Android Project Report on

MINESWEEPER

Student 1 – 191IT129

Student 2 – 191IT128

Student 3 – 191IT248

Student 4 – 191IT106

Department of Information Technology, NITK Surathkal

Date of Submission: 14 June 2020

in partial fulfillment for the award of the degree

of

Bachelor of Technology

In

Information Technology

At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

June 2020

Department of Information Technology, NITK Surathkal
Android Project
Second Semester Evaluation Report (June 2020)

Course Code: IT150

Course Title: Android Project

Project Title: MineSweeper

Project Group:

Name of the Student	Register No.
Maheshwari Mihir Premjibhai	191IT129
Kunal Singh Lohiya	191IT128
Sahildeep Buttar	191It248
Atul Kumar Singh	191IT106

Place: National Institute of Technology Karnataka, Surathkal

Date: 14th June 2020

Contents

1 Introduction

Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the *Open Handset Alliance*, led by Google, and other companies.

Basically, this app is an Android Mobile Game App called “MineSweeper”. It is a single player game. It comes under category of *Games/Entertainment Mobile Apps*. This App is not like other usual game mobile apps. But to play Minesweeper game one has Brainstorm to locate the mines on the grid board and Flag all the mines It’s something like “Do or Die” condition to play. So, this game is not like other mobile apps which are usually follows same pattern, as in this every time random mines are placed on the grid board. So, it can be called as brain game.

According to English, Minesweeper is an agent who search for bombs and diffuse them. This Game is also based on this, that is one has to search the mine on the grid board and flag it.

This Work was done by distributing the project into parts and each parts were done by team members. And we have used whatsapp and Microsoft Teams to co-ordinate with each other.

The brief about the Game is as follows:

MineSweeper App User Manual

Basically This app is a Game app called "MineSweeper". In this game one has to brainstorm and locate the mines on the board.

How to Play & Rules?

* The Game consists of a board with a 12x8 grid.
* On this Grid mines/bombs are present (*these mines are fixed randomly and are placed at different positions every time*).
* If User clicks the mines/bombs then user lose the game.
* Whenever user clicks a position then a number is displayed signifying the number of Bombs around that position in 3x3 matrix considering that position a center. (If complete matrix is not present then only the one layer of squares around particular is considered and number of mines around that position is displayed on that particular.)
*if the user detects any mine then, he/she can click the Bomb Button at the Top-Right Corner of screen, and it transits to Flag icon.
Now if user clicks any square then, it will be marked as flag. *User can Flag a mined and even non-mined square/position.
*On Again clicking the Flag icon at the top-Left corner of screen, the flag icon will transits to Bomb icon which means user is again in the danger Zone (As if he/she clicks the bomb then Will Lose the Game)
*If square clicked is not a mine, then score is added by the number on the square.

How to Win?

*If the user survives to flag all the Mines successfully then, the **Winner becomes the MineSweeper.**

2 Work Done

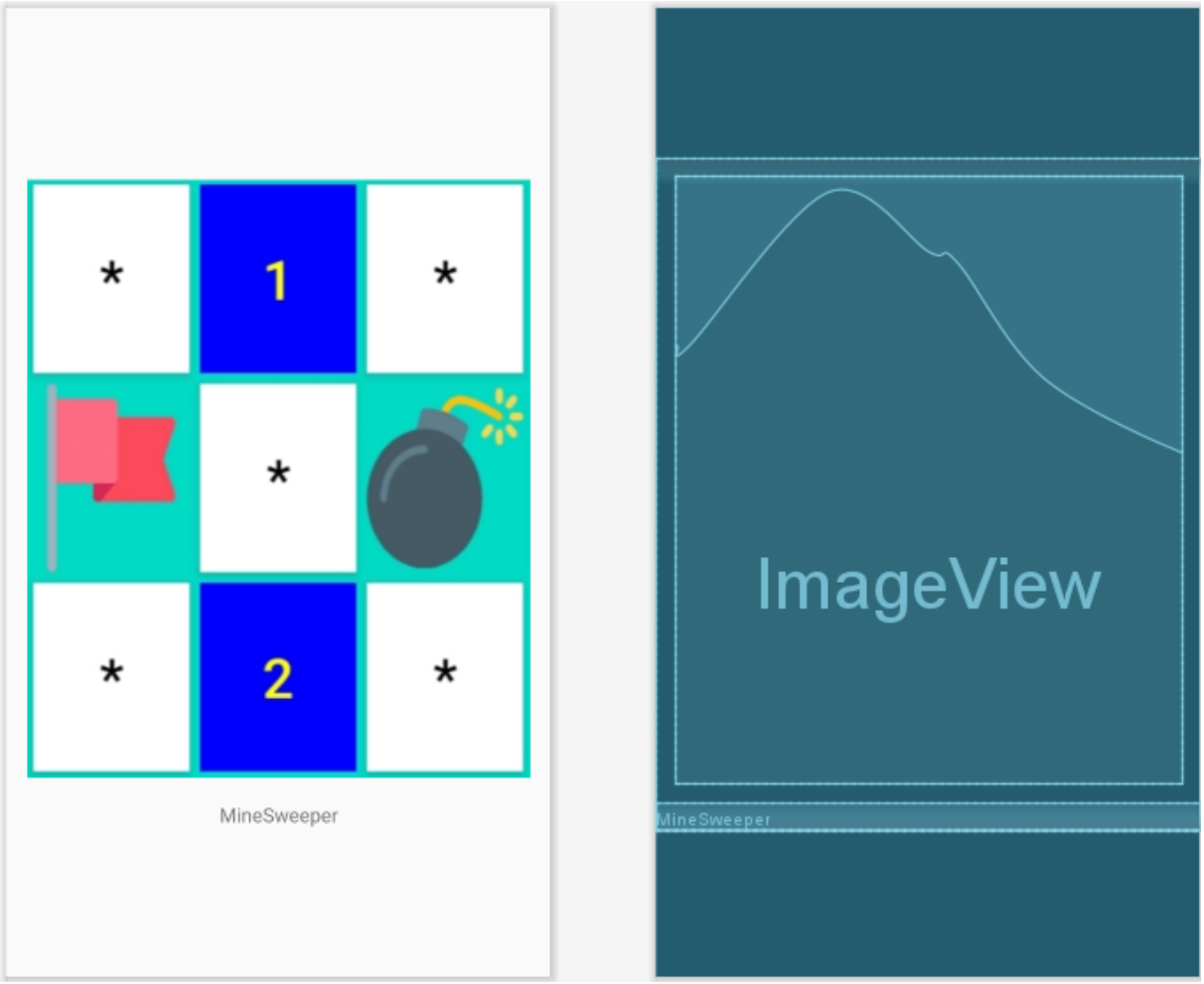
App includes the following .java files and Corresponding .xml files:

1. OnStart.java → activity_on_start.xml
2. MainActivity.java → activity_main.xml
3. UserMAnual.java → activity_user_manual.xml

The Design View of the respective front-end source and the Source code in java is as shown below:

1) activity_on_start.xml & OnStart.java :

In this ImageView and a TextView is used to create the Splash the app icon only when app is initiated.



activity_on_start.xml × OnStart.java × activity_main.xml × MainActivity.java × Use

