Software Design Document

Version 2.0

May 15, 2015

Rohan Gadad

MS Candidate, Information Technology

Rensselaer Polytechnic Institute

Prepared for

ITWS\_6940, Software Engineering

Instructor: Ingrid Liu, Ph.D.

Spring 2015

Contents

[**1.** **Introduction** 3](#_Toc420022058)

[**1.1** **Purpose** 3](#_Toc420022059)

[**1.2** **Scope** 3](#_Toc420022060)

[**1.3** **Out of Scope** 3](#_Toc420022061)

[**1.4** **Project Deliverables** 3](#_Toc420022062)

[**1.5** **Project Management Plan Updates** 4](#_Toc420022063)

[**1.6** **Definitions and Acronyms** 4](#_Toc420022064)

[**2.** **System Architecture** 5](#_Toc420022065)

[**2.1** **Android OS Architecture** 5](#_Toc420022066)

[**2.2** **Notelity Use Case Diagram** 6](#_Toc420022067)

[**2.3** **Design Rationale** 6](#_Toc420022068)

[**3.** **Data Design** 7](#_Toc420022069)

[**3.1** **Data Design** 7](#_Toc420022070)

[**3.2** **Data Descriptions** 7](#_Toc420022071)

[**4.** **Component Design** 8](#_Toc420022072)

[**4.1** **Class Descriptions** 8](#_Toc420022073)

[**4.2** **Detailed Class Descriptions** 9](#_Toc420022074)

[**4.3** **Class diagram** 12](#_Toc420022075)

[**4.4** **Sequence Diagram** 13](#_Toc420022076)

[**5.** **Human Interface Design** 14](#_Toc420022077)

[**5.1** **Overview of User Interface** 14](#_Toc420022078)

[**5.2** **Screen Images** 14](#_Toc420022079)

[**5.3** **Screen Objects and Actions** 15](#_Toc420022080)

[**References** 15](#_Toc420022081)

[Figure 1: Android Architecture[1] 5](#_Toc420022086)

[Figure 2: Use Case Diagram 6](#_Toc420022087)

[Figure 3: Class Diagram 12](#_Toc420022088)

[Figure 4: Sequence Diagram 13](#_Toc420022089)

# **Introduction**

This Software Design Document (SDD) establishes the overall design of the system to be used during the development of the Notelity Android application for the Software Engineering Independent Study of Spring 2015 under the guidance of Dr. Ingrid Liu at Rensselaer Polytechnic Institute.

## **Purpose**

The aim of this project is to develop a productivity app on the Android platform. The app is called Notelity - the basic purpose of the app is to allow the users to take notes. After the development of the app, it will be deployed on the Google Play Store.

## **Scope**

1. View the titles of the notes
2. Modify existing notes or create new notes and store them locally in a SQL/XML based persistent storage
3. Each note will be able to store text
4. Real time search functionality to search for titles and content of the notes
5. Social sharing functionality to share the note on social media sites or email/text messages

## **Out of Scope**

1. Maintenance of the application after the deployment
2. The app will not be expected to work on mobile devices which do not support Android platform

## **Project Deliverables**

Below is a list of artifacts to be delivered

1. **Project documentation**: A complete and up-to-date documentation of the project design and implementation
2. **Source code**: The final working source code of the project
3. **APK file**: The final Android APK file for installation on any Android based smartphone

## **Project Management Plan Updates**

The Project Management Plan will be created and maintained using standard version control processes. Page 3 of this document outlines the Document History, Distribution List, and Plan Approvers.

## **Definitions and Acronyms**

**SDK:** Software Development Kit

**API:** Application Programming Interface

**APK:** Android Application Package

**Android SDK:** 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.

**Android API Level:** API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform. The framework API consists of a core set of packages and classes.

**Shared Preferences:** The SharedPreferences class provides a general framework that allows to save and retrieve persistent key-value pairs of primitive data types.

