VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi – 590 018

Submitted in partial fulfillment of requirements for the

Mobile Application Development Laboratory (18CSMP68)

Mini Project Report

On

"CHESS GAME APPLICATION"

Submitted by

Vipul Y S(4AD19CS103)

Yashas S Bharadwaj(4AD19CS104)

Under the Guidance of

Mr. Raghuram A S

Assistant Professor,

Dept. of Computer Science & Engineering

Mysuru-570028

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A T M E COLLEGE OF ENGINEERING

13th KM Stone, Mysuru-Bannur Road

Mysuru – 570028

A T M E COLLEGE OF ENGINEERING

13th KM Stone, Mysuru-Bannur Road
Mysuru – 570028.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the mini project work entitled **"CHESS GAME APPLICATION"** is a bonafide work carried out by **Vipul Y S (4AD19CS103)** and **Yashas S Bharadwaj (4AD19CS104)** in partial fulfillment for the Mobile App Development Laboratory with Mini Project prescribed by the Visvesvaraya Technological University, Belagavi during the year 2021-2022 for the sixth semester B.E. Computer Science and Engineering. The Mini project report has been approved as it satisfies the academic requirements with respect to the mini project work prescribed for the sixth semester **Mobile App Development Laboratory with Mini Project.**

Signature of the guide	Signature of the Coordinator	Signature of HOD
Mr. Raghuram A S **Assistant Professor** **Dept Of CSE, ATMECE**	**Mr. Raghuram A S** **Assistant Professor** **Dept Of CSE, ATMECE**	**Dr. Puttegowda D** **Professor & HOD** **Dept Of CSE, ATMECE**

Name of the Examiners	**Signature with date**
1._____	**1.**_____
2._____	**2.**_____

ACKNOWLEDGEMENT

We sincerely owe our gratitude to all the persons who helped and guided us in completing this mini project work.

We are thankful to **Dr. L Basavaraj, Principal, ATME College of Engineering**, **Mysuru**, for having supported us in our academic endeavors.

We are extremely thankful to **Dr. Puttegowda D, HOD, Department of Computer Science and Engineering,** for his valuable support and his timely inquiries into the progress of the work.

We express our earnest gratitude towards our co-ordinator**, Mr. Raghuram, Asst. Professor, Department of Computer Science and Engineering,** who helped us in getting things done and was always inspirational.

We are obliged to all **teaching and non-teaching staff members** of **Department of Computer Science and Engineering, Mysuru** for the valuable information provided by them in their respective fields we are grateful for their co- operation during the period of our project.

Lastly we thank almighty, our parents and friends for their constant encouragement without which this project would not be possible.

Vipul Y S (4AD19CS103)

Yashas S Bharadwaj (4AD19CS104)

ABSTRACT

In recent years, the emergence of smart phones has changed the definition of mobile phones. Phone is no longer just a communication tool, but also an essential part of the people's communication and daily life. Various applications added unlimited fun for people's lives. It is certain that the future of the network will be the mobile terminal.

In this project we will be designing and implementing a classic version of chess game with Graphical User Interface. Chess is an intellectual and mental game. Each piece on this game follows the basic rules of chess including capturing check and checkmate and draw. This involves realization of pieces and encoding them using object-oriented programming in to machining and devising algorithm for their moments. This game is played by two players attempting to take over each other's territory by acquiring the opponents' pawns and eventually defeating the king.

The main purpose of our app is to help chessmen play chess anywhere they wish to easily while needing lowest amount of storage possible for the app user. We start with a chessboard set up for the start of a game. Each player has 16 pieces. The white player starts the game all the time. At the beginning, white has 20 possible moves:

- The white player can move any pawn forward one or two positions.
- The white player can move their knight in different ways to start the game.

And then the game continues for both users who play turn by turn until any one of the player wins.

CONTENTS

INTRODUCTION

1.1 Android

Android is open source code mobile phone operating system that comes out by Google. Music player in this project is application software based on Google Android. Music is one of the best ways to relieve pressure in stressful modern society life.

