



**Sarath Vadakkepat**

**Vigneshwar Shankar**

Sudoku Application– Includes Smart Assist and Solver

Acknowledgement

I would sincerely like to thank my parents and friends who encouraged meand helped me to complete this project successfully.

I sincerely express my heartfelt gratitude to Sujith Vadakkepat, Prahlad Suresh, Prachet Verma, Aditya Gautam, Vigneshwar Shankar and everyone for their encouragement and support.

Index

1. Preface…………………………………………………………………………………………………..4
2. Mathematics o f Sudoku……………..…………………………………………………………5
3. Application Details…………………………………………………………………………………6
4. Application Introduction………………………………………………………………………..8
5. Application Architecture………………………………………………………………………..9
6. Game Engine………………………………………………………………………………………..10
7. NumBoard…………………………………………………………………………………………..13
8. Smart Assist…………………………………………………………………………………………14
9. New Game……………………………………………………………………………………………16
10. Solver…………………………………………………………………………………………………..21
11. Credits……………………………………………………………………………………………..….25
12. Exit………………………………………………………………………………………………………26
13. Appendix Codes…………………………………………………………………………………..27
14. Appendix Graphics……………………………………………………………………………...64
15. Appendix Screen Shots………………………………………………………………………..69
16. References……………………………………………………………………………………….….78
17. Developers………………………………………………………………………………………..…79

Preface

Sudoku is one of the most famous game played by billions of people across the world. Sudoku is a logic based, combinational, number placed puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid (also called "boxes", "blocks", "regions", or "sub-squares") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which typically has a unique solution.

The puzzle was popularized in 1986 by the Japanese puzzle company Nikoli, under the name Sudoku, meaning *single number*. It became an international hit in 2005. Although the 9×9 grid with 3×3 regions is by far the most common, many variations exist. Sample puzzles can be 4×4 grids with 2×2 regions; 5×5 grids with pentomino regions have been published under the name *Logi-5*

Android is one of the widely used mobile operating system currently. The love for Sudoku will make the enthusiasts find a medium in which they can play it as they wish. Hence an android application for playing Sudoku will be an ideal choice.

Mathematics of Sudoku

A completed Sudoku grid is a special type of Latin square with the additional property of no repeated values in any of the 9 blocks of contiguous 3×3 cells. The relationship between the two theories is now completely known, after it was proven that a first-order formula that does not mention blocks is valid for Sudoku if and only if it is valid for Latin Squares.

The number of classic 9×9 Sudoku solution grids is 6,670,903,752,021,072,936,960 (sequence A107739 in OEIS), or approximately 6.67×1021. This is roughly 1.2×10−6 times the number of 9×9 Latin squares. Various other grid sizes have also been enumerated—see the main article for details. The number of essentially different solutions, when symmetries such as rotation, reflection, permutation and relabelling are taken into account, was shown to be just 5,472,730,538 (sequence A109741 in OEIS).

The maximum number of givens provided while still not rendering a unique solution is four short of a full grid (77); if two instances of two numbers each are missing from cells which occupy the corners of an orthogonal rectangle, and exactly two of these cells are within one region, there are two ways the numbers can be assigned. Since this applies to Latin squares in general, most variants of Sudoku have the same maximum. The inverse problem—the fewest givens that render a solution unique—was recently proven to be 17. A number of valid puzzles with 17 givens have been found for the standard variation without a symmetry constraint, by Japanese puzzle enthusiasts, and 18 with the givens in rotationally symmetric cells. Over 48,000 examples of Sudoku puzzles with 17 givens resulting in a unique solution are known.

In 2010 mathematicians of the University of Southern California showed that the arrangement of numbers in Sudoku puzzles have greater Shannon entropy than the number arrangements in randomly generated 9×9 matrices. This is because the rules of Sudoku exclude some random arrangements that have an innate symmetry.

APPLICATION DETAILS

Name : Sudoku

Platform : Android

APK name : Sudoku\_SV17

: Sudoku\_SV17v3

: Sudoku\_SV17v47

Versions Supported : v2.3 upwards

Screen Sizes : 3.4” to 4” – Sudoku\_SV17

: 3.2” to 3.4” – Sudoku\_SV17v3

: 4.7” - Sudoku\_SV17v47

Developed in : Eclipse Galileo with ADT plugin

Key Features : Solver

: Smart Assist

Devices Tested on : HTC Desire S

: HTC Wildfire S

: Samsung Galaxy Ace

: Sony Ericsson Xperia Live with Walkman

Developer (Code) : Sarath Vadakkepat

Developer (Graphics) : Vigneshwar Shankar

Documentation : Yes

Size (APK) : 893 KB (0.872 MB)

Sensors Used : Touch Screen

Permissions : None other than default

Build SDK : 10

Targer SDK : 10 and above

ScreenShots : Yes

Size when installed : 1.07 MB

Download Links :

:

:

Application Introduction



Android is one of the most widely used mobile operating system along with Windows Phone 7 and the iOS. More than 50% of the smart phones present in the world currently use Android OS. Sudoku is a game played by millions of people. Sudoku games are available for play in newspapers and there are many websites online which facilitate the players to play online.

Hence, it is ideal and convienient for people if they have a Sudoku application installed in their phone. Thus the users can play the game whenever they wish to and also overcomes problems that exist due to dependence on the newspapers and websites. This application was developed by keeping in mind the general interest of the users and ease. Also, to assist the user there is an option of “Smart Assist” available. The user can also choose to get the solution of a Sudoku table by using the “Solver”.

The application initially was made just to have a Sudoku solver but in due course of time, playing option was added to it. The graphics was custom designed for this and this application is expected to be released as a free of cost application to all users.

The key features and the novelty of this application includes

1. Sudoku Solver
2. Smart Assist
3. Custom NumBoard
4. Customised and Efficient Game Engine

APPLICATION ARCHITECTURE

Begin

Splash screen

Menu

Exit

Credits

Solver

New Game

Level Selector

Get Input Table

Choose Name

Show Details

Data from Files

Pass Data to solving functions

Game Engine

Run the Solver

Display

Display Solution

Smart Assist

Entry Validation

NumBoard

User Entry

Pause

Done

Check

Fig. 1

Finish

GAME ENGINE

The Sudoku puzzles that are used for playing are generated by the game engine after manipulation of data samples. The data samples for each difficulty level i.e. Easy, Medium and Hard and stored in text files. The game engine selects one of the puzzles and sends it through the manipulation architecture that generates another version of the same. The game engine is extremely optimized and highly efficient coding dynamics put to test. The game engine was completely conceptualized and designed by myself. The code was given much iteration to completely optimize its functionality and check for accuracy.

*Algorithm*

1. Pick one data from the sample space according to the level of difficulty as chosen by the user.
2. Initiate random processes to determine the level of randomizations needed.
3. Send data through a series of manipulation according to the aforesaid random process and also the random process of each level.
4. Data shifted by rows and colums and orientations changed depending on the random process that is acting on that part.
5. Data then comes to a permuted form after passing through 6-step random process.
6. Output the final puzzle as a string output of 81 characters.

*Flowchart*

Hard

Medium

Easy

One data chosen

Random process determining levels

Level 4

Level 3

Level 2

Level 1

Random Process 4

Random Process 3

Random Process 2

Random process 1

Intermediate output

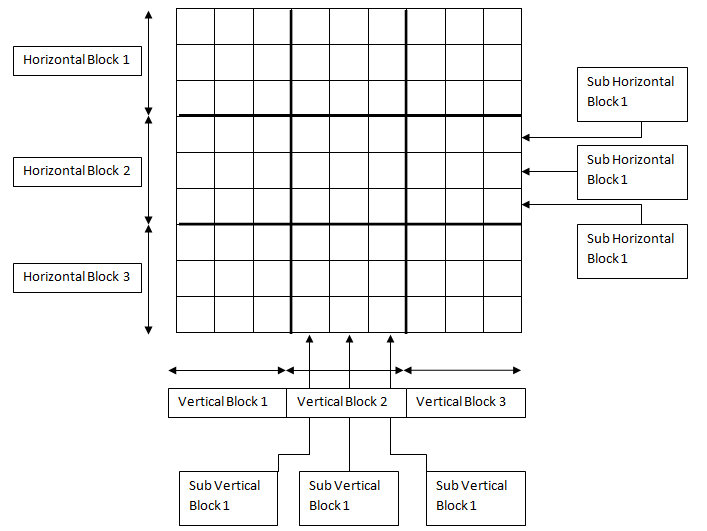
Level 5 – Usage restricted

Output

Fig. 2

The different types of shuffling include

1. Level 1



1. Normal
2. Clockwise Rotation
3. Clockwise Rotation twice
4. Anti-Clockwise Rotation
5. Level 2
6. Level 1
7. Mirrored Horizontal
8. Mirrored Vertical

3. Level 3

Fig. 2(a)

1. Level 2

2. Shift Vertical Block 1, 2, 3 and Horizontal Block 1, 2, 3 in

one of 36 orientations.

4. Level 4

1. Level 3

2. Mirrored Vertical

3. Mirrored Horizontal

4. Supercharged level, Sub Vertical Row 1,2,3 and Sub Horizontal

Column oriented in one of 36 ways.

NumBoard



The interface that facilitates user entry is not

based on the default keyboard provided by the

android operating system. For the application a

new keyboard was designed. Titled “NumBoard”

prominently is displayed as a pop-up when the

user has to enter any number rom 1-9 and it also

features erase option also.

*Algorithm*

1. Listen for clicks on cells
2. Verify is Smart Assist is turned

Fig. 3

1. Display the NumBoard appropriately.
2. Dismiss on user press

*FlowChart*

Dismiss

Display NumBoard

Status of Smart assist

Listen for Clicks on Cells

Fig. 3(a)

Smart Assist

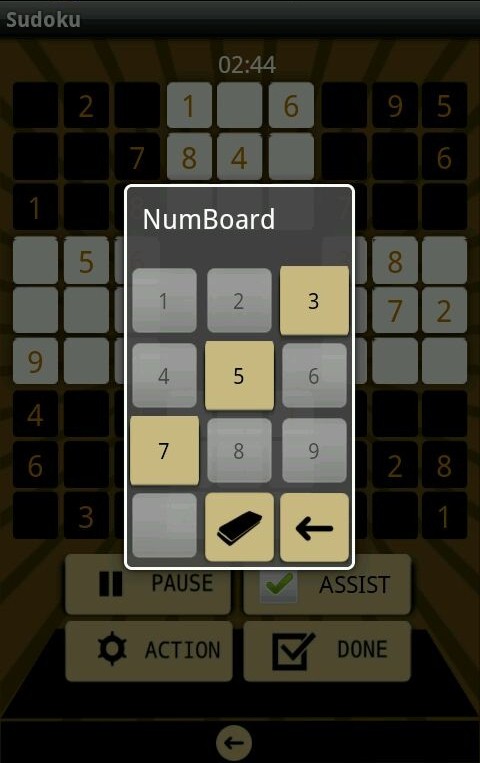
The feature of Smart Assist included in this application is one of the novelty achieved in this application. This feature is aimed in providing help to the users while the user is unable to proceed the game.

The idea of this feature is to simply tell the user which of the numbers are possible to be inserted in a cell depending on the dynamic situations present in the game. This feature can be turned on my checking the “Assist” option available during gameplay. When assist is turned off, the clicking on cell would project the whole NumBoard consisting of all the number from 1-9, but on turning the assist on, clicking on any cell would display the numbers possible to that particular cell thereby disabling the other buttons.

The screen shots depicting this feature are shown below.

With Assist

Without Assist



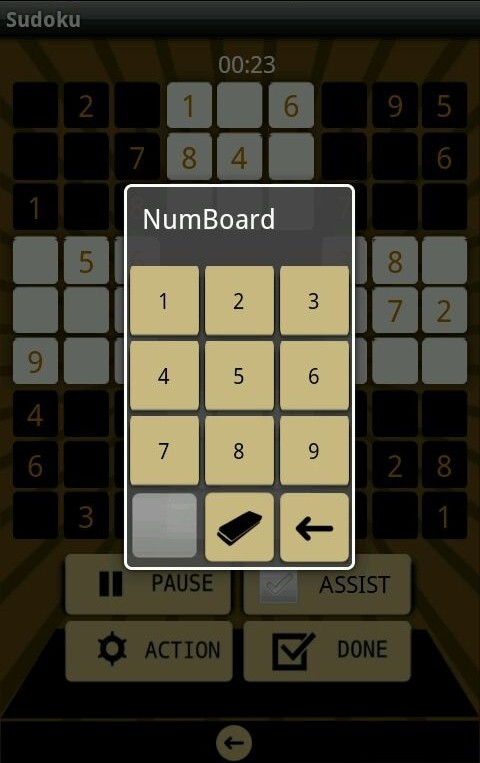


Fig. 4(a)

Fig. 4

*Algorithm*

1. Listen for clicks on cells
2. Status of the “Assist” option
3. If assist it activated, function for checking row, column and cube is executed.
4. Display Set X of numbers where Set X is subset of {1,2,3,4,5,6,7,8,9}
5. Displays the NumBoard with Set X.

