Note Android App

By :- Bhavin Bhesaniya

A

MINOR PROJECT REPORT

ON

Note Mobile Application

IN

Android

Submitted in partial fulfilment of the requirement

For the award of the degree of

Bachelor Of Computer Application(2019-2022)

Submitted To :

B.K.N.M.U

Submitted By :

Bhavin Bhesaniya

Under The Supervision Of :

Mr. Chintan Kakkad

Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I would like to express my sincere thanks to **Mr. Chintan Kakkad Sir**, for his/her valuable guidance and support in completing my project. I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

We are indebted to all the people of our collage, S.S.S.D.I.I.T, concerned, who have actively involved in marking of the project. Last but not the least we would thanks the god who has made us to do the project & also our parents without their support we would not be able to complete the project successfully.

Place : Junagadh

Date : 10/3/22

Index

Project Profile

Project Title	:	NoteApp
Front-End	:	Android
Back-End	:	Room Database
Time Duration	:	1 Month
Developed By	:	Bhavin Bhesaniya
Project-Guide	:	Mr. Chintan Kakkar
Submitted To	:	B.K.N.M.U.

Requirement Gathering

System Analyses :-

System Analyses is the collecting all the information from all the student then collect all information to obtain a clean and through understanding of the places to be developed with view to removing all ambiguities and inconsistencies from the initial student of the problem

- ⇨ Questionaries
- ⇨ Interview
- ⇨ Observation
- ⇨ Record Reviewing

Identification Of Need :-

The daily activities pointing to smooth running of a Service Centre involves recording of large volume of data in record books. The management of student need to be managed properly. In addition, timely preparations of several reports are required. All these require managing services quickly and efficiently.

Preliminary Investigation :-

A request to receive assistance from the information system can be in made for many reasons. But in each case the requested when that request is made, the first system activity the preliminary investigation begins. Collecting the information in this phase through personal interviews.

Questionaries :-

Q-1. Which type of application do you want to develop?

- Android Application

Q-2. What will the purpose of this Android application?

- User make their own note with reminder so they don't forgot their task

Q-3. Do You have any previous application?

- No

Q-4. Tell me functionality that you want in your application?

- 1) Home Screen with View And Search with filter functionality
  2) Add Note with validation, Update note and delete note
  3) Send Notification To User Base on their set time

Q-5. You need any payment gateway method in your application?

- No

Q-6. How much time you want to complete this application?

- 15 days to 1 Month

Q-7. Give title of your application?

- NoteApp

Q-8. Do you have any logo?

- Yes

Q-9. Which user can Access this application?

- Everyone

Q-10. What is the budget for your application?

- 10,000 to 12,000

Project Cost Estimation :-

Line Of Code :-

As the name suggests, LOC count the total number of lines of source code in a project.

Units of LOC are :-

- KLOC- Thousand lines of code
- NLOC- Non-comment lines of code
- KDSI- Thousands of delivered source instruction

Advantages :-

- Universally accepted and is used in many models like COCOMO.
- Estimation is closer to the developer's perspective.
- Simple to use.

Disadvantages :-

- Different programming languages contain a different number of lines.
- No proper industry standard exists for this technique.
- It is difficult to estimate the size using this technique in the early stages of the project

Feasibility Study

Technical Feasibility :-

The main aim for technical feasibility study is to determine whether it is possible to develop the proposed system with the present technologies available and study the technical requirement.

Technical requirement :-

Hardware	Software
- 8 GB RAM	- Android Studio 20.1
- 10 GB Storage	
- Intel i5 10th Gen.	

Economical Feasibility :-

The company already possesses the required hardware and software. There is no investment in hardware and software. The benefits of installing the application lie in the speedy processing of data, faster retrieval of information and increasing volume of data, and all these with greater accuracy and consistency. To sum up, the benefits are great and cost is minimal. Therefore, the project is economically feasible.

Operational Feasibility :-

The system is expected to work smoothly when developed and installed. There has been participation of management and the computer operators in planning and development of the system. There will be a slight change in the format of the reports to which management agreed. There is no disturbance in organizational structure of the company. The new application for the system will have Graphical User Interfaces (GUI). The applications with GUI are very easy to handle and operate. They need to be instructed regarding using of the application software.

Technology Used

Tools / Environment Used :-

Software :-

Platform	Android
IDE(Integrated Development Environment)	Android Studio Artic Fox 20.2
RDBMS (Database)	Android Room Database(SQLite)
JDK(Java Development Kit)	11

Hardware :-

Processor	Intel i5 10th Generation
RAM	8GB DDR4
Laptop	HP 15 3047
Hard Disk	256 GB SSD

Software Requirement To Run Application :-

- Android OS Above Lollipop 5.0 Version

System Requirement :-

Minimum Requirement :- 2GB RAM / Android 5.0 V / 10MB Disk Space

Recommended :- 2GB RAM / Android 8.0 V / 50MB Disk Space

Language and Software Description

In this project we develop android app so we used Java as programming language, Android Studio as IDE to develop project and Android Room Database as database.

JAVA :-

Java has been one of the most popular programming languages for many years. Java is Object Oriented. However, it is not considered as pure object oriented as it provides support for primitive data types (like int, char, etc)

The Java codes are first compiled into byte code (machine independent code). Then the byte code runs on **J**ava **V**irtual **M**achine (JVM) regardless of the underlying architecture.

Java syntax is similar to C/C++. But Java does not provide low level programming functionalities like pointers. Also, Java codes are always written in the form of classes and objects.

Java is used in all kind of applications like Mobile Applications (Android is Java based), desktop applications, web applications, client server applications, enterprise applications and many more.

When compared with C++, Java codes are generally more maintainable because Java does not allow many things which may lead bad/inefficient programming if used incorrectly. For example, non-primitives are always references in Java. So we cannot pass large objects (like we can do in C++) to functions, we always pass references in Java. One more example, since there are no pointers, bad memory access is also not possible. When compared with Python, Java kind of fits between C++ and Python. The programs written in Java typically run faster than corresponding Python programs and slower than C++. Like C++, Java does static type checking, but Python does not.

James Gosling in 1995 developed Java, who is known as the Father of Java.

Features Of Java :-

1. Platform Independent
2. Object-Oriented Programming Language
3. Simple
4. Robust
5. Secure
6. Distributed
7. Multithreading
8. Portable
9. High Performance
10. Dynamic flexibility
11. Sandbox Execution
12. Write Once Run Anywhere
13. Power of compilation and interpretation

Android :-

- Open-source software platform and OS for Mobile Device.
- Android is based on Linux kernel.
- Developed by Google and later the Open Handset Alliance(OHA).
- Allow writing manage Code in the Java Language.
- Android has its own machine. i.e. DVM(Dalvik Virtual Machine)

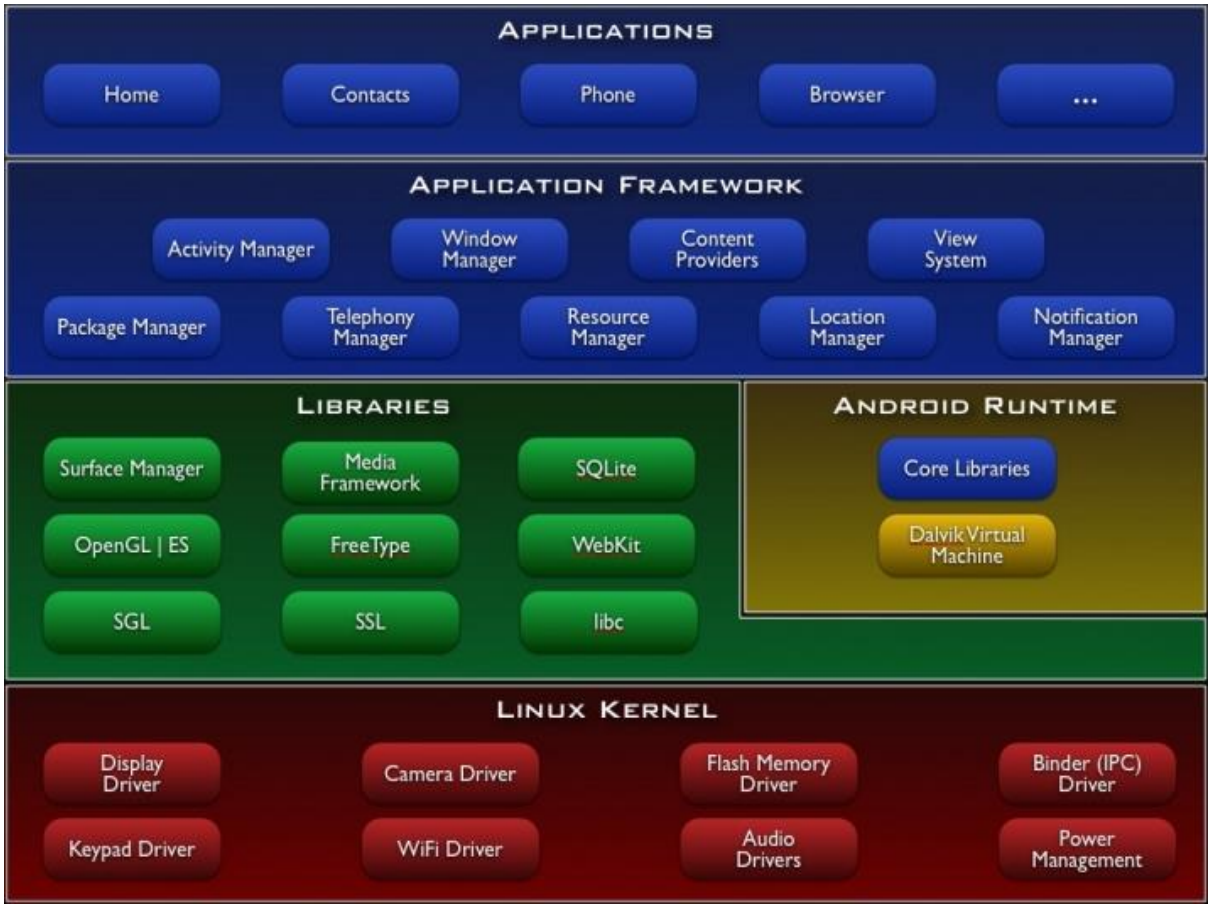
Features Of Android :-

- Background Wi-Fi location still runs even when Wi-Fi is turned off
- Developer logging and analysing enhancements
- It is optimized for mobile devices
- It enables reuse and replacement of component

Android Architecture :-

- The software stack is split into four layers: -

    - Application layer
    - Application framework
    - The libraries and run time
    - The kernel



Linux Kernel :-

- Architecture based on LINUX Kernel
- This layer is core of android Architecture.
- It provides service like power management, Memory Management, Security etc.
- It helps in software or hardware binding for better communication.