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".OnStart"
    android:orientation="vertical"
    android:layout_gravity="center">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="15dp"
        android:src="@drawable/ic_icon_minesweeper" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="MineSweeper"
        android:gravity="center"
        android:textSize="15dp"/>
</LinearLayout>
```

XML Source code.

activity_on_start.xml × OnStart.java × activity_main.xml × MainActivity.java × UserManual.java × string

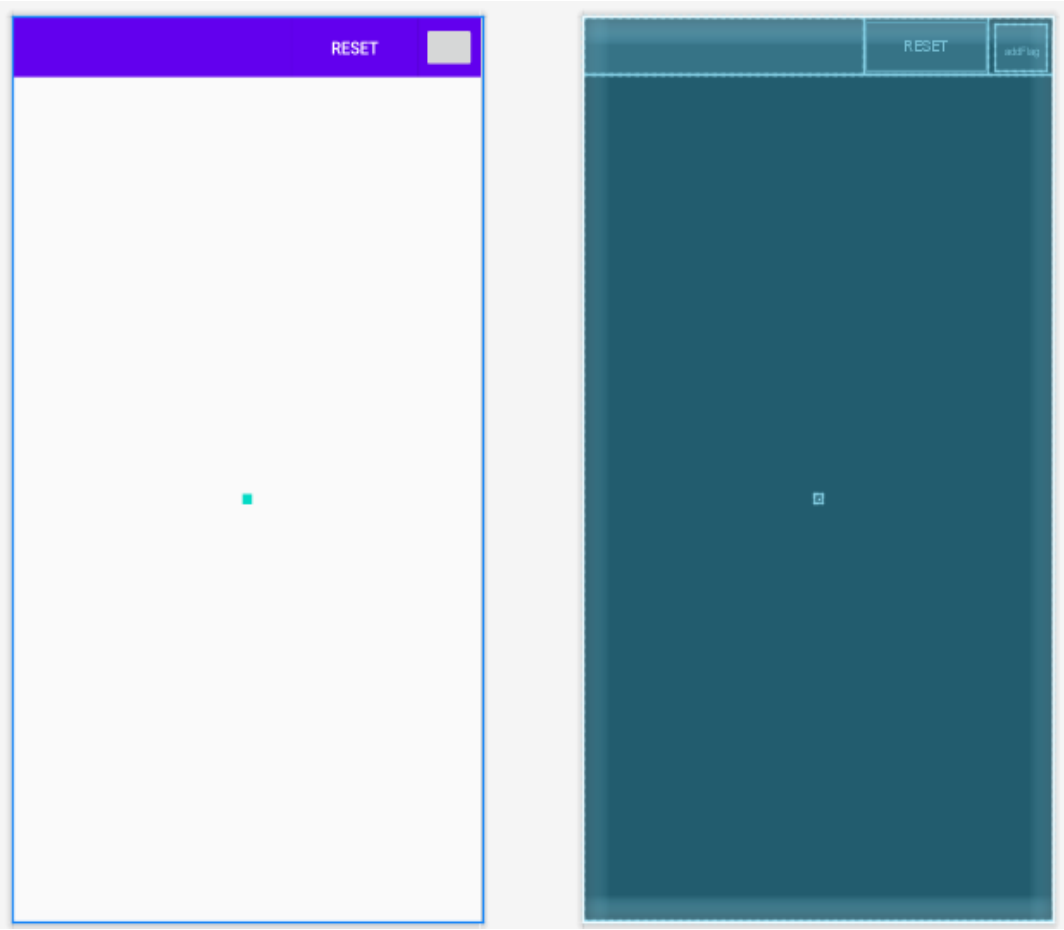
```java
package com.faltu.minesweeperoptimized;

import ...

public class OnStart extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_on_start);

        Handler handler = new Handler();
        handler.postDelayed(() -> {
                Intent intent =new Intent( packageContext: OnStart.this,MainActivity.class);
                startActivity(intent);
                finish();
        }, delayMillis: 2500);
    }
}
```

Java Source Code.

In java code postDelayed() for handler is used to splash the app logo for 2500 milliseconds(2.5seconds), then using intent MainActivity is called and finally this activity is destroyed using finish() method.

2) **activity_main.xml & MainActivity.java :**

In this a TextView to display Score, Button with text “RESET” and an ImageButton is used in LinearLayout with orientation horizontal. And a GridLayout is added with gravity as center and columnNum.



The Buttons in the GridLayout are added Dynamically during Run time in the java source code for this in MainActivity.java

In MainActivity.java initially createMatrix() method is called which creates the back-end matrix of 12x8 called matrix[][].

```java
    private void createMatrix() {
        //initially matrix is -51
        //and Bomb is where matrix[i][j]=-51, where -51 is any random no.
        //It's one of my favourite no.
        for(int i=0;i<n;i++)
        {
            Arrays.fill(matrix[i], val: -51);
        }
        //Bomb is placed in matrix where matrix[i][j]=-1 && for matrix[i][j]=-51 it is empty
        placeBombs();
        //Code for locating no in the matrix;
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<m;j++)
            {
                if(matrix[i][j]==-1)
                {
                    System.out.print("+");
                    continue;
                }
                else
                {
                    matrix[i][j]=noOfBombsAround(i,j);
                    System.out.print(matrix[i][j]);
                }
            }
            System.out.println("");
        }
    }
```

createMatrix() method.

Then placeBomb() method is called internally by the createMatrix(), which sets the mines in the matrix.

```java
    private void placeBombs() {

        Random rand=new Random(); //creating a rand object of java.util.Random class in order, to use rand method.
        //Code for Locating Bomb;

        //Bombi is array of x-co-ordinate of random place where bomb is to be placed
        //Bombj is array of y-co-ordinate of random place where bomb is to be placed
        Arrays.fill(Bombi, val: -1);
        Arrays.fill(Bombj, val: -1);

        for(int i=0;i<noOfBomb;i++) {
            int flag=1;
            //here a and b are random no between 0 to n and 0 to m respectively.
            int a=rand.nextInt(n);
            int b=rand.nextInt(m);
            for(int j=0;j<noOfBomb;j++) {
                if(Bombi[j]==a&&Bombj[j]==b) {
                    i--;
                    flag=0;
                    break; }
            }if(flag==1) {
                Bombi[i]=a;
                Bombj[i]=b;
                matrix[a][b]=-1;
            }
        }
        //debugging the position of bombs.
        for(int i=0;i<noOfBomb;i++)
        {
            System.out.println(Bombi[i]+"  "+Bombj[i]);
        }
    }
```

placeBomb() method.

placeBombs() method place the bombs at the random places at 15 places as the noOfBomb=15.

Hence the matric[][] of size 12x8 is created with int datatype and in which -1 is filled where the bomb is present, and other places are filled with -51.

Now addButtons() method is called which add the buttons in the gridLayout Dynamically.

```
private void addButtons() {
    GridLayout layout = (GridLayout) findViewById(R.id.rootLayout);
    RelativeLayout.LayoutParams params = new RelativeLayout.LayoutParams( w: 120,ViewGroup.LayoutParams.WRAP_CONTENT);
    int marginValue=4;
    params.bottomMargin=marginValue;
    params.topMargin=marginValue;
    params.leftMargin=marginValue;
    params.rightMargin=marginValue;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            buttonArray[i][j]=new Button( context: this);
            buttonArray[i][j].setText("*");
            buttonArray[i][j].setLayoutParams(params);
            buttonArray[i][j].setWidth(layout.getMeasuredWidth());
            buttonArray[i][j].setBackgroundColor(Color.WHITE);
            layout.addView(buttonArray[i][j]);
        }
    }
}
```

addButtons() method

hence the buttons are created with the array of buttons namely buttonArray[][] in the GridLayout and text is set to “*”.

And then showScore() method is called, which initially set the score to textView to “Score:0”

```
private void showScore() {
    //Update the Scores.
    score.setText("  Score : "+Integer.toString(score_count));
}
```

```
65
66      addFlag.setBackgroundResource(R.drawable.ic_bomb);
67      addFlag.setOnClickListener((v) → {
71          //Code to Toggle addFlag Button between Flag and Bomb.
72          if(flagClickNumber==0||flagClickNumber==-1) {
73              flagButtonClicked=true;
74              flagClickNumber=1;
75              addFlag.setBackgroundResource(R.drawable.ic_flag_bomb);
76          }else{
77              flagButtonClicked=false;
78              addFlag.setBackgroundResource(R.drawable.ic_bomb);
79              flagClickNumber=0;
80          }
81      });
83
```

Code for functionality of addFlag button

This code is for the functionality of the adddFlag button on top right corner i.e. to toggle the button between add Flag and Bomb icon.

Then the code for functionality of the buttons is set which is based on whether the addFlag button is Flag or Bomb. This code increment the score and set the text of score TextView to the score.
Score: *score is based on the number in the square if square is not mined.*

```
for(int i=0;i<n;i++)
{
    for(int j=0;j<m;j++)
    {
        final int finalI = i;
        final int finalJ = j;
        buttonArray[i][j].setOnClickListener((v) -> {
                if(flagButtonClicked){
                    buttonArray[finalI][finalJ].setText("");
                    buttonArray[finalI][finalJ].setBackgroundResource(R.drawable.ic_flag_bomb);
                    if(matrix[finalI][finalJ]==-1){
                        safeFlagCount++; //incrementing safeFlagCount if Mine is Flagged.
                    }
                }
                else {
                    if (matrix[finalI][finalJ] != -1) {
                        /*
                        if clicked Button is not a Mine.
                        then diaplaying no of Mines Around.
                        */
                        buttonArray[finalI][finalJ].setBackgroundColor(Color.BLUE);
                        buttonArray[finalI][finalJ].setTextColor(Color.YELLOW);
                        buttonArray[finalI][finalJ].setText(Integer.toString(matrix[finalI][finalJ]));
                        buttonArray[finalI][finalJ].setEnabled(false);
                        score_count+=matrix[finalI][finalJ];
                        if(matrix[finalI][finalJ]==0)
                        {
                            score_count++;
                        }
                        showScore();
                        safeClickCount++;
                    }
                }
```

*And when all the buttons are pressed and all the mines are flagged then the user **Wins** the Game.This also calls the showScore() method which displays the score.*

