saying goes "simple is beautiful" The overarching goal of this conceptual design is to make communication possible, easy and simple for users in the above categories. Figure 1 represents a simple use case model which would only require users to install the app, connect to other devices, and send text/data to other devices (one at a time) from their devices.

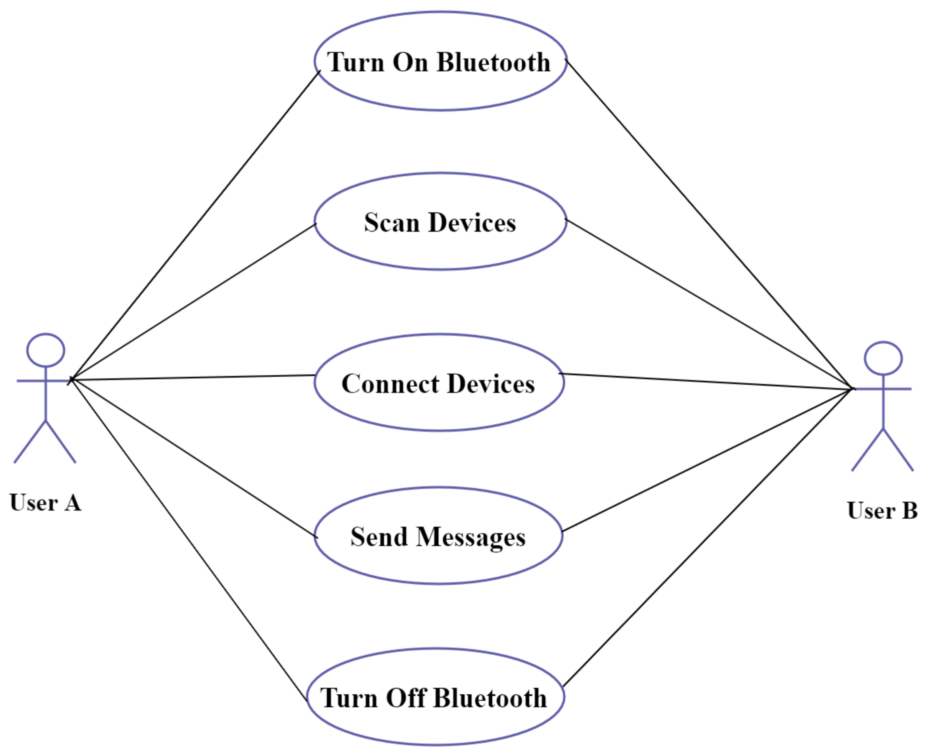

Figure 1. Use case model

Table 1. Use cases description

Use Case	Description
Turn on Bluetooth	If the BT of the device is in OFF mode, the app should make request to enable BT otherwise it should be ready to scan for devices
Scan Devices	The app should perform scanning for the devices which are in their range and displays the list of them, both previously “paired” and newly “discovered” devices
Connect Devices	Selecting and clicking on any of the displayed devices should establish a chat session with the chosen device
Send Messages	Two-way text messages/data should easily be exchange with the already established connection
Turn Off Bluetooth	When done with chatting and transfer of data the BT should be turn OFF

Functional Requirements

Functional requirements represent the most important user stories which in this case is represented by the use cases highlighted in table 1.

