

MANUU Connect

Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology
In
Computer Science & Engineering

By

Saiful Hasan

To

Ambedkar Institute of Advance Communication Technology and Research,
Geeta Colony , New Delhi

October 2017

Certificate (Student)

I hereby declare that the Industrial Training Report entitled ("MANUU Connect") is an authentic record of my own work as requirements of Industrial Training during the period from 5-June 2017 to 5-August 2017 for the award of degree of B.Tech. (Computer Science & Engineering), Ambedkar Institute of Advance Communication Technology and Research, Geeta Colony, New Delhi-58, under the guidance of Mr. Muntasir Alam.

Date: _____

Name : Saiful Hasan

Roll No. : 00710107216

ACKNOWLEDGEMENT

First of all, I would like to give my thanks to God, without his concern this project is not possible.

Secondly to my mentor Mr. Muntasir Alam who gave me this golden opportunity to do this project under his supervision and guidance, without his motivational wording, I was unable to set off this project. His acquiesce gave me a lot of confidence. Apart from these without his little collaboration, I won't have imagined this.

I heartily thank my team member, Mr. Monib Rehman for his support and suggestions during this project work and for creating an effective working environment.

I owe my deep gratitude to students of Maulana Azad National Urdu University, who took keen interest in this project and gave suggestions, till the completion of our project work by providing all the necessary information for developing a good system.

A special thanks to my parents, as whatever I was, whatever I am and whatever I will be, you've never known what I've become because of you.

And finally, a very-very thanks to my friends who helped me a lot bringing this project up to your gallery and gave a feeling of being beatitude.

(Saiful Hasan)

Abstract

The Android application is gaining more importance as the numbers of its users are increasing rapidly. As the number is rising, it is necessary for each and every college and universities to have an android application to share information to its students. One such android application is MANUU Connect; it's designed using XML as front end and Java as backend.

MANUU Connect is an android application which is designed and developed by Utime Tech Pvt. Ltd. for Maulana Azad National Urdu University, Hyderabad. The motive of MANUU Connect is to connect its students through an android application.

The application contains details like current news, current admission, employment news, Result sheet, Course structure, Exam results, etc. The application also contains separate login for Student and Employee.

The main feature of this application is to provide notification to its users each and every time new information is added. The notifications help its users to go through each and every information, which is provided by the university. The notification also saves its users from the hassle of checking application frequently for new information. As soon as new information is available, it will be notified to the users.

This report will give the detail about the background research on MANUU Connect.

TABLE OF CONTENTS

Chapter 1: Android

Introduction	8
Features	9
Architecture.....	13-16
Application Components	16-19

Chapter 2: About the Project

Objective.....	20
Language and Software Used	20
Structure of the project.....	20
Future Scope of the project.....	21
Hardware requirement.....	21

Chapter 3: Methodologies

Introduction.....	22
SDLC.....	22-24
SDLC Models	25
Model Used in the project.....	25-27

Chapter 4: Modules

Home.....	28
Admission.....	28
Employment News.....	28

Result Sheet.....	28
Course Structure.....	28
Exam Result.....	28
Student Login.....	28
Employee Login	29
Contact Us.....	29
Website.....	29

Chapter 5: Components Used in the project

Volley	30-31
Navigation Drawer.....	31-40
Recycler View.....	40-42
Firebase Cloud Messaging.....	42-44
Card View.....	44-46
Web View.....	46-47

Chapter 6: Outputs

Screenshots.....	48-55
------------------	-------

1. Android

1.1 Introduction

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as smart phones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touch screen devices, Google has further developed Android TV for televisions, Android Auto for cars and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Beginning with the first commercial Android device in September 2008, the operating system has gone through multiple major releases, with the current version being 8.0 "Oreo", released in August 2017. Android applications ("apps") can be downloaded from the Google Play store, which features over 2.7 million apps as of February 2017. Android has been the best-selling OS on tablets since 2013, and runs on the vast majority of smart phones. As of May 2017, Android has two billion monthly active users, and it has the largest installed base of any operating system.

Android's source code is released by Google under an open source license, although most Android devices ultimately ship with a combination of free and open source and proprietary software, including proprietary software required for accessing Google services. Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices. Its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with

other operating systems. The extensive variation of hardware in Android devices causes significant delays for software upgrades, with new versions of the operating system and security patches typically taking months before reaching consumers, or sometimes not at all. The success of Android has made it a target for patent and copyright litigation between technology companies.

1.2 Features

1.2.1 General

1.2.1.1 Messaging

SMS and MMS are available forms of messaging, including threaded text messaging and Android Cloud to Device Messaging (C2DM) and now enhanced version of C2DM, Android Google Cloud Messaging (GCM) is also a part of Android Push Messaging services.

1.2.1.2 Web browser

The web browser available in Android is based on the open-source Blink (previously WebKit) layout engine, coupled with Chrome's V8 JavaScript engine. Then the WebKit-using Android Browser scored 100/100 on the Acid3 test on Android 4.0 ICS; the Blink-based browser currently has better standards support. The browser is variably known as 'Android Browser', 'AOSP browser'[Appendix], 'stock browser', 'native browser', and 'default browser'. Starting with Android 4.4 KitKat, Google has mandated that the default browser for Android proper be Google Chrome. Since Android 5.0 Lollipop, the WebView browser that apps can use to display web content without leaving the app has been separated from the rest of the Android firmware in order to facilitate separate security updates by Google.

1.2.1.3 Voice-based features

Google search through voice has been available since initial release. Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards. As of Android 4.1, Google has expanded Voice Actions with ability to talk back and read answers from

Google's Knowledge Graph when queried with specific commands. The ability to control hardware has not yet been implemented.

1.2.1.4 Multi-touch

Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology at the time). Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively.

1.2.1.5 Multitasking

Multitasking of applications, with unique handling of memory allocation, is available.

1.2.1.6 Screen capture

Android supports capturing a screenshot by pressing the power and home-screen buttons at the same time. Prior to Android 4.0, the only methods of capturing a screenshot were through manufacturer and third-party customizations (apps), or otherwise by using a PC connection (DDMS developer's tool). These alternative methods are still available with the latest Android.

1.2.1.7 TV recording

Android TV supports capturing video and replaying it.

1.2.1.8 Video calling

Android does not support native video calling, but some handsets have a customized version of the operating system that supports it, either via the UMTS [Appendix] network (like the Samsung Galaxy S) or over IP. Video calling through Google Talk is available in Android 2.3.4 (Gingerbread) and later. Gingerbread allows Nexus S to place Internet calls with a SIP account. This allows for enhanced VoIP dialing to other SIP accounts and even phone numbers. Skype 2.1 offers video calling in Android 2.3, including front camera

support. Users with the Google+ Android app can video chat with other Google+ users through Hangouts.

1.2.1.9 Multiple language support

Android supports multiple languages.

1.2.1.10 Accessibility

Built-in text-to-speech is provided by TalkBack for people with low or no vision. Enhancements for people with hearing difficulties are available, as are other aids.

