VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018

Mobile Application Development - Mini Project Report

on

SUDOKU SOLVER APP

Submitted in partial fulfillment of the requirements for the VI semester

Bachelor of Engineering in Computer Science

of Visvesvaraya Technological University, Belagavi

Submitted by:

KASHISH 1RN20CS064
POOJITHA C 1RN20CS098

Under the Guidance of:
Mr. Prasanna Kumar
Assistant Professor
Dept. of CSE

Department of Computer Science and Engineering
(Accredited by NBA up to 30-06-2025)
RNS Institute of Technology
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098

2022-2023

RNS Institute of Technology

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(Accredited by NBA up to 30-06-2025)

CERTIFICATE

Certified that the mini project work entitled **"SUDOKU SOLVER APP"** has been successfully carried out by **KASHISH** bearing USN **1RN20CS064** and **POOJITHA C** bearing USN **1RN20CS098**, bonafide students of **"RNS Institute of Technology"** in partial fulfillment of the requirements for the 6th semester of **"Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University"**, Belagavi, during academic year 2022-2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the Mobile Application Development Laboratory requirements.

Signature of the Guide	Signature of the HoD	Signature of the Principal
Mr. Prasanna Kumar	**Dr. Kiran P**	**Dr. Ramesh Babu H S**
Assistant Professor	Professor	Principal
Dept. of CSE	Dept. of CSE	

External Viva:

Name of the Examiners	Signature with Date
1.	
2.	

Acknowledgement

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We are grateful to Management and **Dr. Ramesh Babu H S**, Principal, RNSIT, Bangalore, for his support towards completing this mini project.

We would like to thank **Dr. Kiran P**, Professor & Head, Department of Computer Science & Engineering, RNSIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express our sincere gratitude to our guide **Mr. Prasanna Kumar**, Assistant Professor, Department of CSE, RNSIT, Bangalore, for her able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank all the teaching and non-teaching staff of Department of Computer Science & Engineering, RNSIT, Bengaluru for their constant support and encouragement

Abstract

Sudoku game is well famous and popular game among many players all over the world. This report details the development of a Sudoku game application that is written in Java. The application is even developed to work in Android systems. In addition, the report details the implementation of the complexity of the algorithms used to solve any kind of Sudoku puzzle. Also, how to generate a puzzle with different level of difficulties and make sure there will be only one solution.

The aim of the report is also to discuss the backtracking, brute force algorithms and other logics in order to create and solve Sudoku puzzles. Furthermore, the user-friendly environment is considered in the report as the rules of Sudoku are connected to the interface.

Moreover, the report specifics how well these methods of solving solves the puzzle and achieved the goal of this implementation. The report concludes by evaluating the end application to analyse how good it met its objectives and the performance of solving algorithms. Finally, the report summarises the overall achievements of the application development and indicates other possible extensions.

Sudoku, in its simplest form, is defined as a n 2 × n 2 grid that is initially filled in with a certain number of cells. The objective is to fill every cell with numbers 1 to n 2 , without using any number more than once in the same row, column, or n × n block. Due to the puzzle's popularity, numerous variants have emerged, changing the size of the puzzle, or adding additional rules. In this report, the design, implementation, and evaluation of an android Sudoku solving app is discussed. The developed app allows users to solve a puzzle using pencil marks, while allowing them to request hints which are generated based on an extensive set on human solving techniques. The app was also evaluated by multiple users of variable Sudoku knowledge and familiarity with similar apps.

Contents

List of Figures

Chapter 1

Introduction

1.1 About the Project

Originally, Sudoku is a Japanese puzzle game. It means “Single Number” or “Number Place”. The game consists of a 9x9 grid, which is divided into nine 3x3 grids (called boxes or sub-grids). So, there are 9 rows and 9 columns. The intersection of a row and a column is called a cell. The result of the total cells is 81. Figure 1(a) is an example of a Sudoku puzzle with empty grid. When start the Sudoku puzzle, a player will be provided with random numbers in random cells which called givens; figure 1(b) shows an example of real Sudoku. The objective is to fill the empty cells with only one number from 1 to 9 in the shortest time. There are 3 rules in order to complete and accomplish all of the empty cells:

1) The number must only occur in a column once.

