A

REAL TIME RESEARCH PROJECT REPORT

ON

DICE GAME: ROLL FOR LUCK!

Submitted in partial fulfilment of the requirements for award of the degree of

BACHELOR OF TECHNOLOGY

In

Artificial Intelligence & Data Science

By

Koppala Raja Nandini	**23BH1A7248**
Koyyadi Sai Krishna	**23BH1A7250**
Pillalamarri Vaishnavi	**23BH1A7277**
Sardar Jasmeeth Singh Raj	**23BH1A7237**
Mirza Adnam Sami Beig	**23BH1A7260**

Under the guidance of

Mr. B. SRISHAILAM, **M.Tech, (Ph.D)**

Assistant Professor

Artificial Intelligence & Data science

St. Mary's Engineering College

(Approved by AICTE, NEW DELHI. & Affiliated to JNTU-HYDERABAD, Accredited by NAAC)
Deshmukhi (V), Pochampally (M), Yadadri Bhuvanagiri (D), Telangana -50824

[2024-25]

CERTIFICATE

This is to certify that project report entitled **"DICE GAME: ROLL FOR LUCK!"** is bonafide work carried out in the II/II semester by **K. Raja Nandini (23BH1A7248), K. Sai krishna (23BH1A7250), P. Vaishnavi (23BH1A7277), S. Jasmeeth Singh Raj (23BH1A7237), M.Adnan Sami Beigm (23BH1A7260)** in partial fulfilment award of Bachelor of Technology in **Artificial Intelligence & Data Science** from **St. Mary's Engineering college** during the academic year 2023-27.

INTERNAL GUIDE

Mr. B. SRISHAILAM, M.Tech, (Ph.D)

Dept. of AI&DS

HEAD OF THE DEPARTMENT

Mr. B. SRISHAILAM. M.Tech (Ph.D)

Dept. of AI&DS

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible. We consider it as a privilege to express our gratitude and respect to all those who guided us in the completion of the project.

We are thankful to our internal guide **Mr.B. SRISHAILAM, M.Tech, (Ph.D) in Department of Artificial Intelligence & Data Science, St. Mary's Engineering College** for having been of a source encouragement and for insisting vigour to do this project work.

We are obliged to **Mr. B. SRISHAILAM**, **Head of the Department of Artificial Intelligence & Data Science, St. Mary's Engineering College** for his guidance and suggestion throughout project work.

We take this opportunity to express a deep sense of gratitude to **Dr. T.N. Srinivas Rao, Principal of St. Mary's Engineering College** for allowing us to do this project and for this affectionate encouragement in presenting this project work.

We convey our sincere thanks to **Sri Rev. K.V.K RAO, Chairman** of **St. Mary's Engineering College** for giving us learning environment to grow out self personally as well as professionally.

We would like to express our thanks to all staff members who have helped us directly and indirectly in accomplishing this project work. We also extended our sincere thanks to our parents and friends for their moral support throughout the project work. Above all we thank god almighty for his manifold mercies in carrying out this project work successfully.

Koppala Raja Nandini -23BH1A7248

Koyyadi sai krishna -23BH1A7250

Pillalamarri Vaishnavi -23BH1A7277

Sardar Jasmeeth singh Raj -23BH1A7237

Mirza Adnan Sami Beig -23BH1A7260

DECLARATION

This is to certify that the work report in thesis titled, **"DICE GAME: ROLL FOR LUCK!",** submitted to **the Department of Artificial Intelligence & Data Science, St. Mary's Engineering College** in fulfilment of degree for the award of Bachelor of Technology, is a bonafide work done by us. No part of the thesis is copied from books, journals or internet and wherever the portion is taken, the same has been duly referred in the text. The reported results are based on the project work entirely done by us and not copied from any other sources. Also we declare that the matter embedded in this thesis has not been submitted by us in full or partially there for the award of any degree of any other institution or university previously.

Koppala Raja Nandini -23BH1A7248

Koyyadi sai krishna -23BH1A7250

Pillalamarri Vaishnavi -23BH1A7277

Sardar Jasmeeth singh Raj -23BH1A7237

Mirza Adnan Sami Beig -23BH1A7260

CONTENTS

CHAPTER NAME	PAGE No
1. INTRODUCTION	**1**
1.1 INTRODUCTION	1
1.2 FEATURES	1
2. SYSTEM ANALYSIS	**2-4**
2.1 INTRODUCTION	2
2.2 EXISTING SYSTEM &ITS DISADVANTAGES	3
2.3 PROPOSED SYSTEM &ITS ADVANTAGES	4
3. SYSTEM STUDY	**5-6**
3.1 FEASIBILITY STUDY	**5**
3.1.1 ECONOMICAL FEASIBILITY	5
3.1.2 TECHNICAL FEASIBILITY	5
3.1.3 SOCIAL FEASIBILITY	6
4. SYSTEM REQUIREMENTS	**7**
4.1 HARDWARE REQUIREMENTS	7
4.2 SOFTWARE REQUIREMENTS	7
5. SYSTEM ARCHITECTURE	**8**
6. SYSTEM DESIGN	**9-13**
6.1 UML DIAGRAMS	**9**
6.1.1 CLASS DIAGRAM	10
6.1.2 USE CASE DIAGRAM	
6.1.3 SEQUENCE DIAGRAM	11
6.1.4 ACTIVITY DIAGRAM	12

