

A mini project report on

“HELPING NEEDY - ANDROID APPLICATION”

Submitted to **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD** in the partial fulfillment of the requirements for the award of degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

**G. SUPRITH REDDY
B. BHAGYA SREE
B. PAVANI
K. ANITHA**

**14071A05J9
14071A05I7
14071A05M5
14071A05K5**

Under the guidance of

Dr. A. BRAHMANANDA REDDY
Associate Professor



Department of Computer Science and Engineering

VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY

**Vignana Jyothi Nagar, Bachupally, Nizampet (SO), Hyderabad – 500 090 (Approved by
AICTE & Govt. of A.P. and affiliated to JNTU, Hyderabad)**



**VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI
INSTITUTE OF ENGINEERING & TECHNOLOGY**

AN AUTONOMOUS INSTITUTE UNDER JNTUH

(Approved by AICTE, New Delhi and Govt. of A.P. & Affiliated to JNTUH) Vignana

Jyothi Nagar, Bachupally, Nizampet (S.O), Hyderabad - 500 090, A.P. India. Tel:

+91-40-23042758, 23042759, 23042760, Fax: 91-40-23042761

E-mail - postbox@vnrvjiet.ac.in,

Website: www.vnrvjiet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Mr. G. Suprith Reddy (14071A05J9), Miss. B. Bhagya Sree (14071A05I7), Miss. B. Pavani (14071A05M5) and Miss. K. Anitha (14071A05K5) have successfully completed their mini-project work at COMPUTER SCIENCE AND ENGINEERING Department of VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY, Hyderabad entitled “HELPING NEEDY - ANDROID APPLICATION” in partial fulfillment of the requirements for the award of BACHELOR OF TECHNOLOGY degree during the academic year 2017-2018.

This work is carried out under my supervision and has not been submitted to any other University/Institute for award of any degree/diploma.

Project Guide

Dr. A. BRAHMANANDA REDDY

Associate Professor

Department of CSE

VNR VJIET

HOD / CSE

Mrs. B.V. Kiranmayee

Associate Professor and Head

Computer Science and Engineering

VNR VJIET

DECLARATION

We hereby declare that the project report entitled “HELPING NEEDY - ANDROID APPLICATION” submitted by us to JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING department is a record of bonafide project work carried out by us under the guidance of Dr. A. BRAHMANANDA REDDY. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

G. Suprith Reddy (14071A05J9)

B. Bhagya Sree (14071A05I7)

B. Pavani (14071A05M5)

K. Anitha (14071A05K5)

ACKNOWLEDGEMENT

We should like to express our sincere gratitude to our Head of the department **Mrs. B.V. KIRANMAYEE** for granting us permission to do the project and for continuous encouragement.

We owe the success of our project to our guide **Dr. A. BRAHMANANDA REDDY**, Associate Professor for his guidance in completing our project work successfully and for allowing us to do project in his organization.

We owe a debt of gratitude to our project coordinator **Mr. M. RAVIKANTH**, Assistant Professor whose encouragement has been the principal source of inspiration of this project.

Finally, we express our thanks to all faculty members of the department of Computer Science and Engineering.

ABSTRACT

Several Non-Government organisations which include non-profit, voluntary organisations are providing services through their official sites and applications. If a user who wants to acquire those services must know the particular NGO website URL or link. If not, they must have a specific application to gain the services. To avoid this complication, by developing an application which includes most of the NGO services. Based on the request provided by the user, the respective NGO will respond with a specific answer within a short period of time. This application is plausible to implement with Android Studio using Java and XML languages for frontend and to store data i.e., backend can be implemented using Google's Firebase Cloud. Every individual would sign up for the application when he/she runs it for the first time using their mail id. From this project, we hope to provide an application through which every user would gain services as quickly as possible.

CONTENTS

TOPIC	PAGE NO
Certificate	ii
Declaration	iii
Acknowledgement	iv
Abstract	v
Contents	vi
List of Chapters	vii
List of Figures	ix
List of Tables	xi
List of Abbreviations	xii

LIST OF CHAPTERS

CHAPTERS	PAGE NO
1. INTRODUCTION	01
2. SYSTEM STUDY	
2.1 Feasibility Study	02
2.1.1 Economic Feasibility	02
2.1.2 Technical Feasibility	02
2.1.3 Social Feasibility	03
2.2 Software Requirements	03
2.2.1 Android Architecture	03
2.2.2 Application Framework	05
2.2.3 Android Runtime	06
2.2.4 Linux Kernel	07
2.2.5 Android Studio	07
2.2.6 Firebase	11
3. EXISTING SYSTEM AND PROPOSED SYSTEM	
3.1 Existing System	13
3.2 Proposed System	14
4. SYSTEM ANALYSIS	
4.1 Download Android Studio	15
4.2 Installing Android Studio	16
4.3 Running Android Studio	20
4.4 Starting a New Project	24
4.5 The Project and Editor Windows	28

5. SOFTWARE DESIGN (UML DIAGRAMS)	
5.1 Usecase Diagram	30
5.2 Sequence Diagram	31
5.3 Class Diagram	32
5.4 Activity Diagram	33
5.5 Component Diagram	33
5.6 Deployment Diagram	34
6. IMPLEMENTATION	
6.1 Forms and Reports (Sample Screenshots)	35
6.2 Code Implementation	
6.2.1 Common Java Files	43
6.2.2 Fragment Java Files	54
6.3 Firebase Data Storage (JSON Tree Format)	63
7. SYSTEM TESTING	67
CONCLUSION	69
BIBLIOGRAPHY	70

LIST OF FIGURES

S.NO	NAME	PAGE NO
2.2.1	ARM Architecture	04
2.2.2	Project files (Android)	09
2.2.3	Project files (Problems)	09
2.2.4	Main window (Android Studio)	10
4.2.1	Setup Android Studio	17
4.2.2	SDK and AVD	17
4.2.3	License Agreement	18
4.2.4	Install Locations	18
4.2.5	Shortcut for Android Studio	19
4.2.6	Finish	19
4.3.1	Start Screen	20
4.3.2	Import settings	20
4.3.3	SDK Validation	21
4.3.4	Installation type	21
4.3.5	Review Settings	22
4.3.6	Wizard downloads	23
4.3.7	Hardware Acceleration	23
4.3.8	Welcome Page	24
4.4.1	Create Project	25
4.4.2	Target device (selection)	25

4.4.3	Activity template	26
4.4.4	Customize activity	27
4.4.5	Workspace	27
4.5.1	Project and Editor window	28
5.1	Usecase diagram	30
5.2	Sequence diagram	31
5.3	Class diagram	32
5.4	Activity diagram	33
5.5	Component diagram	33
5.6	Deployment diagram	34
6.1.1	Homepage	35
6.1.2	Register User	36
6.1.3	Register NGO	37
6.1.4	Sign In	37
6.1.5	View Posts	38
6.1.6	React to a post	39
6.1.7	Add post	39
6.1.8	Adding post	40
6.1.9	Post added	41
6.1.10	Your posts	41
6.1.11	No of reacts	42
6.1.12	Delete post	43
6.3.1	Root Node	63

6.3.2	Posts Node	63
6.3.3	Posts Child Node	64
6.3.4	React Node	64
6.3.5	Reacts Child Node	65
6.3.6	User Node	65
6.3.7	Users Child Node	66

LIST OF TABLES

S.NO.	NAME	PAGE NO
7.1	Tests Performed	68

LIST OF ABBREVIATIONS

NAME	ABBREVIATION
JDK	Java Development Kit
SDK	Software Development Kit
AVD	Android Virtual Device
W2A	Welcome to Android
JSON	JavaScript Object Notation
ARM	Android Runtime Machine
IDE	Integrated Development Environment
GCM	Google Cloud Messaging
FCM	Firebase Cloud Messaging
API	Application Programming Interface
APK	Application Package
VM	Virtual Machine
GNOME	GNU Object Model Environment

1. INTRODUCTION

Helping Needy App is helpful in making communication among normal users and NGOs easier and faster. Any registered user (either NGO or normal user) can login into our application and use the services that are included in the application such as adding posts/requirements, viewing and reacting to other posts and receiving the reactions and the contact details of reacted member for your posts. You can also delete your posts immediately after your requirement has been accomplished.

This application mainly focuses on efficiency rather than UI, as the services provided would be delivered at faster rates. The emergencies services such as need for blood donors, donations for flooded regions etc., will be quicker through our application by posting a query and getting the response (which includes the contact details).

We came up with this idea to reduce the complexities of NGOs, by contacting some of the NGOs directly and collecting their problems that they are facing for communication with people. In collection of those feedback, we designed this application so that it will be highly beneficial for their needs. This application is not only based on NGO's perspective, views of normal users are also considered.

This application has real time impact on society by direct usage of app instead of referring to many websites which could increase the number of users as fast as possible. In future, we would include conversation feature that would make communication easier and reliable.

2. SYSTEM STUDY

2.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ **ECONOMICAL FEASIBILITY**
- ◆ **TECHNICAL FEASIBILITY**
- ◆ **SOCIAL FEASIBILITY**

2.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.2 SOFTWARE REQUIREMENTS

Android is a Linux-based operating system for mobile devices such as smartphones and tablet computers. It is developed by the Open Handset Alliance led by Google. Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. Developers write primarily in a customized version of Java.

2.2.1 ANDROID ARCHITECTURE

Android consists of a kernel based on the Linux kernel, with middleware, libraries and APIs written in C and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Android uses the Dalvik virtual machine with just-in-time compilation to run Dalvik dex-code (Dalvik Executable), which is usually translated from Java bytecode^[6].

The main hardware platform for Android is the ARM architecture. There is support for x86 from the Android x86 project, and Google TV uses a special x86 version of Android^[6].

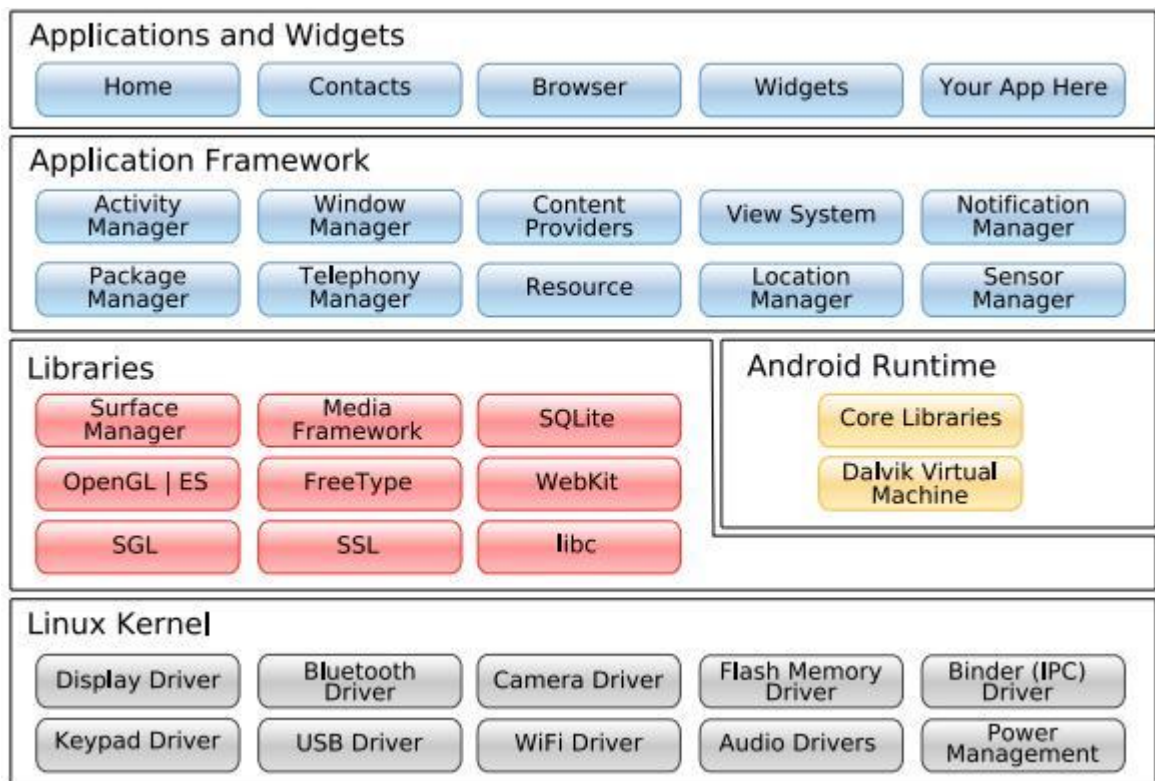


Figure 2.2.1 ARM Architecture^[6]

As shown in figure 2.2.1, Android's kernel is based on the Linux kernel and has further architecture changes by Google outside the typical Linux kernel development cycle. Android does not have a native X Window System nor does it support the full set of standard GNU libraries, and this makes it difficult to port existing Linux applications or libraries to Android^[6].

Certain features that Google contributed back to the Linux kernel, notably a power management feature called wakelocks, were rejected by mainline kernel developers, partly because kernel maintainers felt that Google did not show any intent to maintain their own code. Even though Google announced in April 2010 that they would hire two employees to work with the Linux kernel community, Greg Kroah-Hartman, the current Linux kernel maintainer for the -stable branch, said in December 2010 that he was concerned that Google was no longer trying to get their code changes included in mainstream Linux. Some Google Android developers hinted that "the Android team was getting fed up with the process", because they were a small team and had more urgent work to do on Android^[6].

However, in September 2010, Linux kernel developer Rafael J. Wysocki added a patch that improved the mainline Linux wakeup events framework. He said that Android device drivers that use wakelocks can now be easily merged into mainline Linux, but that Android's opportunistic suspend features should not be included in the mainline kernel. In 2011 Linus Torvalds said that "eventually Android and Linux would come back to a common kernel, but it will probably not be for four to five years"^[6].

In December 2011, Greg Kroah-Hartman announced the start of the Android Mainlining Project, which aims to put some Android drivers, patches and features back into the Linux kernel, starting in Linux 3.3. further integration being expected for Linux Kernel 3.4^[6].

2.2.2 APPLICATION FRAMEWORK

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more^[6].

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user^[6].

Underlying all applications is a set of services and systems, including:

- A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser.
- Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data.
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files.
- A Notification Manager that enables all applications to display custom alerts in the status bar^[6].

Libraries:

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

- **System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- **Media Libraries** - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL** - the underlying 2D graphics engine
- **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- **FreeType** - bitmap and vector font rendering
- **SQLite** - a powerful and lightweight relational database engine available to all applications^[6]

2.2.3 ANDROID RUNTIME

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management^[6].

2.2.4 LINUX KERNEL

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

2.2.5 ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform and FIREBASE, making it easy to integrate Google Cloud Messaging and App Engine^[7].

PROJECT STRUCTURE

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules

- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

- **manifests:** Contains the AndroidManifest.xml file.
- **java:** Contains the Java source code files, including JUnit test code.
- **res:** Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**).

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file^[7].

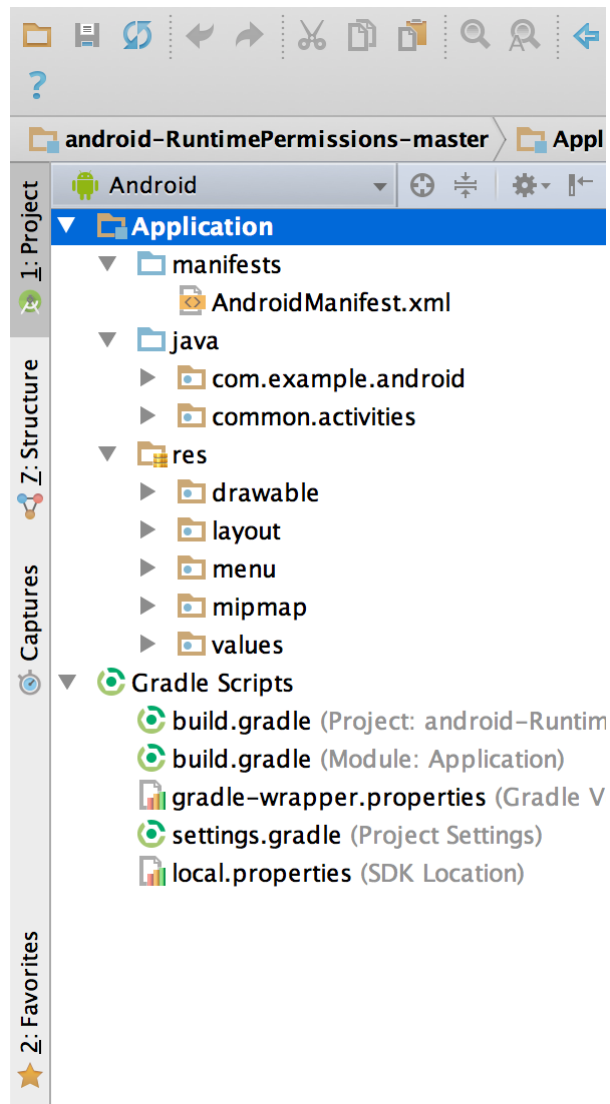


Figure 2.2.2. The project files in Android view^[7].

