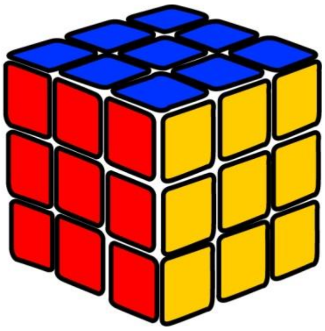
RUBIK’S CUBE (Android Game)

Submitted by: Muhammad Waqas Arif
Supervised by: Dr. Muddassar Azam Sindhu

Department of Computer Science
Quaid-i-Azam University, Islamabad
Session (2013 – 2017)



Dedicated to my parents!

ACKNOWLEDGEMENT

This thesis is about development of “Rubik’s Cube”, a game for android. It has been a challenging but great learning experience. It leads to the understanding of a layer by layer algorithm to solve the Cube.

First of all, I would like to express my gratitude to Allah Almighty, whose blessings I have seen during my time at Quaid-e-Azam University in the form of achievements, success and learning. Secondly, I am indebted to a number of people for their support and generous contribution. I especially appreciate my supervisor Dr. Muddassar Azam Sindu for guiding me throughout the process. I am also thankful to my friends and family for encouraging me to pursue my ideas and reach towards the completion of this game.

ABSTRACT

Rubik’s Cube has been an object of fascination since it has been designed. This Puzzle oriented game object gains popularity from the challenge it offers. The basic idea of the thesis is to translate the mechanism of a standard Rubik’s Cube into a virtual environment and provide an opportunity to people of all age groups to use this object virtually.

The Project progresses by studying and understanding of the logics behind the puzzle and apply that understanding for the development of a simple android game application. The Game is based on a basic Layer by Layer Algorithm and employs Unity 3D as its main tool. The understanding and writing of the program in a simple manner will provide a useful insight for the development of similar game types for the beginners.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1

Software Project Management Plan

1.1 Introduction

This document specifies the project management plan for the Android Game “Rubik’s Cube”. It elaborates the process of software development cycle and specifies deliverables such as Software Requirement Specification, Software Design Description, Software Implementation and Software Test Documentation.

1.1.1 Project Overview

This project deals with the development of an interesting android puzzle game “Rubik’s Cube”. The idea is to translate the physical 3d puzzle of Rubik’s Cube into a virtual environment.

The game requires scrambled 3*3*3 cubies with different coloured faces formed into a larger cube to be assembled in such a way that each face has same colour cubies after being randomized. The rotation is limited to x-y-z axes. The game is single player in nature which caters to different age groups.

1.1.2 Project Deliverables

Project deliverables include:

i. Software Project Management Plan (SPMP)
ii. Software requirement Specifications (SRS)
iii. Software Design Description (SDD)
iv. Software Implementation.
v. Software Test Documentation (STD)

1.2 Project Organization

This section elaborates ‘Software Process Model’ employed, ‘Roles and Responsibilities’ assigned for the completion of the process, and ‘tools and techniques’ used for the execution of the project.

1.2.1 Software Process Model

Waterfall process model will be employed for “Rubik’s Cube”, Android game. As there is no risk factor involved and no change in the requirements of the project, a straight forward model is within understanding to be good for use. Since the team only consists of one man, the team has no delay that is usually caused in waterfall model with more than one-member team. Hence, it will progress smoothly and linearly towards completion.

1.2.2 Roles and Responsibilities

The roles of requirement gathering, analysis, design, implementation and testing are all to be performed by the same person Waqas Arif.

1.2.3 Tools and Techniques

Following are the tools used for this project.

Tools and techniques	
MS Word	Used for documentation.
Project Libre	Used to make time table for project plan distributing tasks over the working months.
StarUML	Used to create UML diagrams.
Unity (v5.6)	Used for development of project
3ds Max	Used for Modeling the Cube
MonoDevelop	Used for implementation
PencilTool	Used for Interface Screens
Adobe Photoshop	Used for editing images
Adobe Illustrator	Used for making schematic Diagrams and Illustration of Steps

Table 1. 1 TOOLS AND TECHNIQUES

1.3 Project Management Plan

This section encompasses the description of tasks and deliverables that will help in managing the project.

1.3.1 Tasks

Analysis and Design tasks are divided into sub tasks:

1.3.1.1 Problem Understanding

i. **Description**
To understand the mechanism of Rubik’s Cube in a virtual environment.

ii. **Deliverables and Milestones**
Study the precedent[1] of physical Rubik’s Cube and other existing systems on Play Store.

iii. **Resources Needed**
A Physical Rubik’s Cube, an Android Device, Play Store access and Waqas Arif are the resources needed for problem understanding.

iv. **Dependencies and Constraints**
Understanding the mechanism of Rubik’s Cube is a time consuming and mind straining process. This is the only constraint faced by the project.

v. **Risks and Contingencies**
There are no risk and contingencies involved in Problem Understanding.

1.3.1.2 Software Project Management Plan

i. **Description**
This task deals with the milestones and approach for Project Management Plan.

ii. **Deliverables and Milestones**
The milestones of this task include Project Management Plan.

iii. **Resources Needed**
Two software ‘Project Libre’ and ‘Microsoft Word’ are the resources needed for this task.

iv. **Dependencies and Constraints**
There are no dependencies and constraints involved in Software Project Management Plan.

v. **Risks and Contingencies**
There are no risk and contingencies involved in Software Project Management Plan.

1.3.1.3 Analysis and Requirement

i. **Description**
The Rubik’s Cube game is analyzed, requirements are identified, SRS is developed and finalized, then Use-cases are developed and following diagrams are made:

- Use-case Diagram
- System Sequence Diagram
- Domain Model

[1] *Precedent - existing system acting as a guide to be considered in similar circumstances*

ii. **Deliverables and Milestones**
SRS is the sole milestone and deliverable for Analysis and Requirement phase.

iii. **Resources Needed**
StarUML, Microsoft Word and Waqas Arif are the resources needed.

iv. **Dependencies and Constraints**
Software Management Plan is prerequisite for the completion of this task.

v. **Risks and Contingencies**
There are no risk and contingencies involved in Analysis and Requirement phase.

1.3.1.4 System Design

i. **Description**
After the analysis is over, the design part starts and includes following diagrams;

- Sequence Diagram
- Design Class Diagram
- Package Diagram

ii. **Deliverables and Milestones**
Software Design Description is the sole milestone and deliverable.

iii. **Resources Needed**
A computer internet connection, MS Word for documentation and Star UML for formation of the above-mentioned diagrams are the resources needed.

iv. **Dependencies and Constraints**
System Analysis is prerequisite for design.

v. **Risks and Contingencies**
There are no risk and contingencies involved.

1.3.1.5 Software Implementation

i. **Description**
This task includes the Execution of the Game through the use of different software.

ii. **Deliverables and Milestones**
Software Implementation is the sole deliverable.

iii. **Resources Needed**
Unity 3d, MonoDevelop, Photoshop and 3ds Max are software needed for the implementation of this phase.

iv. **Dependencies and Constraints**
Programming of the Algorithm was Complex.

v. **Risks and Contingencies**
There are no risk and contingencies involved.

1.3.1.6 Software Test Documentation

i. **Description**

This task requires the testing of the results obtained by Software Implementation and documenting them in written form.

ii. **Deliverables and Milestones**

Documentation of the results obtained by testing

iii. **Resources Needed**

Android device for testing

iv. **Dependencies and Constraints**

Software Implementation is the only dependencies and constraint.

v. **Risks and Contingencies**

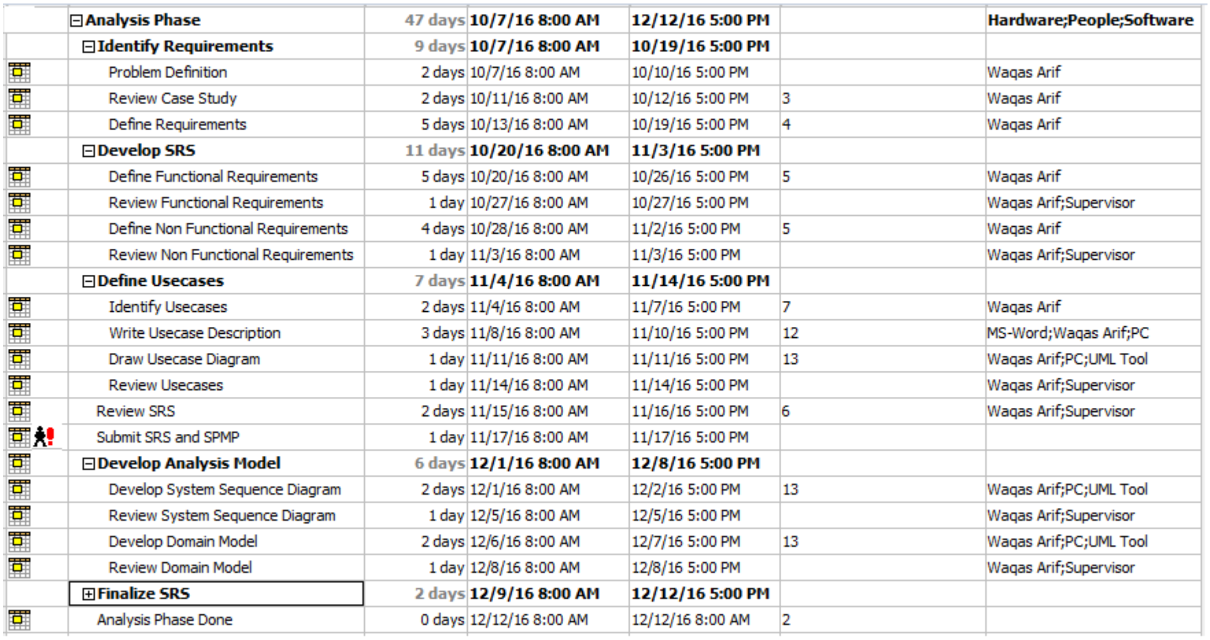
There is no risk and contingencies involved in this phase.

1.3.2 Assignments

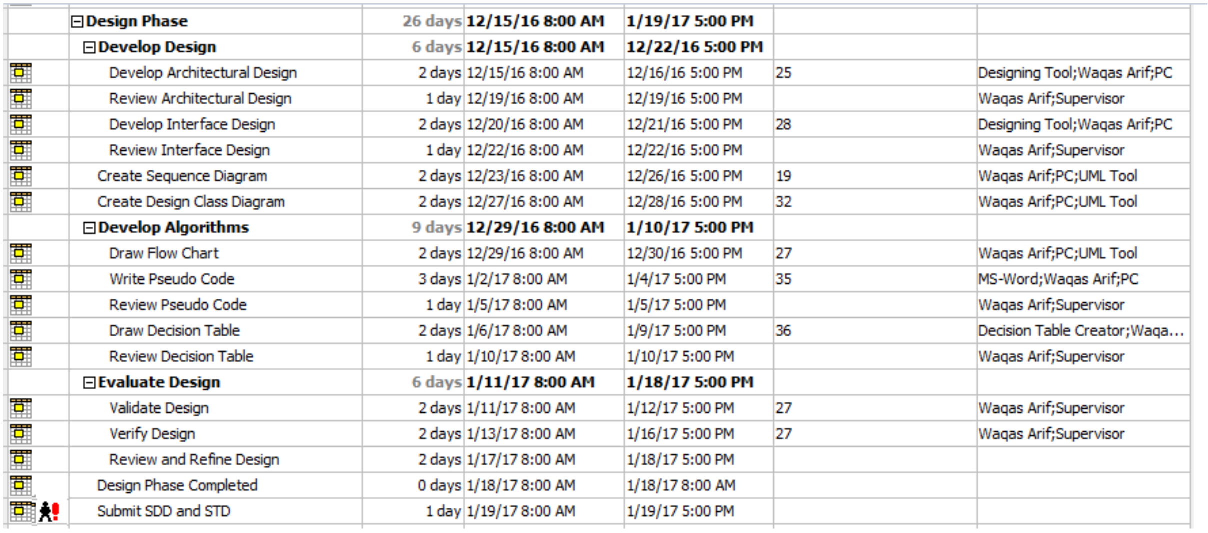
All tasks are assigned to Waqas Arif while Supervisor will advise him where needed.

1.3.3 Timetable

The time table for the project was used to keep track of the project in terms of the deadlines for different activities, the time required for them and defining the milestones in relation to time. This helps in measuring the progress of different stages of the project against time.