Native Libraries :-

- Android has its own libraries, its written in C/C++
- These libraries can't access directly.
- With the help of application framework, we can access this library.

- Android Runtime :
    - android runtime designed specifically for android to meet the needs for running in an embedded environment where you have limited battery, limited memory, limited CPU.

- Davik is the process virtual machine in Google's android operating system.
- It is software that runs the apps on android devices.
- Program are commonly written in java and compiled to byte Code.
- The core library contains all of the collection classes, utilities, IO, all the utilities and tools that your come to expected to use.

Application Framework :-

- Application framework is the toolkit that all application use.
- These applications include the once that come with a phone like home
- It includes application written by google and it include apps that will be

These are as follows:-

Activity manager :-
- It manages the lifecycle of applications.
- It enabled proper management of all the activities.
- All the activities are controlled by activity manager.

Resource manager :-
- It provides access to non-code resources such as graphics etc.

Notification manager :-
- It enables all applications to display custom alerts in status bar.

Location manager :-
- Fires alerts when user enters or leaves specified location.

Package manager :-
- It is use to retrieve the data about installed packages on device.

Window manager :-
- It is use to create views and layouts.

Telephony manager :-
- It is use to handle settings of network connection and all information

Application Layer :-

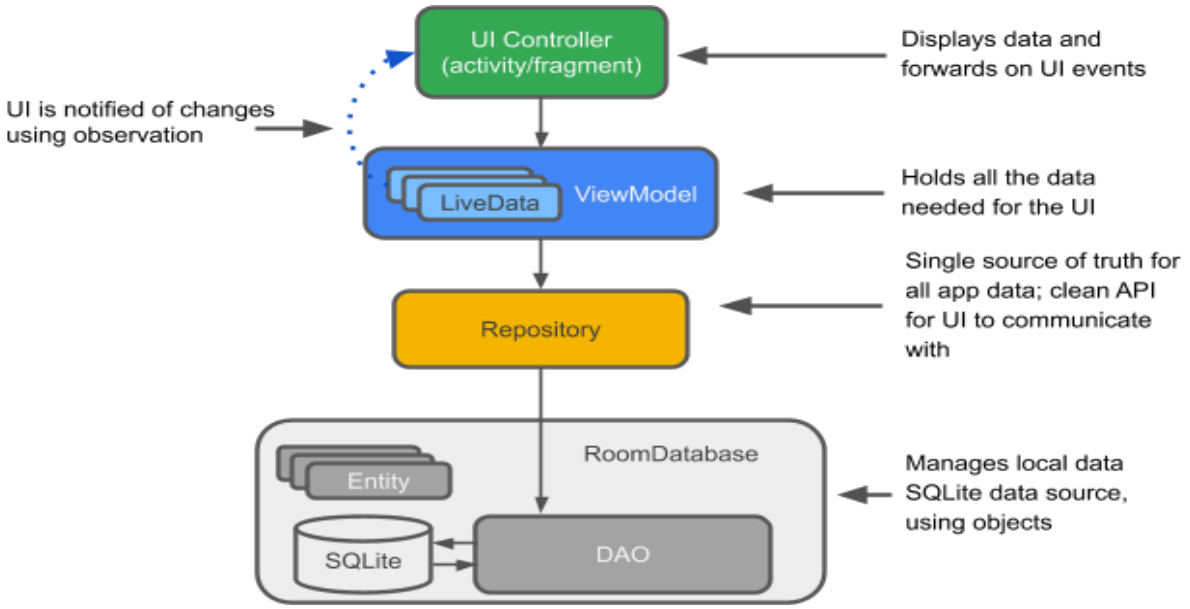
- Final layer on top is Applications.
- Includes home application, contacts application, browser and apps.
- It is the most upper layer in android architecture.

Android Architecture Component :-

- Android operating system provides a strong foundation for building apps that run well on a wide range of devices and form factors.

- Android Architecture Components provide libraries for common tasks such as lifecycle management and data persistence to make it easier to implement the recommended architecture.

- Architecture Components help you structure your app in a way that is robust, testable, and maintainable with less boilerplate code.



Entity :

In the context of Architecture Components, the entity is an annotated class that describes a database table.

SQLite database :

Room persistence library creates and maintains SQLite database for you.

DAO :

Short for *data access object*. A mapping of SQL queries to functions.
You used to have to define these queries in a helper class.
When you use a DAO, your code calls the functions, and the components take care of the rest.

Room database :

Database layer on top of an SQLite database that takes care of mundane tasks that you used to handle with a helper class.
The Room database uses the DAO to issue queries to the SQLite database based on functions called.

Repository :

A class that you create for managing multiple data sources.
In addition to a Room database, the Repository could manage remote data sources such as a web server.

ViewModel :

Provides data to the UI and acts as a communication center between the Repository and the UI.
Hides the backend from the UI.
ViewModel instances survive device configuration changes.

LiveData :

A data holder class that follows the observer pattern, which means that it can be observed. Always holds/caches latest version of data.
Notifies its observers when the data has changed. Generally, UI components observe relevant data.
LiveData is lifecycle-aware, so it automatically manages stopping and resuming observation based on the state of its observing activity or fragment.

What is Room?

- Room is a persistence library that provides an abstraction layer over the SQLite database to allow a more robust database.
- With the help of room, we can easily create the database and perform CRUD operations very easily.

Components of Room?

- Three main components of room are **Entity, Database, and DAO**.

- **Entity**: Entity is a modal class that is annotated with @Entity. This class is having variables that will be our columns and the class is our table.

- **Database**: It is an abstract class where we will be storing all our database entries which we can call Entities.

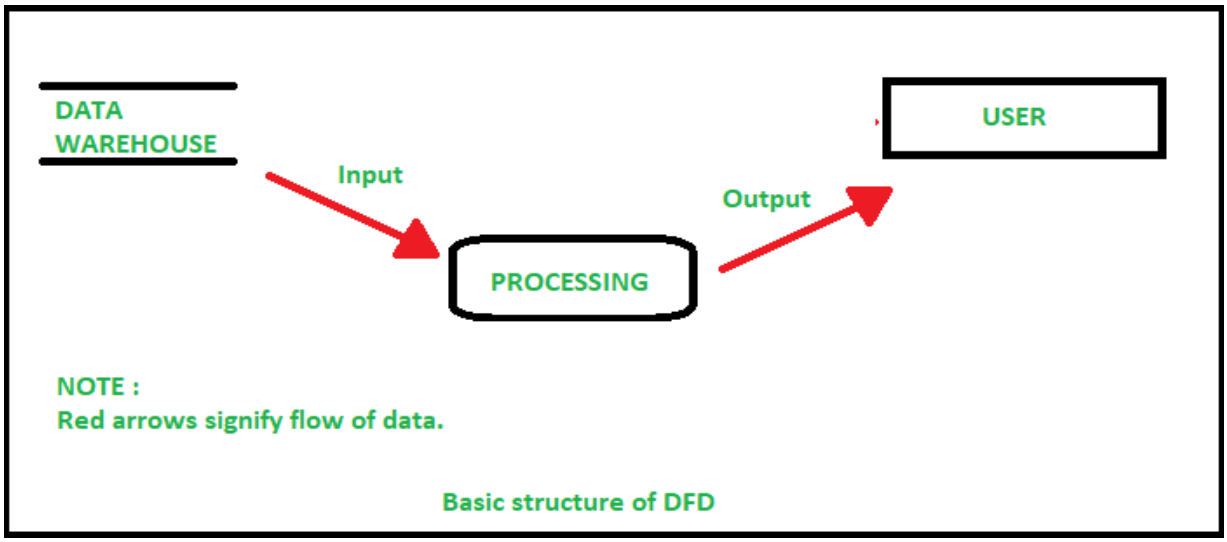
- **DAO**: The full form of DAO is a **Database access object** which is an interface class with the help of it we can perform different operations in our database.

Why Use Room ?

- Compile-time verification of SQL queries. each @Query and @Entity is checked at the compile time, that preserves your app from crash issues at runtime and not only it checks the only syntax, but also missing tables.
- Boilerplate code
- Easily integrated with other Architecture components

Data Flow Diagram

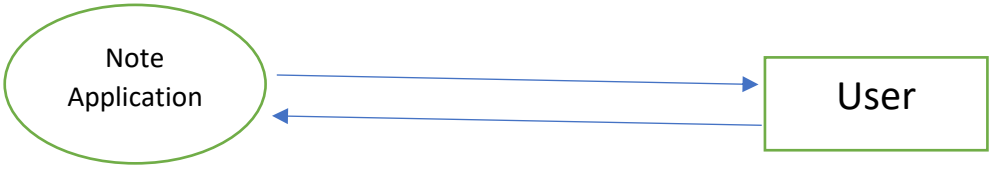
DFD is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modelling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.



