Development of an Android Mobile Application for Writing Daily Diary

*[1]Atharva Shintre, [2]Kirti Panwar Bhati**
School of Electronics, Devi Ahilya University, Indore, India
[1]shintreatharva@gmail.com,[2]kirti.1809@gmail.com*

***Abstract*—Mobile Diary android app is a useful tool for a person trying to improve themselves. It promotes the user to write about their day, to rate how their day went with respect to a particular aspect such as studies or exercise or helping other people. The app is designed on the basis of the following thought – only by writing, reflecting and introspecting on one's actions can one truly understand oneself and hope to improve. The app also calculates user's number of steps walked in the day. Walking is a great exercise with scientifically known benefits – thus, the app promotes physical as well as mental health of the user. All information written by the user is stored inside a local database and can be viewed when required. User's self-ratings are displayed graphically. A pdf file can be generated of the screen as per the user's convenience.**

***Index Terms*—JAVA, KOTLIN, XML, Daily Diary and SQLITE**

I. INTRODUCTION

The world today is busy and fast-paced. Students, often underprepared, get overwhelmed due to stress from not being able to reach performance goals. Young adults require some help and encouragement to analyse routine and understand mistakes so that they may prevent them in the future from recurring. This app is developed to help youth like in developing a habit to write about themselves and their opinions on their own performance on daily basis. In addition to that, the app also encourages physical health by allowing the user to see their step count and record notes on exercise.

In the "Write Entry" module, the app asks user input on nine parameters of importance. These are – rate your day, study, exercise, e-screen, money, help, hygiene, social interaction and healthy diet. The user can rate themselves with stars ranging from zero to five. Simultaneously, the user has an option to write a note on each of the nine parameters. The app has been divided into four modules. Flow diagram of app navigation is given in Figure 1.

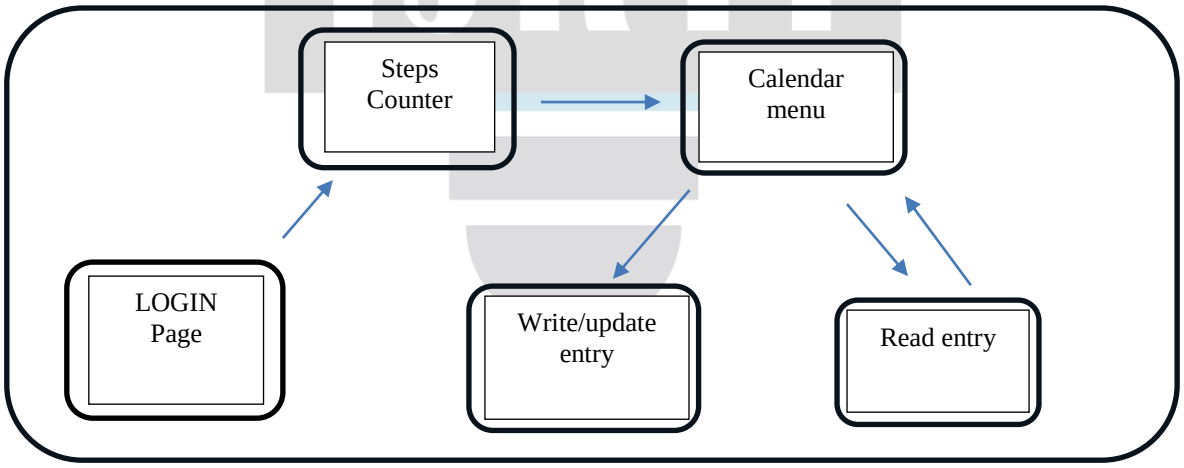


Figure 1 Application Flowchart

Each module of this app serves a special purpose. Additionally, user data is saved into a local database using the SQLite database management system. For this, a database helper class is used along with a user-parameter class. This system allows the user to easily perform CRUD operations on the data.

II. LITERATURE SURVEY

There is plenty of written word on the subject of making an Android application.

Study of the history of the Android operating system. How it came to be with Google leading the Open-Handset Alliance – creating a Linux kernel-based open-source operating system for handsets. A majority of smartphone users around the

world use Android OS based mobile phones. As it is an open-source software, developers are free to modify and use it as they see fit. [3]

The literature survey conducted sheds light on the framework and architecture of the Android OS. Technologies such as "Gradle", which are used in the android applications framework were studied. Basic components of an app, for example "activity" and "service" were studied. Commonly available API and libraries were researched thoroughly on the official android documentation website. The Android Documentation provides in-depth information on all boiler-plate APIs used in android development. Read the usage documents of most commonly used widgets, views, layouts, containers, etc. Recommended practices and protocols were studied to create robust apps.

