**E-Medical Box**

# A REPORT ON PROJECT BASED LEARNING (SEMESTER -II)

*Submitted by*

Yash Rajput (10537)

# FIRST YEAR ENGINEERING



## Society for Computer Technology and Research’s

**PUNE INSTITUTE OF COMPUTER TECHNOLOGY**

## DHANKAWADI, PUNE – 43

## A.Y. 2019-20

# - CERTIFICATE-

This is to certify that the work incorporated in the report entitled **“E-MEDICAL BOX”** is carried out by **Mr. Yash Rajput (Roll No. - 10537),** who is part a group of students

with Project Id **1-B5-2020** under the subject ***Project Based Learning*** during A.Y.

## 2019-2020.

Such material has not been submitted to any other University/ Institute for any financial support. The literature related to the problem investigated has been appropriately cited and duly acknowledged wherever facilities and suggestions have been availed of.

## Date: 26 May 2020 Name & Sign of Project Guide

**Place:** PUNE  **Prof. M.D. Bandkar**

## Name & Sign of PBL Coordinator Name & Sign of Head of Department

**Prof. N. P. Sapkal Prof. E. M. Reddy**

**Abstract**

Our project which is an E-MEDICAL BOX is based upon the designed for the proper completion of medicinal doses. Many medical errors are due to the fact that people in charge of patient or elder's medication have to deal with sorting huge amounts of pills each day. This paper consists on the conception, design and creation of a pillbox prototype intended to solve this deficiency in the medical area as it has the ability of sorting out the pills by itself as well as many other advanced features, with this device being intended to be used by hospitals or retirement homes. This medication pill box is focused on patients who frequently take medications or vitamin supplements, or attendants who deal with the more seasoned or patients. Our smart pill box is programmable that enables medical caretakers or clients to determine the pill amount and timing to take pills, and the service times for every day. Our shrewd pills box contains nine separate sub-boxes. In this manner, medical caretakers or clients can set data for nine distinct pills. At the point when the pill time has been set, the pillbox will remind clients or patients to take pills utilizing sound and light. The warning of pills should be taken will be shown by an android application which is held by the patient. Contrasted and the conventional pill box that requires clients or attendants to stack the crate each day or consistently. Our shrewd pill box would essentially discharge medical attendants or clients' weight on much of the time preloading pills for patients or clients and overlook the measurements which must be taken.

This project is made by the group of six people. This group was divided in the group of three persons one with the application development and other with the Arduino assembling.

The app development group consists of Manojkumar Sirvi, Yash Rajput, Vedant Kulkarni and the other group i.e. the Arduino consists of Tanmay Kale, Rutvij Wamanse, Onkar Saudagar.

In this we are going to take a box which consists of different compartments that can hold medicines. So, basically it will remind the user of this E-box to take the medicine on time. In this we are going to use an Arduino module and besides this an android application will be developed so that user can give input to the Arduino module through it. The input will consist of the dose amount, medicine name, time when to take it and till when the course will run. This will be synchronized with the Arduino module with the help of Bluetooth. Arduino module will consist of the Real Time Clock (RTC) which will look upon the current time and when the time matches with the input given by the user alarm will strike. Hence the user will get to know that its time to take the medicine. Along with the alarm a LED (Light Emitting Diode) will also be there which will illuminate. There will be a push button to stop the alarm. We will use a 9V battery for the power supply. The display will 1x 1602(16x2) LCD (Liquid Crystal Display) with I2C/IIC interface – Blue Backlight displays the date and time which are given by the user and also it will display the time of next dose.

**ACKNOWLEDGEMENT**

I would like to express my special thanks to our PBL coordinator “Mr. M.D. Bandkar” for their able guidance and support in completion my project.

I would also like to thank our HOD “Prof. E.M. Reddy” for his complete support and for giving us the golden opportunity so that we can do this wonderful project our project, which helped me to acquire the knowledge of new things.

I am really thankful to them. This would not complete without their guidance and support.

I am making this project not only for marks but also to fulfill my interest towards making a new thing by acquiring knowledge which will be helpful to complete my project.

Once I thank all of them who helped me and always supported me.

Place: Pune YASH RAJPUT

(10537)

# TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
| 1. | Introduction | 1 |
| 2. | Essentials for Arduino | 2 |
| 3. | Essentials for Application Development | 7 |
| 4. | Learning of Arduino | 12 |
| 5. | Learning of Android Studio | 14 |
| 6. | App Development | 16 |
| 7. | Assembling the Parts of Arduino | 22 |
| 8. | My Work in the Project | 24 |
| 9. | Conclusion | 32 |
| 10. | References | 33 |

|  |  |  |
| --- | --- | --- |
| **Figure**  **No** | **Title** | **Page**  **No** |
| 1. | Arduino Mega | 2 |
| 2. | Bluetooth Module | 3 |
| 3. | Jumper Wire | 3 |
| 4. | Breadboard | 4 |
| 5. | Pin Config Of RTC | 5 |
| 6. | RTC | 5 |
| 7. | LCD | 6 |
| 8. | Features of Java | 8 |
| 9. | Features Of OOPS | 9 |
| 10. | Coding Practice for ARDUINO | 13 |
| 11. | THE USER INTERFACE | 15 |
| 12. | Splash Screen | 17 |
| 13. | First Time User Intro | 18 |
| 14. | Device Mac Address | 19 |
| 15. | Home Screen | 19 |
| 16. | Input Data Screen | 20 |
| 17. | User Guide | 21 |
| 18. | Feedback | 22 |
| 19. | Final Diagram | 24 |

**Nomenclature**

Dimensions of Box:

Length: 21.4 cm

Breadth: 3 cm

Height: 2.2 cm

Weight of the Box: 25 g

Arduino Mega:

Model No.: 2560 REV 3

Length:101.52 mm

Width:53.3 mm

Weight: 37g

Bluetooth:

Model No.: HC-05

Length: 37.3 mm

Width: 15.5 mm

Weight: 3.5 g

## Chapter 1

**INTRODUCTION**

There can be a lot of individuals out there who need constant help – may it be our elderly people, family members, the ones who have special needs. Elders are more affected by the timing of taking a certain drug than others, in order to prevent any dysfunction or illness timing is a must.

But as with aging comes poor eye sight and poor memory, what if the patient has a dementia like Alzheimer. Some people may forget to take the medicines at the correct time and can forget the medicines which they have to take. In order to eliminate the factors of always needed observation like nurses or taking a risk of a missed dose, we had to find an easy, portable and efficient solution. E-Medical Box already exist but most of them are either has limited use, doesn't fit for elder ages or even has a big size that makes it not suitable to take it with you anywhere.

While at the same time it had be fit for the elders and their limited knowledge and experience to implement the ease of use. Size and portability were also an important fact that we had to keep in mind.

Through that same network its connected to the mobile phone, which with it you can set the timing interval for the dose and also notifies you by many ways when the dose time comes. Also, we added a buzzer with a LED to make a type of physical warning, so that it leaves you no choice but to remember the pill time and take it.

The aim of this study is to build an E-Medical Box for Medicine Reminder and Monitoring System. When the pill time has been set, the pillbox will remind clients or patients to take pills utilizing sound and light. The warning of pills should be taken will be shown by an android application which is held by the patient.

## Chapter 2

**ESSENTIALS FOR ARDUINO**

Firstly, we took the information of the components that we are going to be needed in the Arduino setup. The following list gives the brief description about the components that we are going to use.

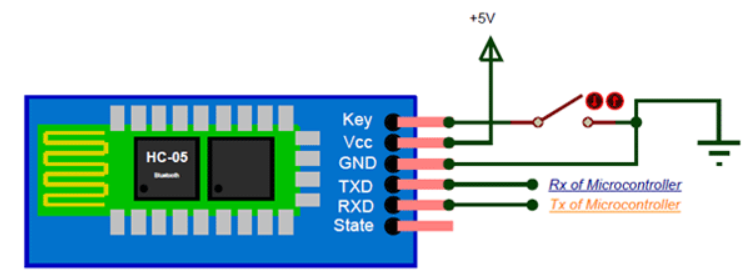
1. Arduino Mega: It is a 8-bit board with 54 digital pins, 16 analog inputs, and 4 serial ports. The **Arduino Mega 2560** is a microcontroller board based on the [ATmega2560](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf). It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila

1.Arduino Mega

1. Bluetooth Module HC- 05:

The **HC-05** is a very cool module which can add two-way (full-duplex) wireless functionality to your projects. You can use this module to communicate between two microcontrollers like Arduino or communicate with any device with Bluetooth functionality like a Phone or Laptop.

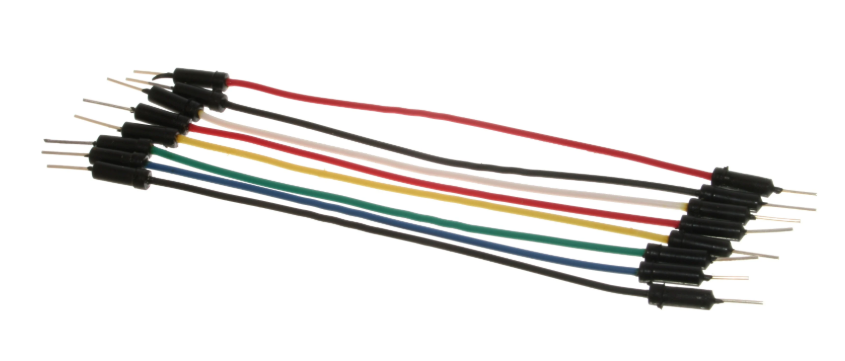
The **HC-05** has two operating modes, one is the Data mode in which it can send and receive data from other Bluetooth devices and the other is the AT Command mode where the default device settings can be changed. We can operate the device in either of these two modes by using the key pin as explained in the pin description. It is very easy to pair the HC-05 module with microcontrollers because it operates using the Serial Port Protocol (SPP). Simply power the module with +5V and connect the Rx pin of the module to the Tx of MCU and Tx pin of module to Rx of MCU. During power up the key pin can be grounded to enter into Command mode, if left free it will by default enter into the data mode. As soon as the module is powered you should be able to discover the Bluetooth device as “HC-05” then connect with it using the default password 1234 and start communicating with it.



