**Information and Communication Technology (ICT) in Agriculture**

**ACKNOWLEDGEMENT**

**ABSTRACT**

Information and Communication Technology (ICT) in agriculture is an emerging field focusing on the enhancement of agricultural and rural development in India. It involves innovative applications using ICT in the rural domain. The advancement of ICT can be utilized for providing accurate and timely relevant information and services to the farmers, thereby facilitating an environment for remunerative agriculture. This paper describes a mobile based application for farmers which would help them in their farming activities. We propose an android based mobile application which would take care of the updates of the different agricultural commodities, weather forecast updates, agricultural news updates. The application has been designed taking Indian farming in consideration.

**Organization Profile**

### LIST OF ABBREVIATIONS

**ADO -** Active data Objects

**SQL -** Standard Query Language

**ASP -** Active Server Page

**IIS -** Internet Information Services

**CLR -** Common Language Runtime

**IL** - Intermediate Language

**XML** - Extended Markup Language

**ISP -** Internet Service Provider

**VLSI -** Very Large Scale Integration

**MSIDE** - Microsoft Integrated Development Environment

**NGWS -** Next Generation Window Service

**CHAPTER 1- INTRODUCTION**

**1.1 Overview**

Agriculture in India is the means of livelihood of almost two third of the workforce in the country. It has always been India’s most important economic sector. Agriculture may be defined as an integrated system of techniques to control the

growth and harvesting of animal and vegetables. It is an uncomplicated endeavor comprising of technical and practical processes that helps in the maintenance of the ecological balance and protects human resources; most importantly it is a viable food production system. The agricultural sector is critically important in any developing economy and so it is in India, where it contributes close to 20% of GDP. Here 60% of the population depends on agriculture, either directly or indirectly. Small-scale producers, who make up the vast majority of Indian farmers, are often unable to access information that could increase yield and lead to better prices for their crops. The rapid growth of mobile telephony and the recent introduction of mobile enabled information services provide a means to overcome existing information asymmetry. It also helps to bridge the gap between the availability and delivery of agriculture inputs and agriculture infrastructure. The increasing penetration of mobile networks and handsets in India therefore present an opportunity to make useful information more widely available. This could help agricultural markets operate more efficiently, and overcome some of the other challenges faced by this sector. Introduction of Information and Communication Technology in Indian agriculture enables the dissemination of requisite information at the right time. This revolution in information technology has made access to relevant information easy and cost-effective. The mass is surprised by the rapid emergence of mobile telephony and consider this connected world to be the virtue of mobile devices. There are so many possible applications arising, but as usual, the challenge is to understand the right place and role of the technology in social, economic, educational interactions.

**1.2 Objective of the project**

The main objective for such project is to develop a mobile phone based solution that helps in farm’s management, leads to agricultural yield improvement and helps in care/maintenance of the farms..

**1.3 Organization of Chapters**

In Chapter 1 we introduce about the project concept and give an overview idea about the project. In Chapter 2, we discuss about the project domain and the detailed description of existing systems by analysis the literature survey of the existing techniques. We also then presented about the techniques and methods of our proposed methods. In our proposed method we also listed out the advantages of using our proposed method. Then we presented the differences between the existing system and proposed system as a tabular representation stating the advantages of our proposed system. In Chapter 3, we made a system analysis of the methods we propose. In Chapter 4, we listed the Hardware requirements and Software Requirements of our project. In Chapter 5, we presented the modules and their description. Then we also depicted the Use-case diagram of our project, then we depicted Class diagram of our project. In Chapter 6, we concluded our proposal and then in Chapter 7 we list out our references made for our proposed method.

**CHAPTER 2- LITERATURE SURVEY**

A literature review is much more than a list of separate reviews of articles and books. They are common and very important in the sciences. A literature review is a critical, analytical summary and synthesis of the current knowledge of a topic. It should compare and relate different theories, findings, and so on, rather than just summarize them individually. It should also have a particular focus or theme to organize the review. It does not have to be an exhaustive account of everything published on the topic. But it should discuss all the more significant academic literature important for that focus

**CHAPTER 3- SYSTEM ANALYSIS**

**EXISTING SYSTEM:**

The existing system uses all the manual work, where the farmers should go physically and search for the Agro shop. Analysing the weather is a big task for farmers. Finding the exact rate and soil cultivation is also a big issue to the farmers. So to solve this problem we came up with a solution that is android application on agriculture.

**PROPOSED SYSTEM:**

Weather is one of the most crucial features for the farmers. They are usually concerned about the changes in weather and other details related to the shifts in the surroundings. It helps in observing the temperature, dew factor, dryness and other

minute details about the weather and forecast for next 5 days. One of the biggest challenges that each farmer faces is decisions related to marketing his grain and how those decisions will affect his bottom line. This application for agriculture enables the farmer to calculate profitability based on where the grain markets are currently trading and to see how higher or lower grain markets are presently. They would be able to get the current market prices depending upon the commodities. It should carry grain and livestock prices from major Indian agricultural market. We have the option of type of soil with cultivation detail. We can search nearest agro shop and we can make direct call to agro department. The application which we are proposing is much easier to understand by user who is familiar with the use of mobile.The application is with very basic options available, but the application has lots of options that can be enhanced in future.

**Literature Review**

**Internet of Things in agriculture, recent advances and future challenges**

The increasing demand for food, both in terms of quantity and quality, has raised the need for intensification and industrialisation of the agricultural sector. The “Internet of Things” (IoT) is a highly promising family of technologies which is capable of offering many solutions towards the modernisation of agriculture. Scientific groups and research institutions, as well as the industry, are in a race trying to deliver more and more IoT products to the agricultural business stakeholders, and, eventually, lay the foundations to have a clear role when IoT becomes a mainstream technology. At the same time Cloud Computing, which is already very popular, and Fog Computing provide sufficient resources and solutions to sustain, store and analyse the huge amounts of data generated by IoT devices.

**An overview of internet of things (IoT) and data analytics in agriculture: Benefits and challenges**

The surge in global population is compelling a shift toward smart agriculture practices. This coupled with the diminishing natural resources, limited availability of arable land, increase in unpredictable weather conditions makes food security a major concern for most countries. As a result, the use of Internet of Things (IoT) and data analytics (DA) are employed to enhance the operational efficiency and productivity in the agriculture sector. There is a paradigm shift from use of wireless sensor network (WSN) as a major driver of smart agriculture to the use of IoT and DA. The IoT integrates several existing technologies, such as WSN, radio frequency identification, cloud computing, middleware systems, and end-user applications. In this paper, several benefits and challenges of IoT have been identified. We present the IoT ecosystem and how the combination of IoT and DA is enabling smart agriculture. Furthermore, we provide future trends and opportunities which are categorized into technological innovations, application scenarios, business, and marketability.

**Evolution of internet of things (IoT) and its significant impact in the field of precision agriculture**

During recent years, one of the most familiar name scaling new heights and creating a benchmark is Internet of Things (IoT). It is indeed the future of communication that has transformed Things (Objects) of the real world into smarter devices. The functional aspect of IoT is to unite every object of the world in such a manner that humans have the ability to control them via Internet. Furthermore, these objects also provide regular as well as timely updates on their current status to its end user. Although IoT concepts were proposed a couple of years ago, it may not be incorrect to quote that this term has become a benchmark for establishing communication among objects. In context to the present standings of IoT, identification of the most prominent applications in the field of IoT have been highlighted and a comprehensive review has been done specifically in the field of Precision Agriculture.

**State-of-the-art internet of things in protected agriculture**

The Internet of Things (IoT) has tremendous success in health care, smart city, industrial production and so on. Protected agriculture is one of the fields which has broad application prospects of IoT. Protected agriculture is a mode of highly efficient development of modern agriculture that uses artificial techniques to change climatic factors such as temperature, to create environmental conditions suitable for the growth of animals and plants. This review aims to gain insight into the state-of-the-art of IoT applications in protected agriculture and to identify the system structure and key technologies. Therefore, we completed a systematic literature review of IoT research and deployments in protected agriculture over the past 10 years and evaluated the contributions made by different academicians and organizations. Selected references were clustered into three application domains corresponding to plant management, animal farming and food/agricultural product supply traceability.