III. Android Application Architecture

Figure 2 shows a block diagram of the app architecture

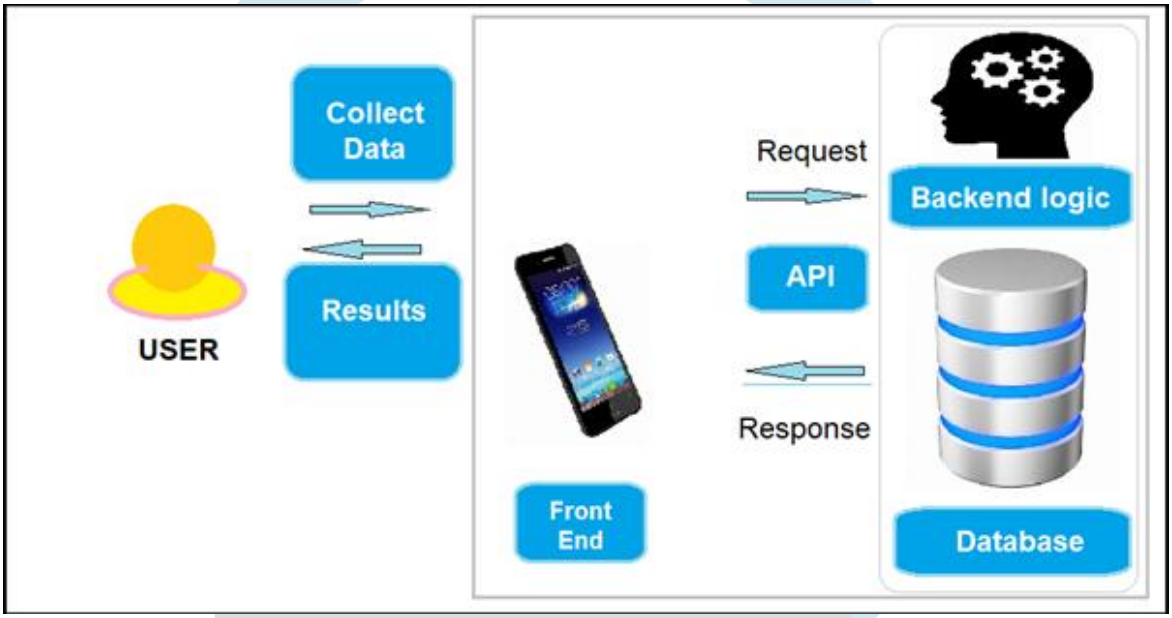


Figure 2 Android application Architecture

IV. Diary app Activity modules

Calendar menu

The home page of the mobile application provides an interactive calendar that can be used to select a date. Two buttons are provided, which serve the following function respectively: Once a date is selected, the user can either write a diary entry/update notes of an existing record or read a diary entry.

Write entry/Update notes

This section of the app allows the user to rate themselves and write notes on various parameters, provide some data such as number of hours studied, exercised, money spent/earned, etc. Steps walked today are shown in the exercise field. User can add a picture to remember the day.

Read diary

The read diary section allows the user to see their previously saved diary entries. It displays data graphically, shows a picture of the day, steps walked, etc. It also shows notes written on that day. Additionally, the user can get an average of their performance over the week, month and year. A maximum of seven photos are displayed while viewing such information. The user can print pdf of the screen by clicking on the save button.

Step counter

The step counter section is not normally visible to the user. It utilizes a Service [1]. This service acts in the background, always counting the user's steps. However, it requires user permission for physical activity to perform its function. When the user has not provided the required permission, the service stops, and this causes a warning to be displayed to the user asking them to give permission, which disappears when the required condition is met.

Login page

Works as an added security measure, it allows only the user to access their personal diary.

Storing data

Data is stored in two forms locally inside the app. The Step counter module stores step count and date in a key-value pair form of shared preferences [2]. The step count information is then sent to the Write diary module where it is displayed in the exercise field. This data, along with other user inputted statistics and notes is then saved into a database.

V. DEVELOPED APPLICATION

Login activity

The log in page is shown when the app is launched. It displays the logo of the app and asks user to enter a username and password. Only by entering the correct credentials can the user navigate to "Write Entry" and "Read Diary" sections of the app.