2.Bluetooth Module

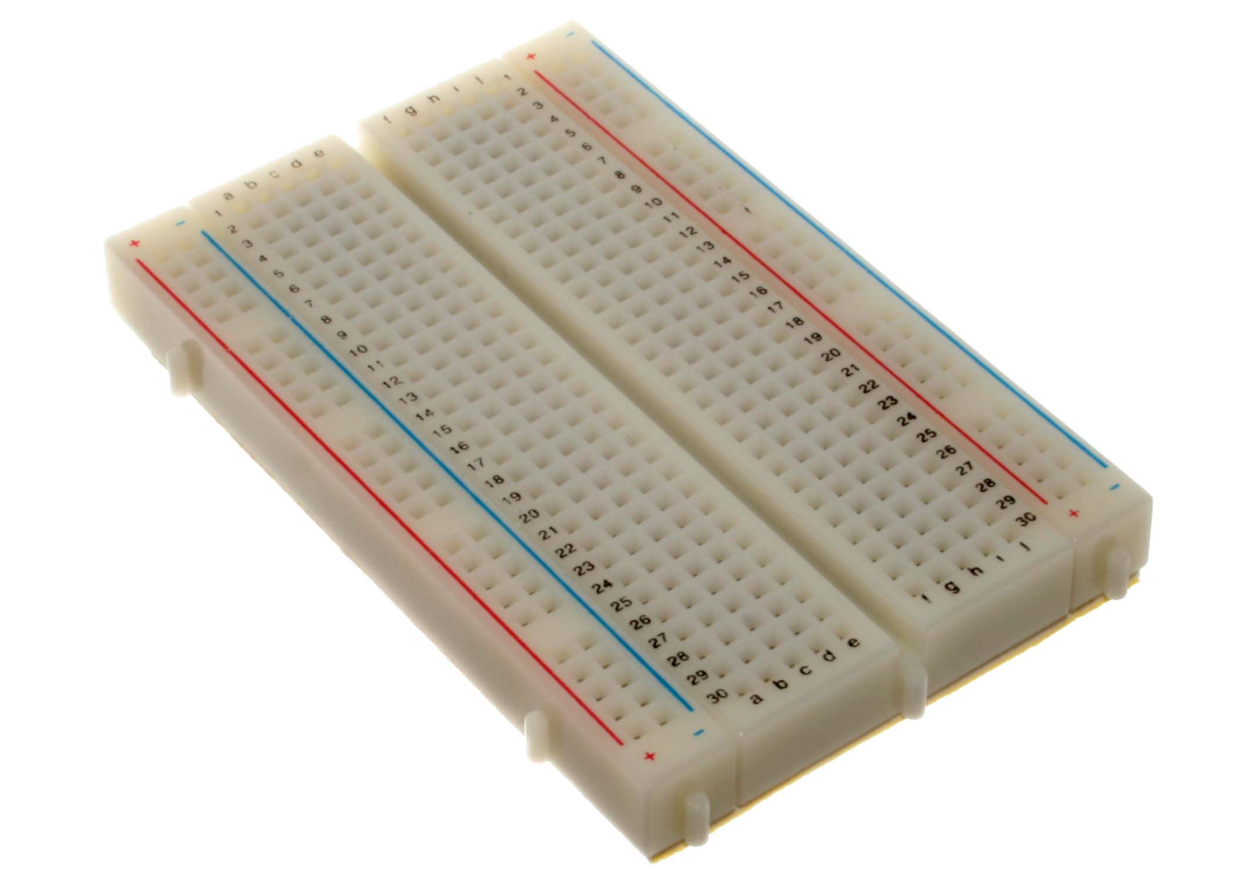
3)M/F Wires: A **jump wire** (also known as jumper wire, or jumper) is an [electrical wire](https://en.wikipedia.org/wiki/Electrical_wire), or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a [breadboard](https://en.wikipedia.org/wiki/Breadboard) or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the [header connector](https://en.wikipedia.org/wiki/Pin_header#Header_connector) of a circuit board, or a piece of test equipment.



3. Jumper Wire

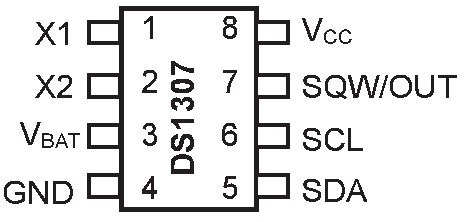
4)Breadboard: A breadboard is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to prototype an electronic circuit.



4. Breadboard

5)Real Time Clock (RTC): Real time clocks (RTC), as the name recommends are clock modules.[The DS1307 real time clock](https://www.elprocus.com/real-time-clock-ds1307-ic-applications/) (RTC) IC is an 8 pin device using an I2C interface. The DS1307 is a low-power clock/calendar with 56 bytes of battery backup SRAM. The clock/calendar provides seconds, minutes, hours, day, date, month and year qualified data. The end date of each month is automatically adjusted, especially for months with less than 31 days.

They are available as integrated circuits (ICs) and supervise timing like a clock and also operate date like a calendar. The main advantage of RTC is that they have an arrangement of battery backup which keeps the clock/calendar running even if there is power failure. An exceptionally little current is required for keeping the RTC animated. We can find these RTCs in many applications like embedded systems and computer mother boards, etc. In this article we are going to see about one of the real time clock (RTC), i.e. DS1307.



5. Pin Config Of RTC



6. RTC

6)LCD: It has 16 pins and the first one from left to right is the Ground pin. The second pin is the VCC which we connect the 5 volts pin on the Arduino Board. Next is the Vo pin on which we can attach a potentiometer for controlling the contrast of the display.

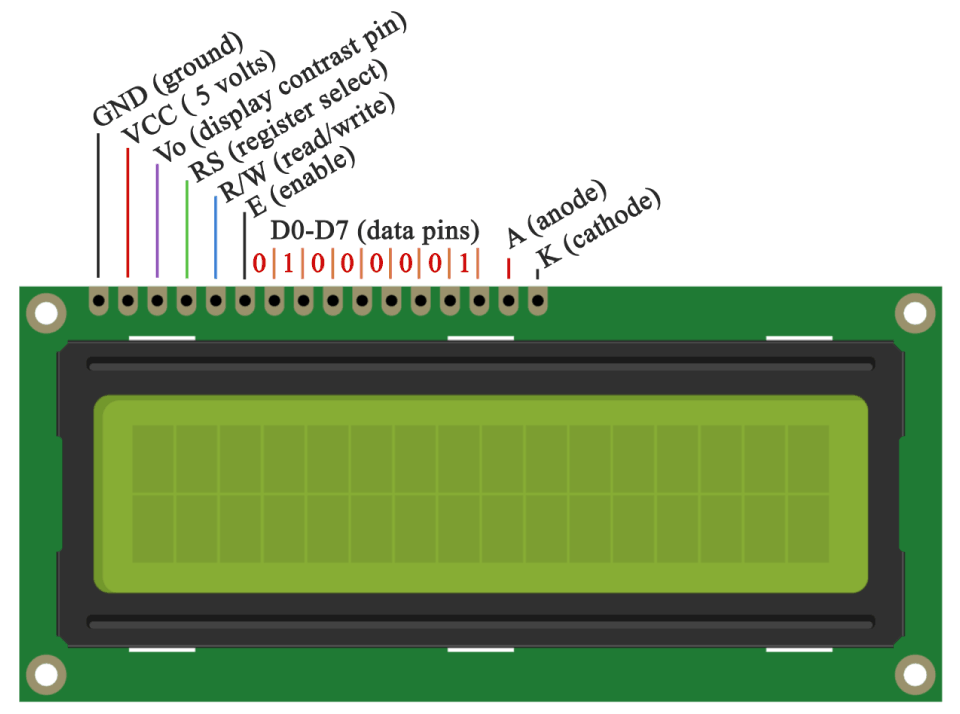
Next, The RS pin or register select pin is used for selecting whether we will send commands or data to the LCD. For example, if the RS pin is set on low state or zero volts, then we are sending commands to the LCD like: set the cursor to a specific location, clear the display, turn off the display and so on. And when RS pin is set on High state or 5 volts we are sending data or characters to the LCD. Next comes the R / W pin which selects the mode whether we will read or write to the LCD. Here the write mode is obvious and it is used for writing or sending commands and data to the LCD. The read mode is used by the LCD itself when executing the program which we don’t have a need to discuss about it in this tutorial.

Next is the E pin which enables the writing to the registers, or the next 8 data pins from D0 to D7. So through this pins we are sending the 8 bits data when we are writing to the registers or for example if we want to see the latter uppercase A on the display we will send 0100 0001 to the registers according to the ASCII table.

And the last two pins A and K, or anode and cathode are for the LED back light.

After all we don’t have to worry much about how the LCD works, as the Liquid Crystal Library takes care for almost everything. From the Arduino’s official website, you can find and see the functions of the library which enable easy use of the LCD. We can use the Library in 4- or 8-bit mode. In this tutorial we will use it in 4-bit mode, or we will just use 4 of the 8 data pins.

This are the main components which were needed. Other Components are Battery connector, 9v Battery, Led, Buzzer, Box, Push Button.