⊟ Analysis Phase	47 days	**10/7/16 8:00 AM**	**12/12/16 5:00 PM**		**Hardware;People;Software**
⊟ Identify Requirements	9 days	**10/7/16 8:00 AM**	**10/19/16 5:00 PM**		
Problem Definition	2 days	10/7/16 8:00 AM	10/10/16 5:00 PM		Waqas Arif
Review Case Study	2 days	10/11/16 8:00 AM	10/12/16 5:00 PM	3	Waqas Arif
Define Requirements	5 days	10/13/16 8:00 AM	10/19/16 5:00 PM	4	Waqas Arif
⊟ Develop SRS	11 days	**10/20/16 8:00 AM**	**11/3/16 5:00 PM**		
Define Functional Requirements	5 days	10/20/16 8:00 AM	10/26/16 5:00 PM	5	Waqas Arif
Review Functional Requirements	1 day	10/27/16 8:00 AM	10/27/16 5:00 PM		Waqas Arif;Supervisor
Define Non Functional Requirements	4 days	10/28/16 8:00 AM	11/2/16 5:00 PM	5	Waqas Arif
Review Non Functional Requirements	1 day	11/3/16 8:00 AM	11/3/16 5:00 PM		Waqas Arif;Supervisor
⊟ Define Usecases	7 days	**11/4/16 8:00 AM**	**11/14/16 5:00 PM**		
Identify Usecases	2 days	11/4/16 8:00 AM	11/7/16 5:00 PM	7	Waqas Arif
Write Usecase Description	3 days	11/8/16 8:00 AM	11/10/16 5:00 PM	12	MS-Word;Waqas Arif;PC
Draw Usecase Diagram	1 day	11/11/16 8:00 AM	11/11/16 5:00 PM	13	Waqas Arif;PC;UML Tool
Review Usecases	1 day	11/14/16 8:00 AM	11/14/16 5:00 PM		Waqas Arif;Supervisor
Review SRS	2 days	11/15/16 8:00 AM	11/16/16 5:00 PM	6	Waqas Arif;Supervisor
Submit SRS and SPMP	1 day	11/17/16 8:00 AM	11/17/16 5:00 PM		
⊟ Develop Analysis Model	6 days	**12/1/16 8:00 AM**	**12/8/16 5:00 PM**		
Develop System Sequence Diagram	2 days	12/1/16 8:00 AM	12/2/16 5:00 PM	13	Waqas Arif;PC;UML Tool
Review System Sequence Diagram	1 day	12/5/16 8:00 AM	12/5/16 5:00 PM		Waqas Arif;Supervisor
Develop Domain Model	2 days	12/6/16 8:00 AM	12/7/16 5:00 PM	13	Waqas Arif;PC;UML Tool
Review Domain Model	1 day	12/8/16 8:00 AM	12/8/16 5:00 PM		Waqas Arif;Supervisor
⊞ Finalize SRS	2 days	**12/9/16 8:00 AM**	**12/12/16 5:00 PM**		
Analysis Phase Done	0 days	12/12/16 8:00 AM	12/12/16 8:00 AM	2	



	⊟Design Phase	26 days	**12/15/16 8:00 AM**	**1/19/17 5:00 PM**		
	⊟Develop Design	6 days	**12/15/16 8:00 AM**	**12/22/16 5:00 PM**		
	Develop Architectural Design	2 days	12/15/16 8:00 AM	12/16/16 5:00 PM	25	Designing Tool;Waqas Arif;PC
	Review Architectural Design	1 day	12/19/16 8:00 AM	12/19/16 5:00 PM		Waqas Arif;Supervisor
	Develop Interface Design	2 days	12/20/16 8:00 AM	12/21/16 5:00 PM	28	Designing Tool;Waqas Arif;PC
	Review Interface Design	1 day	12/22/16 8:00 AM	12/22/16 5:00 PM		Waqas Arif;Supervisor
	Create Sequence Diagram	2 days	12/23/16 8:00 AM	12/26/16 5:00 PM	19	Waqas Arif;PC;UML Tool
	Create Design Class Diagram	2 days	12/27/16 8:00 AM	12/28/16 5:00 PM	32	Waqas Arif;PC;UML Tool
	⊟Develop Algorithms	9 days	**12/29/16 8:00 AM**	**1/10/17 5:00 PM**		
	Draw Flow Chart	2 days	12/29/16 8:00 AM	12/30/16 5:00 PM	27	Waqas Arif;PC;UML Tool
	Write Pseudo Code	3 days	1/2/17 8:00 AM	1/4/17 5:00 PM	35	MS-Word;Waqas Arif;PC
	Review Pseudo Code	1 day	1/5/17 8:00 AM	1/5/17 5:00 PM		Waqas Arif;Supervisor
	Draw Decision Table	2 days	1/6/17 8:00 AM	1/9/17 5:00 PM	36	Decision Table Creator;Waqa…
	Review Decision Table	1 day	1/10/17 8:00 AM	1/10/17 5:00 PM		Waqas Arif;Supervisor
	⊟Evaluate Design	6 days	**1/11/17 8:00 AM**	**1/18/17 5:00 PM**		
	Validate Design	2 days	1/11/17 8:00 AM	1/12/17 5:00 PM	27	Waqas Arif;Supervisor
	Verify Design	2 days	1/13/17 8:00 AM	1/16/17 5:00 PM	27	Waqas Arif;Supervisor
	Review and Refine Design	2 days	1/17/17 8:00 AM	1/18/17 5:00 PM		
	Design Phase Completed	0 days	1/18/17 8:00 AM	1/18/17 8:00 AM		
	Submit SDD and STD	1 day	1/19/17 8:00 AM	1/19/17 5:00 PM		

Figure 1. 1 Time Table

Chapter 2
Software Requirements Specifications

2.1 Introduction

This chapter includes the description of requirements for the development of virtual Rubik's Cube and the process of requirement gathering. This proceeds by Use Case description followed by the formation of Use Case Diagram and System Sequence Diagram.

2.1.1 Purpose

The purpose of this SRS (Software Requirements Specification) is to define the software requirements for the android based puzzle game 'Rubik's Cube'.

This game employs 26 piece arranged in 3*3*3 cubies which are arranged randomly in the form of a larger cube. The aim is to re-arrange all faces of the cubies to get same coloured faces on each face of the larger cube. The version of the game will be "Rubik's Cube 1.0". For developing such game, we require a 3d modelling software, a game engine and a programming software.

2.1.2 Scope

The project defines an Android based puzzle game known as "Rubik's Cube". The purpose of this software is to create an android game through which people of all ages can get an opportunity to solve a "Rubik's Cube" in a virtual environment, by using layer by layer algorithm. The scope of the project encompasses following points;

- User Group:
  This game is meant for users of all age groups with interest in puzzles.
- Features and Limits of the Project:
  i. The scope of game will include only one algorithm (Layer by Layer algorithm) for its development.
  ii. The game will provide timed challenge to its players and includes a best-timed player mention in 'View Best Time'
  iii. Hints will be provided to help the user solve the puzzle
  iv. Help will be provided in the form of written instructions
  v. Solve option will also be provided in case player is unable to solve the cube.
  vi. Player can also solve under different level of difficulty.
- Input and Output:
  The android game will be taking input from a touch screen of any android hand-held device and display the result of the actions on the LCD screen.
- Stake Holders
  i. Game Enthusiast
  ii. Game Developers
  iii. Me

2.1.3 Product Overview

The game consists of 26 cubies connected in a way to form a cube with 9 cubie faces to form a face of the larger cube. The unscrambling of the cube to obtain same colour of faces on all sides is the aim of the game.

Object Oriented analysis will be used for this game because it is an advanced and widely used approach in game making projects and an efficient way to handle game objects.

2.1.4 Definitions

Some of the common terminologies used in this SRS and their definitions are as follows:

Table 2. 1 DEDINITION OF TERMS

Terms	***Definitions***
Cubies	Each of the 26 smaller cubes that make up a Rubik’s Cube.
Centre Pieces	The six interconnected cubies in the middle of each side that can merely rotate around their axel. It has one outer coloured side.
Edge Pieces	The twelve cubies on the edges of the Rubik’s Cube having two different coloured sides.
Corner Pieces	The eight cubies on the corner of the Rubik's Cube having three different coloured orthogonal sides.
Face	Each of the six sides of the Rubik's Cube
Layer	The array of nine cubies that can rotate during one turn.
Turn/Move	The rotation of a Layer.
Legal Move	The rotation of a Layer at an angle of 90 degrees or a multiple of 90 degrees
Quarter turn	The rotation of 90 or -90 degrees of a Layer
Double Turn	The rotation of 180 degree of a Layer.
F (Front)	A layer facing towards you is the front layer
B (Back)	A layer facing away from you is the back layer
U (Up).	A layer which is on top is the up layer
D (Down)	A layer which is at bottom is the down layer
R (Right)	A layer which is on your right is the right layer
L (Left)	A layer which is on your left is the left layer
M (Middle)	A layer which is between the left and right layer
E (Equator)	A layer which is between the top and bottom layer
S (Standing)	A layer which is between the front and back layer

2.1.5 Game Concept

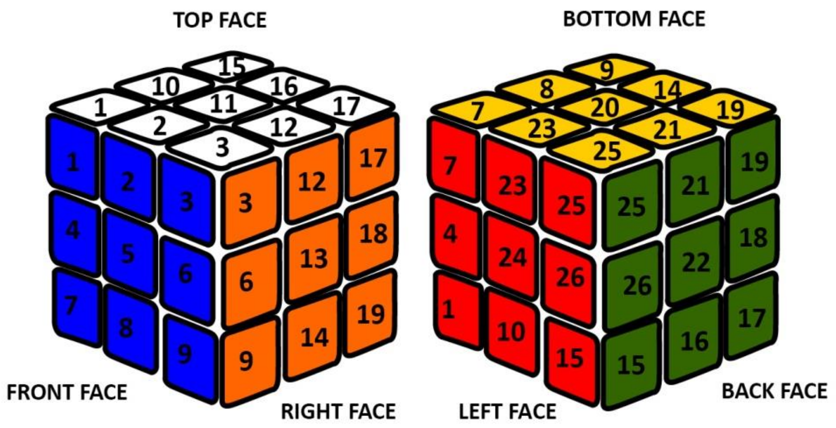


Figure 2. 1 FRONT AND BACK OF RUBIK'S CUBE

Rubik's Cube is a cube that is composed of 26 smaller cubes. These smaller cubes are often referred to as 'cubies'. - There are 6 cubies in the middle of individual faces;

that are rigidly interconnected, which means that they are connected by a six-armed spatial cross. These middle cubies merely rotate around their 'axes'. They are normally called 'centres pieces'.

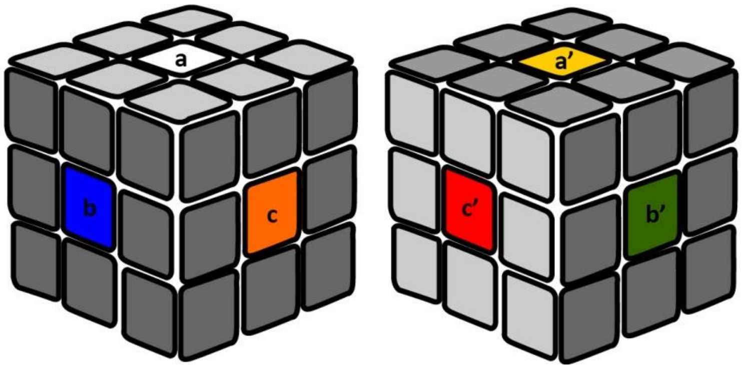


Figure 2. 2 CENTER PIECES

Each of the centre pieces has one colour on its external side, which determines the colour of each Rubik's cube face. The colours are white, red, blue, yellow, orange, and green. The white centre is opposite to yellow centre, the red centre is opposite to orange centre, and the blue centre is opposite to green centre.

There are 12 edge cubies commonly known as 'edge pieces'. Those edge piece have two faces of different colours. And the last types of cubies are eight corner cubies, or just 'corners pieces'.

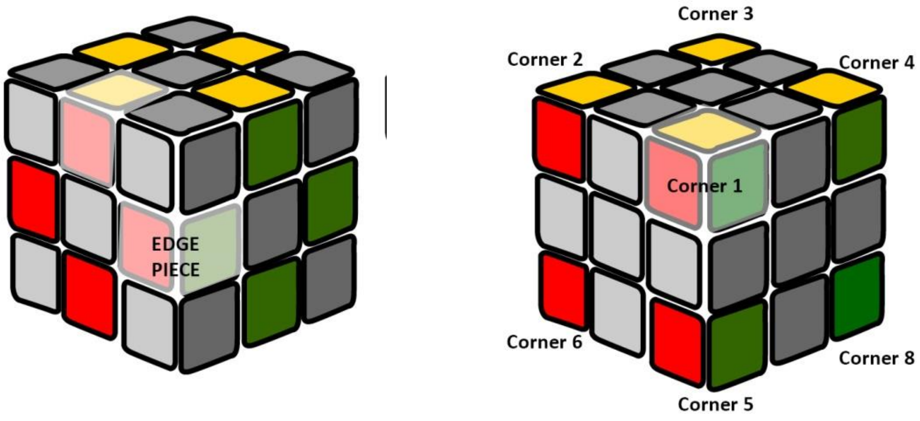


Figure 2. 3 EDGES AND CORNERS

These corners pieces have three differently-coloured faces on their mutually orthogonal sides.