2) The number must only occur in a row once.

3) The number must occur in a sub-grid only once.

Moreover, this means that the solution of any puzzle should have one solution that isunique. The difficulty of the puzzle depends not only how many numbers are given, butalso on the placement of the given cells

1.1.1 Android Studio

Android Studio is the official integrated development environment (IDE) for Google’s Android operating system, built on JetBrains’ IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

1.1.2 Android SDK

The Android SDK is a software development kit that includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

Until around the end of 2014, the officially-supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin.As of 2015, Android Studio, is the official IDE; however, developers are free to use others, but Google made it clear that ADT was officially deprecated since the end of 2015 to focus on Android Studio as the official Android IDE. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

1.1.3 Emulator

The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.

The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.

Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over

USB.

The emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.

1.1.4 Firebase

Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side which makes it easy to use its features more efficiently. It provides services to android, ios, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data. In this architecture, Firebase sits between the server and clients. Your servers can connect to Firebase and interact with the data just like any other client would. In other words, your server communicates with clients by manipulating data in Firebase. Our Security and Firebase Rules language lets you assign full access to your data to your server. Your server code can then listen for any changes to data made by clients, and respond appropriately. In this configuration, even though you're still running a server, Firebase is handling all of the heavy lifting of scale and real-time updates. Firebase initially was an online chat service provider to various websites through API and ran with the name Envolve. It got popular as developers used it to exchange application data like a game state in real time across their users more than the chats. This resulted in the separation of the Envolve architecture and it's chat system. The Envolve architecture was further evolved by it's founders James Tamplin and Andrew Lee,to what modern day Firebase is in the year 2012. In this architecture, Firebase sits between the server and clients. Your servers can connect to Firebase and interact with the data just like any other client would. In other words, your server communicates with clients by manipulating data in Firebase. Our Security and Firebase Rules language lets you assign full access to your data to your server. Your server code can then listen for any changes to data made by clients, and respond appropriately. In this configuration, even though you're still running a server, Firebase is handling all of the heavy lifting of scale and real-time updates.

1.1.5 Java

Java is a programming language independent of all platforms and can be used for multiple operating systems.Keeping security in mind, all other programming languages are developed, including the interpreter, compiler, and run-time environment. A lot of concentration is put on testing to ensure potential early errors are caught.Java is the first choice of android app developers because of ease of use, robustness, security features, and cross-platform development capabilities.

1.1.6 XML

Extensible Markup Language (XML) is a markup language and file format for storing, transmitting, and reconstructing arbitrary data. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The World Wide Web Consortium's XML 1.0 Specification of 1998 and several other related specifications—all of them free open standards—define XML.

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

1.2 Existing System

Since we are advancing in the use of technology there are many bakeries converting to this convenient method where in they can virtually receive orders and the hassle for the customers also decreases since all the need is their phone and the app to order rather than physically going over to the bakery if not to several bakeries because there are chances of anyone to decline the order.

1.2.1 Limitation of Existing System

The Bakery App could be improved by adding some more products in the order section. "Order"can be improved by allowing user to maintain multiple carts. Further changes can be easily done by changing the code. The front end can be made more attractive by using attractive layout. Due to the practical challenges, instant logistics is facing many difficulties, while developing rapidly.This app providers face the complex problem as to control the cost of scheduling riders, while maintaining high quality of customers' service. That is, takeout service providers struggle to efficiently assign orders to riders for instant deliveries. The issue is an inherent contradiction between the undulated distribution of order time, space locations, and available riders' stability. We would further like to improve the app by providing more options for editing user profile.

Some of the major limitations that persons with bakery app are:

- To make customer service interactions so that the administrators aware of the problems faced by the customer.

- To list the products if they are out of stock.
- To add an option to modify or update the products after the order is confirmed.