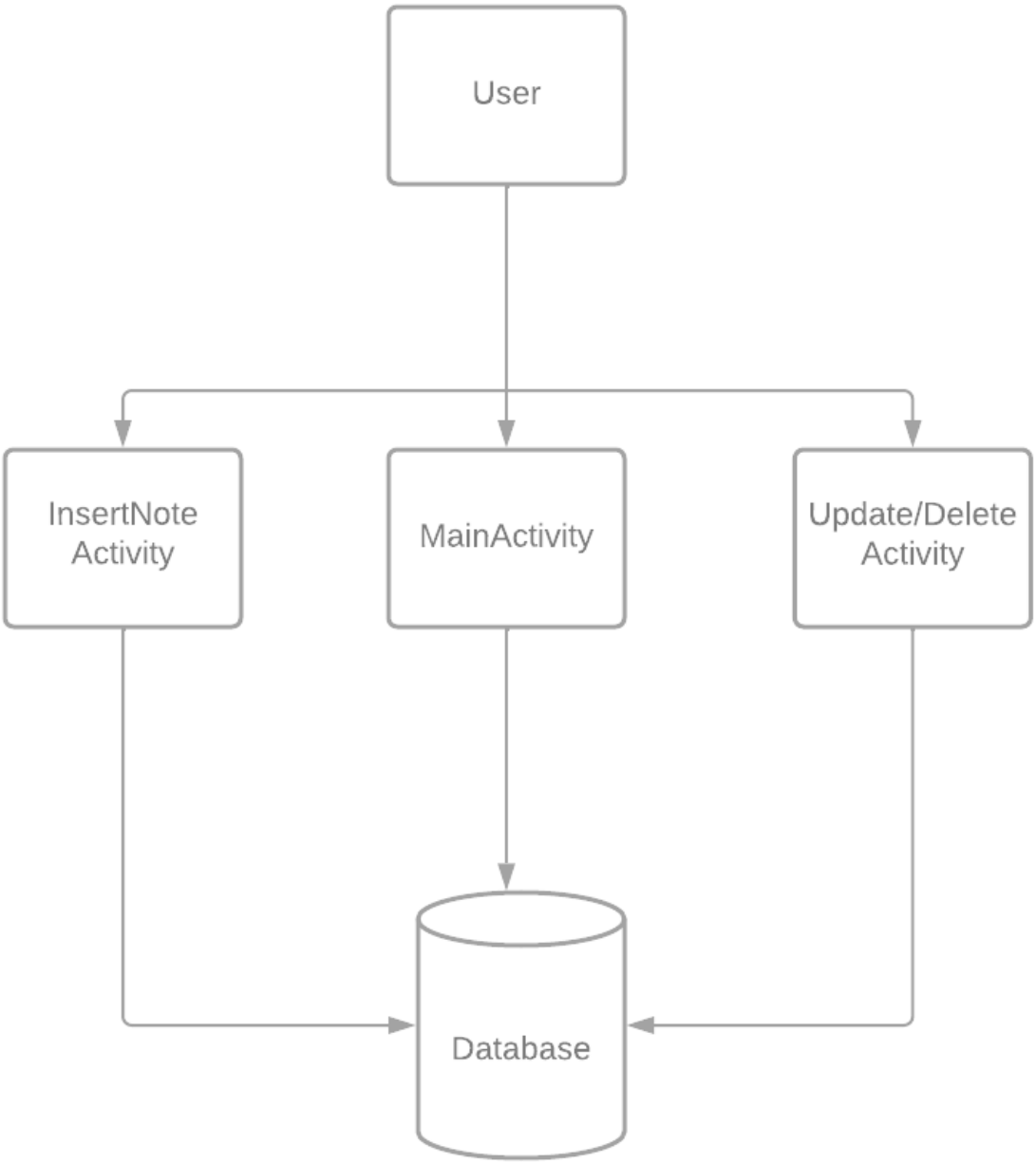
Basic structure of DFD

0 Level Diagram :-

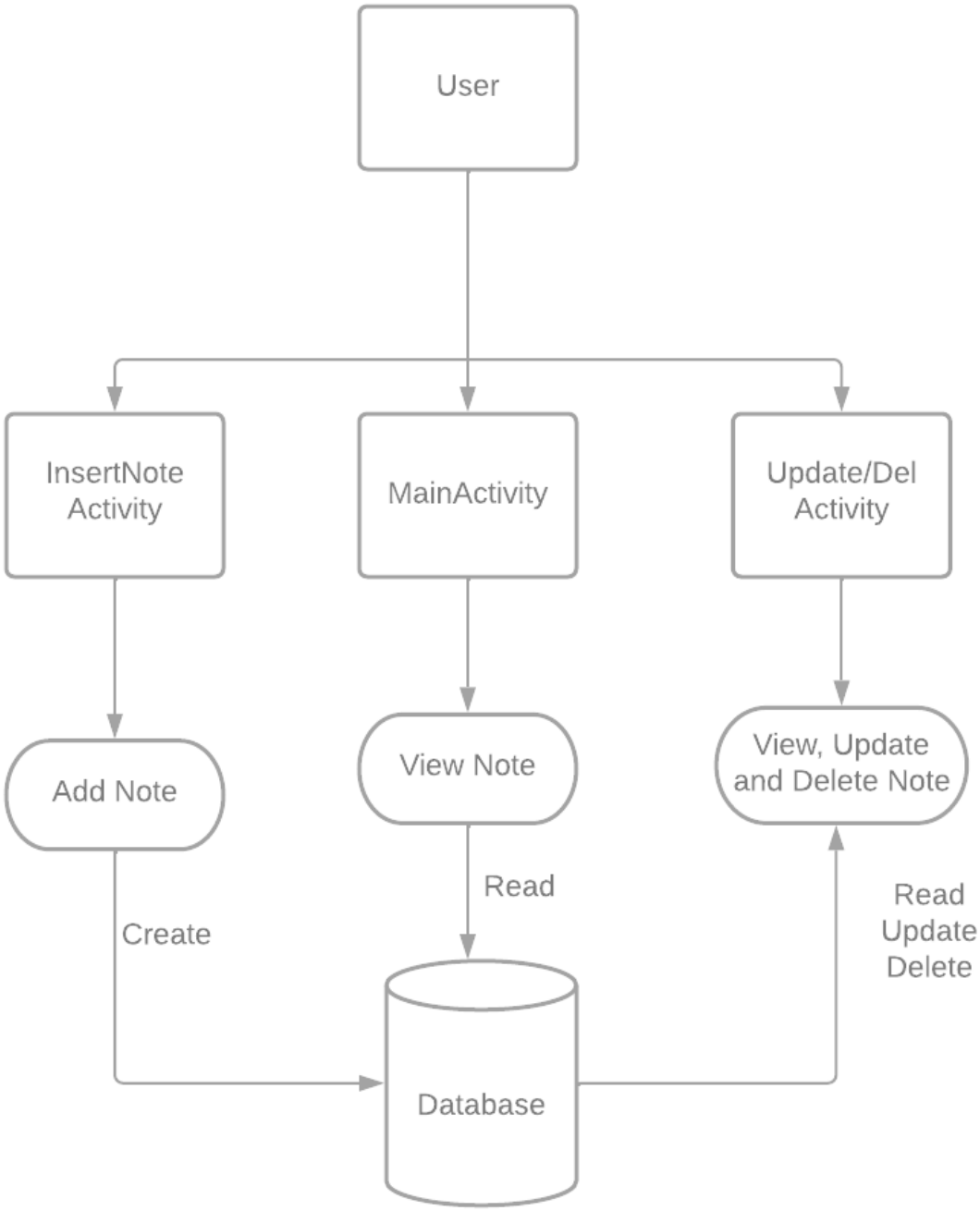
It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.



1 – Level Diagram :-



2 – Level Diagram :-



Gantt Chart

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity.

Gantt charts can be used in managing projects of all sizes and types. These may include building infrastructure like dams, bridges, and highways. They may also include software development and other technologies. Project management tools, such as Microsoft Visio, Project, SharePoint, and Excel, or specialized software, such as Gantt or Match ware, can help in designing Gantt charts.

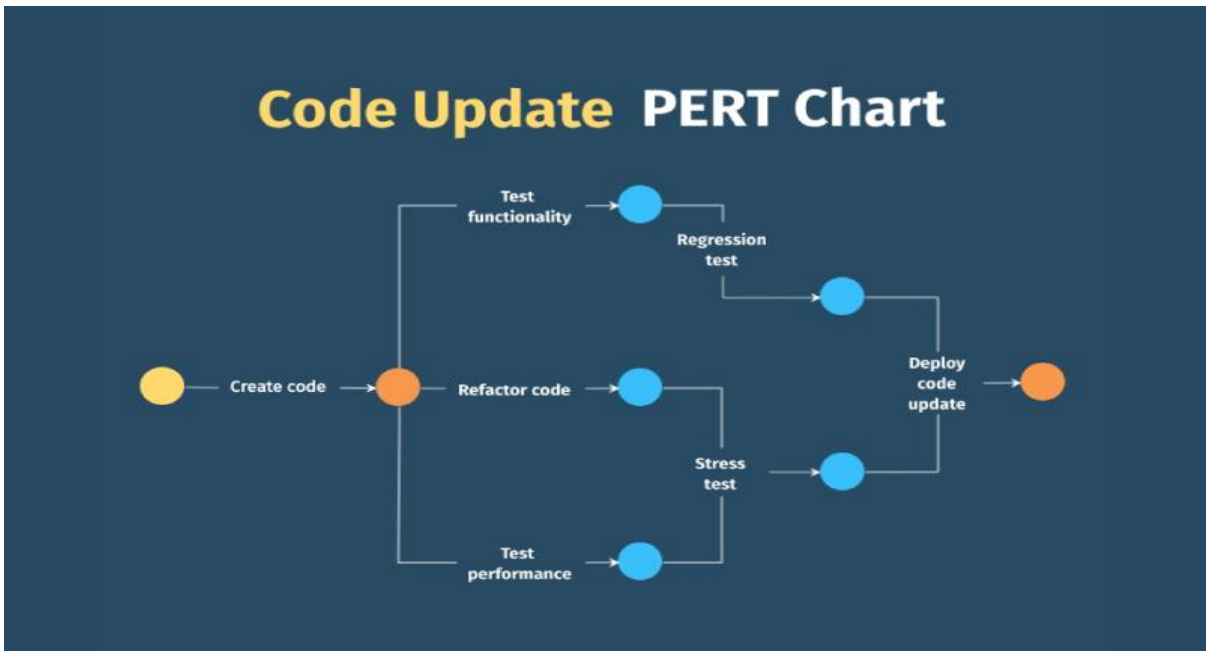
Development Phase	Days
Investigation	15/1/22
System Analyses	19/1/22
System Design	23/1/22
Code	25/1/22
Testing	5/2/22
Documentation	10/2/22

Pert Chart

A PERT chart is project management tool that provides a graphical representation of a project's timeline. The Program Evaluation Review Technique (PERT) breaks down the individual tasks of a project for analysis. PERT charts are considered preferable to Gannt Chart because they identify task dependencies, but they're often more difficult to interpret.

A PERT chart uses circles or rectangles called nodes to represent project events or milestones. These nodes are linked by vectors or lines that represent various tasks. Dependent tasks are items that must be performed in a specific manner. For example, if an arrow is drawn from Task No. 1 to Task No. 2 on a PERT chart, Task No. 1 must be completed before work on Task No. 2 begins.