*Flowchart*

On

Set X = Subset of {1, 2, 3, 4, 5, 6, 7, 8, 9}

Status of Assist Option

Check Row, Column, Cube

Listen for Clicks

Off

Display NumBoard with Set X

Set X = {1, 2, 3, 4, 5, 6, 7, 8, 9}

Fig. 4(b)



The New game option enables the user to start a new game. The user is then given a option to choose the level of hardness i.e. Easy, Medium, Hard.

Once the user has clicked on his selection, the game engine starts and thus prepares the necessary table for the gameplay.

The layout of the new game is shown in the adjacent



figure. When the game is running the table will be

filled with the values to form a Sudoku game and the

timer is started to keep the track of time.

The player also is entitled to the freedom of “Assist”

option when is faced with difficulty while playing.

The different features of the New Game are:

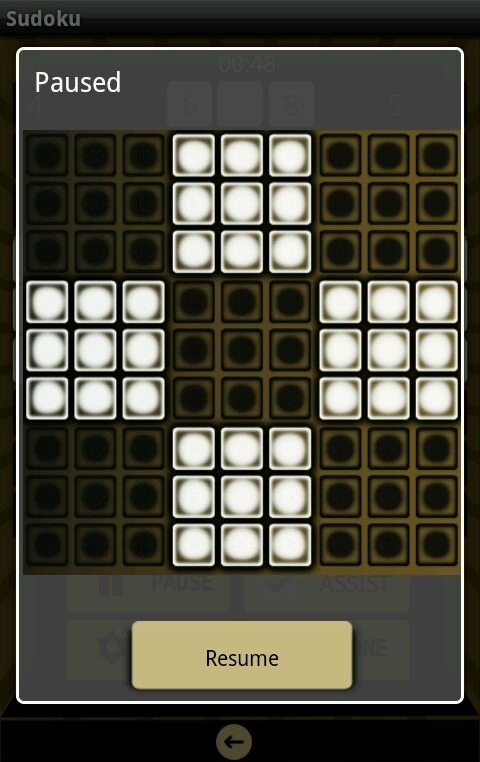
Fig. 5

1. Pause
2. Actions
3. Restart Game
4. New Game
5. Check
6. Quit
7. Done
8. Assist

The various actions’ explanation are explained below

1. *Pause*

The pause button is used to pause the game while it is being played. On pressing the pause button the timer is stopped and a filler screen is shown which hides the content of the table till the user presses the resume button to resume the game.



*Algorithm:*

1. Stop the timer.
2. Display Filler Screen
3. Listen for Click on resume button
4. On- Resume, dismiss popup
5. Continue Timer.

Fig. 5(a)

*Flowchart*

Listen for on Resume

Display Filler Screen

Stop Timer

On Click Pause

Yes

Continue Timer

Dismiss Pop-up

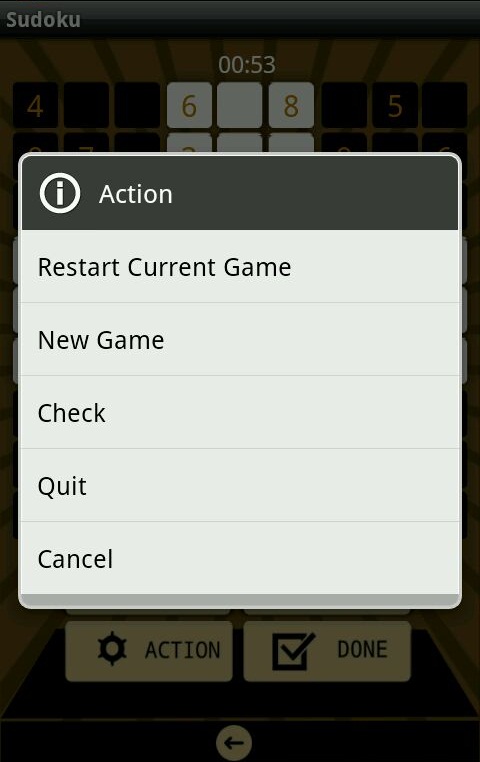
Fig. 5(b)

1. *Actions*

The Actions button provides the user with many more options that helps the user take control of the application. On clicking the button a context menu opens with various options. The options listed in the context menu are

1. Restart Current Game
2. Start New Game
3. Check
4. Quit

Fig. 5(c)



*Algorithm: For Restart, Start and Quit*

1. Listen for Clicks
2. Ask Confirmation
3. If yes, redirect to respective layout.

*Flowchart*

Restart

Redirect to Layout

Ask Confimation

On Click

Dismiss

Quit

New game

Fig. 5(d)

*Algorithm: Check*

1. Listen for Clicks
2. Check for repetition of a number in its

respective row/column/cube.

1. If no mistake, display the right message
2. Else Display message with number of errors

*Flowchart*

Check Colum, Any error?

Check Cube, Any error?

On Click

Check Row, Any error?

Display Respective msg

Total Errors

Fig. 5(e)

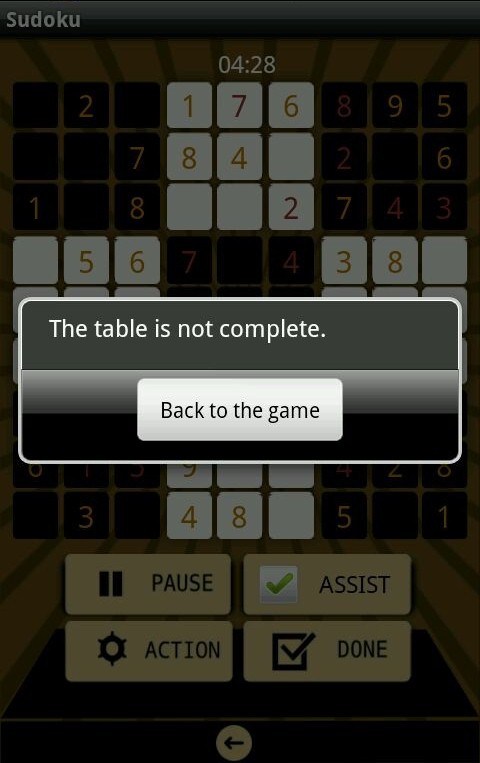
1. *Done*

“Done gives the user the freedom to submit the solution to check if the solution is correct or not. There are 3 possible outcomes on pressing the “done” button. They are :

1. “The table is not complete” – When there are unfilled cells.
2. “The solution is wrong” - When the solution is wrong.

The above 2 notifications are shown as pop-up.

1. On completing the table successfully and



Correctly, the user gets a “trophy” also

Indicating the time taken.

*Algorithm*

1. Listen for clicks
2. Call the checking function which check the

possibility of appearance of a number in a cell.

1. Display appropriate message

Fig. 5(f)



*Flowchart*

Calling required function

Checking Row, Colum, Cube

On Click

On Click

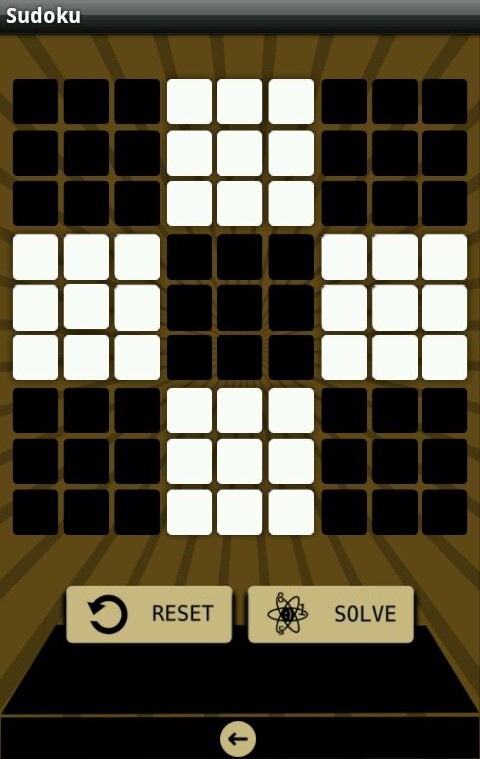
Fig. 5(g)

Fig. 5(h)



The app gives the user an unique feature that enables the player to get the solution of any Sudoku table by entering the values of the corresponding cell.

The user once after selecting the solver option is shown an empty table and the user is free to enter any values into the table. The smart entry feature ensures that only the values accepted to a particular cell which is calculated on a dynamic basis is only taken as input so as to completely reduce the error that can occur during input.



The page layout of the solver is shown alongside

The user is now free to enter any value in any cell.

clicking on any cell would then display the pop up

of keyboard. The numbers are thus available for entry.

Fig. 6



The values entered by the user

****

The solution to the Sudoku table

displayed after solving.

Fig. 6(b)

Fig. 6(a)

*Algorithm*

1. Take input values from user.
2. Validate the user input (with help of algorithm of smart assist)
3. Pass data to the function for finding the solution.

(The algorithm for solving is by Brute-force method using recursion and back tracking strategy).

1. Display the final table after solving.

*Flowchart*

Display

Solver function

User input

Validation of User input

Fig. 6(c)

Solving algorithm backtracking and recursion.

The solver also consists of options of “Reset” and “Solve”

1. Reset – Enables the user to clear the table for obtaining solution

to another Sudoku table.

1. Solve – Enables the user to obtain the solution of the Sudoku

table entered. Solve button functions as a signal/cue

to the application to initiate solving procedure and

display result.



On clicking on the Credits button, the control now shift to the credits layout page which displays the details of the key people involved in this project. The details are widely classified under *Code* and *Design*. On clicking the appropriate names a pop-up is created to provide provision of contact to any of the people involved.



The Layout of the “Credits” page is shown alongside.

The names of the people involved are displayed as

a ‘button’ at the UI level.

Thus enabling the properties of ‘button’ field of

android, the user is now free to click on the names

and thus it displays a ‘pop-up’ with the details. The

‘pop-up’ thus displayed can be dismissed by clicking

on the ‘pop-up’.

Fig. 7







Fig. 7(c)

Fig. 7(b)

Fig. 7(a)



The last option listed on the menu is the “Exit” option. This option is used to close all instances of the app and kill objects used by the program and free the memory and close the application.

This ensures that the application does not run any daemon process and all the threads and processes are killed and no memory is held. This ends the life cycle of the application.

*APPENDIX*

CODES

***Codes***

***1. Main Activity***

**package** com.sarath.sv17;

**import** android.app.Activity;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.os.Handler;

**public** **class** Sudoku\_SV17Activity **extends** Activity {

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

**new** Handler().postDelayed(**new** Runnable() {

@Override

**public** **void** run() {

**final** Intent mainIntent = **new** Intent(Sudoku\_SV17Activity.**this**, Bg2.**class**);

Sudoku\_SV17Activity.**this**.startActivity(mainIntent);

Sudoku\_SV17Activity.**this**.finish();

}

}, 5000);

}

}

***2. Menu***

**package** com.sarath.sv17;

**import** android.app.Activity;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**public** **class** Bg2 **extends** Activity

{

@Override

**public** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*bgmenu*);

Button solver=**null**;

Button new\_game=**null**;

Button exit=**null**;

Button credits=**null**;

solver=(Button)findViewById(R.id.*button2*);

solver.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{ Intent myIntent = **new** Intent(Bg2.**this**, Sudoku\_solver.**class**);

startActivity(myIntent);

}

});

new\_game=(Button)findViewById(R.id.*button1*);

new\_game.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{ Intent myIntent = **new** Intent(Bg2.**this**, GameEngine.**class**);

startActivity(myIntent);

}

});

exit=(Button)findViewById(R.id.*button4*);

exit.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{ System.*exit*(0);

}

});

credits=(Button)findViewById(R.id.*button3*);

credits.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{ Intent myIntent = **new** Intent(Bg2.**this**, Credit.**class**);

startActivity(myIntent);

}

});

}

}

***3. Game Engine***

**package** com.sarath.sv17;

**import** android.app.Activity;

**import** android.app.Dialog;

**import** android.app.ProgressDialog;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**import** java.util.Random;

**import** java.io.\*;

**public** **class** GameEngine **extends** Activity

{

String x;

**int** i,j;

**int** array[][]=**new** **int**[9][9];

**int** ctr=0;

@Override

**public** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

toughness();