7. LCD

## Chapter 3

**EESSENTIALS FOR APP DEVELOPMENT**

Firstly, to make the model working it was essential to make a android application through which it was possible to give input to the Arduino kit. To build a android application following things where necessary:

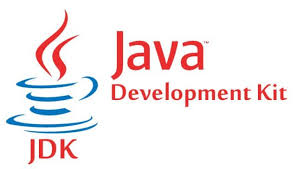
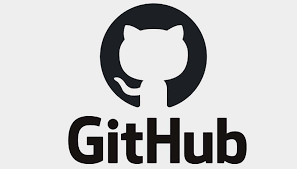
* To develop an application compatible with different android versions.
* To design attractive UI for the application.
* To provide interactive navigation for users.
* To add animations in between activities.
* To save data of different batches as Images and Texts.
* To provide the saved data as per entered details.

PREREQUISITES AND REQUIREMENTS

1. The UI of the app is designed in XML. So for designing layouts XML should be known. Also vector assets , animations , navigations , menus can be designed in XML.
2. Java language is used for the programming the app i.e. working and backend of the app . Along with Java concepts OOP(object oriented programming) like inheritance , method overriding , polymorphism , etc. should be known.
3. A brief introduction of Android studio and its tools should be known.
4. Basic knowledge about Git/GitHub is required for working in group.

**SOFTWARES USED**

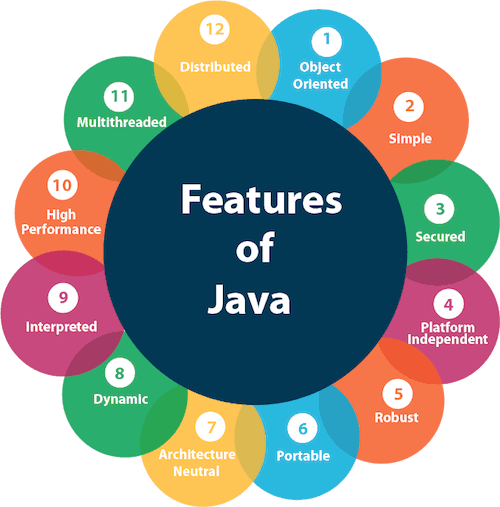
1. Java Development kit (JDK)
2. Android studio (IDE)
3. Android SDK
4. GitHub



**THEORY**

**a. Java**

Java is general-purpose programming language that is class-based , object-oriented and designed to have as few implementation dependencies as possible. Hence java is used in Android studio. Basics of Java are variables , input-output , operators , control flow statements , loops . File handling is needed for working with files in OS. Then Java has object oriented features which explain us structure of Java file that every Java file is a class . Many concepts of OOP are implemented in Java.

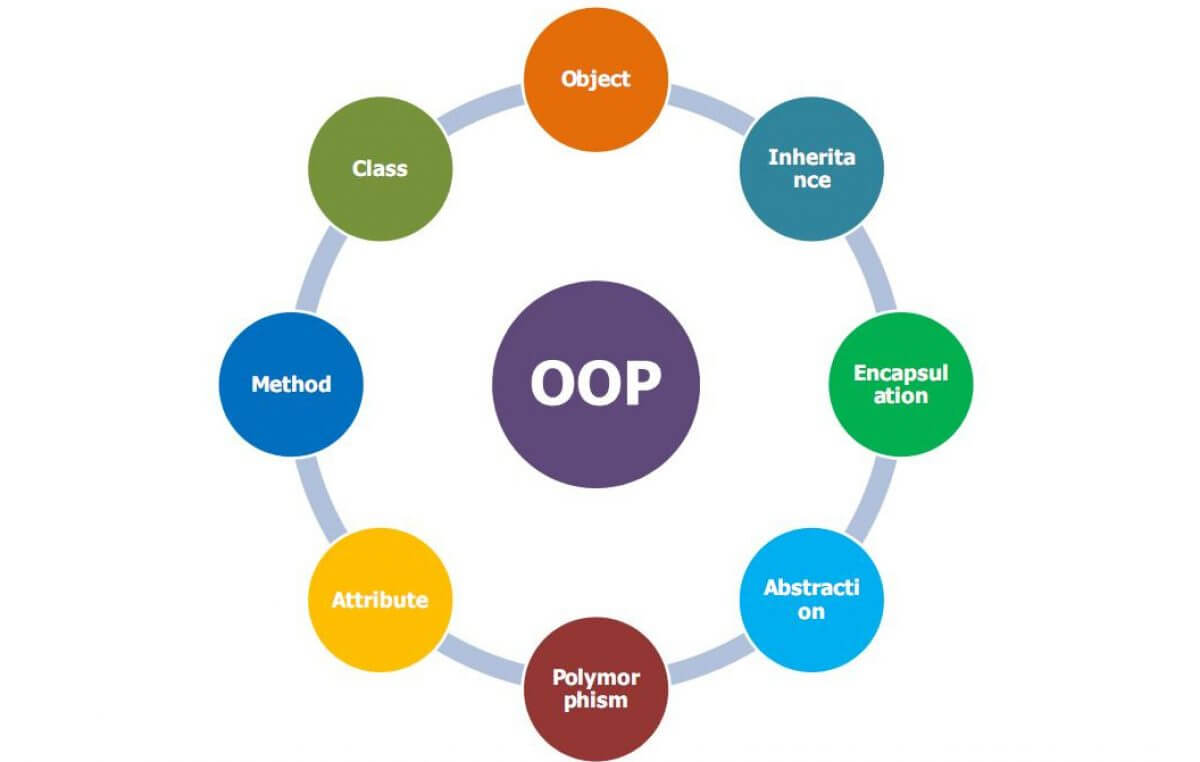


**8.Features of Java**

1. **OOP**

Object oriented programming paradigm based on the concept of “objects” , which can contain data , in the form of fields, code and in the form of procedures.

Terminology:



**9.Features Of OOPS**

**Classes**: A class provides a template or blueprint that describes the structures and behavior of a set of similar objects. **Objects**: An object is specific instance of a class . **Inheritance**: A new class can be created from an existing class often known as child class contains the properties and the behaviors of the parent class. **Polymorphism**: It enables to use one behavioral property in several different forms. **Abstraction and Encapsulation**: Abstractions refers to revealing essential details but hiding the implementation details. Encapsulation is the technique of packing data into a single component (class) to hide implementation details from the users.

**c. XML:**

Extensible Mark-up Language (XML) is a mark-up language that defines a set of rules for encoding [documents](https://en.wikipedia.org/wiki/Electronic_document) in a [format](https://en.wikipedia.org/wiki/File_format) that is both [human-readable](https://en.wikipedia.org/wiki/Human-readable_medium) and [machine-readable](https://en.wikipedia.org/wiki/Machine-readable_data). The [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium)'s XML 1.0 Specification of 1998and several other related specifications—all of them free [open standards](https://en.wikipedia.org/wiki/Open_standard)—define XML.

The design goals of XML emphasize simplicity, generality, and usability across the [Internet](https://en.wikipedia.org/wiki/Internet).[[6]](https://en.wikipedia.org/wiki/XML#cite_note-XML_Goals-6) It is a textual data format with strong support via [Unicode](https://en.wikipedia.org/wiki/Unicode) for different [human languages](https://en.wikipedia.org/wiki/Language). Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary [data structures](https://en.wikipedia.org/wiki/Data_structure)[[7]](https://en.wikipedia.org/wiki/XML#cite_note-7) such as those used in [web services](https://en.wikipedia.org/wiki/Web_service).

Several [schema systems](https://en.wikipedia.org/wiki/XML_schema) exist to aid in the definition of XML-based languages, while programmers have developed many [application programming interfaces](https://en.wikipedia.org/wiki/Application_programming_interface) (APIs) to aid the processing of XML data.

## Applications

The essence of why extensible markup languages are necessary is explained at [Markup language](https://en.wikipedia.org/wiki/Markup_language" \o "Markup language) (for example, see [Markup language § XML](https://en.wikipedia.org/wiki/Markup_language" \l "XML" \o "Markup language)) and at [Standard Generalized Markup Language](https://en.wikipedia.org/wiki/Standard_Generalized_Markup_Language).

Hundreds of document formats using XML syntax have been developed, including [RSS](https://en.wikipedia.org/wiki/RSS), [Atom](https://en.wikipedia.org/wiki/Atom_(standard)), [SOAP](https://en.wikipedia.org/wiki/SOAP), [SVG](https://en.wikipedia.org/wiki/SVG), and [XHTML](https://en.wikipedia.org/wiki/XHTML). XML-based formats have become the default for many office-productivity tools, including [Microsoft Office](https://en.wikipedia.org/wiki/Microsoft_Office) ([Office Open XML](https://en.wikipedia.org/wiki/Office_Open_XML)), [OpenOffice.org](https://en.wikipedia.org/wiki/OpenOffice.org) and [LibreOffice](https://en.wikipedia.org/wiki/LibreOffice) ([OpenDocument](https://en.wikipedia.org/wiki/OpenDocument)), and [Apple](https://en.wikipedia.org/wiki/Apple_Computer)'s [iWork](https://en.wikipedia.org/wiki/IWork)

XML has also provided the base language for [communication protocols](https://en.wikipedia.org/wiki/Communication_protocol) such as [XMPP](https://en.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol). Applications for the [Microsoft](https://en.wikipedia.org/wiki/Microsoft) [.NET Framework](https://en.wikipedia.org/wiki/.NET_Framework) use XML files for configuration, and [property lists](https://en.wikipedia.org/wiki/Property_list) are an implementation of configuration storage built on XML.

Many industry data standards, such as [Health Level 7](https://en.wikipedia.org/wiki/Health_Level_7), [OpenTravel Alliance](https://en.wikipedia.org/wiki/OpenTravel_Alliance" \o "OpenTravel Alliance), [FpML](https://en.wikipedia.org/wiki/FpML" \o "FpML), [MISMO](https://en.wikipedia.org/wiki/MISMO), and [National Information Exchange Model](https://en.wikipedia.org/wiki/National_Information_Exchange_Model) are based on XML and the rich features of the XML schema specification. Many of these standards are quite complex and it is not uncommon for a specification to comprise several thousand pages. In publishing, [Darwin Information Typing Architecture](https://en.wikipedia.org/wiki/Darwin_Information_Typing_Architecture) is an XML industry data standard. XML is used extensively to underpin various publishing formats.