1.2.2 Connectivity

1.2.2.1 Connectivity

Android supports connectivity technologies including GSM/EDGE [Appendix], Bluetooth, LTE [Appendix], CDMA [Appendix], EV-DO[Appendix], UMTS [Appendix], NFC[Appendix], IDEN [Appendix] and WiMAX.

1.2.2.2 Bluetooth

Supports voice dialing and sending contacts between phones, playing music, sending files (OPP), accessing the phone book (PBAP), A2DP[Appendix] and AVRCP[Appendix]. Keyboard, mouse and joystick (HID) support is available in Android 3.1+, and in earlier versions through manufacturer customizations and third-party applications.

1.2.2.3 Tethering

Android supports tethering, which allows a phone to be used as a wireless/wired Wi-Fi hotspot. Before Android 2.2 this was supported by third-party applications or manufacturer customizations.

1.2.3 Media

1.2.3.1 Streaming media support

RTP/RTSP[Appendix] streaming (3GPP PSS, ISMA), HTML progressive download (HTML5 <video> tag). Adobe Flash Streaming (RTMP) and HTTP Dynamic Streaming are supported by the Flash plugin. Apple HTTP Live Streaming is supported by RealPlayer for Android and by the operating system since Android 3.0 (Honeycomb).

1.2.3.2 Media support

Android supports the following audio/video/still media formats: WebM[Appendix], H.263[Appendix], H.264[Appendix], AAC[Appendix], HE-AAC (in 3GP or MP4 container), MPEG-4 SP, AMR[Appendix], AMR-WB (in 3GP container), MP3, MIDI[Appendix], Ogg Vorbis, FLAC, WAV, JPEG, PNG, GIF, BMP, and WebP.

1.2.3.3 External storage

Most Android devices include microSD card slots and can read microSD cards formatted with the FAT32, Ext3 or Ext4 file systems. To allow use of external storage media such as USB flash drives and USB HDDs, some Android devices are packaged with USB-OTG cables. Storage formatted with FAT32 is handled by the Linux Kernel vFAT driver, while 3rd party solutions are required to handle some other file systems such as NTFS, HFS Plus and exFAT.

1.2.4 Hardware support

Android devices can include still/video cameras, touch screens, GPS, accelerometers, gyroscopes, barometers, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers, accelerated 2D bit blits (with hardware orientation, scaling, pixel format conversion) and accelerated 3D graphics.

1.2.5 Other

1.2.5.1 Java support

While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik[Appendix] executables and run on using Android Runtime or in Dalvik in older versions, a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party applications.

1.2.5.2 Handset layouts: The platform works for various screen sizes from smartphone sizes and to tablet size, and can potentially connect to an external screen, e.g. through HDMI, or wirelessly with Miracast. Portrait and landscape orientations are supported and usually switching between by turning. A 2D graphics library, 3D graphics library based on OpenGL ES 2.0 specifications is used.

1.2.5.3 Storage:

SQLite, a lightweight relational database, is used for data storage purposes.

1.2.5.4 Native Apps:

Android apps are also written in HTML.

1.2.5.5 Instant Apps

Android apps are hosted on a specific website path and load instead of the website itself. They are part-apps and load almost instantly without the need for an installation. One of the first apps being developed with such functionality is the B&H app

1.3 Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.

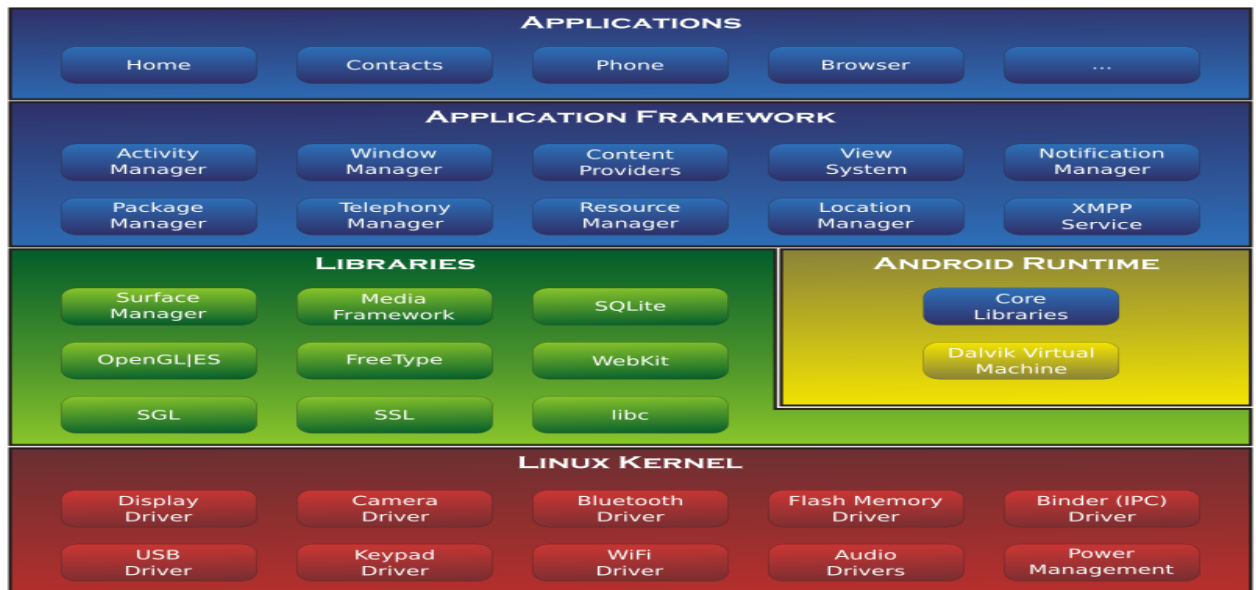


Fig 1.3.1 Android Architecture

1.3.1 Linux kernel

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

1.3.2 Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

1.3.3 Android Libraries

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and

database access. A summary of some key core Android libraries available to the Android developer is as follows –

1.3.3.1 android.app – Provides access to the application model and is the cornerstone of all Android applications.

1.3.3.2 android.content – Facilitates content access, publishing and messaging between applications and application components.

1.3.3.3 android.database – Used to access data published by content providers and includes SQLite database management classes.

1.3.3.4 android.opengl – A Java interface to the OpenGL ES 3D graphics rendering API.

1.3.3.4 android.os – Provides applications with access to standard operating system services including messages, system services and inter-process communication.

1.3.3.5 android.text – Used to render and manipulate text on a device display.

1.3.3.6 android.view – The fundamental building blocks of application user interfaces.

1.3.3.7 android.widget – A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.

1.3.3.8 android.webkit – A set of classes intended to allow web-browsing capabilities to be built into applications.

1.3.4 Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

1.3.5 Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services –

1.3.5.1 Activity Manager – Controls all aspects of the application lifecycle and activity stack.

1.3.5.2 Content Providers – Allows applications to publish and share data with other applications.

1.3.5.3 Resource Manager – Provides access to non-code embedded resources such as strings, color settings and user interface layouts.

1.3.5.4 Notifications Manager – Allows applications to display alerts and notifications to the user.

1.3.5.6 View System – an extensible set of views used to create application user interfaces.

1.3.6 Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

1.4 Android - Application Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file

AndroidManifest.xml that describes each component of the application and how they interact. There are following four main components that can be used within an Android application –

Sr.No	Components & Description
1	Activities They dictate the UI and handle the user interaction to the smart phone screen.
2	Services They handle background processing associated with an application.
3	Broadcast Receivers They handle communication between Android OS and applications.
4	Content Providers They handle data and database management issues.