The Rubik's Cube has 6 sides also known as ‘faces’. All faces can be rotated by a certain angle. This rotation is called a move (or a turn). A legal turn is considered when you rotate it with an angle of 90 degrees. The 90 and -90 degrees’ turns are called ‘quarter turns’. If you rotate a side, you rotate one third of the cube (9 cubies). This array is called a ‘layer’.

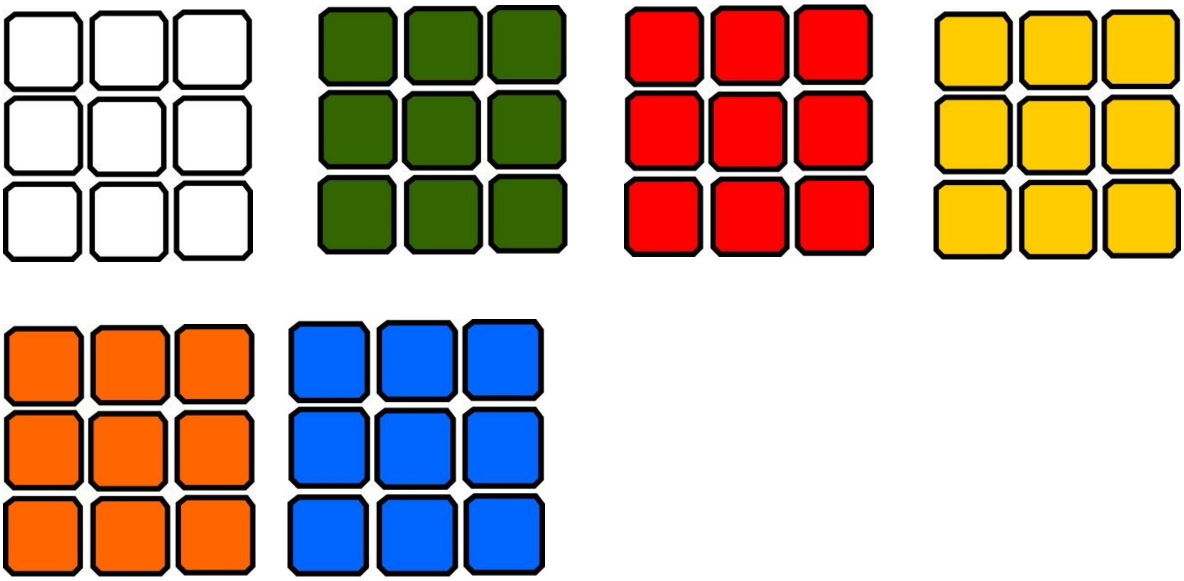


Figure 2. 4 ALL FACES OF CUBE

Having a cube in front of you, the individual layers are rotated thus:

A layer facing towards you is the front layer, and is labelled as F (Front).
A layer facing away from you is the back layer and is labelled as B (Back).
A layer which is up is the top layer and is labelled as T (Top).
A layer which is down is the bottom layer and is labelled as D (Bottom).
A layer which is on your right is the right layer and is labelled as R (Right).
A layer which is on your left is the left layer and is labelled as L (Left).
A layer which is between the left and right layer is labelled as M (Middle).
A layer which is between the top and bottom layer is labelled as E (Equator).
A layer which is between the front and back layer is labelled as S (Standing).

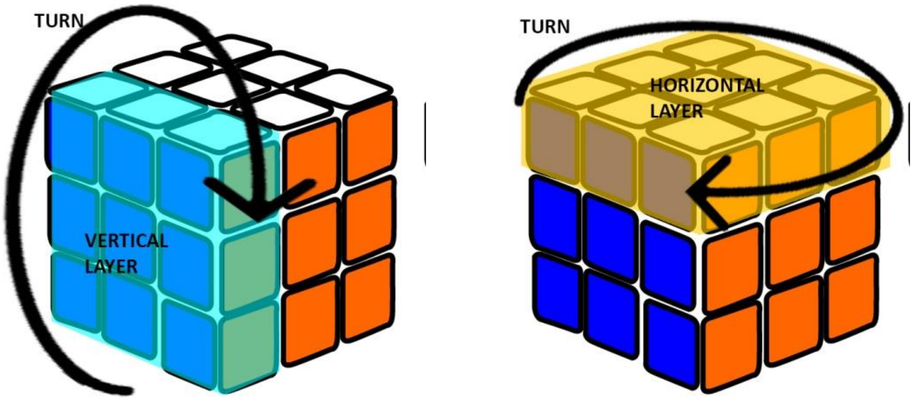


Figure 2. 5 ROTATION OF LAYERS

The move is labelled by the symbol of the layer you are turning (F, B, T, D, L, R). A symbol by itself labels a rotation of the layer. For opposite rotation, the layer symbol is followed by a '1' that is 'F1' for front layer. Turning M, E, and S layers is called 'slice turn'. For M turn the direction is bottom-up, for 'M1' top-down. For E turn the direction is right-left, for 'E1' left-right. For 'S' turn the direction is anticlockwise if seeing from front, for 'S1' clockwise. For a beginner's understanding, all displayed turns are only the quarter ones.

2.1.6 How to play

Firstly, the Rubik's Cube is shuffled (i.e. a random number of moves are applied so that no face is of same colour).

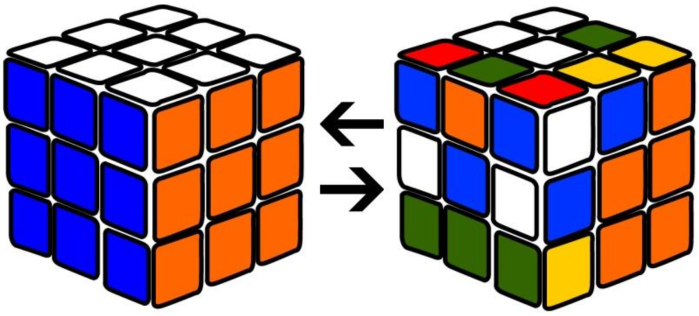


Figure 2. 6 SOLVED AND SCRAMBLED CUBE

Finally, we apply a set of sub-algorithms in a given algorithm so that each face is of same colour under a certain time constraint.

2.1.7 Algorithms Used

The purpose of making this game is to provide people an opportunity to solve the puzzle cube on their android devices. The game uses on Layer by Layer Algorithm with multiple sub-algorithms. The algorithm and its subdivisions are described below:

2.1.7.1 Layer method (beginner's method):

This method divides the cube in layers. You can solve every layer one by one applying this algorithm. The steps of this algorithm are as follows:

i. Solve the White Cross

i. Bring the white centre piece at the top with blue centre piece on the right

ii. Create a white Cross at the top;

iii. Solve the White Cross Sections in the following order; Blue, Orange, Green, Red

iv. The edge piece colours on the top should match the top white centre piece

v. Rotate the blue-white edge piece to the bottom and rotate it until it is directly under the blue centre piece

vi. Now, rotate the right layer twice so that the blue white edge piece is on the top

vii. If the white face is not with the white centre and blue face is not with the blue centre then apply this algorithm; (R1, T, F, T1)

viii. Repeat the steps from v to vii for all the colours in above mentioned order until you achieve a white cross at the top

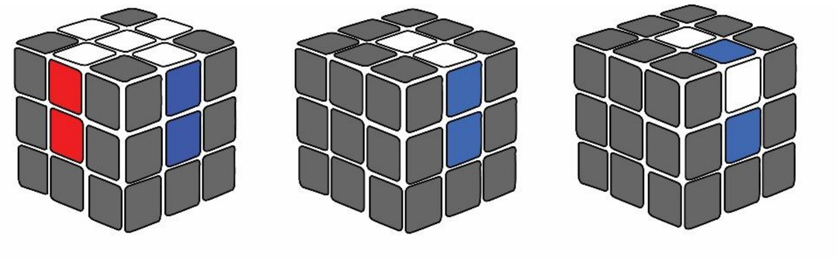


Figure 2. 7 SOLVING WHITE CROSS

ii. Solve the White Corner

i. Locate a Corner Piece with one white face

ii. If it is on the top layer bring it to the bottom layer

iii. Rotate the bottom layer until the color of the other two faces of the corner piece match with the color of the relevant center pieces.

iv. There would be three cases;

<u>CASE 1</u>: Apply the algorithm (F1, D1, F)

<u>CASE 2</u>: Apply the algorithm (R1, D, R)

<u>CASE 3</u>: Apply the algorithm (R1, D1, R, F1, D, D, F)

Repeat this for each corner piece having one white face.

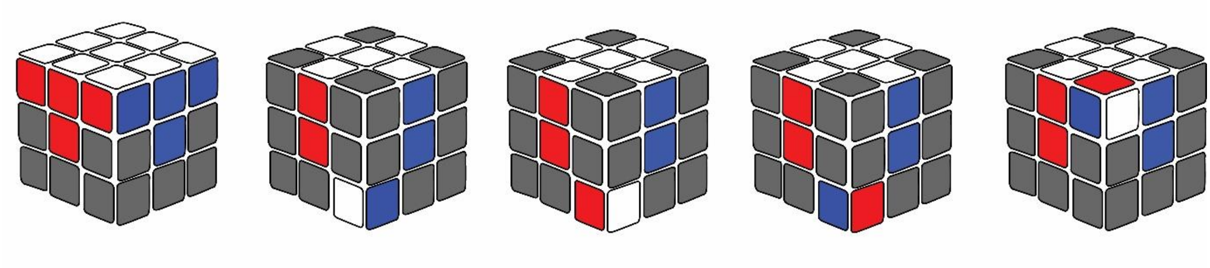


Figure 2. 8 SOLVING WHITE CORNERS

iii. Solve the Second Layer

i. Invert the cube with white center piece facing downwards.

ii. Find the edge on the top layer (other than yellow) whose face matching with the relevant center piece.

iii. There would be two cases

<u>CASE 1</u>: Apply the algorithm to move to left edge (T1, L, T, L1, T, F1, T1, F)

<u>CASE 2</u>: Apply the algorithm to move to right edge (T, R, T1, R1, T1, F, T, F1)

iv. Repeat these steps until you have solved the second layer

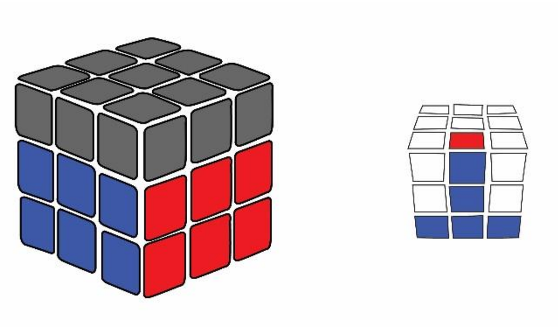


Figure 2. 9 SOLVING MIDDLE LAYER

iv. Solving the Yellow Cross:

i. There are four cases

<u>CASE 1</u>: You have accidently solved the yellow cross

<u>CASE 2</u>: If you have only centre piece yellow on the top. Apply the algorithm (F1, R, T, R1, T1, F)

<u>CASE 3</u>: If we get a yellow line, make its orientation horizontal by moving the top layer Apply the algorithm (F1, R, T, R1, T1, F)

<u>CASE 4</u>: If you get an inverted yellow L, rotate it to the top left Apply the algorithm (F1, T, R, T1, R1, F)

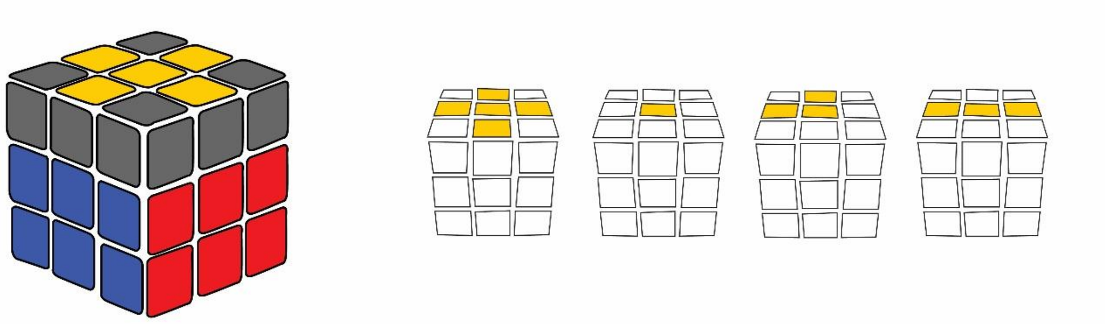


Figure 2. 10 SOLVING YELLOW CROSS

v. Solving the Yellow Face:

i. If any of the three situations are met; Apply the algorithm (R, T, R1, T, R, T, T, R1)

ii. If it is not solved match the situations shown, try again until you get done.