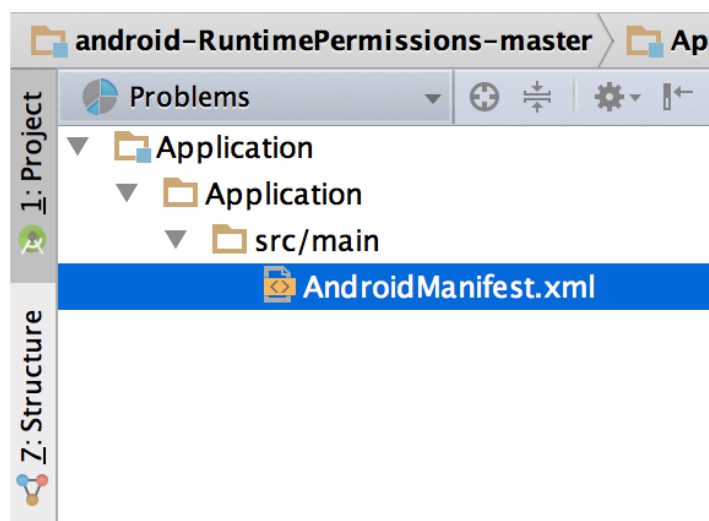


Figure 2.2.3. The project files in Problems view, showing a layout file with a problem^[7].

USER INTERFACE

The Android Studio main window is made up of several logical areas identified in figure 2.2.4.

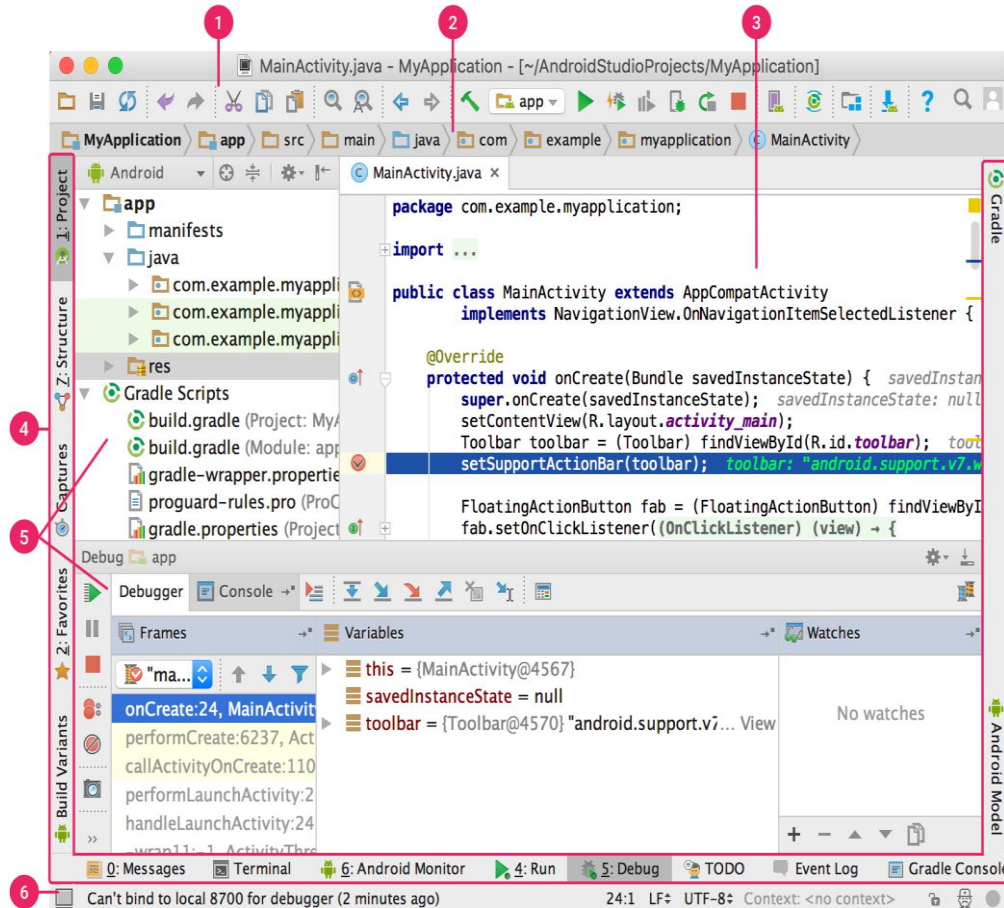


Figure 2.2.4. The Android Studio main window^[7].

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages^[7].

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger^[7].

2.2.6 FIREBASE

Firestore is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014^[10].

Firestore evolved from Envolv, a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolv provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that wasn't chat messages. Developers were using Envolv to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firestore as a separate company in April 2012^[10].

Firestore Inc. raised seed funding in May 2012. The company further raised Series A funding in June 2013. In October 2014, Firestore was acquired by Google. In October 2015, Google acquired Divshot to merge it with the Firestore team. Since the acquisition, Firestore has grown inside Google and expanded their services to become a unified platform for mobile developers. Firestore now integrates with various other Google services to offer broader products and scale for developers. In January 2017,

Google acquired Fabric and Crashlytics from Twitter to join those services to the Firebase team^[10].

SERVICES^[10]

Analytics

Firestore Analytics:

Firestore Analytics is a free app measurement solution that provides insight into app usage and user engagement.

Develop

Firestore Cloud Messaging:

Formerly known as Google Cloud Messaging (GCM), Firestore Cloud Messaging (FCM) is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which currently can be used at no cost.

Firestore Auth:

Firestore Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google. Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firestore.

Firestore Database:

Firestore provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firestore's cloud. The company provides client libraries integrates with Android, iOS, JavaScript, Java, ObjectiveC, swift and Node.js applications. The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the realtime database can secure their data by using the company's server-side-enforced security rules.

Firestore Storage:

Firestore Storage provides secure file uploads and downloads for Firestore apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firestore Storage is backed by Google Cloud Storage.

Firestore Hosting:

Firestore Hosting is a static and dynamic web hosting service that launched on May 13, 2014. It supports hosting static files such as CSS, HTML, JavaScript and other files, as well as dynamic Node.js support through Cloud Functions. The service delivers files over a content delivery network (CDN) through HTTP Secure (HTTPS) and Secure Sockets Layer encryption (SSL). Firestore partners with Fastly, a CDN, to provide the CDN backing Firestore Hosting. The company states that Firestore Hosting grew out of customer requests; developers were using Firestore for its real-time database but needed a place to host their content.

Firestore Test Lab for Android:

Firestore Test Lab for Android provides cloud-based infrastructure for testing Android apps. With one operation, developers can initiate testing of their apps across a wide variety of devices and device configurations. Test results—including logs, videos, and screenshots—are made available in the project in the Firestore console. Even if a developer hasn't written any test code for their app, Test Lab can exercise the app automatically, looking for crashes.

Firestore Crash Reporting:

Crash Reporting creates detailed reports of the errors in the app. Errors are grouped into clusters of similar stack traces and triaged by the severity of impact on app users. In addition to automatic reports, developer can log custom events to help capture the steps leading up to a crash.

3. EXISTING SYSTEM AND PROPOSED SYSTEM

3.1 EXISTING SYSTEM

The NGO applications and websites which are currently using are containing with the details of the specific such address and contact number. There is no specific application that connects all NGOs and common users together to communicate each other and solve particular problems or fulfill the requirements. These applications are not building any bridge among people to interact. Some of these apps and websites we have researched about are NGO Darpan, LetsConnect, Akshayapatra.org etc.

DISADVANTAGES

- Less interaction with people.
- Inefficient in functionality.
- Takes more time for communication.
- Updation of details is improper.

3.2 PROPOSED SYSTEM

Our proposed system avoids the above existing systems complication, by developing an application which includes most of the NGO services. Based on the request provided by the user, the respective NGO will respond with a specific answer within a short period of time. Any registered user (either NGO or normal user) can login into our application and use the services that are included in the application such as adding posts/requirements, viewing and reacting to other posts and receiving the reactions and the contact details of reacted member for your posts. You can also delete your posts immediately after your requirement has been accomplished.

ADVANTAGES

- User friendly.
- More interactive and functional.
- Takes less time for communication.

4. SYSTEM ANALYSIS

4.1 DOWNLOAD ANDROID STUDIO^[7]

Google provides Android Studio for the Windows, Mac OS X, and Linux platforms. You can download this software from the Android Studio homepage. (You'll also find the traditional SDKs, with Android Studio's command-line tools, available from the Downloads page.) Before downloading Android Studio, make sure your platform meets one of the following requirements:

Windows OS

- Microsoft Windows 7/8/10 (32-bit or 64-bit)
- 2 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution
- JDK 8
- For accelerated emulator: 64-bit operating system and Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality

Mac OS

- Mac OS X 10.8.5 or higher, up to 10.11.4 (El Capitan)
- 2 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution
- JDK 6

Linux OS

- **GNOME or KDE desktop:** Tested on Ubuntu 12.04, Precise Pangolin (64-bit distribution capable of running 32-bit applications)

- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.11 or later
- 2 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution
- JDK 8
- For accelerated emulator: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality, or AMD processor with support for AMD Virtualization (AMD-V)

Once you've ensured your operating system is compatible with Android Studio 2.1.1, download the appropriate Android Studio distribution file. The Android Studio download page auto-detected that I'm running 64-bit Windows 8.1 and selected android-studio-bundle-143.2821654-windows.exe for me to download.

Bundled Installer and Android SDK

android-studio-bundle-143.2821654-windows.exe includes an installer and the Android SDK. Alternatively, I could have downloaded a distribution file without the installer and without the SDK.

4.2 INSTALLING ANDROID STUDIO ON 64-BIT WINDOWS 8.1^[7]

We launched android-studio-bundle-143.2821654-windows.exe to start the installation process. The installer responded by presenting the Android Studio Setup dialog box shown in Figure 4.2.1.



Figure 4.2.1. Set up Android Studio^[7]

Clicking Next took me to the following dialog box, which gives you the option to decline installing the Android SDK (included with the installer) and an Android Virtual Device (AVD).

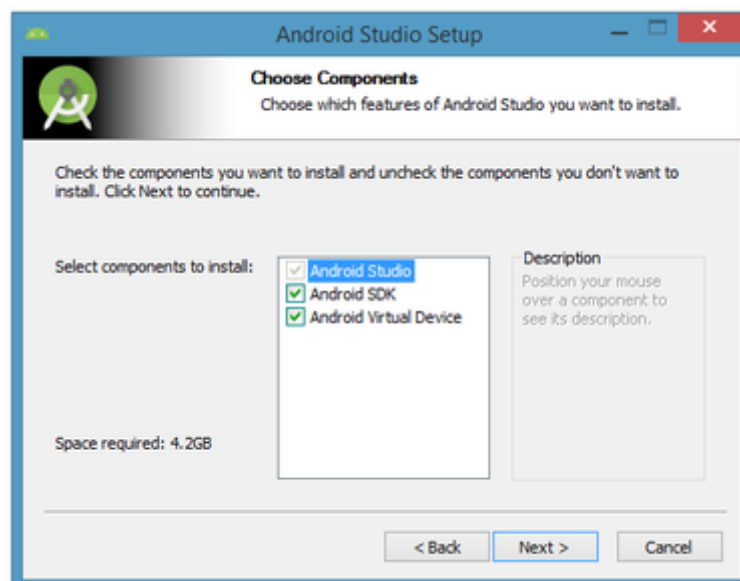


Figure 4.2.2. Do you want to install the Android SDK and AVD?^[7]

We chose to keep the default settings. After clicking Next, you'll be taken to the license agreement dialog box. Accept the license to continue the installation.

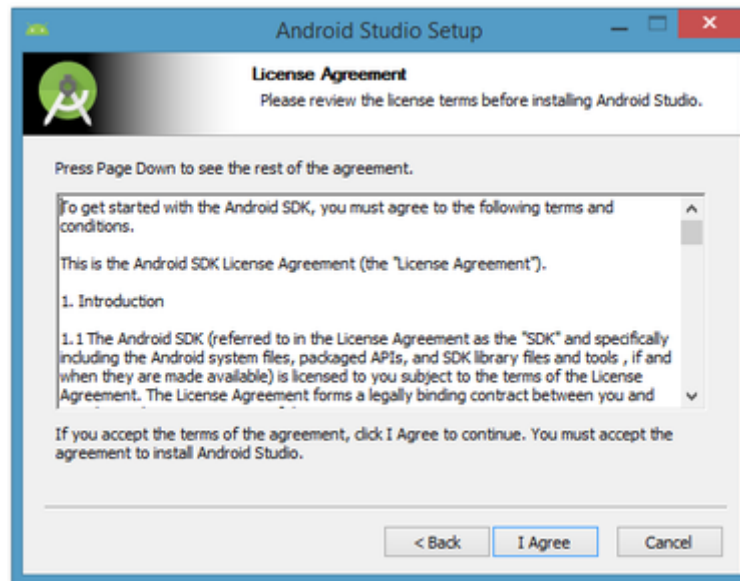


Figure 4.2.3. Accept the license agreement to continue installation^[7]

The next dialog box invites you to change the installation locations for Android Studio and the Android SDK.

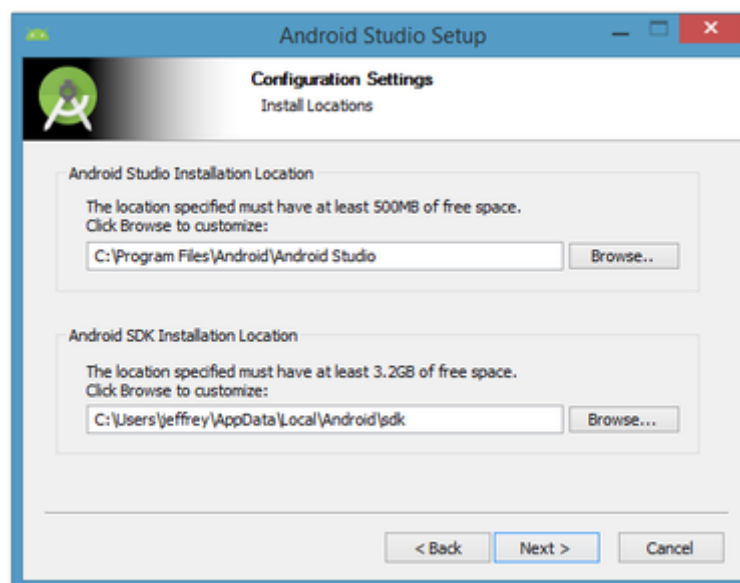


Figure 4.2.4. Set the Android Studio and Android SDK installation locations^[7]

Change the location or accept the default locations and click Next.

The installer defaults to creating a shortcut for launching this program, or you can choose to decline. I recommend that you create the shortcut, then click the Install button to begin installation.

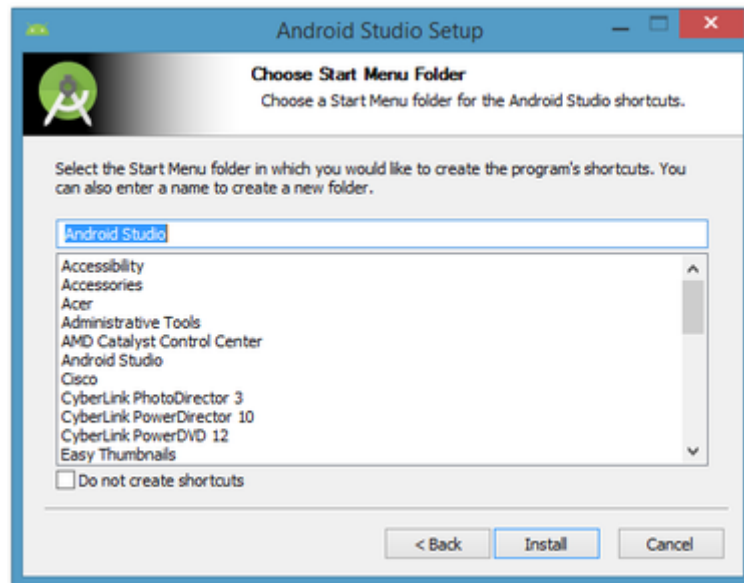


Figure 4.2.5. Create a new shortcut for Android Studio^[7]

The resulting dialog box shows the progress of installing Android Studio and the Android SDK. Clicking the Show Details button will let you view detailed information about the installation progress.