The purpose of this project is to develop a player which can play the mainstream file format. To browse and query the storage space as well as operation of playing can be realised. Meanwhile, this software can play, pause and select songs with latest button and next button

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touch screen mobile devices such as smart phones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The current stable version is Android 10, released on September 3, 2019.

Android Architecture: -

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.

Android Runtime

Android runtime (ART) is the managed runtime used by applications and some system services on Android. ART and its predecessor Dalvik were originally created specifically for the Android project. ART as the runtime executes the Dalvik Executable format and Dex bytecode specification.

ART and Dalvik are compatible runtimes running Dex bytecode, so apps developed for Dalvik should work when running with ART. However, some techniques that work on Dalvik do not work on ART. For information about the most important issues, see Verifying app behaviour on the Android runtime (ART)

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.

Android UI

An Android application user interface is everything that the user can see and interact with

Installation steps of the developing environment

- Step 1: install the Java virtual machine JDK version -18
- Step 2: install the Android SDK: first download the Android SDK
- Download address: http://developer-android-com/sdk/index-html
- Input SDK tools path in the SDK location: D: \ android \ software \ android SDK– Windows and click OK

The Android environment is set up successfully.

1.2 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, This view is organized by modules to provide quick access to your project's key source files

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

- manifests: Contains the AndroidManifest.xml file.
- java: Contains the Java source code files, including JUnit test code.
- res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select Project from the Project dropdown.

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the Problems view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file

1. The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.

2. The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.

3. The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

4. The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

5. The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

6. The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

Tool windows

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

• To expand or collapse a tool window, click the tool's name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.

• To return to the current default tool window layout, click Window > Restore Default Layout or customize your default layout by clicking Window > Store Current Layout as Default.

• To show or hide the entire tool window bar, click the window icon in the bottom

left hand corner of the Android Studio window.

• To locate a specific tool window, hover over the window icon and select the tool window from the menu.

You can also use keyboard shortcuts to open tool windows. Table 1 lists the shortcuts for the most common windows.

If you want to hide all toolbars, tool windows, and editor tabs, click View > Enter Distraction Free Mode. This enables Distraction Free Mode. To exit Distraction Free Mode, click View > Exit Distraction Free Mode.

You can use Speed Search to search and filter within most tool windows in Android Studio. To use Speed Search, select the tool window and then type your search query.

Code completion

Android Studio has three types of code completion, which you can access using keyboard shortcuts.

REQUIREMENT SPECIFICATION

The requirement specification is a comprehensive description of the software and the hardware requirements required to run the project successfully.