XML is widely used in a [Service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture) (SOA). Disparate systems communicate with each other by exchanging XML messages. The message exchange format is standardised as an [XML schema](https://en.wikipedia.org/wiki/XML_schema) (XSD). This is also referred to as the canonical schema. XML has come into common use for the interchange of data over the Internet. [IETF](https://en.wikipedia.org/wiki/History_of_the_Internet#Internet_Engineering_Task_Force) [RFC:3023](https://tools.ietf.org/html/rfc3023), now superseded by [RFC:7303](https://tools.ietf.org/html/rfc7303), gave rules for the construction of [Internet Media Types](https://en.wikipedia.org/wiki/Internet_media_type) for use when sending XML. It also defines the media types application/xml and text/xml, which say only that the data is in XML, and nothing about its [semantics](https://en.wikipedia.org/wiki/Semantics).

**Android Studio**

The most important part for making an Mobile Application is the **ANDROID STUDIO**.

Android studio is the official integrated development environment (IDE) for Google’s Android operating system, build on JetBrains’s IntelliJ IDEA software and designed specifically for Android development.  It is available for download on [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux) based operating systems. It is a replacement for the [Eclipse Android Development Tools](https://en.wikipedia.org/wiki/Eclipse_(software)#Android_Development_Tools) (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the [Google I/O](https://en.wikipedia.org/wiki/Google_I/O) conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, [Kotlin](https://en.wikipedia.org/wiki/Kotlin_(programming_language)) replaced [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) as Google's preferred language for Android app development.[[13]](https://en.wikipedia.org/wiki/Android_Studio#cite_note-13) Java is still supported, as is [C++](https://en.wikipedia.org/wiki/C%2B%2B).

## Features

The following features are provided in the current stable version:

* [Gradle](https://en.wikipedia.org/wiki/Gradle)-based build support
* Android-specific [refactoring](https://en.wikipedia.org/wiki/Code_refactoring) and quick fixes
* [Lint](https://en.wikipedia.org/wiki/Lint_(software)) tools to catch performance, usability, version compatibility and other problems
* [ProGuard](https://en.wikipedia.org/wiki/ProGuard_(software)) integration and app-signing capabilities
* Template-based wizards to create common Android designs and components
* A rich [layout editor](https://en.wikipedia.org/wiki/Graphical_user_interface_builder) that allows users to drag-and-drop UI components, option to [preview layouts](https://en.wikipedia.org/wiki/WYSIWYG) on multiple screen configurations[[17]](https://en.wikipedia.org/wiki/Android_Studio#cite_note-17)
* Support for building [Android Wear](https://en.wikipedia.org/wiki/Android_Wear) apps
* Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine[[18]](https://en.wikipedia.org/wiki/Android_Studio#cite_note-18)
* Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of [IntelliJ](https://en.wikipedia.org/wiki/IntelliJ) (and [CLion](https://en.wikipedia.org/wiki/CLion" \o "CLion)) e.g. [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), and more with extensions, such as [Go](https://en.wikipedia.org/wiki/Go_(programming_language)); and Android Studio 3.0 or later supports [Kotlin](https://en.wikipedia.org/wiki/Kotlin_(programming_language)) and "all Java 7 language features and a subset of Java 8 language features that vary by platform version."[[21]](https://en.wikipedia.org/wiki/Android_Studio#cite_note-21) External projects [backport](https://en.wikipedia.org/wiki/Backporting) some Java 9 features. While IntelliJ states that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.[[23]](https://en.wikipedia.org/wiki/Android_Studio#cite_note-23)

## Chapter 4

**LEARNING OF ARDUINO**

For this project we needed Arduino and we were unaware about how the Arduino works and how to code. So, we made a team of three people to study Arduino and apply it in our project**.**

Before heading towards Arduino we started reading what is Arduino, how actually it works, its codes, etc. We started searching on internet about it. After a long search we came across a website i.e. [www.tutorialspoint.com](http://www.tutorialspoint.com) which was a very good platform to learn Arduino. And also we read different books on programming in Arduino. We headed towards our learning procedure. Firstly, we read an introduction of Arduino that it is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board. After that we learned about the features of Arduino like

* Arduino boards are able to read analogue or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
* You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
* Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.
* Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
* Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

After this we started our coding in Arduino and learned its data types, and also some function libraries like input output functions, character functions, math library and trigonometric functions. After learning all this we tried some projects which were essential for our project like Blinking Led, Fading Led, Keyboard Message, etc. from this website. We also learned about the network communication so that it the Bluetooth module can be synchronized with the application that is developed by out group.

## Hence, we completed our learning and we were ready to assemble the parts.

## 

**10.Coding Practice for ARDUINO**

## 

## Chapter 5

**LEARNING OF ANDROID STUDIO**

For this project we needed Android Studio and we were unaware about how the Android Studio works and how to code. So, we made a team of three people to study Android Studio and apply it in our project**.**

Before heading towards Android Studio we started reading what is Android Studio, how actually it works, its codes, etc. We started searching on internet about it. After a long search we came across a website i.e. [www.tutorialspoint.com](http://www.tutorialspoint.com) and [www.javapoint.com](http://www.javapoint.com) which are very good platform to learn Android Studio. We headed towards our learning procedure.

At first we had learned about the basics of Android Studio. Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

* Android app modules
* Library modules
* Google App Engine modules

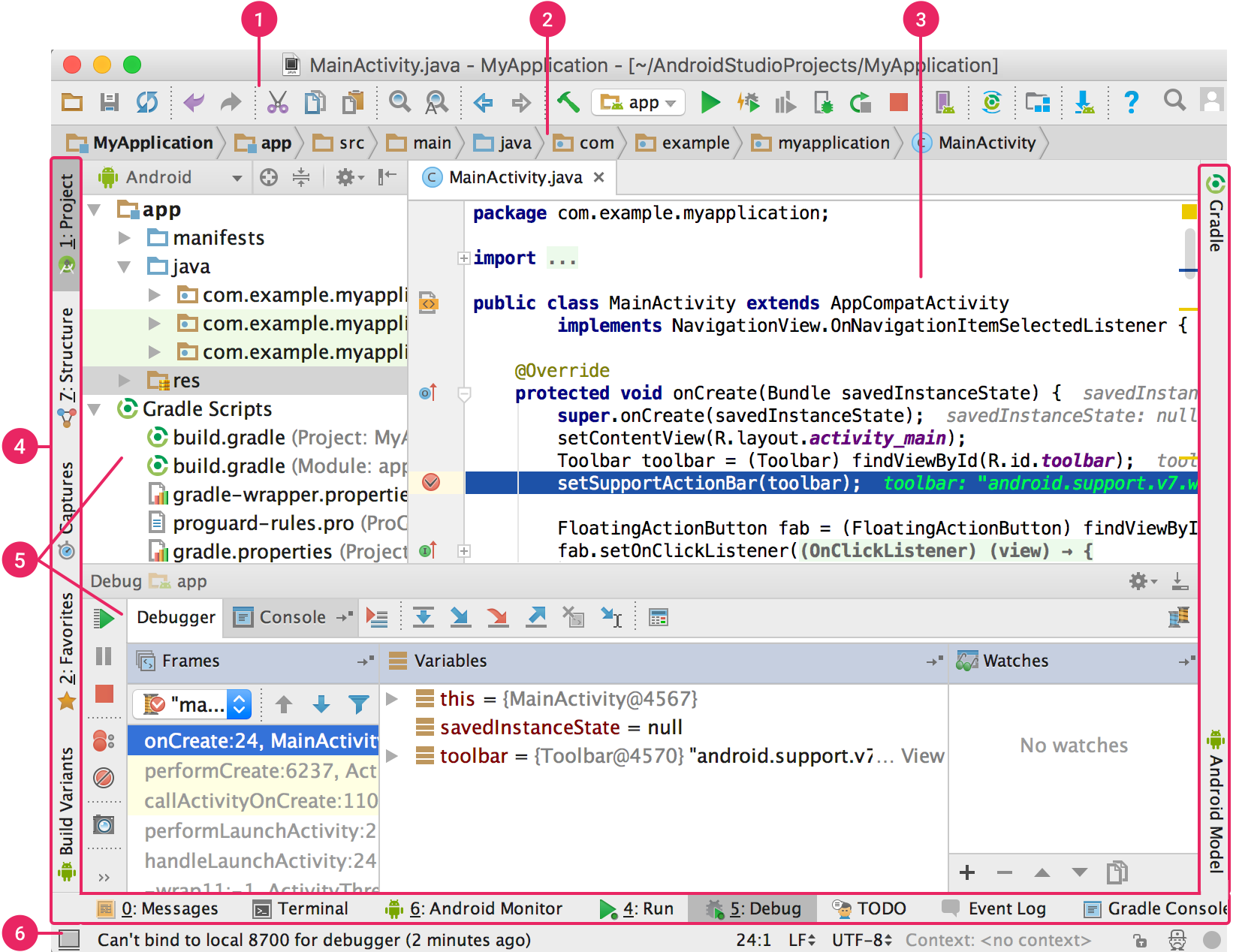
By default, Android Studio displays our project files in the Android project view, as shown in figure below. This view is organized by modules to provide quick access to our project's key source files.

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

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

The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.

1. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
2. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
3. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
4. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
5. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.

****

***11.THE USER INTERFACE***

After learning all the basics of the android studio , we started with an our first hello world sample app. After building that first app we ran it on our mobile phone , which ran successfully.