7. SOFTWARE ENVIRONMENT **14-15**
7.1 ANDROID 14
7.2 DOWNLOAD THE ANDROID SDK 15
7.3 ECLIPSE 15

8. SOURCE CODE **16-27**

9. SYSTEM TESTING **28-31**

10. OUTPUT SCREENS **32-33**

11. CONCLUSION **34**

BIBLIOGRAPHY **35**

LIST OF FIGURES

S.NO	FIGURE NO	FIGURE NAME	Page No
1	5.1	SYSTEM ARCHITECTURE	8
2	6.1.1	CLASS DIAGRAM	9
3	6.1.2	USE CASE DIAGRAM	10
4	6.1.3	SEQUENCE DIAGRAM	11
5	6.1.4	ACTIVITY DIAGRAM	12

LIST OF PLATES

S.NO	PLATE No.	PLATE NAME	PAGE No
1	10.1	User Home Page	32
2	10.2	User selects the option	33
3	10.3	Shows the result	33

ABSTRACT

The dice roller project is a digital simulation of traditional dice used in various games and decision-making activities. The primary objective is to create an interactive and user- friendly application that accurately replicates the randomness and behavior of physical dice. This application allows users to roll different types of dice, such as D6 (six-sided), D20 (twenty-sided), and other standard polyhedral dice, commonly used in board games and role-playing games (RPGs). The core functionality is powered by a pseudo-random number generator, which ensures that each dice roll is unpredictable and fair, closely resembling real-world randomness. The dice roller includes a graphical user interface (GUI) that provides visual and auditory feedback for each roll, enhancing user engagement. Additional features such as roll history, multiple dice selection, and custom dice settings make the tool versatile for various gaming and educational scenarios.

CHAPTER 1
INTRODUCTION

1.1 Introduction

This project is a web-based multiplayer dice rolling game designed to provide an engaging and interactive user experience. It utilizes HTML, CSS, and JavaScript for the frontend, combined with a Flask backend to handle game logic and player interactions. The game supports 1 to 4 players and features animated 3D dice, a clean and modern interface, and additional functionalities like a total score calculator. The aim is to simulate the excitement of physical dice games in a digital format. This project not only showcases core web development skills but also emphasizes UI/UX design and real-time gameplay functionality.

1.2 Features

A dice roller application typically includes a range of features designed to simulate the rolling of physical dice. The core feature is the ability to roll one or multiple dice of various types, such as standard six-sided dice (d6) as well as other common tabletop gaming dice like d4, d8, d10, d12, and d20. Users can often customize the number and type of dice rolled in a single

session. The app may display visual or animated representations of the dice rolls for a more engaging experience. Additional features can include saving or sharing roll results, adding modifiers to the total (useful for role-playing games), keeping a roll history, and supporting preset roll combinations for quick access. Some advanced dice rollers also include probability calculators, custom dice creation, and sound effects to enhance realism. These features make dice rollers useful tools for board games.

CHAPTER 2

SYSTEM ANALYSIS

2.1 Introduction

This project is a modern web-based multiplayer dice rolling game for 1–4 players. It uses a Flask backend with HTML, CSS, and JavaScript for the frontend. The game features animated 3D black dice, a light blue background, and a "Total" button to display the sum of dice rolls. Players interact through a responsive UI, with game logic handled server-side. The system supports basic multiplayer gameplay, random dice generation, and session management. Future enhancements may include real-time multiplayer with Web Sockets, persistent storage, and user profiles. The design focuses on simplicity, performance, and a smooth, engaging user experience.

The core functionality revolves around providing a responsive and seamless gaming experience. Users interact with the system through an intuitive web interface, where the primary game logic is processed on the server side, allowing for better security, scalability, and synchronization in multiplayer sessions. The system includes fundamental features such as random dice number generation, user session management, and turn-based multiplayer interaction.

2.2 EXISTING SYSTEM

This system made for one player only. It run on browser and made with HTML, CSS, JavaScript, and Flask. Player click button to roll dice. Dice is black and show 3D animation. When rolled, it show random number from 1 to 6. Background of game is light blue. There is button to show total of rolled numbers. Game is simple and easy. Flask backend control dice roll and send number to frontend. Player play alone, no need other users. It help for fun or practice. Design is clean, not many buttons. Whole system work smooth and fast.

Disadvantages

The previous project is created only for single player. It doesn't have any option to choose number of players or number of rolls. User just click one button and dice roll one time. No animation look good, dice is simple. No total button, no multiplayer, nothing extra. Background is plain and not styled nice. No sound, no user experience features. Backend only give one number back. Frontend is very basic, no cool design. It work but not interesting. Compared to my project, it is very small, not smart system, and feel like test version only.

2.3 PROPOSED SYSTEM

This new system is better and have more fun. It support 1 to 4 players. Players can choose how many want to play. Dice are black color and show 3D animation when roll. Background is light blue and look nice. Player can press button to roll dice and another button to show total of all dice. Game made using HTML, CSS, JavaScript, and Flask. Backend control the game rules and frontend show the design. It work fast and smooth on browser. Players can take turns. Game look modern, easy to use, and more interesting than old system. It feel complete now.