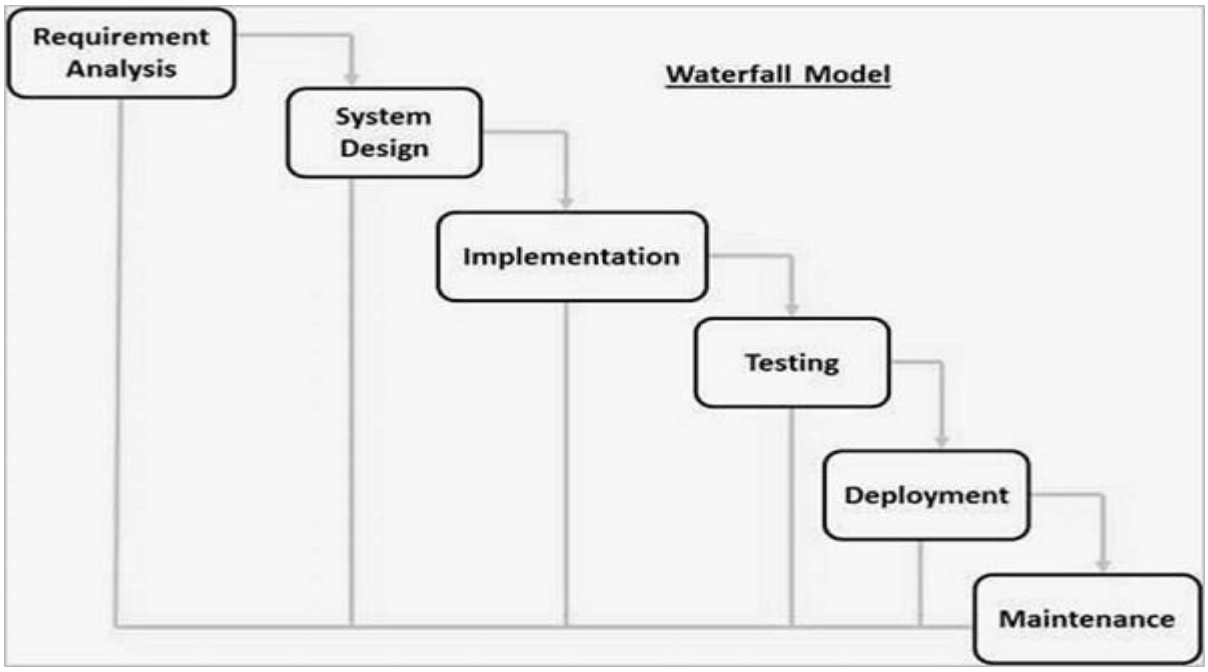
Items at the same stage of production but on different task lines within a project are referred to as parallel tasks. They're independent of each other, but they're planned to occur at the same time.



Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.



The sequential phases in Waterfall model are –

- **Requirement Gathering and analysis** :-

  All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

- **System Design** –The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Advantages :-

- Easy to understand, arrange task and use
- Works well for smaller projects where requirements are very well understood.

Disadvantages :-

- No working software is produced until late during the life cycle.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.

Project Design

User Interface :-

The user interface is perhaps the most important part of an application; it's certainly the most visible. To users, the interface is the application; they probably aren't aware of the code that is executing behind the scenes. No matter how much time and effort you put into writing and optimizing your code, the usability of your application depends on the interface.

User ultimately determines the usability of any application. Interface design is an iterative process; rarely is the first pass at designing an interface for your application going to yield a perfect interface. By getting users involved early in the design process, I have created a better, more usable interface with less effort. Perhaps the most important principle of interface design is one of simplicity.

The best place to start when designing a user interface is to look at some of the best-selling applications from Microsoft or other companies; after all, they probably didn't get to be best sellers because of poor interfaces. One of the main advantages of the Windows operating system is that it presents a common interface across all applications. A user that knows how to use one *Windows-based application* should be able to easily learn any other. Unfortunately, applications that stray too far from the established interface guidelines aren't as easily learned.

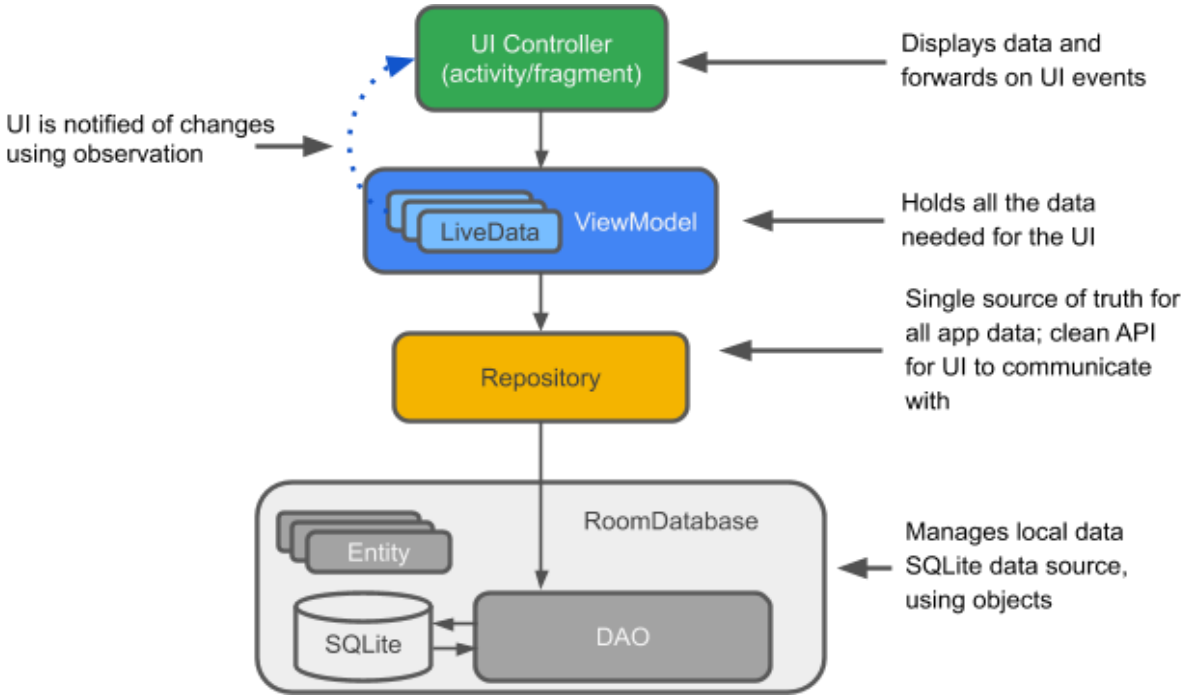
In this application, you'll find many things in common, such as Forms, MDI Interface, Menus, toolbars, ToolTips, context-sensitive menus, Colours, Images, Icons, Fonts, tabbed dialogs, and a simple screen display-based application. It's no coincidence that **Android Studio** provides the capabilities for adding all of these to your own applications.

I don't need to be an artist to create a great user interface — most of the principles of user interface design are the same as the basic design principles taught in any elementary art class.

Although Android Studio makes it easy to create a user interface by simply dragging controls onto a form, a little planning up front can make a big difference in the usability of your application. You might consider drawing your form designs on paper first, determining which controls are needed, the relative importance of the different elements, and the relationships between controls. We can also borrow from your own experience as a user of software. Think about some of the applications that you have used; what works, what doesn't, and how you would fix it.

Designing Database / Data Dictionary :-

In Android we use Room database which is wrapper over SQLite database. So we create our database with follow below Architecture Pattern :



So first of we create Entity class which contain our database table in android

Create Entity Class :-

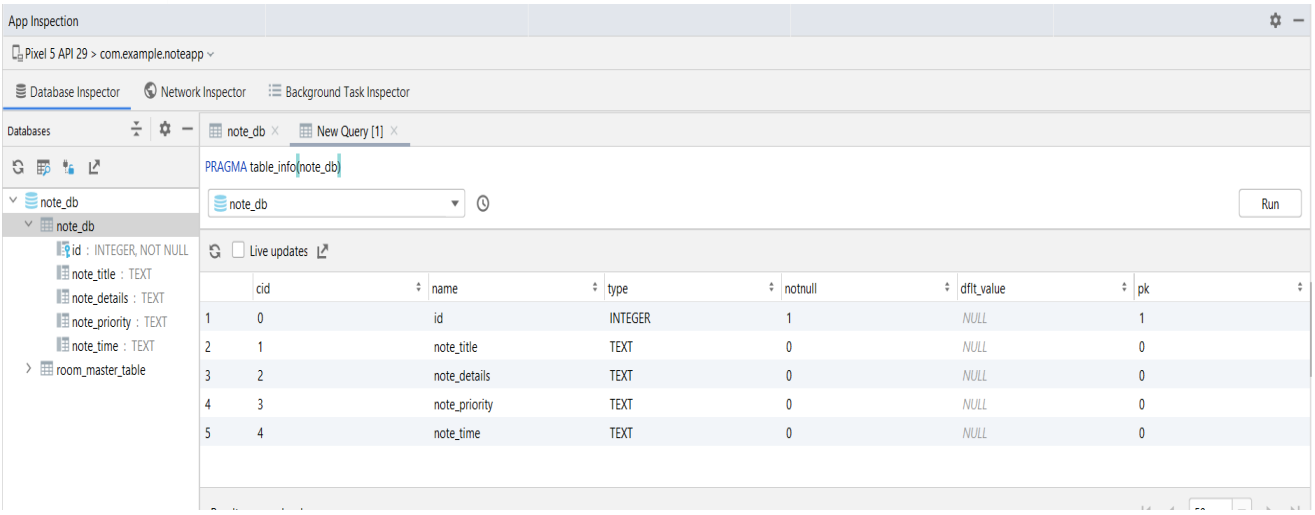
Here we you Room database which provide various annotation which we can use to create Database. In android we create database table using **Entity** class. With @Entiry annotation we can easily create database table.