**Agriculture IoT: Emerging trends, cooperation networks, and outlook**

The arrival of the IoT era has been revolutionizing various fields of our current world. Precision agriculture is recognized as one sustainable, eco-friendly, and profitable mode to improve agriculture yields and quality, and will ultimately come true with the further implementation of IoT techniques in agriculture. To facilitate the implementation, we make a visualization review of the agriculture IoT literature in the last decade, using records of 3168 documents and their 100,205 references in Web of Science. The dynamics of research fronts and intellectual bases bring out emerging trends in both applied IoT techniques and topics of concern in agriculture. Based on the quantity of contributions in the cooperation networks, outstanding countries, institutions, and authors are detected. Moreover, influential studies and scholars are recognized from the citation networks, indicating hot research and trends in the agriculture IoT literature from 2009 to 2018.

**SYSTEM STUDY**

**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

**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.

### 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.

**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.

**CHAPTER 4- SYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENT**

CPU type : Intel Pentium 4

Clock speed : 3.0 GHz

Ram size : 512 MB

Hard disk capacity : 40 GB

Monitor type : 15 Inch color monitor

Keyboard type : internet keyboard

Mobile : ANDROID MOBILE

**SOFTWARE REQUIREMENT**

Operating System: Android

Language : ANDROID SDK 2.3

Documentation : Ms-Office

**CHAPTER 5- MODULE DESCRIPTION**

**SYSTEM MODELS**

* Weather Module
* Cultivate and Rate Module
* Agro Shop Module
* Free Call Module

**MODULE DESCIRPTION:**

**Weather Module**

Weather is one of the most crucial features for the farmers. They are usually concerned about the changes in weather and other details related to the shifts in the surroundings. It helps in observing the temperature, dew factor, dryness and other

minute details about the weather and forecast for next 5 days. One of the biggest challenges that each farmer faces is decisions related to marketing his grain and how those decisions will affect his bottom line. This application for agriculture enables the farmer to calculate profitability based on where the grain markets are currently trading and to see how higher or lower grain markets are presently. They would be able to get the current market prices depending upon the commodities. It should carry grain and livestock prices from major Indian agricultural market.

**Cultivate and Rate Module**

This module help former to cultivate according with the soil. Also it is helpful to the current rate of the cultivation.

**Agro Shop Module**

This Module is used to find the neared agriculture shop.

**Free Call**

This module is used to make free call to the agriculture department…

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user

will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**SOFTWARE ENVIRONMENT**

Android is a [software stack](http://en.wikipedia.org/wiki/Solution_stack) for [mobile devices](http://en.wikipedia.org/wiki/Mobile_devices) that includes an [operating system](http://en.wikipedia.org/wiki/Operating_system), [middleware](http://en.wikipedia.org/wiki/Middleware) and key [applications](http://en.wikipedia.org/wiki/Application_software). [Google Inc.](http://en.wikipedia.org/wiki/Google) purchased the initial developer of the software, Android Inc., in 2005.

Android's [mobile operating system](http://en.wikipedia.org/wiki/Mobile_operating_system) is based on the [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel). Google and other members of the [Open Handset Alliance](http://en.wikipedia.org/wiki/Open_Handset_Alliance) collaborated on Android's development and release.

The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling [Smartphone](http://en.wikipedia.org/wiki/Smartphone) platform.[

The [Android SDK](http://developer.android.com/sdk/index.html) provides the tools and APIs necessary to begin developing applications Android platform using the Java programming language. Android has a large community of developers writing [applications](http://en.wikipedia.org/wiki/Application_software) ("apps") that extend the functionality of the devices. There are currently over 250,000 apps available for Android.

.Features

* **Application framework** enabling reuse and replacement of components
* **Dalvik virtual machine** optimized for mobile devices
* **Integrated browser** based on the open source [WebKit](http://webkit.org/) engine
* **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
* **SQLite** for structured data storage
* **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
* **GSM Telephony** (hardware dependent)
* **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
* **Camera, GPS, compass, and accelerometer** (hardware dependent)
* **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

## Android Architecture



## 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

## 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.

## 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.

The Linux kernel is an operating system [kernel](http://en.wikipedia.org/wiki/Kernel_%28computing%29) used by the [Linux](http://en.wikipedia.org/wiki/Linux) family of [Unix-like](http://en.wikipedia.org/wiki/Unix-like) [operating systems](http://en.wikipedia.org/wiki/Operating_system). It is one of the most prominent examples of [free and open source software](http://en.wikipedia.org/wiki/Free_and_open_source_software).

The Linux kernel is released under the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License) version 2 (GPLv2), (plus some [firmware images](http://en.wikipedia.org/wiki/Firmware) with various licenses), and is developed by contributors worldwide. Day-to-day development takes place on the [Linux kernel mailing list](http://en.wikipedia.org/wiki/Linux_kernel_mailing_list).

The Linux kernel was initially conceived and created by [Finnish](http://en.wikipedia.org/wiki/Finns) [computer science](http://en.wikipedia.org/wiki/Computer_science) student[Linus Torvalds](http://en.wikipedia.org/wiki/Linus_Torvalds) in 1991. Linux rapidly accumulated developers and users who adapted code from other [free software](http://en.wikipedia.org/wiki/Free_software) projects for use with the new operating system. The Linux kernel has received contributions from thousands of programmers.[[10]](http://en.wikipedia.org/wiki/Linux_kernel#cite_note-9) Many [Linux distributions](http://en.wikipedia.org/wiki/Linux_distribution) have been released based upon the Linux kernel.

The Linux kernel has extensive support for and runs on many [virtual machine](http://en.wikipedia.org/wiki/Virtual_machine) architectures both as the host operating system and as a guest operating system. The virtual machines usually emulate [Intel x86](http://en.wikipedia.org/wiki/Intel_x86) family of processors, though in a few cases [PowerPC](http://en.wikipedia.org/wiki/PowerPC) or [ARM](http://en.wikipedia.org/wiki/ARM_architecture) processors are also emulated.

At Google, the team led by Rubin developed a mobile device platform powered by the [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel). Google marketed the platform to handset makers and [carriers](http://en.wikipedia.org/wiki/Mobile_network_operator) on the premise of providing a flexible, upgradable system. Google had lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.[[28]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-EngadgetMobileOS-27)[[29]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-WSJ-28)[[30]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-DT-29)

Speculation about Google's intention to enter the mobile communications market continued to build through December 2006.Reports from the [BBC](http://en.wikipedia.org/wiki/BBC) and [The Wall Street Journal](http://en.wikipedia.org/wiki/The_Wall_Street_Journal) noted that Google wanted its search and applications on mobile phones and it was working hard to deliver that. Print and online media outlets soon reported rumors that Google was developing a Google-branded [handset](http://en.wikipedia.org/wiki/Handset#Telephony). Some speculated that as Google was defining technical specifications, it was showing prototypes to cell phone manufacturers and network operators.

## Hardware running Android

The main supported platform for Android is the [ARM architecture](http://en.wikipedia.org/wiki/ARM_architecture).

The Android OS can be used as an operating system for cellphones, netbooks and [tablets](http://en.wikipedia.org/wiki/Tablet_personal_computer), including the [Dell Streak](http://en.wikipedia.org/wiki/Dell_Streak), [Samsung Galaxy Tab](http://en.wikipedia.org/wiki/Samsung_Galaxy_Tab), TV and other devices.[[68]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-67)[[69]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-68) The first commercially available phone to run the Android operating system was the [HTC Dream](http://en.wikipedia.org/wiki/HTC_Dream), released on 22 October 2008.[[70]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-HTC-69) In early 2010 Google collaborated with [HTC](http://en.wikipedia.org/wiki/HTC) to launch its flagship[[71]](http://en.wikipedia.org/wiki/Android_%28operating_system%29" \l "cite_note-70) Android device, the [Nexus One](http://en.wikipedia.org/wiki/Nexus_One). This was followed later in 2010 with the [Samsung](http://en.wikipedia.org/wiki/Samsung)-made [Nexus S](http://en.wikipedia.org/wiki/Nexus_S).

The early feedback on developing applications for the Android platform was mixed.Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and no public issue-tracking system. (Google announced an issue tracker on 18 January 2008.) In December 2007, MergeLab mobile startup founder Adam MacBeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time."Despite this, Android-targeted applications began to appear the week after the platform was announced. The first publicly available application was the [Snake game](http://en.wikipedia.org/wiki/Snake_%28video_game%29) The [Android Dev Phone](http://en.wikipedia.org/wiki/Android_Dev_Phone) is a [SIM](http://en.wikipedia.org/wiki/Subscriber_Identity_Module)-unlocked and hardware-unlocked device that is designed for advanced developers. While developers can use regular consumer devices purchased at retail to test and use their applications, some developers may choose not to use a retail device, preferring an unlocked or no-contract device.