```
Boolean callDialogBox=false;
String RESULT_IN_DIALODUE = null;
int idForIcon=R.id.reset;
if(safeFlagCount==noOfBomb&&safeClickCount==((n*m)-noOfBomb)){
    RESULT_IN_DIALODUE="You Won The Game with Score : "+Integer.toString(score_count);
    Toast.makeText( context: MainActivity.this, text: "Winnner!!!",Toast.LENGTH_SHORT);
    callDialogBox=true;
}
if(matrix[finalI][finalJ]==-1&&!flagButtonClicked){
    buttonArray[finalI][finalJ].setText("");
    for(int x=0;x<noOfBomb;x++) {
        buttonArray[Bombi[x]][Bombj[x]].setText("");
        buttonArray[Bombi[x]][Bombj[x]].setBackgroundColor(Color.RED);
        buttonArray[Bombi[x]][Bombj[x]].setBackgroundResource(R.drawable.ic_bomb);
    }
    RESULT_IN_DIALODUE="You Lost the Game\n Score : "+Integer.toString(score_count);
    callDialogBox=true;
    Toast.makeText( context: MainActivity.this, text: "You Lost -_-",Toast.LENGTH_SHORT);

}
```

Now the winner condition is checked and if bomb is clicked then the dialog box is prompt which asked to restart the game or to see the doubt in the result.

```
if(callDialogBox) {
    /*
    DialogBox is Called if Game is Lost or Won.
    */
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder( context: MainActivity.this);
    alertDialogBuilder.setTitle(RESULT_IN_DIALODUE);
    alertDialogBuilder.setIcon(R.drawable.ic_restart);
    alertDialogBuilder.setMessage("Do you want to play again!!");
    alertDialogBuilder.setCancelable(false);
    alertDialogBuilder.setPositiveButton( text: "Yes", (dialog, which) -> {
            //if user request to play again
            resetGame();//Resets the Game Board.
    });
    alertDialogBuilder.setNegativeButton( text: "No", (dialog, which) -> {
            Toast.makeText( context: MainActivity.this, text: "Press RESET to restart the Game!!",Toast.LENGTH_SHORT).show();
            setButtonsDisable();//Board is set Disabled.
    });
    alertDialogBuilder.create().show();
}
```

Code for dialog box

If **No** is opted in dialog box: setButtonsDiable() method is called which disables all the. So, that so button can be pressed again.

```java
private void setButtonsDisable() {
    //Disable All Buttons when the bomb is clicked i.e. LOST Game.
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            buttonArray[i][j].setEnabled(false);
        }
    }
}
```

setButtonsDiasble() method

Now reset Button is added functionality to resst the game board which called resetGame() method.

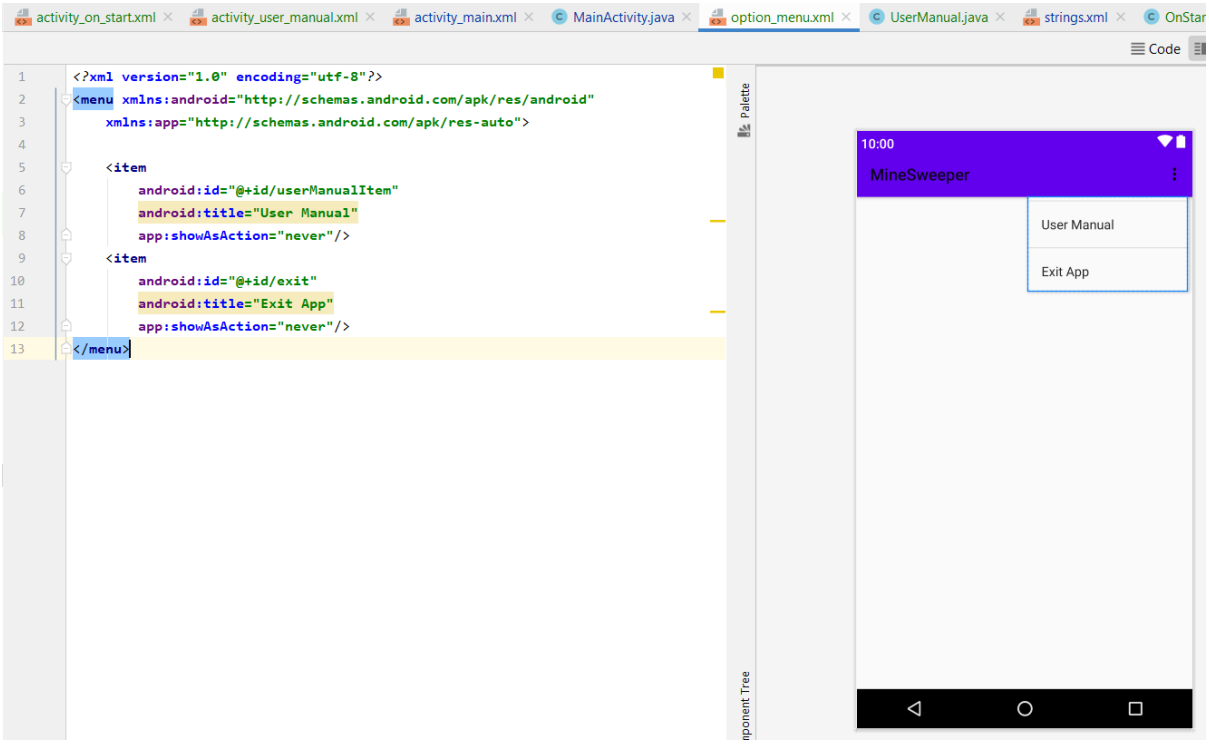
```java
private void resetGame() {

    layout.setEnabled(false);
    createMatrix();
    safeClickCount=0;
    safeFlagCount=0;
    addFlag.setBackgroundResource(R.drawable.ic_bomb);
    score.setText("  Score : 0");
    score_count=0;
    flagButtonClicked=false;
    flagClickNumber=-1;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            buttonArray[i][j].setEnabled(true);
            buttonArray[i][j].setText("*");
            buttonArray[i][j].setBackgroundColor(Color.WHITE);
            buttonArray[i][j].setTextColor(Color.BLACK);
        }
    }
}
```

resetGame() method

The code for option menu at the top right coner in the in the toolbar

```java
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.option_menu,menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId())
        {
            case R.id.userManualItem:
                Intent intent=new Intent( packageContext: MainActivity.this,UserManual.class);
                startActivity(intent);
                return true;
            case R.id.exit:
                onBackPressed();
        }
        return super.onOptionsItemSelected(item);
    }
```

Option Menu code.



Frontend Code for Option Menu

This is all about the activity occurring due to backend source code MainActivity.java

3) activity_user_manual.xml & UserManual.java :

This is made for the *user manual* for the MineSweeper game app. In this a TextView is added,which shows the context of the TextView.

activity_on_start.xml | activity_user_manual.xml | OnStart.java | activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UserManual"
    android:padding="7dp">

    <TextView
        android:id="@+id/userManual"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

XML Source code for user manual

The TextView is set to the html code which is in the string.xml file and an html source is set as value of String name userManual.

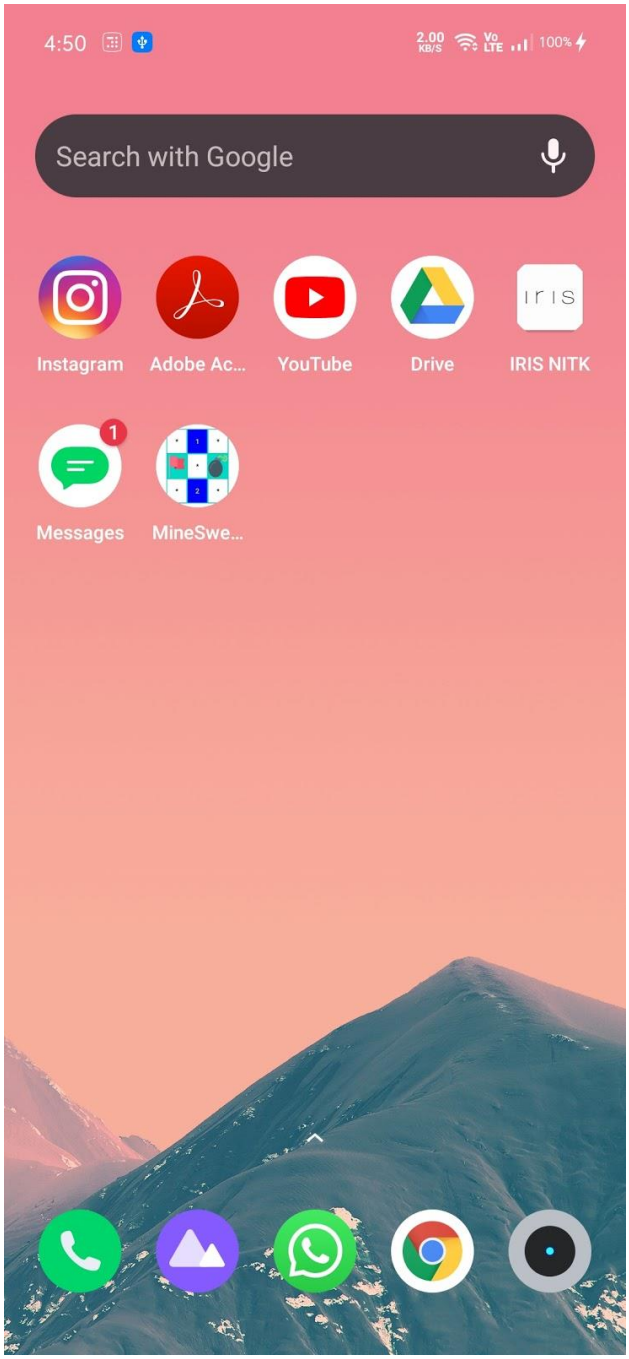
activity_on_start.xml | OnStart.java | activity_main.xml | MainActivity.java | UserManual.java