Code :-

```
package com.example.noteapp.Entity;

import androidx.room.ColumnInfo;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity(tableName = "note_db")
public class Note {

    @PrimaryKey(autoGenerate = true)
    public int id;

    @ColumnInfo(name = "note_title")
    public String note_title;

    @ColumnInfo(name = "note_details")
    public String notes;

    @ColumnInfo(name = "note_priority")
    public String note_priority;

    @ColumnInfo(name = "note_time")
    public String note_time;
}
```

Note :-

We use room database which only created and show table during runtime so we can’t able to see it offline.



In next step in Room Architecture, we need to create a **@Dao** Interface which know as Database Access Object which use to perform query on table.

Create Dao(Data Access Object) :-

- Dao is interface which is annotated with @Dao annotation.

- Here we write our all query that perform on table

Code :-

```
package com.example.noteapp.Dao;

import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Insert;
import androidx.room.Query;
import androidx.room.Update;
import com.example.noteapp.Entity.Note;
import java.util.List;

@Dao
public interface NoteDao {

    @Query("SELECT * FROM note_db")
    LiveData<List<Note>> getAllNotes();

    @Query("SELECT * FROM note_db ORDER BY note_priority DESC")
    LiveData<List<Note>> getHighToLow();
```

```
    @Query("SELECT * FROM note_db ORDER BY note_priority ASC")
    LiveData<List<Note>> getLowToHigh();

    @Insert
    void insertNote(Note note);

    @Update
    void updateNote(Note note);

    @Query("DELETE FROM note_db WHERE id= :id")
    void deleteNote(int id);

}
```

Create Database :-

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Here in android studio provide **@Database** annotation where we provide table name and generate Instance which create database and available everywhere in application.

We First extend our database with RoomDatabase class and provide Entity inside our @Database annotation.

Here we create our database instance which use to perform all database operation in anywhere in application.

Code :-

```
package com.example.noteapp.Database;

import android.content.Context;
import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
import com.example.noteapp.Dao.NoteDao;
import com.example.noteapp.Entity.Note;


@Database(entities = {Note.class}, version = 1, exportSchema = false)
public abstract class NoteDB extends RoomDatabase {

    public static NoteDB INSTANCE;
```

```
    public static NoteDB getDatabaseInstance(Context context) {
        if (INSTANCE == null) {
            INSTANCE =
Room.databaseBuilder(context.getApplicationContext(),
                    NoteDB.class,
                    "note_db")
                    .allowMainThreadQueries()
                    .build();
        }
        return INSTANCE;
    }
    public abstract NoteDao noteDao();
}
```

Now based on Room Architecture we need to create a Repository which handle data in application. Repository is a kind of class which manage all data in application work as an api where all data passed between database and User.

Create Repository :-

Code :-

```
package com.example.noteapp.Repository;

import android.app.Application;
import androidx.lifecycle.LiveData;
import com.example.noteapp.Dao.NoteDao;
import com.example.noteapp.Database.NoteDB;
import com.example.noteapp.Entity.Note;
import java.util.List;

public class NoteRepo {
    public NoteDao noteDao;
    public LiveData<List<Note>> getAllNotes;
    public LiveData<List<Note>> getHighToLow;
    public LiveData<List<Note>> getLowToHigh;

    public NoteRepo(Application application) {
        NoteDB noteDB = NoteDB.getDatabaseInstance(application);
        noteDao = noteDB.noteDao();
        getAllNotes = noteDao.getAllNotes();
        getHighToLow = noteDao.getHighToLow();
        getLowToHigh = noteDao.getLowToHigh();
    }

    public void InsertNote(Note note) {
        noteDao.insertNote(note);
    }
    public void UpdateNote(Note note) {
        noteDao.updateNote(note);
    }
```

```
    public void DeleteNote(int id) {
        noteDao.deleteNote(id);
    }
}
```

Our Repository build successfully now we need to create ViewModel. ViewModel is class which hold data of the application and provide direct to the Activity(User Interface). Here we get data from user interface and pass to repository and updated data to UI from repository.

Create ViewModel :-

Code :-

```
package com.example.noteapp.ViewModel;

import android.app.Application;
import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;
import com.example.noteapp.Entity.Note;
import com.example.noteapp.Repository.NoteRepo;

import java.util.List;

public class NoteViewModel extends AndroidViewModel {

    public NoteRepo repo;
    public LiveData<List<Note>> getAllNote;
    public LiveData<List<Note>> getHighToLow;
    public LiveData<List<Note>> getLowToHigh;

    public NoteViewModel(@NonNull Application application) {
        super(application);
        repo = new NoteRepo(application);
        getAllNote = repo.getAllNotes;
        getHighToLow = repo.getHighToLow;
        getLowToHigh = repo.getLowToHigh;
    }

    public void InsertNote(Note note) {
        repo.InsertNote(note);}

    public void UpdateNote(Note note) {
        repo.UpdateNote(note);    }
    public void DeleteNote(int id) {
        repo.DeleteNote(id);}}
```

- Now we have done our all-Database Setup step based in architecture.

Backup Database For Application :-

To ensure that your data is safeguarded, I define a solid backup strategy. You can backup all database as well as One can also backup all data including entire application according to needs on any of the given media: Floppy, Hard Disk, CD, DVD or even on a Zip Disk or a thumb storage drive or on a local network, etc. This software also takes automatic backups.

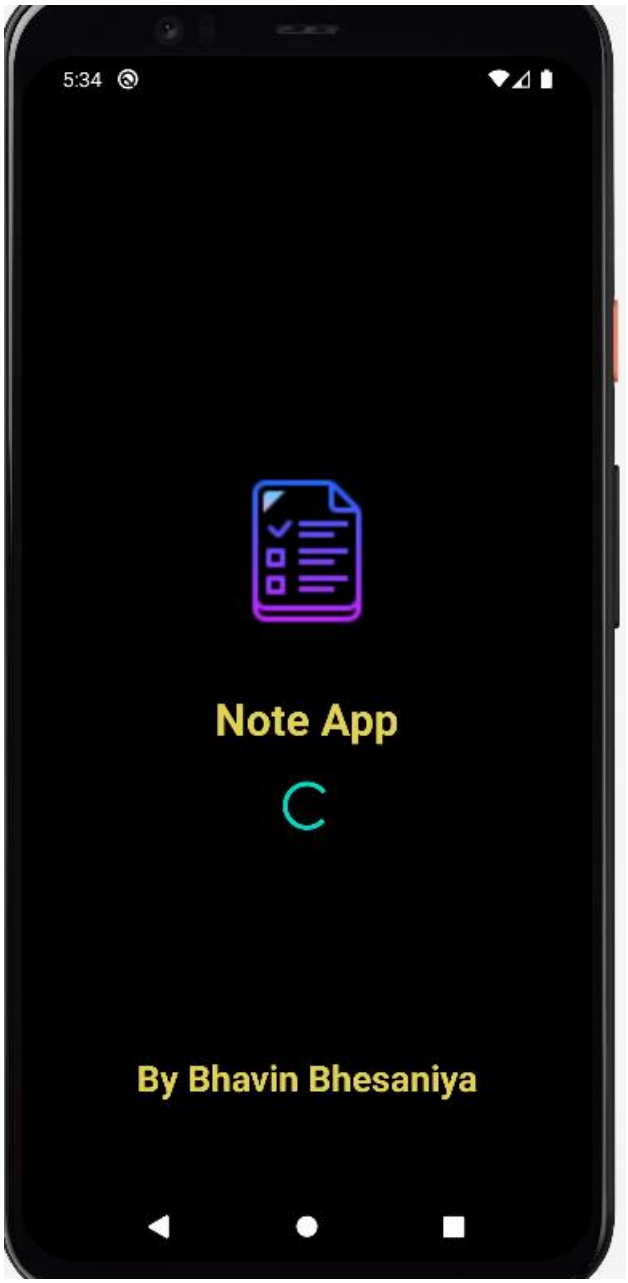
Today's hardware and software components are many times more reliable than those of the not-so-distant past; however, they will never be perfect, and there is always the chance of having some type of failure. It is therefore prudent to always have some type of "safety net" in place at all times, so that if a situation arises that results in the loss or damage of online data, production can be restored quickly with minimal or no loss of data. When people think of situations that could result in the loss of data, they typically think of disk crashes.

Some of these problems include :-

- Disk subsystem failure
- Systems software failure
- Accidental or malicious use of deletion statements
- Accidental or malicious use of updating statements
- Destructive viruses
- Natural disasters (fire, flood, earthquake, and so on)
- Theft

Screen And Source Code Of The Project

Screen :- 1 SplashScreen.java



Code :-

```
package com.example.noteapp.Activity;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import androidx.appcompat.app.AppCompatActivity;
import com.example.noteapp.MainActivity;
import com.example.noteapp.R;

public class SplashScreen extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen);

        getSupportActionBar().hide();

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                startActivity(new Intent(SplashScreen.this,
MainActivity.class));
                finish();
            }
        }, 10000);
    }
}
```

Here we need to Create adapter class which bind our data in recycler view with Activity. This adapter help class insert, update, read and delete data. Adapter class mainly use to bind data with recycler view. Whenever data is updated in database direct data change in recycler view which give us user great experience.