The Android [software development kit](http://en.wikipedia.org/wiki/Software_development_kit) (SDK) includes a comprehensive set of development tools.[[80]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-79) These include a [debugger](http://en.wikipedia.org/wiki/Debugger), [libraries](http://en.wikipedia.org/wiki/Software_library), a handset [emulator](http://en.wikipedia.org/wiki/Emulator) (based on [QEMU](http://en.wikipedia.org/wiki/QEMU)), documentation, sample code, and tutorials. The SDK is downloadable on the [android developer website](http://developer.android.com/sdk/index.html). Currently supported development platforms include computers running [Linux](http://en.wikipedia.org/wiki/Linux_kernel) (any modern desktop [Linux distribution](http://en.wikipedia.org/wiki/List_of_GNU/Linux_distributions)), [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X) 10.4.9 or later, [Windows XP](http://en.wikipedia.org/wiki/Windows_XP) or later. The officially supported [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is [Eclipse](http://en.wikipedia.org/wiki/Eclipse_%28software%29) (currently 3.5 or 3.6) using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use [command line](http://en.wikipedia.org/wiki/Command_line) tools ([Java Development Kit](http://en.wikipedia.org/wiki/Java_Development_Kit) and [Apache Ant](http://en.wikipedia.org/wiki/Apache_Ant) are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).[[81]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-80)

Android applications are packaged in [.apk](http://en.wikipedia.org/wiki/APK_%28file_format%29) format and stored under /data/app folder on the Android OS (the folder is accessible to root user only for security reasons). APK package contains .dex files(compiled byte code files called [Dalvik](http://en.wikipedia.org/wiki/Dalvik_Virtual_Machine) executables), resource files, etc.

### Android Operation System

Android is an operating system based on [Linux](http://www.vogella.de/articles/Ubuntu/article.html) with a [Java](http://www.vogella.de/articles/JavaIntroduction/article.html) programming interface. It provides tools, e.g. a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). [Android](http://www.vogella.de/articles/Android/article.html) is created by the Open Handset Alliance which is lead by Google.

Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. Dalvik uses special bytecode. Therefore you cannot run standard Java bytecode on Android. Android provides a tool "dx" which allows to convert Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) To simplify development Google provides the Android Development Tools (ADT) for [Eclipse](http://www.vogella.de/articles/Eclipse/article.html) . The ADT performs automatically the conversion from class to dex files and creates the apk during deployment.

Android supports 2-D and 3-D graphics using the OpenGL libraries and supports data storage in a [SQLite](http://www.vogella.de/articles/AndroidSQLite/article.html) database.

Every [Android](http://www.vogella.de/articles/Android/article.html) applications runs in its own process and under its own userid which is generated automatically by the Android system during deployment. Therefore the application is isolated from other running applications and a misbehaving application cannot easily harm other Android applications.

### Important Android components

An Android application consists out of the following parts:

* Activity - Represents the presentation layer of an Android application, e.g. a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application.
* Views - The User interface of an Activities is build with widgets classes which inherent from "android.view.View". The layout of the views is managed by "android.view.ViewGroups".
* [Services](http://www.vogella.de/articles/AndroidServices/article.html) - perform background tasks without providing an UI. They can notify the user via the notification framework in Android.
* [Content Provider](http://www.vogella.de/articles/Android/article.html#contentprovider) - provides data to applications, via a content provider your application can share data with other applications. Android contains a SQLite DB which can serve as data provider
* [Intents](http://www.vogella.de/articles/AndroidIntent/article.html) are asynchronous messages which allow the application to request functionality from other services or activities. An application can call directly a service or activity (explicit intent) or asked the Android system for registered services and applications for an intent (implicit intents). For example the application could ask via an intent for a contact application. Application register themself to an intent via an IntentFilter. Intents are a powerful concept as they allow to create loosely coupled applications.
* Broadcast Receiver - receives system messages and implicit intents, can be used to react to changed conditions in the system. An application can register as a broadcast receiver for certain events and can be started if such an event occurs.
* A Java Virtual Machine (JVM) enables a set of computer software programs and data structures to use a [virtual machine](http://en.wikipedia.org/wiki/Virtual_machine) model for the execution of other computer programs and [scripts](http://en.wikipedia.org/wiki/Scripting_language). The model used by a JVM accepts a form of computer [intermediate language](http://en.wikipedia.org/wiki/Intermediate_language) commonly referred to as [Java bytecode](http://en.wikipedia.org/wiki/Java_bytecode). This language conceptually represents the instruction set of a [stack-oriented](http://en.wikipedia.org/wiki/Stack-oriented_programming_language), [capability architecture](http://en.wikipedia.org/wiki/Capability_architecture). [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) states there are over 4.5 billion JVM-enabled devices
* A JVM can also execute bytecode compiled from programming languages other than Java. For example, [Ada](http://en.wikipedia.org/wiki/Ada_%28programming_language%29) source code can be compiled to execute on a JVM. JVMs can also be released by other companies besides Oracle (the developer of Java) — JVMs using the "Java" trademark may be developed by other companies as long as they adhere to the JVM specification published by Oracle and to related contractual obligations.
* Java was conceived with the concept of WORA: "[write once, run anywhere](http://en.wikipedia.org/wiki/Write_once,_run_anywhere)". This is done using the Java Virtual Machine. The JVM is the environment in which java programs execute. It is software that is implemented on non-virtual hardware and on standard [operating systems](http://en.wikipedia.org/wiki/Operating_system).
* JVM is a crucial component of the [Java platform](http://en.wikipedia.org/wiki/Java_%28software_platform%29), and because JVMs are available for many hardware and software [platforms](http://en.wikipedia.org/wiki/Platform_%28computing%29), Java can be both [middleware](http://en.wikipedia.org/wiki/Middleware) and a platform in its own right,[[clarification needed](http://en.wikipedia.org/wiki/Wikipedia:Please_clarify)] hence the trademark [write once, run anywhere](http://en.wikipedia.org/wiki/Write_once,_run_anywhere). The use of the same bytecode for all platforms allows Java to be described as "compile once, run anywhere", as opposed to "write once, compile anywhere", which describes cross-platform [compiled languages](http://en.wikipedia.org/wiki/Compiled_language). A JVM also enables such features as [automated exception handling](http://en.wikipedia.org/wiki/Automated_exception_handling), which provides "root-cause" debugging information for every software error ([exception](http://en.wikipedia.org/wiki/Exception_handling)), independent of the source code.
* A JVM is distributed along with a [set of standard class libraries](http://en.wikipedia.org/wiki/Java_Class_Library) that implement the Java [application programming interface](http://en.wikipedia.org/wiki/Application_programming_interface) (API). Appropriate APIs bundled together form the Java Runtime Environment (JRE).
* Java's execution environment is termed the Java Runtime Environment, or JRE.
* Programs intended to run on a JVM must be compiled into a standardized portable binary format, which typically comes in the form of [.class](http://en.wikipedia.org/wiki/Class_%28file_format%29) files. A program may consist of many classes in different files. For easier distribution of large programs, multiple class files may be packaged together in a [.jar](http://en.wikipedia.org/wiki/Jar_%28file_format%29) file (short for Java archive).
* The Java application launcher, java, offers a standard way of executing Java code. Compare javaw.[[2]](http://en.wikipedia.org/wiki/Java_Virtual_Machine#cite_note-1)
* The JVM [runtime](http://en.wikipedia.org/wiki/Run-time_system) executes .class or .jar files, [emulating](http://en.wikipedia.org/wiki/Emulator) the JVM [instruction set](http://en.wikipedia.org/wiki/Instruction_set) by [interpreting](http://en.wikipedia.org/wiki/Interpreter_%28computing%29) it, or using a [just-in-time compiler](http://en.wikipedia.org/wiki/Just-in-time_compilation) (JIT) such as Oracle's [HotSpot](http://en.wikipedia.org/wiki/HotSpot_%28Java%29). JIT compiling, not interpreting, is used in most JVMs today to achieve greater speed. There are also [ahead-of-time compilers](http://en.wikipedia.org/wiki/AOT_compiler) that enable developers to precompile class files into native code for particular platforms.
* Like most virtual machines, the Java Virtual Machine has a [stack](http://en.wikipedia.org/wiki/Stack_machine)-based architecture akin to a microcontroller/microprocessor. However, the JVM also has low-level support for Java-like classes and methods, which amounts to a highly idiosyncratic[[clarification needed](http://en.wikipedia.org/wiki/Wikipedia:Please_clarify" \o "Wikipedia:Please clarify)] [memory model](http://en.wikipedia.org/wiki/Java_Memory_Model) and capability-based architecture.