1.3 Problem Statement

The task is to solve the sudoku puzzle in as much less time as possible. It can be done by investigating different techniques for solving sudoku and comparing them for the most efficient solution. Sudoku itself can be solved using brute-force in a reasonable amount of time in most cases, but there are special cases where it takes a long time to brute-force. Therefore our task is to try to find efficient algorithms for all instances of the problem and evaluate them while using the optimal solver to get the solution for the sudoku puzzle. There are two main constraints that determine the efficiency of a optimal solver. They are Time consumption and Memory consumption whose degree of satisfaction determines the quality of a optimal sudoku solver. These constraints usually vary from algorithm to algorithm.

Chapter 2

Requirement Analysis

2.1 Hardware Requirements

The hardware requirements are very minimal and the program can be run on most of the machines

Processor : Qualcomm Snapdragon processor

Processor Speed : 1.4 GHz

RAM : 2 GB

Storage Space : 10 GB

Display Resolution : 1024*768

I/O Elements : Camera, Speaker, Microphone, GPS

Network : 5 Mbps

2.2 Software Requirement

Operating System : Android / iOS

Chapter 3

System Design

3.1 System Architecture

A standard Sudoku contains 81 cells, in a 9×9 grid, and has 9 boxes, each box being the intersection of the first, middle, or last 3 rows, and the first, middle, or last 3 columns. Each cell may contain a number from one to nine, and each number can only occur once in each row, column, and box. A Sudoku starts with some cells containing numbers (clues), and the goal is to solve the remaining cells. Proper Sudoku have one solution. Players and investigators use a wide range of computer algorithms to solve Sudoku, study their properties, and make new puzzles, including Sudoku with interesting symmetries and other properties.

Solving a Sudoku puzzle can be rather tricky, but the rules of the game are quite simple. Solving a sudoku puzzle does not require knowledge of mathematics; simple logic suffices. 9 The objective of sudoku is to enter a digit from 1 through 9 in each cell, in such a way that:

I. Each horizontal row contains each digit exactly once.

II. Each vertical column contains each digit exactly once

III. Each subgrid or region contains each digit exactly once.

The project consists of the following parts as shown in figure 3.1

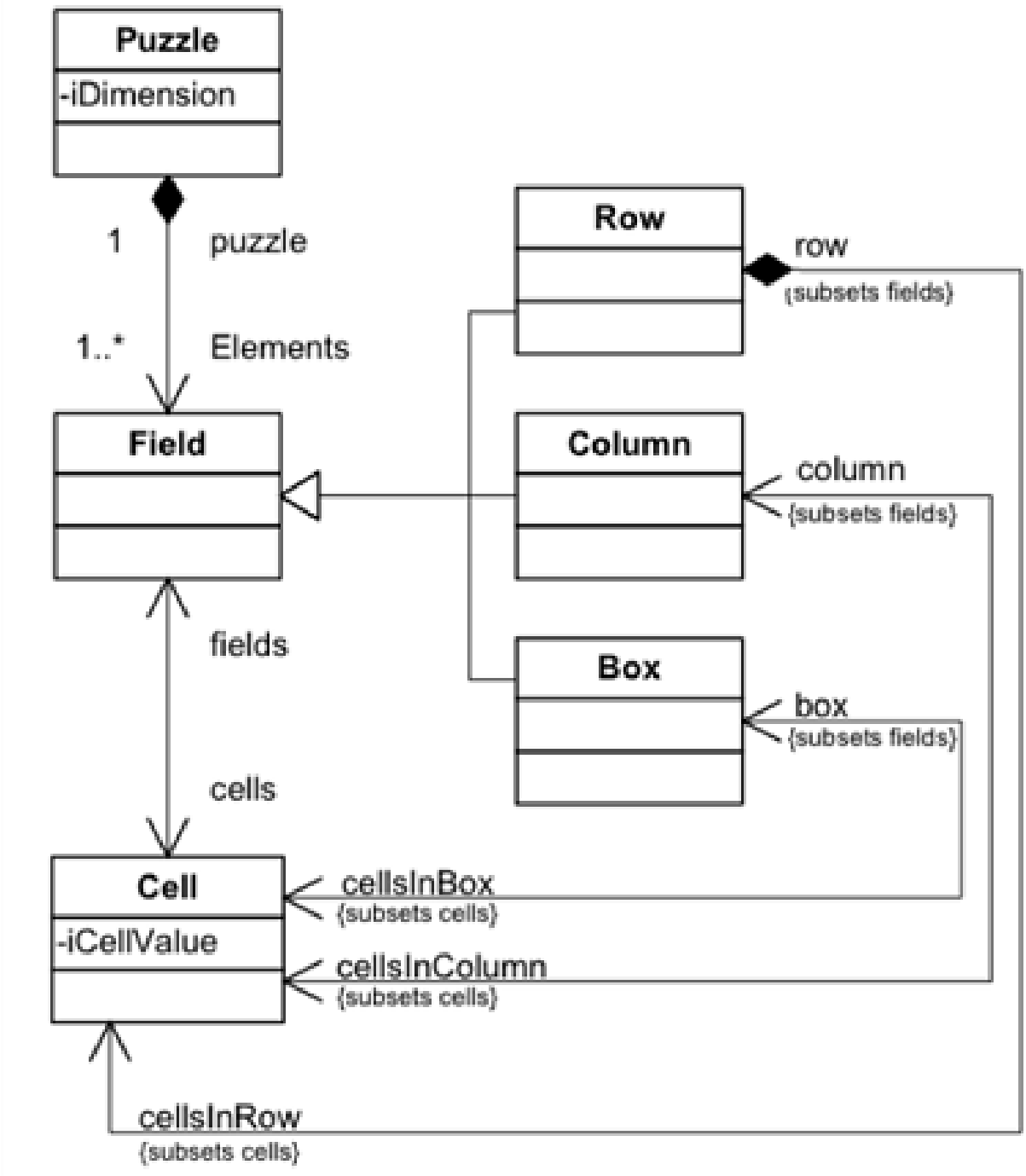


Figure 3.1: Flowchart of the application

Chapter 4

Implementation

4.1 MainActivity.java

```
package com.practicalcoding.sudokusolver;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
    private SudokuBoard gameBoard;
    private Solver gameBoardSolver;
    private Button solveBTN;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        gameBoard = findViewById(R.id.SudokuBoard);
        gameBoardSolver = gameBoard.getSolver();
        solveBTN = findViewById(R.id.solveButton);
    }public void BTNOnePress(View view){
        gameBoardSolver.setNumberPos(1);
        gameBoard.invalidate();
    }public void BTNTwoPress(View view){
        gameBoardSolver.setNumberPos(2);
```

```
    gameBoard.invalidate();
}public void BTNThreePress(View view){
    gameBoardSolver.setNumberPos(3);
    gameBoard.invalidate();
}public void BTNFourPress(View view){
    gameBoardSolver.setNumberPos(4);
    gameBoard.invalidate();
}public void BTNFivePress(View view){
    gameBoardSolver.setNumberPos(5);
    gameBoard.invalidate();
}public void BTNSixPress(View view){
    gameBoardSolver.setNumberPos(6);
    gameBoard.invalidate();
}public void BTNSevenPress(View view){
    gameBoardSolver.setNumberPos(7);
    gameBoard.invalidate();
}public void BTNEightPress(View view){
    gameBoardSolver.setNumberPos(8);
    gameBoard.invalidate();
}public void BTNNinePress(View view){
    gameBoardSolver.setNumberPos(9);
    gameBoard.invalidate();
}public void solve(View view){
    if (solveBTN.getText().toString().equals(getString(R.string.solve))){
        solveBTN.setText(getString(R.string.clear));
        gameBoardSolver.getEmptyBoxIndexes();
        SolveBoardThread solveBoardThread = new SolveBoardThread();
        new Thread(solveBoardThread).start();
        gameBoard.invalidate();
    }else{
        solveBTN.setText(getString(R.string.solve));
        gameBoardSolver.resetBoard();
        gameBoard.invalidate();
    }}class SolveBoardThread implements Runnable{
```

```
        @Override
        public void run(){
            gameBoardSolver.solve(gameBoard);
        }}}
```