Advantages

This project have many good things. It allow 1 to 4 players, so friends can play together. Dice have 3D animation, so it look cool and fun. Background color is soft and nice for eyes. Player can click button to roll dice and also see total with one click. Game is fast and no need to install, just open in browser. Design is clean and easy to understand. Backend with Flask make system work smooth. It give better experience than simple one-player game. Players enjoy more because of multiplayer and good user interface. Whole game feel more smart and modern.

CHAPTER 3

SYSTEM STUDY

3.1 Feasibility Study

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During syntent input forthe feasibility study of the proposed system is to be carried out. This isto ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditure must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must have a high demand for the available technical resources. This will lead to high demand for the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system

3.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 Hardware Requirements

Processor	:	intelcore i5 or Above
RAM	:	8 GB minimum
Hard Disk	:	256 GB SSD or above
Input Devices	:	Keyboard, Mouse.

4.2 Software Requirements

Operating System	:	Windows 10/11 or ubuntu 20.04+
Coding Language	:	HTML, CSS, JavaScript,
Tool	:	Visual Studio Code

CHAPTER 5

SYSTEM ARCHITECTURE

System architecture is the structure that show how parts of system work together. It tell how frontend and backend talk to each other.

In this project, frontend use HTML, CSS, and JavaScript for game screen and user actions. Backend made with Flask, it handle dice roll, player turns, and send results to frontend. Both parts work together to make full game.

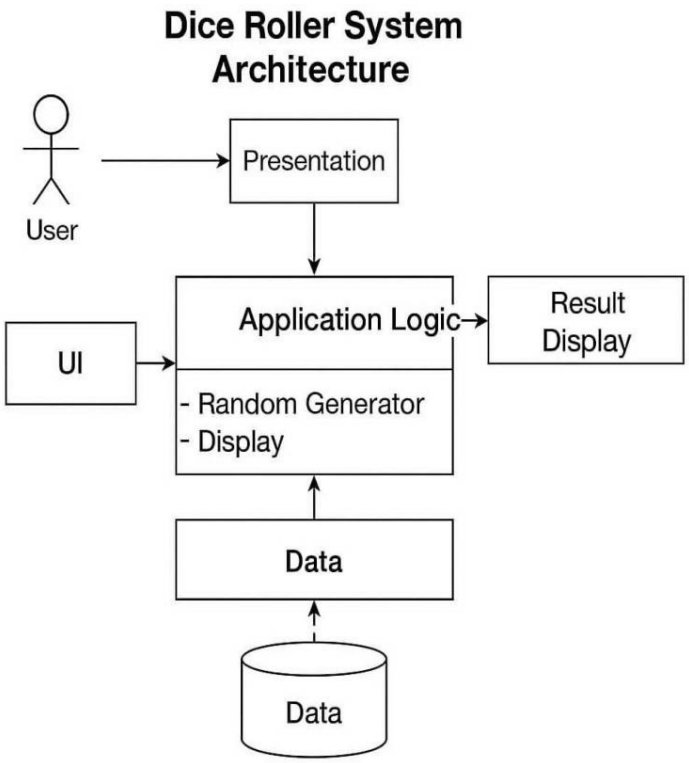


Figure: 5.1 System Architecture

CHAPTER 6

SYSTEM DESIGN

6.1 UML DIAGRAM

UML Diagrams (Unified Modeling Language Diagrams) are standardized visual representations used in software engineering to describe the structure, behavior, and interactions of a software system. UML provides a universal way for developers, designers, and stakeholders to visualize and document different aspects of a system, making it easier to understand, communicate, and manage complex software development projects.

There are four types of UML diagrams:

- Class Diagram: Shows classes, attributes, methods, and relationships like inheritance and association.
- Use Case Diagram: Illustrates system functionality from a user's perspective.
- Sequence Diagram: Shows how objects interact over time through messages.
- Activity Diagram: Represents workflows of actions or business processes.

UML diagrams are widely used in object-oriented design and can support various software development methodologies like Agile, Waterfall, or RUP. They are especially helpful during system analysis, design, and documentation phases.

By using UML, teams can reduce misunderstandings, improve design quality, and provide a solid blueprint before coding begins. UML diagrams are typically created using tools like Lucid chart, Star UML, or Visual Paradigm.

6.1.1 CLASS DIAGRAM

Class diagram show how different parts (classes) of system connect and work. It show class name, data inside, and what actions it can do.

In this project, class diagram can have Player class, Dice class, and Game class. Each class have own role like rolling dice, storing player info, and managing turns.