**Android Studio Installation Process**

For a few years now it's been clear that Android dominates the mobile OS landscape. This Java-based technology has sparked a new gold rush, with programmers competing to make money from their mobile apps. Android jobs are also plentiful, as shown by a quick job search using Indeed.com. To be successful, Android developers need a good grasp of the Java language, Android APIs, and Android app architecture. It's also essential to use an appropriate and effective development environment. For many years, Eclipse IDE with the ADT plugin was the preferred platform for Android development. Today it's Android Studio. If you're new to Android Studio, this tutorial series will get you started. I'll briefly introduce the Android development platform, then show you how to download, install, and run the software. After that, we'll spend most of our time actually using Android Studio to develop an animated mobile app. In Part 1 you'll start up your first Android project and get to know the project workspace in Android Studio. In Part 2 you'll code the app, learning how to use Android Studio to enter source code and resources into the project. Finally, in Part 3 we'll build and run the app using both an emulated hardware device and an Amazon Kindle Fire HD 7" tablet.

After you're comfortable with developing a basic mobile app in Android Studio, we'll explore more advanced topics like debugging, performance monitoring, and profiling with Android Studio. We'll also look at extending Android Studio with three useful plugins.

**Get started with Android Studio**

Android Studio is Google's officially supported IDE for developing Android apps. Based on IntelliJ IDEA, Android Studio is freely available under Apache License 2.0. The most recent stable version, 2.1.1, includes the following features:

* A unified environment where you can develop for all Android devices.
* Support for building Android TV apps and Android Wear apps.
* Template-based wizards to create common Android designs and components.
* A rich layout editor that lets users drag-and-drop user interface components, and that offers an option to preview layouts on multiple screen configurations.
* Android-specific refactoring and quick fixes.
* Gradle-based build support.
* Lint tools to catch performance, usability, version compatibility, and other problems.
* ProGuard integration and app-signing capabilities.
* A fast and feature-rich emulator.
* Instant Run to push changes to your running app without building a new APK (Application PacKage Zip file).
* Built-in support for Google Cloud Platform, enabling integration with Google Cloud Messaging and App Engine.
* C++ and NDK support.
* Plugin architecture for extending Android Studio via plugins.

**Download Android Studio**

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)

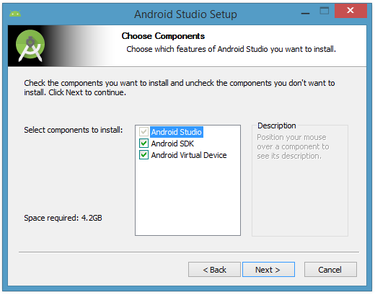
Once you've ensured your operating system is compatible with Android Studio 2.1.1, download the appropriate Android Studio distribution file. The Android Studio download page auto-detected that I'm running 64-bit Windows 8.1 and selected android-studio-bundle-143.2821654-windows.exe for me to download.

**Installing Android Studio on 64-bit Windows 8.1**

I 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 1.



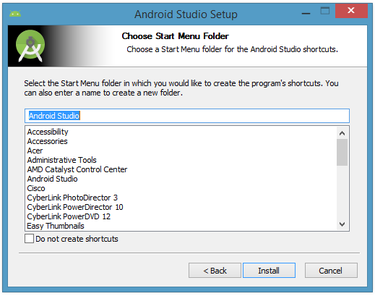
Clicking Next took me to the following dialog box, which gives you the option to decline installing the Android SDK (included with the installer) and an Android Virtual Device (AVD).



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

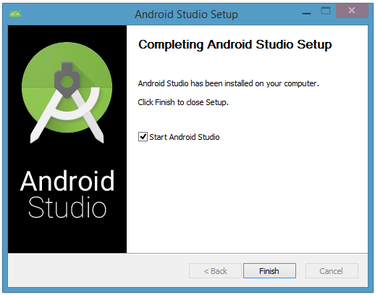
change the location or accept the default locations and click Next.

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.



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:



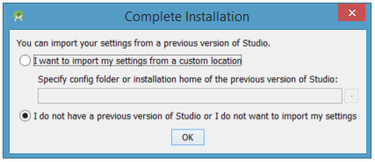
To complete your installation, leave the Start Android Studio box checked and click Finish.

**Running Android Studio**

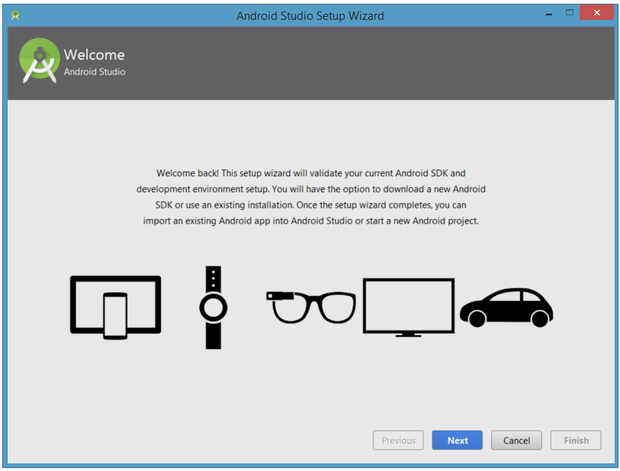
Android Studio presents a splash screen when it starts running:



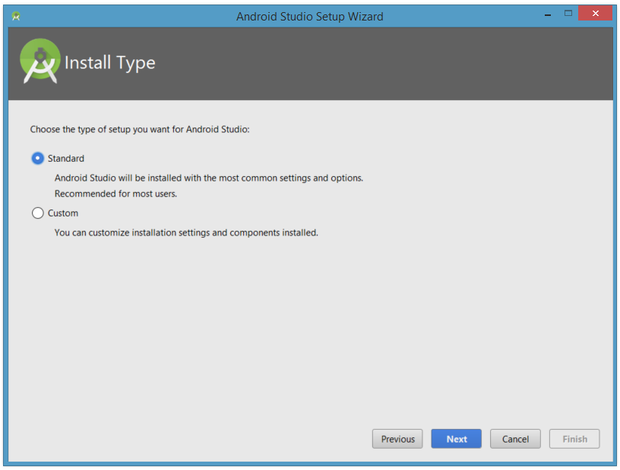
On your first run, you'll be asked to respond to several configuration-oriented dialog boxes. The first dialog box focuses on importing settings from any previously installed version of Android Studio.



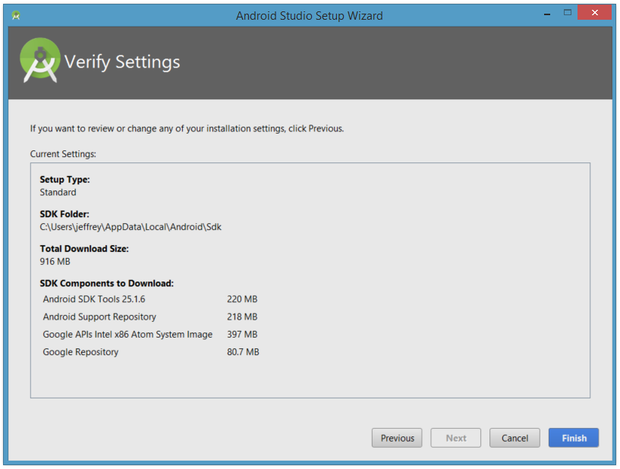
If you're like me, and don't have a previously installed version, you can just keep the default setting and click OK. Android Studio will respond with a slightly enhanced version of the splash screen, followed by the Android Studio Setup Wizard dialog box:



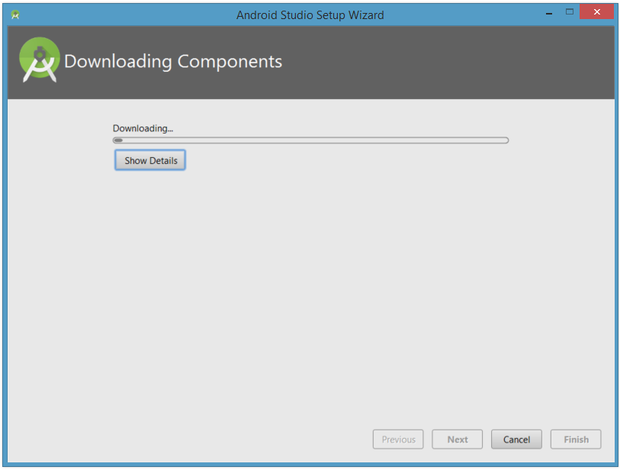
When you click Next, the setup wizard invites you to select an installation type for your SDK components. For now I recommend you keep the default standard setting.