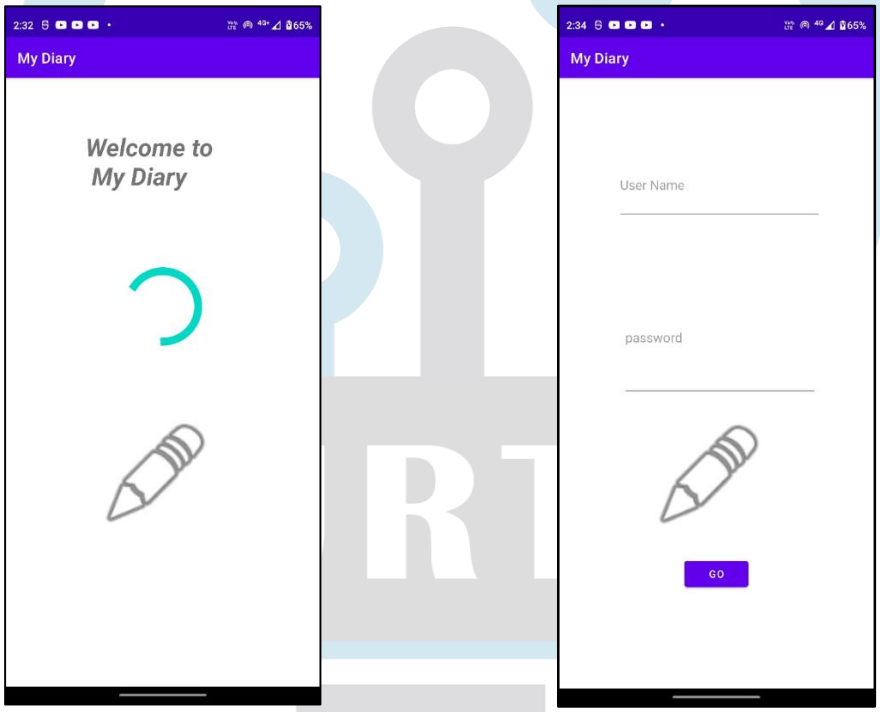


Figure 3: Java code for splash screen in Login activity

Step count activity

After successfully logging in, users will be shown a warning sign asking them to give permission to the app. This is because the "Sensor Service" which counts steps, requires the permission [6] of physical activity to function. When the user gives this permission, the warning sign is no longer displayed, and the app opens directly to the "Calendar Menu" activity.