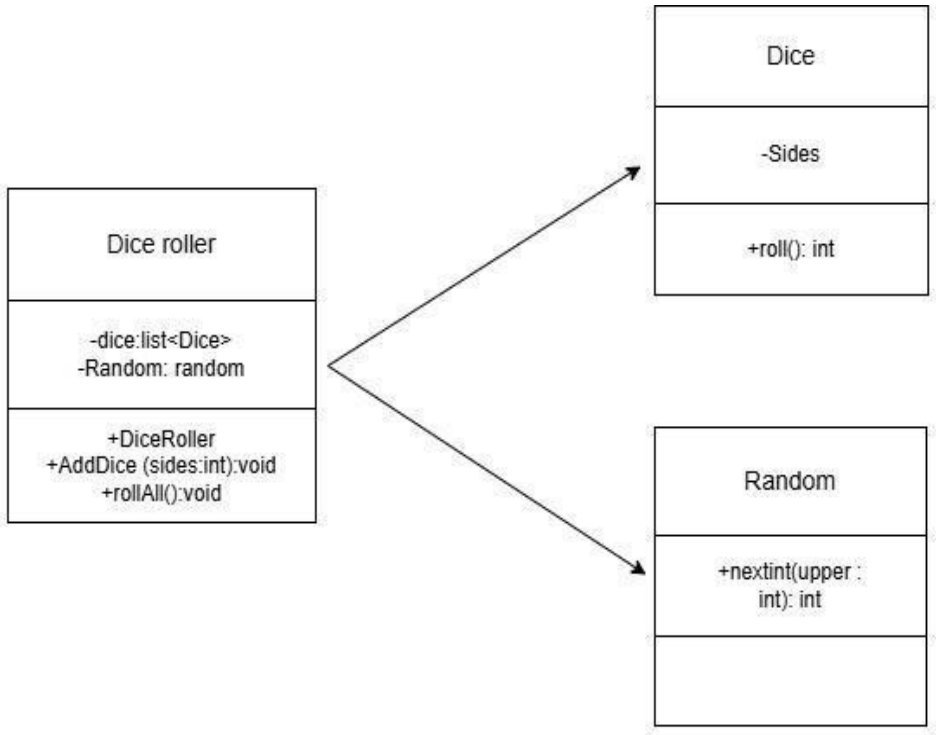


Figure: 6.1.1 Class diagram

6.1.2 USE CASE DIAGRAM

Use case diagram show what user can do in system. It show actions between user and system like a map.

In this project, use case diagram have options like "Roll Dice", "View Total", and "Select Players". It help to understand user steps in game.

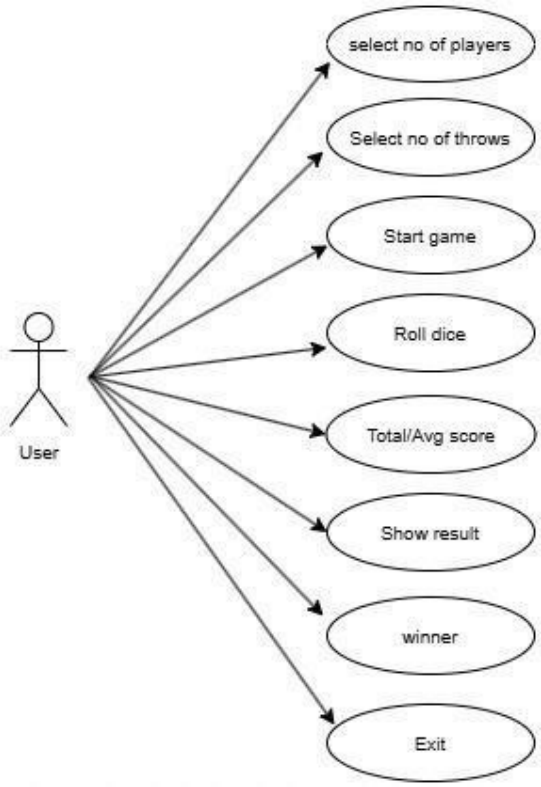


Figure: 6.1.2 Use Case diagram

6.1.3 SEQUENCE DIAGRAM

Sequence diagram show step-by-step how system parts talk in time order. It show message flow between user, frontend, and backend.

In this project, it show how player click roll, frontend send request, backend give dice number, and frontend show result.

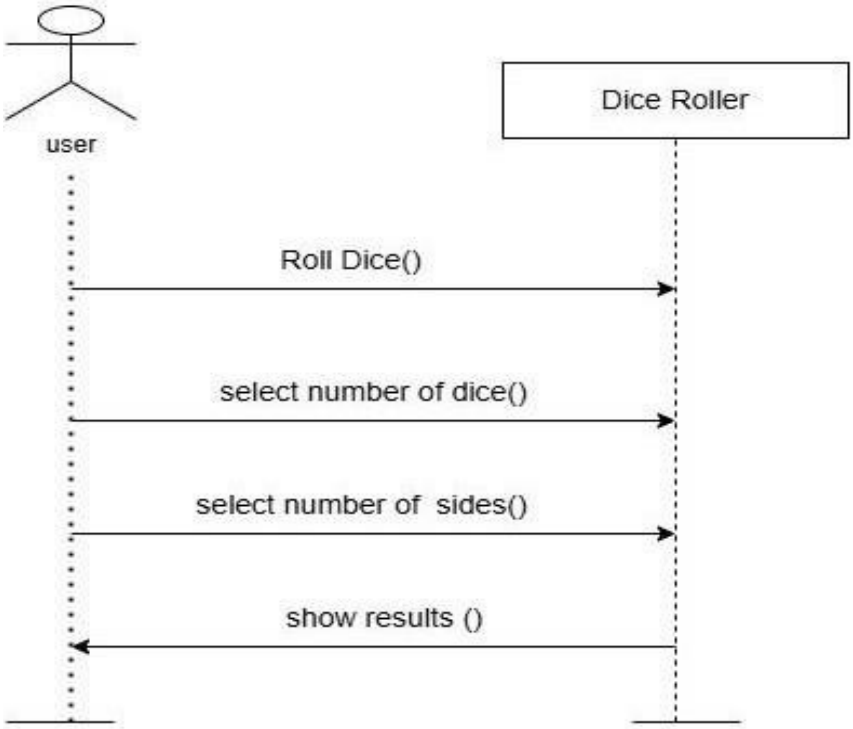


Figure: 6.1.3 sequence diagram

6.1.4 ACTIVITY DIAGRAM

Activity diagram show flow of actions in system. It tell what happen first, next, and after.