1.1 Hardware Requirements:

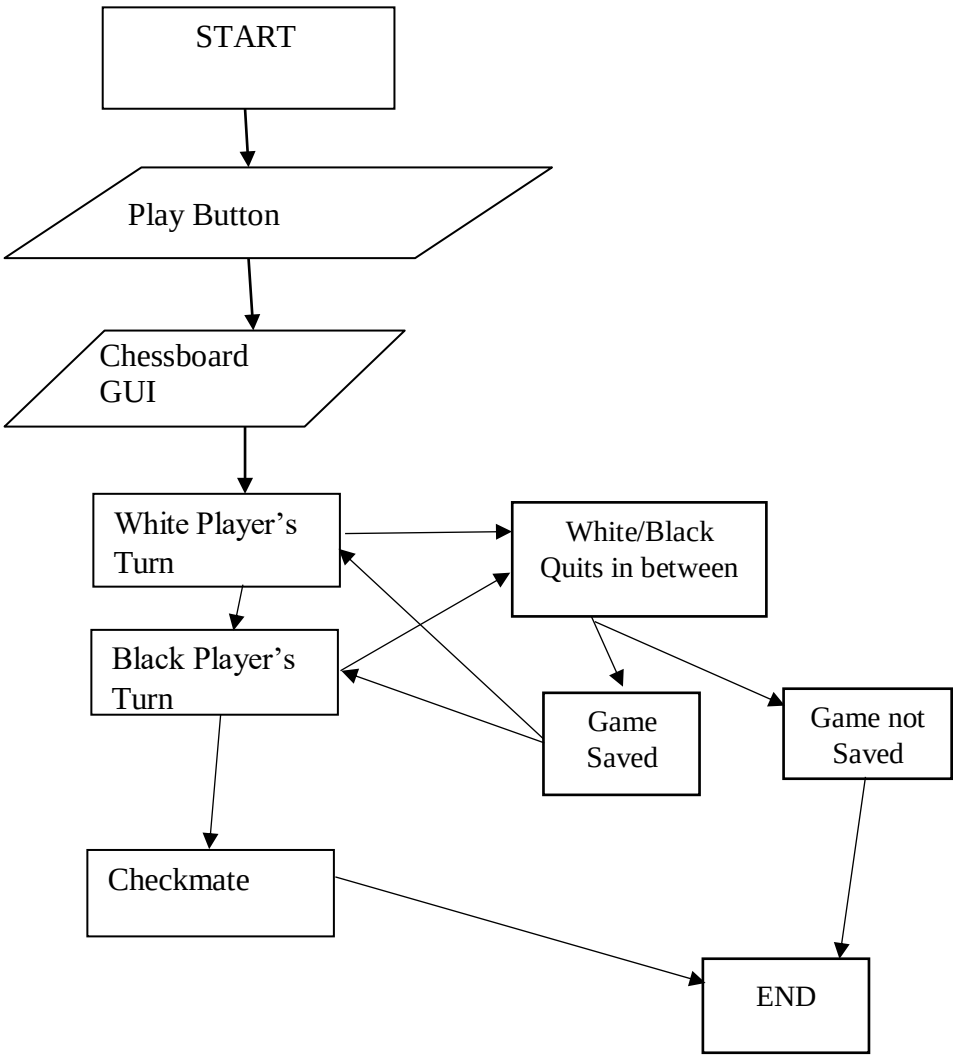
- Processor: intel/AMD processor.
- RAM: 8GB.
- Input: Keyboard/mouse.
- Display: Monitor.
- Memory: 4GB.

1.2 Software Requirements:

- Operating system: WINDOWS 10
- Language used: Xml and Java.
- Software: Android Studio.

IMPLEMENTATION

FLOWCHART



Implementation is the stage where all planned activities are put into action. Before the implementation of a project, the implementors (spearheaded by the project committee or executive) should identify their strength and weaknesses (internal forces), opportunities and threats (external forces)

.

Source Code

Android Manifest – The AndroidManifest.xml file consists of all the required permissions and declarations essential for the music player app to run smoothly. Below are the permissions that we have kept in the chess application.

MainActivity.java

```
package com.sdt944.chess;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

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

    public void btnPlayClick(View view) {
```

```
        Intent switchActivityIntent = new Intent(this, ChessBoard.class);
        startActivity(switchActivityIntent);
    }
}
```

Chessboard Class

```
package com.sdt944.chess;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;

import android.animation.ArgbEvaluator;
import android.animation.ValueAnimator;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.MotionEvent;
import android.view.View;
import android.widget.FrameLayout;

public class ChessBoard extends AppCompatActivity {
    public ConstraintLayout backgroundLayout;
    public FrameLayout boardLayout;

    public Chess chess = null;
```