The dialog box will inform you when installation has finished. When you click Next, you should see the following:



Figure 4.2.6. Leave the Start Android Studio check box checked to run this software^[7]
To complete your installation, leave the Start Android Studio box checked and click Finish.

4.3 RUNNING ANDROID STUDIO^[7]

Android Studio presents a splash screen when it starts running:



Figure 4.3.1. Android Studio's start screen^[7]

On your first run, you'll be asked to respond to several configuration-oriented dialog boxes. The first dialog box focuses on importing settings from any previously installed version of Android Studio.

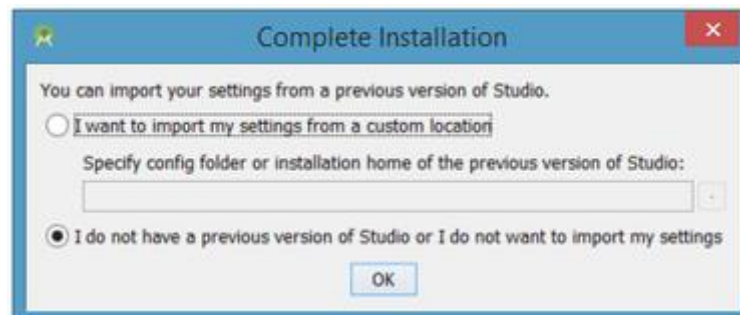


Figure 4.3.2. Import settings^[7]

If you're like me, and don't have a previously installed version, you can just keep the default setting and click OK. Android Studio will respond with a slightly enhanced version of the splash screen, followed by the Android Studio Setup Wizard dialog box:

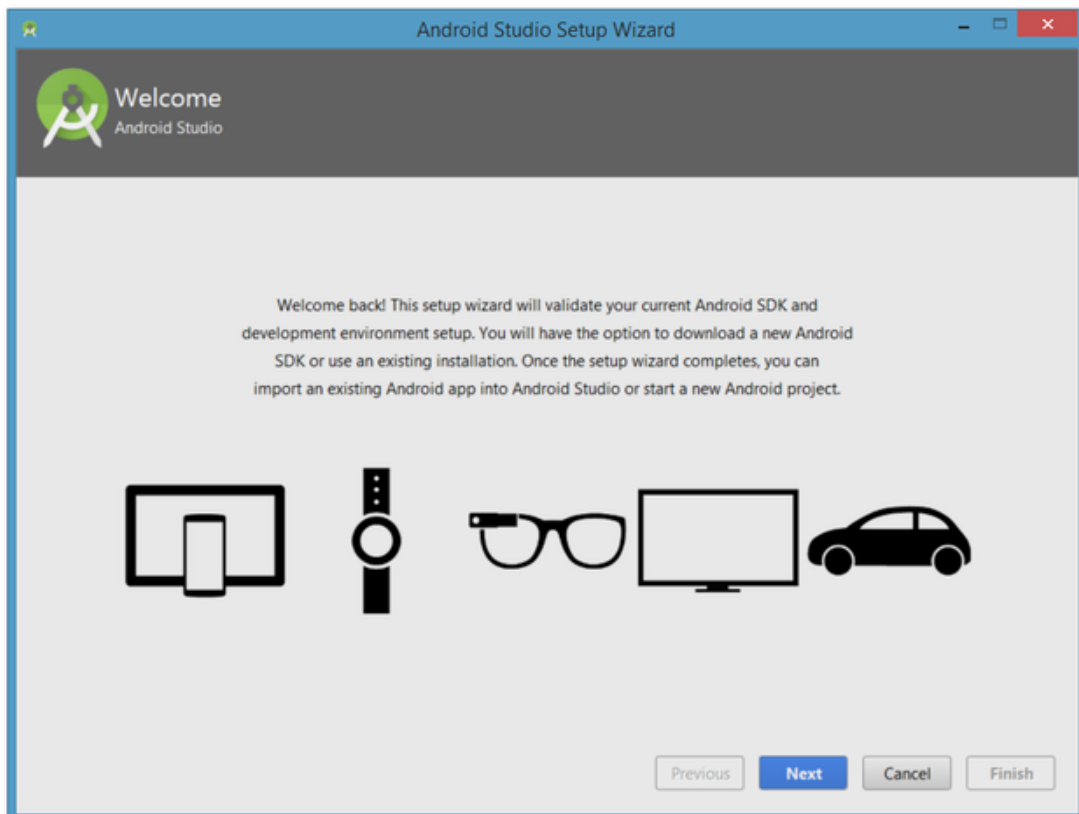


Figure 4.3.3. Validate your Android SDK and development environment setup^[7]

When you click Next, the setup wizard invites you to select an installation type for your SDK components. For now, I recommend you keep the default standard setting.

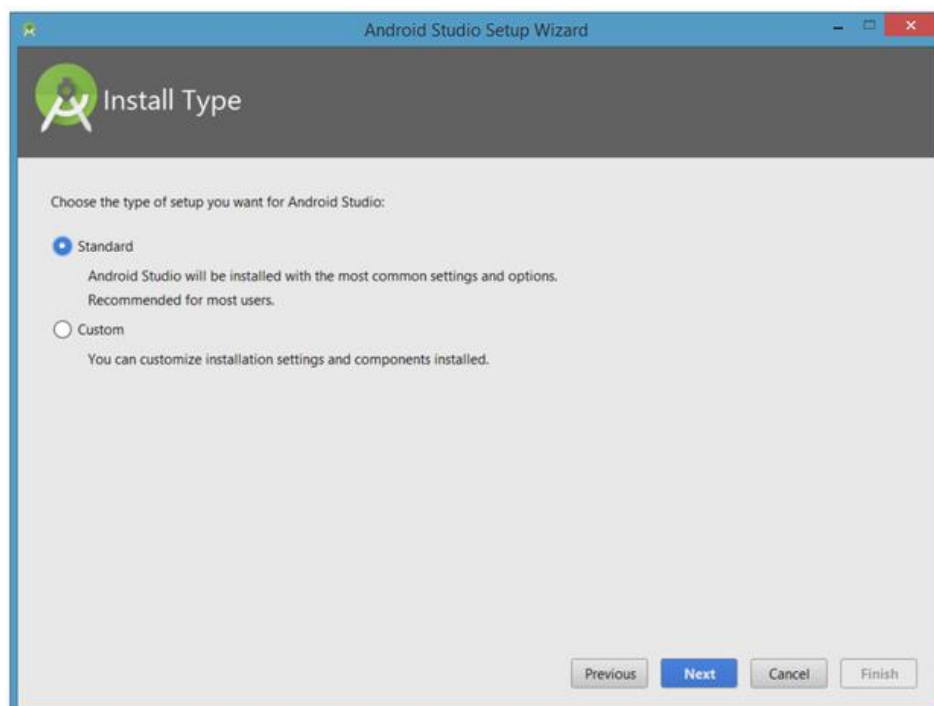


Figure 4.3.4. Choose an installation type^[7]

Click Next and verify your settings, then click Finish to continue.

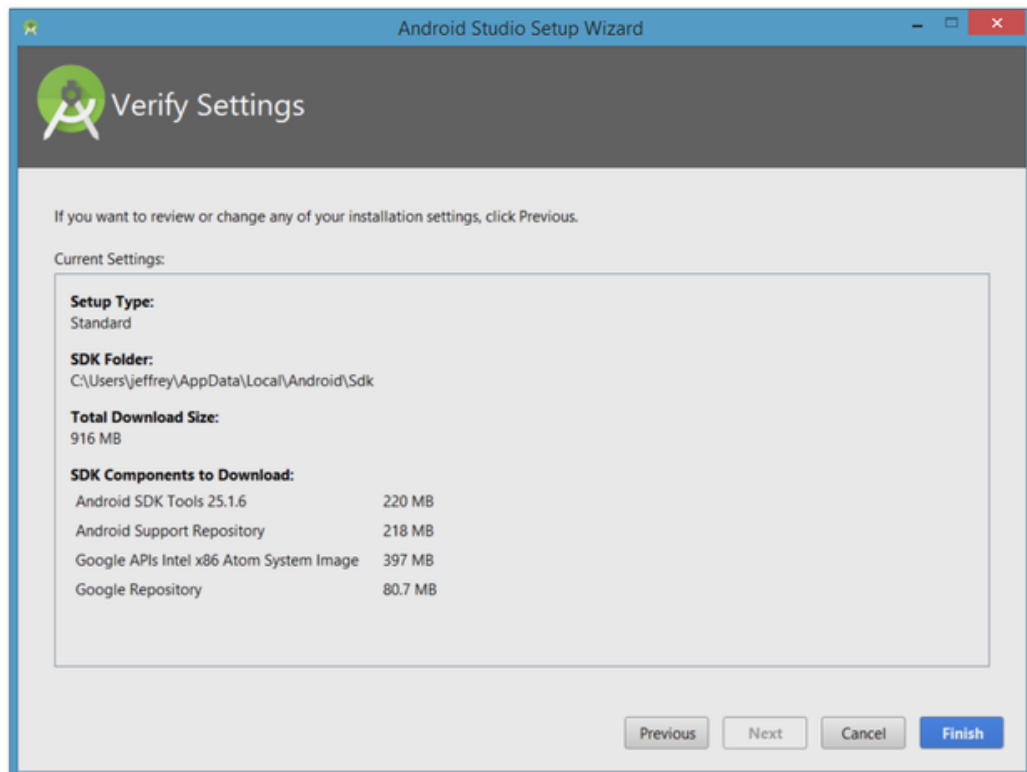


Figure 4.3.5. Review settings^[7]

The wizard will download and unzip various components. Click Show Details if you want to see more information about the archives being downloaded and their contents.

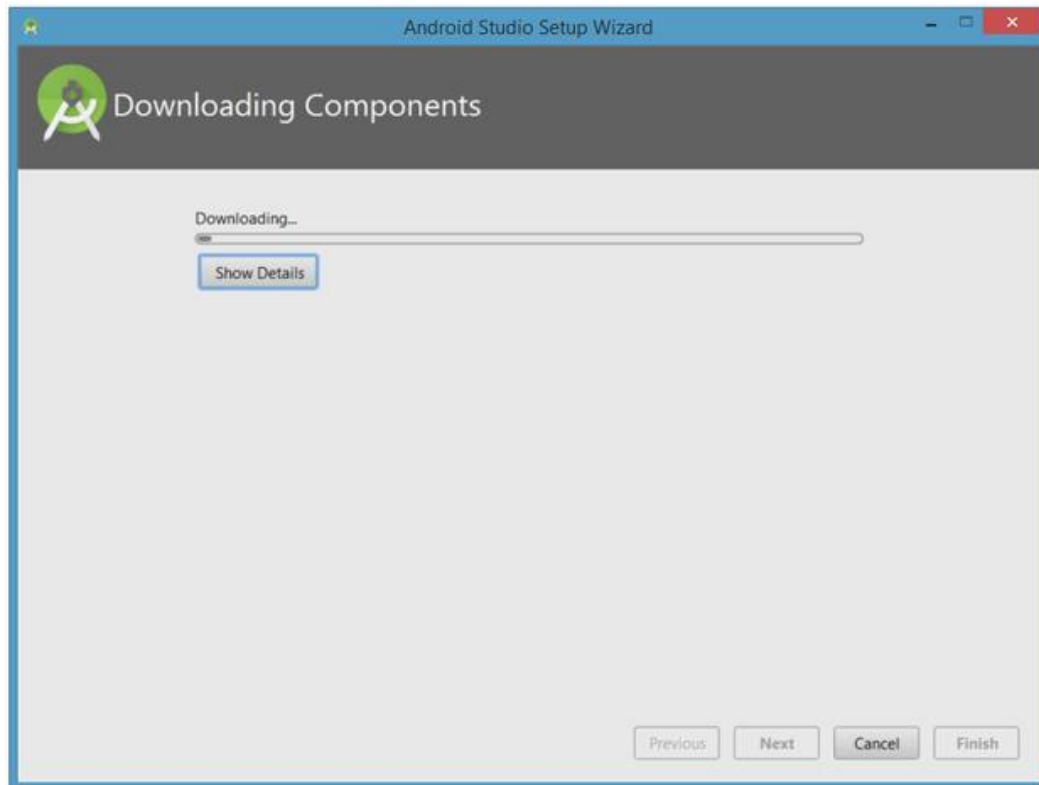


Figure 4.3.6. The wizard downloads and unzips Android Studio components^[7]
If your computer isn't Intel based, you might get an unpleasant surprise after the components have completely downloaded and unzipped:

```
g:\a\1\android-studio\android-studio
m2repository/com/google/firebase/firebase-core/9.0
.1/firebase-core-9.0.1.pom.md5
Android SDK is up to date.
Unable to install Intel HAXM
Your CPU does not support required features (VT-x or SVM).
Unfortunately, your computer does not support hardware
accelerated virtualization.
Here are some of your options:
1) Use a physical device for testing
2) Develop on a Windows/OSX computer with an Intel processor
that supports VT-x and NX
3) Develop on a Linux computer that supports VT-x or SVM
4) Use an Android Virtual Device based on an ARM system image
(This is 10x slower than hardware accelerated
virtualization)

Creating Android virtual device
Unable to create a virtual device: Unable to create Android
virtual device
```

Figure 4.3.7. Intel-based hardware acceleration is unavailable^[7]
Your options are to either put up with the slow emulator or use an Android device to speed up development. I'll discuss the latter option later in the tutorial.

Finally, click Finish to complete the wizard. You should see the Welcome to Android Studio dialog box:



Figure 4.3.8. Welcome to Android Studio^[7]

You'll use this dialog to start up a new Android Studio project, work with an existing project, and more. You can access it anytime by double-clicking the Android Studio shortcut on your desktop.

Your first Android Studio mobile app

The quickest way to get to know Android Studio is to use it to develop an app. We'll start with a variation on the "Hello, World" application: a little mobile app that displays a "Welcome to Android" message.

4.4 STARTING A NEW PROJECT^[7]

From our setup so far, you should still have Android Studio running with the Welcome to Android Studio dialog box. From here, click Start a new Android Studio project. Android Studio will respond with the Create New Project dialog box shown in Figure 4.4.1.

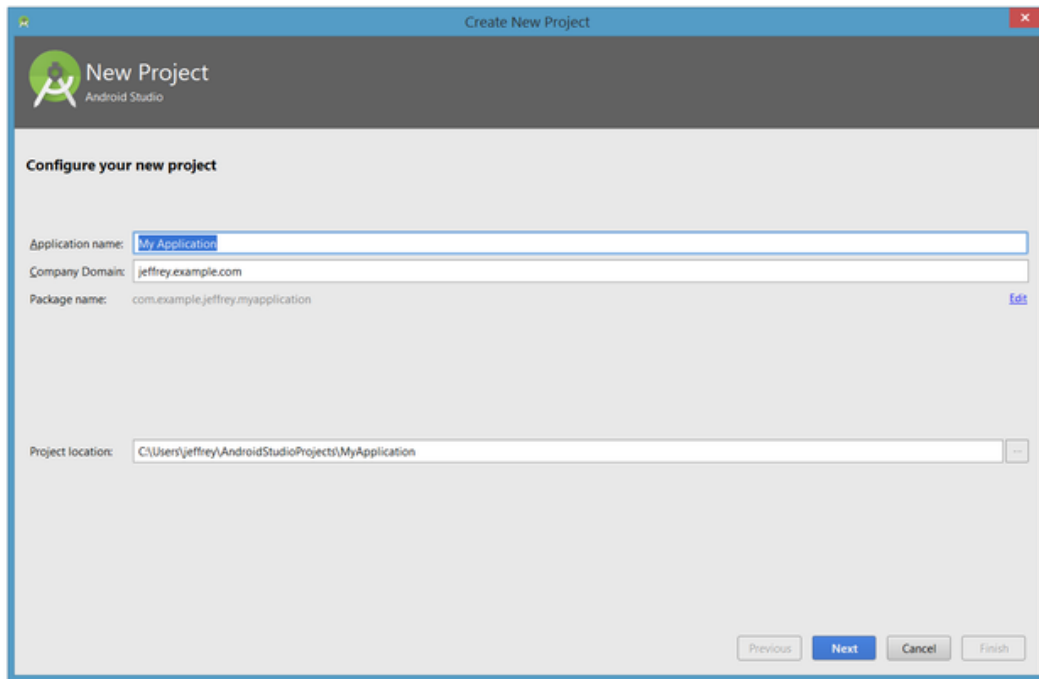


Figure 4.4.1. Create a new project^[7]

Enter W2A (Welcome to Android) as the application name and *javajeff.ca* as the company domain name. You should then see C:\Users\jeffrey\AndroidStudioProjects\W2A as the project location. Click Next to select your target devices.