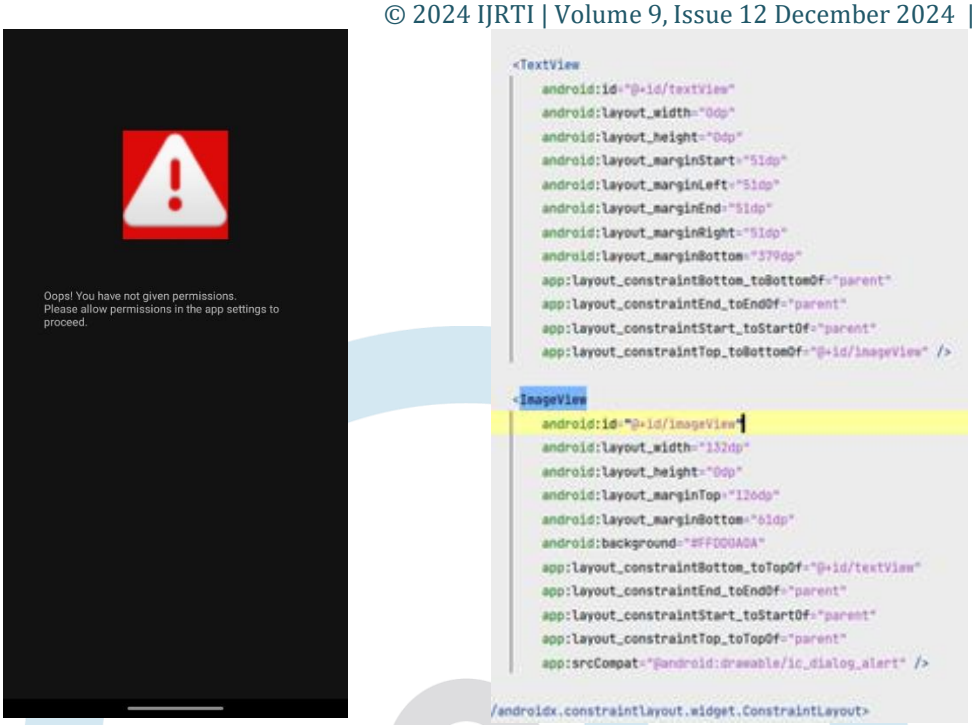


Figure 4 Asking for permission page and code snippet of XML file

The code snippet depicted in Figure 5 is used to extend the sensor event listener class which provides the service of counting steps in the background. The step count is then sent to the next activity. Step count is reset daily to zero.

```
override fun onResume() {
    super.onResume()
    running = true

    // Returns the number of steps taken by the user since the last reboot while activated
    // This sensor requires permission android.permission.ACTIVITY_RECOGNITION.
    // So don't forget to add the following permission in AndroidManifest.xml present in manifest folder of the app.
    val stepSensor = sensorManager?.getDefaultSensor(Sensor.TYPE_STEP_COUNTER)


    if (stepSensor == null) {
        // This will give a toast message to the user if there is no sensor in the device
        Toast.makeText( context: this,  text: "No sensor detected on this device", Toast.LENGTH_SHORT).show()
    } else {
        // Rate suitable for the user interface
        sensorManager?.registerListener( listener: this, stepSensor, SensorManager.SENSOR_DELAY_UI)
        Log.d( tag: "Resume",  msg: "ok")
    }
}

override fun onSensorChanged(event: SensorEvent?) {

    // Calling the TextView that we made in activity_main.xml
    // by the id given to that TextView

    if (running) {
        totalSteps = event!!.values[0]
        Log.d( tag: "totes", totalSteps.toString())
        // Current steps are calculated by taking the difference of total steps
        // and previous steps
        currentSteps = totalSteps.toInt() - previousTotalSteps.toInt()

        // It will show the current steps to the user
        Log.d( tag: "current",  msg: "$currentSteps")
        if (aaikidate != previousDate){
            Log.d( tag: "onsensorchanged", previousDate)
            resetSteps()
        }
```

Figure 5 Kotlin Code snippet for obtaining user step count

Calendar menu

Figure 6 showcases the “Calendar menu”. It features a calendar view – consisting of an interactive calendar and two buttons: Write Start writing and Read diary. The user is prompted to select a date by clicking on a date on the calendar and then proceed to write/update an entry or read it.

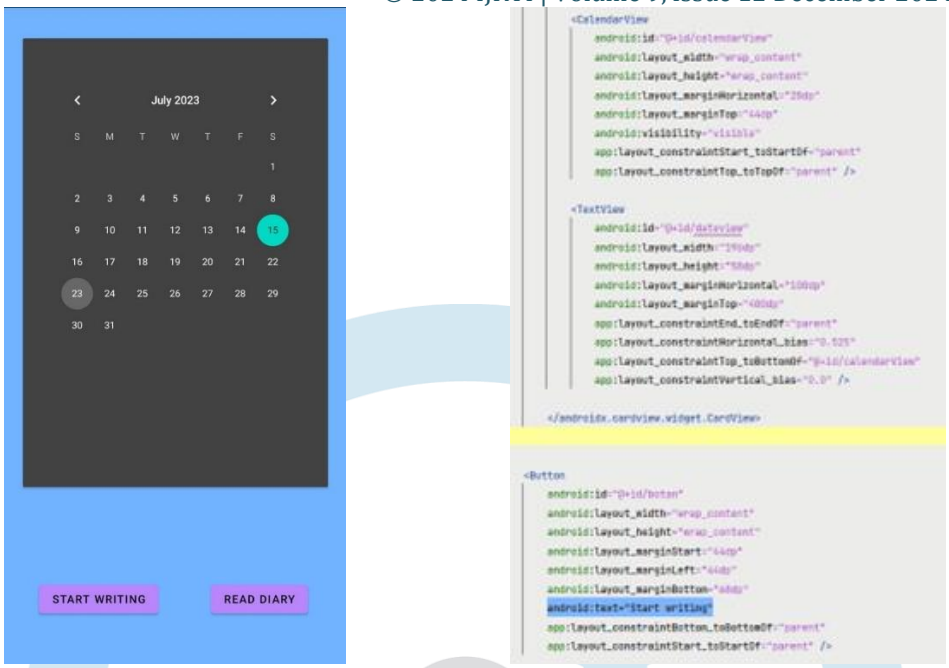


Figure 6 Calendar Menu activity and XML layout

Write/Update Diary Entry page

Figure 7 depicts the mechanism by which the user is able to either write an entry for today or update an entry for any previous day. It contains code snippet of XML layout file.