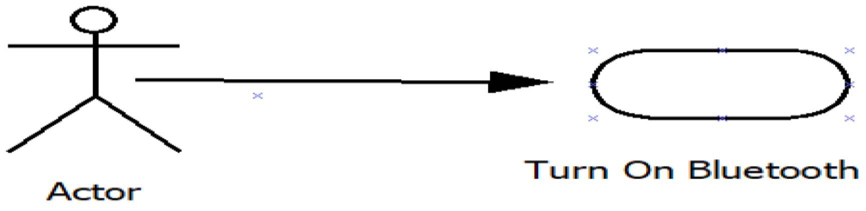

Figure 2. Use case: Turn on Bluetooth

Table 2. Use case: Turn on Bluetooth

Use Case Number	1
Use Case Name	Turn on Bluetooth
Summary	User needs to turn on device Bluetooth
Actor	Any user
Trigger	At the request of the app via “allow” button or device’s BT icon
Precondition	Installation of the Bluetooth Chat App
Scenario	• After installing the app user needs to open/initialize the app • App request for BT to be turned on if not already turned on • BT is turned on at the request of the app if not already turned on • BT Chat App makes no request when BT is already on
Postcondition	After BT is turned on the user starts to use the app to discover and connect to other devices

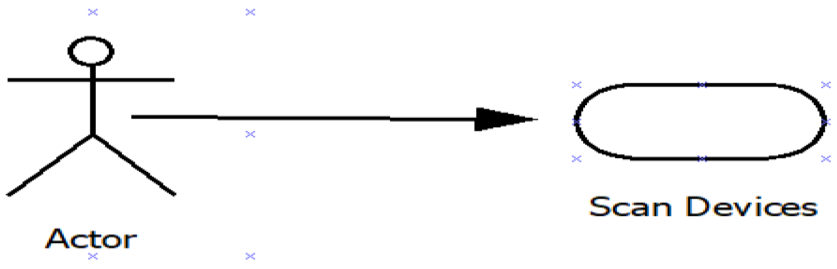

Figure 3. Use case: Scan devices

Table 3. Use case: Scan devices

Use Case Number	2
Use Case Name	Scan devices
Summary	Scan for other devices within range
Actor	Any user
Trigger	Connect button
Precondition	App initiation
Scenario	• At the click of the "connect" button the app displays the names and MAC addresses of previously discovered devices and newly scanned/discovered devices
Postcondition	After app discovers devices, the user can then choose which other user to chat and exchange text messages with

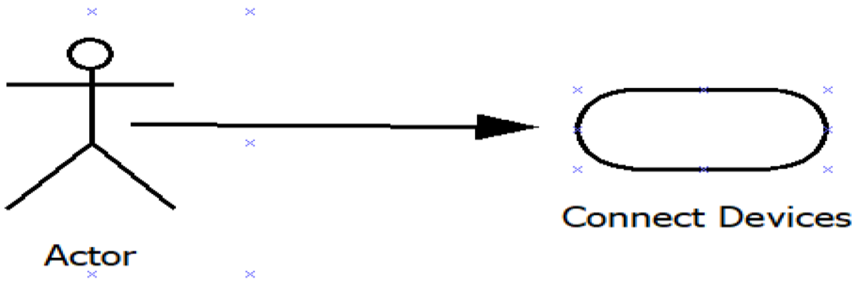

Figure 4. Use case: Connect devices

Table 4. Use case: Connect devices

Use Case Number	3
Use Case Name	Connect devices
Summary	User connects to device (one at a time) to communicate with
Actor	Any user
Trigger	Clicking on any active discovered device
Precondition	Other connecting device must be available for connection
Scenario	• User can inform the other party of possible connection to be certain of connectivity • In an uncertain situation the user can try connecting with presumably active device
Postcondition	User sends messages

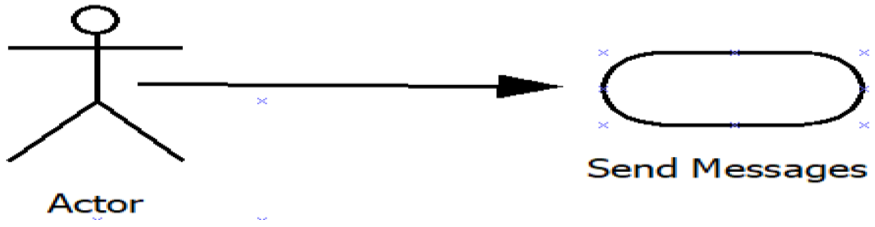

Figure 5. Use case: Send messages

Table 5. Use case: Send messages

Use Case Number	4
Use Case Name	Send messages
Summary	Send typed text messages
Actor	Any user
Trigger	Send button
Precondition	Connection to another device
Scenario	• User writes a message by clicking on the text field • Sent and received messages are made visible in space on the app