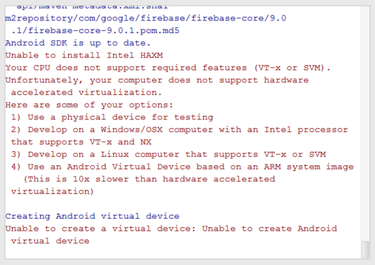
Click Next and verify your settings, then click Finish to continue.



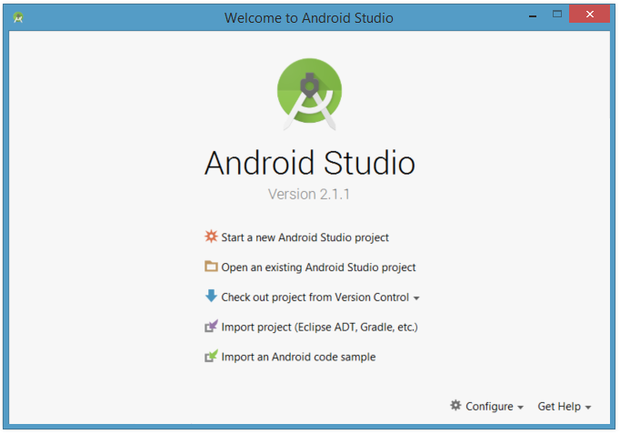
The wizard will download and unzip various components. Click Show Details if you want to see more information about the archives being downloaded and their contents.



If your computer isn't Intel based, you might get an unpleasant surprise after the components have completely downloaded and unzipped:



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:



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.

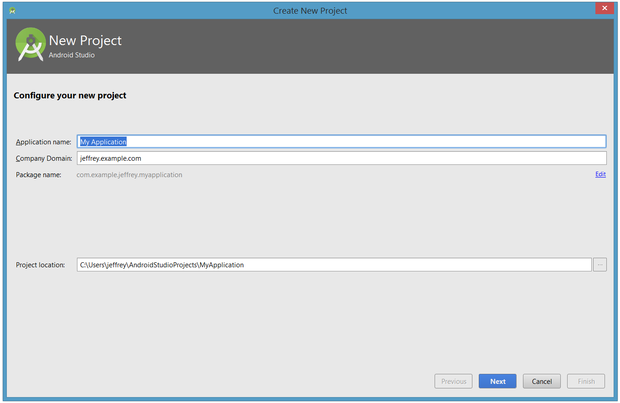
**Your first Android Studio mobile app**

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.

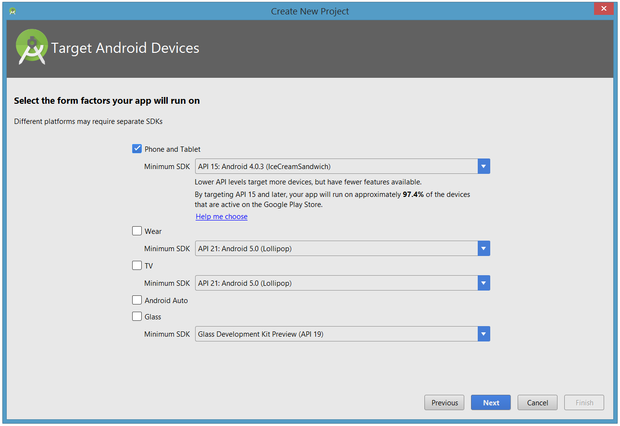
In the steps that follow, you'll start a new Android Studio project and get to know the project workspace, including the project editor that you'll use to code the app in Part 2.

**Starting a new project**

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 15.

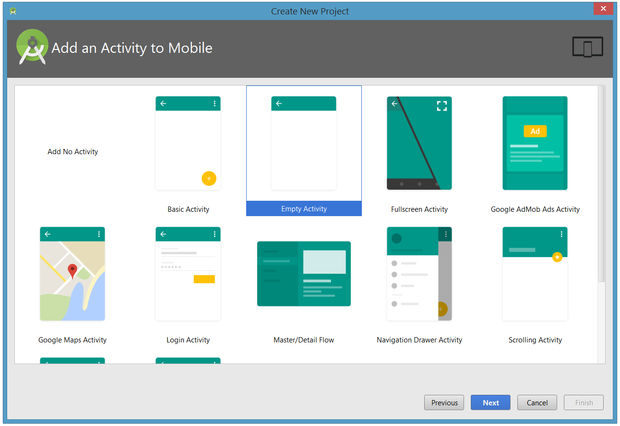


Enter W2A (Welcome to Android) as the application name and javajeff.ca as the company domain name. You should then see C:\Users\jeffrey\AndroidStudioProjects\W2A as the project location. Click Next to select your target devices.

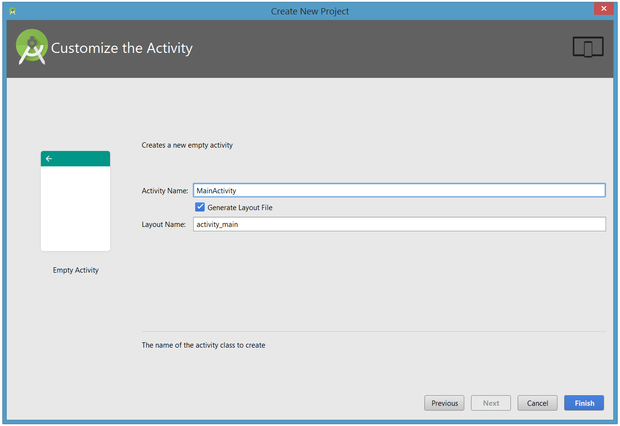


Android Studio lets you select form factors, or categories of target devices, for every app you create. I would have preferred to keep the default API 15: Android 4.0.3 (IceCreamSandwich) minimum SDK setting (under Phone and Tablet), which is supported by my Amazon Kindle Fire HD tablet. Because Android 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.

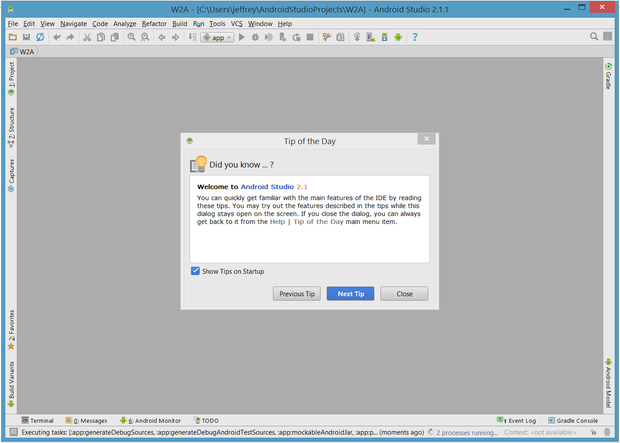
Click Next, and you will be given the opportunity to choose a template for your app's main activity. For now we'll stick with Empty Activity. Select this template and click Next.



Next you'll customize the activity:



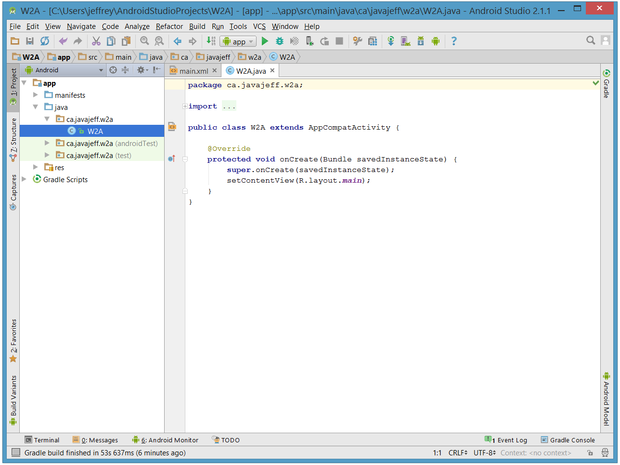
Enter W2A as the activity name and main as the layout name, and click Finish to complete this step. Android Studio will respond that it is creating the project, then take you to the project workspace.



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.

**The project and editor windows**

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.



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.

## 4. Error handling

Things are not always working as they should be. Several users report that get the following errors:

1. Project ... is missing required source folder: 'gen'
2. The project could not be built until build path errors are resolved.
3. Unable to open class file R.java.

To solve this error select from the menu Project -> Clean.