In this project, it show steps like start game, select players, roll dice, see result, and end game. It help to see full game process.

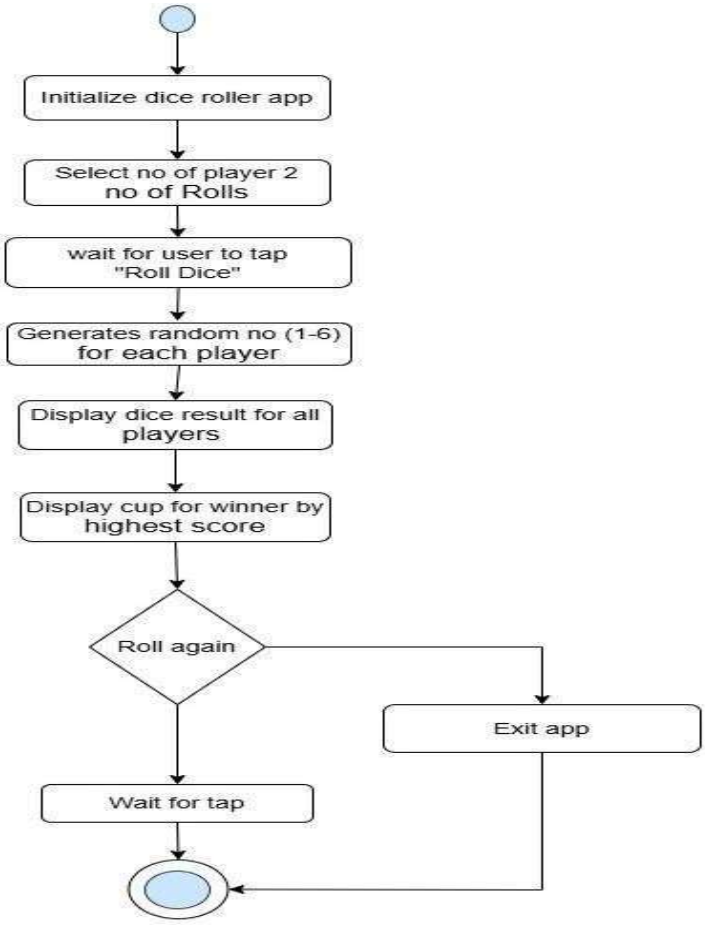


Figure: 6.1.4 Activity diagram

CHAPTER 7

SOFTWARE ENVIRONMENT

Modern web and mobile applications rely on various programming languages and software tools. HTML, CSS, and JavaScript form the core technologies for front-end development. In the context of Android development, these languages can be used in hybrid applications or web-based views embedded within native apps. The environment setup is crucial for efficient development and testing.

7.1 ANDROID

Android is an open-source mobile operating system developed by Google. It supports both native and hybrid application development. While Java and Kotlin are primarily used for native apps, Android also allows developers to embed web content using WebView, which supports HTML, CSS, and JavaScript.

Developers can use these web technologies to create responsive UI, dynamic content, and interactive features that run inside an Android app. Frameworks like Apache Cordova, PhoneGap, and Ionic enable hybrid app development using standard web technologies.

Use of HTML, CSS, JavaScript in Android:

HTML structures the app content.

CSS styles the content for better visual presentation.

JavaScript adds interactivity and logic.

7.2 DOWNLOAD THE ANDROID SDK

To begin Android development, including web-based applications within Android, the Android SDK (Software Development Kit) is essential. It provides tools like:

Android Debug Bridge (ADB)

Emulator

Build tools

Platform APIs

Developers must download the SDK using Android Studio, the official IDE for Android development. Alternatively, SDK tools can be manually downloaded for integration with other IDEs like Eclipse.

For web-based development:

HTML, CSS, and JS files can be loaded into a WebView.

Testing tools in the SDK can simulate different screen sizes and resolutions.

7.3 ECLIPSE

Eclipse is a versatile and widely-used IDE that supports multiple programming languages through plugins. Though Android Studio has largely replaced Eclipse for Android development, it is still possible to use Eclipse with the ADT (Android Development Tools) Plugin.

For HTML, CSS, and JavaScript development:

Eclipse provides excellent support via plugins like Web Tools Platform (WTP).

Developers can create and manage web projects.

Integrated preview and debugging tools make it easier to test HTML/CSS/JS functionality.