Table 1.4.1 Android Components

1.4.1 Activities

An activity represents a single screen with a user interface, in short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of Activity class as follows –

```
public class MainActivity extends Activity {  
  
}
```

1.4.2 Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of Service class as follows –


```
public class MyService extends Service {  
  
}
```

1.4.3 Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcaster as an Intent object.

```
public class MyReceiver extends BroadcastReceiver {  
  
    public void onReceive(context,intent){ }  
  
}
```

1.4.4 Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider {  
  
    public void onCreate(){ }  
  
}
```

1.4.5 Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are –

S.No	Components & Description
1	Fragments Represents a portion of user interface in an Activity.
2	Views UI elements that are drawn on-screen including buttons, lists forms etc.
3	Layouts View hierarchies that control screen format and appearance of the views.
4	Intents Messages wiring components together.
5	Resources External elements, such as strings, constants and drawable pictures.
6	Manifest Configuration file for the application.

Table 1.4.2 Additional Components

2. Title of the Project: MANUU Connect

2.1 Objective

The Android application is gaining more importance as the numbers of its users are increasing rapidly. As the number is rising, it is necessary for each and every college and universities to have an android application to share information to its students.

The application contains details like current news, current admission, employment news, Result sheet, Course structure, Exam results, etc. The application also contains separate login for Student and Employee.

The main feature of this application is to provide notification to its users each and every time new information is added. The notifications help its users to go through each and every information, which is provided by the university. The notification also saves its users from the hassle of checking application frequently for new information. As soon as new information is available, it will be notified to the users.

2.2 Language and Software Used

- Front end: XML
- Back end: Java
- Operating System: Android
- Tool: Android Studio

2.3 Structure of the project

2.3.1 Proposed System

In the proposed system we assume that every student have enrollment number and enrollment password which is provided by the university. Similarly every employee has user name and password. The individual details like mark sheets, timetable, etc can only be viewed by the students and employees of the university by providing their user name and password. However details like Admissions, Course structure, result sheet, etc are available for all users.

2.4 Future Scope of the project

- Online fee payment can be added
- Attendance functionality can be added
- Online tutorials can be added
- Online Library functionality can be added

2.5 Hardware requirement

- **Operating System:** Android (Min Sdk Version: 21)

3. Methodologies

3.1 Introduction

A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system. It is also termed as software development life cycle. It is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.

3.2 What is SDLC?

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



Fig 3.2.1 SDLC Life Cycle

A typical Software Development Life Cycle consists of the following stages –

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party

modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

3.3 SDLC Models

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as "Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry:

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models

3.4 Model used in the project:

Iterative Model

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

3.4.1 Iterative Model - Design

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional

capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

The following illustration is a representation of the Iterative and Incremental model –

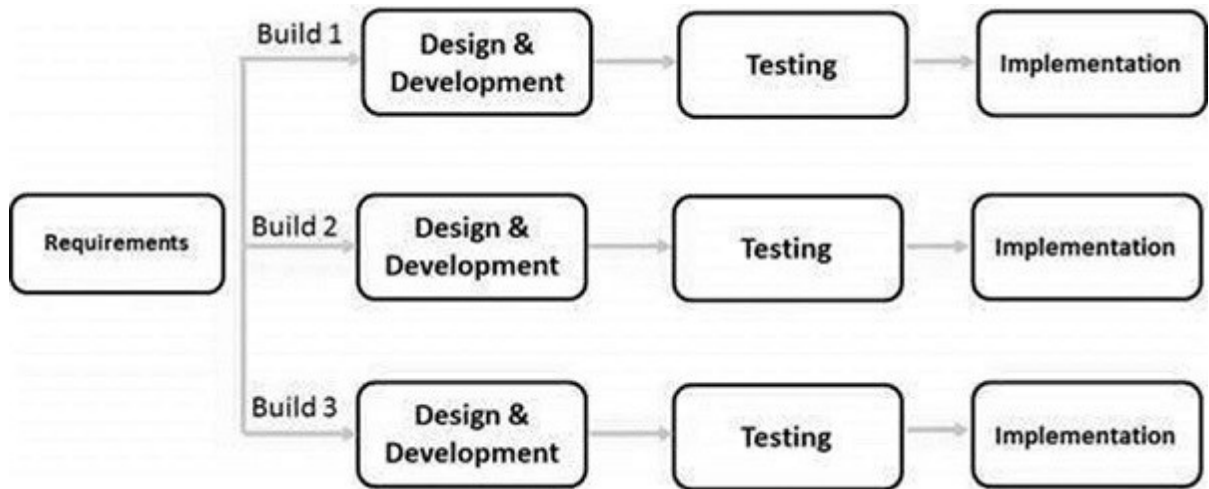


Fig. 3.4.1 Iterative Phases

Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software.

3.4.2 Iterative Model - Application

Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios –

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- There are some high-risk features and goals which may change in the future.

4. Modules

4.1 Home

4.2 Admission

4.3 Employment News

4.4 Result Sheets

4.4.1 Inputs

- Academic Session
- Degree Level
- Exam
- Course Name

4.5 Course Structure

4.5.1 Inputs

- Academic Session
- Degree Level
- Course Name

4.6 Exam Results

4.6.1 Input

- Enrollment Number

4.7 Student Login

4.7.1 Inputs

- Enrollment Number
- Enrollment Password
- Captcha

4.8 Employee Login

4.8.1 Inputs

- User Name
- Password
- Captcha

4.9 Contact Us

- Address
- Call

4.10 Website

5. Components Used in the Project

5.1 Volley

Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster.

Volley offers the following benefits:

- Automatic scheduling of network requests.
- Multiple concurrent network connections.
- Transparent disk and memory response caching with standard HTTP [cache coherence](#).
- Support for request prioritization.
- Cancellation request API. You can cancel a single request, or you can set blocks or scopes of requests to cancel.
- Ease of customization, for example, for retry and backoff.
- Strong ordering that makes it easy to correctly populate your UI with data fetched asynchronously from the network.
- Debugging and tracing tools.

Volley excels at RPC-type [Appendix] operations used to populate a UI, such as fetching a page of search results as structured data. It integrates easily with any protocol and comes out of the box with support for raw strings, images, and JSON. By providing built-in support for the features you need, Volley frees you from writing boilerplate code and allows you to concentrate on the logic that is specific to your app.

Volley is not suitable for large download or streaming operations, since Volley holds all responses in memory during parsing. For large download operations, consider using an alternative like [Download Manager](#).

The core Volley library is developed on [Git Hub](#) and contains the main request dispatch pipeline as well as a set of commonly applicable utilities, available in the Volley "toolbox." The easiest way to add Volley to your project is to add the following dependency to your app's build gradle file:

```
dependencies {  
  
    ...  
  
    compile 'com.android.volley:volley:1.0.0'  
  