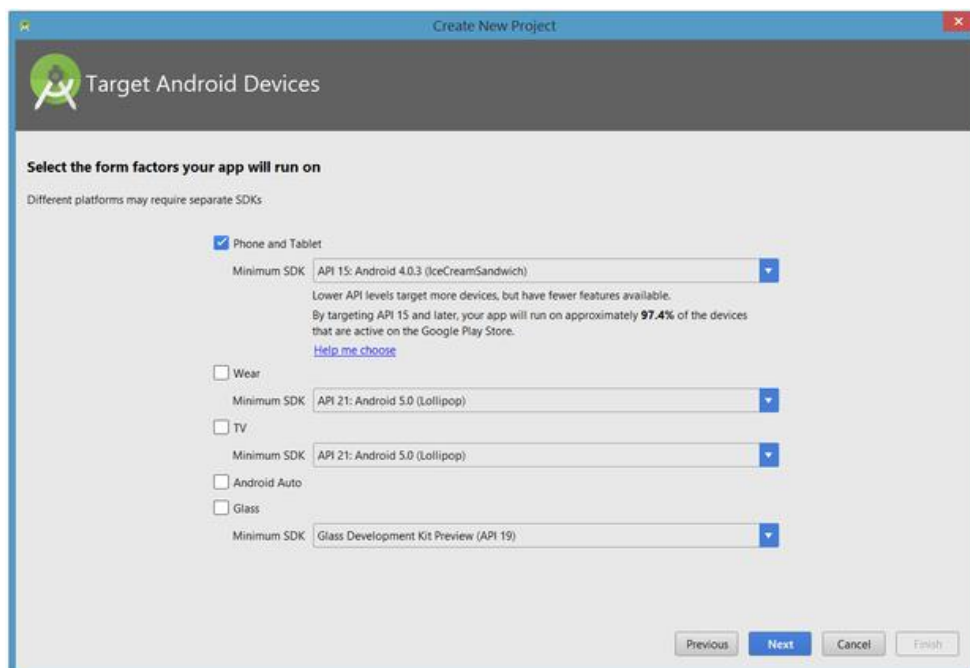


Figure 4.4.2. Select your target device categories^[7]

Android Studio lets you select *form factors*, or categories of target devices, for every app you create. We would have preferred to keep the default API 15: Android 4.0.3 (IceCreamSandwich) minimum SDK setting (under Phone and Tablet), which is supported by my Amazon Kindle Fire HD tablet. Because Android Studio doesn't currently support this API level (even when you add the 4.0.3 system image via the SDK Manager), I changed this setting to API 14: Android 4.0 (IceCreamSandwich), which is also supported by my tablet.

Click Next, and you will be given the opportunity to choose a template for your app's main activity. For now, we'll stick with Empty Activity. Select this template and click Next.

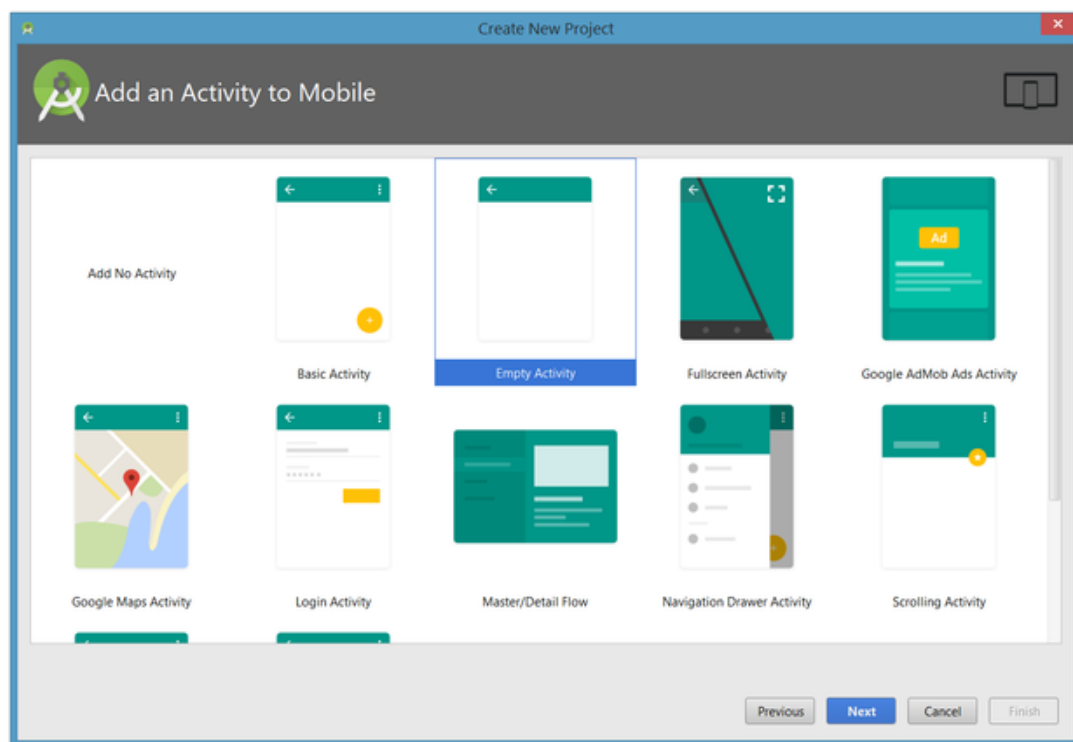


Figure 4.4.3. Specify an activity template^[7]

Next, you'll customize the activity:

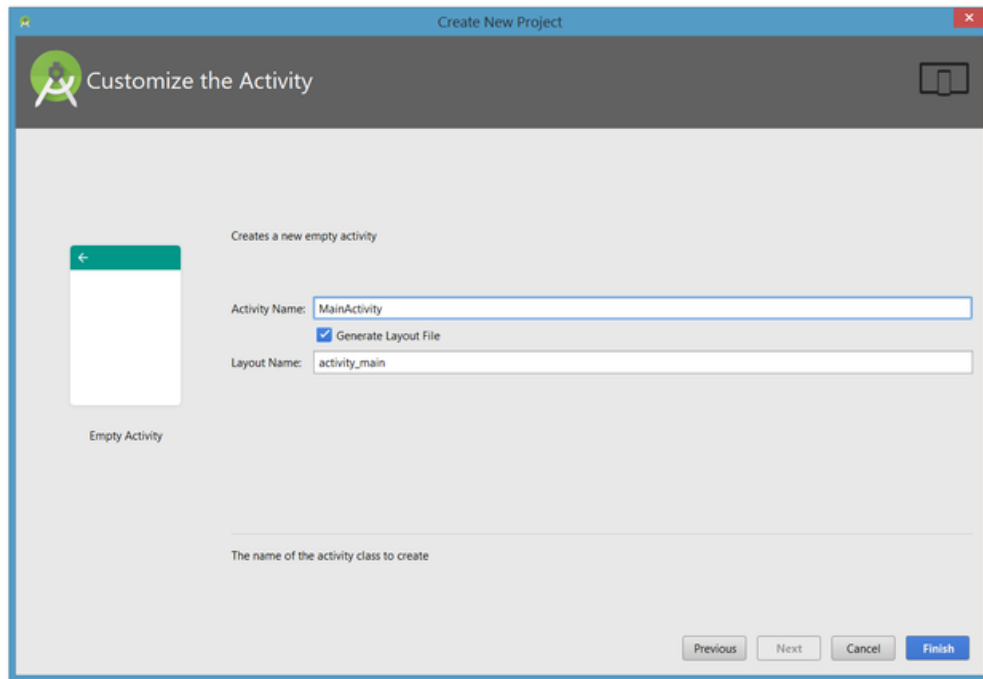


Figure 4.4.4. Customize your activity^[7]

Enter *W2A* as the activity name and *main* as the layout name, and click Finish to complete this step. Android Studio will respond that it is creating the project, then take you to the project workspace.

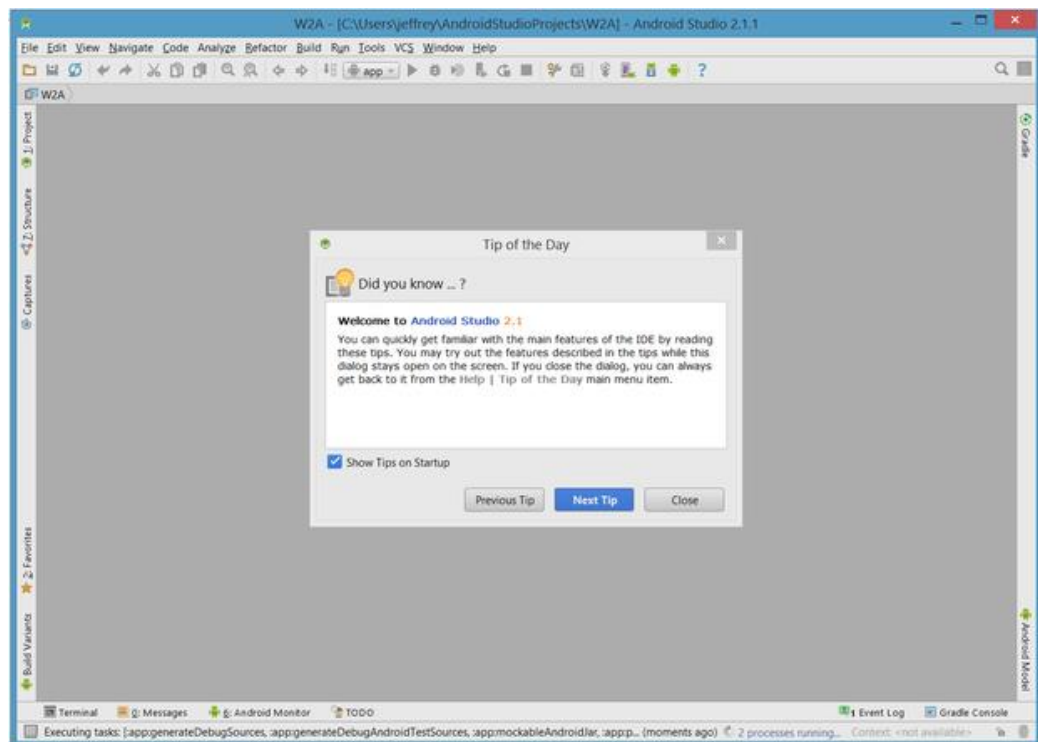


Figure 4.4.5. Android Studio workspace^[7]

The project workspace is organized around a menu bar, a tool bar, a work area, additional components that lead to more windows (such as a Gradle Console window), and a status bar. Also note the Tip of the Day dialog box, which you can disable if you like.

Accessing AVD and SDK managers from menu and tool bars

To access the traditional AVD Manager or SDK Manager, select Android from the Tools menu followed by AVD Manager or SDK Manager from the resulting pop-up menu (or click their tool bar icons).

4.5 THE PROJECT AND EDITOR WINDOWS^[7]

When you enter the project workspace, W2A is identified as the current project, but you won't immediately see the project details. After a few moments, these details will appear in two new windows.

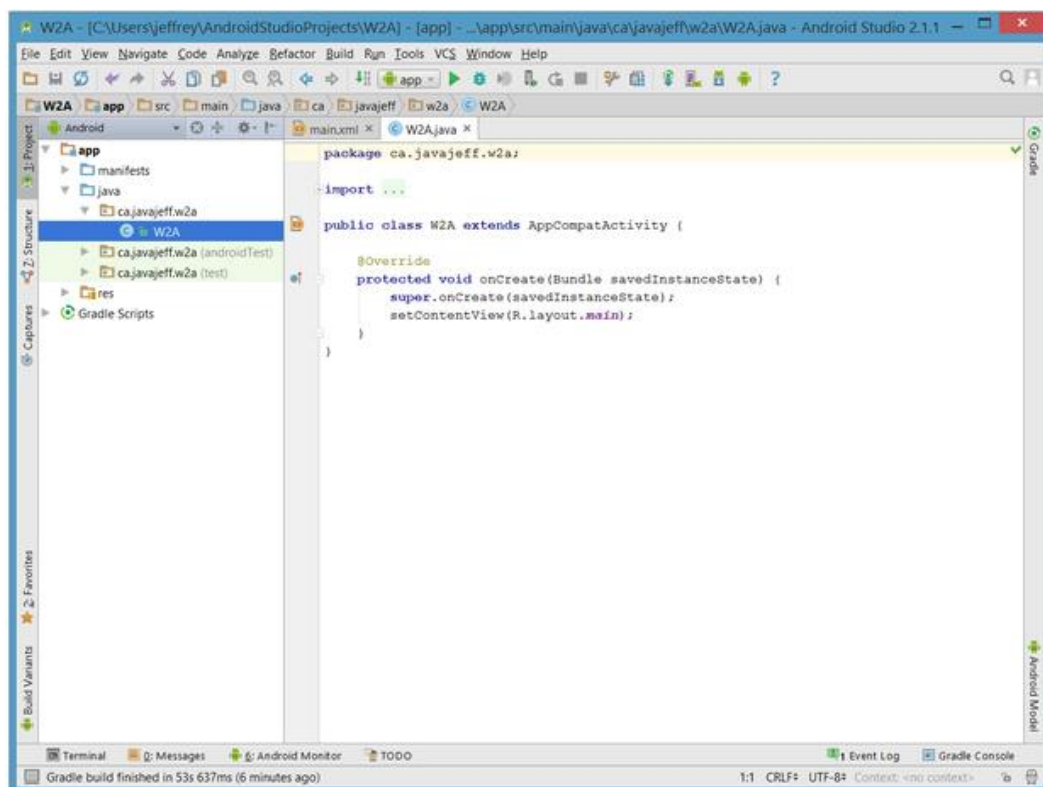


Figure 4.5.1. The project and editor windows^[7]

The project window is organized into a tree whose main branches are App and Gradle Scripts. The App branch is further organized into manifests, java, and res subbranches:

- manifests stores `AndroidManifest.xml`, which is an XML file that describes the structure of an Android app. This file also records permission settings (where applicable) and other details about the app.
- java stores an app's Java source files according to a package hierarchy, which is `ca.javajeff.w2a` in this example.
- res stores an app's resource files, which are organized into drawable, layout, mipmap, and values subbranches:
- drawable: an initially empty location in which to store an app's artwork
- layout: a location containing an app's layout files; initially, `main.xml` (the main activity's layout file) is stored here
- mipmap: a location containing various `ic_launcher.png` files that store launcher screen icons of different resolutions
- values: a location containing `colors.xml`, `dimens.xml`, `strings.xml`, and `styles.xml`

The Gradle Scripts branch identifies various `.gradle` (such as `build.gradle`) and `.properties` (such as `local.properties`) files that are used by the Gradle-based build system.

5. SOFTWARE DESIGN (UML DIAGRAMS)

5.1 USE CASE DIAGRAM

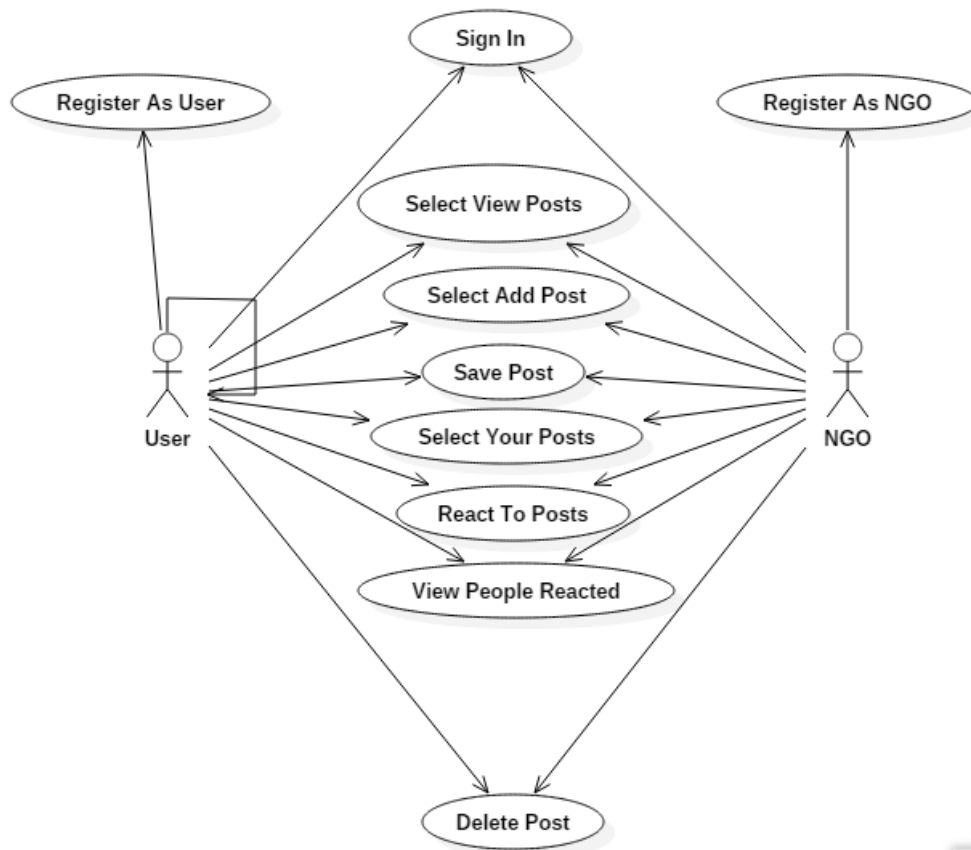


Fig.5.1 Use Case Diagram

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system(s) should or can perform in collaboration with one or more external users of the system (actors).