}

**public** **void** randomise()

{

ProgressDialog dialog = ProgressDialog.*show*(GameEngine.**this**, "", "Loading.", **true**);

dialog.show();

**int** n1=random\_func(4);

**if**(n1==1) random\_level1();

**else** **if**(n1==2)

{ random\_level1();

random\_level2(); }

**else** **if**(n1==3)

{ random\_level1();

random\_level2();

random\_level3(); }

**else** **if**(n1==4)

{ random\_level1();

random\_level2();

random\_level3();

random\_level4(); }

get\_string();

Intent intent =**new** Intent(GameEngine.**this**, Gameplay.**class**);

intent.putExtra("question", x);

startActivity(intent);

finish();

}

**public** **void** toughness()

{ **final** Context mContext = **this**;

**final** Dialog dialog = **new** Dialog(mContext);

dialog.setContentView(R.layout.*toughness*);

dialog.setTitle("Level");

dialog.show();

Button Easy=(Button) dialog.findViewById(R.id.*buttoneasy*);

Button Medium=(Button) dialog.findViewById(R.id.*buttonmedium*);

Button Hard=(Button) dialog.findViewById(R.id.*buttonhard*);

Easy.setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ GetEasy();

dialog.dismiss();

randomise();

}

});

Medium.setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ GetMedium();

randomise();

dialog.dismiss();

}

});

Hard.setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ GetHard();

randomise();

dialog.dismiss();

}

});

**return**;

}

**public** **void** GetEasy()

{ **int** easy1=random\_func(80);

**int** counter=0;

**try** {

counter=0;

String str="";

InputStream is = **this**.getResources().openRawResource(R.drawable.*data\_easy*);

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is));

**while** ((str = reader.readLine()) != **null**)

{

counter++;

**if**(counter==easy1)

{ x=str;

**break**;

}

}

}

**catch** (IOException e) { }

**return**;

}

**public** **void** GetMedium()

{ **int** medium1=random\_func(80);

**int** counter=0;

**try** {

counter=0;

String str="";

InputStream is = **this**.getResources().openRawResource(R.drawable.*data\_medium*);

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is));

**while** ((str = reader.readLine()) != **null**)

{ counter++;

**if**(counter==medium1)

{x=str;

**break**;

}

}

}

**catch** (IOException e) { }

**return**;

}

**public** **void** GetHard()

{ **int** hard1=random\_func(80);

**int** counter=0;

**try** {

counter=0;

String str="";

InputStream is = **this**.getResources().openRawResource(R.drawable.*data\_hard*);

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is));

**while** ((str = reader.readLine()) != **null**)

{ counter++;

**if**(counter==hard1)

{ x=str;

**break**;

}

}

}

**catch** (IOException e) { }

**return**;

}

**public** **void** random\_level1()

{ **int** n2=random\_func(4);

**if**(n2==1) north();

**else** **if**(n2==2) south();

**else** **if**(n2==3) east();

**else** **if**(n2==4) west();

**return**;

}

**public** **void** random\_level2()

{ **int** n2=random\_func(3);

**if**(n2==1) mirror\_vertical();

**else** **if**(n2==2) mirror\_horizontal();

**else** **if**(n2==3) {}

**return**;

}

**public** **void** random\_level3()

{ **int** n2=random\_func(36);

**switch**(n2)

{ **case** 1: **break**;

**case** 2: shift\_v(1,3,2);

**break**;

**case** 3: shift\_v(2,1,3);

**break**;

**case** 4: shift\_v(2,3,1);

**break**;

**case** 5: shift\_v(3,1,2);

**break**;

**case** 6: shift\_v(2,3,1);

**break**;

**case** 7: shift\_h(1,3,2);

**break**;

**case** 8: shift\_h(1,3,2);

shift\_v(1,3,2);

**break**;

**case** 9: shift\_h(1,3,2);

shift\_v(2,1,3);

**break**;

**case** 10: shift\_h(1,3,2);

shift\_v(2,3,1);

**break**;

**case** 11: shift\_h(1,3,2);

shift\_v(3,1,2);

**break**;

**case** 12: shift\_h(1,3,2);

shift\_v(3,2,1);

**break**;

**case** 13: shift\_h(2,1,3);

**break**;

**case** 14: shift\_h(2,1,3);

shift\_v(1,3,2);

**break**;

**case** 15: shift\_h(2,1,3);

shift\_v(2,1,3);

**break**;

**case** 16: shift\_h(2,1,3);

shift\_v(2,3,1);

**break**;

**case** 17: shift\_h(2,1,3);

shift\_v(3,1,2);

**break**;

**case** 18: shift\_h(2,1,3);

shift\_v(3,2,1);

**break**;

**case** 19: shift\_h(2,3,1);

**break**;

**case** 20: shift\_h(2,3,1);

shift\_v(1,3,2);

**break**;

**case** 21: shift\_h(2,3,1);

shift\_v(2,1,3);

**break**;

**case** 22: shift\_h(2,3,1);

shift\_v(2,3,1);

**break**;

**case** 23: shift\_h(2,3,1);

shift\_v(3,1,2);

**break**;

**case** 24: shift\_h(2,3,1);

shift\_v(3,2,1);

**break**;

**case** 25: shift\_h(3,1,2);

**break**;

**case** 26: shift\_h(3,1,2);

shift\_v(1,3,2);

**break**;

**case** 27: shift\_h(3,1,2);

shift\_v(2,1,3);

**break**;

**case** 28: shift\_h(3,1,2);

shift\_v(2,3,1);

**break**;

**case** 29: shift\_h(3,1,2);

shift\_v(3,1,2);

**break**;

**case** 30: shift\_h(3,1,2);

shift\_v(3,2,1);

**break**;

**case** 31: shift\_h(3,2,1);

**break**;

**case** 32: shift\_h(3,2,1);

shift\_v(1,3,2);

**break**;

**case** 33: shift\_h(3,2,1);

shift\_v(2,1,3);

**break**;

**case** 34: shift\_h(3,2,1);

shift\_v(2,3,1);

**break**;

**case** 35: shift\_h(3,2,1);

shift\_v(3,1,2);

**break**;

**case** 36: shift\_h(3,2,1);

shift\_v(3,2,1);

**break**;

}

**return**;

}

**public** **void** shift\_h(**int** a, **int** b, **int** c)

{ east();

shift\_v(a,b,c);

west();

**return**;

}

**public** **void** shift\_hm(**int** a, **int** b, **int** c)

{ east();

shift\_vm(a,b,c);

west();

**return**;

}

**public** **void** random\_level4()

{

**int** n2=random\_func(3);

**if**(n2==1) mirror\_vertical();

**else** **if**(n2==2) mirror\_horizontal();

**else** **if**(n2==3)

{

n2=random\_func(2);

**if**(n2==2) supercharged();

}

**return**;

}

**public** **void** supercharged()

{ **int** n2=random\_func(36);

**switch**(n2)

{

**case** 1: **break**;

**case** 2: shift\_vm(1,3,2);

**break**;

**case** 3: shift\_vm(2,1,3);

**break**;

**case** 4: shift\_vm(2,3,1);

**break**;

**case** 5: shift\_vm(3,1,2);

**break**;

**case** 6: shift\_vm(2,3,1);

**break**;

**case** 7: shift\_hm(1,3,2);

**break**;

**case** 8: shift\_hm(1,3,2);

shift\_vm(1,3,2);

**break**;

**case** 9: shift\_hm(1,3,2);

shift\_vm(2,1,3);

**break**;

**case** 10: shift\_hm(1,3,2);

shift\_vm(2,3,1);

**break**;

**case** 11: shift\_hm(1,3,2);

shift\_vm(3,1,2);

**break**;

**case** 12: shift\_hm(1,3,2);

shift\_vm(3,2,1);

**break**;

**case** 13: shift\_hm(2,1,3);

**break**;

**case** 14: shift\_hm(2,1,3);

shift\_vm(1,3,2);

**break**;

**case** 15: shift\_hm(2,1,3);

shift\_vm(2,1,3);

**break**;

**case** 16: shift\_hm(2,1,3);

shift\_vm(2,3,1);

**break**;

**case** 17: shift\_hm(2,1,3);

shift\_vm(3,1,2);

**break**;

**case** 18: shift\_hm(2,1,3);

shift\_vm(3,2,1);

**break**;

**case** 19: shift\_hm(2,3,1);

**break**;

**case** 20: shift\_hm(2,3,1);

shift\_vm(1,3,2);

**break**;

**case** 21: shift\_hm(2,3,1);

shift\_vm(2,1,3);

**break**;

**case** 22: shift\_hm(2,3,1);

shift\_vm(2,3,1);

**break**;

**case** 23: shift\_hm(2,3,1);

shift\_vm(3,1,2);

**break**;

**case** 24: shift\_hm(2,3,1);

shift\_vm(3,2,1);

**break**;

**case** 25: shift\_hm(3,1,2);

**break**;

**case** 26: shift\_hm(3,1,2);

shift\_vm(1,3,2);

**break**;

**case** 27: shift\_hm(3,1,2);

shift\_vm(2,1,3);

**break**;

**case** 28: shift\_hm(3,1,2);

shift\_vm(2,3,1);

**break**;

**case** 29: shift\_hm(3,1,2);

shift\_vm(3,1,2);

**break**;

**case** 30: shift\_hm(3,1,2);

shift\_vm(3,2,1);

**break**;

**case** 31: shift\_hm(3,2,1);

**break**;

**case** 32: shift\_hm(3,2,1);

shift\_vm(1,3,2);

**break**;

**case** 33: shift\_hm(3,2,1);

shift\_vm(2,1,3);

**break**;

**case** 34: shift\_hm(3,2,1);

shift\_vm(2,3,1);

**break**;

**case** 35: shift\_hm(3,2,1);

shift\_vm(3,1,2);

**break**;

**case** 36: shift\_hm(3,2,1);

shift\_vm(3,2,1);

**break**;

}

**return**;

}

**public** **int** random\_func(**int** n)

{ Random rand=**new** Random();

**int** num=rand.nextInt(n);

**return** num+1;

}

**public** **void** north()

{ ctr=0;

**for**(i=0;i<9;i++)

**for**(j=0;j<9;j++)

array[i][j]=Integer.*parseInt*(String.*valueOf*(x.charAt(ctr++)));

get\_string();

**return**;

}

**public** **void** south()

{ east();

east();

get\_string();

**return**;

}

**public** **void** west()

{ ctr=0;

**for**(i=0;i<9;i++)

**for**(j=8;j>=0;j--)

array[j][i]=Integer.*parseInt*(String.*valueOf*(x.charAt(ctr++)));

get\_string();

**return**; }

**public** **void** east()

{ ctr=0;

**for**(i=8;i>=0;i--)

**for**(j=0;j<9;j++)

array[j][i]=Integer.*parseInt*(String.*valueOf*(x.charAt(ctr++)));

get\_string();

**return**;

}

**public** **void** get\_string()

{ x="";

**for**(i=0;i<9;i++)

**for**(j=0;j<9;j++)

x=x+String.*valueOf*(array[i][j]);

**return**;

}

**public** **void** mirror\_vertical()

{ ctr=0;

**for**(i=0;i<9;i++)

**for**(j=8;j>=0;j--)

array[i][j]=Integer.*parseInt*(String.*valueOf*(x.charAt(ctr++)));

get\_string();

**return**;

}

**public** **void** mirror\_horizontal()

{ ctr=0;

**for**(i=8;i>=0;i--)

**for**(j=0;j<9;j++)

array[i][j]=Integer.*parseInt*(String.*valueOf*(x.charAt(ctr++)));

get\_string();

**return**;

}

**public** **void** shift\_v(**int** a,**int** b,**int** c)

{

String str[]=**new** String[3];

str[0]=x.substring(0,27);

str[1]=x.substring(27,54);

str[2]=x.substring(54,81);

**for**(i=0;i<3;i++)

{

**for**(**int** j=0;j<9;j++)

array[i][j]=str[a-1].charAt(i\*9+j)-48;

**for**(**int** j=0;j<9;j++)

array[i+3][j]=str[b-1].charAt(i\*9+j)-48;

**for**(**int** j=0;j<9;j++)

array[i+6][j]=str[c-1].charAt(i\*9+j)-48;

}

get\_string();

**return**;

}

**public** **void** shift\_vm(**int** a,**int** b,**int** c)

{ String str[]=**new** String[3];

str[0]=x.substring(27,36);

str[1]=x.substring(36,45);

str[2]=x.substring(45,54);

**for**(**int** j=0;j<9;j++)

array[3][j]=str[a-1].charAt(j)-48;

**for**(**int** j=0;j<9;j++)

array[4][j]=str[b-1].charAt(j)-48;