}
```

You can also clone the Volley repository and set it as a library project:

1. Git clone the repository by typing the following at the command line:

2. **git clone <https://github.com/google/volley>**

3. Import the downloaded source into your app project.

5.2 Navigation Drawer

The navigation drawer is a panel that displays the app's main navigation options on the left edge of the screen. It is hidden most of the time, but is revealed when the user swipes a finger from the left edge of the screen or, while at the top level of the app, the user touches the app icon in the action bar.

5.2.1 Navigation Drawer Design

5.2.1.1 Create a Drawer Layout

To add a navigation drawer, declare your user interface with a [DrawerLayout](#) object as the root view of your layout. Inside the [DrawerLayout](#), add one view that contains the main content for the screen (your primary layout when the drawer is hidden) and another view that contains the contents of the navigation drawer.

For example, the following layout uses a [DrawerLayout](#) with two child views: a [FrameLayout](#) to contain the main content (populated by a [Fragment](#) at runtime), and a [ListView](#) for the navigation drawer.

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <!-- The main content view -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <!-- The navigation drawer -->
    <ListView android:id="@+id/left_drawer"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:choiceMode="singleChoice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp"
        android:background="#111"/>
</android.support.v4.widget.DrawerLayout>
```

This layout demonstrates some important layout characteristics:

- The main content view (the [FrameLayout](#) above) **must be the first child** in the [DrawerLayout](#) because the XML order implies z-ordering and the drawer must be on top of the content.

- The main content view is set to match the parent view's width and height, because it represents the entire UI when the navigation drawer is hidden.
- The drawer view (the ListView) **must specify its horizontal gravity** with the `android:layout_gravity` attribute. To support right-to-left (RTL) languages, specify the value with "start" instead of "left" (so the drawer appears on the right when the layout is RTL).
- The drawer view specifies its width in dp units and the height matches the parent view. The drawer width should be no more than 320dp so the user can always see a portion of the main content.

5.2.1.2 Initialize the Drawer List

In your activity, one of the first things to do is initialize the navigation drawer's list of items. How you do so depends on the content of your app, but a navigation drawer often consists of a ListView, so the list should be populated by an Adapter (such as ArrayAdapter or SimpleCursorAdapter).

For example, here's how you can initialize the navigation list with a [string array](#):

```
public class MainActivity extends Activity {
    private String[] mPlanetTitles;
    private DrawerLayout mDrawerLayout;
    private ListView mDrawerList;
    ...

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mPlanetTitles = getResources().getStringArray(R.array.planets_array);
    }
}
```



```

mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
mDrawerList = (ListView) findViewById(R.id.left_drawer);

// Set the adapter for the list view
mDrawerList.setAdapter(new ArrayAdapter<String>(this,
    R.layout.drawer_list_item, mPlanetTitles));
// Set the list's click listener
mDrawerList.setOnItemClickListener(new DrawerItemClickListener());

...
}
}

```

This code also calls `setOnItemClickListener()` to receive click events in the navigation drawer's list. The next section shows how to implement this interface and change the content view when the user selects an item.

5.2.1.3 Handle Navigation Click Events

When the user selects an item in the drawer's list, the system calls `onItemClick()` on the `OnItemClickListener` given to `setOnItemClickListener()`.

What you do in the `onItemClick()` method depends on how you've implemented your app structure. In the following example, selecting each item in the list inserts a different Fragment into the main content view (the `FrameLayout` element identified by the `R.id.content_frame` ID):

```

private class DrawerItemClickListener implements ListView.OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        selectItem(position);
    }
}

```

```

/** Swaps fragments in the main content view */
private void selectItem(int position) {
    // Create a new fragment and specify the planet to show based on position
    Fragment fragment = new PlanetFragment();
    Bundle args = new Bundle();
    args.putInt(PlanetFragment.ARG_PLANET_NUMBER, position);
    fragment.setArguments(args);

    // Insert the fragment by replacing any existing fragment
    FragmentManager fragmentManager = getFragmentManager();
    fragmentManager.beginTransaction()
        .replace(R.id.content_frame, fragment)
        .commit();

    // Highlight the selected item, update the title, and close the drawer
    mDrawerList.setItemChecked(position, true);
    setTitle(mPlanetTitles[position]);
    mDrawerLayout.closeDrawer(mDrawerList);
}

@Override
public void setTitle(CharSequence title) {
    mTitle = title;
    getActionBar().setTitle(mTitle);
}

```

5.2.1.4 Listen for Open and Close Events

To listen for drawer open and close events, call `setDrawerListener()` on your `DrawerLayout` and pass it an implementation of `DrawerLayout.DrawerListener`. This

interface provides callbacks for drawer events such as `onDrawerOpened()` and `onDrawerClosed()`.

However, rather than implementing the `DrawerLayout.DrawerListener`, if your activity includes the `actionbar`, you can instead extend the `ActionBarDrawerToggle` class. The `ActionBarDrawerToggle` implements `DrawerLayout.DrawerListener` so you can still override those callbacks, but it also facilitates the proper interaction behavior between the action bar icon and the navigation drawer (discussed further in the next section).

As discussed in the [Navigation Drawer](#) design guide, you should modify the contents of the action bar when the drawer is visible, such as to change the title and remove action items that are contextual to the main content. The following code shows how you can do so by overriding `DrawerLayout.DrawerListener` callback methods with an instance of the `ActionBarDrawerToggle` class:

```
public class MainActivity extends Activity {
    private DrawerLayout mDrawerLayout;
    private ActionBarDrawerToggle mDrawerToggle;
    private CharSequence mDrawerTitle;
    private CharSequence mTitle;
    ...

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ...

        mTitle = mDrawerTitle = getTitle();
        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout,
            R.string.drawer_open, R.string.drawer_close) {
```

```

    /** Called when a drawer has settled in a completely closed state. */
    public void onDrawerClosed(View view) {
        super.onDrawerClosed(view);
        getActionBar().setTitle(mTitle);
        invalidateOptionsMenu(); // creates call to onPrepareOptionsMenu()
    }

    /** Called when a drawer has settled in a completely open state. */
    public void onDrawerOpened(View drawerView) {
        super.onDrawerOpened(drawerView);
        getActionBar().setTitle(mDrawerTitle);
        invalidateOptionsMenu(); // creates call to onPrepareOptionsMenu()
    }
};

// Set the drawer toggle as the DrawerListener
mDrawerLayout.setDrawerListener(mDrawerToggle);
}

/* Called whenever we call invalidateOptionsMenu() */
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    // If the nav drawer is open, hide action items related to the content view
    boolean drawerOpen = mDrawerLayout.isDrawerOpen(mDrawerList);
    menu.findItem(R.id.action_websearch).setVisible(!drawerOpen);
    return super.onPrepareOptionsMenu(menu);
}
}

```

The next section describes the [ActionBarDrawerToggle](#) constructor arguments and the other steps required to set it up to handle interaction with the action bar icon.

5.2.1.5 Open and Close with the App Icon

Users can open and close the navigation drawer with a swipe gesture from or towards the left edge of the screen, but if you're using the [action bar](#), you should also allow users to open and close it by touching the app icon. And the app icon should also indicate the presence of the navigation drawer with a special icon. You can implement all this behavior by using the [ActionBarDrawerToggle](#) shown in the previous section.