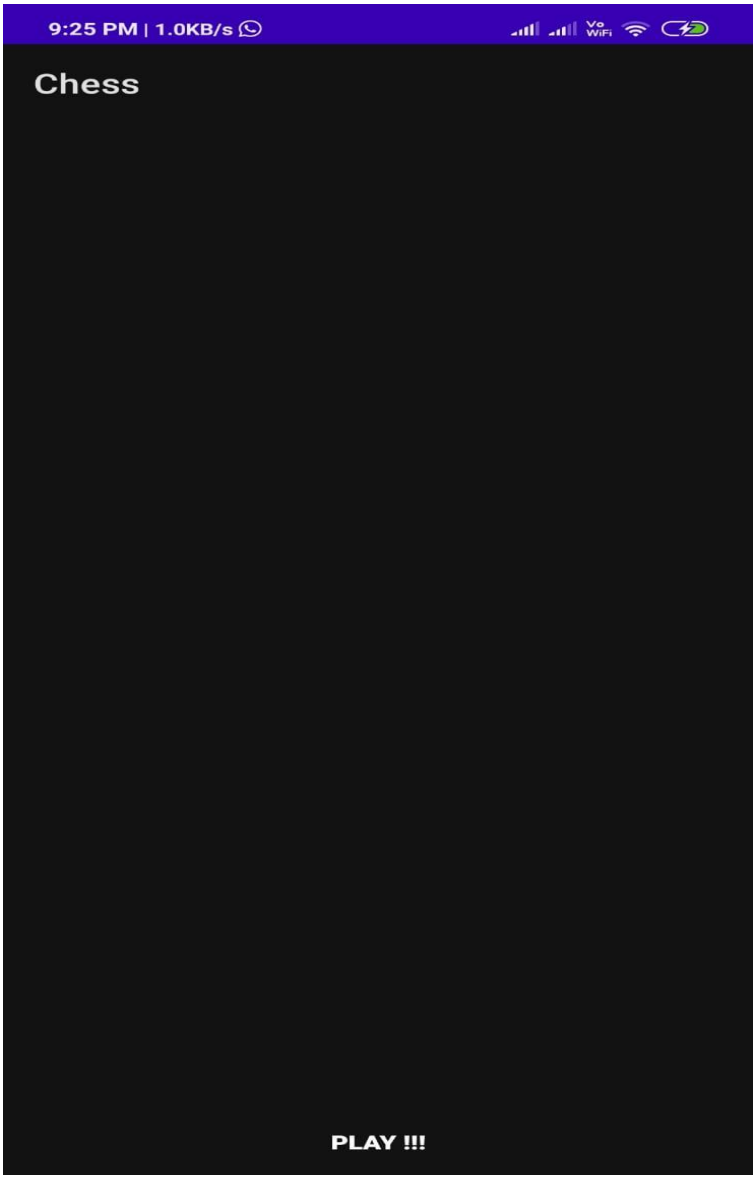
```
public int displayWidth;
public int displayHeight;
public int displayMinDimensions;

public int blackColor;
public int whiteColor;

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

    //hiding actionbar
    this.getSupportActionBar().hide();

    //change background
    backgroundLayout = findViewById(R.id.backgroundLayout);

    //initiate black and white colors
    blackColor = getResources().getColor(R.color.white);
    whiteColor = getResources().getColor(R.color.black);


    //set display params
    DisplayMetrics displayMetrics = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
    displayHeight = displayMetrics.heightPixels;
    displayWidth = displayMetrics.widthPixels;
    displayMinDimensions = Math.min(displayWidth, displayHeight);

    boardLayout = findViewById(R.id.boardLayout);
```

```
        boardLayout.getLayoutParams().height = displayMinDimensions;
        boardLayout.getLayoutParams().width = displayMinDimensions;

        if (Storage.chess == null) {
            Storage.chess = chess = new Chess(this, displayMinDimensions, boardLayout);
        } else {
            chess = Storage.chess;
            chess.changeLayout(this, displayMinDimensions, boardLayout);
        }

        findViewById(R.id.boardImage).setOnTouchListener(this::onTouch);
    }

    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() != MotionEvent.ACTION_DOWN)
            return false;
        int t = displayMinDimensions / 8;
        //chess.onBoardClick((int)event.getX() % (displayMinDimensions/8),
    (int)event.getY()%(displayMinDimensions/8));
        chess.onBoardClick(((int) event.getX()) / t, ((int) event.getY()) / t);
        return true;
    }

    public void showPromotionActivity() {
        startActivityForResult(new Intent(this, PawnPromotion.class), 1);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
```