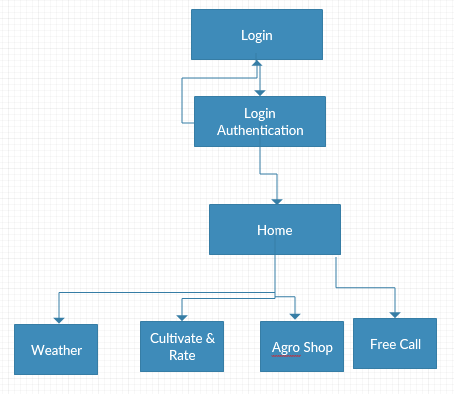
If you having problems with your own code you can use the LogCat viewer as described in [LogCat Viewer](http://www.vogella.de/articles/Android/article.html#views) .

**SYSTEM DESIGN:**

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

**Data flow diagram**

****

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

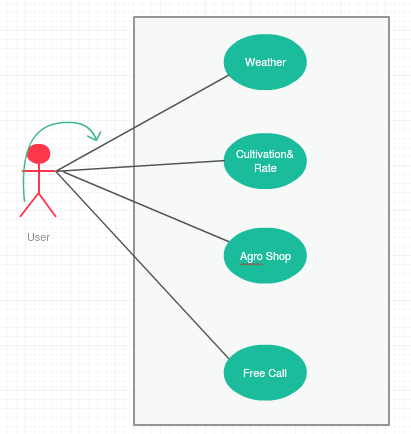
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

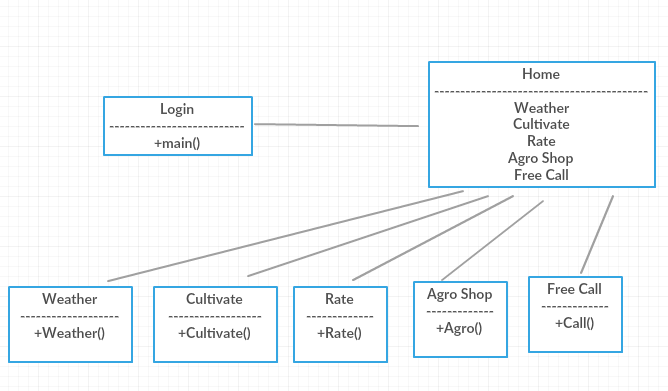
**Use Case Diagram**

****

**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

**Class Diagram**

****

### 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**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**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.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

**6.3 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.

**CHAPTER 6- CONCLUSION**

It can be possible to design a application for Smartphone OS such as Android mobile by which we can provide user friendly app to farmers. The purpose of this application is to implement a user friendly agriculture app means of communication between mobile phones and farmers. The system will allow users to search for all necessary requirement of a farmers.

**References**

[1] “World population prospects 2019: Highlights,” [Online]. Available: https://population.un.org/wpp/Publications/Files/WPP2019\_Highlights.pdf. Accessed on: Mar. 24, 2020.

[2] “AQUASTAT,” [Online]. Available: http://www.fao.org/aquastat/en/overview/methodology/water-use. Accessed on: Mar. 24, 2020.

[3] “More people, more food, worse water?” [Online]. Available: http://www.fao.org/3/ca0146en/CA0146EN.pdf. Accessed on: Mar. 24, 2020.

[4] “The future of food and agriculture: Alternative pathways to 2050,” [Online]. Available: http://www.fao.org/3/CA1553EN/ca1553en.pdf. Accessed on: Mar. 24, 2020.

[5] J. M. Talavera, L. E. Tobón, J. A. Gómez, M. A. Culman, J. M. Aranda, D. T. Parra, L. A. Quiroz, A. Hoyos, and L. E. Garreta, “Review of IoT applications in agro-industrial and environmental fields,” Comput. Electron. Agr., vol. 142, pp. 283–297, Nov. 2017. doi: 10.1016/j.compag.2017.09.015

[6] P. P. Ray, “Internet of things for smart agriculture: Technologies, practices and future direction,” J. Ambient Intell. Smart Environ., vol. 9, no. 4, pp. 395–420, Jun. 2017. doi: 10.3233/AIS-170440

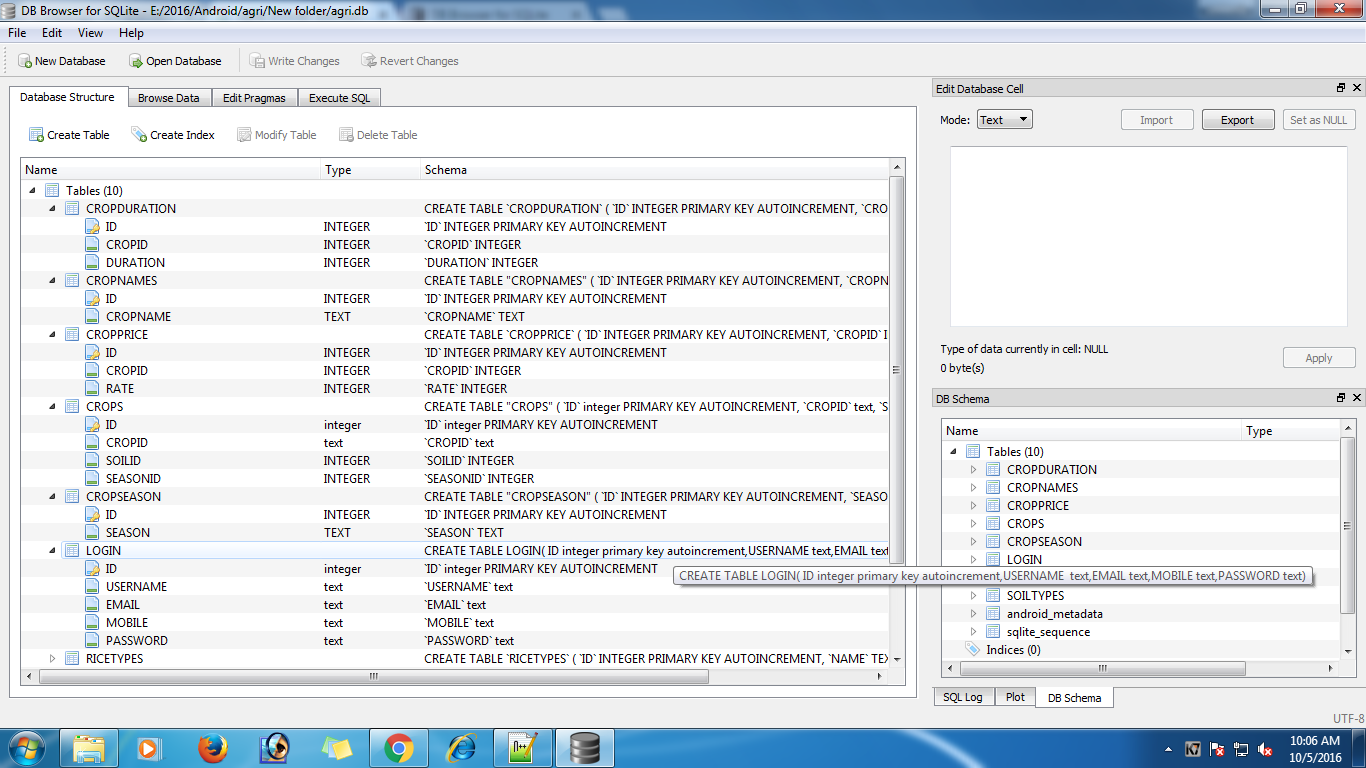
[7] A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas, “Internet of things in agriculture, recent advances and future challenges,” Biosyst. Eng., vol. 164, pp. 31–48, Dec. 2017. doi: 10.1016/j.biosystemseng.2017.09.007