**for**(**int** j=0;j<9;j++)

array[5][j]=str[c-1].charAt(j)-48;

get\_string();

**return**;

}

***4. Game Engine – Alternate***

**package** com.sarath.sv17;

**import** android.app.Activity;

**import** android.app.Dialog;

**import** android.app.ProgressDialog;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**import** java.util.Random;

**import** java.io.\*;

**public** **class** GameEngine\_alternate **extends** Activity{

String x="";

String x1="";

@Override

**public** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

toughness();

}

**public** **void** randomise()

{

ProgressDialog dialog = ProgressDialog.*show*(GameEngine\_alternate.**this**, "", "Loading.", **true**);

dialog.show();

**int** n1=random\_func(4);

**if**(n1==1) random\_level1();

**else** **if**(n1==2) random\_level2();

**else** **if**(n1==3) random\_level3();

**else** **if**(n1==4) random\_level4();

Intent intent =**new** Intent(GameEngine\_alternate.**this**, Gameplay.**class**);

intent.putExtra("question", x);

startActivity(intent);

finish();

}

**public** **void** toughness()

{

**final** Context mContext = **this**;

**final** Dialog dialog = **new** Dialog(mContext);

dialog.setContentView(R.layout.*toughness*);

dialog.setTitle("Level");

dialog.show();

Button Easy=(Button) dialog.findViewById(R.id.*buttoneasy*);

Button Medium=(Button) dialog.findViewById(R.id.*buttonmedium*);

Button Hard=(Button) dialog.findViewById(R.id.*buttonhard*);

Easy.setOnClickListener(**new** View.OnClickListener()

{

@Override

**public** **void** onClick(View view)

{

GetEasy();

dialog.dismiss();

randomise();

}

});

Medium.setOnClickListener(**new** View.OnClickListener()

{

@Override

**public** **void** onClick(View view)

{

GetMedium();

randomise();

dialog.dismiss();

}

});

Hard.setOnClickListener(**new** View.OnClickListener()

{

@Override

**public** **void** onClick(View view)

{

GetHard();

randomise();

dialog.dismiss();

}

});

**return**;

}

**public** **void** GetEasy()

{ **int** easy1=random\_func(80);

**int** counter=0;

**try** {

counter=0;

String str="";

InputStream is = **this**.getResources().openRawResource(R.drawable.*data\_easy*);

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is));

**while** ((str = reader.readLine()) != **null**)

{ counter++;

**if**(counter==easy1)

{ x=str;

**break**;

}

}

}

**catch** (IOException e) { }

**return**;

}

**public** **void** GetMedium()

{ **int** medium1=random\_func(80);

**int** counter=0;

**try** {

counter=0;

String str="";

InputStream is = **this**.getResources().openRawResource(R.drawable.*data\_medium*);

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is));

**while** ((str = reader.readLine()) != **null**)

{ counter++;

**if**(counter==medium1)

{ x=str;

**break**;

}

}

}

**catch** (IOException e) { }

**return**;

}

**public** **void** GetHard()

{ **int** hard1=random\_func(80);

**int** counter=0;

**try** {

counter=0;

String str="";

InputStream is = **this**.getResources().openRawResource(R.drawable.*data\_hard*);

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is));

**while** ((str = reader.readLine()) != **null**)

{ counter++;

**if**(counter==hard1)

{ x=str;

**break**;

}

}

}

**catch** (IOException e) { }

**return**;

}

**public** **void** random\_level1()

{

**int** n2=random\_func(4);

**if**(n2==2) smart(8,1,0,8,1,0,1,0);

**else** **if**(n2==3) smart(0,-1,8,8,1,0,2,0);

**else** **if**(n2==4) smart(8,1,0,8,1,0,1,0);

**return**;

}

**public** **void** random\_level2()

{ random\_level1();

**int** n2=random\_func(3);

**if**(n2==1) smart(0,-1,8,8,1,0,1,0);

**else** **if**(n2==2) smart(8,1,0,0,-1,8,1,0);

**return**;

}

**public** **void** random\_level3()

{

random\_level2();

**int** n2=random\_func(36)-1;

**int** a1=(**int**)(n2/6);

**int** a2=n2-(a1\*6);

**int** array[]={1,2,3,1,3,2,2,1,3,2,3,1,3,1,2,3,2,1};

shift\_h(array[a1\*3],array[(a1\*3)+1],array[(a1\*3)+2]);

shift\_v(array[a2\*3],array[(a2\*3)+1],array[(a2\*3)+2]);

**return**;

}

**public** **void** random\_level4()

{ random\_level3();

random\_level2();

**int** n2=random\_func(2);

**if**(n2==2)

{ n2=random\_func(2);

**if**(n2==2) supercharged();

}

**return**;

}

**public** **int** random\_func(**int** n)

{

Random rand=**new** Random();

**return** (rand.nextInt(n)+1);

}

**public** **void** smart(**int** a,**int** b,**int** c,**int** a1,**int** b1,**int** c1,**int** code,**int** e)

{

**if**(e==0) x="";

**for**(**int** i=a;(-i\*b)<=c;i=i-b)

{ **for**(**int** j=a1;(-j\*b1)<=c1;j=j-b1)

{ **if**(code==1) x=x+x1.charAt((i\*9)+j);

**if**(code==2) x=x+x1.charAt((j\*9)+i);

}

}

**if**(e==0) x1=x;

}

**public** **void** shift\_v(**int** xx, **int** yy, **int** zz)

{

x="";

smart(((xx-1)\*3),-1,((xx-1)\*3)+2,0,-1,8,1,1);

smart(((yy-1)\*3),-1,((yy-1)\*3)+2,0,-1,8,1,1);

smart(((zz-1)\*3),-1,((zz-1)\*3)+2,0,-1,8,1,1);

x1=x;

}

**public** **void** shift\_h(**int** xx, **int** yy, **int** zz)

{

x="";

**int** num=0;

**while**(num<=8)

{ x=x+x1.substring((num\*9)+((xx-1)\*3),((num\*9)+((xx-1)\*3))+3)+x1.substring((num\*9)+((yy-1)\*3),((num\*9)+((yy-1)\*3))+3);

x=x+x1.substring((num\*9)+((zz-1)\*3),((num\*9)+((zz-1)\*3))+3);

num++;

}

x1=x;

}

**public** **void** shift\_vm(**int** xx,**int** yy,**int** zz)

{

x="";

x=x+x1.substring(0,27);

smart(xx+2,-1,xx+2,0,-1,8,1,1);

smart(yy+2,-1,yy+2,0,-1,8,1,1);

smart(zz+2,-1,zz+2,0,-1,8,1,1);

x=x+x1.substring(54,81);

x1=x;

}

**public** **void** shift\_hm(**int** xx, **int** yy, **int** zz)

{

x="";

**int** num=0;

**while**(num<=8)

{

x=x+x1.substring((num\*9),(num\*9)+3);

x=x+x1.charAt((num\*9)+xx+2)+x1.charAt((num\*9)+yy+2)+x1.charAt((num\*9)+zz+2);

x=x+x1.substring((num\*9)+6,((num\*9)+6)+3);

num++;

}

x1=x;

}

**public** **void** supercharged()

{

**int** n2=random\_func(36)-1;

**int** a1=(**int**)(n2/6);

**int** a2=n2-(a1\*6);

**int** array[]={1,2,3,1,3,2,2,1,3,2,3,1,3,1,2,3,2,1};

shift\_hm(array[a1\*3],array[(a1\*3)+1],array[(a1\*3)+2]);

shift\_vm(array[a2\*3],array[(a2\*3)+1],array[(a2\*3)+2]);

**return**;

}

}

***5. Game Play***

**package** com.sarath.sv17;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.app.Dialog;

**import** android.content.Context;

**import** android.content.DialogInterface;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.ContextMenu;

**import** android.view.ContextMenu.ContextMenuInfo;

**import** android.view.MenuItem;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.~~AbsoluteLayout~~;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** android.widget.RelativeLayout;

**import** android.widget.TextView;

**import** android.widget.CheckBox;

**import** android.widget.Chronometer;

**import** android.widget.Toast;

**import** android.os.SystemClock;

**public** **class** Gameplay **extends** Activity {

/\*\* Called when the activity is first created. \*/

Chronometer mChronometer;

**int** sudoku\_array\_game[][]=**new** **int**[9][9];

**int** check\_array[][]=**new** **int**[9][9];

Button numarray\_button[]=**new** Button[10];

**int** array\_button\_dialog[]=**new** **int**[10];

TextView display\_textview\_array[]=**new** TextView[81];

EditText display\_EditText\_array[]=**new** EditText[81];

String x;

View v2;

**int** to\_return=0, id1, toggle=0, flag\_done=0;

**int** cell\_id;

**long** elapsedMillis=0;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*new\_play*);

EditText num00=(EditText) findViewById(R.id.*num00*);

cell\_id=num00.getId();

mChronometer = (Chronometer) findViewById(R.id.*chronometer1*);

x=getIntent().getExtras().getString("question");

start();

CheckBox chkIos = (CheckBox) findViewById(R.id.*checkBox1*);

chkIos.setOnClickListener(**new** OnClickListener()

{ @Override

**public** **void** onClick(View v)

{

**if** (((CheckBox) v).isChecked())

toggle=1;

**else**

toggle=0;

}

});

}

**public** **int** check\_between()

{

**int** flag=0;

capturescreen();

**for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

**if**(check\_array[i][j]==0&&sudoku\_array\_game[i][j]!=0)

**if**(!(val(sudoku\_array\_game[i][j],i,j)))

flag++;

**return** flag;

}

**public** **boolean** val(**int** num,**int** r,**int** c)

{

**int** ctr=0;

**for**(**int** i=0;i<9;i++)

**if**(sudoku\_array\_game[r][i]==num)

ctr++;

**if**(ctr!=1)

**return** **false**;

ctr=0;

**for**(**int** i=0;i<9;i++)

**if**(sudoku\_array\_game[i][c]==num)

ctr++;

**if**(ctr!=1)

**return** **false**;

ctr=0;

r = (r / 3) \* 3 ;

c = ( c/ 3) \* 3 ;

**for**( **int** i = 0;i<3;i++ )

**for**( **int** j = 0;j<3;j++ )

**if**(sudoku\_array\_game[r+i][c+j]==num)

ctr++;

**if**(ctr!=1)

**return** **false**;

**return** **true** ;

}

**public** **void** done()

{

flag\_done=0;

**for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

**if**(sudoku\_array\_game[i][j]==0)

{

flag\_done=1;

**return**;

}

**for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

**if**(check\_array[i][j]==0)

**if**(!(val(sudoku\_array\_game[i][j],i,j)))

flag\_done=2;

**return**;

}

**public** **void** popup\_done()

{

stop\_timer();

**final** Context mContext = **this**;

**final** Dialog dialog = **new** Dialog(mContext);

dialog.setContentView(R.layout.*highscore*);

dialog.setTitle(" Congratulations");

dialog.show();

RelativeLayout rl1=(RelativeLayout) dialog.findViewById(R.id.*done*);

rl1.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{

dialog.dismiss();

back\_to\_main();

}

});

**int** min=(**int**)(elapsedMillis/60000);

**int** sec=(**int**)((elapsedMillis/1000)-min\*60);

TextView pop\_do=(TextView) dialog.findViewById(R.id.*textView1*);

pop\_do.setText("You have solved the puzzle in \n"+min+" Minute(s) "+sec+" Second(s).");

reset\_timer();

**return**;

}

**public** **void** back\_to\_main()

{

AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);

builder.setMessage("Next is what?")