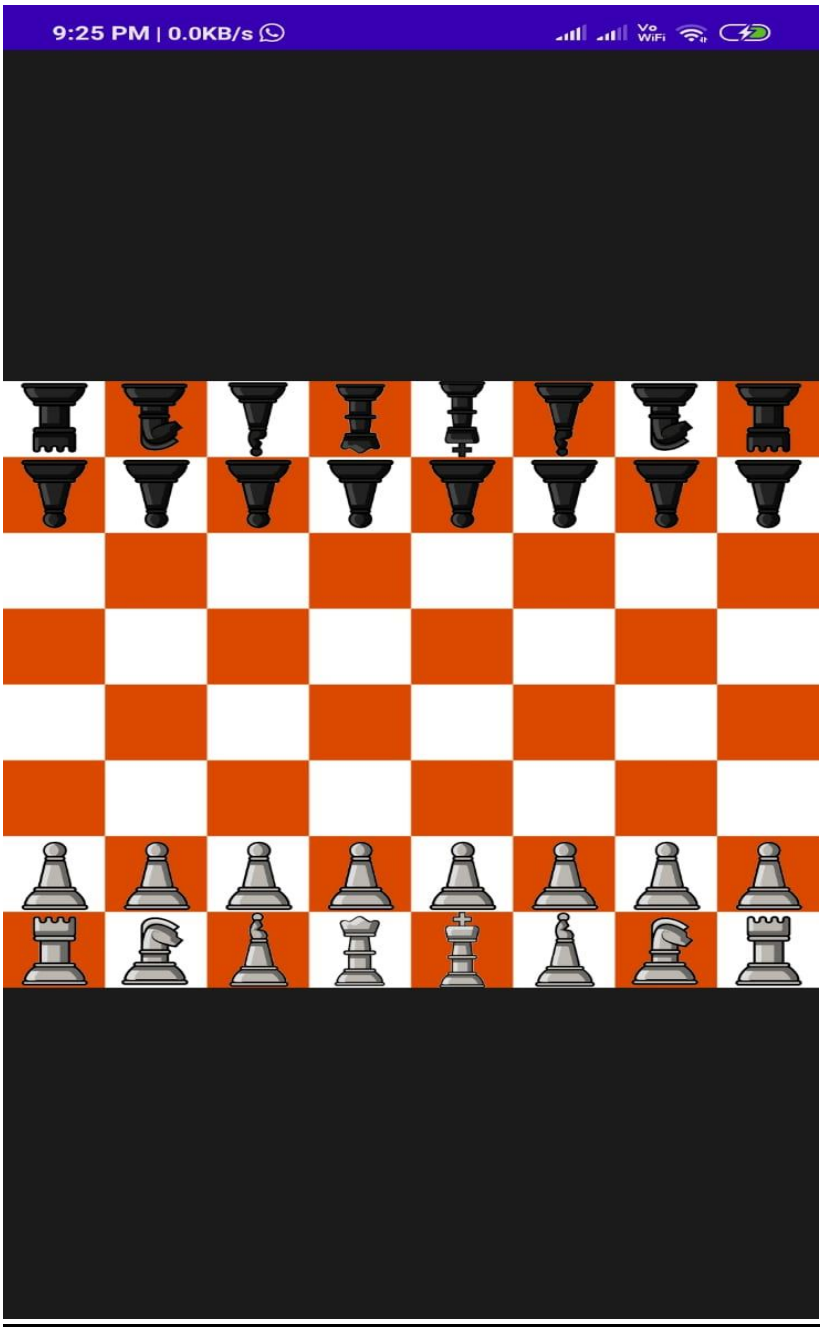
```
    //Toast.makeText(this, Storage.result.toString(), Toast.LENGTH_SHORT).show();
    chess.promotionResault(Storage.result);
  }

  @Override
  public void onBackPressed() {
    //todo : save game on the device storage
    new AlertDialog.Builder(this)
        .setMessage(getResources().getString(R.string.saveBoardPrompt))
        .setCancelable(false)
        .setPositiveButton(getResources().getString(R.string.yes), (dialog, id) -> {
          finish();
        })
        .setNegativeButton(getResources().getString(R.string.no), (dialog, id) -> {
          Storage.chess = null;
          finish();
        })
        .show();
  }

  public void animateTurnChange(Chessman.PlayerColor turn) {
    ValueAnimator colorAnimation;
    if (turn == Chessman.PlayerColor.White)
      colorAnimation = ValueAnimator.ofObject(new ArgbEvaluator(), whiteColor,
blackColor);
    else
      colorAnimation = ValueAnimator.ofObject(new ArgbEvaluator(), blackColor,
whiteColor);

        colorAnimation.setDuration(100); // milliseconds
```

```
        colorAnimation.addUpdateListener(animator ->
    backgroundLayout.setBackgroundColor((int) animator.getAnimatedValue()));
        colorAnimation.start();
      }

    }
```