5.2 SEQUENCE DIAGRAM

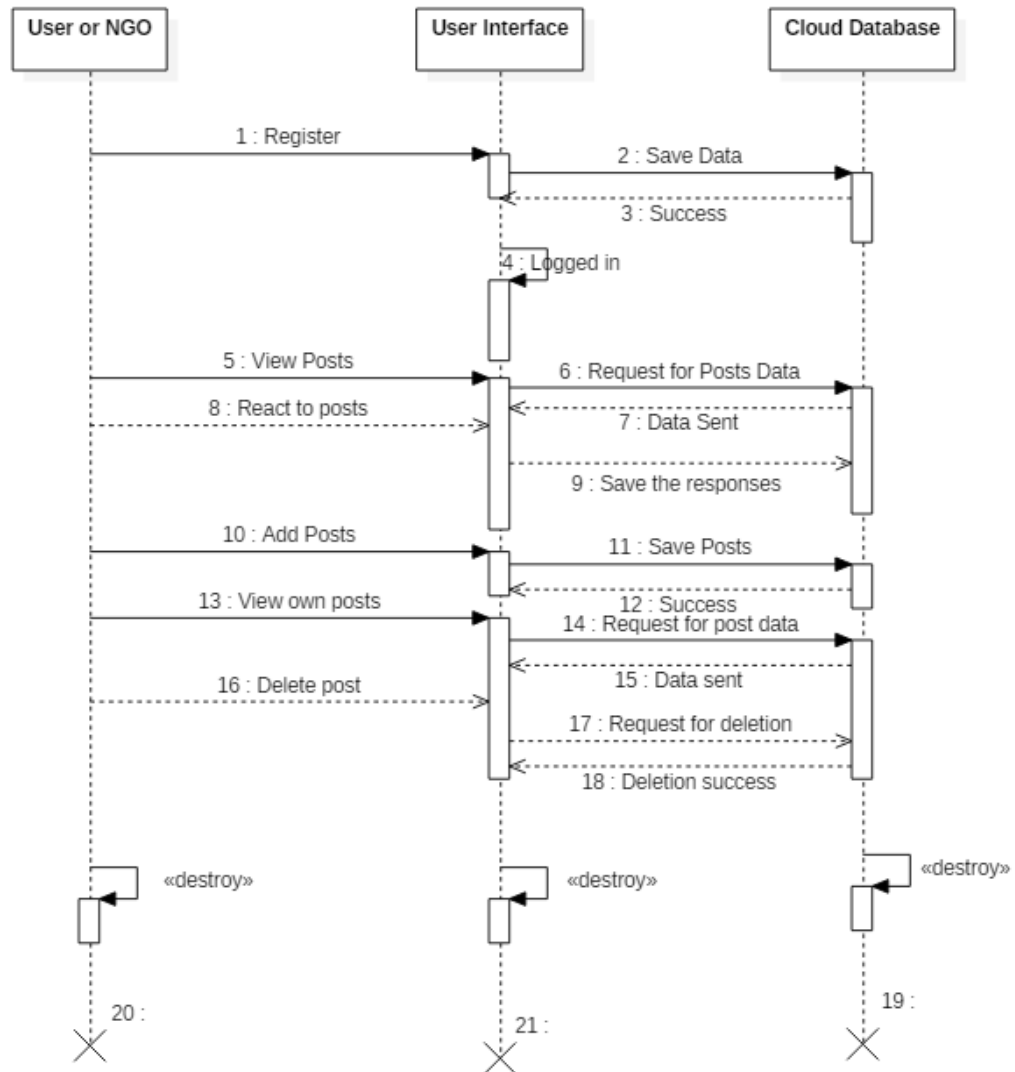


Fig 5.2 Sequence Diagram

Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. It shows objects arranged in time sequence.

5.3 CLASS DIAGRAM

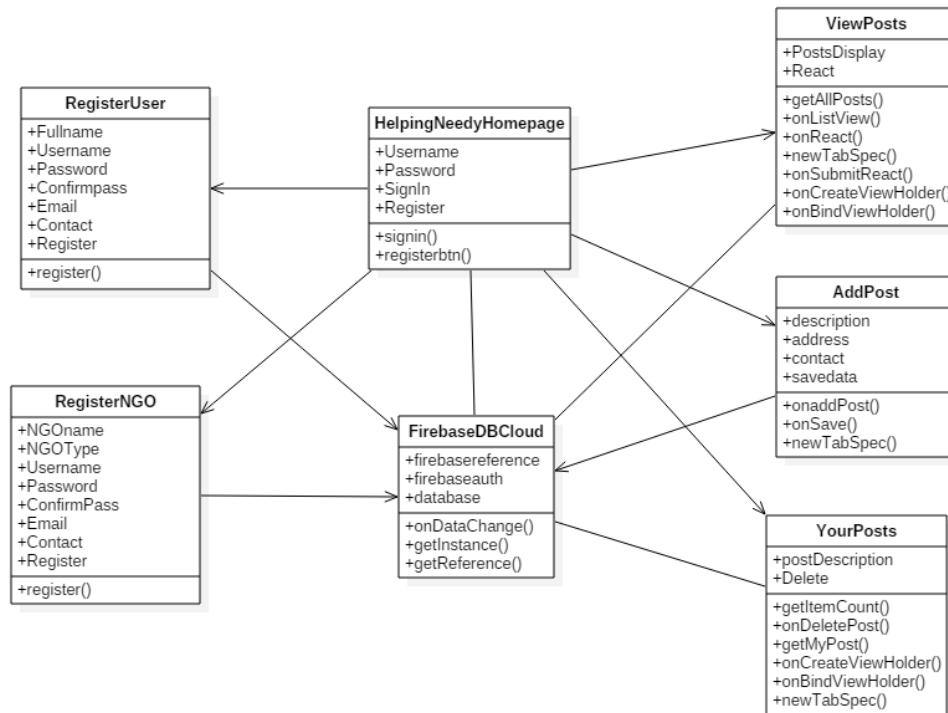


Fig 5.3 Class Diagram

Class diagram is a type of static structure that describes the system by showing the system's classes, their attributes, operations or methods and the relationships among objects.

5.4 ACTIVITY DIAGRAM

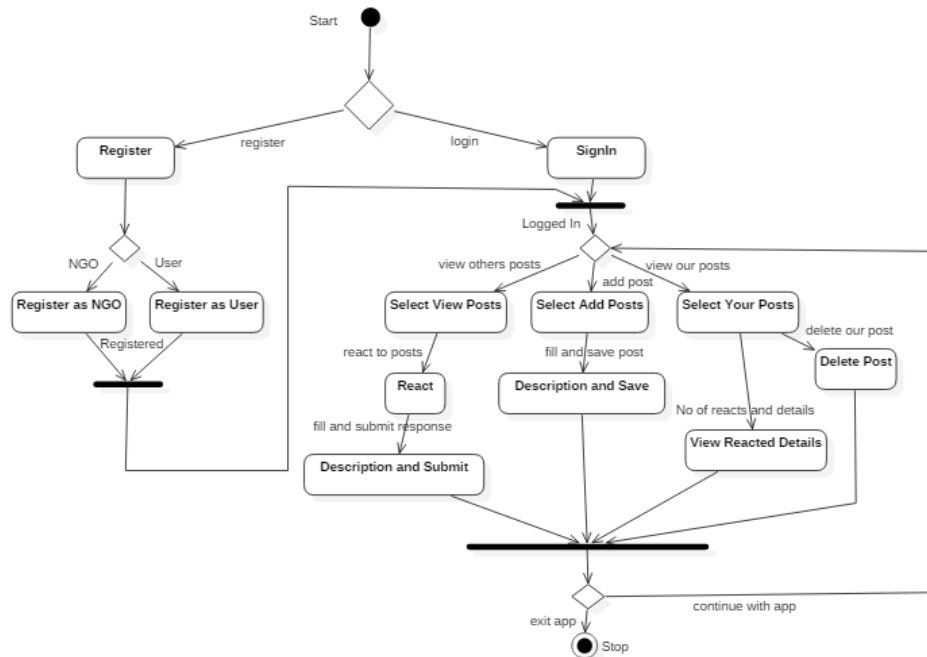


Fig 5.4 Activity Diagram

Activity diagram is graphical representation of workflow of stepwise activities and actions with support for choice, iteration and concurrency.

5.5 COMPONENT DIAGRAM

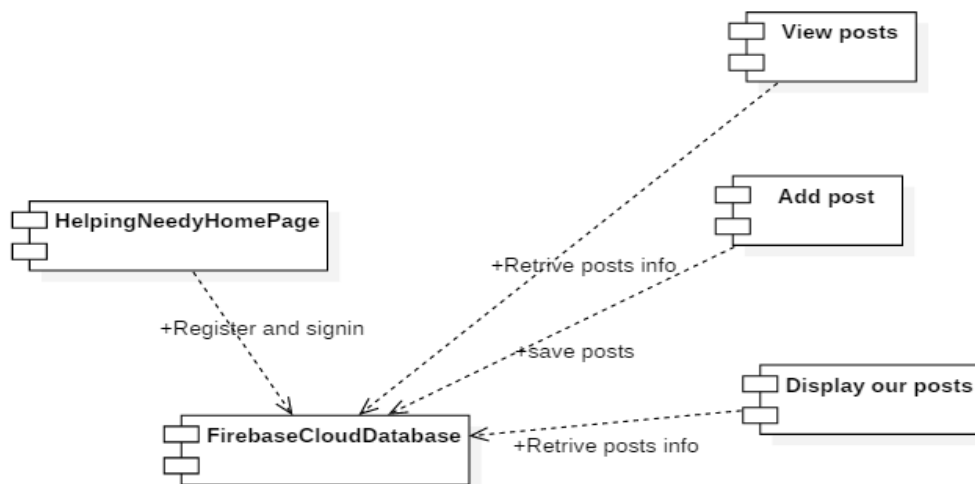


Fig 5.5 Component Diagram

Component diagram describes the components used for functionality of the system.

5.6 DEPLOYMENT DIAGRAM

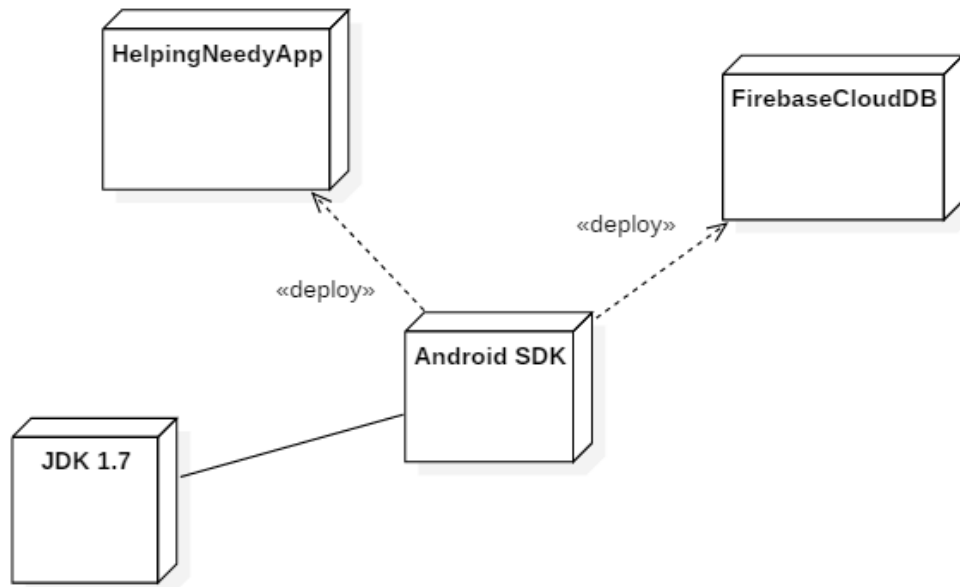


Fig 5.6 Deployment Diagram

Deployment diagram is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets.

6. IMPLEMENTATION

The interface of this application is very clean and easy to use. It easily gets to the point without many taps and clicks throughout the app. To build the app we used Android Studio as our tool, in which, we designed interfaces using XML coding and to run those UI elements Java classes has been implemented. To store the users' data such as user information (Username, Password, EmailID etc), posts information etc., the application has been integrated to Firebase Cloud Database using specific dependencies in App level and Project level gradle files. Additionally, Android Manifest files are used to register every activity that will be useful for Android Runtime Machine to execute application quickly without any ambiguity.

6.1 FORMS AND REPORTS (SAMPLE SCREENSHOTS)

Home Page

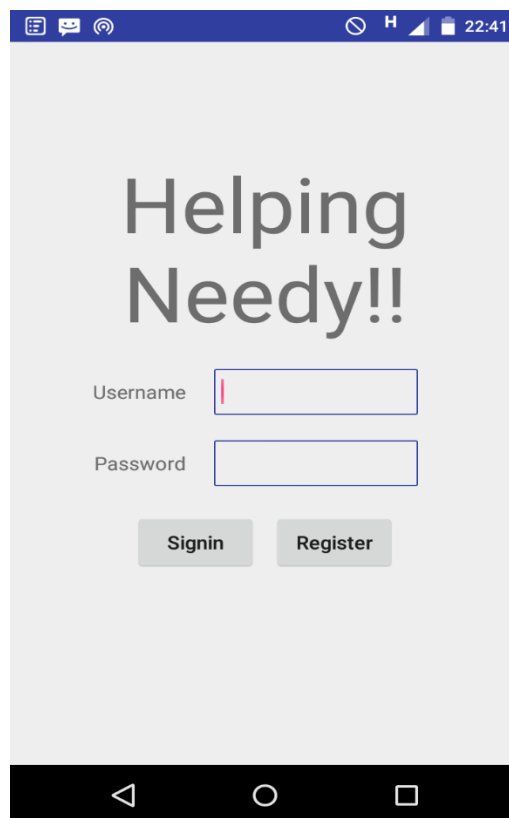
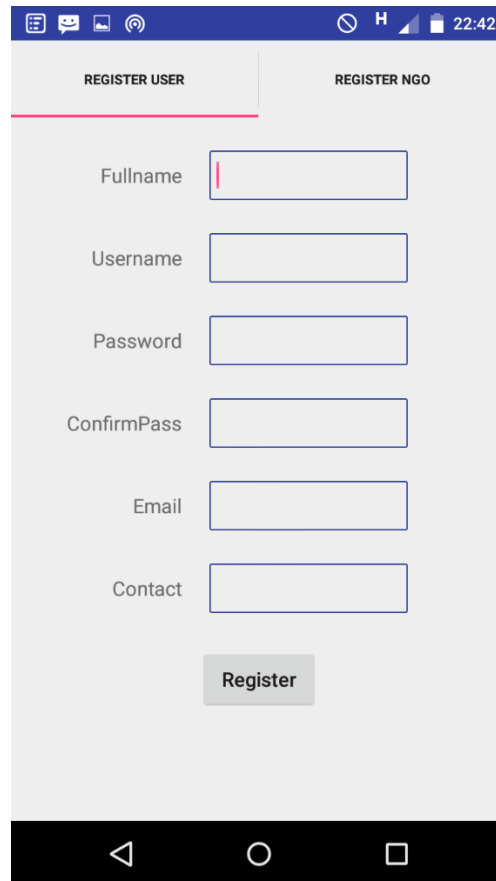


Fig. 6.1.1 Home page

Application home page when the it runs for the first time and user needs to choose an option between “Register” and “SignIn” click to proceed further.