4.2 Solver.java

```
package com.practicalcoding.sudokusolver;
import java.util.ArrayList;
class Solver {int[][] board;
ArrayList<ArrayList<Object>> emptyBoxIndex;
int selected_row;int selected_column;
Solver(){
selected_row = -1;selected_column = -1;
board = new int[9][9];
for (int r=0; r<9; r++){
    for (int c=0; c<9; c++){
         board[r][c] = 0;
       }
   }emptyBoxIndex = new ArrayList<>();
    }
public void getEmptyBoxIndexes(){
        for (int r=0; r<9; r++){
            for (int c=0; c<9; c++){
                if (this.board[r][c] == 0){
                    this.emptyBoxIndex.add(new ArrayList<>());
                    this.emptyBoxIndex.get(this.emptyBoxIndex.size()-1).add(r);
                    this.emptyBoxIndex.get(this.emptyBoxIndex.size()-1).add(c);
 }}}}
private boolean check(int row, int col){
        if (this.board[row][col] > 0){
            for (int i=0; i<9; i++){
                if (this.board[i][col] == this.board[row][col] && row != i){
```

```
                return false;
            }if (this.board[row][i] == this.board[row][col] && col != i){
                return false;
            }}
            int boxRow = row/3;
        int boxCol = col/3;
        for (int r=boxRow*3; r<boxRow*3 + 3; r++){
            for (int c=boxCol*3; c<boxCol*3 + 3; c++){
                if (this.board[r][c] == this.board[row][col] && row != r && col !=
                    return false;
                }}}}
    return true;
}
public boolean solve(SudokuBoard display){
    int row = -1;
    int col = -1;
    for (int r=0; r<9; r++){
        for (int c=0; c<9; c++){
            if (this.board[r][c] == 0){
                row = r;
                col = c;
                break;
            }
        }
    }if (row == -1 || col == -1){
        return true;
    } for (int i=1; i<10; i++){
        this.board[row][col] = i;
        display.invalidate();
        if (check(row, col)){
            if (solve(display)){
                return true;
            } } this.board[row][col] = 0;
    }return false;
```

```
}public void resetBoard(){
        for (int r=0; r<9; r++){
            for (int c=0; c<9; c++){
                board[r][c] = 0;
            }}
        this.emptyBoxIndex = new ArrayList<>();
}
public void setNumberPos(int num){
        if (this.selected_row != -1 && this.selected_column != -1){
            if (this.board[this.selected_row-1][this.selected_column-1] == num){
                this.board[this.selected_row-1][this.selected_column-1] = 0;
            }else{
                this.board[this.selected_row-1][this.selected_column-1] = num;
                if (!check(this.selected_row-1, this.selected_column-1)){
                    this.board[this.selected_row-1][this.selected_column-1] = -num;
                }}}}public int[][] getBoard(){
        return this.board;
    }public ArrayList<ArrayList<Object>> getEmptyBoxIndex(){
        return this.emptyBoxIndex;
    }public int getSelectedRow(){
        return selected_row;
    } public int getSelectedColumn(){
        return selected_column;
    } public void setSelectedRow(int r){
        selected_row = r;
    } public void setSelectedColumn(int c){
        selected_column = c;
}}
```

4.3 Solver.java

```
<resources>
    <string name="app_name">Sudoku Solver</string>
    <string name="One">1</string>
```

```
    <string name="Two">2</string>
    <string name="Three">3</string>
    <string name="Four">4</string>
    <string name="Five">5</string>
    <string name="Six">6</string>
    <string name="Seven">7</string>
    <string name="Eight">8</string>
    <string name="Nine">9</string>
    <string name="solve">Solve</string>
    <string name="clear">Clear</string>
</resources>
```