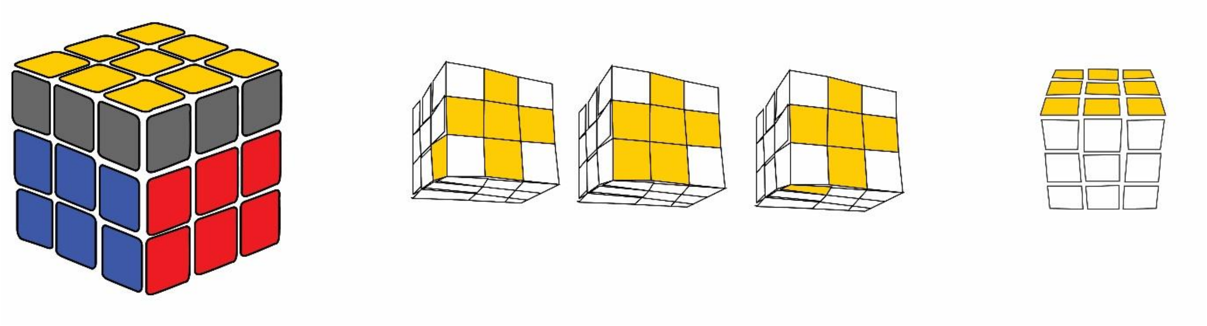


Figure 2. 11 SOLVING YELLOW PHASE

vi. Solving the Top Layer

i. Look that the top two corner faces of the same face have same color for at least one face; Apply this logarithm, Orient the that face to the back layer, and apply the algorithm
(R1, F1, R1, B, B, R, F, R1, B, B, R, R, T1)

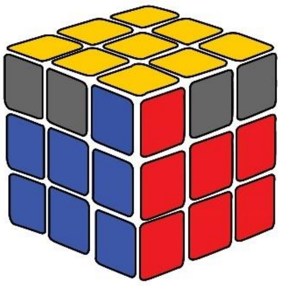


Figure 2. 12 SOLVING TOP LAYER 01

ii. Look that the top two corner faces of the same face have same color for all faces; There are two cases
<u>CASE 1</u>: Apply the algorithm
(F1, F1, T, L1, R1, F1, F1, L, R, T, F1, F1)
<u>CASE 2</u>: Apply the algorithm
(F1, F1, T1, L1, R1, F, F, L, R, T1, F, F)

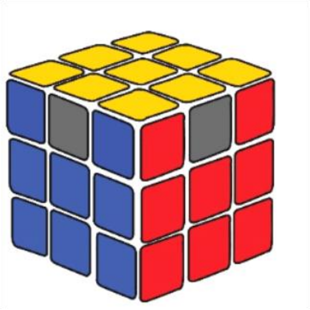


Figure 2. 13 SOLVING TOP LAYER 02

2.2 Specific Requirements

This section contains the requirements for this system in terms of their description and features.

2.2.1 External Interface Requirements

The external interfaces are defined below in detail:

2.2.1.1 User Interface

The user interface is perhaps the most important part of an application; because it certainly is the most visible. So, interface of this game will be user friendly, just like a real 3D Rubik's Cube. It will include a visual GUI to show the Main Menu and Sub Menus. Buttons will be used to display different options on the menus. Backgrounds will be used to make the interface more attractive. Once the game is in playing mode, everything a player needs will be clearly visible on the screen and easily accessible including the timer. This will make the handling of the game easier and understandable. Player will use touch input to control the game. The screen will be updated after each move the player make.

2.2.1.2 Hardware Interface

Since the Rubik's Cube is Android based game application, therefore there will only be touch screen input. It is for Android Systems of KitKat v4.4.2 or higher. There should be at least 1GB RAM memory space for running of this game and 30 MB storage free space for installation. A Wi-Fi connection will be needed for downloading and installing the game application.

2.2.1.3 Software Interface

This game will run only on an Android Operating Systems of version KitKat v4.4.2 or higher. It will be implemented on Unity3D game engine (version 5.3). C# scripting language will be used as a development tool for implementation. 3ds Max will be used to create the model of Rubik's Cube and Photoshop will be used to create textures.

2.2.1.4 Communication Protocols

No communication protocols are required because the game software is standalone.

2.2.2 Software Product Features

Some of the software product features are defined below in the form of Use Cases

UC-1: Make Turn

This is the scenario where the player starts from the Main menu. The player selects Play Option. Then Selects the Free Play Option or any Level the player wants to play from the Levels Menu. The game starts and the Rubik's cube gets scrambled according to the level. The timer starts and the user can now play the game by swiping the different layers. The game end as soon as the player has solved the cube and the Congratulation message is displayed on the screen.

Table 2. 2 USE CASE FOR MAKE TURN

UC-1: Make Turn	
Primary actor	Player.
Stakeholder	Player will be able to Make the Turn.
Pre-condition	Player should be in Main Menu.
Post-condition	Game has been played successfully and Congratulation message is being displayed.
Main Success Scenario	1. Player selects Play option. 2. System displays the Play menu option. 3. Player select Free Play option or any other level of his choice. 4. System start the game 5. System shuffles the cube according to the level selected. 6. System starts the Timer. 7. Player makes a new turn on the screen, until he solves the cube. 8. System stops the Timer 9. System saves the best time. 10. System displays Congratulations!! on screen.
Alternate Flows	1a. Player selects option other than the play 1. Player selects back option from that menu. 2. System returns to main menu. 3. Player selects Play option again. 7a. Player selects solve option. 1. System will solve the cube 2. System will display solved message with play again option 3. Player selects the play again and tries again to solve the cube. 8-10a. System does not work appropriately 1. Player selects the back option. 2. Player selects free play option again. 3. Player completes the game again.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

UC-2: Change Settings

This is the scenario where the player starts from the Play Mode. The player selects Settings option. A list of various settings is Displayed. Then the player set his preferred settings and hit save. The settings get saved and player returns to the game.

Table 2. 3 USE CASE FOR CHANGE SETTINGS

UC-2: Change Settings	
Primary actor	Player.
Stakeholder & Interests	Player will be able to change game settings.
Pre-condition	Player should be in Play Mode.
Post-condition	New settings have taken effect.
Main Success Scenario	1. Player selects Settings option. 2. System pauses the game 3. System displays the list of settings. 4. Player changes the settings 5. Player selects the save option 6. System saves the new settings 7. System closes the settings display. 8. System resumes the game. 9. New settings take effect.
Alternate Flows	5a. Player selects the back option 1. System closes the settings and resumes the game. 2. Player select the settings option again. 3. System displays the settings. 4. Player changes the settings and saves them. 9a. new setting does not take effect 1. Player closes the game and Start it again 2. System shows the main menu. 3. Player navigate to play mode and selects Settings option. 4. System displays the settings. 5. Player changes the settings and saves it. 6. New settings take effect.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

UC-3: Restart Game

This is the scenario where the game is running in Play mode. The player completes the game on its own or with the solve option and the message is being displayed with the restart option on it. The Player selects the restart option and System restarts the game.

Table 2. 4 USE CASE FOR RESTART GAME

UC-3: Restart Game	
Primary actor	Player.
Stakeholder & Interests	Player will be able to restart the game.
Pre-condition	Game should be completed.
Post-condition	Game has successfully been restarted.
Main Success Scenario	1. Player completes the game. 2. System displays the relevant message with restart option on it. 3. Player selects the restart option. 4. System restart the game 5. System reshuffles the cube according to the level selected. 6. System starts the Timer.
Alternate Flows	3a Player selects the restart option and noting happens. 1. Player closes the Play Mode by selecting back option. 2. Player selects the same level as before.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

UC-4: Take Hint:

This option is used by the player when he is playing the game. The hint option is selected on the game menu which provide him access to the hint stored in it. The hint flashes before him for a few seconds in order to help him play further.

Table 2. 5 USE CASE FOR TAKE HINT

UC- 4: Take Hint	
Primary actor	Player.
Stakeholder & Interests	Player will be able to take the hint.
Pre-condition	Player should be playing the game.
Post-condition	Player has successfully taken the hint.
Main Success Scenario	1. Player is playing the game. 2. Player selects the Hint Option. 3. Game displays the Hint for a few seconds.
Alternate Flows	2a. Payer selects the hint option and nothing happens. 1. Player tries to press again. 2. The Hint gets displayed.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

UC-5: Solve Cube:

The player while playing the game selects the option to solve the cube. The cube gets simulated according to a calculated solution and he can see a solved cube on the screen and a message get displayed saying cube has been solved.

Table 2. 6 USE CASE FOR SOLVE CUBE

UC- 5: Solve Cube	
Primary actor	Player.
Stakeholder & Interests	Player wants to simulate the solution.
Pre-condition	Player should be playing the game.
Post-condition	Simulation successfully completes, the cube has been solved and solved message is displayed.
Main Success Scenario	1. Player is playing the game. 2. Player selects the solve option 3. System calculates the solution. 4. System simulates the calculated solution. 5. Cube gets successfully solved 6. System displays the solved message on the screen.
Alternate Flows	2a. Player selects the solve option and nothing happens. 1. Player select the back option and go to previous menu 2. Player select the level again 3. System starts the game. 4. Player select solve 5. The cube gets solved.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

UC-6: Undo Turn:

The player while playing the game selects the option to undo the turn. The system will undo the last turn.

Table 2. 7 USE CASE FOR UNDO TURN

UC- 6: Undo Turn	
Primary actor	Player.
Stakeholder & Interests	Player wants to undo the last turn.
Pre-condition	Player should be playing the game.
Post-condition	Player has undone the last turn.
Main Success Scenario	1. Player is playing the game. 2. Player selects the undo option 3. System undo the last turn.
Alternate Flows	2a. Player presses the undo option and noting happens. 1. Player waits for the a few seconds 2. Player selects the option again 3. System undo the last turn.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

UC-7: View Instructions:

This is the scenario where the player goes to the main menu to get instructions about solving the cube. He obtains help in the form of written instructions or algorithm.

Table 2. 8 USE CASE FOR VIEW INSTRUCTIONS

UC- 7: View Instructions	
Primary actor	Player.
Stakeholder & Interests	Player will be able to view the instructions.
Pre-condition	Main Menu should be opened.
Post-condition	The instructions have been displayed successfully.
Main Success Scenario	1. Player selects ‘Instructions’ option. 2. System displays a document explaining different steps of algorithm.
Alternate Flows	1a. Game crashes 1. The player will forcefully close the application. 2. Restart the application. 3. Then try to select Instructions option again. 2a. Steps does not get displayed. 1. Player goes back to the main menu 2. Player again selects the instruction option.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

UC-8: View Best Time:

This is the scenario in which the best-timed performances on a device are shown to the player in order to know his ranking compared to previous time according to different levels. The Best three timed performances are displayed on the screen.

Table 2. 9 USE CASE FOR VIEW BEST TIME

UC-8: View Best Time	
Primary actor	Player.
Stakeholder & Interests	Player will be able to view his list of best time
Pre-condition	Main Menu should be opened.
Post-condition	Best Times has been viewed successfully.
Main Success Scenario	1. Player selects Best Time option. 2. System displays Best Time list.
Alternate Flows	1a. Game crashes 1. The Player will forcefully close the application. 2. The Player will go into the main menu 3. The Player will select Best Time option. 2a. List does not display.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Few times.

UC- 9: Reset Best Time:

This is the situation when the player wants to open the previous menu other than the main menu.

Table 2. 10 USE CASE FOR RESET BEST TIME

UC- 9: Reset Best Time	
Primary actor	Player.
Stakeholder & Interests	Player will be able to Reset all the Best Time stored up till now.
Pre-condition	Player should be in the Best Time Menu
Post-condition	Player successfully clears all the previously stored Best Time.
Main Success Scenario	1. Player selects the reset option. 2. System clears all the Best Time and return them to 0.
Alternate Flows	1a. Game crashes 1. Player will forcefully close the application. 2. Player will restart the application. 3. Player Navigates to Best Time menu. 4. Player selects reset option 5. System resets every stored time to zero.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

UC-10: Exit Game:

This is the option provided on the main menu. This option provides the user an opportunity to exit the game.

Table 2. 11 USE CASE FOR EXIT GAME

UC- 10: Exit Game	
Primary actor	Player.
Stakeholder & Interests	Player will be able to Exit the game.
Pre-condition	Main Menu should be opened.
Post-condition	Game has been exit successfully.
Main Success Scenario	1. Player selects the Exit option. 2. Game exits application.
Alternate Flows	1a. Game crashes 1. The player will forcefully close the application.
Special Requirements	None.
Technology	Touch input and LCD Display output.
Frequency	Multiple times.