To make [ActionBarDrawerToggle](#) work, create an instance of it with its constructor, which requires the following arguments:

- The [Activity](#) hosting the drawer.
- The [DrawerLayout](#).
- A drawable resource to use as the drawer indicator.
- A String resource to describe the "open drawer" action (for accessibility).
- A String resource to describe the "close drawer" action (for accessibility).

Then, whether or not you've created a subclass of [ActionBarDrawerToggle](#) as your drawer listener, you need to call upon your [ActionBarDrawerToggle](#) in a few places throughout your activity lifecycle:

```
public class MainActivity extends Activity {  
    private DrawerLayout mDrawerLayout;  
    private ActionBarDrawerToggle mDrawerToggle;  
    ...  
  
    public void onCreate(Bundle savedInstanceState) {  
        ...  
    }  
}
```

```

mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
mDrawerToggle = new ActionBarDrawerToggle(
    this,          /* host Activity */
    mDrawerLayout, /* DrawerLayout object */
    R.string.drawer_open,  /* "open drawer" description */
    R.string.drawer_close /* "close drawer" description */
) {

    /** Called when a drawer has settled in a completely closed state. */
    public void onDrawerClosed(View view) {
        super.onDrawerClosed(view);
        getActionBar().setTitle(mTitle);
    }

    /** Called when a drawer has settled in a completely open state. */
    public void onDrawerOpened(View drawerView) {
        super.onDrawerOpened(drawerView);
        getActionBar().setTitle(mDrawerTitle);
    }
};

// Set the drawer toggle as the DrawerListener
mDrawerLayout.setDrawerListener(mDrawerToggle);

getActionBar().setDisplayHomeAsUpEnabled(true);
getActionBar().setHomeButtonEnabled(true);
}

@Override
protected void onCreate(Bundle savedInstanceState) {

```

```

        super.onCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has occurred.
        mDrawerToggle.syncState();
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        mDrawerToggle.onConfigurationChanged(newConfig);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Pass the event to ActionBarDrawerToggle, if it returns
        // true, then it has handled the app icon touch event
        if (mDrawerToggle.onOptionsItemSelected(item)) {
            return true;
        }
        // Handle your other action bar items...

        return super.onOptionsItemSelected(item);
    }

    ...
}

```

5.3 Recycler View

Many apps need to display user-interface elements based on large data sets, or data that frequently changes. For example, a music app might need to display information about thousands of albums, but only a dozen of those albums might be on-screen at a time. If

the app created UI widgets for each of those albums, the app would end up using a lot of memory and storage, potentially making the app slow and crash-prone. On the other hand, if the app created UI widgets each time a new album scrolled onto the screen and destroyed the widgets when it scrolled off, that would also cause the app to run slowly, since creating UI objects is a resource-intensive operation.

To address this common situation, the Android Support Library provides the RecyclerView suite of objects. RecyclerView and its associated classes and interfaces help you to design and implement a dynamic user interface that runs efficiently. You can use these classes as they are, or customize them to suit your specific needs.

5.3.1 Dynamic View Structure

Under the RecyclerView model, several different components work together to display your data. Some of these components can be used in their unmodified form; for example, your app is likely to use the RecyclerView class directly. In other cases, we provide an abstract class, and your app is expected to extend it; for example, every app that uses RecyclerView needs to define its own view holder, which it does by extending the abstract RecyclerView.ViewHolder class.

The overall container for your dynamic user interface is a RecyclerView object. You add this object to your activity's or fragment's layout; the RecyclerView, in turn, fills itself with smaller views representing the individual items. The RecyclerView uses the layout manager you provide to arrange the items. You can use one of our standard layout managers (such as LinearLayoutManager or GridLayoutManager), or implement your own.

The individual items are represented by view holder objects. These objects are instances of the class you define by extending RecyclerView.ViewHolder. Each view holder is in charge of displaying a single item, and has its own view. For example, if a RecyclerView is used to display a user's music collection, each view holder might represent a single album. In this case, the view holder's view might contain elements to display the album's title and artwork, and might respond to clicks by playing or pausing that album. The RecyclerView creates only as many view holders as are needed to display the on-screen

portion of the dynamic content, plus a few extra. As the user scrolls through the list, the RecyclerView takes the off-screen views and rebinds them to the data which is scrolling onto the screen.

The view holder objects are managed by an adapter, which you create by extending the RecyclerView.Adapter abstract class. The adapter creates view holders as needed. The adapter also binds the view holders to their data. It does this by assigning the view holder to a position, and calling the adapter's onBindViewHolder() method. That method uses the view holder's position to determine what the contents should be. For example, in the case of the music collection, the adapter might call onBindViewHolder() to assign a particular view holder to position 420. The method would find out which album is in that spot in the list, and fill in the view holder's widgets with the appropriate artwork and title.

The RecyclerView model does a lot of optimization work so you don't have to:

When the view is first populated, it creates and binds some view holders on either side of the list. For example, if the view is displaying albums 0 through 9, the RecyclerView creates and binds those view holders, and might also create and bind the view holder for position 10. That way, if the user scrolls the list, the next element is ready to display.

As the user scrolls the list, the RecyclerView creates new view holders as necessary. It also saves the view holders which have scrolled off-screen, so they can be reused. If the user switches the direction they were scrolling, the view holders which were scrolled off the screen can be brought right back. On the other hand, if the user keeps scrolling in the same direction, the view holders which have been off-screen the longest can be rebound to new data. The view holder does not need to be created or have its view inflated; instead, the app just updates the view's contents to match the new item it was bound to.

When the displayed items change, the app notifies the adapter by calling an appropriate RecyclerView.Adapter.notify...() method. The adapter's built-in code then rebinds just the affected items.

5.4 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost.

Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user reengagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.

5.4.1 Key capabilities

Send notification messages or data messages	Send notification messages that are displayed to your user. Or send data messages and determine completely what happens in your application code. See Message types .
Versatile message targeting	Distribute messages to your client app in any of three ways — to single devices, to groups of devices, or to devices subscribed to topics.
Send messages from client apps	Send acknowledgments, chats, and other messages from devices back to your server over FCM's reliable and battery-efficient connection channel.

5.4.2 Working

An FCM implementation includes two main components for sending and receiving:

- A trusted environment such as Cloud Functions for Firebase or an app server on which to build, target and send messages.
- An iOS, Android, or Web (JavaScript) client app that receives messages.

You can send messages via the Admin SDK or the HTTP and XMPP APIs. For testing or for sending marketing or engagement messages with powerful built-in targeting and analytics, you can also use the Notifications composer.