CHAPTER 8

SOURCE CODE

8.1 HTML

```
<!DOCTYPE html>

<htmllang="en">

<head>

 <meta charset="UTF-8" />

 <meta name="viewport" content="width=device-width, initial-scale=1.0" />

 <title>DICE GAME: ROLL FOR LUCK!</title>

 <link rel="stylesheet" href="style.css" />

</head>

<body>

 <div class="container">

  <h1>DICE GAME: ROLL FOR LUCK!</h1>


  <div class="controls">

   <label>

    Players:

    <select id="player-count">

     <option value="1">1</option>

     <option value="2" selected>2</option>

     <option value="3">3</option>
```

```
      <option value="4">4</option>
    </select>
   </label>
   <label>
    Throws each:
    <input type="number" id="throws-count" min="1" max="10" value="5" />
   </label>
   <button id="start-button">Start Game</button>
  </div>

  <div id="game-area" class="hidden">
   <div class="status-bar">
    <span id="current-player">Player 1</span>
    <span id="throws-remaining">Throws left: 5</span>
   </div>

   <div class="dice" id="dice">
    <img src="images/dice1.jpg" alt="Dice" id="dice-image" />
   </div>
   <button id="roll-button">Roll Dice</button>

   <div id="scoreboards"></div>
  </div>
 </div>

<!-- JDice rollsound -->
```

```
  <audio    id="roll-sound"  src="https://freesound.org/data/previews/341/341695_3248244-lq.mp3"></audio>

  <script src="script.js"></script>
</body>
</html>
```

CSS

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
}

body {
  background: linear-gradient(135deg, #0fb9df, #1ac0ee);
  min-height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 1rem;
}

.container {
  background: white;
  padding: 2rem 3rem;
```

```
  border-radius: 1.5rem;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.15);
  width: 150%;
  max-width: 500px;
  text-align: center;
}

h1 {
  color: #2c3e50;
  margin-bottom: 1.5rem;
}

.controls {
  display: flex;
  justify-content: center;
  gap: 1rem;
  margin-bottom: 1rem;
  flex-wrap: wrap;
}

.controls label {
  font-size: 0.9rem;
  color: #34495e;
}

.controls select,
```

```
.controls input {
  margin-left: 0.3rem;
  padding: 0.2rem 0.4rem;
  border-radius: 5px;
  border: 1px solid #bdc3c7;
}

button {
  padding: 0.6rem 1.2rem;
  font-size: 1rem;
  background: #e67e22;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: transform 0.2s ease, background 0.2s ease;
}

button:hover {
  background: #d35400;
  transform: scale(1.05);
}

.hidden {
  display: none;
}
```

```
.status-bar {
  display: flex;
  justify-content: space-between;
  margin: 1rem 0;
  font-weight: bold;
  color: #2c3e50;
}

.dice {
  width: 120px;
  height: 120px;
  margin: 0 auto 1rem;
  perspective: 800px;
  background: #f4f7f8;
  border-radius: 12px;
  padding: 0.5rem;
  background-size: cover;
  background-position: center;
}

.dice img {
  width: 100%;
  animation: none;
}
```

```
@keyframes roll {
  0%  { transform: rotateX(0deg) rotateY(0deg); }
  25%  { transform: rotateX(360deg) rotateY(180deg); }
  50%  { transform: rotateX(720deg) rotateY(360deg); }
  75% { transform: rotateX(1080deg) rotateY(540deg);}
  100% { transform: rotateX(1440deg) rotateY(720deg);}
}

#scoreboards {
  margin-top: 1.5rem;
}

.scoreboard {
  margin-bottom: 1rem;
  text-align: left;
}

.scoreboard h3 {
  margin-bottom: 0.5rem;
  color: #2c3e50;
}

.scoreboard ul {
  list-style: none;
}
```

```
.scoreboard li {
  padding: 0.2rem 0;
  font-size: 0.9rem;
}
```

JAVA SCRIPT

```
const startBtn = document.getElementById("start-button");
const rollBtn = document.getElementById("roll-button");
const gameArea =document.getElementById("game-area");
const playerCountSelect = document.getElementById("player-count");
const throwsInput = document.getElementById("throws-count");
const currentPlayerSpan = document.getElementById("current-player");
const throwsRemainingSpan =document.getElementById("throws-remaining");
const diceImg = document.getElementById("dice-image");
const scoreboardsDiv = document.getElementById("scoreboards");

let players = [];
let currentPlayerIndex = 0;
let throwsPerPlayer = 5;
let totalThrowsLeft = 0;

// Dice images linked
const diceImages = [

  "images/dice1.jpg",//1
  "images/dice2.jpg",//2
  "images/dice3.jpg",//3
```

```
  "images/dice4.jpg",//4
  "images/dice5.jpg",//5
  "images/dice6.jpg" //6
];

// Start Game
startBtn.addEventListener("click", () => {
  const numPlayers = parseInt(playerCountSelect.value);
  throwsPerPlayer = parseInt(throwsInput.value) || 1;
  totalThrowsLeft =throwsPerPlayer * numPlayers;
  currentPlayerIndex = 0;

  players = [];
  for (let i= 1; i <= numPlayers; i++) {
    players.push({ name: Player ${i}, rolls: [], total: 0 });
  }

  renderScoreboards();
  updateStatus();
  rollBtn.disabled = false;
  gameArea.classList.remove("hidden");
});

// Roll Dice
rollBtn.addEventListener("click", () => {
  // Restart animation
```

```
diceImg.style.animation = "none";
diceImg.offsetHeight; // Force reflow
diceImg.style.animation = "roll 1s ease";

setTimeout(() => {
  const roll = Math.floor(Math.random() * 6) + 1;
  diceImg.src = diceImages[roll- 1];

  const player =players[currentPlayerIndex];
  player.rolls.push(roll);
  player.total += roll;

  totalThrowsLeft--;

  renderScoreboards();

  if (totalThrowsLeft > 0) {
    currentPlayerIndex = (currentPlayerIndex + 1) % players.length;
    updateStatus();
  } else {
    currentPlayerSpan.textContent ="Game Over!";
    throwsRemainingSpan.textContent = "";
    rollBtn.disabled = true;
    highlightWinner();
  }
}, 1000);
```