Class :- NoteAdatper.java

Code :

```
package com.example.noteapp.Adapter;

import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.example.noteapp.Entity.Note;
import com.example.noteapp.MainActivity;
import com.example.noteapp.R;
```

```
import com.example.noteapp.UpdateNoteActivity;
import java.util.ArrayList;
import java.util.List;

public class NoteAdatper extends
RecyclerView.Adapter<NoteAdatper.NoteViewHolder> {

    MainActivity mainActivity;
    List<Note> notes;
    List<Note> allNote;

    public NoteAdatper(MainActivity mainActivity, List<Note> notes) {
        this.mainActivity = mainActivity;
        this.notes = notes;
        allNote = new ArrayList<>(notes);
    }

    public void searchNote(List<Note> filterName) {
        this.notes = filterName;
        notifyDataSetChanged();
    }

    @NonNull
    @Override
    public NoteViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        return new
NoteViewHolder(LayoutInflater.from(mainActivity).inflate(R.layout.item_note
, parent, false));
    }

    @Override
    public void onBindViewHolder(@NonNull NoteViewHolder holder, int
position) {
        Note note = notes.get(position);
        holder.title.setText(note.note_title);
        holder.noteDetail.setText(note.notes);
        holder.noteTime.setText(note.note_time);

        if (note.note_priority.equals("3")) {
holder.notePriority.setBackgroundResource(R.drawable.red_shape);
        } else if (note.note_priority.equals("2")) {
holder.notePriority.setBackgroundResource(R.drawable.yellow_shape);
        } else {
holder.notePriority.setBackgroundResource(R.drawable.green_shape);
        }

        holder.itemView.setOnClickListener(v -> {
            Intent intent = new Intent(mainActivity,
        UpdateNoteActivity.class);
            intent.putExtra("id", note.id);
            intent.putExtra("title", note.note_title);
            intent.putExtra("note", note.notes);
            intent.putExtra("priority", note.note_priority);
            intent.putExtra("time", note.note_time);
            mainActivity.startActivity(intent);
        });
    }

    @Override
```

```
    public int getItemCount() {
        return notes.size();
    }

    public static class NoteViewHolder extends RecyclerView.ViewHolder {
        TextView title, noteDetail, noteTime;
        View notePriority;

        public NoteViewHolder(@NonNull View itemView) {
            super(itemView);
            title = itemView.findViewById(R.id.showNoteTitle);
            noteDetail = itemView.findViewById(R.id.showDetail);
            notePriority = itemView.findViewById(R.id.showNotePriority);
            noteTime = itemView.findViewById(R.id.showNoteTime);
        }
    }
}
```

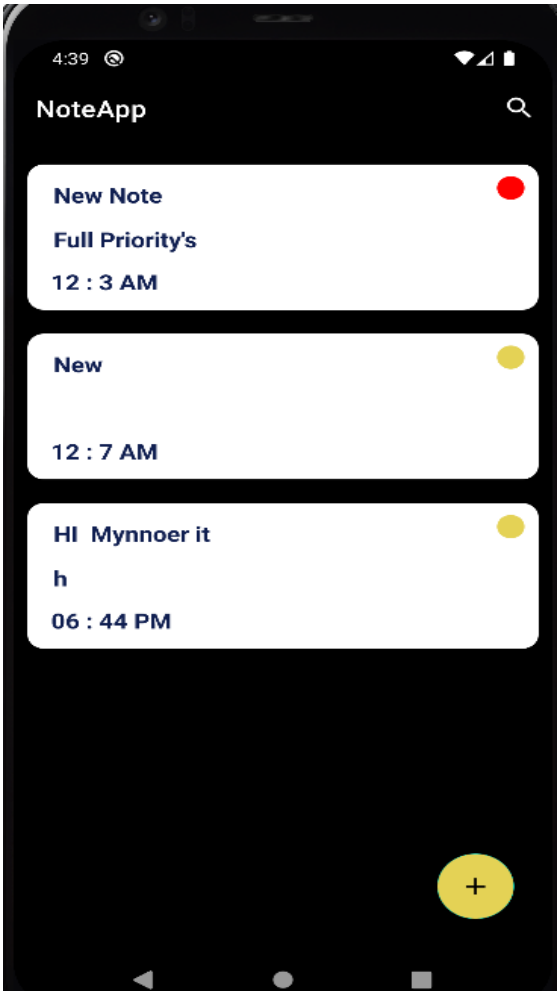
Screen :- 2 MainActivity.java

```
package com.example.noteapp;
import android.annotation.SuppressLint;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.SearchView;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProviders;
import androidx.recyclerview.widget.RecyclerView;
import androidx.recyclerview.widget.StaggeredGridLayoutManager;
import com.example.noteapp.Adapter.NoteAdatper;
import com.example.noteapp.Entity.Note;
import com.example.noteapp.ViewModel.NoteViewModel;
import com.example.noteapp.databinding.ActivityMainBinding;
import java.util.ArrayList;
import java.util.List;


public class MainActivity extends AppCompatActivity implements
View.OnClickListener {

    ActivityMainBinding binding;
    TextView noFilter, highToLowFilter, lowToHighFilter;
    ImageView showFilter;
    RecyclerView noteRecyclerView;
    NoteAdatper noteAdatper;
    NoteViewModel noteViewModel;
    boolean filterVisible = true, fabOnOff = true;
    List<Note> filterNoteList;
```

```
    @SuppressLint("WrongViewCast")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        showFilter = findViewById(R.id.showFilter);
        noFilter = findViewById(R.id.NoFilterTv);
        highToLowFilter = findViewById(R.id.HighToLowTv);
        lowToHighFilter = findViewById(R.id.LowToHighTv);


        binding.addBtn.setOnClickListener(this);
        showFilter.setOnClickListener(this);
        noFilter.setOnClickListener(this);
        highToLowFilter.setOnClickListener(this);
        lowToHighFilter.setOnClickListener(this);
        binding.addNewNote.setOnClickListener(this);

        binding.addNewNote.setVisibility(View.GONE);
        binding.showFilter.setVisibility(View.GONE);
        noFilter.setVisibility(View.GONE);
        highToLowFilter.setVisibility(View.GONE);
        lowToHighFilter.setVisibility(View.GONE);
        createNotificationChannel();

        noteViewModel =newModelProviders.of(this).get(NoteViewModel.class);
        noteRecyclerView = findViewById(R.id.noteRecyclerView);
        loadData(0);
    }


    @Override
    public void onClick(View v) {
        int id = v.getId();
        if (id == R.id.addNewNote) {
            startActivity(new Intent(MainActivity.this,
InsertNoteActivity.class));
        } else if (id == R.id.addBtn) {
            if (fabOnOff) {
                binding.addNewNote.setVisibility(View.VISIBLE);
                binding.showFilter.setVisibility(View.VISIBLE);
                fabOnOff = false;
            } else {
                binding.addNewNote.setVisibility(View.GONE);
                binding.showFilter.setVisibility(View.GONE);
                fabOnOff = true;
            }
        } else if (id == R.id.showFilter) {
            if (filterVisible) {
                noFilter.setVisibility(View.VISIBLE);
                highToLowFilter.setVisibility(View.VISIBLE);
                lowToHighFilter.setVisibility(View.VISIBLE);
                filterVisible = false;
            } else {
                noFilter.setVisibility(View.GONE);
                highToLowFilter.setVisibility(View.GONE);
                lowToHighFilter.setVisibility(View.GONE);
```

```
                filterVisible = true;
            }
        } else if (id == R.id.NoFilterTv) {
            loadData(0);
        noFilter.setBackgroundResource(R.drawable.yellow_border);
        highToLowFilter.setBackgroundResource(R.drawable.filter_list_btn);
        lowToHighFilter.setBackgroundResource(R.drawable.filter_list_btn);
        } else if (id == R.id.HighToLowTv) {
            loadData(1);
        highToLowFilter.setBackgroundResource(R.drawable.yellow_border);
        noFilter.setBackgroundResource(R.drawable.filter_list_btn);
        lowToHighFilter.setBackgroundResource(R.drawable.filter_list_btn) }
        else {
        loadData(2);
        lowToHighFilter.setBackgroundResource(R.drawable.yellow_border);
         highToLowFilter.setBackgroundResource(R.drawable.filter_list_btn);
        noFilter.setBackgroundResource(R.drawable.filter_list_btn);
        }
    }

    private void loadData(int i) {
        if (i == 0) {
            noteViewModel.getAllNote.observe(this, notes -> {
                setAdatper(notes);
                filterNoteList = notes;});
        } else if (i == 1) {
            noteViewModel.getHighToLow.observe(this, notes -> {
                setAdatper(notes);
                filterNoteList = notes;});
        } else if (i == 2) {
            noteViewModel.getLowToHigh.observe(this, notes -> {
                setAdatper(notes);
                filterNoteList = notes;});
        }
    }

    public void setAdatper(List<Note> notes) {
        noteRecyclerView.setLayoutManager(new StaggeredGridLayoutManager(1,
StaggeredGridLayoutManager.VERTICAL));
        noteAdatper = new NoteAdatper(MainActivity.this, notes);
        noteRecyclerView.setAdapter(noteAdatper);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.search_note, menu);
        MenuItem menuItem = menu.findItem(R.id.app_bar_search);
        SearchView searchView = (SearchView) menuItem.getActionView();
        searchView.setQueryHint("Search Note...");
        searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                return false;
            }@Override
            public boolean onQueryTextChange(String newText) {
                NotesFilter(newText);
                return false;
            }
        });
```

```
        return super.onCreateOptionsMenu(menu);}

    private void NotesFilter(String newText) {
        ArrayList<Note> filterName = new ArrayList<>();
        for (Note note : this.filterNoteList) {
            if (note.note_title.contains(newText)) {
                filterName.add(note);}}
        this.noteAdatper.searchNote(filterName); }
    private void createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            CharSequence name = "bhavin channel";
            String description = "Checking alarm";
            int important = NotificationManager.IMPORTANCE_HIGH;
            NotificationChannel channel = new
NotificationChannel("Android", name, important);
            channel.setDescription(description);
            NotificationManager notificationManager =
getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);}}}
```