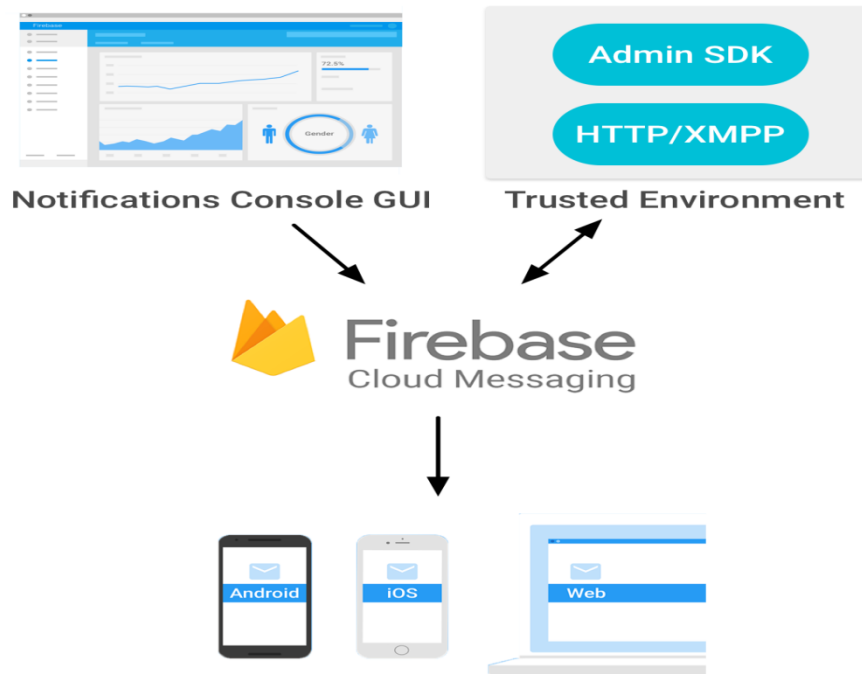


Fig 5.4.1 Working of Firebase

5.5 Card View

CardView extends the FrameLayout class and lets you show information inside cards that have a consistent look across the platform. CardView widgets can have shadows and rounded corners.

To create a card with a shadow, use the `card_view:cardElevation` attribute. CardView uses real elevation and dynamic shadows on Android 5.0 (API level 21) and above and falls back to a programmatic shadow implementation on earlier versions. For more information, see Maintaining Compatibility.

Use these properties to customize the appearance of the CardView widget:

- To set the corner radius in your layouts, use the `card_view:cardCornerRadius` attribute.

- To set the corner radius in your code, use the `CardView.setRadius` method.
- To set the background color of a card, use the `card_view:cardBackgroundColor` attribute.

The following code example shows you how to include a CardView widget in your layout:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    ... >

    <!-- A CardView that contains a TextView -->
    <android.support.v7.widget.CardView
        xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:id="@+id/card_view"
        android:layout_gravity="center"
        android:layout_width="200dp"
        android:layout_height="200dp"
        card_view:cardCornerRadius="4dp">

        <TextView
            android:id="@+id/info_text"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </android.support.v7.widget.CardView>
</LinearLayout>
```

5.5.1 Add Dependencies

The RecyclerView and CardView widgets are part of the v7 Support Libraries. To use these widgets in your project, add these Gradle dependencies to your app's module:

```
dependencies {  
    ...  
    compile 'com.android.support:cardview-v7:21.0.+'  
    compile 'com.android.support:recyclerview-v7:21.0.+'  
}
```

5.6 Web View

If you want to deliver a web application (or just a web page) as a part of a client application, you can do it using `WebView`. The `WebView` class is an extension of Android's `View` class that allows you to display web pages as a part of your activity layout. It does not include any features of a fully developed web browser, such as navigation controls or an address bar. All that `WebView` does, by default, is show a web page.

A common scenario in which using `WebView` is helpful is when you want to provide information in your application that you might need to update, such as an end-user agreement or a user guide. Within your Android application, you can create an Activity that contains a `WebView`, then use that to display your document that's hosted online.

Another scenario in which `WebView` can help is if your application provides data to the user that always requires an Internet connection to retrieve data, such as email. In this case, you might find that it's easier to build a `WebView` in your Android application that shows a web page with all the user data, rather than performing a network request, then parsing the data and rendering it in an Android layout. Instead, you can design a web page that's tailored for Android devices and then implement a `WebView` in your Android application that loads the web page.

5.6.1 Adding a `WebView` to Your Application

To add a [WebView](#) to your Application, simply include the `<WebView>` element in your activity layout. For example, here's a layout file in which the [WebView](#) fills the screen:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

To load a web page in the [WebView](#), use [loadUrl\(\)](#). For example:




```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```

Before this will work, however, your application must have access to the Internet. To get Internet access, request the [INTERNET](#) permission in your manifest file. For example:

```
<manifest ... >
    <uses-permission android:name="android.permission.INTERNET" />
    ...
</manifest>
```

6. Output

6.1 Home Page

 **MANUU**  

MASTER OF ARTS IN ISLAMIC STUDIES FIRST SEMESTER (SRINAGAR)
Published Date: 2017-08-10 19:45:53.623

MASTER OF EDUCATION FIRST SEMESTER (SRINAGAR)
Published Date: 2017-08-10 12:13:18.740

BACHELOR OF EDUCATION (B.ED.) SRINAGAR FIRST SEMESTER
Published Date: 2017-08-10 10:49:03.067

B. SC LIFE SCIENCES (Z.B.C) SECOND SEMESTER
Published Date: 2017-08-07 12:01:10.540



B. SC. PHYSICAL SCIENCES (M.P.C) SECOND SEMESTER
Published Date: 2017-08-07 11:59:50.823


B. SC. PHYSICAL SCIENCES (M.P.CS) SECOND SEMESTER
Published Date: 2017-08-07 11:58:17.863

B. A. SECOND SEMESTER 2017 RESULT DECLARED
Published Date: 2017-08-07 02:25:45.660


B COM Second Semester 2017 Result Declared
Published Date: 2017-08-07 02:24:47.537

6.2 Result Sheets



مولانا آزاد نیشنل اردو یونیورسٹی
MAULANA AZAD NATIONAL URDU UNIVERSITY
(A Central University established by an Act of Parliament in 1998)
(Accredited with 'A' Grade by NAAC)




Academic Session:	2016 - 2017	▼
Degree Level:	Bachelors	▼
Exam :	End Semester Examination May-2017	▼
Course Name :	B. TECH. COMPUTER ENGINEERING SEM 2	▼

VIEW

RESET

© 2017 Maulana Azad National Urdu University.

6.3 Result Sheet

<div>  <div> MAULANA AZAD NATIONAL URDU UNIVERSITY <small>(A Central University established by an Act of Parliament in 1998) (Accredited with 'A' Grade by NAAC)</small> </div> </div>			
<div> RESULT SHEET B. TECH. COMPUTER ENGINEERING SECOND SEMESTER End Semester Examination May-2017 </div>			
Roll Number	NAME OF THE CANDIDATES	ENROLLMENT	RESULT / REMARK
16BTCS001HY	ABDULLAH	A165035	PASSED
16BTCS002HY	AHMAD FARAZ	A165056	PASSED
16BTCS003HY	AHMAD HUSAIN	A165065	PASSED
16BTCS004HY	AJAZ AHMED	A165057	PASSED
16BTCS005HY	AMEER AZAM	A165079	PASSED
16BTCS006HY	AMIR SUHEL	A165075	PASSED
16BTCS007HY	ASFAHAN AHMAD	A165037	FAILED IN BTCS150BSP- [TM UEP IAP]; BTCS150ESP- [TM UEP IAP]; BTCS150PCP-[UEP IAP]; PROMOTED
16BTCS008HY	ATHAR PARVEZ	A165064	PASSED
16BTCS009HY	FAIZANE RASOOL	A165042	PASSED
16BTCS010HY	FAIZUN NABI	A165045	PASSED
16BTCS011HY	HAROON RASHID	A166427	PASSED
16BTCS012HY	IBTESHAM KALAM	A165058	PASSED
16BTCS013HY	IMTIYAZ AHMED	A165054	PASSED

6.4 Share Result Sheet



MAULANA AZAD NATIONAL URDU UNIVERSITY
(A Central University established by an Act of Parliament in 1998)
 (Accredited with 'A' Grade by NAAC)

RESULT SHEET
B. TECH. COMPUTER ENGINEERING SECOND

Share using

Tap an icon below to share your content directly.