.setCancelable(**false**)

.setPositiveButton("Main menu", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

finish();

}

})

.setNegativeButton("Cancel", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

})

.setNeutralButton("New game", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

Intent myIntent = **new** Intent(Gameplay.**this**, GameEngine.**class**);

startActivity(myIntent);

finish();

}

});

AlertDialog alert = builder.create();

alert.show();

**return**;

}

**public** **void** start()

{

start\_timer();

put\_table();

setup\_display\_textview();

setup\_edittext\_listener();

display();

Button button1=(Button)findViewById(R.id.*button\_solve*);

button1.setOnClickListener(**new** OnClickListener()

{

@Override

**public** **void** onClick(View v)

{

registerForContextMenu(v);

openContextMenu(v);

unregisterForContextMenu(v);

}});

Button button2=(Button)findViewById(R.id.*reset*);

button2.setOnClickListener(**new** OnClickListener()

{

@Override

**public** **void** onClick(View v)

{

capturescreen();

done();

popup\_alert();

}});

Button pause=(Button)findViewById(R.id.*pause*);

pause.setOnClickListener(**new** OnClickListener()

{

@Override

**public** **void** onClick(View v)

{

stop\_timer();

pause\_pop();

}});

Button bback=(Button)findViewById(R.id.*b\_back*);

bback.setOnClickListener(**new** OnClickListener()

{

@Override

**public** **void** onClick(View v)

{

alerts(1);

}});

}

**public** **void** alerts(**int** code)

{

**final** **int** cd=code;

String show="";

String positive="Yes";

String negative="No";

**if**(code==1)

show="Back, are you sure?";

**if**(code==2)

show="Restart, are you sure?";

**if**(code==3)

show="New Game, are you sure?";

**if**(code==4)

show="Quit, are you sure?";

AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);

builder.setMessage(show)

.setCancelable(**false**)

.setPositiveButton(positive, **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id)

{

**if**(cd==1||cd==4) finish();

**if**(cd==2)

{

reset\_timer();

start();

}

**if**(cd==3)

{

Intent myIntent = **new** Intent(Gameplay.**this**, GameEngine.**class**);

startActivity(myIntent);

finish();

}

}

})

.setNegativeButton(negative, **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

AlertDialog alert = builder.create();

alert.show();

**return**;

}

**public** **void** start\_timer()

{

**int** stoppedMilliseconds = 0;

String chronoText = mChronometer.getText().toString();

String array[] = chronoText.split(":");

**if** (array.length == 2)

{

stoppedMilliseconds = Integer.*parseInt*(array[0]) \* 60 \* 1000

+ Integer.*parseInt*(array[1]) \* 1000;

}

**else** **if** (array.length == 3)

{

stoppedMilliseconds = Integer.*parseInt*(array[0]) \* 60 \* 60 \* 1000

+ Integer.*parseInt*(array[1]) \* 60 \* 1000

+ Integer.*parseInt*(array[2]) \* 1000;

}

mChronometer.setBase(SystemClock.*elapsedRealtime*() - stoppedMilliseconds);

mChronometer.start();

}

**public** **void** stop\_timer()

{

mChronometer.stop();

elapsedMillis = SystemClock.*elapsedRealtime*()- mChronometer.getBase();

}

**public** **void** reset\_timer()

{

mChronometer.setBase(SystemClock.*elapsedRealtime*());

mChronometer.stop();

}

@Override

**public** **void** onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)

{

**super**.onCreateContextMenu(menu, v, menuInfo);

menu.setHeaderTitle("Action");

menu.add(0, v.getId(), 0, "Restart Current Game");

menu.add(0, v.getId(), 0, "New Game");

menu.add(0, v.getId(), 0, "Check");

menu.add(0, v.getId(), 0, "Quit");

menu.add(0, v.getId(), 0, "Cancel");

}

**public** **void** popup\_alert()

{

AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);

**if**(flag\_done==1||flag\_done==2)

{

**if**(flag\_done==1)

{

builder.setMessage("The table is not complete.")

.setCancelable(**false**)

.setPositiveButton("Back to the game", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

}

**if**(flag\_done==2)

{

builder.setMessage("Sorry, the solution is incorrect.")

.setCancelable(**false**)

.setPositiveButton("Back to the game", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

}

AlertDialog alert = builder.create();

alert.show();

}

**if**(flag\_done==0)

{

popup\_done();

**return**;

}

}

@Override

**public** **boolean** onContextItemSelected(MenuItem item)

{

**if**(item.getTitle()=="Restart Current Game") alerts(2);

**else** **if**(item.getTitle()=="New Game") alerts(3);

**else** **if**(item.getTitle()=="Cancel") **return** **false**;

**else** **if**(item.getTitle()=="Quit") alerts(4);

**else** **if**(item.getTitle()=="Check")

{

AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);

**int** errors=check\_between();

**if**(errors==0)

{

builder.setMessage("So far so good.")

.setCancelable(**false**)

.setPositiveButton("Back to the game", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

}

**if**(errors>0)

{

builder.setMessage("You have made "+errors+" error(s).")

.setCancelable(**false**)

.setPositiveButton("Back to the game", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

}

AlertDialog alert = builder.create();

alert.show();

}

**else**

{

**return** **false**;

}

**return** **true**;

}

**public** **void** pause\_pop()

{

**final** Context mContext = **this**;

**final** Dialog dialog = **new** Dialog(mContext);

dialog.setContentView(R.layout.*poppause*);

dialog.setTitle("Paused");

dialog.show();

Button back=(Button) dialog.findViewById(R.id.*resume*);

back.setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{

dialog.dismiss();

start\_timer();

}

});

**return**;

}

**public** **void** setup\_edittext\_listener()

{

**final** ~~AbsoluteLayout~~ a1=(~~AbsoluteLayout~~) findViewById(R.id.*abs2*);

**for**(**int** i=0;i<81;i++)

{

v2 = a1.getChildAt(i);

**if**(v2 **instanceof** EditText)

{

EditText a2=(EditText) v2;

a2.setOnClickListener(**new** View.OnClickListener()

{

@Override

**public** **void** onClick(View view) {

**int** ids=view.getId();

**if**(toggle==1)

check1(ids);

**if**(toggle==0)

**for**(**int** i1=0;i1<10;i1++)

array\_button\_dialog[i1]=0;

numBoard(ids);

}

});

}

}

}

**public** **void** dialog\_function\_display()

{

**final** ~~AbsoluteLayout~~ a3=(~~AbsoluteLayout~~) findViewById(R.id.*abs2*);

**int** diff=id1-cell\_id;

TextView temp=(TextView) a3.getChildAt(diff);

temp.setTextColor(0xff993333);

**if**(to\_return!=0)

temp.setText(String.*valueOf*(to\_return));

**if**(to\_return==0)

temp.setText(String.*valueOf*(""));

capturescreen();

**return**;

}

**public** **void** check1(**int** ids1)

{

**int** i2=0;

**int** xstart=0,ystart=0;

display();

**int** cellcode=ids1-cell\_id;

**int** xx1=(**int**)(cellcode/9);

**int** xx2=cellcode-(9\*xx1);

**for**(**int** i1=0;i1<10;i1++)

array\_button\_dialog[i1]=0;

i2=xx1;

**for**(**int** j2=0;j2<9;j2++)

array\_button\_dialog[sudoku\_array\_game[i2][j2]]=1;

i2=xx2;

**for**(**int** j2=0;j2<9;j2++)

array\_button\_dialog[sudoku\_array\_game[j2][i2]]=1;

**if**(xx1<3) xstart=0;

**if**(xx1>2&&xx1<6) xstart=3;

**if**(xx1>5) xstart=6;

**if**(xx2<3) ystart=0;

**if**(xx2>2&&xx2<6) ystart=3;

**if**(xx2>5) ystart=6;

**for**(**int** k=xstart;k<xstart+3;k++)

**for**(**int** k1=ystart;k1<ystart+3;k1++)

array\_button\_dialog[sudoku\_array\_game[k][k1]]=1;

**return**;

}

**public** **void** capturescreen()

{

setup\_display\_edittext();

**int** ctr=0;

**for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

sudoku\_array\_game[i][j]=returnstr(display\_EditText\_array[ctr++].getText().toString());

**return**;

}

**public** **int** returnstr(String input1)

{

**if**(input1.length()==0)

**return** 0;

**else**

**return** Integer.*parseInt*(input1);

}

**public** **void** numBoard(**int** id2)

{

id1=id2;

**final** Context mContext = **this**;

**final** Dialog dialog = **new** Dialog(mContext);

dialog.setContentView(R.layout.*keyboard*);

dialog.setTitle("NumBoard");

dialog.show();

numarray\_button[0]=(Button) dialog.findViewById(R.id.*num0*);

numarray\_button[1]=(Button) dialog.findViewById(R.id.*num1*);

numarray\_button[2]=(Button) dialog.findViewById(R.id.*num2*);

numarray\_button[3]=(Button) dialog.findViewById(R.id.*num3*);

numarray\_button[4]=(Button) dialog.findViewById(R.id.*num4*);

numarray\_button[5]=(Button) dialog.findViewById(R.id.*num5*);

numarray\_button[6]=(Button) dialog.findViewById(R.id.*num6*);

numarray\_button[7]=(Button) dialog.findViewById(R.id.*num7*);

numarray\_button[8]=(Button) dialog.findViewById(R.id.*num8*);

numarray\_button[9]=(Button) dialog.findViewById(R.id.*num9*);

**for**(**int** h=1;h<10;h++)

**if**(array\_button\_dialog[h]==1)

numarray\_button[h].setEnabled(**false**);

**for**(**int** h=1;h<10;h++)

**if**(array\_button\_dialog[h]==0)

numarray\_button[h].setBackgroundResource(R.drawable.*b\_assist*);

numarray\_button[1].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{

to\_return=1;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[2].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{

to\_return=2;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[3].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{

to\_return=3;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[4].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=4;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[5].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=5;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[6].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{

to\_return=6;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[7].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=7;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[8].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=8;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[9].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=9;

dialog\_function\_display();

dialog.dismiss();

}

});

numarray\_button[0].setBackgroundResource(R.drawable.*b\_eraser*);

numarray\_button[0].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=0;

dialog\_function\_display();

dialog.dismiss();

}

});

Button back=(Button) dialog.findViewById(R.id.*numback*);

back.setEnabled(**false**);

Button back1=(Button) dialog.findViewById(R.id.*numback1*);

back1.setBackgroundResource(R.drawable.*b\_x*);

back1.setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{

dialog.dismiss();

}

});

**return**;

}

**public** **void** setup\_display\_edittext()

{

~~AbsoluteLayout~~ llMain = (~~AbsoluteLayout~~)findViewById(R.id.*abs2*);

**for**(**int** i=0;i<81;i++)

{

View v1 = llMain.getChildAt(i);

**if**(v1 **instanceof** EditText)

display\_EditText\_array[i]=(EditText) v1;

}

**return**;

}

**public** **void** put\_table()

{

**for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

check\_array[i][j]=0;

**int** ctr1=0;

**for**(**int** i=0;i<9;i++)

{

**for**(**int** j=0;j<9;j++)

{

sudoku\_array\_game[i][j]=Integer.*parseInt*(String.*valueOf*(x.charAt(ctr1++)));

**if**(sudoku\_array\_game[i][j]!=0)

check\_array[i][j]=1;

}

}

**return**;

}

**public** **void** setup\_display\_textview()

{

~~AbsoluteLayout~~ llMain = (~~AbsoluteLayout~~)findViewById(R.id.*abs2*);

**for**(**int** i=0;i<81;i++)

{

View v = llMain.getChildAt(i);

**if**(v **instanceof** TextView)

{

((TextView)v).setTextColor(0xffbe7b00);

display\_textview\_array[i]=(TextView) v;

}

}

}

**public** **void** display()

{ **int** ctr=0;

**for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

{ String string\_dis=display\_String(sudoku\_array\_game[i][j]);

display\_textview\_array[ctr++].setText(string\_dis);

}

**return**;

}

**public** String display\_String(**int** num)

{ String g="";

**if**(num==0)

**return** g;

**else**

**return** String.*valueOf*(num);

}

}

***6. Solver***

**package** com.sarath.sv17;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.app.ProgressDialog;

**import** android.os.Bundle;

**import** android.os.Handler;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.~~AbsoluteLayout~~;

**import** android.widget.Button;

**import** android.widget.TextView;

**import** android.widget.EditText;

**import** android.content.Context;

**import** android.content.DialogInterface;

**import** android.app.Dialog;

**public** **class** Sudoku\_solver **extends** Activity{

**int** sudoku\_array[][]=**new** **int**[9][9];

TextView display\_textview\_array[]=**new** TextView[81];

EditText display\_EditText\_array[]=**new** EditText[81];

EditText input\_string;

**int** cell\_id;

View v2;

**int** to\_return=0, id1;

Button numarray\_button[]=**new** Button[10];

**int** array\_button\_dialog[]=**new** **int**[10];

@Override

**public** **void** onCreate(Bundle savedInstanceState)

{ **super**.onCreate(savedInstanceState);

setContentView(R.layout.*sudoku\_solver*);

setup\_display\_textview();

EditText num00=(EditText) findViewById(R.id.*num00*);

cell\_id=num00.getId();

start();

}

**public** **void** setup\_display\_textview()

{ ~~AbsoluteLayout~~ llMain = (~~AbsoluteLayout~~)findViewById(R.id.*abs2*);

**for**(**int** i=0;i<81;i++)

{ View v = llMain.getChildAt(i);

**if**(v **instanceof** TextView)

{ ((TextView)v).setTextColor(0xffbe7b00);

display\_textview\_array[i]=(TextView) v; }}

**return**;

}

**public** **void** setup\_display\_edittext()

{ ~~AbsoluteLayout~~ llMain = (~~AbsoluteLayout~~)findViewById(R.id.*abs2*);