Postcondition	Received messages can be read and responded to

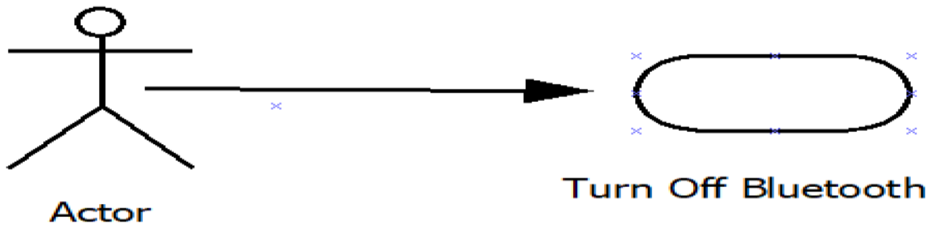

Figure 6. Use case: Turn off Bluetooth

Table 6. Use case: Turn off Bluetooth

Use Case Number	5
Use Case Name	Turn off Bluetooth
Summary	Turn Bluetooth off
Actor	Any user
Trigger	Device BT icon
Precondition	Done with all communications via BT
Scenario	• User disengages from all communications through BT by deactivating BT via the device BT icon
Postcondition	Close app

Nonfunctional Requirements

Parts of the nonfunctional requirements that would be highlighted here are usability, reliability, performance, supportability (i.e. implementation quality and robustness), technical complexity and security.

- **Usability:** This app is a mobile app designed specifically for Android users with interfaces written in simple plain English. Thus, all that is required of the users is to simply know how to use mobile device and be able to read and write text messages. The app is simple and easy to use as well as devoid of any requirement to become productive or proficient at its usage.

- **Reliability:** Once installed, this app does not require or need internet or GSM connection to run. It would be available 24 hours a day, 7 days a week on a mobile device. There is reliable data transfer between devices due to BT connectivity reliability
- **Performance:** Messages are intended to be delivered in splits of a seconds and since the app is meant to be lightweight, the CPU speed and RAM capacity of the device should not be of great concern
- **Implementation quality and robustness:** The programming language(s) to be used in the app development is object-oriented; thus, each task is meant to be independent of each other and as such, easy to maintain. Codes are meant to be readable and easy to understand and all design architecture are expected to be well documented as well
- **Technical complexity:** the implementation strategies are meant to follow a simple line of implementation for easy readability of codes and understandability of activities with little or no dependencies
- **Security:** connection to other devices could only be establish via authentication by some authentication key generation

Wireframe

This wireframe section is intended to describe in brief the main user interface information as visually presented on the wireframe structure and layout of the app under designed for development. Figure 7 is a designed view of the wireframe for the app to be developed.