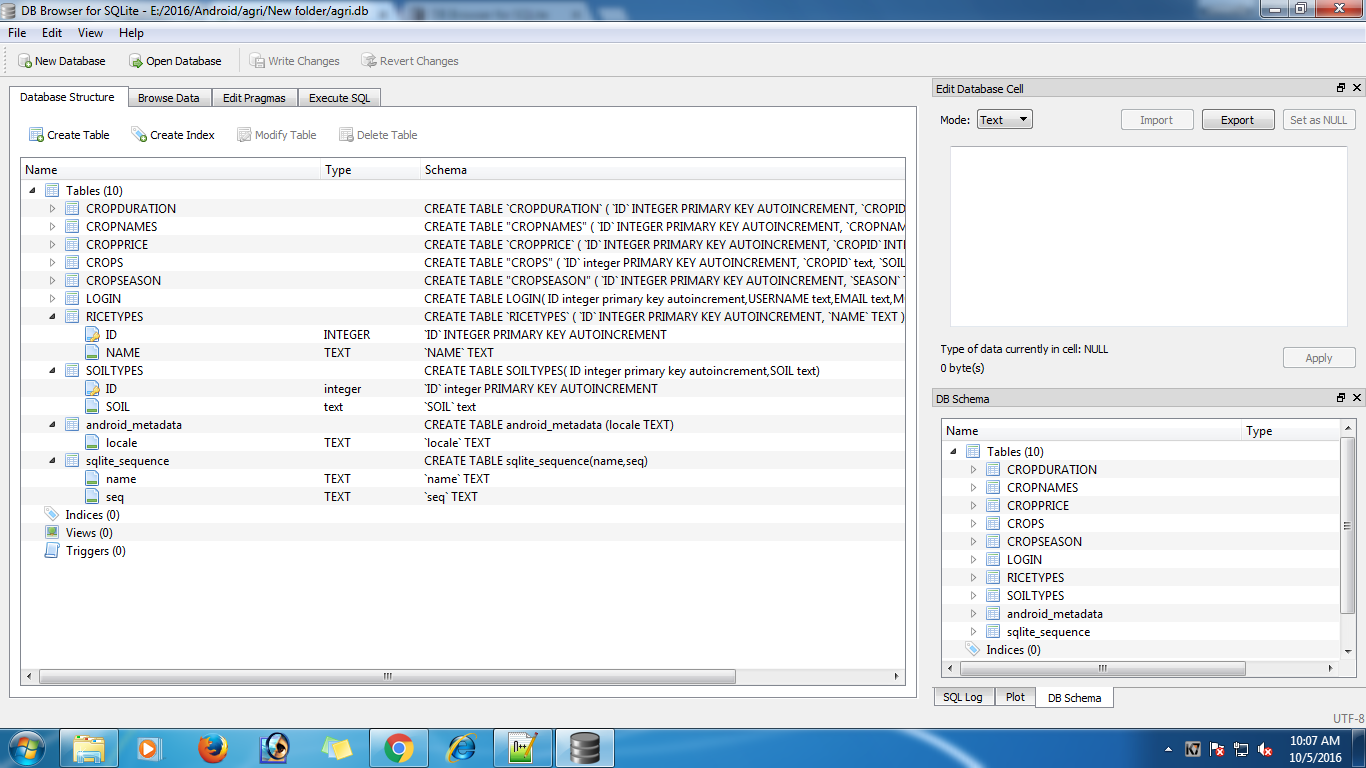
[8] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, “An overview of internet of things (IoT) and data analytics in agriculture: Benefits and challenges,” IEEE Int. Things J., vol. 5, no. 5, pp. 3758–3773, Oct. 2018. doi: 10.1109/JIOT.2018.2844296

[9] A. Khanna and S. Kaur, “Evolution of internet of things (IoT) and its significant impact in the field of precision agriculture,” Comput. Electron. Agr., vol. 157, pp. 218–231, Feb. 2019. doi: 10.1016/j.compag.2018.12.039

[10] X. J. Shi, X. S. An, Q. X. Zhao, H. M. Liu, L. M. Xia, X. Sun, and Y. M. Guo, “State-of-the-art internet of things in protected agriculture,” Sensors, vol. 19, no. 8, Article No. 1833, Apr. 2019. doi: 10.3390/s19081833

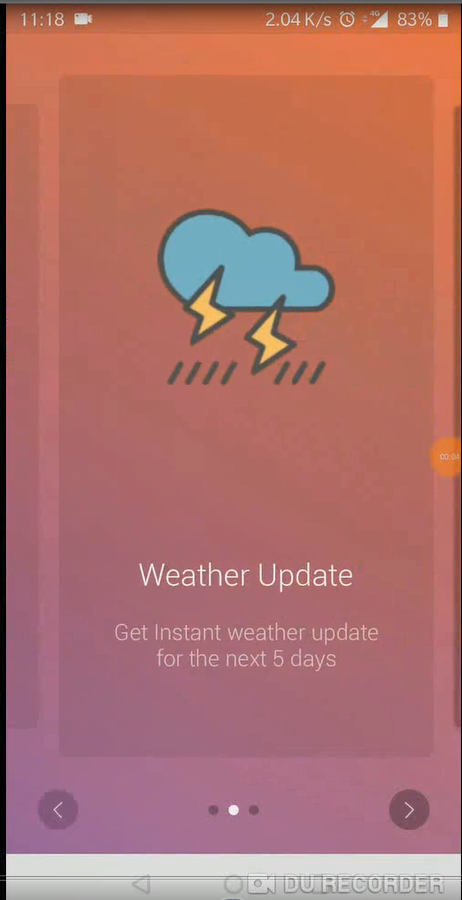
**Database Design:**



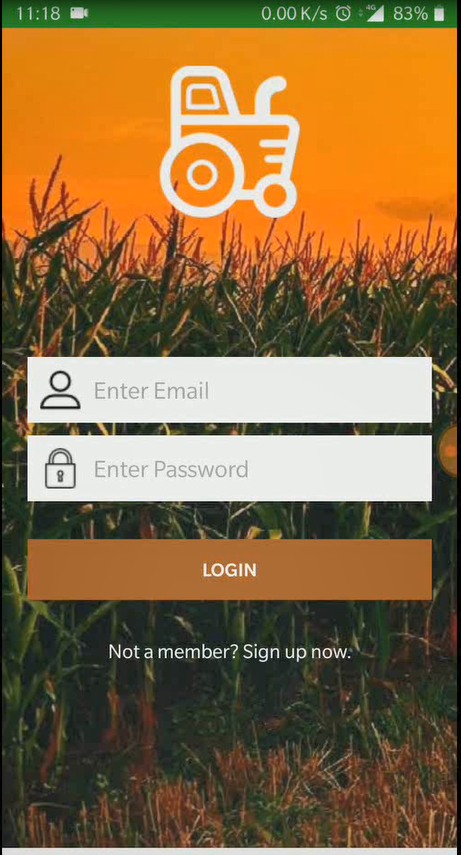


**Project Screen Shot:**

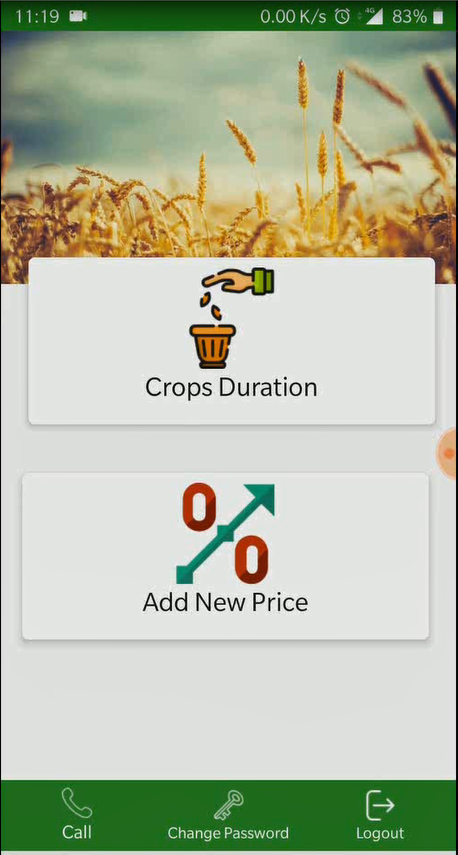
Slider :



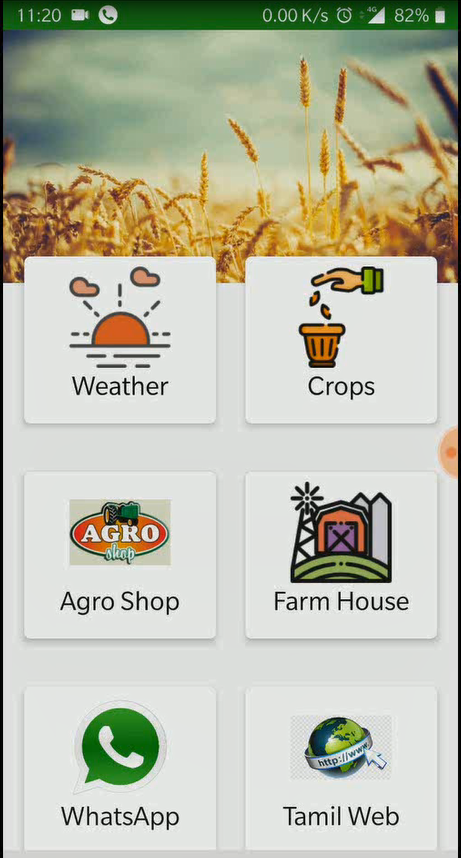
Login :