# **System Architecture**

## **Android OS Architecture**

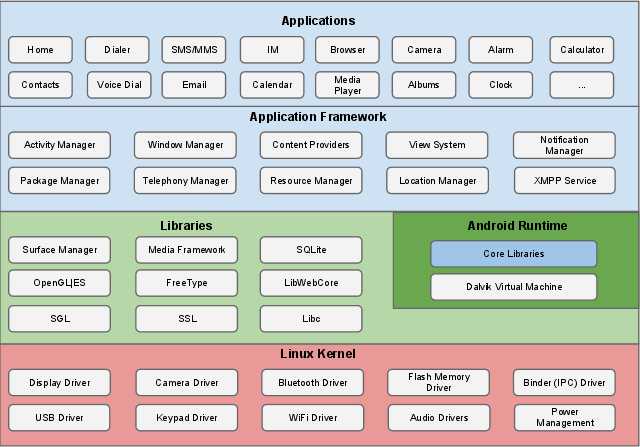


Figure 1: Android Architecture[1]

Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation, Android is designed primarily for touchscreen mobile devices such as smartphones and tablet computers, with specialized user interfaces for televisions (Android TV), cars (Android Auto), and wrist watches (Android Wear).

## **Notelity Use Case Diagram**

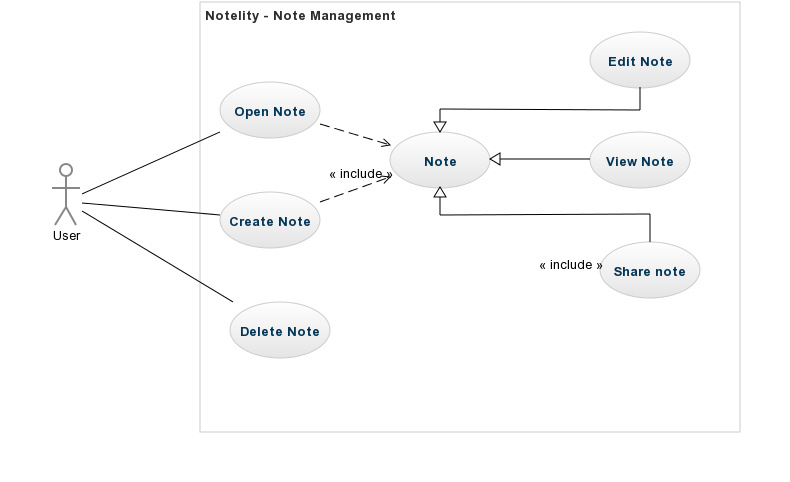


Figure 2: Use Case Diagram

## **Design Rationale**

The main goal during the development of this app has been simplicity. A note taking app should present the user simple options so that the user can focus on being productive than concentrating on navigating the app. All the features developed for the app take this into account and present the user a safe and simple way to store his data. Though simple, the app is not feature constrained. Advanced features such as real time search, and social sharing features are also included to increase the appeal of the app to advanced users too.

# **Data Design**

## **Data Design**

The SharedPreferences class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. SharedPreferences can be used to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if the application is killed).

Shared preferences are not strictly for saving "user preferences," such as what ringtone a user has chosen. User preferences can be created for the application using PreferenceActivity, which provides an Activity framework to create user preferences, which will be automatically persisted (using shared preferences).

To get a SharedPreferences object for your application, use one of two methods [2]:

**getSharedPreferences()** - Use this if you need multiple preferences files identified by name, which you specify with the first parameter.

**getPreferences()** - Use this if you need only one preferences file for your Activity. Because this will be the only preferences file for your Activity, you don't supply a name.

## **Data Descriptions**

**Body:** This represents the body of the note. In this section the user writes/edits the main content of his note

**Date Created:** The date created is the key of the Shared Preferences

**Title:** The title is the first line of the note body. This is not stored explicitly and is parsed from the notes whenever required

# **Component Design**

## **Class Descriptions**

**Activities**

* + 1. **MainActivity**: The Main Activity loads up when the application is started. Upon start, the onCreate() function is called. The onCreate() function is equivalent a class constructor.

The onCreate() initializes other components such as the top menu, the titles of the notes, search functionality etc.

* + 1. **NoteEditorActivity:** The Note Editor Activity starts when the user taps on the note title. Here the user can edit the app or share the app using the Share button.