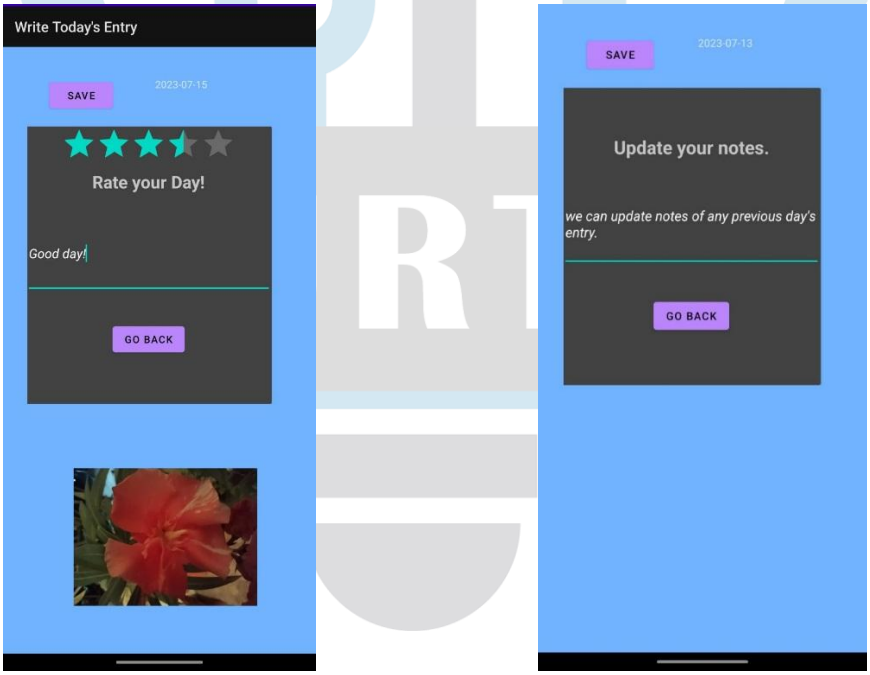
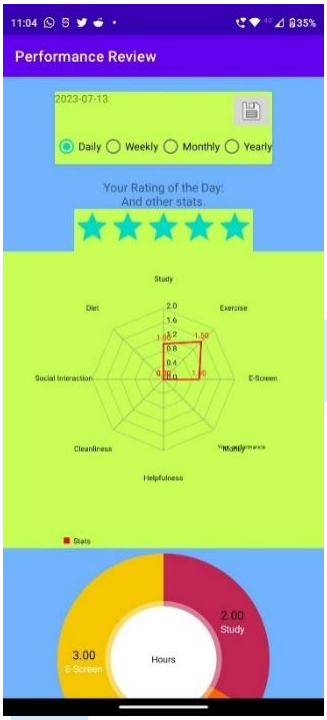


Figure 7 Write/Update entry activity page

Read diary activity

Figure 8 portrays the presentation of a diary entry of a particular day. The user can also see their average performance over a week, month or year. A maximum of 7 pictures are shown while viewing these options. Additionally, the user can print a pdf of their performance by clicking on the print button at the top right of the screen.




```
<ImageButton
    android:id="@+id/savepdf"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:contentDescription="MyDiary"
    app:srcCompat="@android:drawable/ic_menu_save"
    android:layout_gravity="end"/>

<TextView
    android:id="@+id/textView2"
    android:background="@color/lgreen"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />

<RadioGroup
    android:id="@+id/radioGroup2"
    android:layout_marginTop="50dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    >

    <RadioButton
        android:id="@+id/butt1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Daily" />
```

Figure 8 Read diary activity XML layout code snippet

Figure 9 represents the code snippet for the Java logical part of the read diary activity.

```
radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup radioGroup, int i) {
        int test = radioGroup.getCheckedRadioButtonId();
        print.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                bitmap = loadBitmapfromview(linearLayout, linearLayout.getWidth(), linearLayout.getHeight());
                createPdf();
            }
        });
        viewFlipper.startFlipping();
        if (test == findViewById(R.id.butt1).getId()) {
            viewFlipper.setVisibility(View.GONE);
            Float[] rej = dbHandler.getstats(dote);
            dbHandler.close();
            calstats(rej);
        } else if (test == findViewById(R.id.butt2).getId()) {
            Float[] rej = dbHandler.getstatsforxdays(dote, limit: 7);
            dbHandler.close();
            calstats(rej);
            load7img();

        } else if (test == findViewById(R.id.butt3).getId()) {
            Float[] rej = dbHandler.getstatsforxdays(dote, limit: 30);
            dbHandler.close();
            calstats(rej);
            load7img();
        } else {
            Float[] rej = dbHandler.getstatsforxdays(dote, limit: 365);
            dbHandler.close();
            calstats(rej);
            load7img();
        }
```