2.2.3 Use Case Diagram:

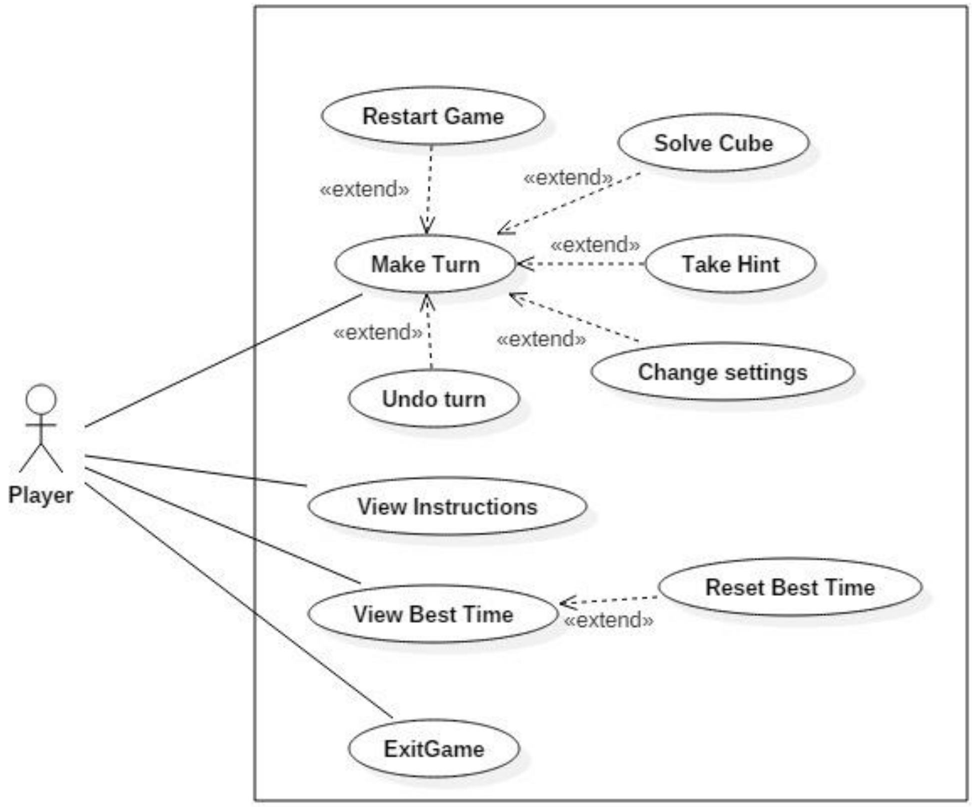


Figure 2. 1 USE CASE DIAGRAM

2.2.4 Software System Attributes

Some of the system attributes are defined below;

2.2.4.1 Reliability

The system should be reliable and should fulfil its purpose. It should never be able to crash unless or until some system or device failure occurs. The code should be without errors. The system will be 90% reliable.

2.2.4.2 Availability

The system is always available on Google Play Store of Android systems. It will require internet connection for downloading and must not take much time to launch the game application.

2.2.4.3 Security

There are no such security constraints.

2.2.4.4 Maintainability

The system has the ability to easily adapt to new features and updates. All upgrades can quickly and safely be performed with in minimum time. Files will be well commented. Authorship will be mentioned. Object oriented design will be used. Camel Case naming convention will be used.

2.2.4.5 Portability

Since the system will be designed in Unity3D, which provides portability to many operating systems, so this game supports other Android systems as well (v4.4 or higher). It is 95% portable.

2.2.4.6 Performance

The system must not take initial load time of more than 10 seconds in more than 90% of the times. Changing screens will require very little computation and thus will occur in very less time taking not more than a couple of seconds. As a whole the system will be 90% performance efficient. The frame rate will be more than 30 frames per second.

2.2.5 Database Requirements

There is no such need of a database so no requirements are specified.

2.2.6 Domain Model

It is a conceptual model of all the topics related to a specific problem. It describes the entities their attributes, roles and relationships of the entities, and the constraints that specify the problem domain. Domain Model for Rubik’s Cube is given below;

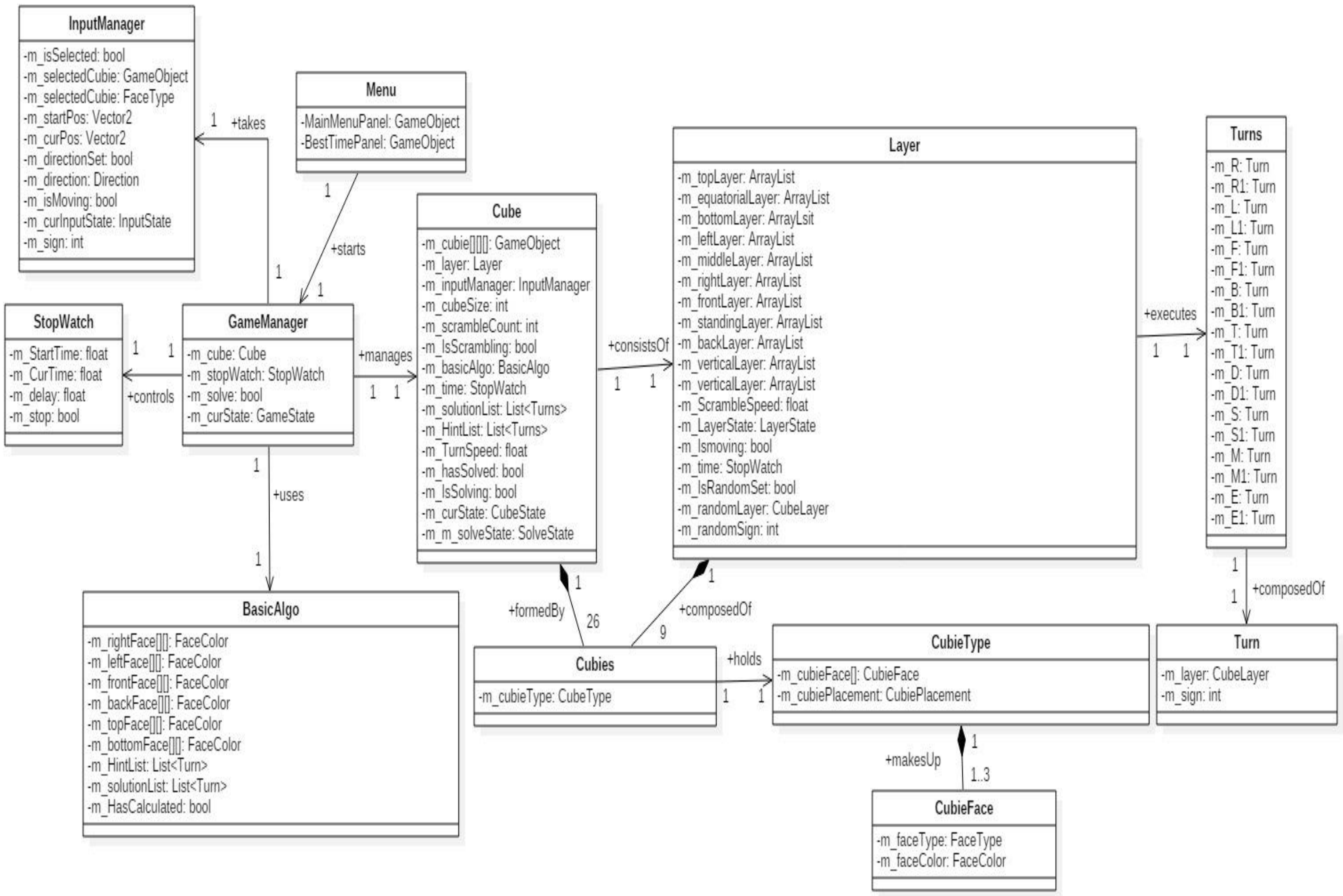


Figure 2. 14 DOMAIN MODEL

Chapter 3
Software Design Description

3.1 Introduction

The Design Overview section deals with the description of Architecture of the system and product overview.

3.1.1 Design Overview

The Software Design Document provides the documentation that helps in designing the software. This document will convey the design philosophy and architecture of Rubik’s Cube game application. It will also define how the developer intends to implement the application to a function according to the software requirements previously submitted. This document contains the Architectural design, component diagram, sequence diagram and design class diagram.

3.1.2 Purpose

The purpose of the Software Design Document is to provide a detailed description of the design of Rubik’s Cube Game. It will aid in constructing a system that would be efficient. It will also provide necessary information for the designing of software and system to be built.

3.2 Requirements Traceability Matrix

Requirement Traceability Matrix or RTM captures all requirements proposed by the client or development team and their traceability in a single document delivered at the conclusion of the life-cycle. In other words, it is a document that maps and traces user requirement with test cases. The main purpose of Requirement Traceability Matrix is to see that all test cases are covered so that no functionality should remain missing during the testing.

Table 3. 1 REQUIREMENT TRACEABILITY MATRIX

Requirements/ Use Case	Domain Model	Class Diagram	Sequence Diagram	Test Case
UC -1	Figure 2.14	Figure 3.4	Figure 3.2	TC -1
UC -2	Figure 2.14	Figure 3.4	Figure 3.2	TC -2
UC -3	Figure 2.14	Figure 3.4	Figure 3.2	TC -3
UC -4	Figure 2.14	Figure 3.4	Figure 3.2	TC -5
UC -5	Figure 2.14	Figure 3.4	Figure 3.2	TC -6
UC -6	Figure 2.14	Figure 3.4	Figure 3.2	TC -4
UC -7	Figure 2.14	Figure 3.4	Figure 3.3	TC -7
UC -8	Figure 2.14	Figure 3.4	Figure 3.3	TC -8
UC -9	Figure 2.14	Figure 3.4	Figure 3.3	TC -9
UC- 10	Figure 2.14	Figure 3.4	Figure 3.3	TC- 10

3.3 System Architectural Design

System Architecture Diagram is used to represent the components of a system and the interaction between them. Interaction between components of our system is shown in the form of a diagram.

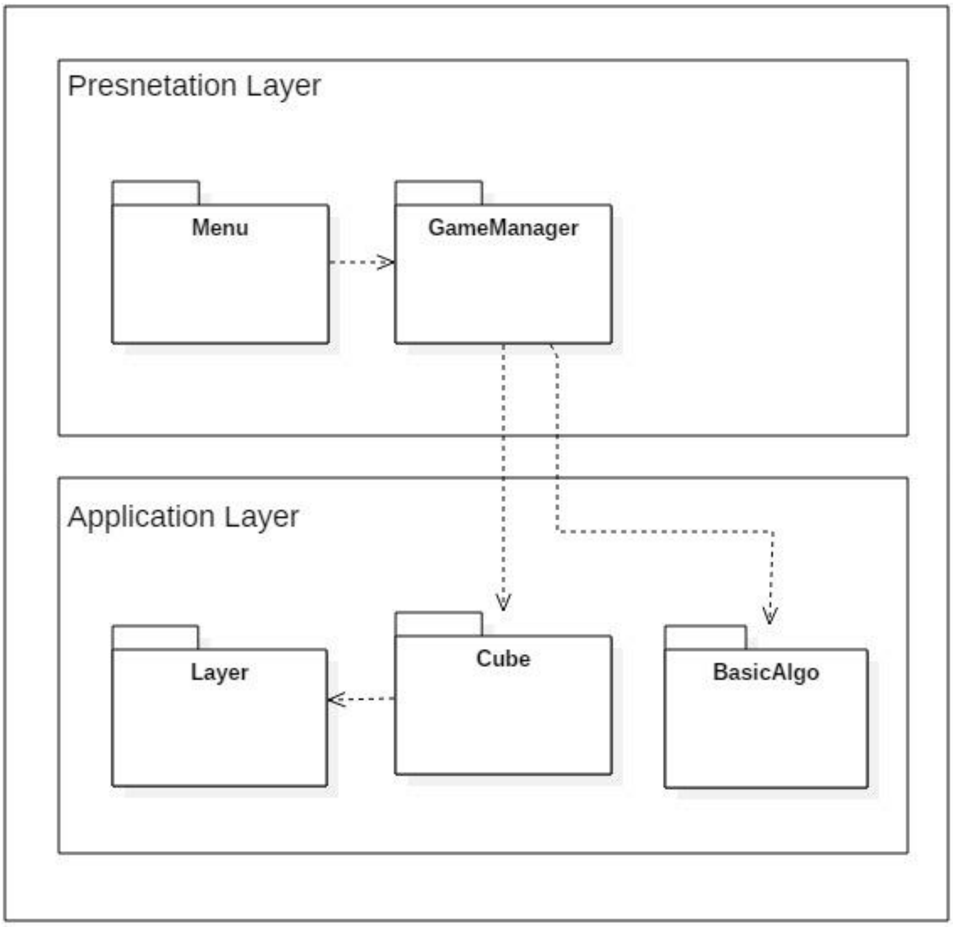


Figure 3. 1 SYSTEM ARCHITECTURE DIAGRAM

3.3.1 Chosen System Architecture

Chosen system architecture is 2-tier because there is no database. There are following two layers;

- Presentation Layer
- Application Layer

3.3.2 System Interface Description

System interface describes the flow of resources. It is the logical characteristics of each interface between the software product and the hardware components of the system.