Screen 3 :- InsertNoteActivity.java

Code :-

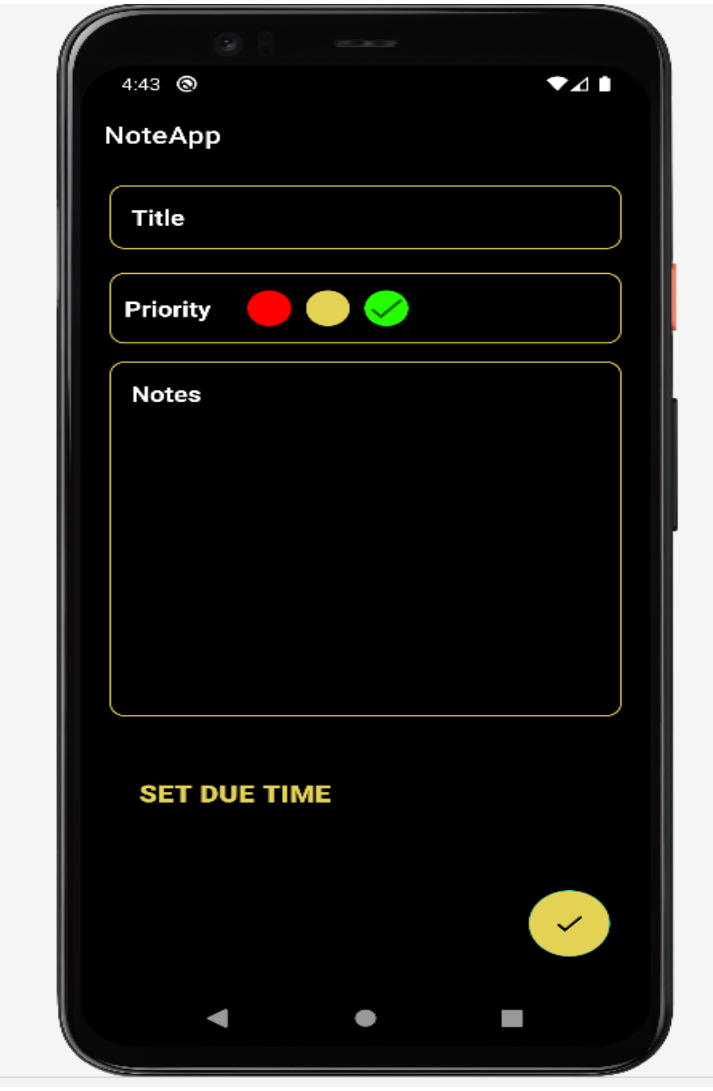
```
package com.example.noteapp;

import android.app.AlarmManager;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;
import androidx.lifecycle.ViewModelProviders;

import com.example.noteapp.Entity.Note;
import com.example.noteapp.ViewModel.NoteViewModel;
import com.example.noteapp.databinding.ActivityInsertNoteBinding;
import com.google.android.material.timepicker.MaterialTimePicker;
import com.google.android.material.timepicker.TimeFormat;

import java.util.Calendar;

public class InsertNoteActivity extends AppCompatActivity implements
View.OnClickListener {

    ActivityInsertNoteBinding binding;
    String note_title, note_detail, priority = "1", note_time;

    NoteViewModel noteViewModel;

    NotificationManagerCompat notificationCompat;
    Notification notification;

    //Time Picker
    int cyear, cmonth, cday;
    MaterialTimePicker picker;
    Calendar calendar = Calendar.getInstance();


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityInsertNoteBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        noteViewModel =
ViewModelProviders.of(this).get(NoteViewModel.class);

        binding.priorityHigh.setOnClickListener(this);
        binding.priorityMedium.setOnClickListener(this);
        binding.priorityLow.setOnClickListener(this);
        binding.setTimeBtn.setOnClickListener(this);
        binding.saveNoteBtn.setOnClickListener(this);
```

```
    }

    @Override
    public void onClick(View v) {
        int id = v.getId();
        if (id == R.id.saveNoteBtn) {
            note_title = binding.noteTitle.getText().toString();
            note_detail = binding.notesDetail.getText().toString();
            note_time = binding.timeDispTime.getText().toString();
            if (!note_title.isEmpty() || !note_detail.isEmpty()) {
                if (!note_time.isEmpty()) {
                    CreateNotes(note_title, note_detail);
                } else {
                    Toast.makeText(this, "Please Select Time",
Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(this, "Please Enter Details",
Toast.LENGTH_SHORT).show();
            }
        } else if (id == R.id.setTimeBtn) {
            showTimePicker();
        } else if (id == R.id.priorityHigh) {
            priority = "3";

binding.priorityHigh.setImageResource(R.drawable.ic_baseline_done_24);
            binding.priorityMedium.setImageResource(0);
            binding.priorityLow.setImageResource(0);
        } else if (id == R.id.priorityMedium) {
            priority = "2";

binding.priorityMedium.setImageResource(R.drawable.ic_baseline_done_24);
            binding.priorityLow.setImageResource(0);
            binding.priorityHigh.setImageResource(0);
        } else if (id == R.id.priorityLow) {
            priority = "1";

binding.priorityLow.setImageResource(R.drawable.ic_baseline_done_24);
            binding.priorityMedium.setImageResource(0);
            binding.priorityHigh.setImageResource(0);
        }
    }

    private void CreateNotes(String note_title, String note_detail) {
        Note note = new Note();
        note.note_title = note_title;
        note.notes = note_detail;
        note.note_priority = priority;
        note.note_time = binding.timeDispTime.getText().toString();
        noteViewModel.InsertNote(note);
        callNotification();
        scheduleNotification(getNotification(note_title),
calendar.getTimeInMillis());
        Toast.makeText(getApplicationContext(), "Note Added SuccessFully",
Toast.LENGTH_SHORT).show();
        finish();
    }

    public void callNotification() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
```

```
            NotificationChannel channel = new NotificationChannel("Mych",
"My Channel", NotificationManager.IMPORTANCE_DEFAULT);
            NotificationManager manager =
getSystemService(NotificationManager.class);
            manager.createNotificationChannel(channel);
        }
        NotificationCompat.Builder builder = new
NotificationCompat.Builder(this, "Mych")
                .setSmallIcon(R.drawable.ic_launcher)
                .setContentTitle("Note App Notification")
                .setContentText("Note Created Successfully")
                .setTimeoutAfter(7000);
        notification = builder.build();
        notificationCompat = NotificationManagerCompat.from(this);
        notificationCompat.notify(1, notification);
    }

    private void showTimePicker() {
        picker = new MaterialTimePicker.Builder()
                .setTimeFormat(TimeFormat.CLOCK_12H)
                .setHour(12)
                .setMinute(0)
                .setTitleText("Select Time")
                .build();

        picker.show(getSupportFragmentManager(), "Android");
        picker.addOnPositiveButtonClickListener(v -> {
            if (picker.getHour() > 12) {
                binding.timeDispTime.setText(String.format("%02d",
(picker.getHour() - 12)) + " : " + String.format("%02d",
picker.getMinute()) + " PM");
            } else {
                binding.timeDispTime.setText(picker.getHour() + " : " +
picker.getMinute() + " AM");
            }
            calendar.set(Calendar.HOUR_OF_DAY, picker.getHour());
            calendar.set(Calendar.MINUTE, picker.getMinute());
            calendar.set(Calendar.SECOND, 0);
            calendar.set(Calendar.MILLISECOND, 0);
        });
    }

    private Notification getNotification(String note_title) {
        Intent i = new Intent(this, MainActivity.class);
        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 1, i,
0);

        NotificationCompat.Builder builder = new
NotificationCompat.Builder(InsertNoteActivity.this, "Android")
                .setSmallIcon(R.drawable.ic_launcher)
                .setContentTitle("Your Note Reminder : ")
                .setContentText(note_title)
                .setAutoCancel(true)
                .setDefaults(Notification.DEFAULT_ALL)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setContentIntent(pendingIntent);
        return builder.build(); }
    private void scheduleNotification(Notification notification, long
```

```
delay) {
        Intent notificationIntent = new Intent(this, AlarmReceiver.class);
        notificationIntent.putExtra("NOTIFICATION_ID", 1);
        notificationIntent.putExtra("NOTIFICATION", notification);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 1,
notificationIntent, PendingIntent.FLAG_UPDATE_CURRENT);
        AlarmManager alarmManager = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        assert alarmManager != null;
        alarmManager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,
delay, pendingIntent); }}
```

Screen :