```java
package com.faltu.minesweeperoptimized;

import ...

public class UserManual extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user_manual);
        TextView userManual= (TextView)findViewById(R.id.userManual);
        userManual.setText(Html.fromHtml(getString(R.string.userManual)));
    }
}
```

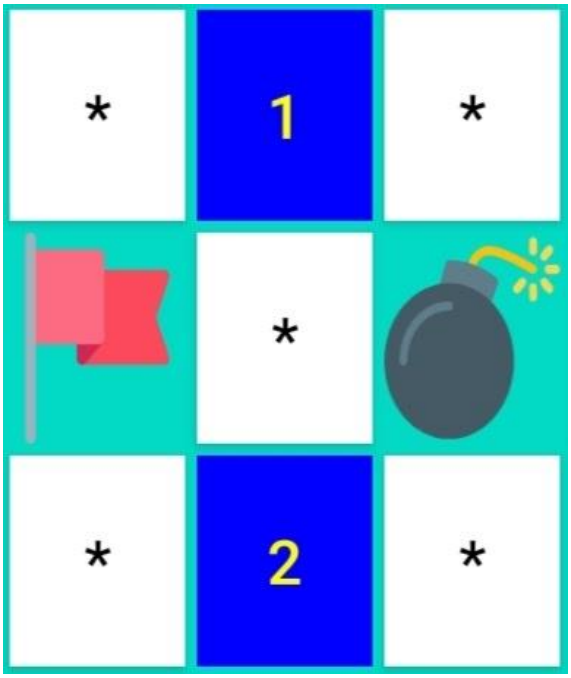
Code to implement the html code to the TextView

3 Results and Discussion

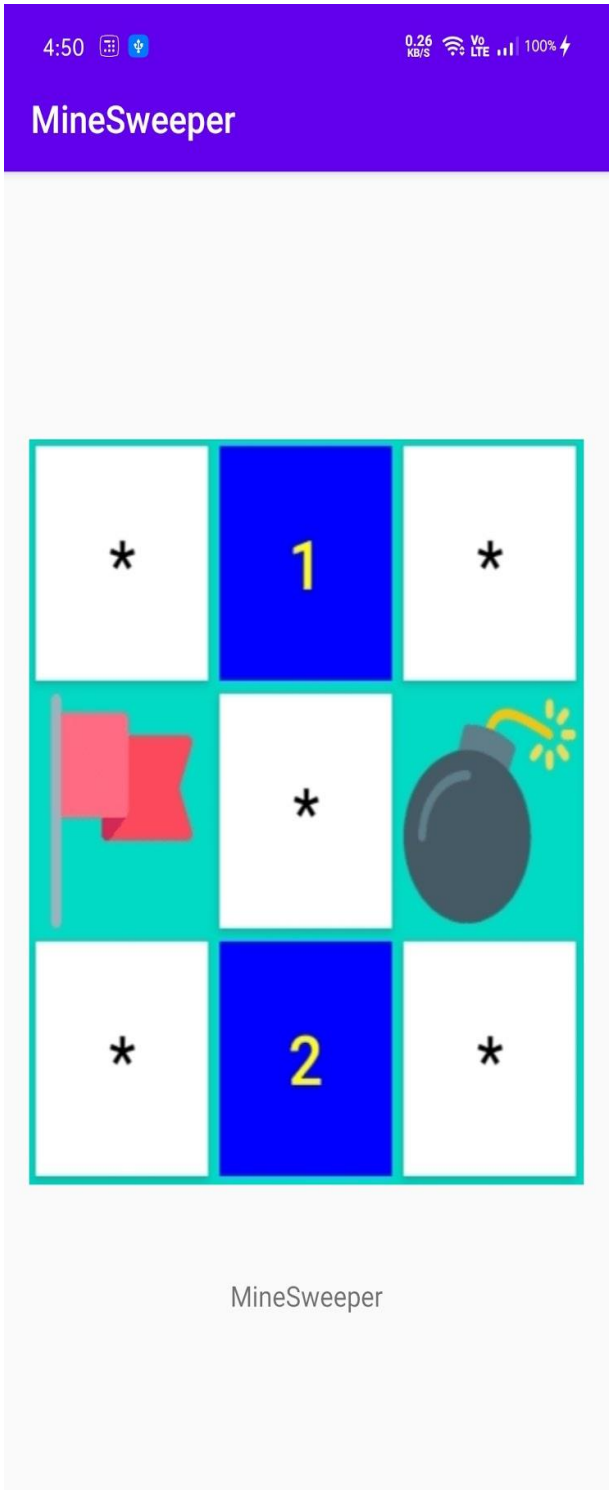
The ScreenShots of the Game with all the possibly covered cases.



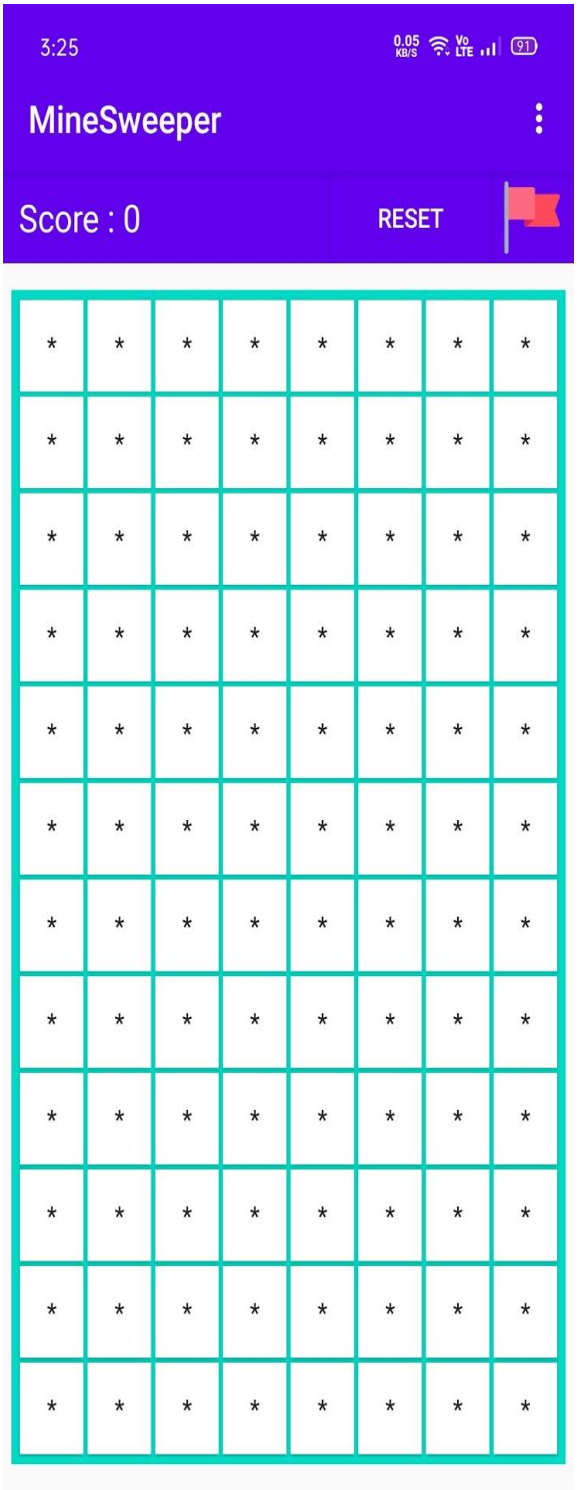
App as seen in the android mobile.



App icon of MiniSweeper Game



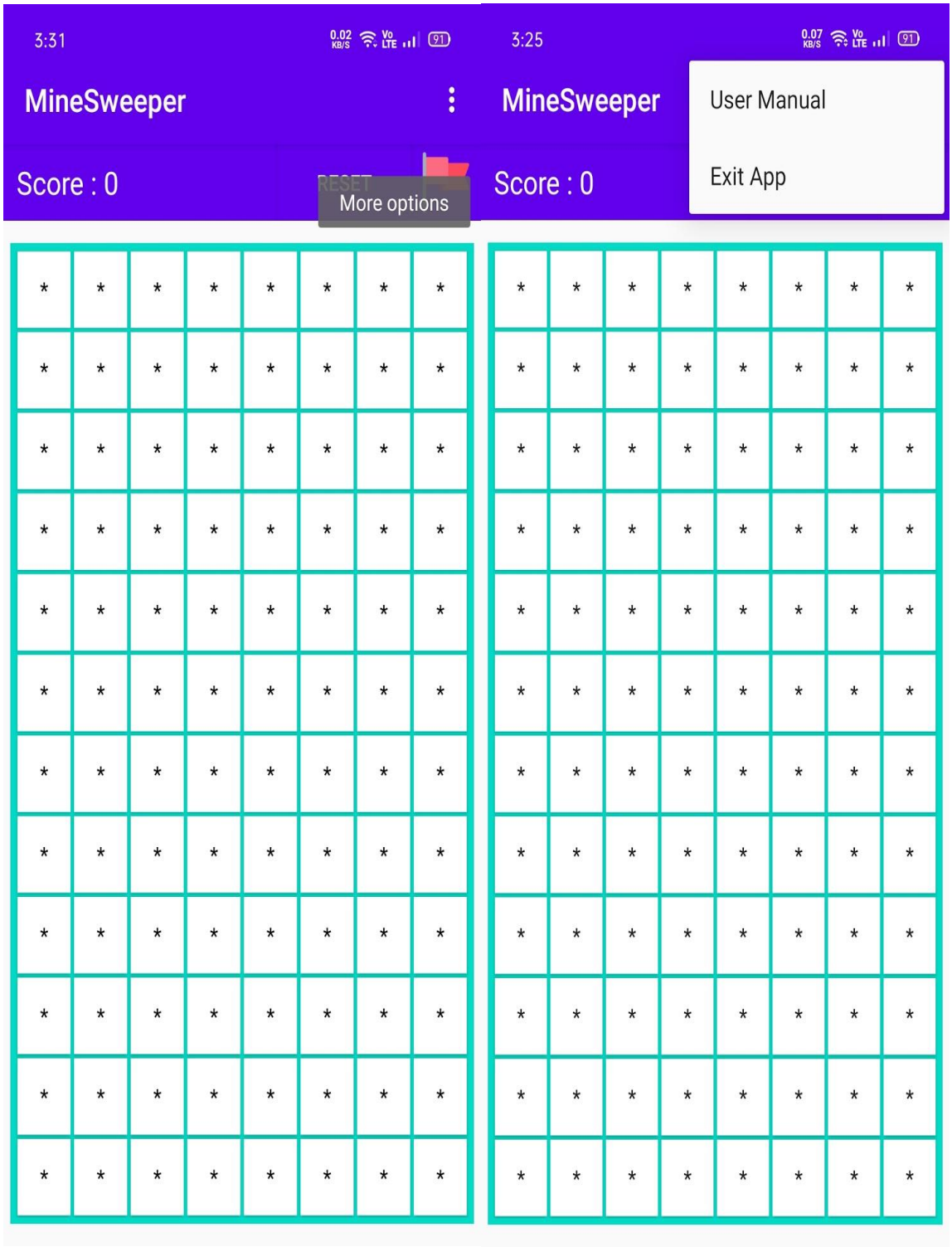
Splash Screen in the beginning



Game Board

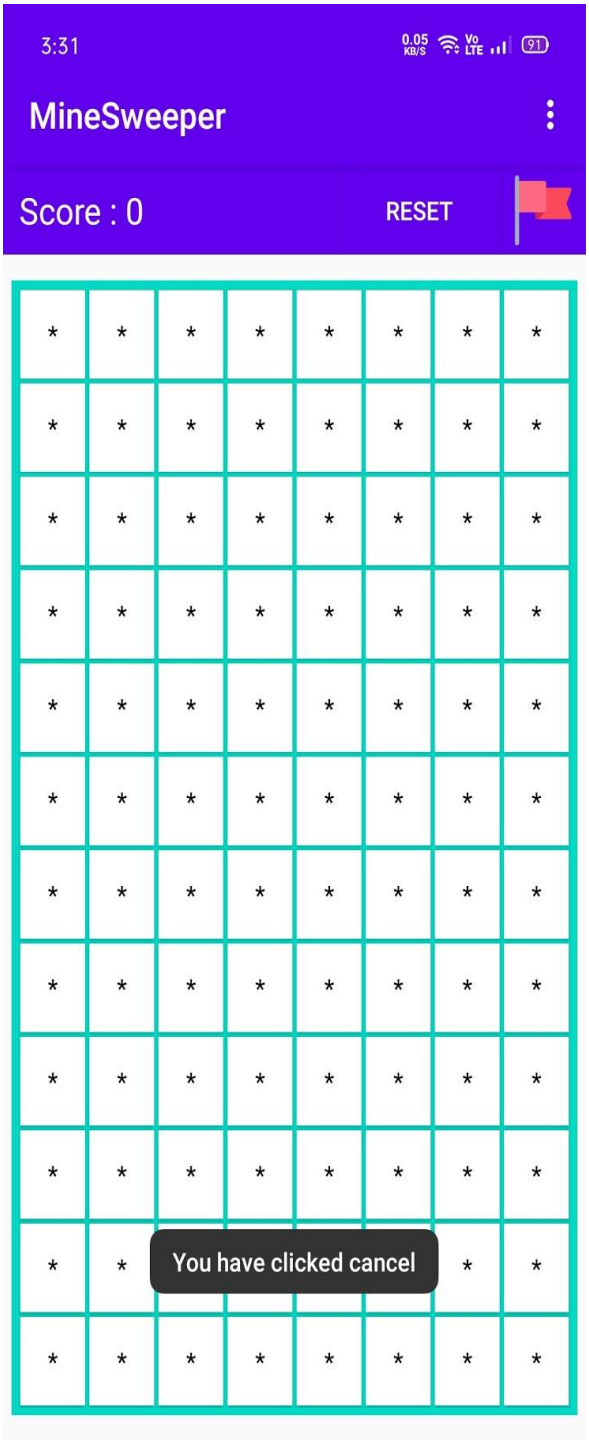
After opening the app first diagram will come on your screen.

This is the default screen.There game begins.In the second diagram gird is given we have to avoid to click on the mines and score maximum .In second diagram given score, grid, menu button, change '*' to flag button.



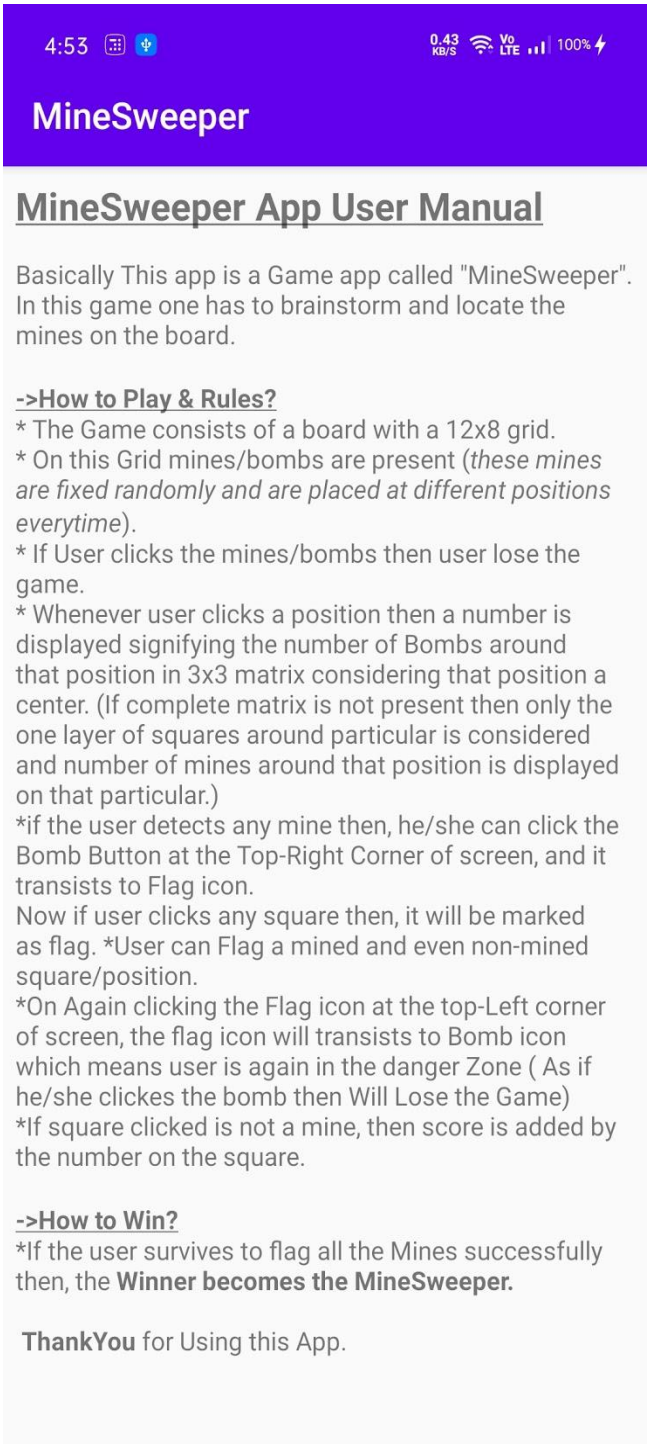
This diagram show that clicking on more option button(three dot button) it show more option that contain two choice first one is user manual which show about app and other one is Exit app which exit you from the app.