3.4 Detailed Description of Components

Here is the detailed description of components of system architecture. It includes UML diagrams to show the organization and writing of the components of a system.

3.4.1 Player

Player can swipe the layer of a cube. Player can also rotate the cube. He can take Hint where he stuck. He can try to solve the puzzle in best given time. He can interact with the User Interface and features of the game.

3.4.2 Cube

A Cube is a 3x3x3 3D cube, consists of different colors. It is made up of different layers. It interacts with the player when he takes a turn.

3.4.3 Layer

A single Layer consists of nine cubies. There are six layers in one cube. Player can swipe a layer to take a turn for solving the puzzle.

3.4.4 Game Manager

It controls all the other components and their interactions.

3.4.5 Algorithm

Algorithm is used to give hints by predicting the next turn on the cube.

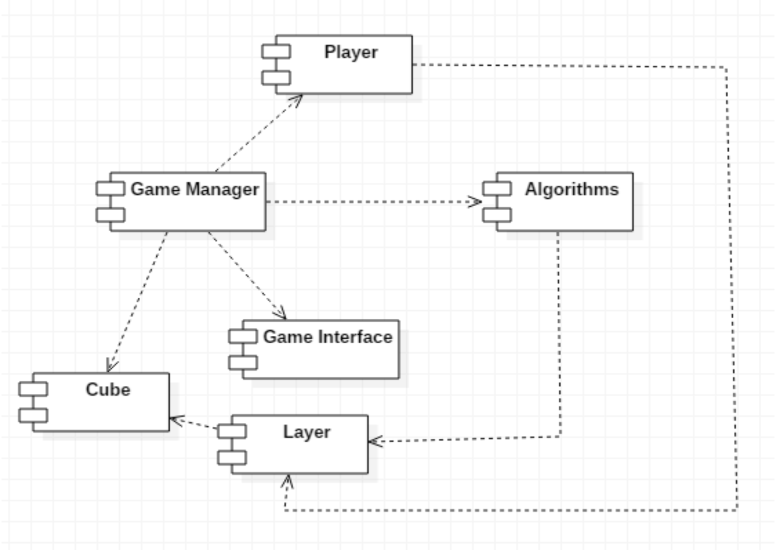


Figure 3. 2 COMPONENT DIAGRAM

3.5 Sequence Diagram

Sequence diagram depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the system. We identify how the functions are carried out in the system. So, this can help in making the different functionalities of the game. We carried out sequence diagram in each use case.

Sequence diagram for this project is shown below:

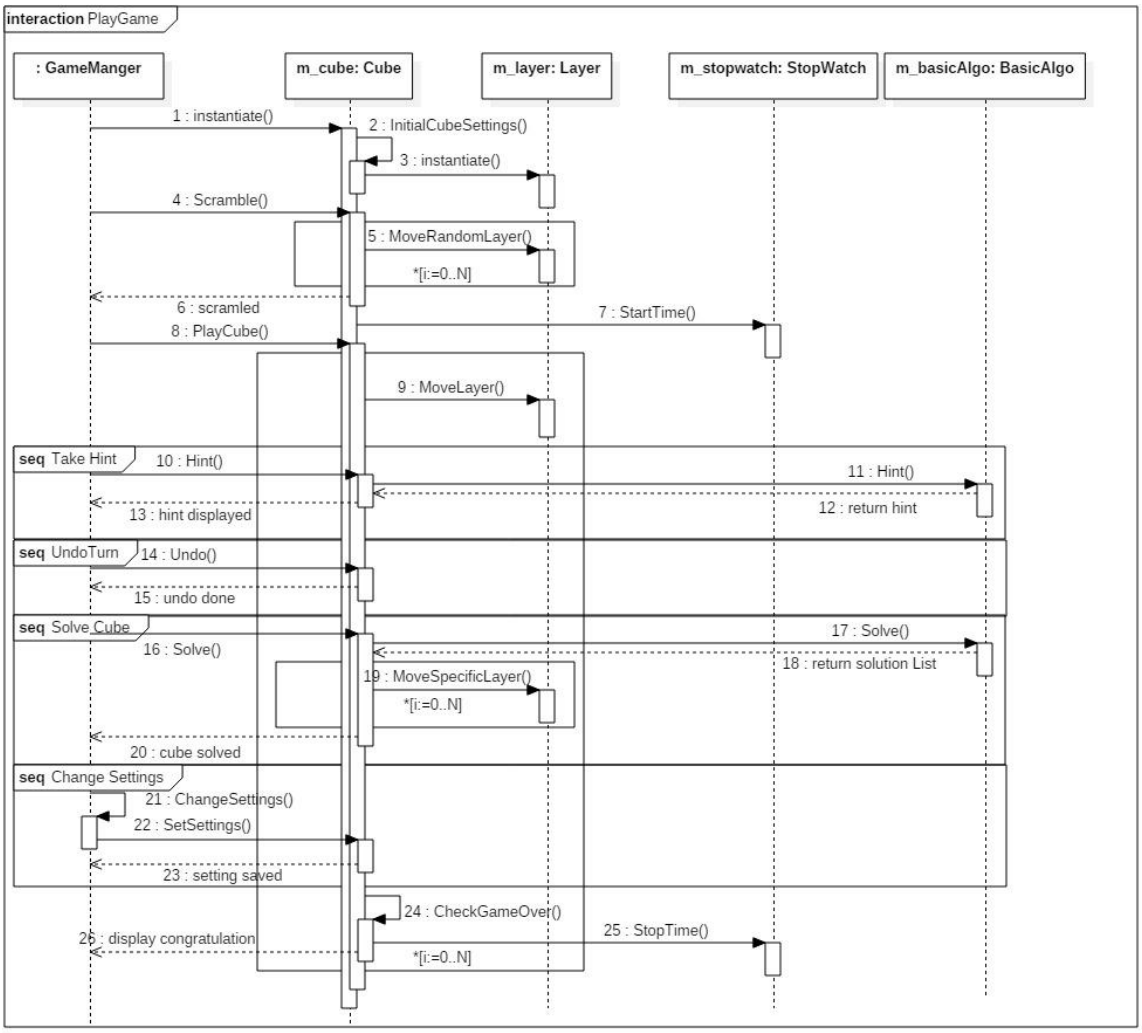


Figure 3. 3 SEQUENCE DIAGRAM 1

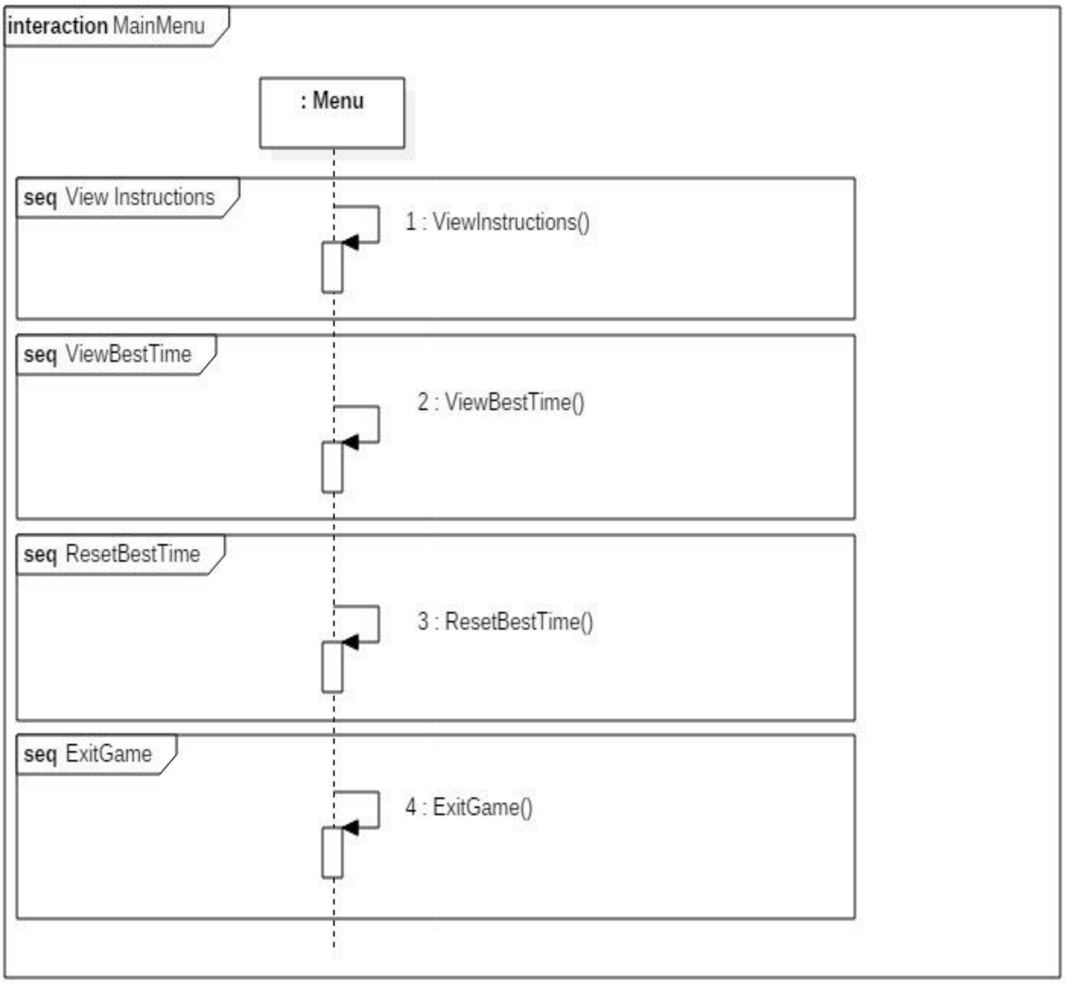


Figure 3. 4 SEQUENCE DIAGRAM 2

3.6 Design Class Diagram

Design class diagram is given below:

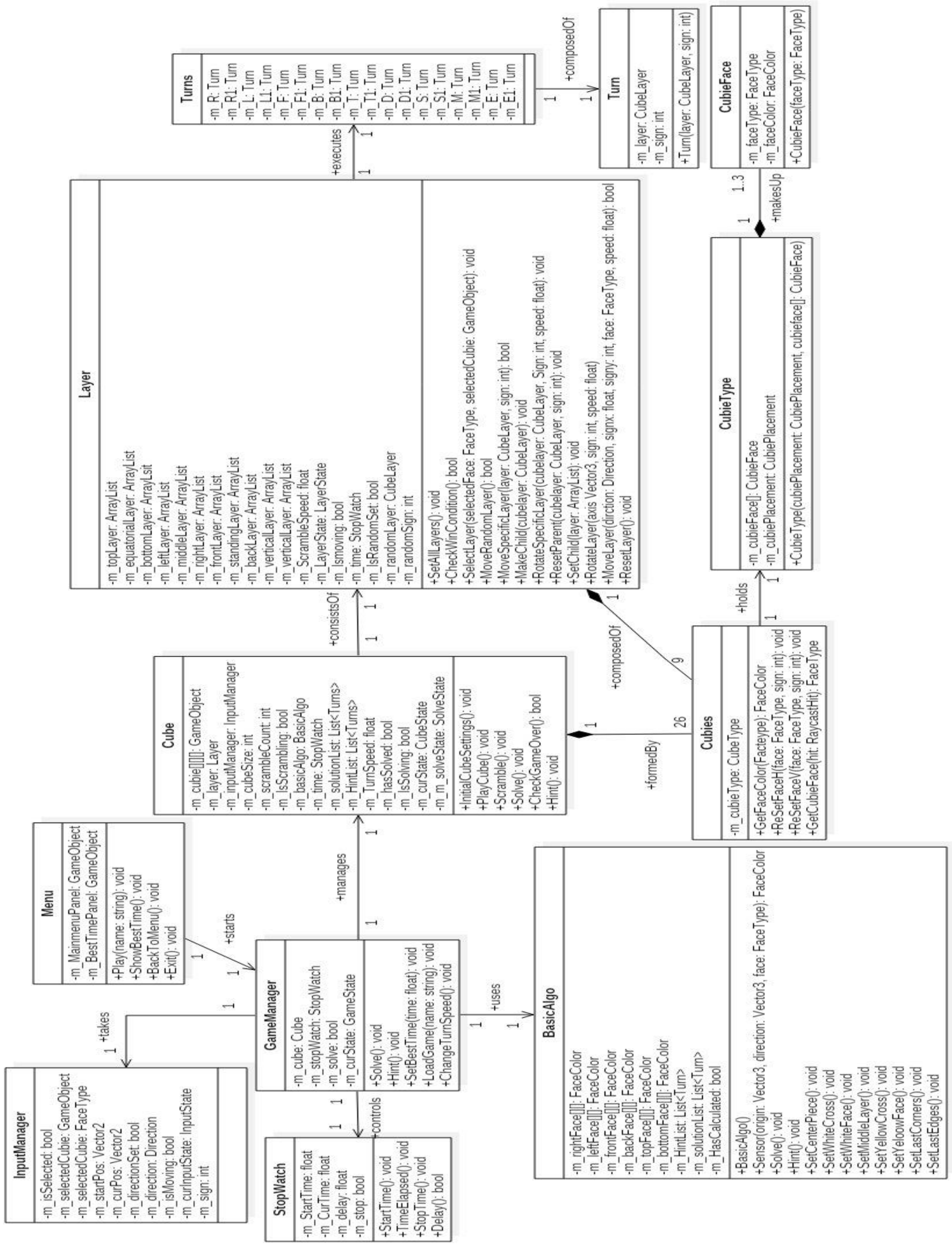


Figure 3. 5 DESIGN CLASS DIAGRAM

3.7 User Interface Design

User interface is the logical characteristics of each interface between the software product and its users. In this section user interface of this game is discussed.