Update Note Activity :-

Code :-

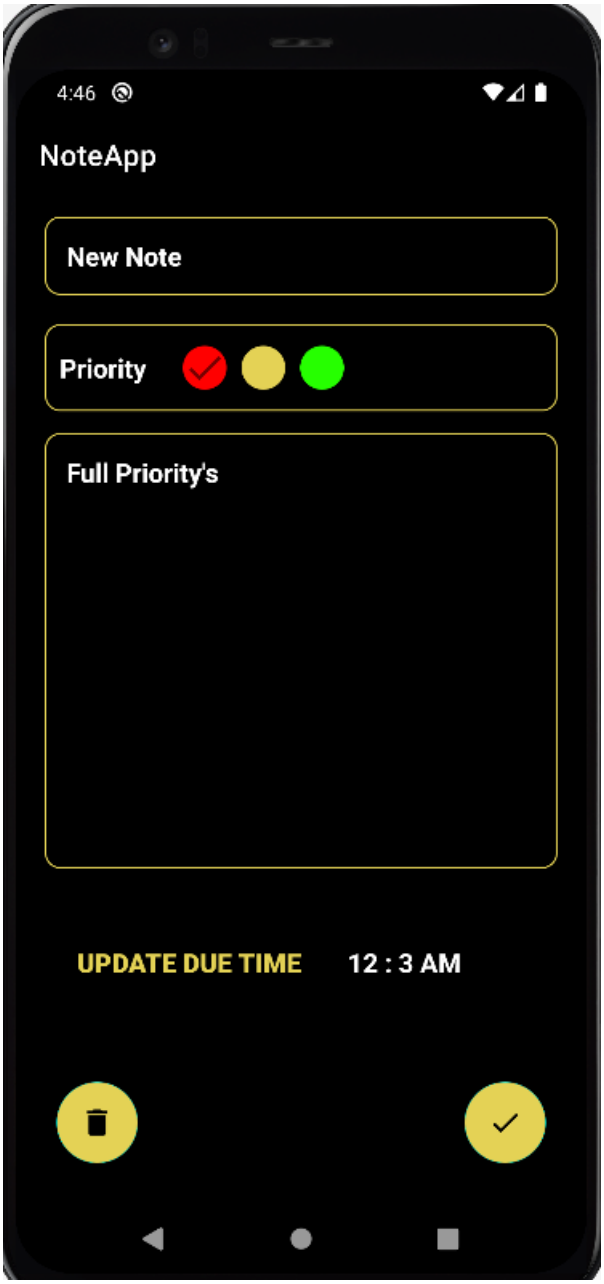
```
package com.example.noteapp;

import android.app.AlarmManager;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;
import androidx.lifecycle.ViewModelProviders;
import com.example.noteapp.Entity.Note;
import com.example.noteapp.ViewModel.NoteViewModel;
import com.example.noteapp.databinding.ActivityUpdateNoteBinding;
import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.google.android.material.timepicker.MaterialTimePicker;
import com.google.android.material.timepicker.TimeFormat;

import java.util.Calendar;
import java.util.Date;

public class UpdateNoteActivity extends AppCompatActivity implements
View.OnClickListener {

    ActivityUpdateNoteBinding binding;
    String up_note_title, up_note_detail, priority = "1", up_note_date,
up_note_time;
    int up_note_id;
    NoteViewModel noteViewModel;

    //Time Picker
    int cyear, cmonth, cday;
    MaterialTimePicker picker;
    Calendar calendar = Calendar.getInstance();

    //Notification
    NotificationManagerCompat notificationCompat;
    Notification notification;

    Date datetm;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityUpdateNoteBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        noteViewModel =
ViewModelProviders.of(this).get(NoteViewModel.class);
```

```
        up_note_id = getIntent().getIntExtra("id", 0);
        up_note_title = getIntent().getStringExtra("title");
        up_note_detail = getIntent().getStringExtra("note");
        priority = getIntent().getStringExtra("priority");
        up_note_date = getIntent().getStringExtra("date");
        up_note_time = getIntent().getStringExtra("time");

        binding.upNoteTitle.setText(up_note_title);
        binding.upNotesDetail.setText(up_note_detail);
        if (priority.equals("3")) {

binding.upPriorityHigh.setImageResource(R.drawable.ic_baseline_done_24);
        } else if (priority.equals("2")) {
 binding.upPriorityMedium.setImageResource(R.drawable.ic_baseline_done_24);
        } else {
binding.upPriorityLow.setImageResource(R.drawable.ic_baseline_done_24);
        }
        binding.timeDispTime.setText(up_note_time);
        binding.upPriorityHigh.setOnClickListener(this);
        binding.upPriorityMedium.setOnClickListener(this);
        binding.upPriorityLow.setOnClickListener(this);
        binding.setTimeBtn.setOnClickListener(this);
        binding.updateNoteBtn.setOnClickListener(this);
        binding.deleteNoteBtn.setOnClickListener(this);
    }
    private void UpdateNotes(String title, String noteDetail) {
        Note note = new Note();
        note.id = up_note_id;
        note.note_title = title;
        note.notes = noteDetail;
        note.note_priority = priority;
        note.note_time = binding.timeDispTime.getText().toString();
        scheduleNotification(getNotification(title),
calendar.getTimeInMillis());
        noteViewModel.UpdateNote(note);
        callNotification();
        Toast.makeText(getApplicationContext(), "Note Updated
SuccessFully", Toast.LENGTH_SHORT).show();
        finish();
    }

    @Override
    public void onClick(View v) {
        int id = v.getId();
        if (id == R.id.updateNoteBtn) {
            String title = binding.upNoteTitle.getText().toString();
            String noteDetail = binding.upNotesDetail.getText().toString();
            String noteTime = binding.timeDispTime.getText().toString();
            if (!title.isEmpty() || !noteDetail.isEmpty()) {
                if (!noteTime.isEmpty()) {
                    UpdateNotes(title, noteDetail);
                } else {
                    Toast.makeText(this, "Please Select Time",
Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(this, "Please Enter Details",
Toast.LENGTH_SHORT).show();
            }
        } else if (id == R.id.upPriorityHigh) {
```

```
            priority = "3";

binding.upPriorityHigh.setImageResource(R.drawable.ic_baseline_done_24);
            binding.upPriorityMedium.setImageResource(0);
            binding.upPriorityLow.setImageResource(0);
        } else if (id == R.id.upPriorityMedium) {
            priority = "2";

binding.upPriorityMedium.setImageResource(R.drawable.ic_baseline_done_24);
            binding.upPriorityLow.setImageResource(0);
            binding.upPriorityHigh.setImageResource(0);
        } else if (id == R.id.upPriorityLow) {
            priority = "1";

binding.upPriorityLow.setImageResource(R.drawable.ic_baseline_done_24);
            binding.upPriorityMedium.setImageResource(0);
            binding.upPriorityHigh.setImageResource(0);
        } else if (id == R.id.setTimeBtn) {
            showTimePicker();
        } else if (id == R.id.deleteNoteBtn) {
            BottomSheetDialog sheetDialog = new
BottomSheetDialog(UpdateNoteActivity.this);
            View view =
LayoutInflater.from(UpdateNoteActivity.this).inflate(R.layout.delete_bottom
_sheet, (LinearLayout) findViewById(R.id.bottomSheet));
            sheetDialog.setContentView(view);
            sheetDialog.show();

            Button yesDelete, noDelete;
            yesDelete = view.findViewById(R.id.yesDeleteBtn);
            noDelete = view.findViewById(R.id.noDeleteBtn);

            yesDelete.setOnClickListener(v1 -> {
                noteViewModel.DeleteNote(up_note_id);
                finish();
                Toast.makeText(getApplicationContext(), "Not Successfully
deleted", Toast.LENGTH_SHORT).show();
            });
            noDelete.setOnClickListener(v1 -> {
                sheetDialog.dismiss();             });     }     }

    private void showTimePicker() {
        picker = new MaterialTimePicker.Builder()
                .setTimeFormat(TimeFormat.CLOCK_12H)
                .setHour(12)
                .setMinute(0)
                .setTitleText("Select Time")
                .build();

        picker.show(getSupportFragmentManager(), "Android");
        picker.addOnPositiveButtonClickListener(v -> {
            if (picker.getHour() > 12) {
                binding.timeDispTime.setText(String.format("%02d",
        (picker.getHour() - 12)) + " : " + String.format("%02d",
        picker.getMinute()) + " PM");
            } else {
                binding.timeDispTime.setText(picker.getHour() + " : " +
        picker.getMinute() + " AM");
            }
            calendar.set(Calendar.HOUR_OF_DAY, picker.getHour());
```

```
            calendar.set(Calendar.MINUTE, picker.getMinute());
            calendar.set(Calendar.SECOND, 0);
            calendar.set(Calendar.MILLISECOND, 0); });}

    public void callNotification() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new
        NotificationChannel("MyNoteUp", "My Channel",
        NotificationManager.IMPORTANCE_DEFAULT);
         NotificationManager manager =
        getSystemService(NotificationManager.class);
            manager.createNotificationChannel(channel);
         } NotificationCompat.Builder builder = new
        NotificationCompat.Builder(this, "MyNoteUp")
         .setSmallIcon(R.drawable.ic_launcher)
        .setContentTitle("Note App Notification")
                .setContentText("Note Updated Successfully")
                .setTimeoutAfter(4000);
         notification = builder.build();
         notificationCompat = NotificationManagerCompat.from(this);
         notificationCompat.notify(1, notification);
    }
    private Notification getNotification(String note_title) {
        Intent i = new Intent(this, MainActivity.class);
        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK
        Intent.FLAG_ACTIVITY_CLEAR_TASK);
         PendingIntent pendingIntent = PendingIntent.getActivity(this, 1, i,
0);
NotificationCompat.Builder builder = new
NotificationCompat.Builder(UpdateNoteActivity.this, "Android")
                .setSmallIcon(R.drawable.ic_launcher)
                .setContentTitle("Reminder For your Note : ")
                .setContentText(note_title)
                .setAutoCancel(true)
                .setDefaults(Notification.DEFAULT_ALL)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setContentIntent(pendingIntent);
         return builder.build();
    }

    private void scheduleNotification(Notification notification, long
delay) {
         Intent notificationIntent = new Intent(this, AlarmReceiver.class);
         notificationIntent.putExtra("NOTIFICATION_ID", 1);
         notificationIntent.putExtra("NOTIFICATION", notification);
         PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 1,
notificationIntent, PendingIntent.FLAG_UPDATE_CURRENT);
         AlarmManager alarmManager = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
         assert alarmManager != null;
         alarmManager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,
delay, pendingIntent);
    }
}
```

Screen :-


4:46
NoteApp
New Note
Priority
Full Priority's
UPDATE DUE TIME
12 : 3 AM

Testing

Testing process contains many types of phases testing can also be divided into various types. Testing is a process to find the error and to combine it with the development of the software.

Error may be occurs when the objects are improperly specified and declared, failed, fault, test case and suite are connected with the phases process.

BLACK-BOX TESTING :-

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioural Testing.

TYPES OF BLACK-BOX TESTING :-

- **Functional testing** – This black box testing type is related to the functional requirements of a system; it is done by software testers.
- **Non-functional testing** – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** – Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

WHITE-BOX TESTING :-

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

TYPES OF WHITE-BOX TESTING :-

- **Unit Testing :-** It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.
- **Testing for Memory Leaks** :- Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.
- **White Box Penetration testing :-** In this testing, the tester/developer has full information of the application's source code, detailed network information, IP addresses involved and all server information the application runs on. The aim is to attack the code from several angles to expose security threats

- **White Box Mutation Testing** :- Mutation testing is often used to discover the best coding techniques to use for expanding a software solution.

Test case

Test Suit Id – 1 Test Id – 1 Executed By : Bhavin Bhesaniya			Description :- Check Record Insert, update, delete and Notification and search Filter.		
No.	**Task**	**Expected Output**	**Actual Result**	**Pass / Fail**	**Remark**
1.	Insert Note	Insert Successfully	Insert Successfully	Pass	_
2.	Update Note	Update Successfully	Null Pointer Exception	Fail	Note id note founded
3.	Delete Note	Delete Successfully	Delete Successfully	Pass	_
4.	Display Record	Display Record in Recycler view	Note Display	Pass	_
5.	Send Notification	Note Create/ Updated Successfully	Note Create/ Updated Successfully	Pass	_
6.	Apply Search Filter	Display Note In Recycler view based on search	Record Not Found	Fail	Note not display

Test Suit Id : 1 Test Case Id : 2 Executed By : bhavin bhesaniya			Description :- This Test case check fix issuses that failed in test case and send note notification reminder.		
No.	**Task**	**Expected Output**	**Actual Result**	**Pass / Fail**	**Remark**
1.	Update Note	Updated Successfully	Updated Successfully	Pass	_
2.	Apply Search Filter	Display Note In Recycler view based on search	Record Displayed	Pass	_
3.	Send Note Reminder	Send Note Reminder Notification	Successfully Notification Send	Pass	_

Limitation

There are nothing we will say that this things is perfect. In this manner there are some limitations are there :-

- If User choose due time before current time will show notification directly at the time note created
- Note not remove automatically after reach due time

Future Enhancement

We will try our best to every aspect of conditions and solve these limitations. In future we will work with to solve this limitation and add new Features like :-

- Add Image In Note Details
- Schedule Note With Date

Bibliography

There are some study materials which is used by me to develop my software system are :-

- https://developer.android.com/
- https://www.geeksforgeeks.org/java/
- https://www.tutorialspoint.com/android/index.htm

=> END OF PROJECT REPORT <=