Saiful



B.E. Techian



Raja Bhandari MSIT



Aftab



WhatsApp



hike



Facebook



Facebook



Messages Truecaller



Messages



Direct Message

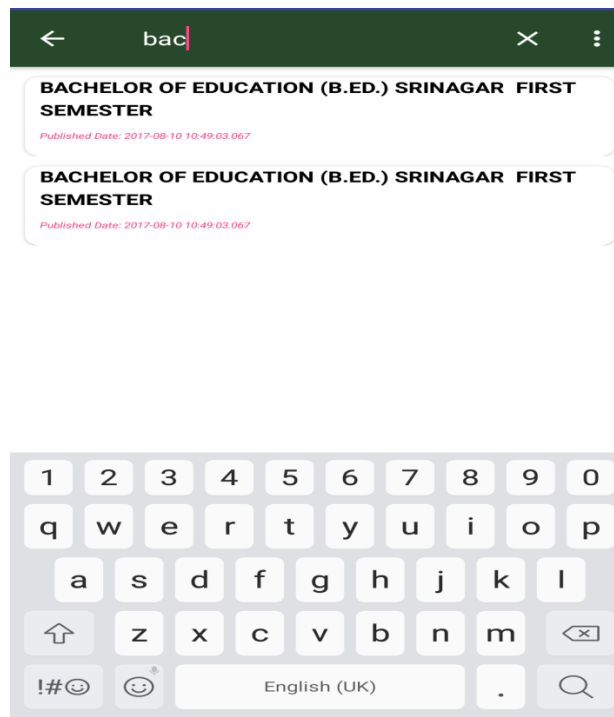


Tweet

6.5 Course Structure



6.10 Search View



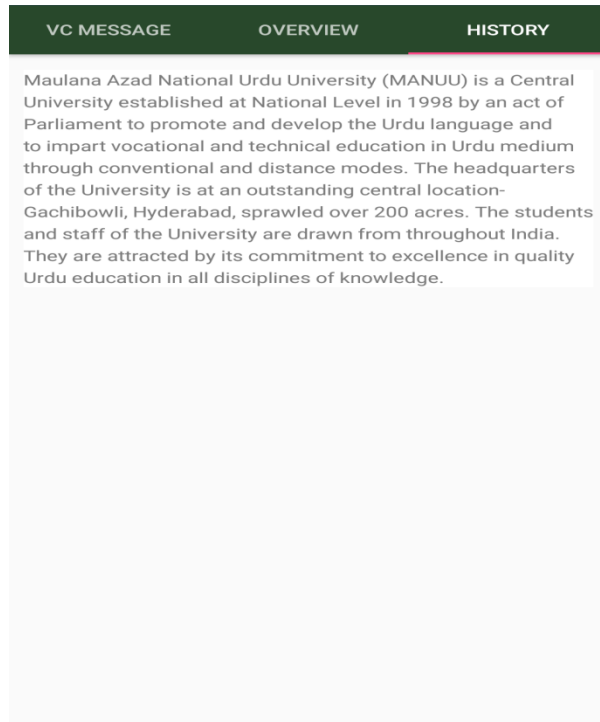
6.11 VC Message

VC MESSAGE	OVERVIEW	HISTORY
<p>Dear friends, It is a feeling of joy to be with you. Maulana Azad National Urdu University, equipped with the mandate to quench the thirst of knowledge seekers through Urdu Medium, is a unique institution in real sense. MANUU has undergone massive changes in the recent years. Now the University campus is dotted with new well designed buildings that are host to a large number of newly created departments, Centres and Civil Services Examination Coaching Academy. There are also new hostel buildings for the boys and girls besides a well equipped indoor stadium. The way forward from here is to focus on reaching out to the unreached; to consolidate what has been built; to create good infrastructure at our off campuses; to improve the quality of education; to increase the knowledge base in Urdu and; to create conducive environment for quality research. MANUU is a unique university which imparts education to its students through Urdu medium. This characteristic of MANUU presents us with monumental challenges. When MANUU began functioning in 1998 its primary route to offer education was through distance education mode. It remained so for the next few years and the number of students registered with the University swelled to about one lakh at one point of time. But somehow the attention to distance education mode got diluted. MANUU has taken a decision recently to work on distance education mode seriously and clear all the hurdles in its way. Right now there are thousands of students enlisted with the distance education mode, more than half of them being female. It has been decided to plug the lacuna in the system and tighten the monitoring mechanism at all the 160 Study Centres; nine Regional and five Sub-Regional Centres with the help of latest</p>		

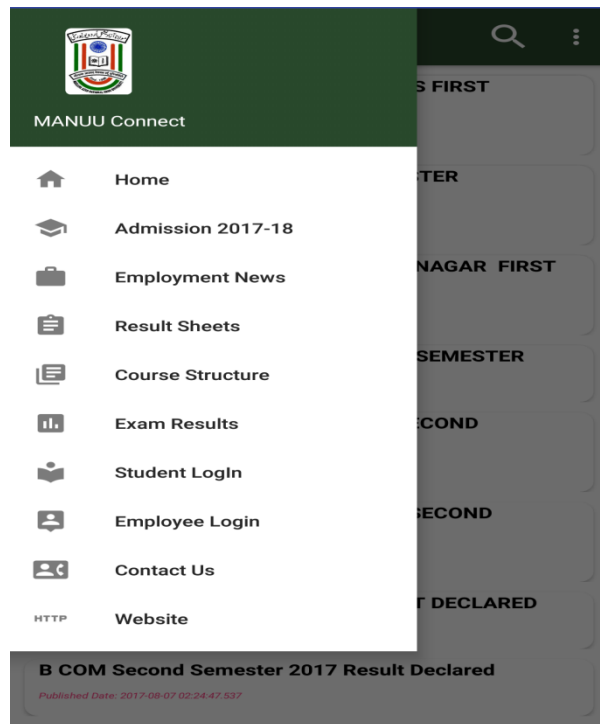
6.12 Overview

VC MESSAGE	OVERVIEW	HISTORY
<p>Maulana Azad National Urdu University (MANUU) is a Central University established at National Level in 1998 by an act of Parliament to promote and develop the Urdu language and to impart vocational and technical Education in Urdu medium through conventional and distance modes. The Headquarters of the University is at an outstanding central location-Gachibowli, Hyderabad, sprawled over 200 acres. The students and staff of the University are drawn from throughout India. They are attracted by its commitment to excellence in quality Urdu education in all disciplines of knowledge. The objectives of the University are as follows: to promote and develop the Urdu language to impart education and training in vocational and technical subjects through the medium of Urdu to provide wider access to people desirous of pursuing programmes of higher education and training in Urdu medium through Campus and Distance modes and to provide focus on women education. The University's campus education as on today boasts of seven Schools of Studies: namely, 1) School of Languages, Linguistics and Indology, 2) School of Commerce and Business Management, 3) School of Journalism and Mass Communication 4) School of Arts and Social Sciences 5) School of Sciences 6) School of Education and Training and 7) School of Computer Science Information Technology. These seven Schools run 24 Departments, which in addition to offering masters programmes, provide research programmes at M. Phil. and Ph D level. These Schools have already churned out a considerable number of M. Phil. Scholars. The focus of all these Schools is to search new areas of knowledge and to advance knowledge through research and application. The University has established three Industrial Training Institutes</p>		