3.7.1 Description of the User Interface

In Rubik's Cube game, Player can interact with game by using touch screen interface of Android device. When Player clicks on game icon, a main menu will appear. Main menu has four buttons Play, Instructions, Best Time and Exit Game. When player will click on Play, game mode appears in which there is a cube, a settings icon and a back icon, solve icon and hint icon and a slider to adjust the speed of the turn.

3.7.2 Screen Images

Following are few screen images of game.

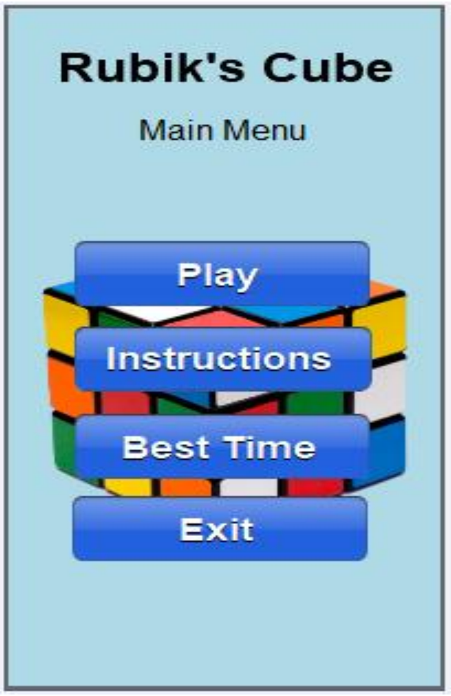


Figure 3. 6 MainMenu

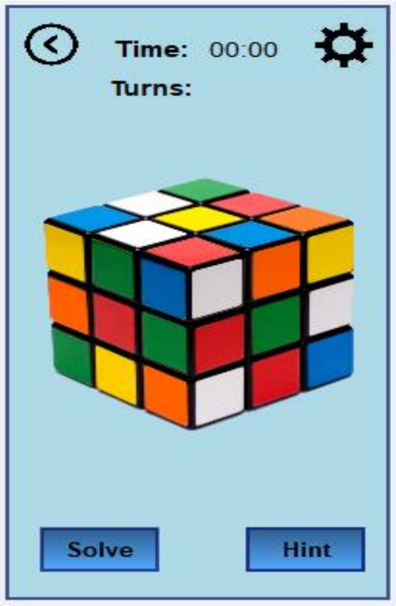


Figure 3. 7 PLAY MODE

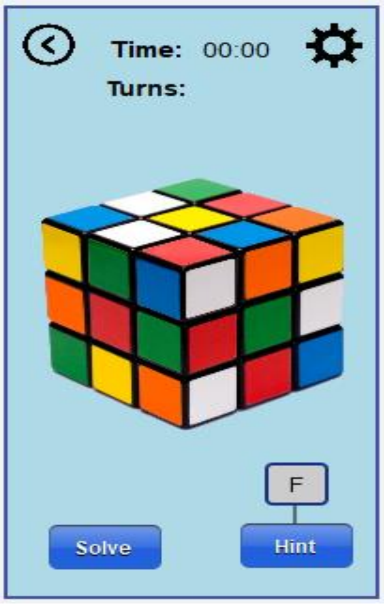
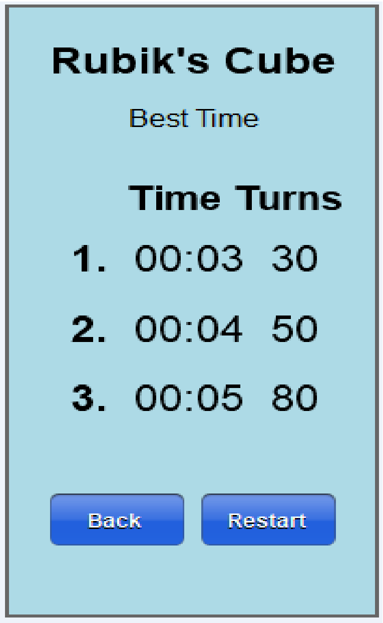


Figure 3. 8 HINT


Rubik's Cube

Best Time

	Time	Turns
1.	00:03	30
2.	00:04	50
3.	00:05	80

Figure 3. 10 VIEW BEST TIME

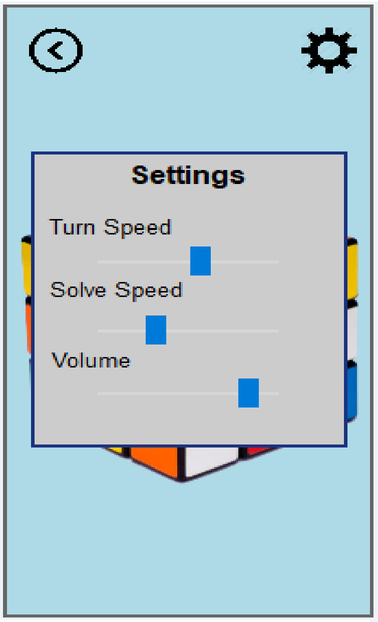
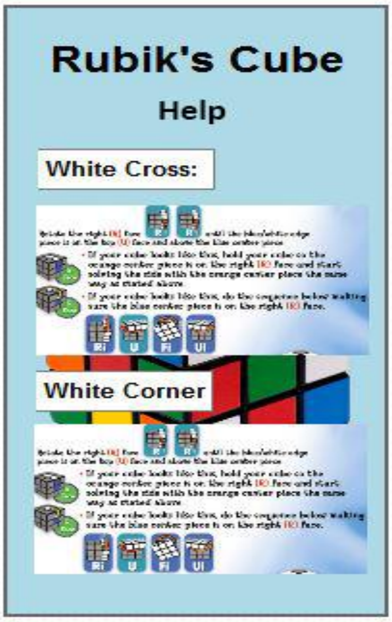


Figure 3. 9 CHANGE SETTINGS



Rubik's Cube

Help

White Cross:

Rotate the right (R) face until the blue/white edge piece is at the top (U) face and above the blue center piece

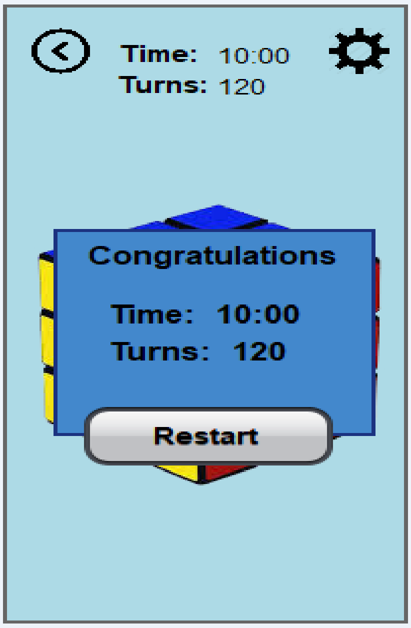
- If your cube looks like this, hold your cube so the orange center piece is on the right (R) face and start solving the side with the orange center piece the same way as started above.
- If your cube looks like this, do the sequence below making sure the blue center piece is on the right (R) face.

White Corner

Rotate the right (R) face until the blue/white edge piece is at the top (U) face and above the blue center piece

- If your cube looks like this, hold your cube so the orange center piece is on the right (R) face and start solving the side with the orange center piece the same way as started above.
- If your cube looks like this, do the sequence below making sure the blue center piece is on the right (R) face.

Figure 3. 12 VIEW INSTRUCTIONS



Time: 10:00
Turns: 120

Congratulations

Time: 10:00
Turns: 120

Figure 3. 11 WIN MESSAGE

Chapter 4
Software Implementation Document

4.1 Introduction

In this chapter, the framework and language selection for the project is provided. It also includes screenshots for the implementation of the game.

4.1.1 Framework Selection

The framework used for this game is developed in Unity3D v5.6. It supports Android Software Development Kit (SDK). This development tool is preferred because it is a best game developing engine. It is portable and cross platform which means that the same code, developed via Unity engine, can be ported on many platforms with minimal modifications. It runs on Android Operating System.

4.1.2 Language Selection

- **C#**
  The language chosen for this game is C# because it is used in Unity3D v5.6. It is a managed language which means that it manages memory for you.

4.1.3 Operating System

This game will run only on an Android Operating Systems of version KitKat v4.4.2 or higher.

4.1.4 Algorithms

The purpose of making this game is to provide an opportunity to people to solve the Rubik’s cube using Algorithm.

4.1.4.1 Layer method (beginner's method):

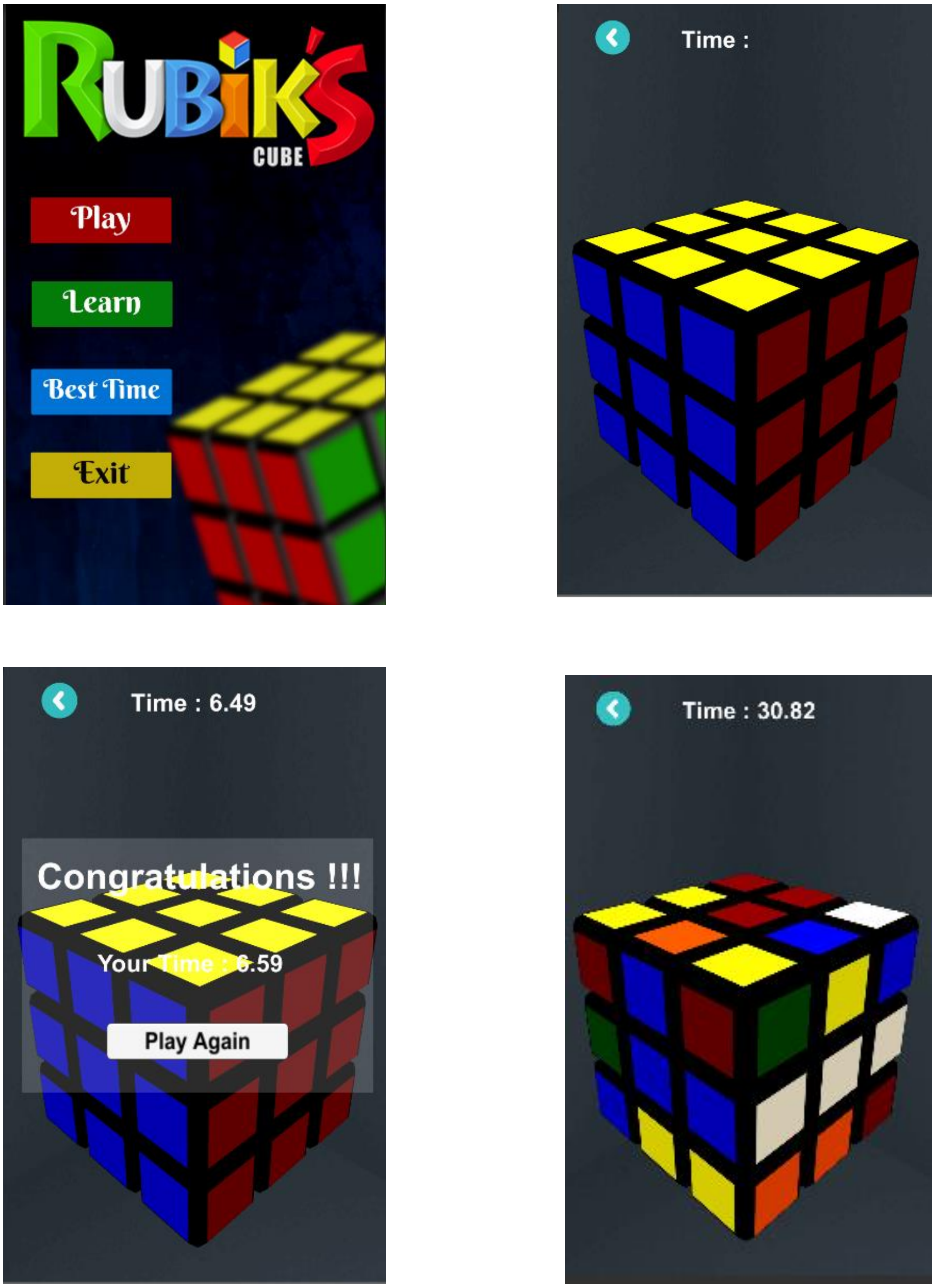
This method divides the cube in layers. You can solve every layer one by one applying this algorithm. The steps of this algorithm are as follows:

i. Solve the White Cross
ii. Solve the White Corner
iii. Solve the Middle Layer
iv. Solve the Top Layer
v. Position the Yellow Corner

4.1.5 Application Screenshots

The Application Screen Shots provide an idea of the game; Initially the application looked like the screen shots below and then it developed to the screen shot provided on the next page.