Register User

If a **normal user** is new to the application, then register option can be choosen to proceed and fill the required fields for registration into application and can use the services of the application.



The screenshot displays a mobile application interface for user registration. At the top, there is a status bar with various icons and the time 22:42. Below this, the application has two tabs: 'REGISTER USER' and 'REGISTER NGO'. The 'REGISTER USER' tab is currently selected, indicated by a pink underline. The form contains six input fields: 'Fullname', 'Username', 'Password', 'ConfirmPass', 'Email', and 'Contact'. Each field has a corresponding label to its left. A 'Register' button is positioned at the bottom of the form. The entire form is set against a light gray background.

Fig. 6.1.2 Register User

Register NGO

If an **NGO** is new to the application, then register option can be choosen to proceed and fill the required fields for registration into application and can use the services of the application.

REGISTER USER REGISTER NGO

NGO Name

NGO Type

Username

Password

ConfirmPass

Email

Contact

Register

Fig. 6.1.3 Register NGO

Sign In

Helping Needy!!

Username

Password

Signin Register

Fig. 6.1.4 Sign In

If a user is already registered, then he/she can login into the application by giving valid credentials and click “SignIn” to proceed.

View Posts

After signing in, first page i.e., VIEW POSTS page opens up and user can view the posts that are added by other users. User can display the full description of the post by clicking on it and a pop up would come showing full details, can be closed on clicking “Ok”. User can also react to the posts by clicking on “React”.

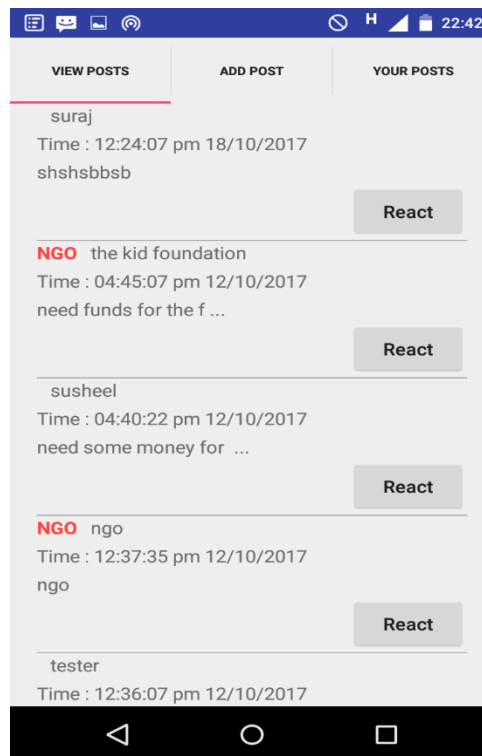


Fig. 6.1.5 View Posts

React to a Post

After clicking on React, user has to fill in the above fields and click on “Submit” to send the response to the particular post.

Fig. 6.1.6 React to a Post

Add Post

Fig. 6.1.7 Add post

Choosing ADD POST tab, we get the above page.

Adding Post / Requirement

User has to fill in the above fields to add any post and click on “Save” to post to be saved and displayed to others.

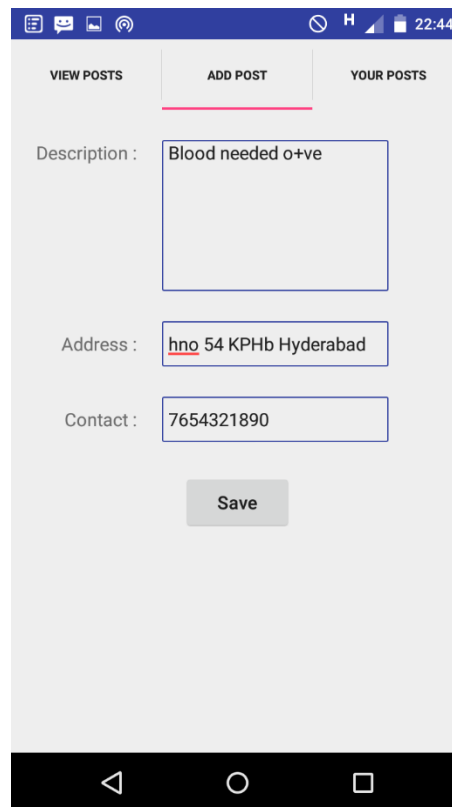


Fig. 6.1.8 Adding Post

Post added successfully

After clicking Save button, we get success message that it is posted successfully.

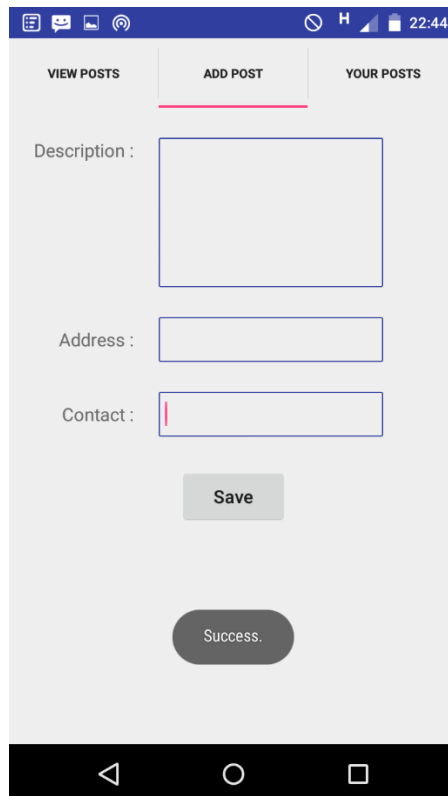


Fig. 6.1.9 Posted Added

Your Posts

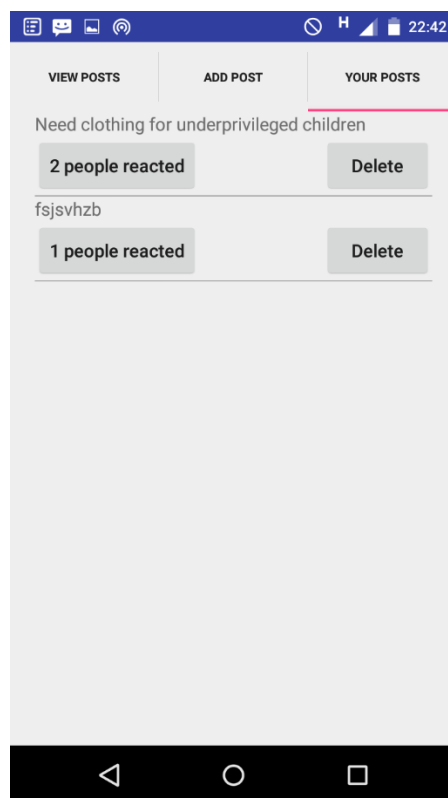


Fig. 6.1.10 Your posts

On clicking the tab YOUR POSTS, users can see their posts and reactions to their posts.

Number of people reacted to your post and their details

On clicking “ X people reacted “ (X is a number such as 0,1,2,...) , the details of the reacted user will be displayed and further we can contact for any extra details.

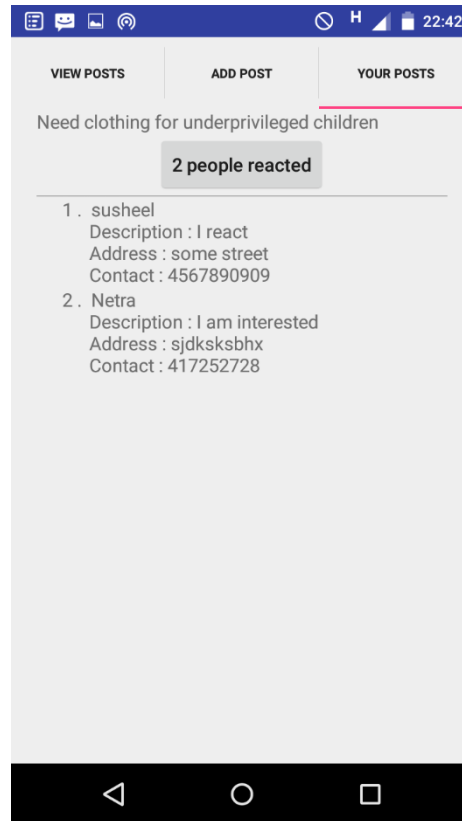


Fig. 6.1.11 No of reacts

Delete your post – post successfully deleted

User can delete his/her posts any time by clicking on “Delete” button.

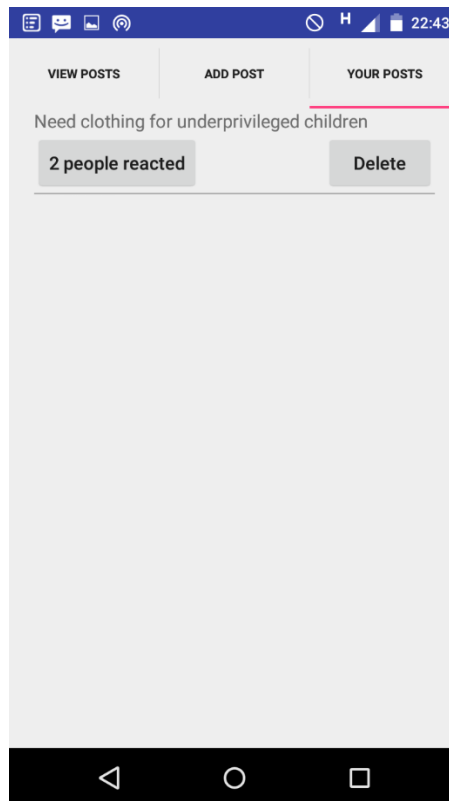


Fig. 6.1.12 Delete post

6.2 CODE IMPLEMENTATION

6.2.1 Common Java Files

FirstActivity.java:

```
package com.yang.education;

import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.app.FragmentManager;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.TabHost;
import android.widget.Toast;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.yang.education.Fregment.FirstTabListFragment;
import com.yang.education.Fregment.FirstTabSubmitFragment;
import com.yang.education.Fregment.ThirdTabMyPostFragment;
import com.yang.education.Fregment.ThirdTabViewReactFragment;
import com.yang.education.Model.PostModel;
import com.yang.education.Model.ReactModel;
```

```

public class FirstActivity extends AppCompatActivity {

    public static String authID;
    public static String postID;
    public static String postDescription;
    private FirebaseDatabase database = FirebaseDatabase.getInstance();
    private EditText etDescription;
    private EditText etAddress;
    private EditText etContact;
    private FrameLayout flTab1Frame;

    FirstTabListFragment firstTabListFragment;
    FirstTabSubmitFragment firstTabSubmitFragment;
    ThirdTabMyPostFragment thirdTabMyPostFragment;
    ThirdTabViewReactFragment thirdTabViewReactFragment;
    private android.app.FragmentManager mFirstFragmentManager;
    private android.app.FragmentManager mThirdFragmentManager;
    private FrameLayout flTab3Frame;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_first);
        TabHost host = (TabHost) findViewById(R.id.tabHost);
        host.setup();

        //Tab 1
        TabHost.TabSpec spec = host.newTabSpec("Tab One");
        spec.setContent(R.id.tab1);
        spec.setIndicator("View Posts");
        host.addTab(spec);

        //Tab 2
        spec = host.newTabSpec("Tab Two");
        spec.setContent(R.id.tab2);
        spec.setIndicator("Add Post");
        host.addTab(spec);

        //Tab 3
        spec = host.newTabSpec("Tab Three");
        spec.setContent(R.id.tab3);
        spec.setIndicator("Your Posts");
        host.addTab(spec);

        //tab2
        etDescription = (EditText) findViewById(R.id.etDescription);
        etAddress = (EditText) findViewById(R.id.etAddress);
        etContact = (EditText) findViewById(R.id.etContact);

        etDescription.setBackground(MyApplication.getRoundDrawable(FirstActivity.this,
            1, Color.TRANSPARENT, ContextCompat.getColor(this,
            R.color.colorPrimaryDark), 1));

        etAddress.setBackground(MyApplication.getRoundDrawable(FirstActivity.this,
            1, Color.TRANSPARENT, ContextCompat.getColor(this,
            R.color.colorPrimaryDark), 1));

        etContact.setBackground(MyApplication.getRoundDrawable(FirstActivity.this,
            1, Color.TRANSPARENT, ContextCompat.getColor(this,
            R.color.colorPrimaryDark), 1));

        Button btnAdd = (Button) findViewById(R.id.btnAdd);
        btnAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                onAddPost(etDescription.getText().toString(),
                    etAddress.getText().toString(), etContact.getText().toString());
            }
        });
    }
}

```



```

    });

    //tab1
    flTab1Frame = (FrameLayout) findViewById(R.id.flTab1Frame);
    mFirstFragmentManager = getFragmentManager();
    firstTabListFragment = new FirstTabListFragment();
    android.app.FragmentTransaction fragmentTransaction =
mFirstFragmentManager.beginTransaction();
    fragmentTransaction.replace(flTab1Frame.getId(),
firstTabListFragment)
        .addToBackStack("tab1")
        .commit();

    //tab3
    flTab3Frame = (FrameLayout) findViewById(R.id.flTab3Frame);
    mThirdFragmentManager = getFragmentManager();
    thirdTabMyPostFragment = new ThirdTabMyPostFragment();
    fragmentTransaction = mThirdFragmentManager.beginTransaction();
    fragmentTransaction.replace(flTab3Frame.getId(),
thirdTabMyPostFragment)
        .addToBackStack("tab3")
        .commit();

}

@Override
public void onBackPressed() {
    super.onBackPressed();
    int firstTabCount = mFirstFragmentManager.getBackStackEntryCount();
    int thirdTabCount = mThirdFragmentManager.getBackStackEntryCount();
    if (firstTabCount == 1 || thirdTabCount == 1) finish();
}

public void onAddPost(String description, String address, String
contact) {
    if (description.isEmpty()) {
        Toast.makeText(this, "Please input Description.",
Toast.LENGTH_SHORT).show();
        return;
    }
    if (address.isEmpty()) {
        Toast.makeText(this, "Please input Address.",
Toast.LENGTH_SHORT).show();
        return;
    }
    if (contact.isEmpty()) {
        Toast.makeText(this, "Please input Contact.",
Toast.LENGTH_SHORT).show();
        return;
    }
    PostModel post = new PostModel(MyApplication.username, description,
address, contact);
    DatabaseReference ref = database.getReference().child("posts");
    ref.push().setValue(post);
    Toast.makeText(this, "Success.", Toast.LENGTH_SHORT).show();
    etDescription.setText("");
    etAddress.setText("");
    etContact.setText("");
}

public void onReact() {
    firstTabSubmitFragment = new FirstTabSubmitFragment();
    android.app.FragmentTransaction fragmentTransaction =
mFirstFragmentManager.beginTransaction();
    fragmentTransaction.replace(
        flTab1Frame.getId(), firstTabSubmitFragment)
        .addToBackStack("tab1")
        .commit();
}

```

```

    }

    public void onViewReact() {
        thirdTabViewReactFragment = new ThirdTabViewReactFragment();
        android.app.FragmentTransaction fragmentTransaction =
mThirdFragmentManager.beginTransaction();
        fragmentTransaction.replace(
            flTab3Frame.getId(), thirdTabViewReactFragment)
            .addToBackStack("tab3")
            .commit();
    }

    public void onDeletePost() {
        DatabaseReference ref =
database.getReference().child("posts").child(postID);
        ref.setValue(null);
    }

    public void onListView() {
        android.app.FragmentTransaction fragmentTransaction =
mFirstFragmentManager.beginTransaction();
        fragmentTransaction.replace(
            flTab1Frame.getId(), firstTabListFragment)
            .addToBackStack("tab1")
            .commit();
    }

    public void onSubmitReact(String description, String address, String
contact) {
        if (description.isEmpty()) {
            Toast.makeText(this, "Please input Description.",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (address.isEmpty()) {
            Toast.makeText(this, "Please input Address.",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (contact.isEmpty()) {
            Toast.makeText(this, "Please input Contact.",
Toast.LENGTH_SHORT).show();
            return;
        }
        ReactModel react = new ReactModel(authID, MyApplication.username,
description, address, contact);
        DatabaseReference ref = database.getReference().child("reacts");
        ref.push().setValue(react);
        onListView();
    }
}

```

MainActivity.java:

```

package com.yang.education;

import android.content.Intent;
import android.graphics.Color;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;