Dialog box ask to exit



If “NO” then toast is displayed.

After clicking on exit first Dialog box will come to on the screen that confirm that you want to exit or not. If you click on “YES” that it will exit and if you click on “NO” than is it will cancel the decision to exit the app.And Toast is displayed



4:53

MineSweeper

MineSweeper App User Manual

Basically This app is a Game app called "MineSweeper". In this game one has to brainstorm and locate the mines on the board.

->How to Play & Rules?

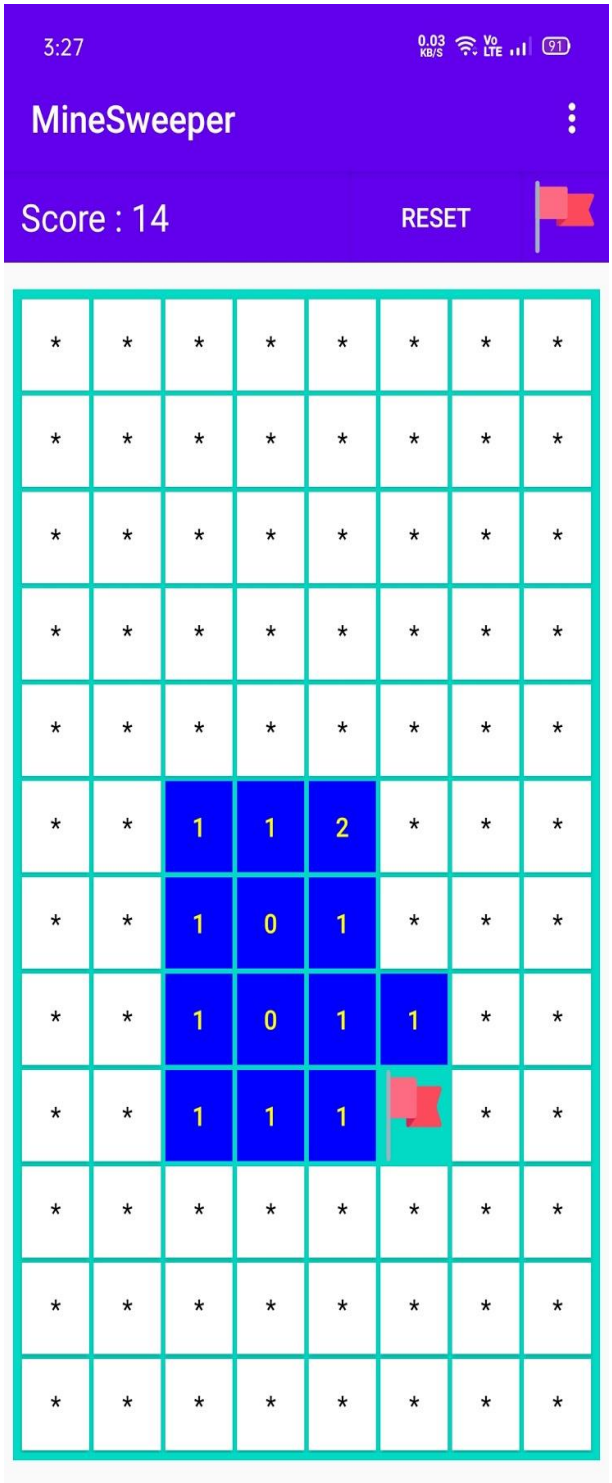
* The Game consists of a board with a 12x8 grid.
* On this Grid mines/bombs are present (*these mines are fixed randomly and are placed at different positions everytime*).
* If User clicks the mines/bombs then user lose the game.
* Whenever user clicks a position then a number is displayed signifying the number of Bombs around that position in 3x3 matrix considering that position a center. (If complete matrix is not present then only the one layer of squares around particular is considered and number of mines around that position is displayed on that particular.)
*if the user detects any mine then, he/she can click the Bomb Button at the Top-Right Corner of screen, and it transists to Flag icon.