INITIAL SCREEN SHOTS:



Time :

Time : 6.49

Congratulations !!!

Your Time : 6.59

Time : 30.82

Figure 4. 1 SCREEN SHOTS OF MID STAGES OF DEVELOPMENT

SCREEN SHOTS AFTER DEVELOPING THE GAME FURTHER:

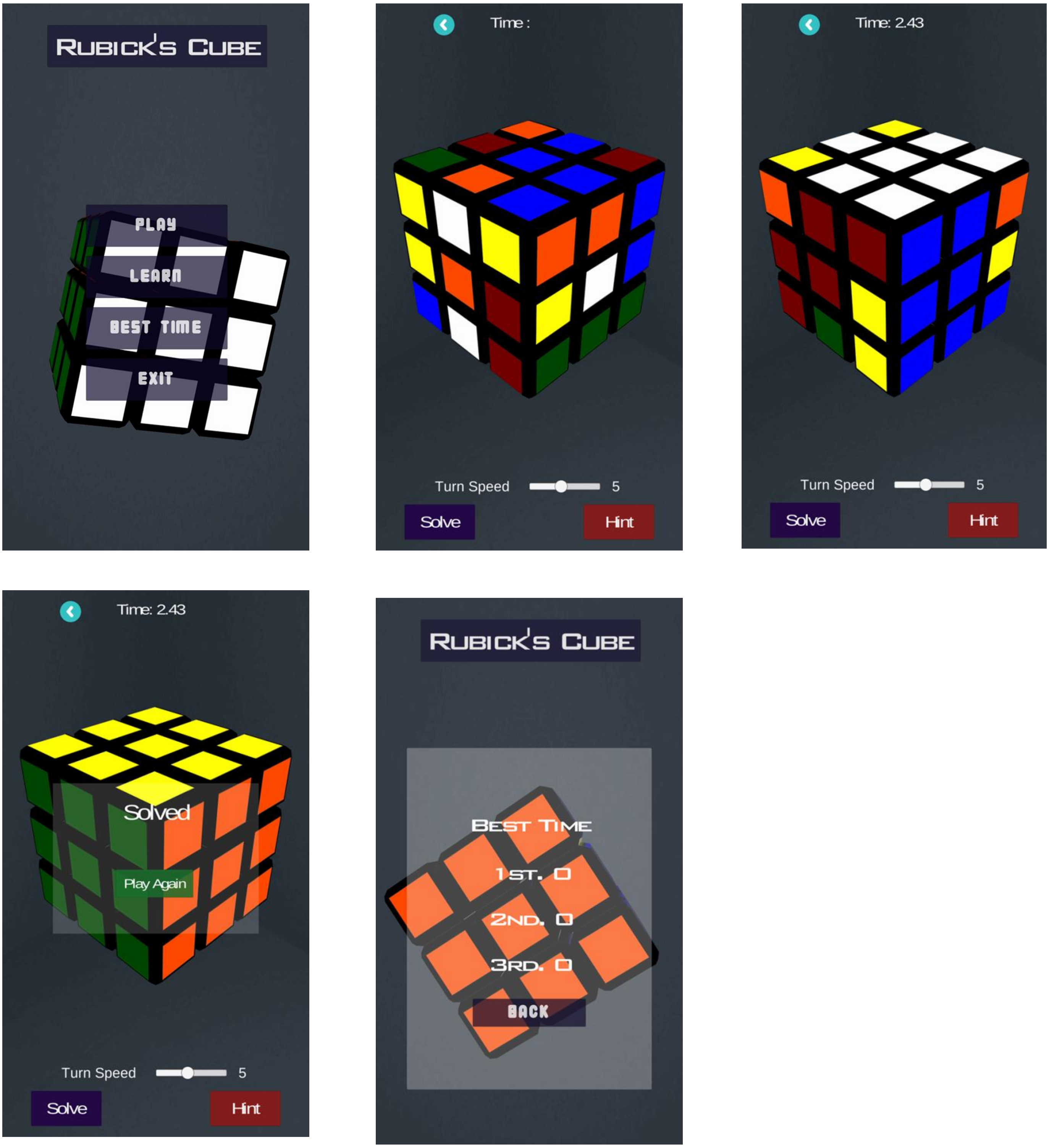


Figure 4. 2 SCREEN SHOTS OF BEST TIME AND PLAY AGAIN

Chapter 5
Software Test Document

5.1 Introduction

Software test document is a standard that specifies the form of a set of documents for use in eight defined stages of software testing and system testing, each stage potentially producing its own separate type of document. Testing is the process of evaluating a system or its components with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item. Testing documentation involves the documentation of artifacts that should be developed before or during the testing of Software.

5.1.1 System Overview

The product defines an Android based puzzle game known as "Rubik's Cube". The purpose of this software is to create an android game through which people of all ages can play and learn how to solve a "Rubik's Cube" by using different algorithms. People can also challenge themselves by solving it under a given time. They can also take hints which helps them solving the cube. The android game will be taking input from a touch screen of any android hand-held device and displaying the result of the actions on the LCD screen.

5.1.2 Test Approach

Testing approach used would be User acceptance testing (UAT) also called end user testing. It consists of a process of verifying that a solution works for the user. It is not system testing (ensuring software does no crash and meets documented requirements), but rather is there to ensure that the solution will work for the user. This should be undertaken by a subject-matter expert (SME), preferably the owner or client of the solution under test, and provides a summary of the findings for confirmation to proceed after trail or overview. In software development, UAT as one of the final stages of a project often occurs before a client or customer accepts the new system. Users of the system perform test in line with what occur in real life scenarios.

5.1.3 Testing Objectives

For checking whether the requirement in SRS are fulfilled or not we have to make test on these cases.

UAT has following objectives.

i. User Acceptance test make test case against the requirements.

ii. User Acceptance test check actual function, input, expected result, actual result, procedure to make test case, pass/fail status against each test case

5.2 Test Plan

5.2.1 Features to be tested

Features to be tested are all according to user/player prospective. For example

- Make Turn
- Change Settings
- Restart game
- Take Hint
- Solve Game
- Undo turn
- View Instructions
- View best time
- Reset best time
- Exit game

5.2.2 Features not to be tested

Features not to be tested are from the developer’s point of view. For example

- Frame rate of game
- How much power is used by processor
- How much memory is consumed by the game
- Software risk factor
- Maintainability of game

5.2.3 Testing Tools and Environment

Since this is UAT testing (testing by the user) so there is no specific tools and environment is required. All a user need is an Android device with game installed.

5.3 Test Cases

5.3.1 Test Case for Make Turn

Table 5. 1 TEST CASE FOR MAKE TURN

Test Case 1: Play Game	
Purpose	Make Turn
Setup	1. Open game app 2. Go to main menu 3. Select Play option
Instructions	1. Select Start option 2. Swipe layers to take turn.
Expected Result	Game is played successfully.
Observed Result	Game is played successfully.
Frequency	Pass: 3 Fail: 1
Verdict	Passed

5.3.2 Test Case for Change Settings

Table 5. 2 TEST CASE FOR CHANGE SETTINGS

Test Case 2: Change Settings	
Purpose	Change Settings
Setup	1. Go to main menu 2. Select Play option
Instructions	1. Select gear icon 2. Change to desired settings
Expected Result	Settings have been changed successfully.
Observed Result	Settings have been changed successfully.
Frequency	Pass: 3 Fail: 1
Verdict	Passed

5.3.3 Test Case for Restart Game

Table 5. 3 TEST CASE FOR RESTART GAME

Test Case 3: Restart Game	
Purpose	Restart game
Setup	1. Go to main menu 2. Select Play option
Instructions	1. Select restart option
Expected Result	Cube has been reshuffled and timer has been reset successfully.
Observed Result	Cube has been reshuffled and timer has been reset successfully.
Frequency	Pass: 2 Fail: 1
Verdict	Passed

5.3.4 Test Case for Undo Turn

Table 5. 4 TEST CASE FOR UNDO TURN

Test Case 4: Undo Turn	
Purpose	Undo Turn
Setup	1. Open game app 2. Go to main menu 3. Select Play option
Instructions	1. Swipe layer on the screen. 2. Select undo option
Expected Result	Layer has been undone successfully.
Observed Result	Layer has been undone successfully.
Frequency	Pass: 1 Fail: 1
Verdict	Passed

5.3.5 Test Case for Take Hint

Table 5. 5 TEST CASE FOR TAKE HINT

Test Case 5: Take Hint	
Purpose	Take Hint
Setup	1. Go to main menu 2. Select Play option 3. Select Start option
Instructions	1. Select Hint option.
Expected Result	Hint has been displayed successfully.
Observed Result	Hint has been displayed successfully.
Frequency	Pass: 3 Fail: 0
Verdict	Passed

5.3.6 Test Case for Solve Cube

Table 5. 6 TEST CASE FOR SOLVE CUBE

Test Case 6: Solve Cube	
Purpose	Solve Cube
Setup	1. Go to Play mode from the main menu
Instructions	Press solve cube button
Expected Result	Cube gets solved
Observed Result	Cube gets solved
Frequency	Pass:4 Fail:1
Verdict	Passed

5.3.7 Test Case for View Instructions

Table 5. 7 TEST CASE FOR VIEW INSTRUCTIONS

Test Case 7: View Instructions	
Purpose	View instructions
Setup	1. Go to main menu
Instructions	1. Select help option
Expected Result	Algorithm steps have been displayed successfully.
Observed Result	Algorithm steps have been displayed successfully.
Frequency	Pass: 2 Fail: 1
Verdict	Passed

5.3.8 Test Case for View Best Time

Table 5. 8 TEST CASE FOR VIEW BEST TIME

Test Case 8: View Best Time	
Purpose	View best time
Setup	1. Go to main menu
Instructions	1. Select best time option
Expected Result	Best time list has been reset successfully.
Observed Result	Best time list has been reset successfully.
Frequency	Pass: 2 Fail: 0
Verdict	Passed

5.3.9 Test Case to Reset Best Time

Table 5. 9 TEST CASE TO RESET BEST TIME

Test Case 9: Reset Best Time	
Purpose	Reset best time
Setup	2. Go to main menu 3. Select best time option
Instructions	2. Select reset option
Expected Result	Best time list has been reset successfully.
Observed Result	Best time list has been reset successfully.
Frequency	Pass: 2 Fail: 0
Verdict	Passed

5.3.10 Test Case for Exit Game

Table 5. 10 TEST CASE FOR EXIT GAME

Test Case 10: Exit Game	
Purpose	Exit game
Setup	None
Instructions	1. Select Exit option
Expected Result	Game application has been closed successfully.
Observed Result	Game application has been closed successfully.
Frequency	Pass: 1 Fail:1
Verdict	Passed

Chapter 6
Conclusion and Future Enhancements

6.1 Summary

The product defines an Android based puzzle game known as “Rubik’s Cube”. The purpose of this software is to create an android game through which people of all ages can play and learn how to solve a “Rubik’s Cube” by using different algorithms. People can also challenge themselves by solving it under a given time. They can also take hints which help them solving the cube. The android game will be taking input from a touch screen of any android hand-held device and displaying the result of the actions on the LCD screen.

6.2 Future Enhancements

Game Enhancement and Future work can be done by:

i. Making game online based so that anyone could challenge another user to beat the highest score

ii. The game will be further developed, where other algorithms such as Fredrick’s Algorithm and ZZ Algorithm would be explored and implemented to solve the cube.

iii. LAN option will be added with simulation which would help in learning the solution of Rubik’s Cube in a better way.

iv. VR option can be explored to make the game more interesting and interactive for the user

v. Camera angles would be explored further

vi. This understanding of the algorithms would further be incorporated into making a 4*4*4 Rubik’s cube as an advanced and more challenging version of this game.

The Understanding and the programme written for this game in future would act as an aid for further development on the similar concepts. This would serve as a helpful guide for students and novice game developers who want to develop such similar applications.

Additional Material

There are no additional requirements.

Appendix C: References

Following is the list of documents and Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document.

Books:

1. C. Larman, APPLYING UML AND PATTERNS an Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed., Massachusetts: Pearson Education, 2005
2. Roger S. Pressman, Software Engineering - A Practitioner's Approach, McGraw Hill, 7th Edition, 2010

Websites:

3. http://rubikscube.info/concepts.php