**for**(**int** i=0;i<81;i++)

{ View v1 = llMain.getChildAt(i);

**if**(v1 **instanceof** EditText)

display\_EditText\_array[i]=(EditText) v1; }

**return**;

}

**public** **void** capturescreen()

{ setup\_display\_edittext();

**int** ctr=0;

**for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

sudoku\_array[i][j]=returnstr(display\_EditText\_array[ctr++].getText().toString());

**return**;

}

**public** **void** check1(**int** ids1)

{ **int** i2=0;

**int** xstart=0,ystart=0;

**int** cellcode=ids1-cell\_id;

**int** xx1=(**int**)(cellcode/9);

**int** xx2=cellcode-(9\*xx1);

**for**(**int** i1=0;i1<10;i1++)

array\_button\_dialog[i1]=0;

i2=xx1;

**for**(**int** j=0;j<9;j++)

array\_button\_dialog[sudoku\_array[i2][j]]=1;

i2=xx2;

**for**(**int** j=0;j<9;j++)

array\_button\_dialog[sudoku\_array[j][i2]]=1;

**if**(xx1<3) xstart=0;

**if**(xx1>2&&xx1<6) xstart=3;

**if**(xx1>5) xstart=6;

**if**(xx2<3) ystart=0;

**if**(xx2>2&&xx2<6) ystart=3;

**if**(xx2>5) ystart=6;

**for**(**int** i=xstart;i<xstart+3;i++)

**for**(**int** j=ystart;j<ystart+3;j++)

array\_button\_dialog[sudoku\_array[i][j]]=1;

**return**;

}

**public** **void** setup\_edittext\_listener()

{ **final** ~~AbsoluteLayout~~ a1=(~~AbsoluteLayout~~) findViewById(R.id.*abs2*);

**for**(**int** i=0;i<81;i++)

{ v2 = a1.getChildAt(i);

**if**(v2 **instanceof** EditText)

{ EditText a2=(EditText) v2;

a2.setOnClickListener(**new** View.OnClickListener() {

@Override

**public** **void** onClick(View view) {

**int** ids=view.getId();

check1(ids);

numBoard(ids); } }); } }

**return**;

}

**public** **void** display()

{ **int** ctr=0;

**for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

display\_textview\_array[ctr++].setText(String.*valueOf*(sudoku\_array[i][j]));

**return**;

}

**public** **void** start()

{ **for**(**int** i=0;i<9;i++)

**for**(**int** j=0;j<9;j++)

sudoku\_array[i][j]=0;

setup\_edittext\_listener();

Button button1=(Button)findViewById(R.id.*button\_solve*);

button1.setOnClickListener(**new** OnClickListener()

{ @Override

**public** **void** onClick(View v)

{

start\_solve();

}});

Button button2=(Button)findViewById(R.id.*reset*);

button2.setOnClickListener(**new** OnClickListener()

{ @Override

**public** **void** onClick(View v)

{

reset(); }});

Button quit=(Button) findViewById(R.id.*bback1*);

quit.setOnClickListener(**new** OnClickListener()

{ @Override

**public** **void** onClick(View v) {

quit\_ask();

}});

**return**;

}

**public** **void** quit\_ask()

{

AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);

builder.setMessage("Back, are you sure?")

.setCancelable(**false**)

.setPositiveButton("Yes", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

finish();

}

})

.setNegativeButton("No", **new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** id) {

dialog.cancel();

}

});

AlertDialog alert = builder.create();

alert.show();

**return**;

}

**public** **void** dialog\_function\_display()

{ **final** ~~AbsoluteLayout~~ a3=(~~AbsoluteLayout~~) findViewById(R.id.*abs2*);

**int** diff=id1-cell\_id;

TextView temp=(TextView) a3.getChildAt(diff);

**if**(to\_return!=0) temp.setText(String.*valueOf*(to\_return));

**if**(to\_return==0) temp.setText(String.*valueOf*(""));

capturescreen();

**return**;

}

**public** **void** numBoard(**int** id2)

{ id1=id2;

**final** Context mContext = **this**;

**final** Dialog dialog = **new** Dialog(mContext);

dialog.setContentView(R.layout.*keyboard*);

dialog.setTitle("NumBoard");

dialog.show();

numarray\_button[0]=(Button) dialog.findViewById(R.id.*num0*);

numarray\_button[1]=(Button) dialog.findViewById(R.id.*num1*);

numarray\_button[2]=(Button) dialog.findViewById(R.id.*num2*);

numarray\_button[3]=(Button) dialog.findViewById(R.id.*num3*);

numarray\_button[4]=(Button) dialog.findViewById(R.id.*num4*);

numarray\_button[5]=(Button) dialog.findViewById(R.id.*num5*);

numarray\_button[6]=(Button) dialog.findViewById(R.id.*num6*);

numarray\_button[7]=(Button) dialog.findViewById(R.id.*num7*);

numarray\_button[8]=(Button) dialog.findViewById(R.id.*num8*);

numarray\_button[9]=(Button) dialog.findViewById(R.id.*num9*);

**for**(**int** h=1;h<10;h++)

**if**(array\_button\_dialog[h]==1)

numarray\_button[h].setEnabled(**false**);

**for**(**int** h=1;h<10;h++)

**if**(array\_button\_dialog[h]==0)

numarray\_button[h].setBackgroundResource(R.drawable.*b\_assist*);

numarray\_button[1].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=1;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[2].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=2;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[3].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=3;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[4].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=4;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[5].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=5;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[6].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=6;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[7].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=7;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[8].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=8;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[9].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=9;

dialog\_function\_display();

dialog.dismiss(); } });

numarray\_button[0].setBackgroundResource(R.drawable.*b\_eraser*);

numarray\_button[0].setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{ to\_return=0;

dialog\_function\_display();

dialog.dismiss(); } });

Button back1=(Button) dialog.findViewById(R.id.*numback1*);

back1.setBackgroundResource(R.drawable.*b\_x*);

back1.setOnClickListener(**new** View.OnClickListener()

{ @Override

**public** **void** onClick(View view)

{

dialog.dismiss(); } });

Button back=(Button) dialog.findViewById(R.id.*numback*);

back.setEnabled(**false**);

**return** ;

}

**public** **void** reset()

{ setContentView(R.layout.*sudoku\_solver*);

setup\_display\_textview();

start();

**return**;

}

**public** **void** start\_solve()

{ run\_solve();

**final** ProgressDialog dialog = ProgressDialog.*show*(Sudoku\_solver.**this**, "", "Solving.", **true**);

**new** Handler().postDelayed(**new** Runnable() {

@Override

**public** **void** run() {

dialog.dismiss();

display();

}

}, 2000);

**return**;

}

/\*SUDOKU SOLVER\*/

**public** **void** run\_solve()

{ **try**

{ start\_finding( 0, 0 ) ; }

**catch**( Exception e ) { }

}

**public** **void** next\_cell\_find( **int** r, **int** c ) **throws** Exception

{ **if**( c < 8 ) start\_finding( r, c + 1 ) ;

**else** start\_finding( r + 1, 0 ) ;

}

**public** **void** start\_finding( **int** r, **int** c ) **throws** Exception

{ **if**( r > 8 ){

**throw** **new** Exception( "Solution found" ) ; }

**if**(sudoku\_array[r][c] != 0 )

next\_cell\_find( r, c ) ;

**else**

{ **for**( **int** num = 1; num < 10; num++ )

{ **if**( checkRow(r,num) && checkCol(c,num) && checkBox(r,c,num) )

{ sudoku\_array[r][c] = num ;

next\_cell\_find( r, c ) ; } }

sudoku\_array[r][c] = 0 ; }

}

**protected** **boolean** checkBox( **int** row, **int** col, **int** num )

{ row = (row / 3) \* 3 ;

col = (col / 3) \* 3 ;

**for**( **int** r = 0; r < 3; r++ )

**for**( **int** c = 0; c < 3; c++ )

**if**(sudoku\_array[row+r][col+c] == num )

**return** **false** ;

**return** **true** ;

}

**protected** **boolean** checkRow( **int** row, **int** num )

{ **for**( **int** col = 0; col < 9; col++ )

**if**( sudoku\_array[row][col] == num )

**return** **false** ;

**return** **true** ;

}

**protected** **boolean** checkCol( **int** col, **int** num )

{ **for**( **int** row = 0; row < 9; row++ )

**if**( sudoku\_array[row][col] == num )

**return** **false**;

**return** **true** ;

}

**public** **int** returnstr(String input1)

{ **if**(input1.length()==0) **return** 0;

**else** **return** Integer.*parseInt*(input1);

}

}

***7.Credit***

**package** com.sarath.sv17;

**import** android.app.Activity;

**import** android.app.Dialog;

**import** android.content.Context;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**import** android.widget.RelativeLayout;

**public** **class** Credit **extends** Activity{

@Override

**public** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*credit*);

Button sv1=(Button)findViewById(R.id.*sv1*);

sv1.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{

display(1);

}

});

Button sv2=(Button)findViewById(R.id.*sv2*);

sv2.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{

display(2);

}

});

Button vs1=(Button)findViewById(R.id.*vs1*);

vs1.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{

display(3);

}

});

Button back=(Button)findViewById(R.id.*b\_back*);

back.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{

finish();

}

});

}

**public** **void** display(**int** a)

{

**final** Context mContext = **this**;

**final** Dialog dialog = **new** Dialog(mContext);

dialog.setContentView(R.layout.*sv1*);

RelativeLayout rl1=(RelativeLayout) dialog.findViewById(R.id.*rlsv*);

**if**(a==1) rl1.setBackgroundResource(R.drawable.*bg\_sarath*);

**if**(a==2) rl1.setBackgroundResource(R.drawable.*bg\_sujith*);

**if**(a==3) rl1.setBackgroundResource(R.drawable.*bg\_vig*);

rl1.setOnClickListener(**new** View.OnClickListener()

{

**public** **void** onClick(View view)

{

dialog.dismiss();

}

});

dialog.show();