Admin Login :



User Login :



**Coding:**

**Copy your coding file and paste here**

**LOGIN.JAVA**

package com.example.admin.agriculture;

import android.app.Dialog;

import android.content.Intent;

import android.content.SharedPreferences;

import android.preference.PreferenceManager;

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.TextView;

import android.widget.Toast;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class login extends AppCompatActivity {

private static final String EMAIL\_PATTERN = "^[a-zA-Z0-9#\_~!$&'()\*+,;=:.\"(),:;<>@\\[\\]\\\\]+@[a-zA-Z0-9-]+(\\.[a-zA-Z0-9-]+)\*$";

private Pattern pattern = Pattern.compile(EMAIL\_PATTERN);

private Matcher matcher;

public boolean validateEmail(String email) {

matcher = pattern.matcher(email);

return matcher.matches();

}

public boolean isFirstStart;

LoginDatabaseAdapter loginDataBaseAdapter;

String userName,chkpass;

EditText edtuser,pass1;

Button btnlogin;

TextView txtregister;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

loginDataBaseAdapter=new LoginDatabaseAdapter(this);

loginDataBaseAdapter=loginDataBaseAdapter.open();

edtuser=(EditText)findViewById(R.id.email);

pass1=(EditText)findViewById(R.id.password);

btnlogin=(Button)findViewById(R.id.btnLogin);

txtregister=(TextView)findViewById(R.id.txtLinkToRegisterScreen);

btnlogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

userName = edtuser.getText().toString();

chkpass = pass1.getText().toString();

if (userName.length() == 0) {

edtuser.setError("Required");

edtuser.setFocusable(true);

} else if (!validateEmail(userName)) {

edtuser.setError("Not a valid email address!");

edtuser.setFocusable(true);

} else if (chkpass.length() == 0) {

pass1.setError("Requires");

pass1.setFocusable(true);

} else {

if(userName.equals("admin@gmail.com") && chkpass.equals("ADMIN"))

{

Intent i=new Intent(login.this,AdminDashboard.class);

startActivity(i);

}else {

String storedPassword = loginDataBaseAdapter.getSinlgeEntry(userName.toString());

Toast.makeText(getApplicationContext(), storedPassword+"++++++++++++++++++++", Toast.LENGTH\_LONG).show();

if (chkpass.equals(storedPassword)) {

Toast.makeText(login.this, storedPassword+"Congrats: Login Successfull", Toast.LENGTH\_LONG).show();

Intent i = new Intent(getApplicationContext(), MainActivity.class);

i.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

startActivity(i);

} else {

Toast.makeText(login.this, "User Name or Password does not match", Toast.LENGTH\_LONG).show();

}

}

}

}

});

txtregister.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent i=new Intent(login.this,register.class);

startActivity(i);

}

});

}

@Override

protected void onDestroy() {

super.onDestroy();

// Close The Database

loginDataBaseAdapter.close();

}

}

MAINACTIVITY.JAVA

package com.example.admin.agriculture;

import android.Manifest;

import android.app.Activity;

import android.app.Dialog;

import android.content.ComponentName;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.SharedPreferences;

import android.net.Uri;

import android.preference.PreferenceManager;

import android.support.annotation.NonNull;

import android.support.design.widget.BottomNavigationView;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.support.v7.widget.CardView;

import android.support.v7.widget.Toolbar;

import android.telephony.PhoneNumberUtils;

import android.text.InputType;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.Window;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.TextView;

import android.widget.Toast;

import com.karumi.dexter.Dexter;

import com.karumi.dexter.MultiplePermissionsReport;

import com.karumi.dexter.PermissionToken;

import com.karumi.dexter.listener.PermissionRequest;

import com.karumi.dexter.listener.multi.MultiplePermissionsListener;

import java.util.List;

public class MainActivity extends AppCompatActivity {

CardView one,two,three,four,five,six;

ImageView calling;

LoginDatabaseAdapter loginDatabaseAdapter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.dashboard);

/\* Toolbar toolbar = (Toolbar) findViewById(R.id.main);

setSupportActionBar(toolbar);

getSupportActionBar().setTitle(null);\*/

loginDatabaseAdapter = new LoginDatabaseAdapter(this);

loginDatabaseAdapter = loginDatabaseAdapter.open();

one = (CardView) findViewById(R.id.weather);

two = (CardView) findViewById(R.id.crops);

three = (CardView) findViewById(R.id.sales);

four=(CardView)findViewById(R.id.farmhouse);

five=(CardView)findViewById(R.id.chat);

six=(CardView)findViewById(R.id.tamilweb);

BottomNavigationView bottomNavigationView = (BottomNavigationView) findViewById(R.id.bottom\_navigation);

if (bottomNavigationView != null) {

bottomNavigationView.setOnNavigationItemSelectedListener(

new BottomNavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(@NonNull MenuItem item) {

selectMenu(item);

return false;

}

});

Dexter.withActivity(this).withPermissions(Manifest.permission.VIBRATE,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE,

Manifest.permission.ACCESS\_FINE\_LOCATION,

Manifest.permission.CALL\_PHONE,

Manifest.permission.ACCESS\_COARSE\_LOCATION).withListener(new MultiplePermissionsListener() {

@Override

public void onPermissionsChecked(MultiplePermissionsReport report) {

// check if all permissions are granted

if (report.areAllPermissionsGranted()) {

// do you work now

}

// check for permanent denial of any permission

if (report.isAnyPermissionPermanentlyDenied()) {

// permission is denied permenantly, navigate user to app settings

}

}

@Override

public void onPermissionRationaleShouldBeShown(List<PermissionRequest> permissions, PermissionToken token) {

token.continuePermissionRequest();

}

}).onSameThread()

.check();

one.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent i = new Intent(MainActivity.this, weathe.class);

startActivity(i);

}

});

two.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent i = new Intent(MainActivity.this, crops.class);

startActivity(i);

}

});

three.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent i = new Intent(MainActivity.this, sales.class);

startActivity(i);

}

});

four.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

/\*Intent intent = new Intent();

intent.setData(Uri.parse("http://casidiablo.net"));

startActivity(intent);\*/

Intent i = new Intent(MainActivity.this, FarmHouse.class);

startActivity(i);

}

});

five.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent sendIntent = new Intent("android.intent.action.MAIN");

sendIntent.setComponent(new ComponentName("com.whatsapp","com.whatsapp.Conversation"));

sendIntent.putExtra("jid", PhoneNumberUtils.stripSeparators("919566355386")+"@s.whatsapp.net");

startActivity(sendIntent);

}

});

six.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

//Intent intent = new Intent();

//intent.setData(Uri.parse("http://www.aed.tn.gov.in/"));

//startActivity(intent);

Intent intent = new Intent(MainActivity.this, Tamilweb.class);

startActivity(intent);

}

});

}

}

@Override

public void onBackPressed() {

}

protected void selectMenu(MenuItem item) {

item.setChecked(true);

switch (item.getItemId()) {

case R.id.call:

Intent intent = new Intent(Intent.ACTION\_DIAL);

intent.setData(Uri.parse("tel:0123456789"));

startActivity(intent);

break;

case R.id.chpwd:

ViewDialog alert = new ViewDialog();

alert.showDialog(this, "Error de conexión al servidor");

break;

case R.id.logout:

Intent i = new Intent(MainActivity.this, login.class);

i.setFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);

finish();

startActivity(i);

break;

}

}

public class ViewDialog {

public void showDialog(Activity activity, String msg){

final Dialog dialog = new Dialog(activity);

dialog.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

dialog.setCancelable(false);

dialog.setContentView(R.layout.dialog);

final EditText useremail = (EditText) dialog.findViewById(R.id.email);

final EditText newpwd = (EditText) dialog.findViewById(R.id.pwd);

Button change = (Button) dialog.findViewById(R.id.Ok);

Button dialogButton = (Button) dialog.findViewById(R.id.Cancel);

change.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

loginDatabaseAdapter.updatepassword(useremail.getText().toString(),newpwd.getText().toString());

Toast.makeText(MainActivity.this,"Password Updated", Toast.LENGTH\_LONG).show();

}

});

dialogButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

dialog.dismiss();

}

});

dialog.show();

}

}

}