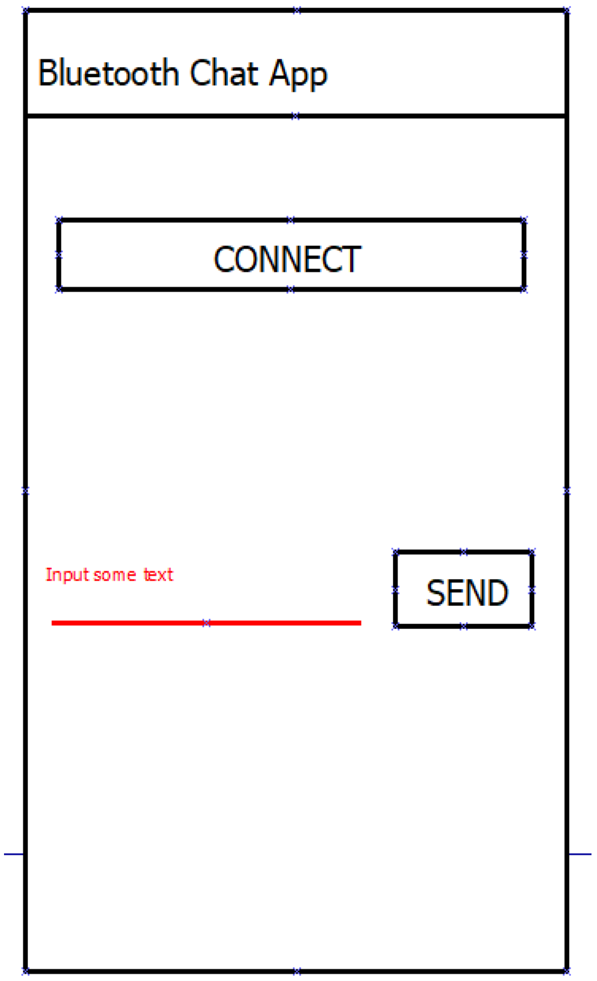

Figure 7. BT Chat App wireframe

The wireframe in figure 7 exhibits some user interfaces such as the “connect”, “send” and “input some text” interfaces. Although not shown in figure 7, the operation of this app would also depend on such interfaces as “hardware”, “software” and “communication” interfaces.

User Interface

Connect Interface: This is a connection interface that enables users to query local Bluetooth adapter for paired BT devices and scan for other BT devices as well. It displays paired devices and help establish RFCOMM channel or sockets made ready for data transfer over Bluetooth between connecting remote devices.

Input Some Text Interface: This is a messaging interface that enables users to edit and type messages into a text field during chatting over a mobile device through Bluetooth connectivity.

Send Interface: This is the interface that provide users with the ability and the platform to send already typed messages to other remotely connected Bluetooth devices.

Message Exchange Display Interface: This is the dynamic space that exit between the connect and the send (plus input text) interfaces that enables the user to view exchanged messages and allows the users to scroll through pass current and old exchanges.

Navigational map/task flow chat of app

Figure 8 is an example of a device connections and data exchange between two users with device A and device B respectively.