**Backend Classes**

1. **NoteItem:** The Note Item class is a data structure for the notes. It has 2 parts to it – the key and the value. The key is the date/time stamp stored as a string. The value is a multi line string which represents the title and body of the note.
2. **NotesDataSource:** The Shared Preferences is implemented here in Notes Data Source. It also provides update, remove and find functions to interact with the stored data.

**Helper Classes**

1. **MyAdapter:** The My Adapter class is a custom implementation of the Android Adapter. This is used to update the view of the Main Activity. Here the row colors, dates of the notes are combined with the titles and a single view is created which is used whenever the Main Activity is called.

**Other UI Components**

Several XML files are used to represent the UI of the application. There are also some images to represent the ‘Create Note’ or ‘Share’.

## **Detailed Class Descriptions**

**MainActivity class**

EDITOR\_ACTIVITY\_REQUEST: This flag is sent to the SharedPreferences storage while editing and creating new notes

MENU\_DELETE\_ID: This flag is sent to SharedPrefences while deleting a particular note

currentNoteId: The row id of the item for which the context menu is being displayed.

datasource: An instance of the NotesDataSource class

button1: Title Sort

button2: Date Sort

titleFlag, dateFlag: Keep track of the type of sort already in place, so as to reverse it (ie ascending to descending and vice versa)

notesList: An ArrayList of notes indexed according to the date created, with the 0th index pointing to the 1st note created.

onCreate: Function called when the app is first opened (equivalent to main() function in normal programming language construct). Initializes the view of the app and loads the necessary functions and event handlers used by the app.

sortTitleDisplay: This function deals with sorting the note titles according to ascending or descending order alphabetically.

sortDateDisplay: This function deals with sorting the note titles according to ascending or descending order based on the date and time created.

refreshDisplay: A function which refreshes the view of the list of notes whenever a change has occurred (i.e. a note has been edited or has been deleted). It is called from within other classes as well.

searchFunction: This helper class is called from within OnQueryTextListener in the onCreateOptionsMenu class. This is activated when the user clicks on the search icon.

onCreateOptionsMenu: This is automatically called when the app is started. It also initializes the ‘New Note’ or ‘+’ icon and the Search widget in the menu bar.

onOptionsItemSelected: This is automatically called when an event that can occur at the Menu bar needs to be caught. From this method, other activities or functions can be started.

createNote: Creates a note when the ‘+’ icon is touched.

onListItemClick: Opens a note when a note title is touched

onActivityResult: When a note has been edited, the results are modified in the SharedPreferences

onCreateContextMenu: The delete option is created using the context menu when a user does a ‘long press’ on the title of the note.

onContextItemSelected: When the user presses the delete button, the note is deleted.

**NoteEditorActivity**

Note: An instance of the NoteItem class

onCreate: Function called when the activity is started (equivalent to main() function in normal programming language construct). Initializes the view of the editor activity and loads the necessary functions and event handlers used by it.

onCreateOptionsMenu: Manages the menu bar at the top

saveAndFinish: Function that is called when the user clicks on the back button. This creates a new intent and calls the MainActivity class.

onOptionsItemSelected: This is automatically called when an event that can occur at the Menu bar needs to be caught. From this method, other activities or functions can be started.

**NoteItem Class**

key, text: A key, value pair of the map. This is the main data structure of the app.

getKey, setKey, getText, setText: Getters and setters

getNew: Creates a new (key, value) mapping

**NotesDataSource Class**

PREFKEY: This is the name of our local data storage file.

notePrefs: An instance of Android’s build in SharedPreferences class. SharedPreferences is a simple type of persistent data storage available on Android. Persistent means that the data is not deleted when the app is exited. SharedPreferences uses XML.

NotesDataSource: Constructor. Here the SharedPreferences file is set as Private. This means that other apps cannot access it. This allows a decent level of security for the notes. However, it is important to the note that, a user may access these files from a debug mode.

findAll: Returns a List of all the notes

update: Updates a specific note when it has been modified

remove: Deletes a specific note once the user has chosen to delete it.

**MyAdapter Class**

context: Maintains the context of MainActivity, so that the view of MainActivity class can be updated from this class

data: An ArrayList of all the notes in the app

layoutResourceId: Maintains the ID of the ListView in the MainActivity

randomColors: Stores the last color assigned so that 2 consecutive rows do not have the same color

MyAdapter: Constructor.

getView: Creates the view on the MainActivity class. Sets the colors, matches the notes with the dates, updates the views etc.