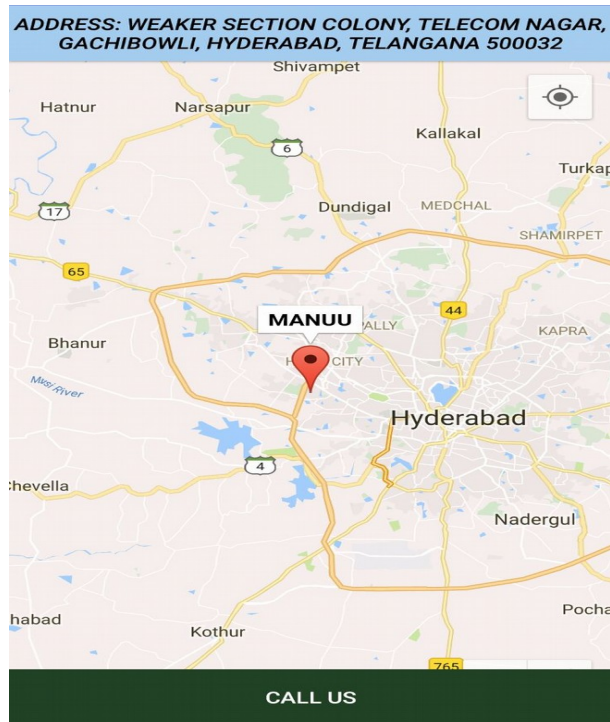
6.13 History



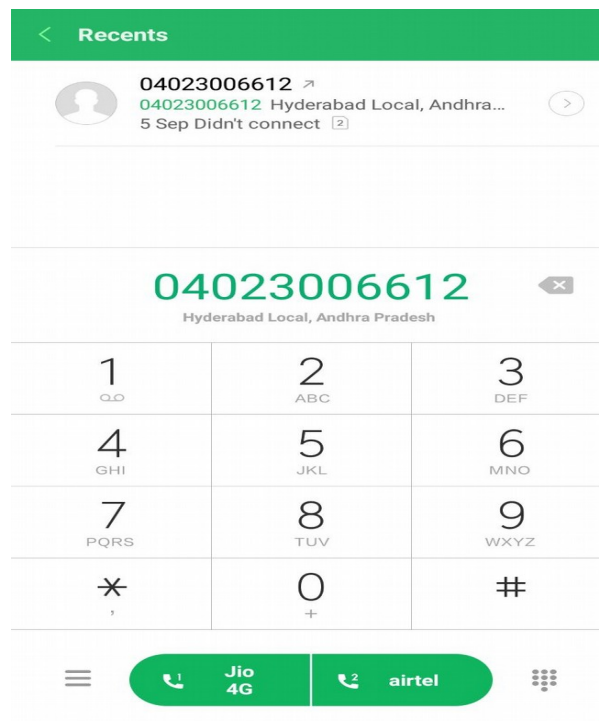
6.14 Navigation Drawer



6.15 Contact Us



6.16 Call Us



BIBLIOGRAPHY

- About Android from [https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))
- About Android Components from
https://www.tutorialspoint.com/android/android_application_components.htm
- About Volley from
<https://developer.android.com/training/volley/index.html>
- About Navigation Drawer from
<https://developer.android.com/training/implementing-navigation/nav-drawer.html>
- About Firebase cloud messaging from
<https://firebase.google.com/docs/cloud-messaging/>
- About Software Development Life Cycle from Software Engineering from K.K Aggarwal
- About Card View from
<https://developer.android.com/training/material/lists-cards.html>
- About Web View from
<https://developer.android.com/guide/webapps/webview.html#AddingWebView>

Appendix

Heading	Description	Page No.
AOSP Browser	AOSP browser is an unbranded browser using either Webkit or Blink as its rendering engine, which predated the availability of Chrome on Android, and can still be included in place of Chrome on devices, or in addition to Chrome on devices.	7
UMTS	UMTS (Universal Mobile Telecommunications Service) is a third-generation (3G) broadband, packet-based transmission of text, digitized voice, video, and multimedia at data rates up to 2 megabits per second (Mbps).	8
GSM	Global System for Mobile communication	9
LTE	Long-Term Evolution (LTE) is a standard for high-speed wireless communication for mobile devices	9
CDMA	Code Division Multiple Access	9
EV-DO	Evolution Data Only/Evolution Data Optimized, is a 3G mobile broadband technology	9
UMTS	The Universal Mobile Telecommunications System (UMTS) is a third generation mobile cellular system for networks based on the GSM standard.	9
NFC	Near-field communication (NFC) is a set of communication protocols that enable two electronic devices, one of which is usually a portable device such as a Smartphone, to establish communication by bringing them within 4 cm of each other	9
IDEN	Integrated Digital Enhanced Network (iDEN) is a mobile telecommunications technology, developed by Motorola	9
A2DP	A2DP stands for Advanced Audio Distribution Profile. This is the Bluetooth Stereo profile which defines how high quality stereo audio can be streamed from one device to another over a Bluetooth connection -	9
AVRCP	Audio/Video Remote Control Profile* is a Bluetooth profile that allows Bluetooth devices to control media playback on remote devices	9
RTP	The Real-time Transport Protocol (RTP) is a network protocol for	9

	delivering audio and video over IP networks	
WebM	WebM is an open, royalty-free, media file format designed for the web.	10
H.263	H.263 is a video compression standard originally designed as a low-bit-rate compressed format for videoconferencing.	10
H.264	H.264 or MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC) is a block-oriented motion-compensation-based video compression standard	10
AAC	Advanced Audio Coding (AAC) is a proprietary audio coding standard for lossy digital audio compression	10
AMR	The Adaptive Multi-Rate (AMR or AMR-NB or GSM-AMR) audio codec is an audio compression format optimized for speech coding.	10
MIDI	Musical Instrument Digital Interface) is a technical standard that describes a communications protocol, digital interface and electrical connectors	10
Dalvik	Dalvik is a part of the software stack that makes up the Android platform.	10
RPC	Remote Procedure Calls (RPC) is a powerful technique for constructing distributed, client-server based applications.	28