## Chapter 6

**APP DEVELOPMENT**

## On moving towards developing an app which can save data entered by user to the mobile phone of user as well as transfer the data with the help of Bluetooth this was not the easy task for us. So we completed the App Development in these following steps:

## Splash Screen

## First Time User Intro

## Device Mac Address

## Home screen

## Input Data Screen

## User Guide

## Feedback

## 1]. Splash Screen

## This is the app starting screen which is shown to the user when app starts. This screen is visible for 2.5 seconds only and after that the user get move on to the next screen which is

## ( First Time User Intro) .

## 

## *12.* *Splash Screen*

## 2]. First Time User Intro

## This Screen shows a small intro about this app which helps the user to know about this app. The Intro screen consists of INPUT DATA, CONNECT TO DEVICE, SHARE DATA.

## This is the screen which appears Only once when the App is installed and opened. After reopening or opening second time that app this screen will not be visible. Hence this is only one time visible screen.

## 

## 

## *13.First Time User Intro*

## 3]. Device MAC Address

## Device MAC Address is the address of the Bluetooth Module of Aurdino . MAC Address is different for every Bluetooth Module. User will be provided with MAC address of the Bluetooth Module that user have to enter in this screen.

## 

## *14.Device MAC Address*

## 4]. Home Screen

## Home Screen shows the different Box no. to the user to select their respective box.

## Every box layout have a “ ADD DATA “ button . On clicking the button the screen get move to the another screen named INPUT Screen .

## 

## *15.Home Screen*

## 5]. Input Data Screen

## Input data screen consists of

## Course Till

## Time

## Medicine Name

## Dose

## Save data , Clear data

## a). When user enters the course till days and click on the down arrow then the start date and end date of course appears.

## b). User can add max upto three time in a day by clicking on ADD MORE button.

## c). After entering all data user can save data to mobile device. After that for connection with Aurdino the MAC Address should be filled if not then it will show a message to user and if it is already filled then it will start establishing a secure connection with Bluetooth module.

## d). After a connection user can share the data with Aurdino.

## 

## *16.Input Data Screen*

## 

## *a). b). c).*

## 6]. User guide

## 

## User can go to user guide by clicking the menu bar (top left corner). User guide basically have a small guidance for using this app such as connection, save data ,etc.

## 

## *17.User Guide*

## 7]. Feedback

## User can provide feedback about this app or any issue related to this app such as cannot establish a connection , any data issue , app bug , etc.

## 

**18.Feedback**

## 

## Chapter 7

**ASSEMBLING THE PARTS OF ARDUINO**

Arduino Nano has 22 digital input/output pins. 5 buttons, 1 buzzer, 1 LED, 1 LCD screen, Bluetooth module and one Real-Time-Clock will be connected to these digital input/output pins of the Arduino Nano using male/female connecting wires.

For the bluetooth module the Tx(transmission) of the module goes to the Rx(recieve) port of the arduino (Rx19 port) and the Rx port of the bluetooth module goes to the Tx port of the arduino (Tx18 port) and the bluetooth modules VCC(+5V) goes to the VCC(+5V) of the Arduino and ground(GND) goes to the ground of the arduino.

For the real time clock(RTC) module the VCC(+3.3V) goes to the VCC(+3.3V) of the arduino, the SDA port of the RTC module goes to the SDA20 port of the arduino and SCL port of the RTC module goes to the SCL21 port of the arduino.

For the LCD screen the VCC(+5V) and GND goes to the VCC and VCC of the arduino. The VO port of the LCD screen goes to the potentiometer, RS port of the LCD screen goes to the 12 port of the arduino, RS port of the LCD screen goes to the 12 port of the arduino, E port of the LCD screen goes to the 11 port of the arduino, RS port of the LCD screen goes to the 12 port of the arduino, DB5 port of the LCD screen goes to the 8 port of the arduino, K port of the LCD screen goes to the GND port of the arduino, R\_W port of the LCD screen goes to the GND port of the arduino.

For all LEDs one connection is to the GND and the other connection goes to port 2,3,4,5,6 respectively for each LED and they are also connected with a resistor (220 ohm).

For the push button one connection goes to the ground of the arduino and the other connection goes to the 13 port of the arduino , it is also connected with a resistor.

The arduino is given a main power suppy,it can receive power from either main or a battery pack,it has a USB B port to transmit and receive information from the computer it is connected to.Potentio-meter is also required to control the LCD screen.The entire circuit is assembled on a bread board. A screenshot of a computer

Description automatically generated

19.Final Diagram

## Chapter 8

## My Work

## When we are making the EMB application we deal with make difficult parameters and this is very challenging for us because nobody in our group is having experience in this field. But we decide to make an application for the project so it looks more effective and comfortable to the user.

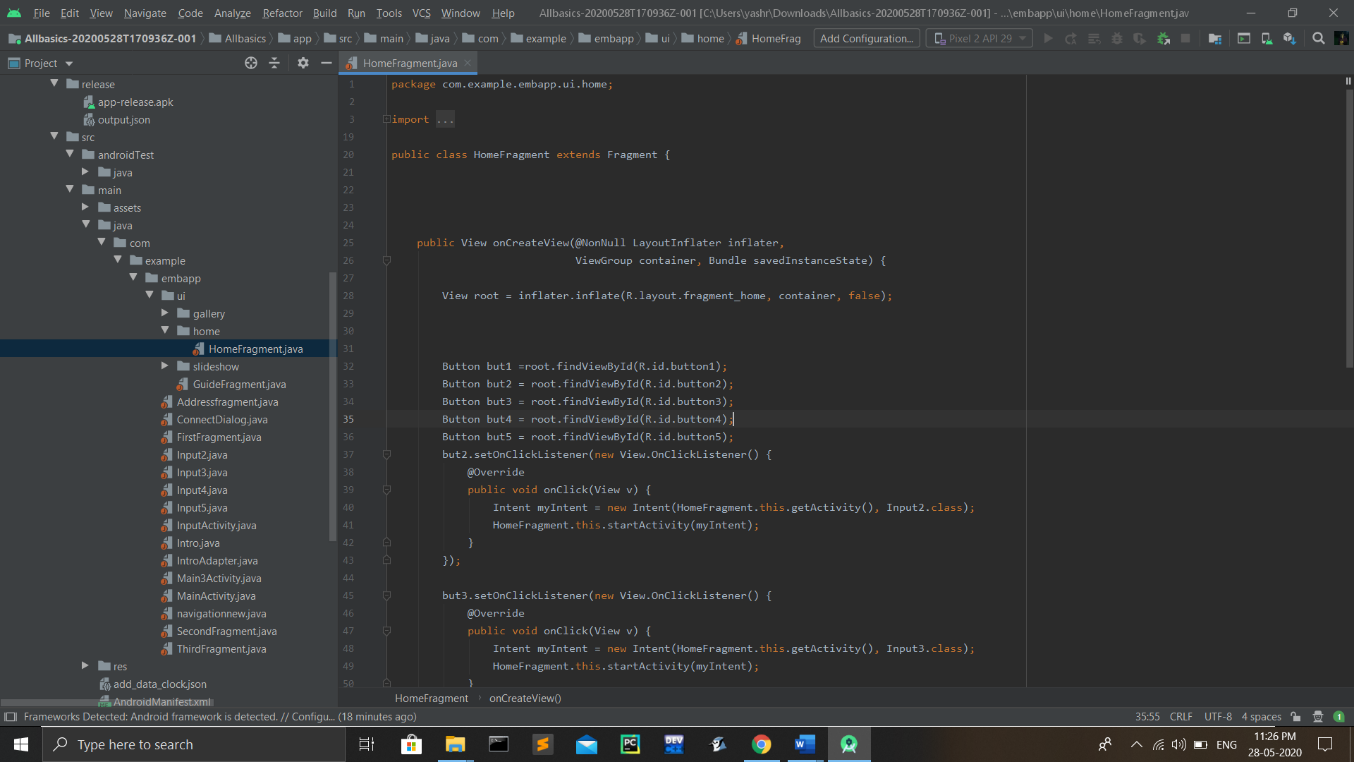
## I work as a full stack developer who works both the front and back ends of the application. I know JAVA which we are going to use so Myself, Manoj, and Vedant which are part of the making android application start learning more JAVA and then we learn XML for the front in development. Then we divide the work into some parts and start doing.

## I decide to make HOME PAGE as it is the main and important page for the application then followed by ABOUT US, GUIDE, CLEAR DATA, etc.