Figure 9 Read diary Java code snippet

VI. WORKING OF SQLITE DATABASE

Figure 10 depicts the database saved locally inside the mobile app. A table is used that stores data in 25 columns. The primary key for accessing records is date. The database is created using a java class file which provides constructors to store user-inputted data into the database using the database helper class.

	seldate	rateday	special	photo	study	studyhr	studynotes	exercise	exerhr	exernote	stepcount	escreen	escreennot
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	2023-07-01	5.0	great	BLOB	5.0	5.5		5.0	1.0		0	5.0	could do bet
2	2023-07-02	3.0		BLOB	3.5	3.0		3.5	1.0		0	3.5	
3	2023-07-03	2.0		BLOB	1.0	0.0		2.0	0.5		0	1.5	
4	2023-07-04	3.0	decent	BLOB	4.0	4.0	updated: good	3.0	1.0		0	4.0	
5	2023-07-05	4.0		BLOB	3.0	1.0		1.0	0.0		0	1.5	
6	2023-07-06	3.5		BLOB	4.5	5.0		3.5	2.0		0	3.5	
7	2023-07-07	4.0		BLOB	3.5	1.0		3.5	2.0		0	0.5	
8	2023-07-08	4.0		BLOB	4.0	4.0		4.0	2.0		0	4.0	
9	2023-07-09	3.5		BLOB	3.5	2.0		3.5	1.0		0	3.5	
10	2023-07-10	2.5		BLOB	1.5	0.5		1.5	0.5		0	2.0	
11	2023-07-11	3.5		BLOB	3.5	2.0		3.5	3.0		0	4.0	

Figure 10 Viewing app database using DB Browser(SQLite)

Reading Entry from Database

Figure 11 shows that Entry is also read and displayed in the Write Entry activity if a draft has already been saved for today or if notes have been saved for some previous day. The method get notes reads record of a particular date from the database and then showcases those notes in the "Update notes" section for the user to edit and update. Similarly, a method for reading other user inputted data is also used to allow user to update an entry multiple times a day.

```
3 usages
public String[] getnotes(String date){ //also added the photo file path string here
    SQLiteDatabase sqLiteDatabase = this.getReadableDatabase();
    Cursor cursor = sqLiteDatabase.query( table: "mystats", new String[]{"seldate", "special", "studynotes",
            "exernote", "escreennotes", "moneynotes", "helpnotes", "hygnotes", "sointnotes",
            "dietnotes"}, selection: "seldate=?", new String[]{date}, groupBy: null, having: null, orderBy: null);
    if (cursor != null && cursor.moveToFirst()){
        Log.d( tag: "getnote", cursor.getString( i: 2));
        String[] stringer = {cursor.getString( i: 0), cursor.getString( i: 1), cursor.getString( i: 2),
                cursor.getString( i: 3), cursor.getString( i: 4), cursor.getString( i: 5),
                cursor.getString( i: 6), cursor.getString( i: 7), cursor.getString( i: 8),
                cursor.getString( i: 9)};
        cursor.close();
        sqLiteDatabase.close();
        return stringer;

    }
    else {
        String error = "Record does not exist";
        return new String[]{error, error, error, error, error, error, error, error, error, error};
    }
```

Figure 11 Reading data in Write Entry activity

Figure 12 depicts Reading the data for the Read Diary module. This method gives an average of weekly/monthly or yearly performance of the user. It utilizes SQLite queries operators to return the average of the required number of records.

```
public Float[] getstatsforxdays(String date, int limit) {
    SQLiteDatabase sqLiteDatabase = getReadableDatabase();

      String sQuery = "select"
    Cursor cursor = sqLiteDatabase.query( table: "mystats", new String[]{"avg(rateday)", "avg(study)",
                    "avg(exercise)", "avg(escreen)", "avg(mospent)", "avg(help)",
                    "avg(hyg)", "avg(soint)", "avg(dietC)", "avg(studyhr)", "avg(exerhr)", "avg(escreenhours)"
                    , "avg(stepcount)", "avg(moneyhand)"}, selection: "seldate<=?", new String[]{date}, groupBy: null, having: null,
            orderBy: "seldate desc limit "+limit);
    if (cursor != null && cursor.moveToFirst()) {
        Float [] numstack = {cursor.getFloat( i: 0), cursor.getFloat( i: 1), cursor.getFloat( i: 2),
                cursor.getFloat( i: 3), cursor.getFloat( i: 4), cursor.getFloat( i: 5),
                cursor.getFloat( i: 6), cursor.getFloat( i: 7), cursor.getFloat( i: 8),
                cursor.getFloat( i: 9), cursor.getFloat( i: 10), cursor.getFloat( i: 11),
                cursor.getFloat( i: 12), cursor.getFloat( i: 13)};
        cursor.close();
        sqLiteDatabase.close();
       // Log.d("getweek", cursor.getString(2)); //gives null point exce. println in log.d needs non-null value

        return numstack;
        }
    else {
        float error = 0F;
        return new Float[]{error, error, error, error, error, error, error, error, error, error, error, error, error, error};
    }
```