ViewHolder: This is for the ‘Title Sort’ and ‘Date Sort’ buttons

getRandomColors: Assigns random colors for each row

## **Class diagram**

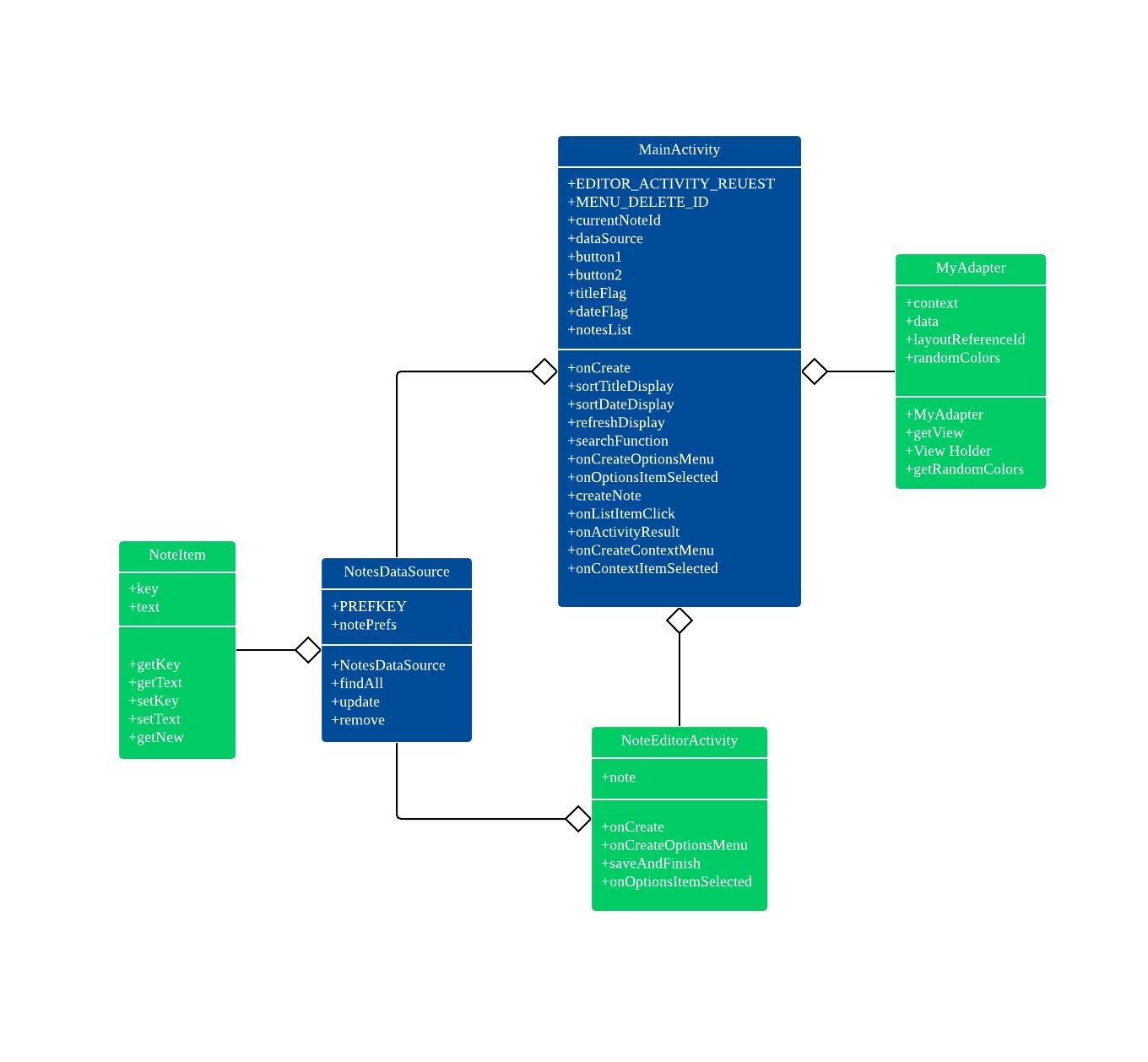