**1]. HOME PAGE :-**

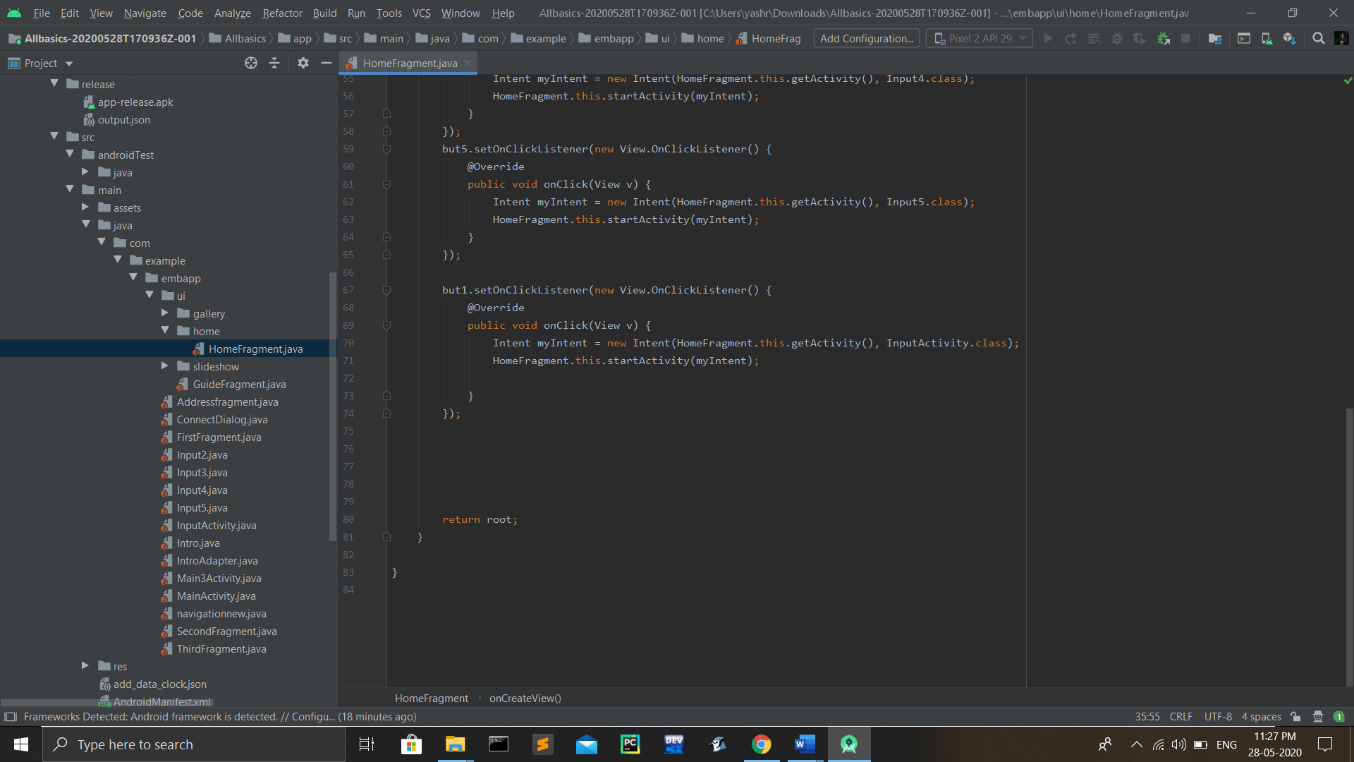
**a).** The home page is the main page of any application. When we open the application the first screen which we saw is the Home Page. It contains the other options to access like in our case the box details, the menu button for accessing feedback, about us, guide, etc.

**b).** The first thing is to think about the structure of the Home Page then implement it on the .xml form or we can say that create it on the Android Studio.



**Fig.01**

**c).** Then the second thing is to make the menu button for accessing the feedback, guide, about as and the device option for the EMB application.

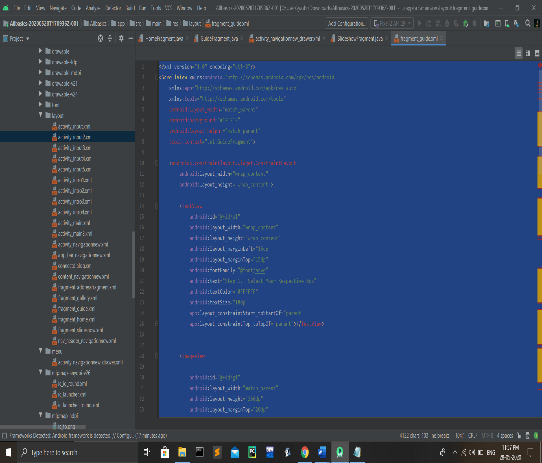
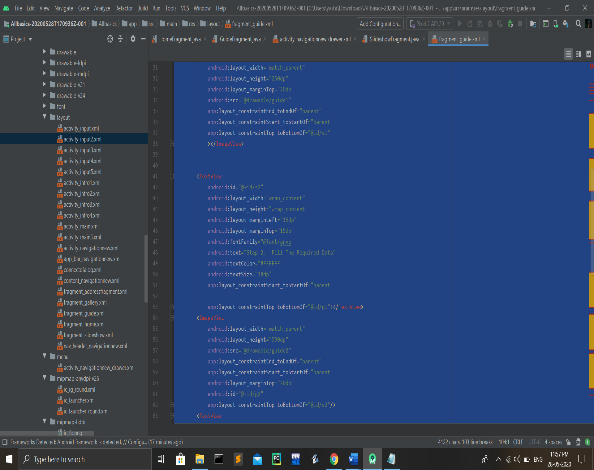


**Fig.02**

**2]. GUIDE :-**

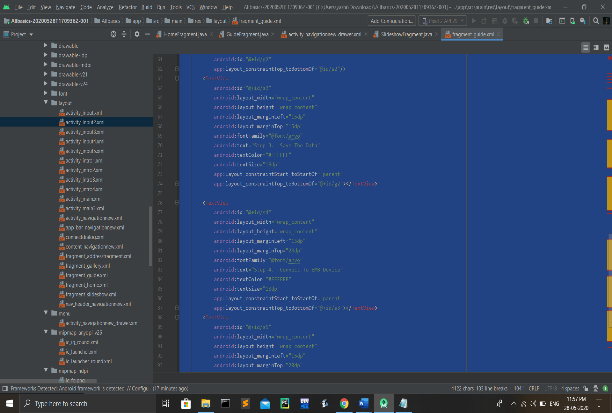
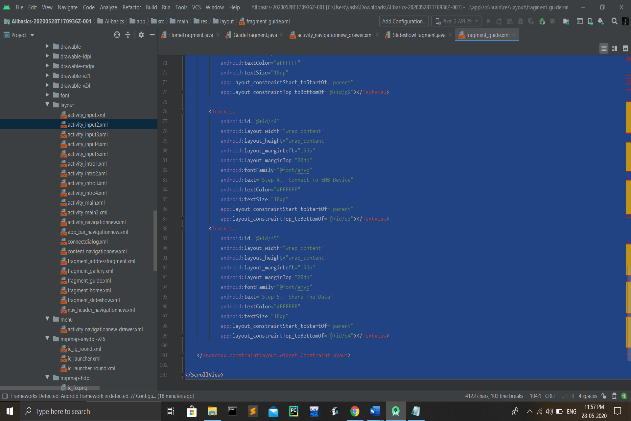
**a).** The guide option is important because if the new person is using the application, it's hard for him to use but if he knew how the application works so it can easily use it.

**b).** First working with the interface of the guide option and putting the different instructions with the pictures and showing different parameters and steps for the proper guidance.