```

```

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.kaopiz.kprogresshud.KProgressHUD;

public class MainActivity extends AppCompatActivity {

    private EditText etUserName;
    private EditText etPassword;
    private FirebaseDatabase database = FirebaseDatabase.getInstance();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        etUserName = (EditText) findViewById(R.id.etUserName);
        etPassword = (EditText) findViewById(R.id.etPassword);
        Button btnSignin = (Button) findViewById(R.id.btnSignin);
        Button btnRegister = (Button) findViewById(R.id.btnRegister);

        etUserName.setBackground(MyApplication.getRoundDrawable(MainActivity.this,
1, Color.TRANSPARENT, ContextCompat.getColor(this,
R.color.colorPrimaryDark), 1));

        etPassword.setBackground(MyApplication.getRoundDrawable(MainActivity.this,
1, Color.TRANSPARENT, ContextCompat.getColor(this,
R.color.colorPrimaryDark), 1));
        btnSignin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                onSingin();
            }
        });
        btnRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                onRegister();
            }
        });
    }

    private void onSingin() {
        final String username = etUserName.getText().toString();
        final String password = etPassword.getText().toString();
        if (username.isEmpty()) {
            Toast.makeText(this, "Please input Username.",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (password.isEmpty()) {
            Toast.makeText(this, "Please input Password.",
Toast.LENGTH_SHORT).show();
            return;
        }
        final KProgressHUD hud = KProgressHUD.create(this).show();
        DatabaseReference ref = database.getReference().child("users");

        ref.orderByChild("username").equalTo(username).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                hud.dismiss();
                if (dataSnapshot.exists()) {
                    for (DataSnapshot val: dataSnapshot.getChildren()) {
                        if
(val.child("password").getValue().equals(password)) {
                            MyApplication.username = username;
                            startActivity(new Intent(MainActivity.this,

```

```

FirstActivity.class));
        } else {
            Toast.makeText(MainActivity.this, "Incorrect
password.", Toast.LENGTH_LONG).show();
        }
    }
    } else {
        Toast.makeText(MainActivity.this, "There is not user.",
Toast.LENGTH_LONG).show();
    }
}

@Override
public void onCancelled(DatabaseError databaseError) {
    hud.dismiss();
}
});
}

private void onRegister() {
    startActivity(new Intent(MainActivity.this,
RegisterActivity.class));
}
}

```

MyApplication.java:

```

package com.yang.education;

import android.app.Activity;
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Point;
import android.graphics.drawable.Drawable;
import android.graphics.drawable.GradientDrawable;
import android.location.Address;
import android.location.Geocoder;
import android.support.multidex.MultiDexApplication;
import android.view.Display;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Toast;

import com.yang.education.Model.UserModel;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MyApplication extends MultiDexApplication {

    public static String username;
    @Override
    public void onCreate() {
        super.onCreate();
        username = null;
    }

    public static int getDeviceWidth(Activity activity) {
        Display display = activity.getWindowManager().getDefaultDisplay();
        Point size = new Point();
        display.getSize(size);
    }
}

```

```

        return size.x;
    }

    public static int getDeviceHeight(Activity activity) {
        Display display = activity.getWindowManager().getDefaultDisplay();
        Point size = new Point();
        display.getSize(size);

        return size.y;
    }

    public static int getScaledLength(Activity activity, int length) {
        int scaledLength = (int) (length *
activity.getResources().getDisplayMetrics().density);
        return scaledLength;
    }

    public static String getFormattedDateTimeString(String format, long
timestamp) {
        return getFormattedDateTimeString(format, new Date(timestamp *
1000));
    }

    public static String getFormattedDateTimeString(String format, Date
date) {
        DateFormat sdf = new SimpleDateFormat(format);
        return sdf.format(date);
    }

    public static void showKeyboard(Context context, View view) {
        view.requestFocus();
        InputMethodManager imm = (InputMethodManager)
context.getSystemService(Context.INPUT_METHOD_SERVICE);
        imm.showSoftInput(view, InputMethodManager.SHOW_IMPLICIT);
    }

    public static void hideKeyboard(Activity activity){
        try {
            InputMethodManager inputManager = (InputMethodManager)
activity.getSystemService(Context.INPUT_METHOD_SERVICE);

            inputManager.hideSoftInputFromWindow(activity.getCurrentFocus().getWindowTok
en(), InputMethodManager.HIDE_NOT_ALWAYS);
        } catch (NullPointerException e) {
            e.printStackTrace();
        }
    }

    public static String getAddressFromLatitudeAndLongitude(Context context,
double latitude, double longitude) {
        String address = "";
        try {
            Geocoder geocoder;
            List<Address> addresses = null;
            geocoder = new Geocoder(context, Locale.getDefault());
            addresses = geocoder.getFromLocation(latitude, longitude, 1);
            if (addresses != null && addresses.size()>0) {
                Address returnedAddress = addresses.get(0);
                for (int j = 0; j <=
returnedAddress.getMaxAddressLineIndex(); j++) {
                    address = address + returnedAddress.getAddressLine(j);
                }
            }
        } catch (Exception e) {
            Toast.makeText(context, "Google Map error\n" +
e.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
        }
        return address;
    }

```

```

    }

    public static String getPackageName(Context context) {
        String packageName = context.getPackageName();
        return packageName;
    }

    public static String getAppName(Context context) {
        Resources appR = context.getResources();
        String appName = appR.getText(appR.getIdentifier("app_name",
"string", context.getPackageName()).toString());
        return appName;
    }

    public static boolean isEmailValid(String email) {
        boolean isValid = false;

        String expression = "^[\\w\\.\\-]+@([\\w\\.\\-]+\\.)+[A-Z]{2,4}$";
        CharSequence inputStr = email;

        Pattern pattern = Pattern.compile(expression,
Pattern.CASE_INSENSITIVE);
        Matcher matcher = pattern.matcher(inputStr);
        if (matcher.matches()) {
            isValid = true;
        }
        return isValid;
    }

    public static Drawable getRectDrawable(Activity activity, int
solidColor, int strokeColor, int strokeWidth) {
        GradientDrawable drawable = new GradientDrawable();
        drawable.setShape(GradientDrawable.RECTANGLE);
        drawable.setColor(solidColor);
        drawable.setStroke(getScaledLength(activity, strokeWidth),
strokeColor);
        return drawable;
    }

    public static Drawable getRoundDrawable(Activity activity, int
cornerRadius, int solidColor, int strokeColor, int strokeWidth) {
        GradientDrawable drawable = new GradientDrawable();
        drawable.setShape(GradientDrawable.RECTANGLE);
        drawable.setCornerRadius(getScaledLength(activity, cornerRadius));
        drawable.setColor(solidColor);
        drawable.setStroke(getScaledLength(activity, strokeWidth),
strokeColor);
        return drawable;
    }

    public static Drawable getRoundDrawable(Activity activity, int
cornerRadius, int solidColor) {
        GradientDrawable drawable = new GradientDrawable();
        drawable.setShape(GradientDrawable.RECTANGLE);
        drawable.setCornerRadius(getScaledLength(activity, cornerRadius));
        drawable.setColor(solidColor);
        return drawable;
    }

    public static Drawable getOvalDrawable(Activity activity, int
solidColor, int strokeColor, int strokeWidth) {
        GradientDrawable drawable = new GradientDrawable();
        drawable.setShape(GradientDrawable.OVAL);
        drawable.setColor(solidColor);
        drawable.setStroke(getScaledLength(activity, strokeWidth),
strokeColor);
        return drawable;
    }

```

```

    }

    public static Drawable getOvalDrawable(Activity activity, int
solidColor) {
        GradientDrawable drawable = new GradientDrawable();
        drawable.setShape(GradientDrawable.OVAL);
        drawable.setColor(solidColor);
        return drawable;
    }
}

```

RegisterActivity.java:

```

package com.yang.education;

import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TabHost;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.kaopiz.kprogresshud.KProgressHUD;
import com.yang.education.Model.UserModel;

public class RegisterActivity extends AppCompatActivity {

    private FirebaseDatabase database = FirebaseDatabase.getInstance();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        TabHost host = (TabHost) findViewById(R.id.tabHost);
        host.setup();

        //Tab 1
        TabHost.TabSpec spec = host.newTabSpec("Tab One");
        spec.setContent(R.id.tab1);
        spec.setIndicator("Register User");
        host.addTab(spec);

        //Tab 2
        spec = host.newTabSpec("Tab Two");
        spec.setContent(R.id.tab2);
        spec.setIndicator("Register NGO");
        host.addTab(spec);

        // tab1
        final EditText etUserName = (EditText) findViewById(R.id.etUserName);
        final EditText etFullName = (EditText) findViewById(R.id.etFullName);
        final EditText etPassword = (EditText) findViewById(R.id.etPassword);
        final EditText etConfirmPassword =
(EditText) findViewById(R.id.etConfirmPassword);
        final EditText etEmail = (EditText) findViewById(R.id.etEmail);
        final EditText etContact = (EditText) findViewById(R.id.etContact);
        Button btnRegister = (Button) findViewById(R.id.btnRegister);
    }
}

```

```

etUserName.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etFullName.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etPassword.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etConfirmPassword.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etEmail.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etContact.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

        final EditText etNGOName = (EditText) findViewById(R.id.etNGOName);
        final EditText etNGOType = (EditText) findViewById(R.id.etNGOType);
        final EditText etNGOUserName =
(EditText) findViewById(R.id.etNGOUserName);
        final EditText etNGOPassword =
(EditText) findViewById(R.id.etNGOPassword);
        final EditText etNGOConfirmPassword =
(EditText) findViewById(R.id.etNGOConfirmPassword);
        final EditText etNGOEmail = (EditText) findViewById(R.id.etNGOEmail);
        final EditText etNGOContact =
(EditText) findViewById(R.id.etNGOContact);
        Button btnNGORegister = (Button) findViewById(R.id.btnNGORegister);

etNGOName.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etNGOType.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etNGOUserName.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etNGOPassword.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etNGOConfirmPassword.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etNGOEmail.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

etNGOContact.setBackground(MyApplication.getRoundDrawable(RegisterActivity.this, 1, Color.TRANSPARENT, ContextCompat.getColor(this, R.color.colorPrimaryDark), 1));

        btnRegister.setOnClickListener(new View.OnClickListener() {
            @Override

```



```

        public void onClick(View view) {
            RegisterUser(0, "normal", etUserName.getText().toString(),
etFullName.getText().toString(), etPassword.getText().toString(),
etConfirmPassword.getText().toString(),
                etEmail.getText().toString(),
etContact.getText().toString());
        }
    });

    btnNGORegister.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            RegisterUser(1, etNGOType.getText().toString(),
etNGOUserName.getText().toString(), etNGOName.getText().toString(),
etNGOPassword.getText().toString(),
                etNGOConfirmPassword.getText().toString(),
etNGOEmail.getText().toString(), etNGOContact.getText().toString());
        }
    });

    public void RegisterUser(final int flag, final String type, final String
username, final String fullname, final String password, String confirm,
final String email, final String contact) {
        if (flag == 1 && type.isEmpty()) {
            Toast.makeText(this, "Please input NGO Type.",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (fullname.isEmpty()) {
            if (flag == 0) {
                Toast.makeText(this, "Please input Username.",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "Please input NGO Name.",
Toast.LENGTH_SHORT).show();
            }
            return;
        }
        if (username.isEmpty()) {
            Toast.makeText(this, "Please Username.",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (password.isEmpty()) {
            Toast.makeText(this, "Please input password.",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (!password.equals(confirm)) {
            Toast.makeText(this, "Passwords don't match.",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (!MyApplication.isEmailValid(email)) {
            Toast.makeText(this, "Invalid Email.",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (contact.isEmpty()) {
            Toast.makeText(this, "Please input contact.",
Toast.LENGTH_SHORT).show();
            return;
        }

        final KProgressHUD hud = KProgressHUD.create(this).show();
        final DatabaseReference ref =
database.getReference().child("users");

```

```

ref.orderByChild("username").equalTo(username).addListenerForSingleValueEven
t(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        hud.dismiss();
        if (dataSnapshot.exists()) {
            onFaillRegister();
        } else {
            UserModel user;
            if (flag == 0) {
                user = new UserModel("normal", type, fullname,
username, password, email, contact);
            } else {
                user = new UserModel("NGO", type, fullname,
username, password, email, contact);
            }
            ref.push().setValue(user);
            MyApplication.username = user.getUsername();
            startActivity(new Intent(RegisterActivity.this,
FirstActivity.class));
            finish();
        }
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        hud.dismiss();
    }
});
}

public void onFaillRegister() {
    Toast.makeText(this, "Username is used.",
Toast.LENGTH_SHORT).show();
}
}

```

6.2.2 Fragments Java Files

FirstTabListFragment.java:

```

package com.yang.education.Fregment;

import android.app.Activity;
import android.content.Context;
import android.content.DialogInterface;
import android.graphics.Color;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.yang.education.FirstActivity;

```

```

import com.yang.education.Model.PostModel;
import com.yang.education.MyApplication;
import com.yang.education.R;

import java.util.ArrayList;
import java.util.List;

public class FirstTabListFragment extends android.app.Fragment {

    private FirstActivity parent;
    private RecyclerView lvFirstTabList;
    private List<PostModel> mPostList = new ArrayList<>();
    private ListViewAdapter mAdapter;
    private LinearLayoutManager mLinearManager;

    private FirebaseDatabase database = FirebaseDatabase.getInstance();

    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        parent = (FirstActivity) getActivity();
        View view = inflater.inflate(R.layout.fregment_firsttab_list,
container, false);
        lvFirstTabList =
(RecyclerView) view.findViewById(R.id.lvFirstTabList);
        mLinearManager = new LinearLayoutManager(parent);
        getAllPost();
        return view;
    }

    public void getAllPost() {
        DatabaseReference ref = database.getReference().child("posts");
        ref.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                mPostList.clear();
                for (DataSnapshot val:dataSnapshot.getChildren()) {
                    if
(val.child("username").getValue().toString().equals(MyApplication.username))
continue;
                    PostModel post = new
PostModel(val.child("username").getValue().toString(),
val.child("description").getValue().toString(),
                    val.child("address").getValue().toString(),
val.child("contact").getValue().toString());
                    post.setId(val.getKey());

post.setTimestamp(val.child("timestamp").getValue().toString());
                    mPostList.add(post);
                }
                lvFirstTabList.setLayoutManager(mLinearManager);
                mAdapter = new ListViewAdapter(parent, mPostList);
                lvFirstTabList.setAdapter(mAdapter);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }

    private class ListViewAdapter extends
RecyclerView.Adapter<ListViewAdapter.ViewHolder> {
        Context context;
        List<PostModel> mPostList;
        public ListViewAdapter(Context con, List<PostModel> posts) {

```

```

        context = con;
        mPostList = posts;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
    {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item_firsttab, parent, false);
        ViewHolder viewHolder = new ViewHolder(view);
        return viewHolder;
    }

    private String ChangeDescription(String description) {
        if (description.length() < 20) return description;
        return description.subSequence(0, 20) + " ...";
    }

    @Override
    public void onBindViewHolder(final ViewHolder holder, int position)
    {
        PostModel post = mPostList.get(getItemCount() - position - 1);
        holder.tvUserName.setText(post.getUsername());
        holder.tvTime.setText("Time : " +
            MyApplication.getFormattedDateTimeString("hh:mm:ss a dd/MM/yyyy",
            Long.parseLong(post.getTimestamp())));

        holder.tvDescription.setText(ChangeDescription(post.getDescription()));
        holder.descriptionDetails = "Description : " +
            post.getDescription() +
            "\n\nContact : " + post.getContact()
            +
            "\n\nAddress : " +
            post.getAddress();
        holder.postID = post.getId();
        String username = post.getUsername();
        DatabaseReference ref = database.getReference().child("users");
        ref.orderByChild("username").equalTo(username).addListenerForSingleValueEven
        t(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                for (DataSnapshot val:dataSnapshot.getChildren()) {
                    if (val.child("type").getValue().equals("NGO")) {
                        holder.tvNGOType.setText("NGO");
                    } else {
                        holder.tvNGOType.setText("");
                    }
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }

    @Override
    public int getItemCount() {
        return mPostList.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        public TextView tvUserName;
        public TextView tvTime;
        public TextView tvDescription;
        public TextView tvNGOType;
    }