```
});

function updateStatus() {
  const player = players[currentPlayerIndex];
  currentPlayerSpan.textContent =player.name;
  throwsRemainingSpan.textContent =Total Throws Left: ${totalThrowsLeft};
}

function renderScoreboards() {
  scoreboardsDiv.innerHTML="";
  players.forEach((p, idx) => {
    const sb =document.createElement("div");
    sb.classList.add("scoreboard");
    sb.innerHTML = `
      <h3>${p.name} — Total: ${p.total}</h3>
      <ul>
        ${p.rolls.map((r, i) => <li>Roll${i+1}: ${r}</li>).join("")}
      </ul>
    `;
    scoreboardsDiv.appendChild(sb);
  });
}

function highlightWinner() {
  const maxTotal= Math.max(...players.map(p => p.total));
  players.forEach((p, idx) => {
```

```
    const sb =scoreboardsDiv.children[idx];
    if (p.total === maxTotal) {
      const h3 =sb.querySelector("h3");
      h3.style.color = "#c0392b";
      h3.textContent += " 👑 ";
    }
  });
}
```

CHAPTER 9

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTINGS:

9.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

9.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent.

9.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available specified by the business and technical requirements, system documents, available manuals

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted. Invalid Input: identified classes of invalid input must be rejected. Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised. Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

9.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

9.1.5 White Box Testing

White Box Testing is a test in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is the purpose.It is used to test areas that cannot be reached from a black box level.

9.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings. structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually, and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed
- Features to be tested
- Verifythat the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integra software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up-software applications at the company level-interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9.1.7 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements. Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 10