Now if user clicks any square then, it will be marked as flag. *User can Flag a mined and even non-mined square/position.
*On Again clicking the Flag icon at the top-Left corner of screen, the flag icon will transists to Bomb icon which means user is again in the danger Zone (As if he/she clickes the bomb then Will Lose the Game)
*If square clicked is not a mine, then score is added by the number on the square.

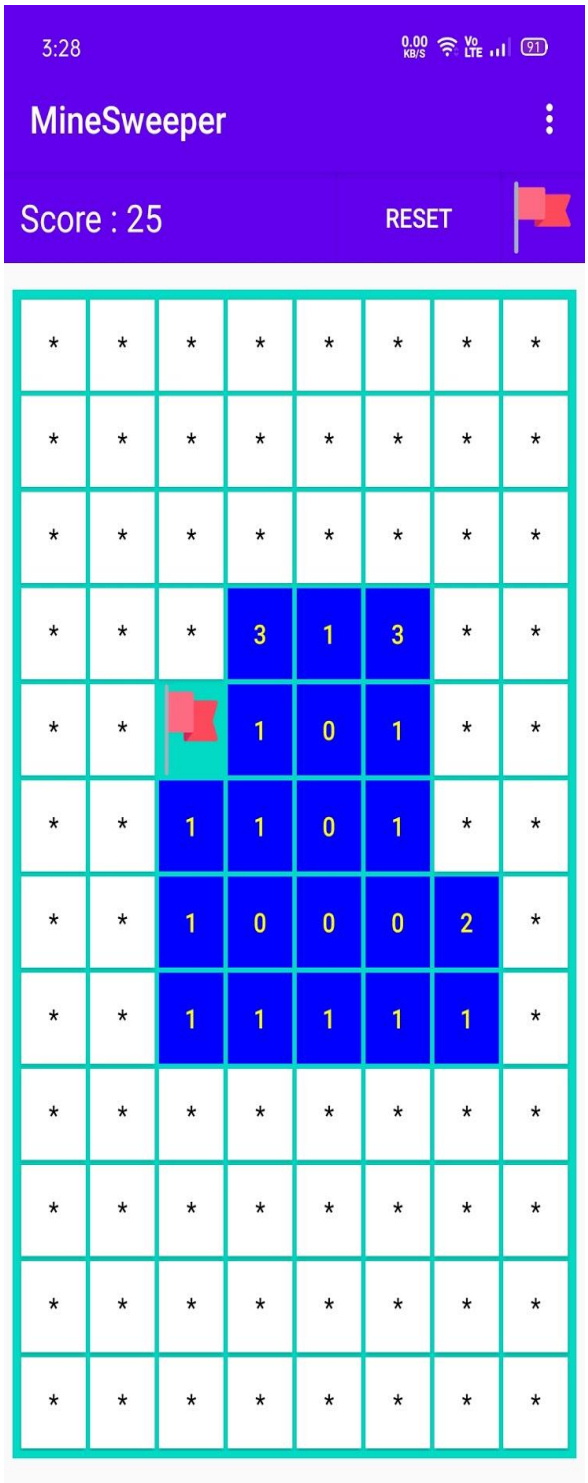
->How to Win?

*If the user survives to flag all the Mines successfully then, the **Winner becomes the MineSweeper.**

ThankYou for Using this App.

On clicking on the user manual this will show you that show how to play the game and about the game.And how to win the game.

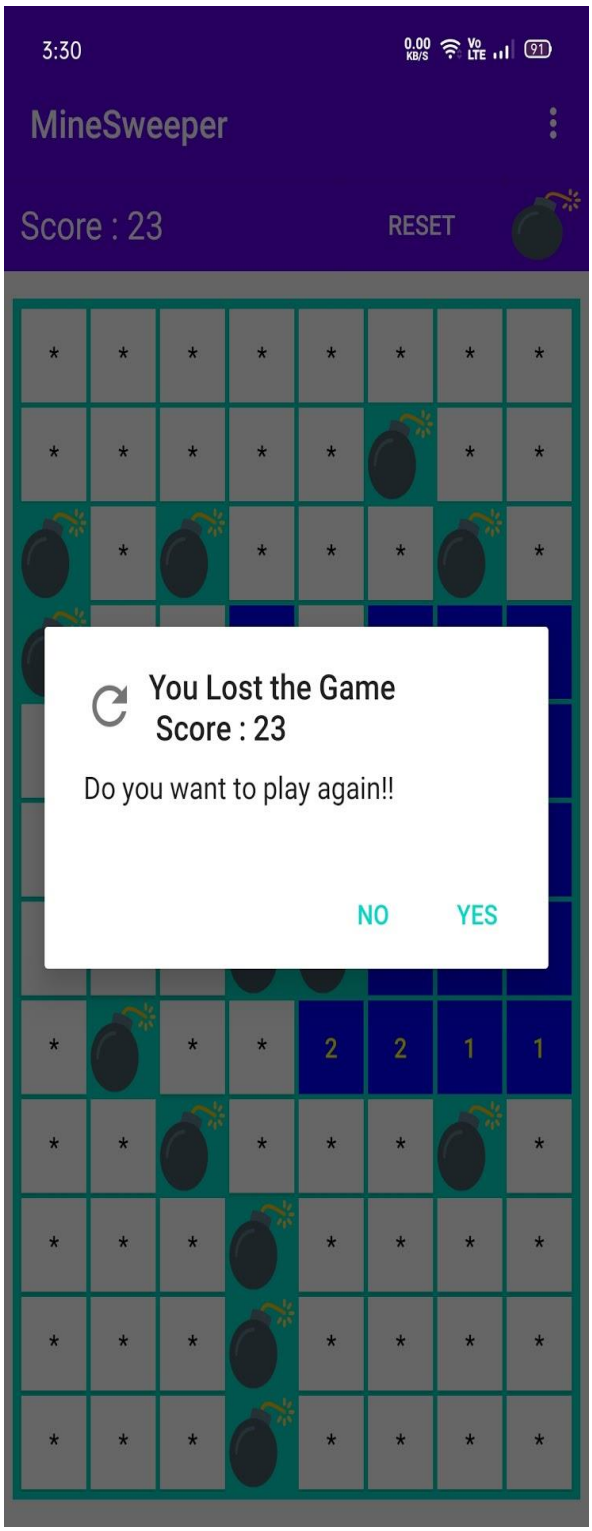


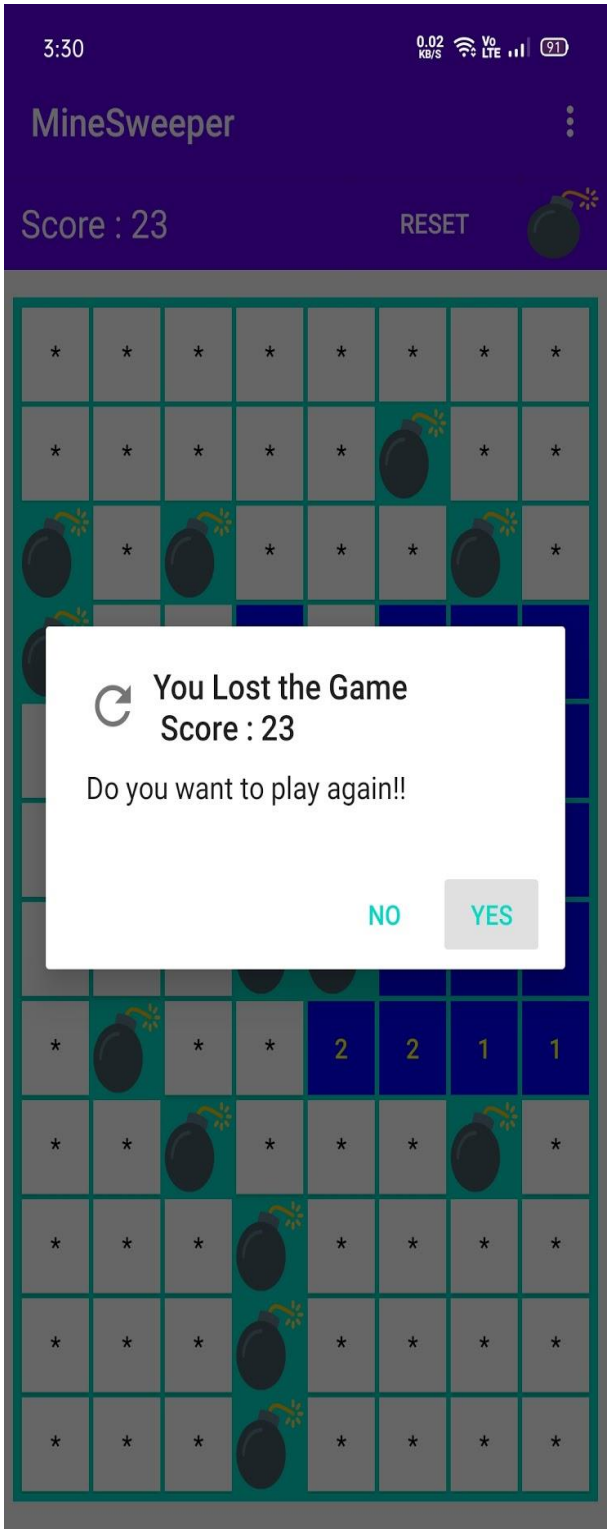


In first screenshot score is 14 which is the sum of numbers on the squares i.e. score=1+1+2+1+0+1+1+0+1+1+1+1+1=14

i.e. sore :1 if 0
1 if 1
2 if 2
3 if 3 and so on….

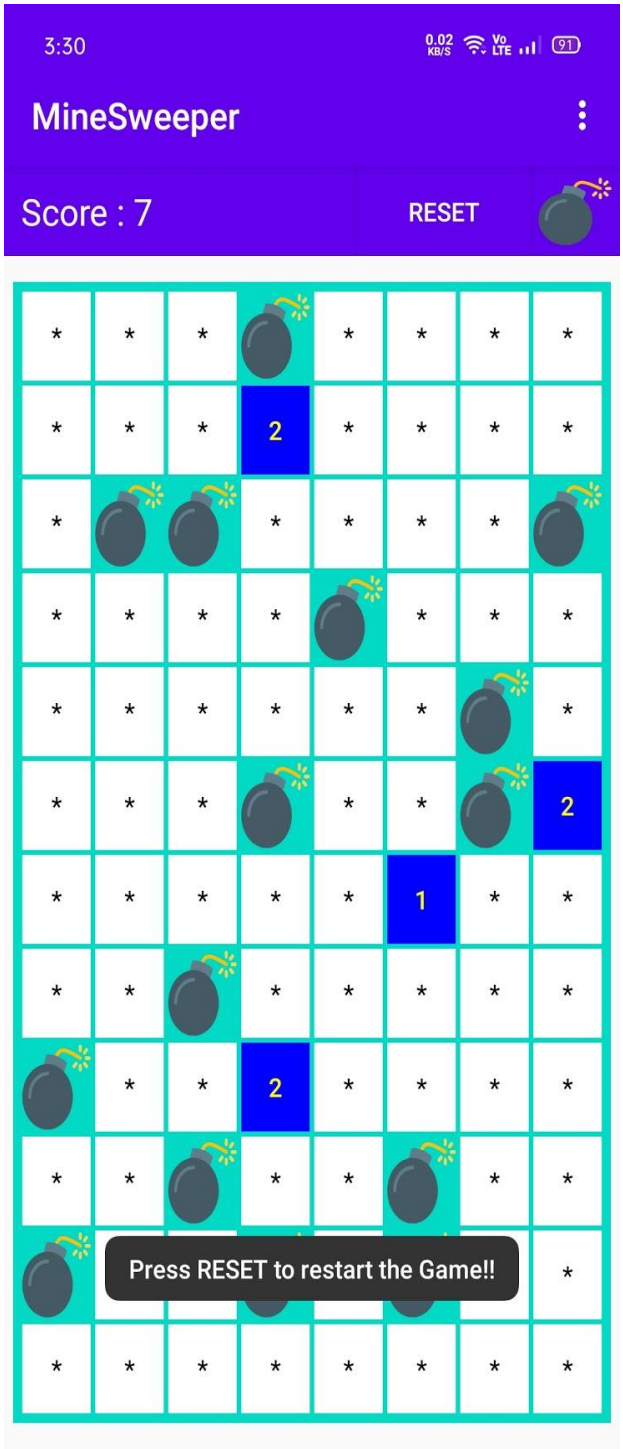
On clicking box it will show the no of mines around the box if its not a mine.
If 1,2 and 0,3 will come that it will add into the score. And if you guess that there is a mine then plug a flag there and refuse the mine.