Figure : Class Diagram

## **Sequence Diagram**

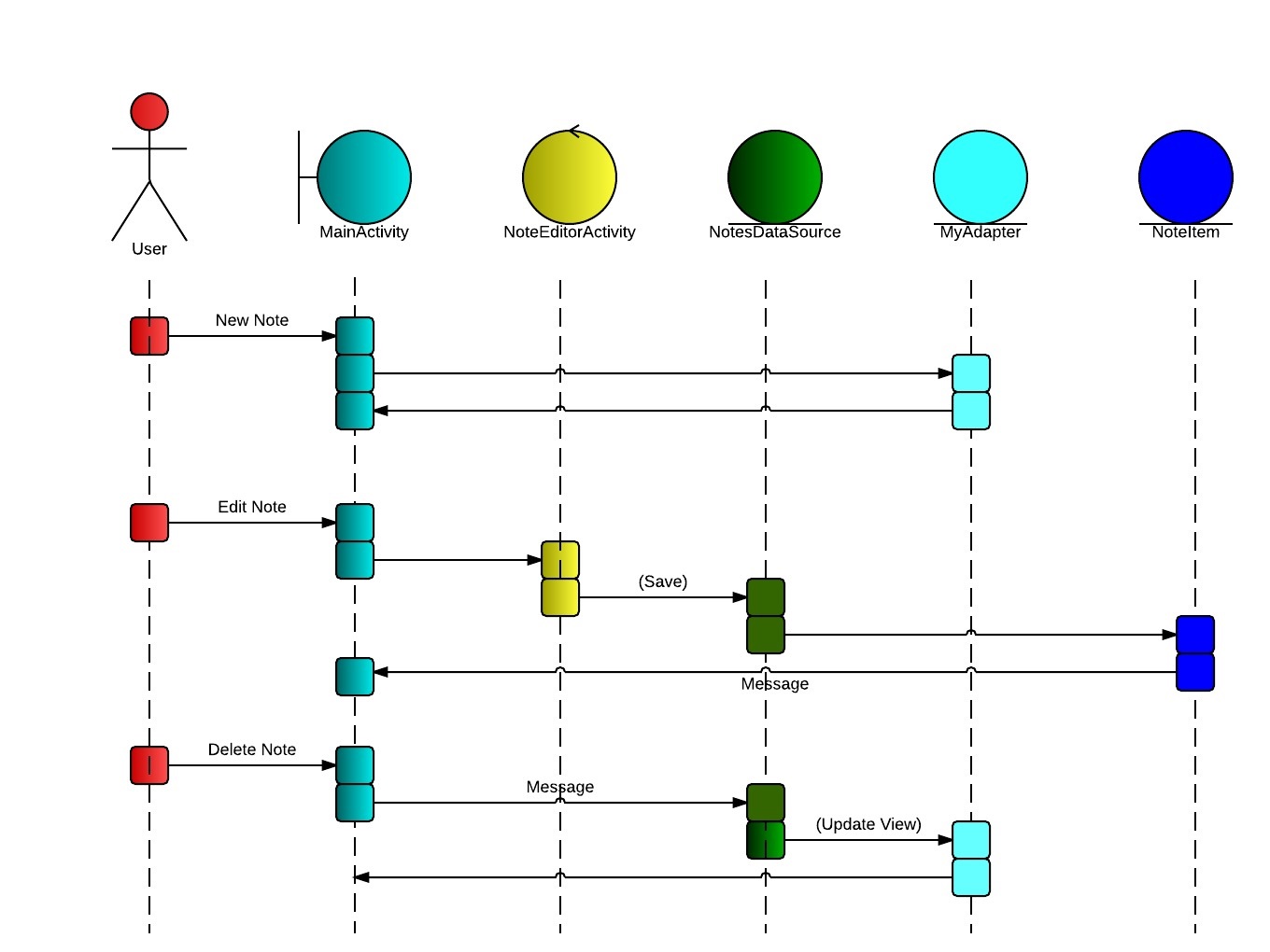
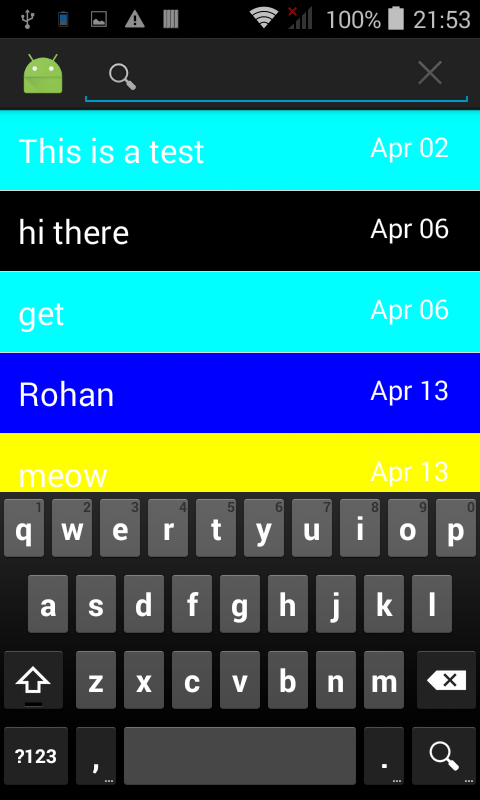
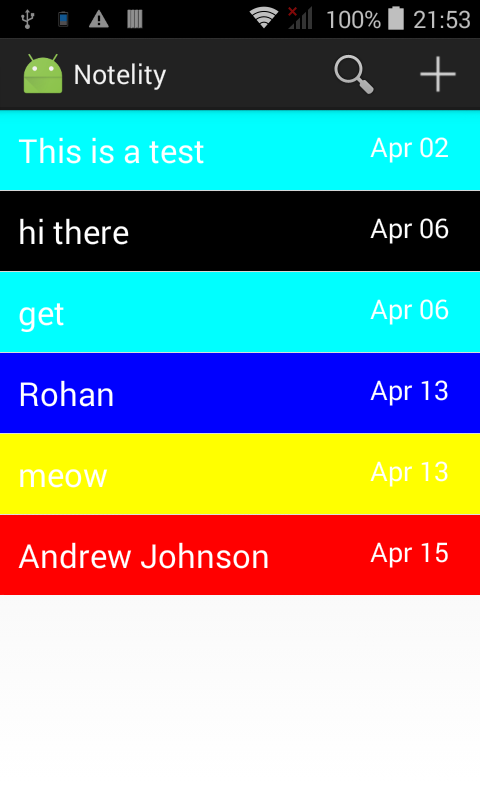