**return**;

}

}

Graphics















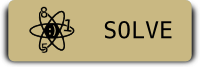








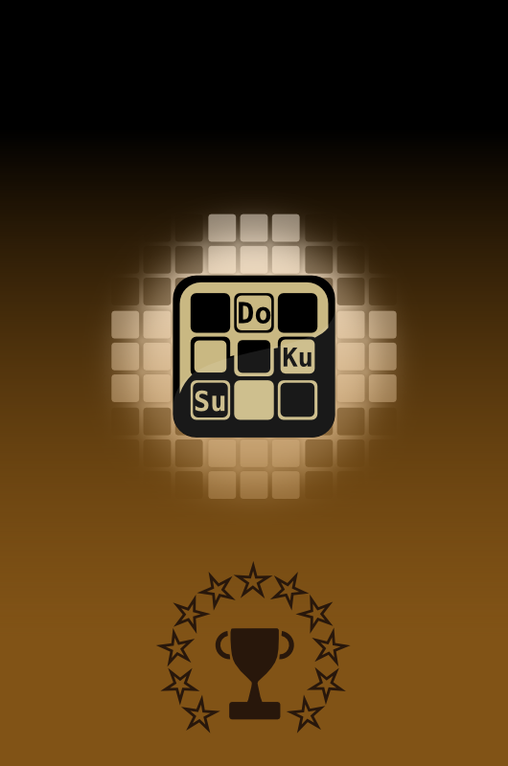








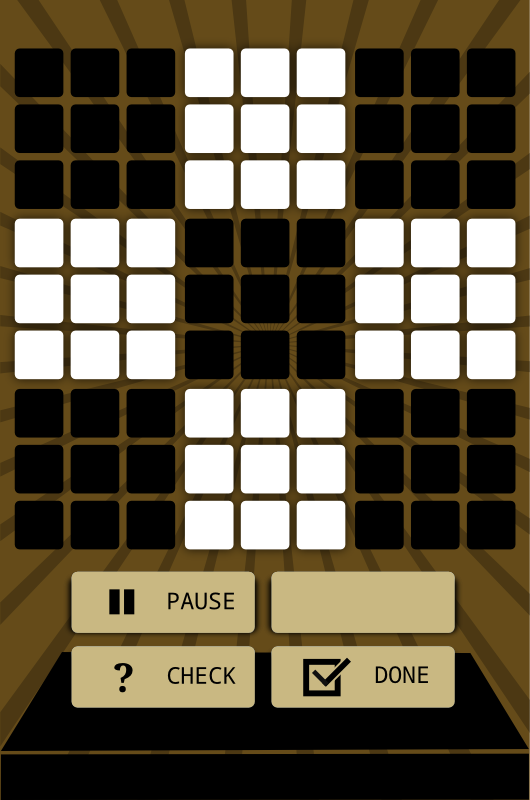




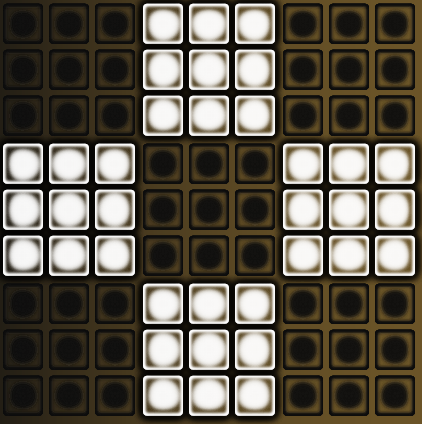


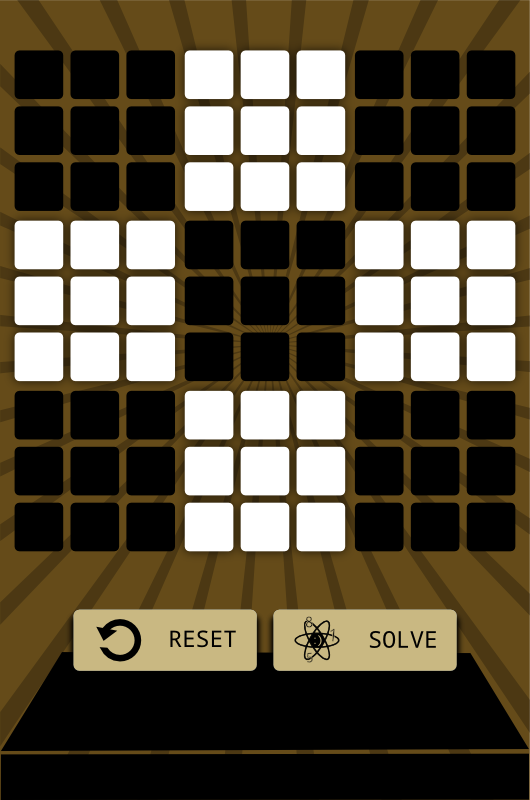


















Screen Shots

Taken From an actual Device

Device Name : HTC Desire S

Manufacturer : HTC

Screen Size : 3.7”