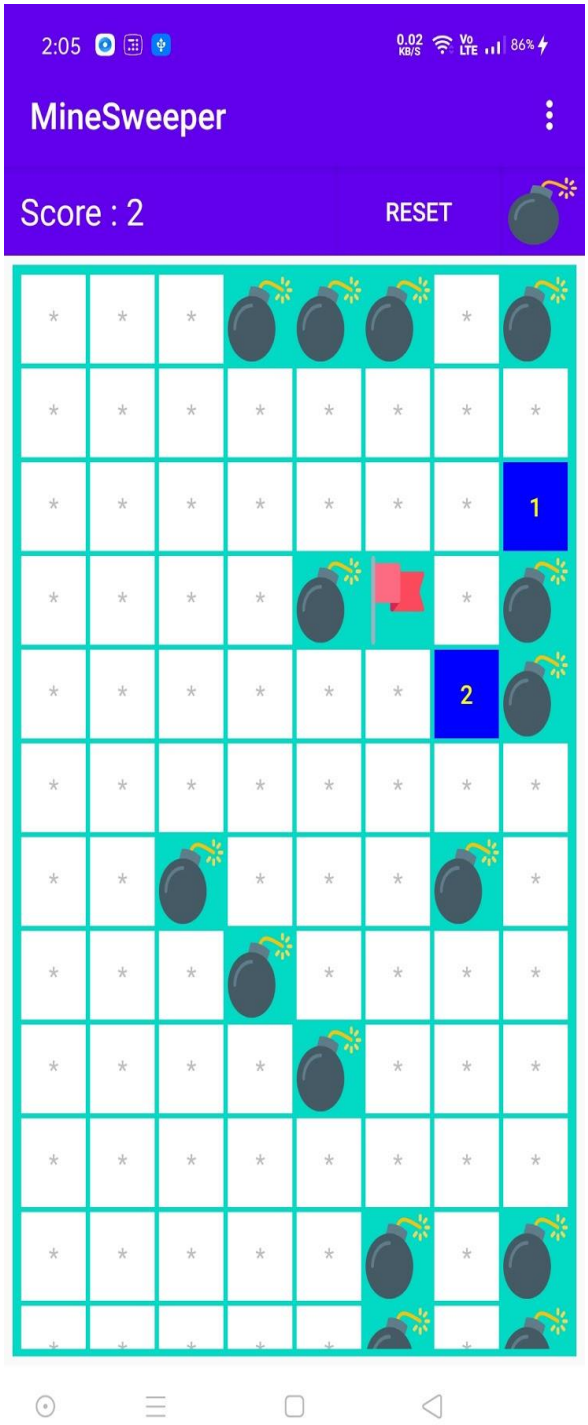




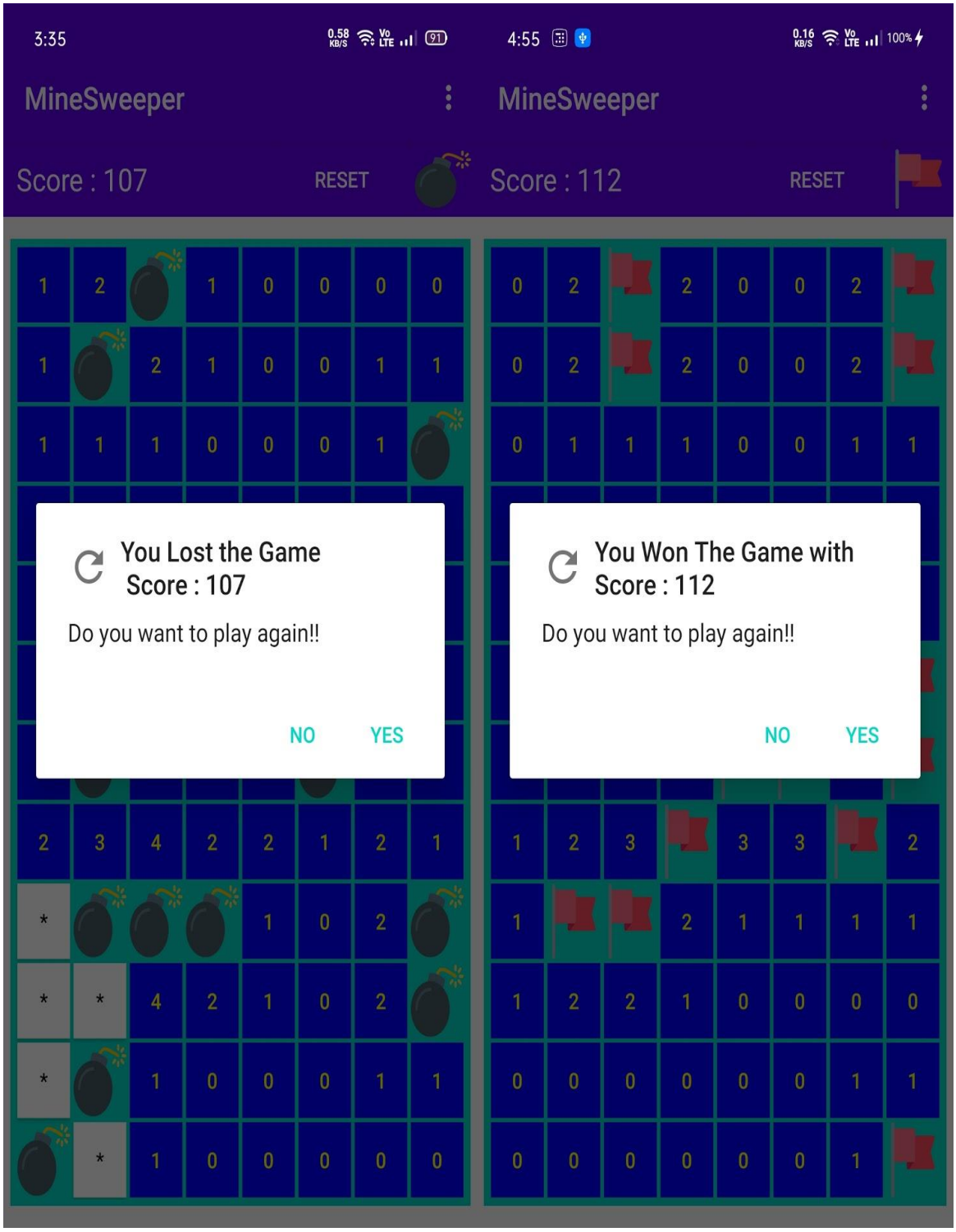
]

If you unfortunately click on"MINES" then the game is over the score will come towards you and you have given the option to continue the game or not.If you click on "YES" then it will restart otherwise it will exit.





If you click on “mines” then you will lose that means game is over you have to start again from zero. To restart the game you have to click on “RESET” button on clicking “RESET” button the game will start again.



On clicking boxes the will open and the number will add to your score if you open every box without clicking on mines then you will score maximum score.

4 Conclusion and Future work

This app will help people to reduce their stress and feel fresh and relax and increase their IQ.. To play this game we have to concentrate on game which increase our concentration power.This game checks the user's ability.

During the process of making this app we learn many new things which will help us in future .Now we have basic idea how to make app's. This game is basic to any person(child, man, woman, old person) can play game easily.

During the process of creating the game, there were several obstacles that were encountered. For example, there was difficulty in developing a manner in which all of the surrounding locations of a selected location could be checked for mines. This problem was solved by making an if-else selection structure that could check the values of the surrounding locations in the array that represented the game board.

Another difficulty that was encountered was developing a method to win the game because the mines were randomly generated. Once the sum is reached, the user has cleared the minefield and won the game.

In future this App can be update by **using database** for the user and login to the game using email-id to excess ones account in in all devices. And **high score** can also be added. The **difficulty level** can also be added by **increasing the number of bombs** and **varying the size of the board.**

In the future, with greater knowledge about programming in JAVA, we look to overcome these minor drawbacks and develop an app that would help users to win the game easily and give hints to don't select mines or refuse mines and score maximum. And also include the option to show the previous high score and make the game multilevel that make game more interesting.

ThankYou