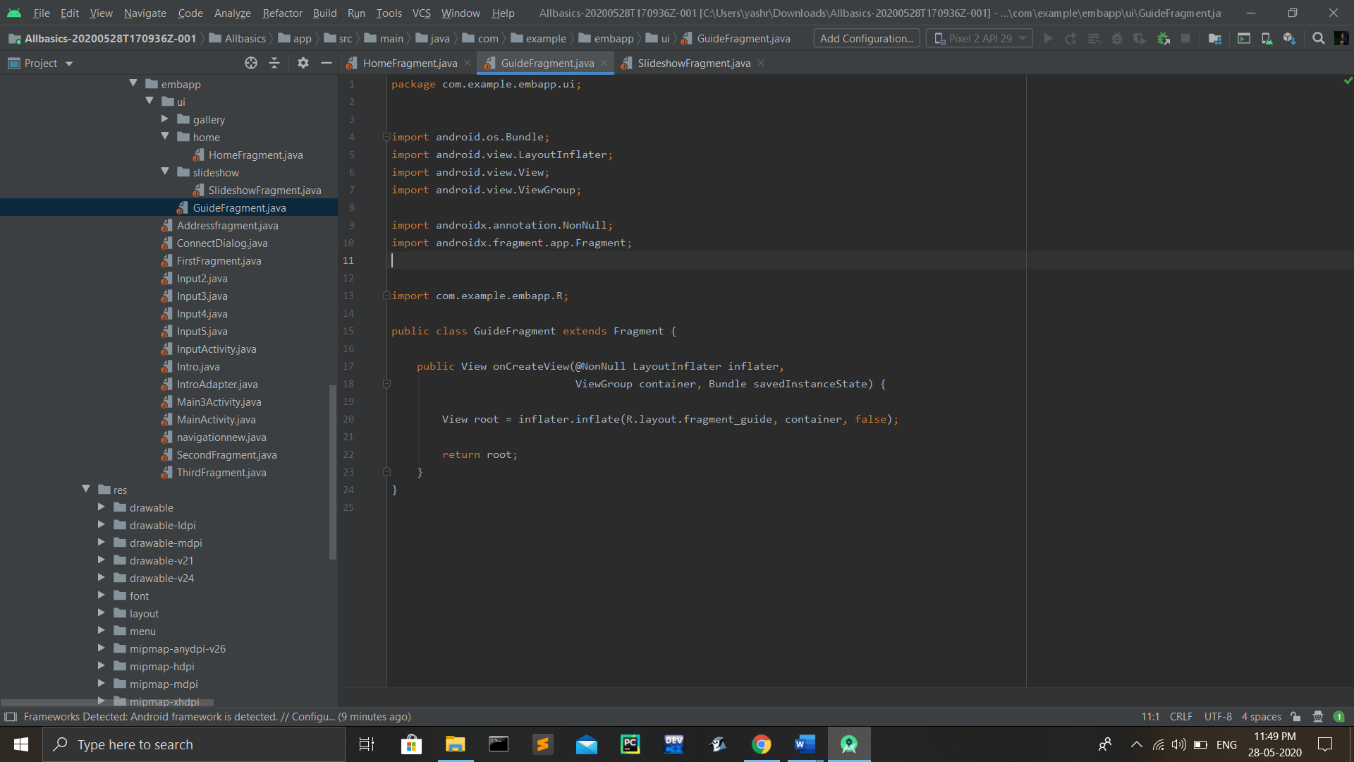
**Fig.03 Fig.04**

**c).** Measuring all the information and creating a proper guide tool interference and arranging it step by step order.

**Fig.05 Fig.06**

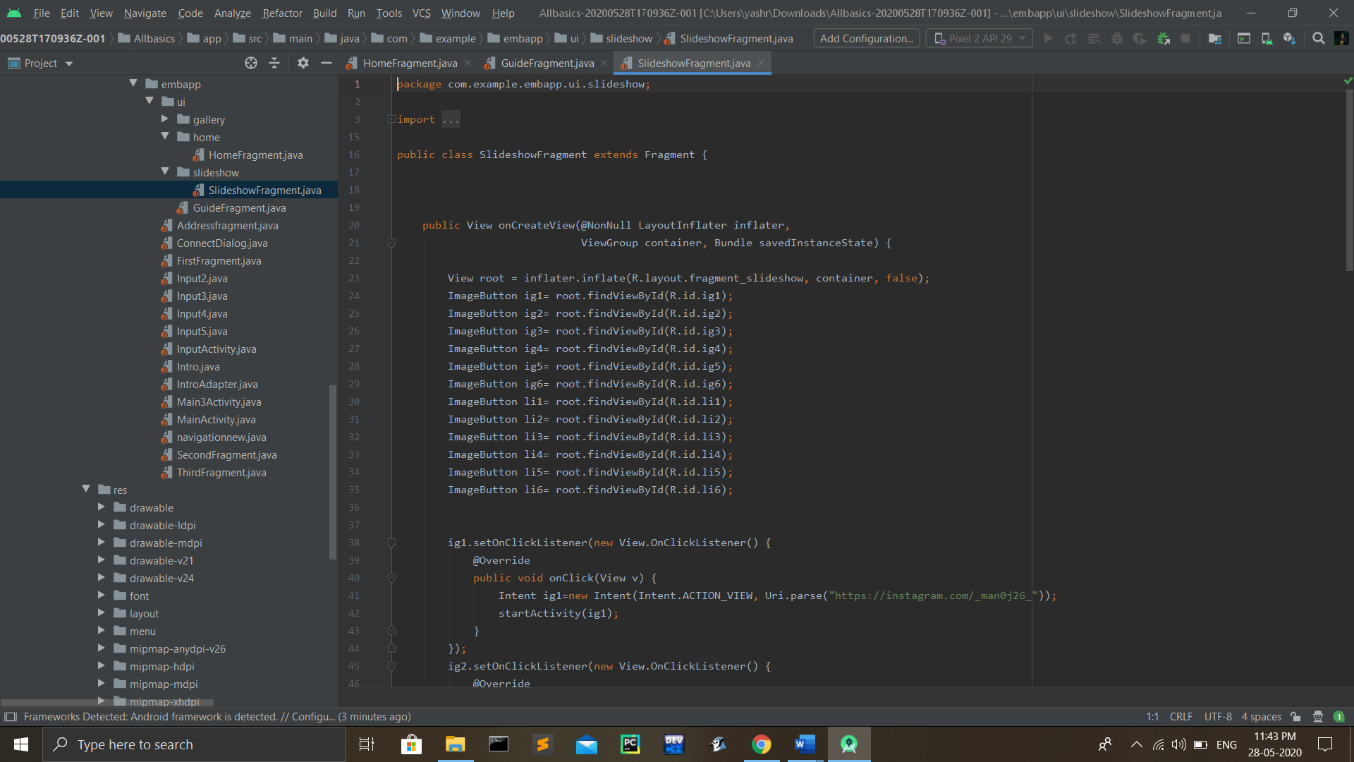
**d).** Now, the front end work is completed and we have to do the back end development in java and then the back end is connected with the front end to control the functioning of the key.



**Fig.07**

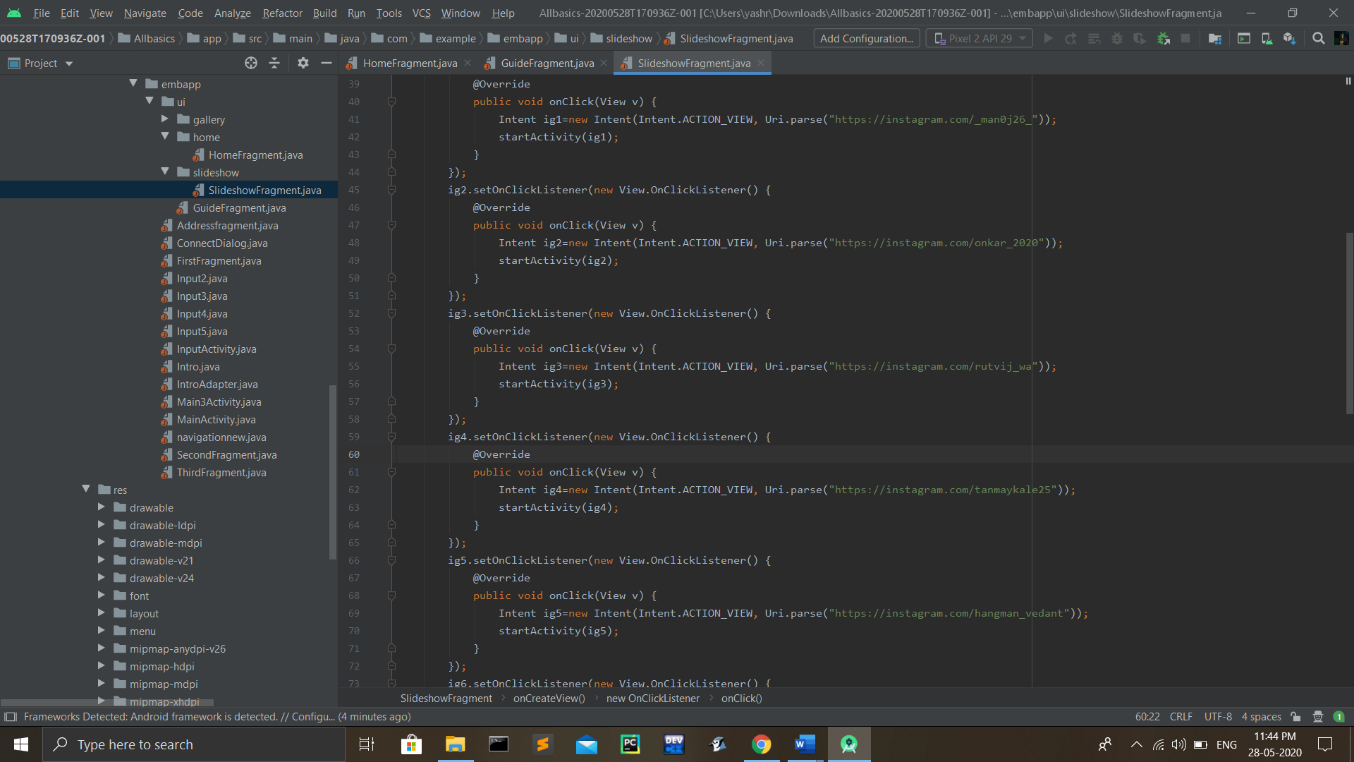
**3]. ABOUT US :-**

**a).** The About Us page is created same as the guide page which is open through the menu button. This page is for giving information about all of us. I also linked the Instagram account and the linked in page for better perceive which is directly open from the EMP application.



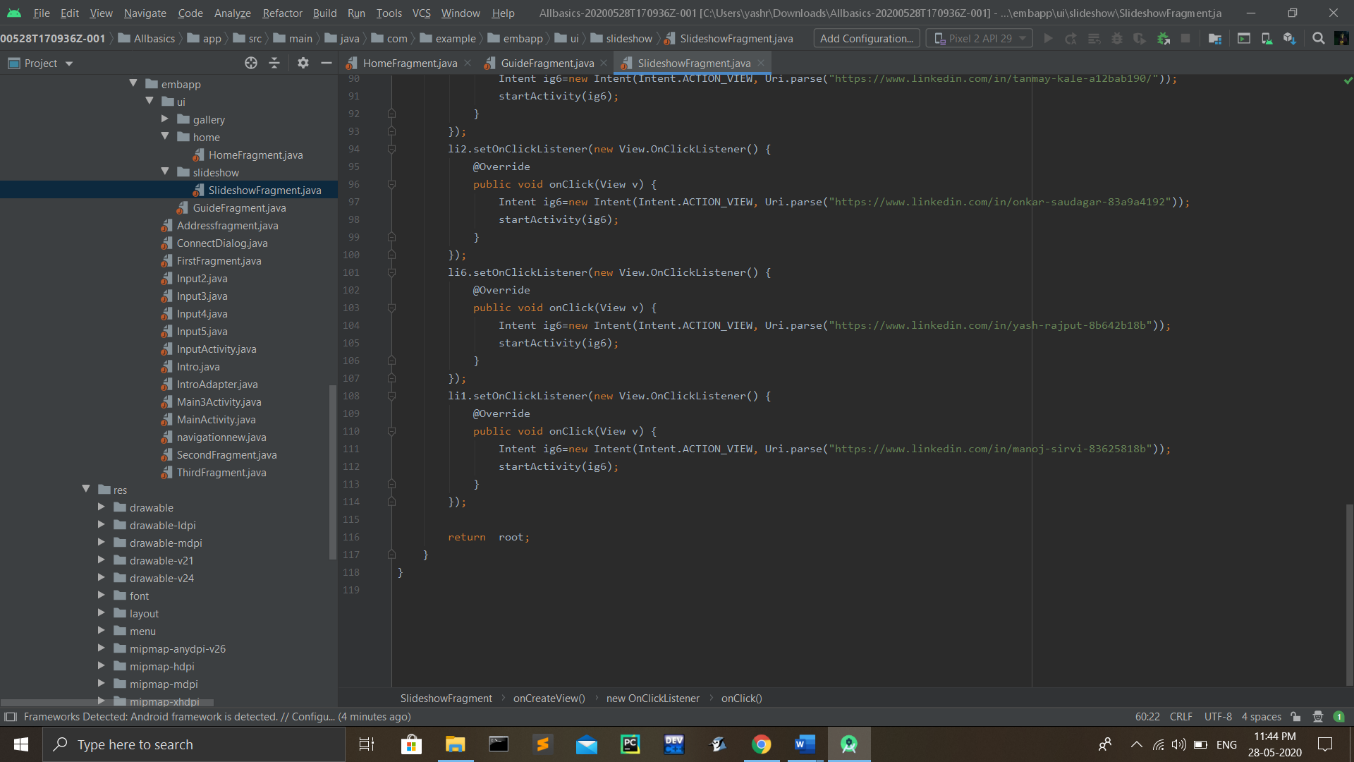
**Fig.08**

**b).** Then I think about the design of front end of the About Us page and created with the help of XML. Then I start working on the back end and connected with the front end.