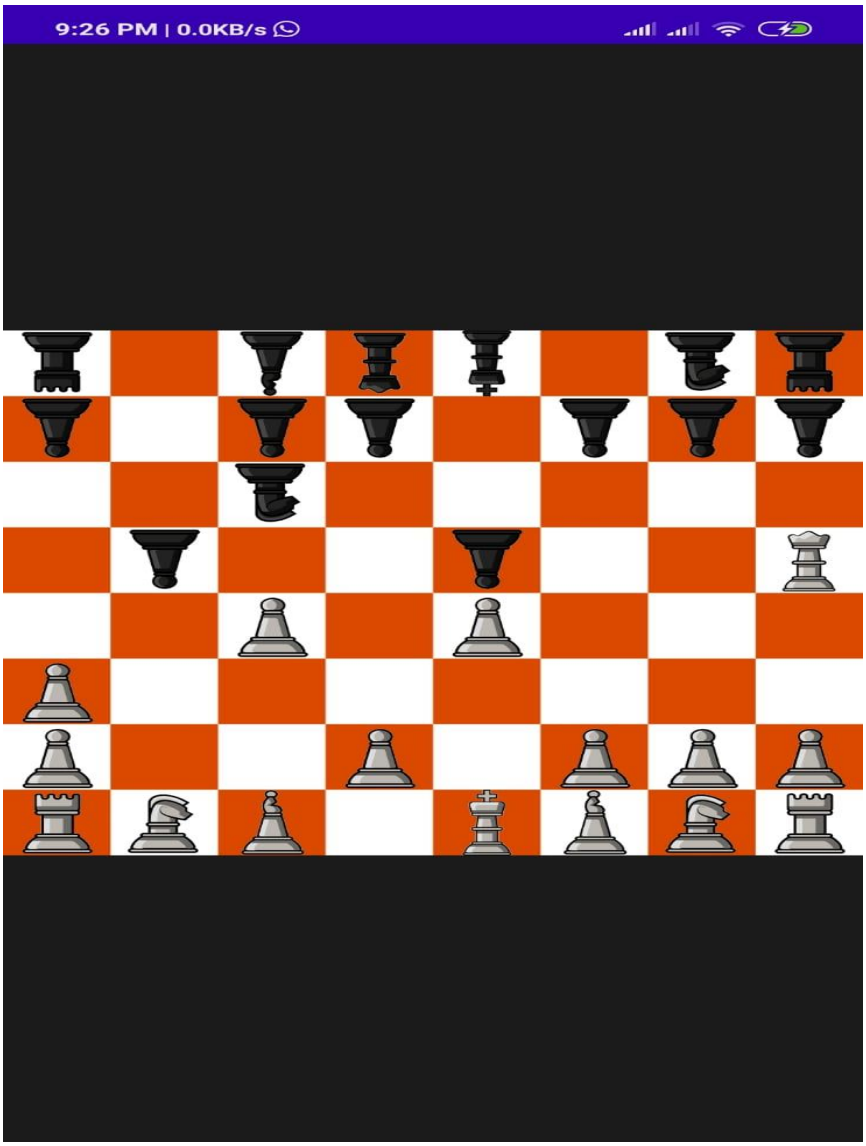
SNAPSHOTS



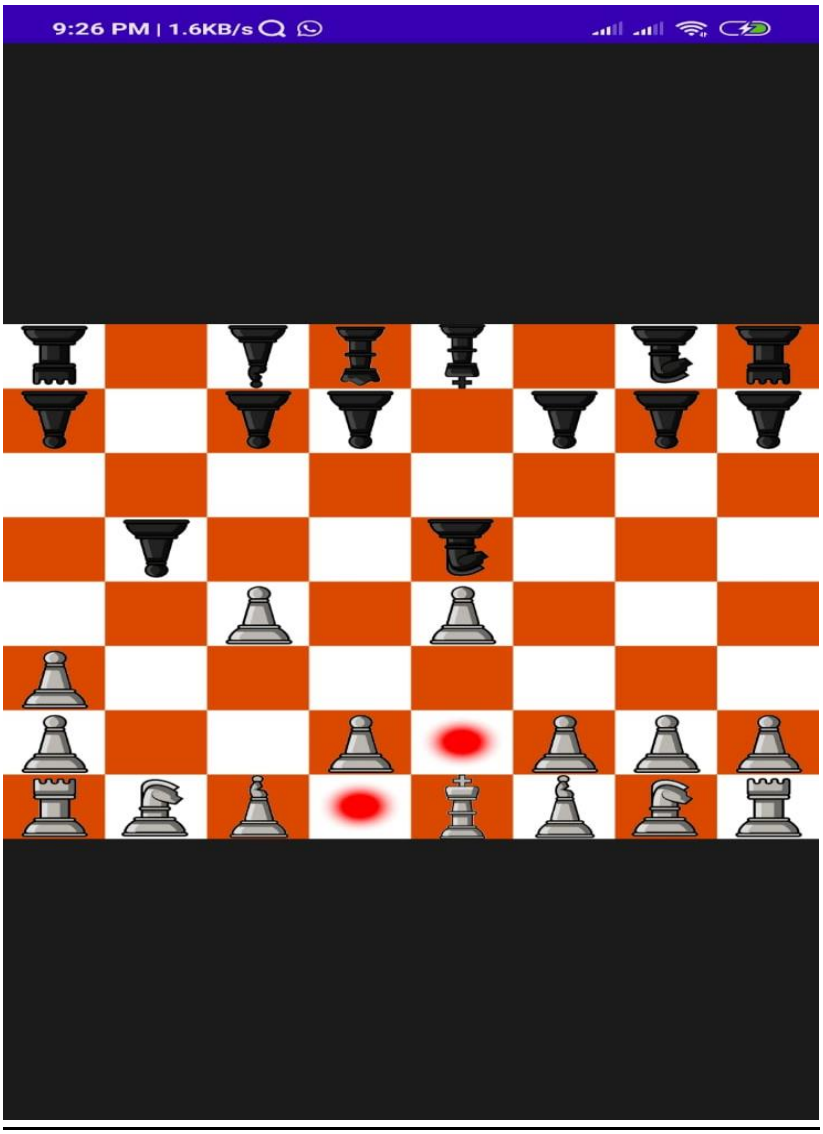
Snapshot 4.1: Launching of Application



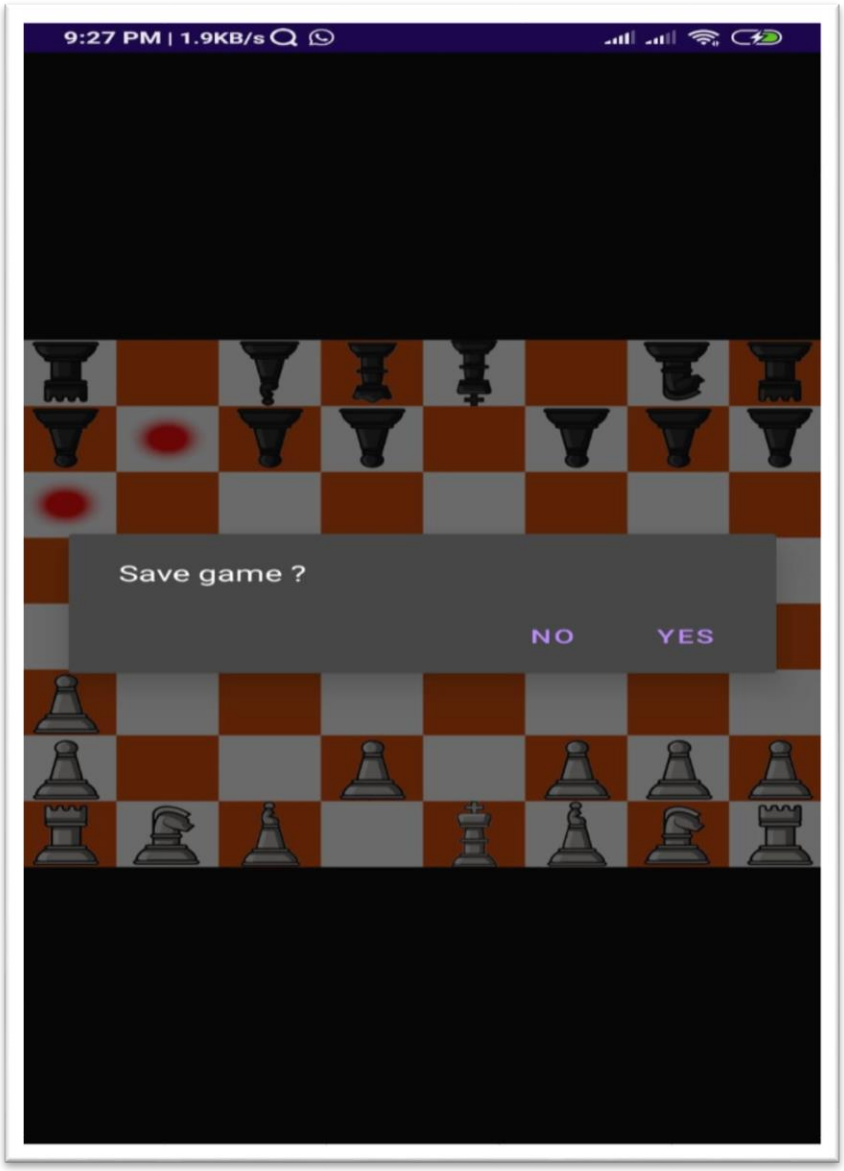
Snapshot 4.2: Starting of Game



Snapshot 4.3: In Game moment



Snapshot 4.4: Pawn Movements



Snapshot 4.5: Saving of Game

CONCLUSION

Through the development of chess application on Android platform, we get a clear understanding that the chess game is designed by keeping in mind the Human-Human Interaction principles. The application is simple and allows the user to play chess. The design of the application is classic and elegant with fewer colours and objects on it and the goal is very clear to the user: playing chess

This development implicated the popular mobile terminal development technology. This is the combination management of Java language in the open-source mobile platform based on Android Studio configuration file.

This design of chess app is based on Android system requires elaborate design of the Chess framework, by adopting ANDROID STUDIO 3.1.2 + Java language as technical support of this system, with the Android plug-in tools, and combination of Latest Android SDK version led to the comprehensive and smoothly design and development of the mobile terminal

REFERENCES

Reference Books:

1. "Android Java Programming for New generation of Mobile devices" by Zigurd Mennieks
2. "Android Cookbook" by Ian F Darwin

Website Links:

- https://en.wikipedia.org/wiki/AndroidStudio
- https://developer.android.com/studio/projects
- https://en.wikipedia.org/wiki/Android_Studio#Features
- https://www.youtube.com/
- https://www.udemy.com/
- https://app.diagrams.net/
- https://www.edureka.co/