OUTPUT SCREENS

10.1 USER HOME PAGE

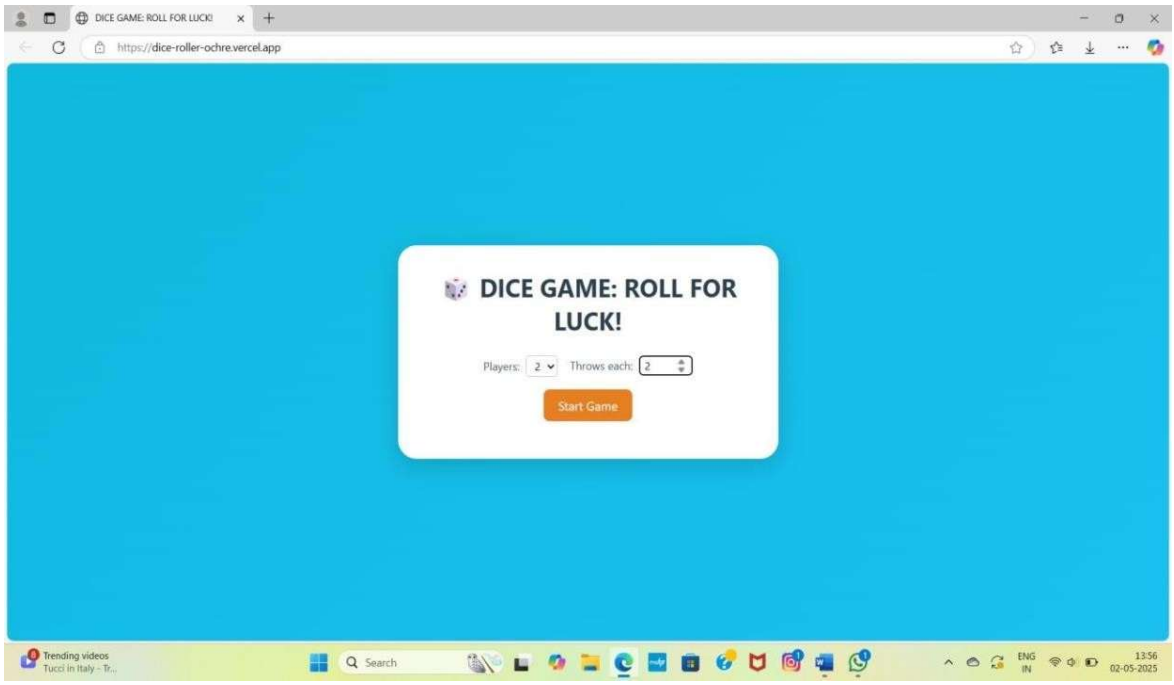
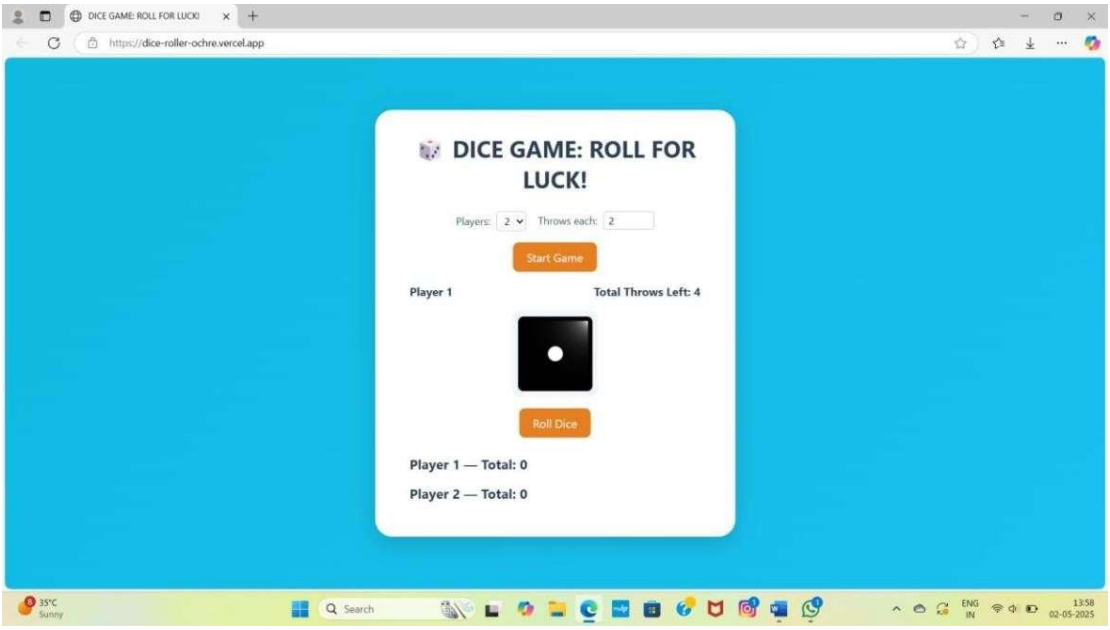


Figure: 10.1 User Home Page

10.2 USER SELECT THE OPTION



Fgure: 10.2 User Select The Option

10.3 SHOWS THE RESULT

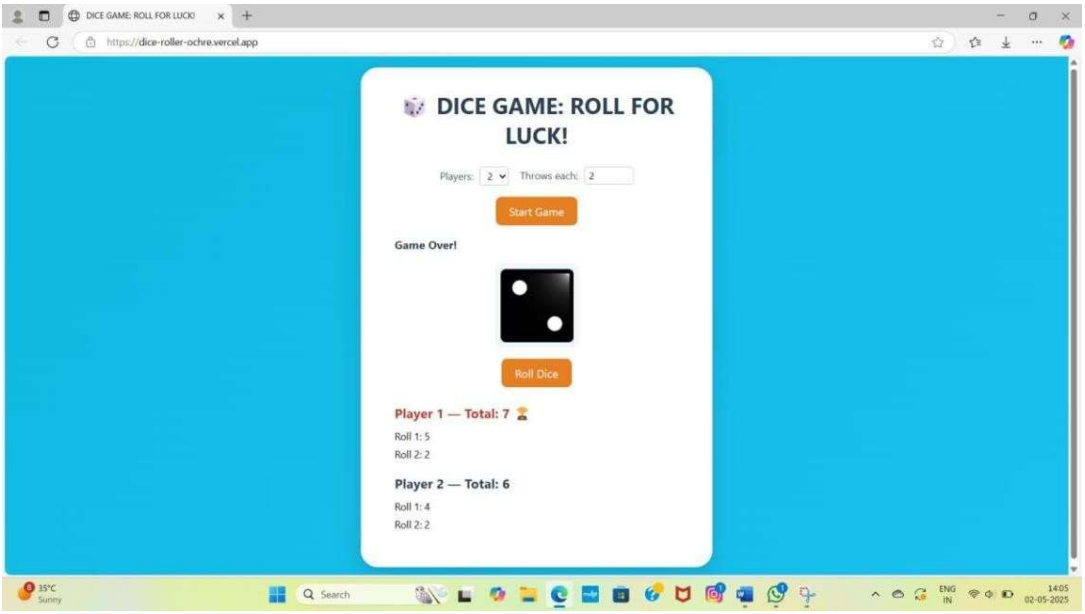


Figure: 10.3 Shows the Result

CHAPTER 11

CONCLUSION

The dice roller is a practical tool designed to simulate the randomness of real dice. It is commonly used in games, decision-making, and educational purposes to demonstrate probability and randomness. This project showcases how simple programming logic, such as generating random numbers, can be applied to solve real-world problems. It offers a fair and efficient alternative to physical dice, eliminating the chance of bias. The dice roller can be easily enhanced to support multiple dice, different dice types, or customized rules. Overall, it is a fun, interactive, and useful tool that highlights the importance of randomness in digital applications.

FUTURE ENHANCEMENT

Future work for a dice roller could include enhancing its visual appeal, adding advanced features like custom die types and dice notation, and integrating it with other applications for broader use. This could also involve improving the physics of the simulation, making it more realistic or playful.

BIBLIOGRAPHY

1. HTML & CSS: Design and Build Websites – Jon Duckett

(Reference for designing web pages and understanding responsive layout principles).

2. JavaScript: The Definitive Guide – David Flanagan

(Used to understand and implement interactivity on the website).

3. W3Schools – (For HTML, CSS, and JavaScript tutorials and examples)

MDN Web Docs (Mozilla Developer Network)

(Technical references and best practices for web development).

4. GitHub Documentation– (For version control and GitHub Pages deployment)

Stack Overflow – (For troubleshooting errors and improving code snippets

5. FreeCodeCamp – (Helpful for frontend development tutorials and real-world

project examples)

Responsive Web Design Basics – Google Web Fundamentals

https://web.dev/responsive-web-design-basics