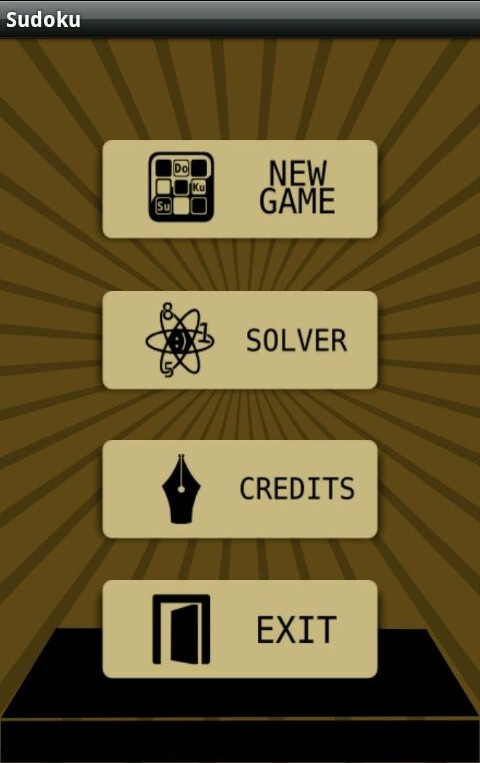
OS Version : v2.3 Gingerbread

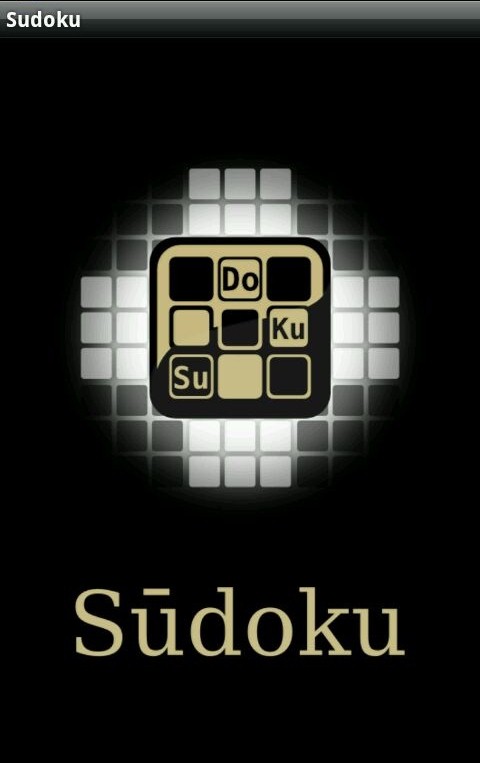
Date : 19th July, 2012

Courtesy : Prahlad Suresh

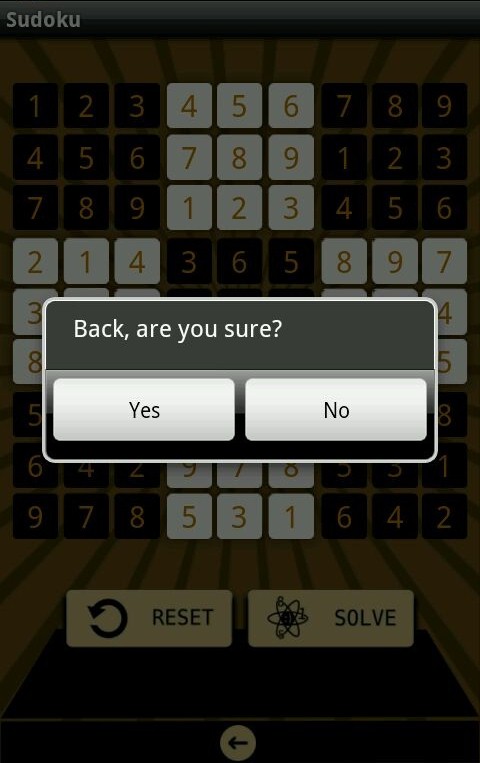






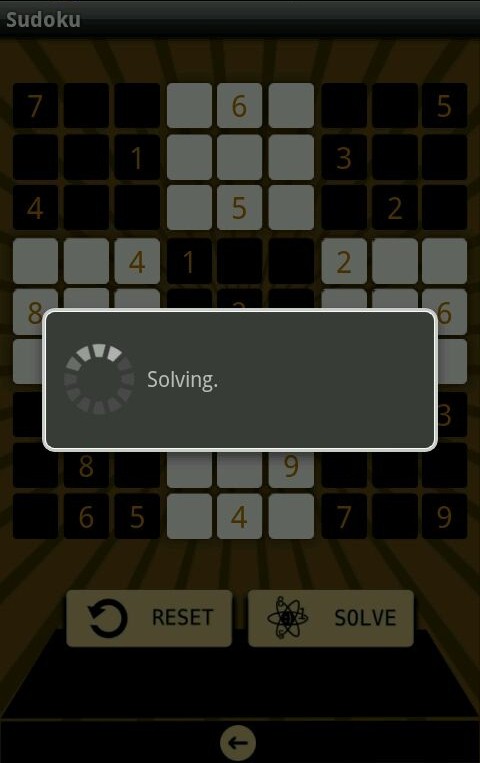




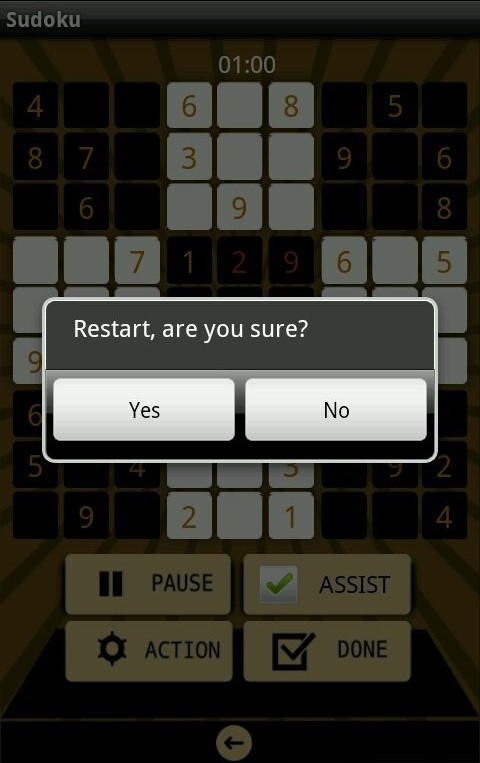


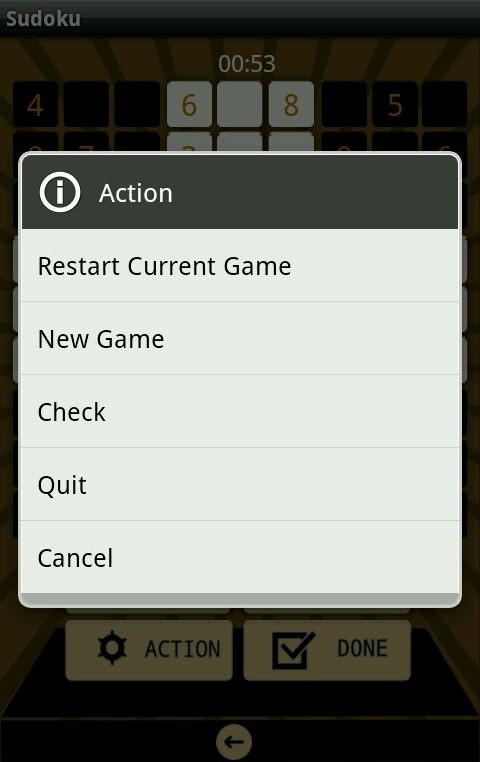


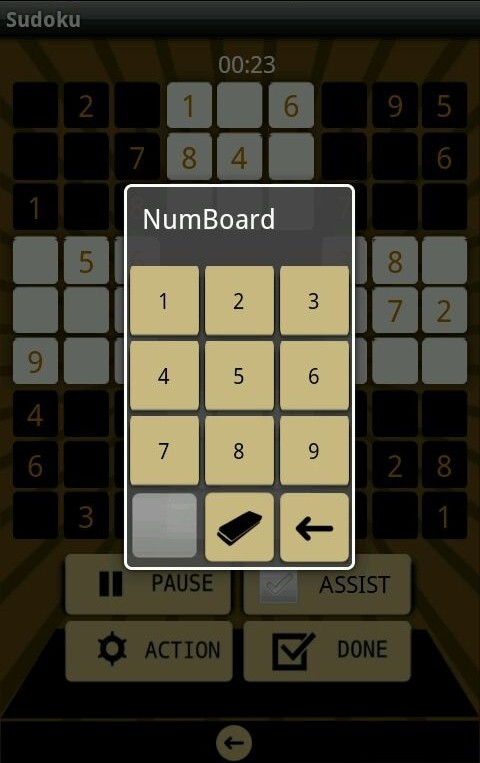




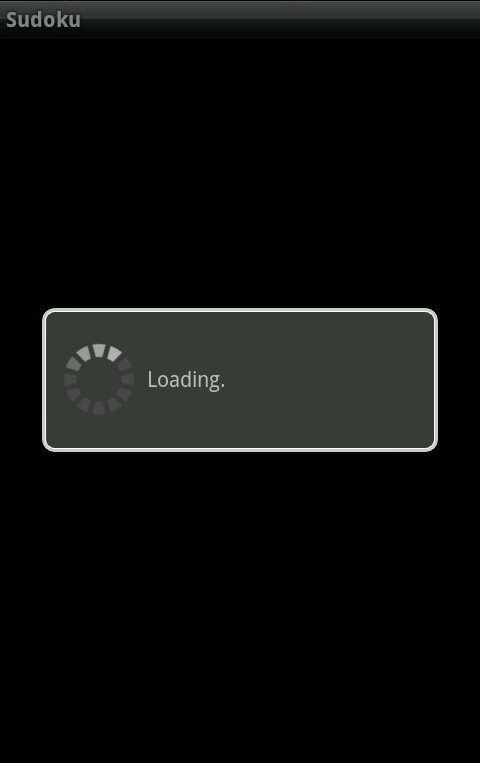






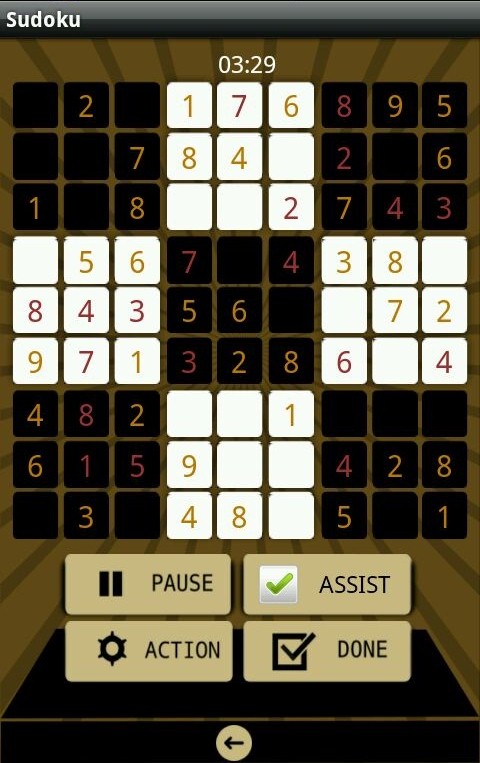


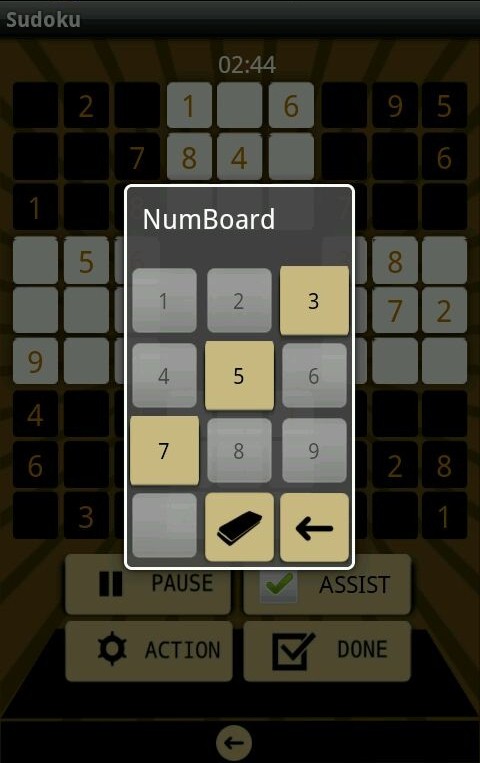




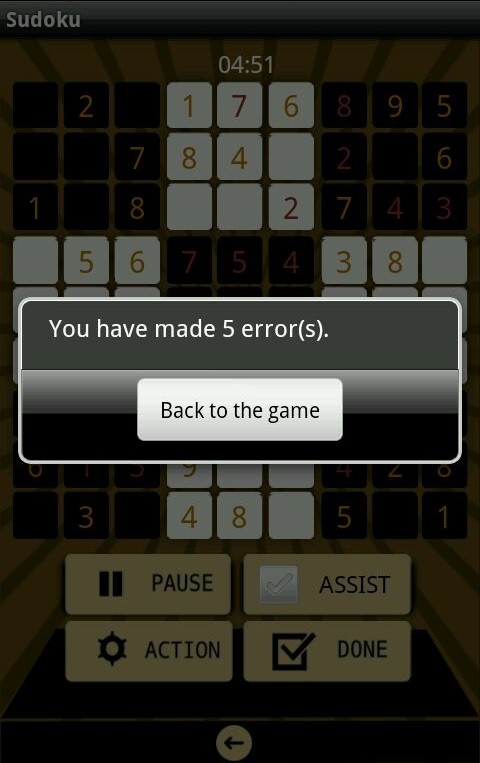


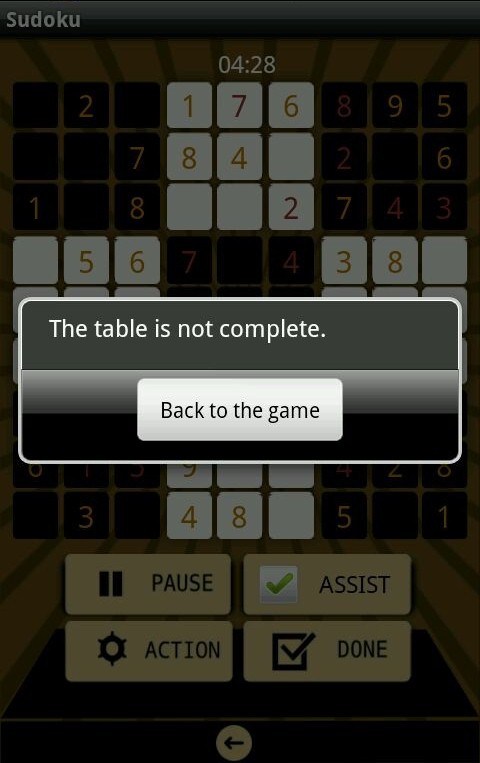


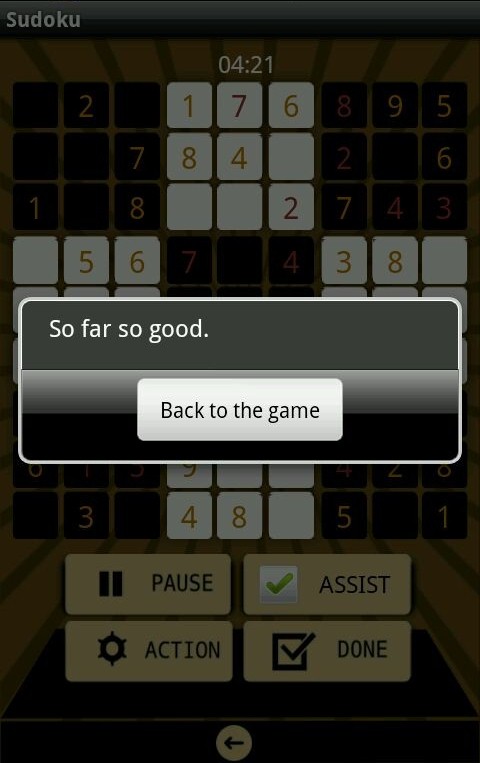




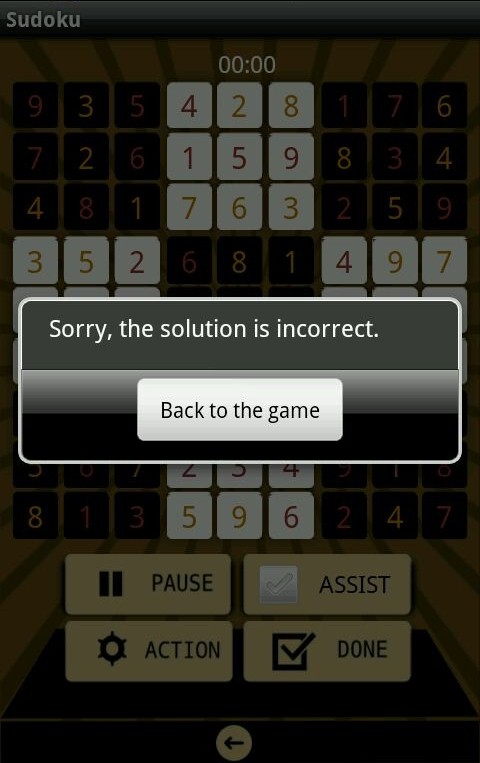


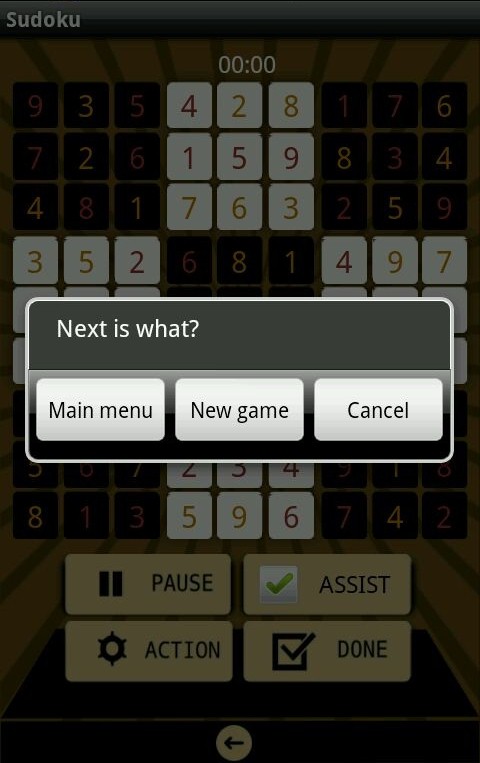




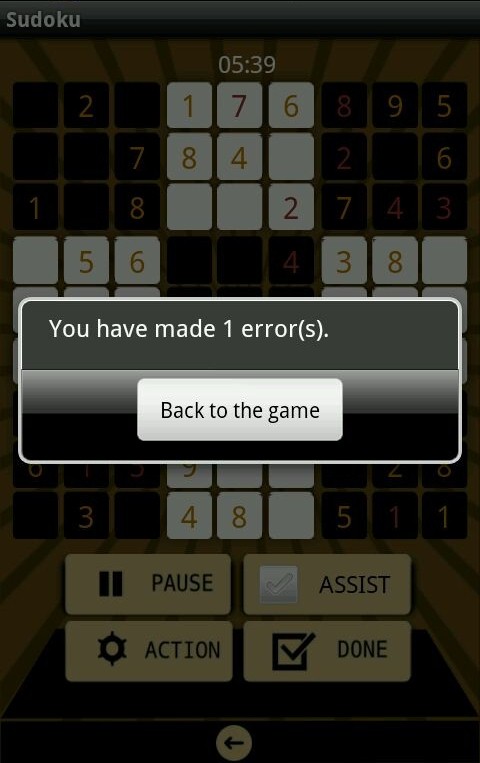


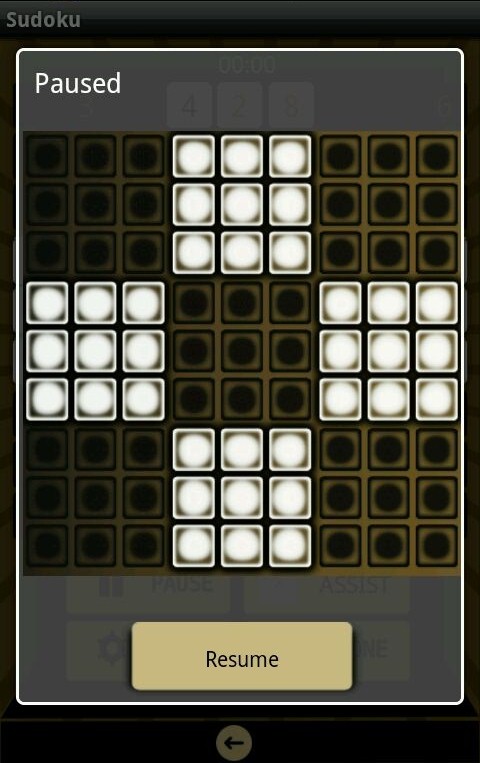


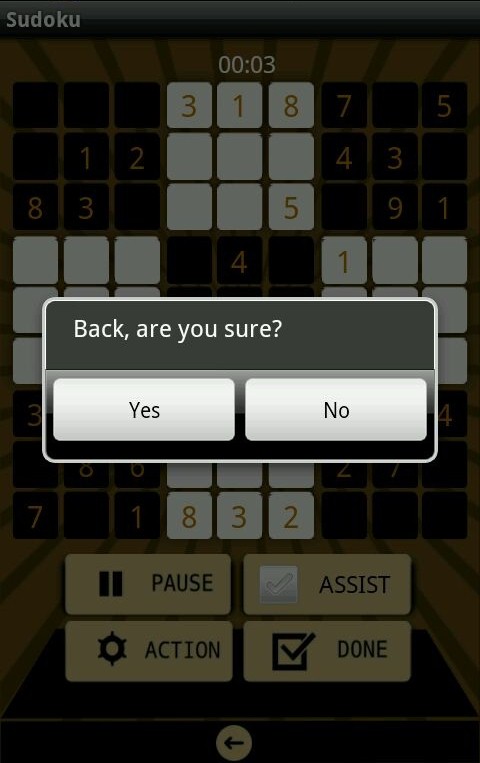












References

The project was developed mainly using the official android documentation

<http://developer.android.com/guide/components/index.html>

Various tech blogs depicting the use of codes in sample programs were referred for guidance.

Developer Page

***Code:***

Sarath Vadakkepat

@srthvadakkepat

[sarath.vadakkepat@gmail.com](mailto:sarath.vadakkepat@gmail.com)

***Graphics:***

Vigneshwar Shankar

@vgnshwar

[vigneshwar.s@gmail.com](mailto:vigneshwar.s@gmail.com)