4.4 attrs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="SudokuBoard">
        <attr name="boardColor" format="color"/>
        <attr name="cellFillColor" format="color"/>
        <attr name="cellsHighlightColor" format="color"/>
        <attr name="letterColor" format="color"/>
        <attr name="letterColorSolve" format="color"/>
        <attr name="letterColorError" format="color"/>
    </declare-styleable>
</resources>
```

Chapter 5

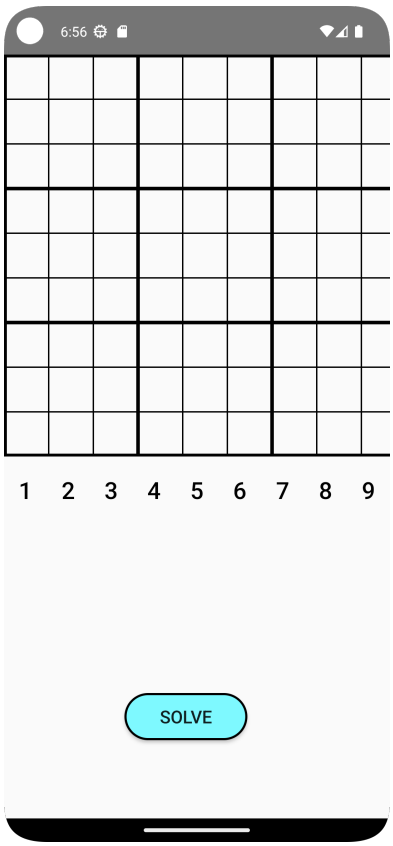
Result Analysis

5.1 Testing

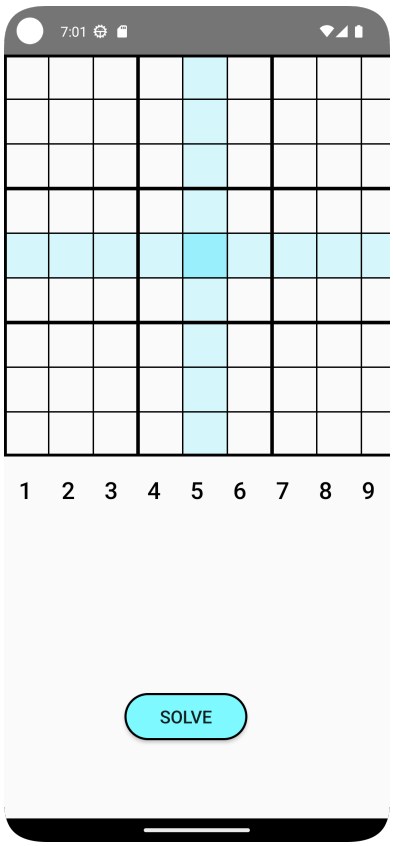
Table 5.1 gives details of validation.

Table 5.1: Test Case Validation

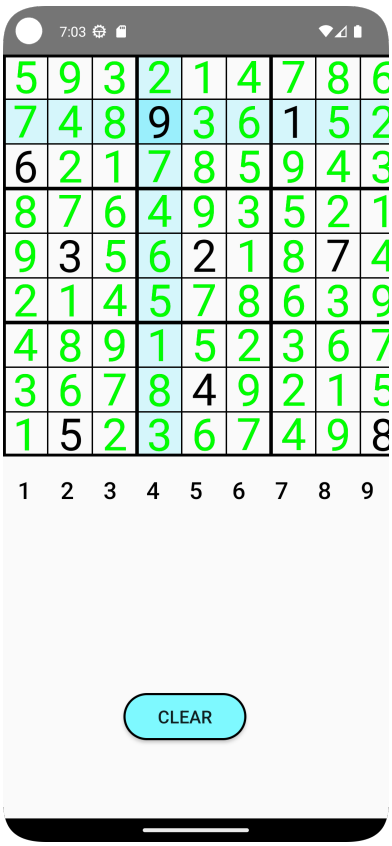
Test Case No.	**Input**	**Expected Output**	**Actual Output**
1	onClick Event	On clicking the sudokuboard the corresponding row and column should be selected	The appropriate row and column is selected
2	Buttons 1 to 9	The pressed button should be displayed on the sudokuboard as input to solve the puzzle	The appropriate number is displayed on the sudokuboard
3	Solve Button	When the user presses the solve button the sudokuboard puzzle should be solved	The sudokuboard puzzle is solved
4	Clear Button	When the user presses the clear button the sudokuboard has to be cleared	The sudokuboard is cleared



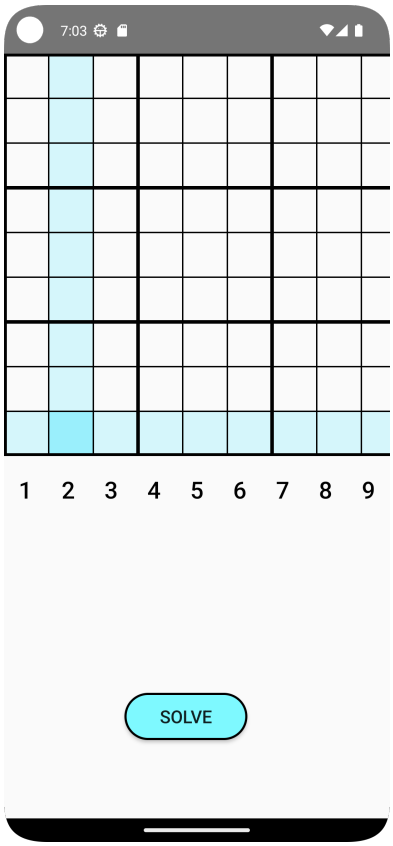
(a) layout of Sudoku Board



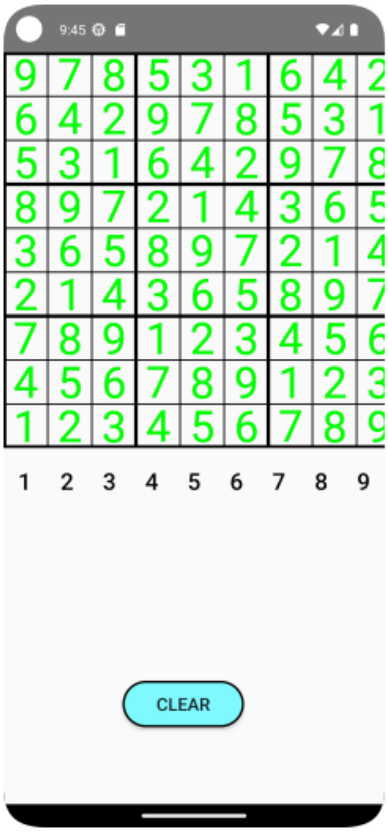
(b) Clicking on Sudoku Board



(c) Solved Sudoku Board



(d) Sudoku Board is cleared



(e) Solved without input

Chapter 6

Conclusion