****

**Fig.09**

**c).** The most difficult part is to attach the Linked in profile and the Instagram page which is going to open directly. Means when we click on the Instagram or the Linked in button then we redirected on the given application.

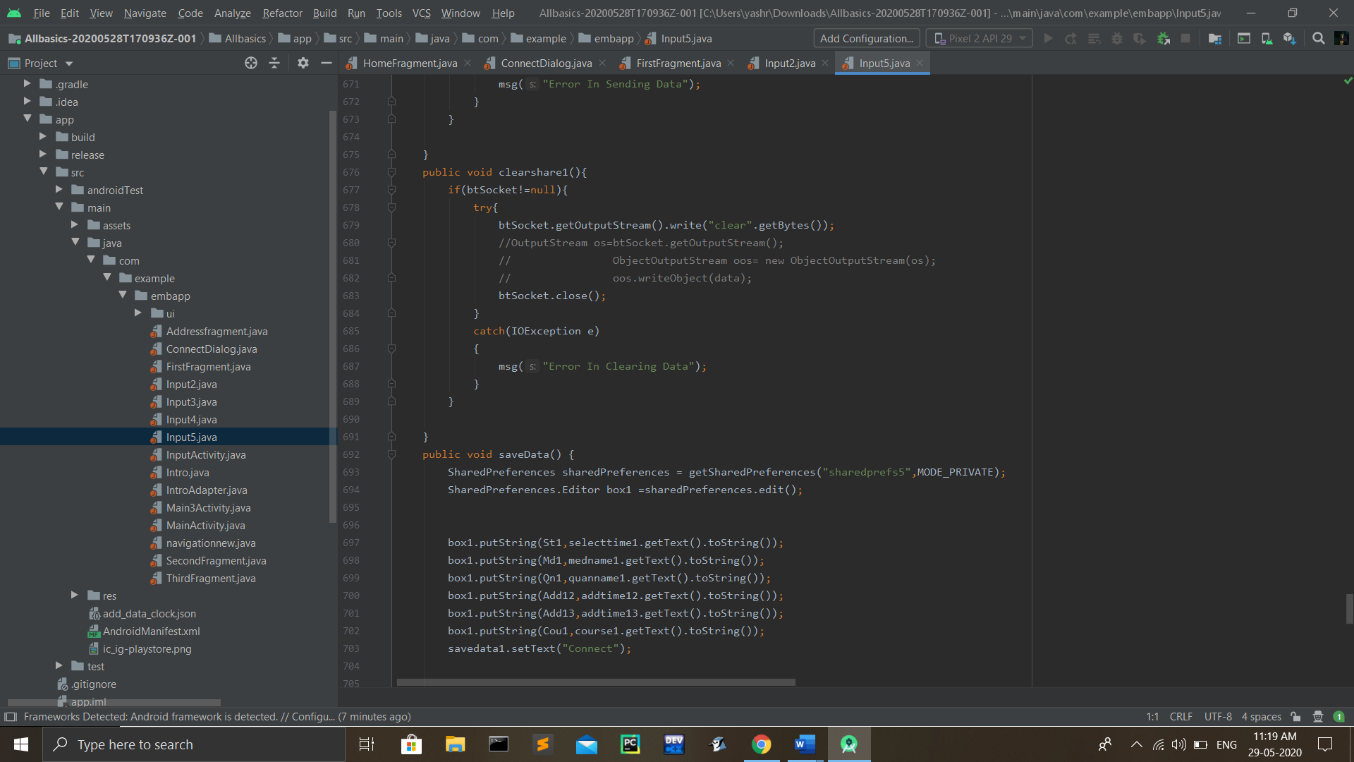
****

**Fig.10**

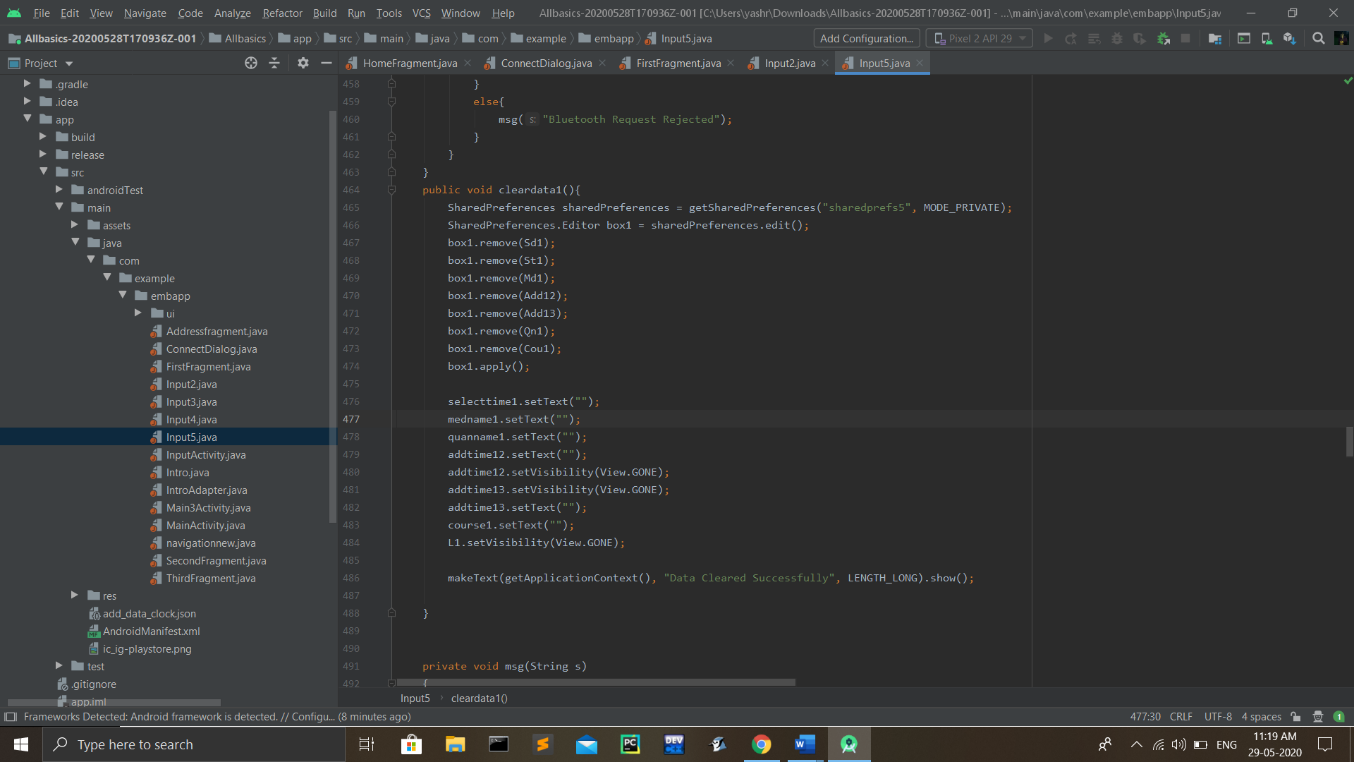
**4]. CLEAR DATA :-**

**a).** Clearing the data is the one most important part of the application and for the whole project including the Aurdino. The data which we take from the user is saved in the form of array in the memory of the mobile then we connect the mobile with the Aurdino through the Bluetooth and transfer the same array which contains the data like time, data, no.of medicine, etc.

**b).** The Aurdino works on the given data but if we have to re-enter the data then we need to erase the data from the android application as well as we should delete the data from the Aurdino and then taking it back.

****

**Fig.11**

****

**Fig.12**

**5]. CONNECTION WITH AURDINO :-**

**a).** The array of data is needed to transfer from Mobile to Aurdino. For this purpose we choose Bluetooth, first we have to establish a proper and secure connection between the devices so manoj and I work on the Bluetooth socket which connects with the Bluetooth module of Aurdino.

b). After the connection is established with the Bluetooth module, the user input data will be transferred in the form of an array. And after the transferring of the data, the Bluetooth socket must be closed.  **Fig.13**

## 

## 

## Chapter 9

**CONCLUSION**

An E-Medical Box illuminates the elders to take medication. It productively controls the season of senior citizens to take medication. It additionally diminishes the proportion that patient misses and defers taking medication. The remote user interface joins with the RoboRemo software programming so that the parental figures can help the patient. Which adds more functionality by applying more usability through networking locally or using the internet if it is available. The outline design is too appropriate for the medication bundles. Later on, we trust that the vitality sparing and compact can be considered. In this way, missing and deferring taking solution can be totally eliminated.

We successfully the goal of our project is to provide healthy and tension free life to those users who are taking regularly pills and to provide this product at affordable cost also.

**REFERENCES**

**ARDUINO**

1. Exploring Arduino: Tools and Techniques for Engineering Wizardry

Book by Jeremy Blum

1. Arduino Workshop: A Hands-On Introduction with 65 Projects

Book by John Boxall

1. Arduino Cookbook

Book by Michael Margolis

1. Arduino Programming in 24 Hours, Sams Teach Yourself

Book by Richard Blum

1. Arduino Programming: The Ultimate Beginner’s Guide to Learn Arduino Programming Step by Step

Book by Ryan Turner

**APP DEVELOPMENT**

1. Android Developers documentation <https://developer.android.com/docs>
2. Lottie animation documentation <https://airbnb.io/lottie/>
3. Java documentation <https://docs.oracle.com/en/java/>