```

```

        public String postID;
        public String descriptionDetails;
        public LinearLayout llListViewItem;
        public ViewHolder(View view) {
            super(view);
            tvUserName = (TextView)view.findViewById(R.id.tvUserName);
            tvTime = (TextView)view.findViewById(R.id.tvTime);
            tvDescription =
(TextView)view.findViewById(R.id.tvDescription);
            llListViewItem =
(LinearLayout)view.findViewById(R.id.llListViewItem);
            tvNGOType = (TextView)view.findViewById(R.id.tvNGOType);

            Button btnReact = (Button)view.findViewById(R.id.btnReact);
            btnReact.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    parent.authID = postID;
                    parent.onReact();
                }
            });
            llListViewItem.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                AlertDialog alertDialog = new
AlertDialog.Builder(parent).create();
                alertDialog.setTitle("Details");
                alertDialog.setMessage(descriptionDetails);
                alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL,
"OK",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int which) {
                            dialog.dismiss();
                        }
                    });
                alertDialog.show();
            }
        });
    }
}

```

FirstTabSubmitFragment.java:

```

package com.yang.education.Fregment;

import android.content.Context;
import android.graphics.Color;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.content.ContextCompat;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;

```

```

import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.yang.education.FirstActivity;
import com.yang.education.Model.PostModel;
import com.yang.education.MyApplication;
import com.yang.education.R;

import java.util.ArrayList;
import java.util.List;

public class FirstTabSubmitFragment extends android.app.Fragment {

    private FirstActivity parent;

    private FirebaseDatabase database = FirebaseDatabase.getInstance();
    private EditText etReactContact;
    private EditText etReactAddress;
    private EditText etReactDescription;

    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        parent = (FirstActivity) getActivity();
        View view = inflater.inflate(R.layout.fregment_firsttab_submit,
container, false);

        etReactDescription =
(EditText) view.findViewById(R.id.etReactDescription);
        etReactAddress = (EditText) view.findViewById(R.id.etReactAddress);
        etReactContact = (EditText) view.findViewById(R.id.etReactContact);

        etReactDescription.setBackground(MyApplication.getRoundDrawable(parent, 1,
Color.TRANSPARENT, ContextCompat.getColor(parent,
R.color.colorPrimaryDark), 1));
        etReactAddress.setBackground(MyApplication.getRoundDrawable(parent,
1, Color.TRANSPARENT, ContextCompat.getColor(parent,
R.color.colorPrimaryDark), 1));
        etReactContact.setBackground(MyApplication.getRoundDrawable(parent,
1, Color.TRANSPARENT, ContextCompat.getColor(parent,
R.color.colorPrimaryDark), 1));

        Button btnSubmit = (Button) view.findViewById(R.id.btnSubmit);
        btnSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

parent.onSubmitReact(etReactDescription.getText().toString(),
etReactAddress.getText().toString(), etReactContact.getText().toString());
            }
        });
        return view;
    }
}

```

ThirdTabMyPostFragment.java:

```

package com.yang.education.Fregment;

import android.content.Context;
import android.os.Bundle;
import android.support.annotation.IntegerRes;
import android.support.annotation.Nullable;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;

```

```

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;
import com.yang.education.FirstActivity;
import com.yang.education.Model.PostModel;
import com.yang.education.MyApplication;
import com.yang.education.R;

import java.util.ArrayList;
import java.util.List;

public class ThirdTabMyPostFragment extends android.app.Fragment {

    private FirstActivity parent;
    private RecyclerView lvThirdTabList;
    private List<PostModel> mPostList = new ArrayList<>();
    private ListViewAdapter mAdapter;
    private LinearLayoutManager mLinearManager;

    private FirebaseDatabase database = FirebaseDatabase.getInstance();

    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        parent = (FirstActivity) getActivity();
        View view = inflater.inflate(R.layout.fragment_thirdtab_post,
container, false);
        lvThirdTabList =
(RecyclerView) view.findViewById(R.id.lvThirdTabList);
        mLinearManager = new LinearLayoutManager(parent);
        getMyPost();
        return view;
    }

    public void getMyPost() {
        DatabaseReference ref = database.getReference().child("posts");
        Query query =
ref.orderByChild("username").equalTo(MyApplication.username);
        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                mPostList.clear();
                for (DataSnapshot val:dataSnapshot.getChildren()) {
                    PostModel post = new
PostModel(val.child("username").getValue().toString(),
val.child("description").getValue().toString(),
                    val.child("address").getValue().toString(),
val.child("contact").getValue().toString());
                    post.setTimestamp(val.child("timestamp").getValue().toString());
                    post.setId(val.getRef().getKey());
                    mPostList.add(post);
                }
                lvThirdTabList.setLayoutManager(mLinearManager);
                mAdapter = new ListViewAdapter(parent, mPostList);
                lvThirdTabList.setAdapter(mAdapter);
            }
        });
    }
}

```

```

        @Override
        public void onCancelled(DatabaseError databaseError) {
        }
    });
}

private class ListViewAdapter extends
RecyclerView.Adapter<ListViewAdapter.ViewHolder> {
    Context context;
    List<PostModel> mPostList;
    public ListViewAdapter(Context con, List<PostModel> posts) {
        context = con;
        mPostList = posts;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
    {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.list_item_thirddtab, parent, false);
        ViewHolder viewHolder = new ViewHolder(view);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(final ViewHolder holder, final int
position) {
        final PostModel post = mPostList.get(position);
        holder.tvMyDescription.setText(post.getDescription());
        holder.postID = post.getId();
        DatabaseReference ref = database.getReference().child("reacts");
        Query query = ref.orderByChild("authID").equalTo(post.getId());
        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                int count = 0;
                for (DataSnapshot val:dataSnapshot.getChildren()) {
                    count++;
                }
                holder.btnShowReact.setText(Integer.toString(count) + "
people reacted");
                holder.reactedCount = count;
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }

    @Override
    public int getItemCount() {
        return mPostList.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
        public TextView tvMyDescription;
        public Button btnShowReact;
        public Button btnDelete;
        public int reactedCount;
        public String postID;
        public ViewHolder(View view) {
            super(view);
            tvMyDescription =
(TextView)view.findViewById(R.id.tvMyDescription);
            reactedCount = 0;
            btnShowReact = (Button)view.findViewById(R.id.btnShowReact);
            btnShowReact.setOnClickListener(new View.OnClickListener() {
                @Override

```



```

        TextView tvReactDescription =
            (TextView) view.findViewById(R.id.tvReactDescription);
        tvReactDescription.setText(parent.postDescription);
        btnShowReact_2 = (Button) view.findViewById(R.id.btnShowReact_2);
        mReactList = view.findViewById(R.id.lvThirdTabReactList);
        getAllReacts();
        return view;
    }

    private void getAllReacts() {
        DatabaseReference ref = database.getReference().child("reacts");
        Query query = ref.orderByChild("authID").equalTo(parent.postID);
        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                mReacts.clear();
                for (DataSnapshot val:dataSnapshot.getChildren()) {
                    ReactModel react = new
ReactModel(val.child("authID").getValue().toString(),
val.child("username").getValue().toString(),
                    val.child("description").getValue().toString(),
val.child("address").getValue().toString(),
val.child("contact").getValue().toString());
                    mReacts.add(react);
                }
                btnShowReact_2.setText(Integer.toString(mReacts.size()) + "
people reacted");
                mLinearManager = new LinearLayoutManager(parent);
                mReactList.setLayoutManager(mLinearManager);
                mAdapter = new ListViewAdapter(parent, mReacts);
                mReactList.setAdapter(mAdapter);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }

    private class ListViewAdapter extends
RecyclerView.Adapter<ListViewAdapter.ViewHolder> {
        Context context;
        List<ReactModel> mReacts;
        public ListViewAdapter(Context con, List<ReactModel> reacts) {
            context = con;
            mReacts = reacts;
        }

        @Override
        public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
        {
            View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.list_item_thirdtab_react, parent,
false);
            ViewHolder viewHolder = new ViewHolder(view);
            return viewHolder;
        }

        @Override
        public void onBindViewHolder(final ViewHolder holder, final int
position) {
            final ReactModel react = mReacts.get(position);
            holder.tvReactDetail.setText(Integer.toString(position + 1) + "
. " + react.getUsername() + "\n          Description : " +
react.getDescription() + "\n          Address : "
            + react.getAddress() + "\n          Contact : " +
react.getContact());
        }
    }

```

```

@Override
public int getItemCount() {
    return mReacts.size();
}

public class ViewHolder extends RecyclerView.ViewHolder {
    public TextView tvReactDetail;
    public ViewHolder(View view) {
        super(view);
        tvReactDetail =
            (TextView) view.findViewById(R.id.tvReactDetail);
    }
}
}

```

6.3 FIREBASE DATA STORAGE (JSON TREE FORMAT)

Root Node



Fig. 6.3.1 Root Node

Root node consists three children (posts, reacts and users).

Posts Node



Fig. 6.3.2 Posts Node

Posts node contains information or data related to posts and their relations to users.

Posts child Node

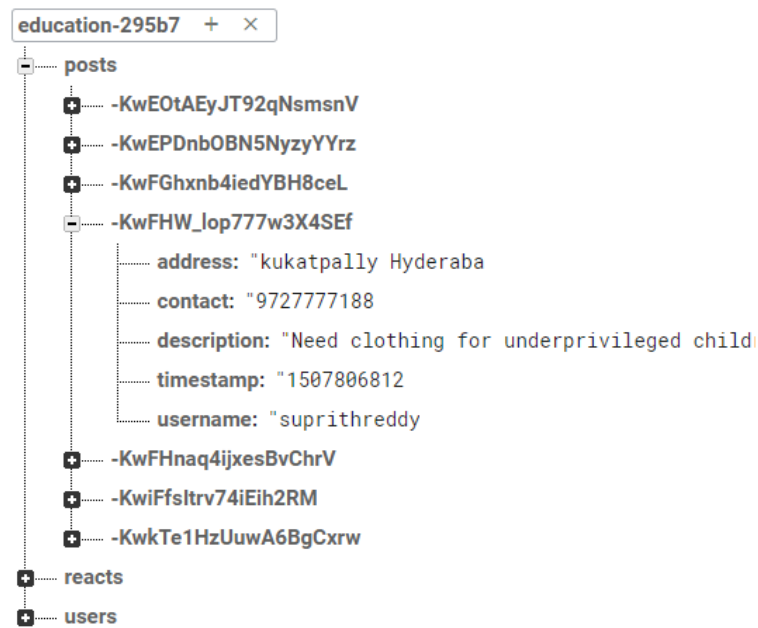


Fig. 6.3.3 Posts child Node

Posts child node contains specific user posts which is rooted to node id.

React Node

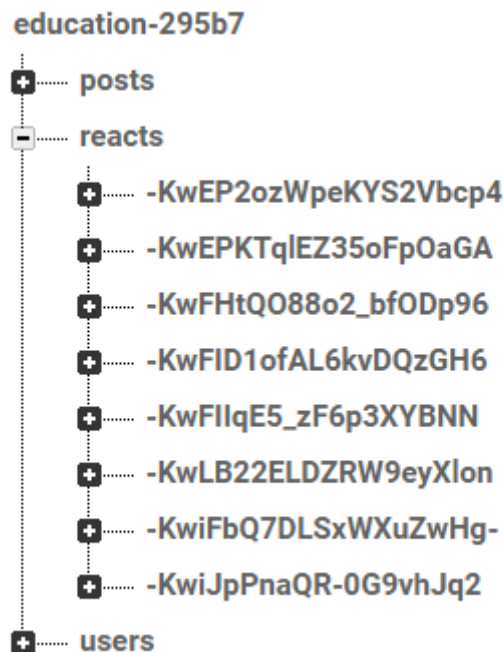


Fig. 6.3.4 React Node

React node contains information or data related to reactions and their relations to users.

Reacts child Node



Fig. 6.3.5 Reacts child Node

Posts child node contains specific user posts which is rooted to node id.

User Node

User node contains roots of the user information as its children.

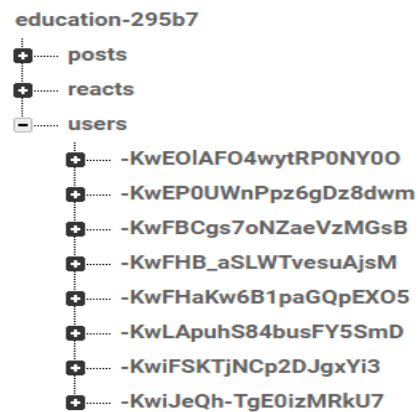


Fig 6.3.6 User Node

Users child Node

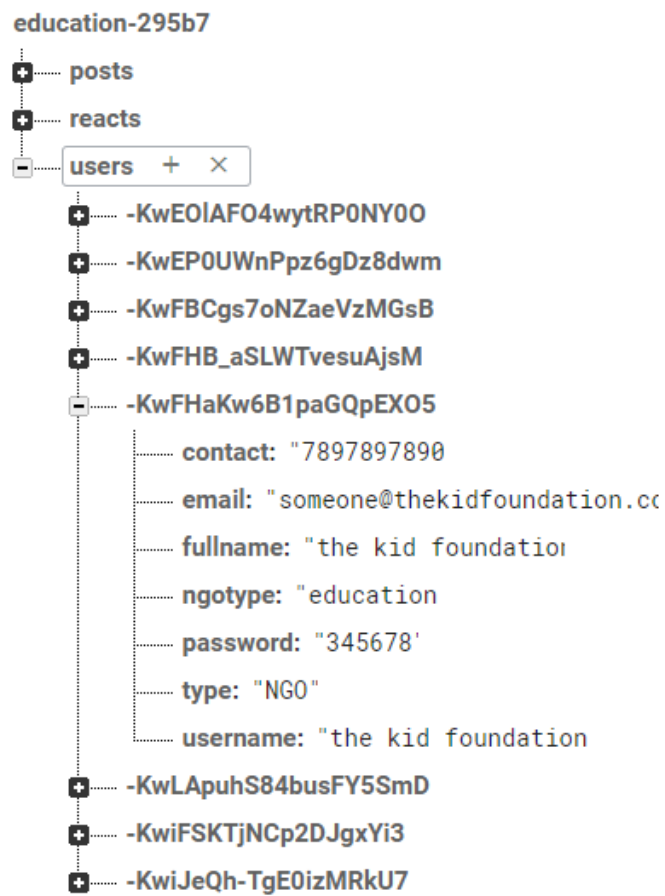


Fig. 6.3.7 Users child Node

User child node contains data related to user details such as username, full name, password, contact, etc., of specific user rooted to node id.

7. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Performance Testing

It is based on the performance levels of the application, despite of number of users simultaneously running the application.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Test Case ID	Test Case	Expecting behavior	Exhibiting behavior	Result
1	Testing the Mobile for User Registration.	Data should be sent to cloud server.	Updating regularly and dynamically without any issues.	Pass
2	Testing the Mobile for authenticated user login.	It has to give access to authorised user by user credentials.	Access given to only authorised users instead failure message toasts up.	Pass
3	Testing the App for storage of posts and reactions to posts.	It has to save the description and contact details in Database (Cloud).	Storing the posts and reactor information in the Database.	Pass
4	Testing the App for proper retrieval of posts and displaying them to user.	It has to orderly display the posts information by retriving from database.	It is showing the updated posts or most recent post on top of the page.	Pass
5	Testing the App for proper deletion of posts.	It has delete the posts from the DB on clicking delete button.	Post gets deleted from DB on clicking delete.	Pass

Table 7.1. Tests Performed

CONCLUSION

HelpingNeedyApp is a novel app that collects and stores specific requirement details of a user where that data has been visualized to others. One of the best feature in this application it will display the specific reactions or responses which are highly helpful in emergency durations. Based on the information provided by a user, people around us would get benefit through our App.

HelpingNeedyApp will help to break the barricades among the common users and NGOs.

Future Scope: In near future a chat (conversation) module also could be developed and integrated to this application for more clear communication.

Social Impact: This application has real time impact on society by direct usage of app instead of referring to many websites which could increase the number of users as fast as possible that includes NGOs and normal users.

BIBLIOGRAPHY

- [1] HeadFirst Android Development by Jonathan Simon
- [2] Beginning Android 4 Application Development by Wei-Meng Lee
- [3] Professional Android 2 Application Development by Reto Meier(Wrox)
- [4] Programming Android by Zigurd Mednieks, Laird Dornin, G. Blake Meike,
Masumi Nakamura
- [5] Firebase Tutorial by Simplified Coding and Google Firebase Docs.

Sites Referred:

- [6] <http://www.wikipedia.org/android-operatingsystem/>
- [7] <http://developer.android.com/guide/index.html>
- [8] <http://mobile.tutsplus.com/tutorials/android/java-tutorial/>
- [9] <http://www.simplifiedcoding.com/firebase-tutorial/>
- [10] <http://www.firebase.google.com/docs/>

PHOTOGRAPH