This study has shown that the pencil-and-paper algorithm is a feasible method to solve any Sudoku puzzles. The algorithm is also an appropriate method to find a solution faster and more efficient compared to the brute force algorithm. The proposed algorithm is able to solve such puzzles with any level of difficulties in a short period of time (less than one second).

The testing results have revealed that the performance of the pencil-and-paper algorithm is better than the brute force algorithm with respect to the computing time to solve any puzzle.

The brute force algorithm seems to be a useful method to solve any Sudoku puzzles and it can guarantee to find at least one solution. However, this algorithm is not efficient because the level of difficulties is irrelevant to the algorithm. In other words, the algorithm does not adopt intelligent strategies to solve the puzzles. This algorithm checks all possible solutions to the puzzle until a valid solution is found which is a time consuming procedure resulting an inefficient solver. As it has already stated the main advantage of using the algorithm is the ability to solve any puzzles and a solution is certainly guaranteed.

References

1. Erik Hellman, ”Android Programming-Pushing the Limits”, 1” Edition, Wiley India Pvt Ltd, 2014. ISBN-13: 978-8126547197.

2. Dawn Griffiths and David Griffiths, ”Head First Android Development”, 1” Edition, O’Reilly SPD Publishers, 2015 ISBE-13: 978-9352131341.

3. Bill Phillips, Chris Stewart and Kristin Manicano, ”Android Programming: The Big Nerd.

4. https://www.geeksforgeeks.org/sudoku-backtracking-7/.