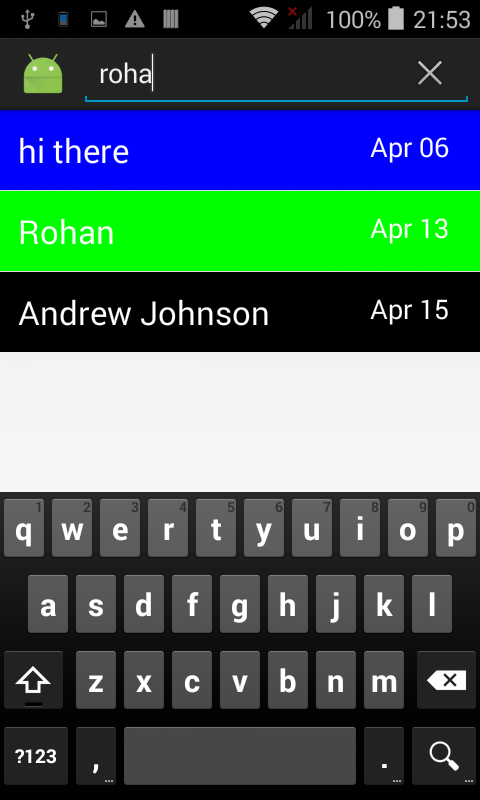
Figure : Sequence Diagram

# **Human Interface Design**

## **Overview of User Interface**

## **Screen Images**





## **Screen Objects and Actions**

# **References**

1. https://source.android.com/devices/tech/security/
2. http://developer.android.com/guide/topics/data/data-storage.html