Figure 12 method for obtaining the average performance of a period of days

Figure 13 presents a code snippet that reuses the get stats method which returns user inputted statistics of their performance. The data returned by this method is then populated into radar and pie charts for visual presentation in the “Read Entry” activity. The “MP Android” Library is used to generate graphs and charts.

```
4 usages
public void calstats(Float[] rej) {
    RatingBar ratb = findViewById(R.id.showrate);
    RadarChart radarChart = findViewById(R.id.radar);
    PieChart pieChart = findViewById(R.id.pie);
    TextView showstep = findViewById(R.id.showstep);
    TextView showmoney = findViewById(R.id.showmoney);

    ArrayList<RadarEntry> fivepointparams = new ArrayList<>();
    fivepointparams.add(new RadarEntry(rej[1]));
    fivepointparams.add(new RadarEntry(rej[2]));
    fivepointparams.add(new RadarEntry(rej[3]));
    fivepointparams.add(new RadarEntry(rej[4]));
    fivepointparams.add(new RadarEntry(rej[5]));
    fivepointparams.add(new RadarEntry(rej[6]));
    fivepointparams.add(new RadarEntry(rej[7]));
    fivepointparams.add(new RadarEntry(rej[8]));

    String[] labels = {"Study", "Exercise", "E-Screen", "Money",
    XAxis xAxis = radarChart.getXAxis();
    xAxis.setValueFormatter(new IndexAxisValueFormatter(labels))
    YAxis yAxis = radarChart.getYAxis();
    yAxis.setLabelCount(5);

    RadarDataSet RadarDataSetForFiveParams = new RadarDataSet(fi
    RadarDataSetForFiveParams.setColor(Color.RED);
    RadarDataSetForFiveParams.setLineWidth(2f);
    RadarDataSetForFiveParams.setValueTextColor(Color.RED);
    RadarDataSetForFiveParams.setValueTextSize(10f);
    RadarData radarData = new RadarData();
    radarData.addDataSet(RadarDataSetForFiveParams);
    radarChart.getDescription().setText("Your performance");
    radarChart.setData(radarData);

    ratb.setRating(rej[0]);
```

Figure 13 Charting graphs in read data activity.

Figure 14 presents the code snippet for a method that is used to read data pertaining to picture of the day. The picture is saved as a byte array in the form of a blob data type inside the database. We use the following method to return this blob and convert the byte array to bitmap and then obtaining an image from the bitmap.

```
public byte[] getpic(String date){
    SQLiteDatabase sqLiteDatabase = this.getReadableDatabase();
    Cursor cursor = sqLiteDatabase.query( table: "mystats", new String[]{"photo"}, selection: "pelindro=?", new String[]{date}, groupBy: null, having: null, orderBy: null);
    if (cursor != null && cursor.moveToFirst()){
        byte[] byter = cursor.getBlob( i: 0);
        cursor.close();
        sqLiteDatabase.close();
        //          return new byte[] {cursor.getBlob(0)} //this line gives incompatible type error for some reason
        return byter;
    }
    else{
        return null;
    }
}
```

```
ImageView pic = findViewById(R.id.pic);
TextView potd = findViewById(R.id.potd);
byte[] photon = dbHandler.getpic(dote);
dbHandler.close();
if (photon == null) {
    pic.setVisibility(View.GONE);
    potd.setVisibility(View.GONE);
} else {
    Bitmap bitmap = BitmapFactory.decodeByteArray(photon, offset: 0, photon.length);
    pic.setImageBitmap(bitmap);

}
```

Figure 14 Fetching picture data and converting it to a viewable image.