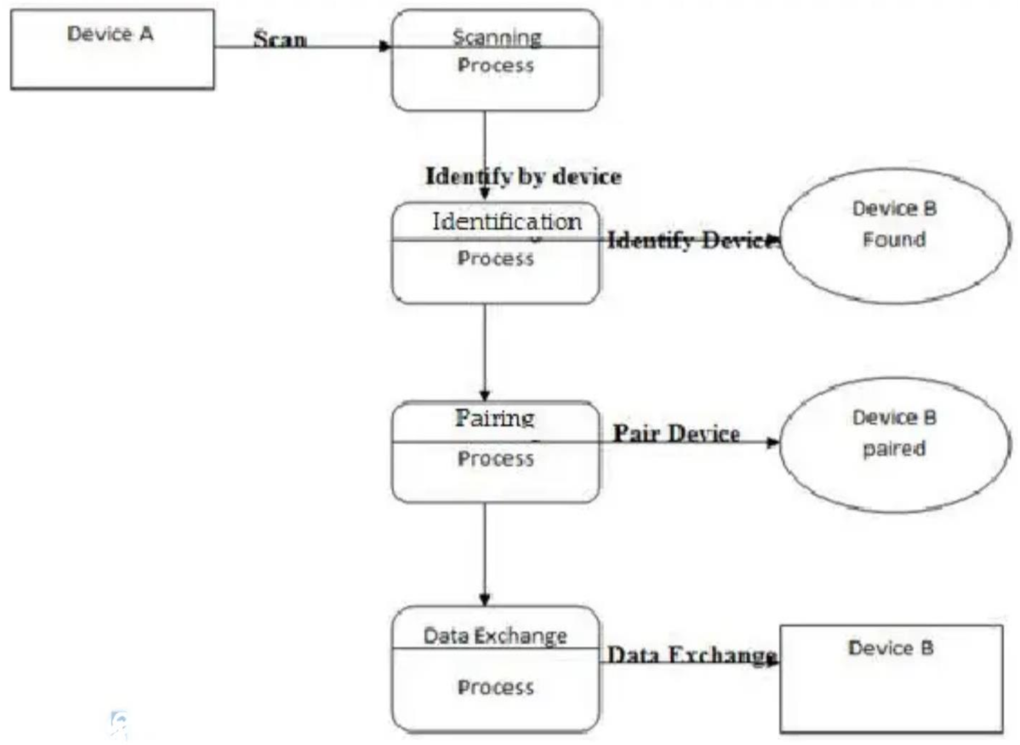

Figure 8. An example of a task flow chart for Bluetooth Chat app

A Review against the 10 Heuristics for User Interface Design

At this point in the design of this app, a review of the user interfaces design for usability may be inevitable against some recognized usability principles otherwise known as heuristics or heuristics evaluation. This app is being evaluated under the following heuristics:

- **Visibility of system status:** when initialized, this app is meant to:
  - Ask for permission to turn on Bluetooth if not already turned on (a form of statutory report)
  - When granted the permission to do so it should show the system status of turning on Bluetooth
  - A "not connected" should be displayed at the upper left-hand corner to indicate that the device is yet to be connected to any other device.
  - Clicking on connect button and when choosing a device to connect to fails the app should report such
  - All the above are vital information on the status of the system made visible for users to see and know about the current state of the app workings
- **Match between system and the real world:** All the above information about the system status are expected to be written in natural language such as English and presented in logical format or form.
- **User control and freedom:** during text message exchange messages can easily be deleted and retyped at the wish of the user before being sent. However, to undo sent messages may be impossible as the exchanges are expected to happen in splits of a seconds. Message edit before being sent appears to be a best practice in this case.
- **Consistency and standard:** the connect and send interfaces as well as the input text interface meant exactly what they represent preventing users from guessing what their functions are.
- **Error prevention:** users should get notifications of connectivity failure in order to avoid sending typed messages in error. This is an error prevention mechanism that the app is expected to possess
- **Recognition rather than recall:** this app is designed to be much less memory and CPU intensive. Besides, users are not expected to be proficient at using device or chat app to be

able to use this app. The simplicity of its layout and structure make it easy for any user to use.

- **Flexibility and efficiency of use:** talking about use, every user can develop their skills to pass messages quicker or faster than another user without any major impact on the app.
- **Aesthetic and minimalist design:** the layout and structure of this app looks compact, simple, with fewer features and expected to work beautifully.
- **Error identification and recovery:** for example, when any of the app's interface is unavailable the app can easily be reinitialized using the device back button. This simply enables users to know what went wrong and how it can be fixed.
- **Help and documentation:** This app interfaces do not include help, however ,this writing could serve as a documentation for it.

Reference

Krupali. 2014. Jakob Nielsen's 10 Usability Heuristics Explained. Available online at https://www.slideshare.net/crafted/10-usability-heuristics-explained

Nielsen, J. 1994. 10 Usability Heuristics for User Interface Design. Available online at https://www.nngroup.com/articles/ten-usability-heuristics/

Sahu, P. 2015. Bluetooth Chatting System: Mobile App Development Using Android Technology. Available online at https://www.slideshare.net/PanchhiSahu/presentation1-47146996

Selvaraj, S. 2011. Ten Usability Heuristics with Examples. Available online at https://www.slideshare.net/sacsprasath/ten-usability-heuristics-with-example?next_slideshow=1

Tosun, C; Sabirsiz, G. & Shaidolda. 2015. Software Requirements Specification V1.0 NoNET. IEEE Standard 830-1998 vol. 1 (1).