Additionally, the user can get a pdf copy of their performance screen by clicking on the Print icon. This is achieved using pdf document class. [9]

VII. RESULT

The android mobile application consists of five activities: Login, Step counter, Calendar menu, Write/Update entry and Read Entry. Each of these activities provides diverse functionalities.

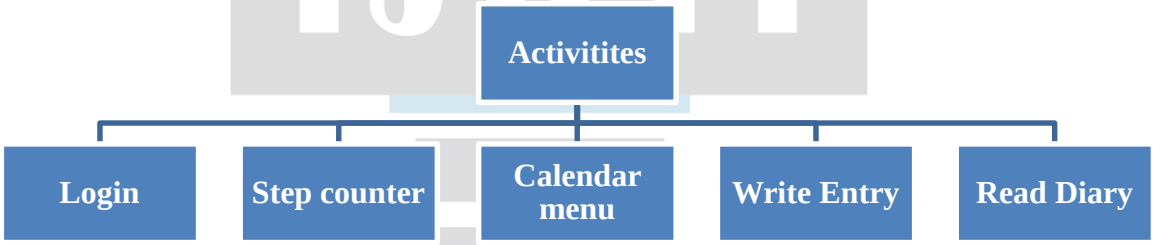


Figure 15 Different activities used in the mobile diary app.

The LOGIN activity allows an extra measure of security to keep the user's data secure and maintain privacy.

The STEP COUNTER activity uses a service that primarily runs in the background – even when the app is closed. The modern smartphone contains a sensor called an accelerometer. This sensor sends a signal when the device experiences acceleration. The step counter service counts a number of times the accelerometer sends this signal to determine the number of steps walked by the user. This activity utilises shared preferences to calculate total steps walked today. When the app is opened, this activity launches first and sends the current number of steps walked to the next activity using intent put extra method.

The CALENDAR MENU activity receives the number of steps walked today from step counter and the relays this data to write entry activity. Additionally, the calendar menu determines if the date selected is today's or not and then accordingly lets the user to either write an entry/update the entry for today or update notes of a previous day's entry.

The calendar menu sends date selected to write entry activity and read entry activity where it is displayed in a text box.

The WRITE ENTRY activity uses add entry or update notes methods of the database helper class to take data inputted by the user in the edit text boxes and fill them up in the database. It also takes step count and date selected values from the calendar menu using intent get String extra method.

The READ ENTRY activity allows the user to look at their diary entry in an appealing format using graphs and charts, pictures and notes. The user can view statistics for today, review their weekly/monthly and yearly performance and print out a pdf of the page.

VIII. CONCLUSION

The Android Mobile Diary is a useful tool for encouraging looking within oneself and trying to understand and improve oneself. Most college students today own a smartphone, but many of them tend to waste this gift of technology on useless activities such as engaging in narcissistic behaviour on social media platforms, wasting time watching web serials, playing video games, etc. Electronic media can cause addiction in its users, which results in students getting depressed.
The diary app will help user to spend their time more wisely and on important topics such as talking to people, maintaining a healthy diet, walking a fixed number of steps daily, studying sincerely, spending money carefully, etc.
The user is also encouraged to check on their performance from previous days every once in a while, and update notes to get a better understanding of how their thought process has changed over time.

REFERENCES

[1] Developers, "Services Overview," [Online]. Available: https://developer.android.com/guide/components/services. [Accessed March 2023]

[2] GeeksforGeeks, "Shared Preferences in Android," [Online]. Available: https://www.geeksforgeeks.org/shared-preferences-in-android-with-examples/.

[3] Wikipedia, "Android OS," [Online]. Available: https://en.wikipedia.org/wiki/Android_(operating_system).

[4] DbBrowser, "Using DbBrowser," [Online]. Available: https://sqlitebrowser.org/. [Accessed April 2023].

[5] A. Developers, "Data Storage in Android," [Online]. Available: https://developer.android.com/training/data-storage/. [Accessed April 2023].

[6] A. Developers, "Permissions in android," [Online]. Available: https://developer.android.com/guide/topics/permissions/overview. [Accessed April 2023].

[7] A. Developers, "Sqlite in Android," [Online]. Available: https://developer.android.com/training/data-storage/sqlite. [Accessed May 2023].

[8] PhilJay, "MPAndroidChart," [Online]. Available: https://github.com/PhilJay/MPAndroidChart. [Accessed May 2023].

[9] A.Developers, "Pdf Document in Android," [Online]. Available: https://developer.android.com/reference/android/graphics/pdf/PdfDocument. [Accessed June 2023]