A

MINI PROJECT REPORT

ON

Draw and Guess using MERN

Submitted in partial fulfillment of the requirements

For the award of Degree of

BACHELOR OF ENGINEERING

IN

CSE

Submitted By

ANIRUDH VUPPALA	**245321733004**
VARSHA MALLIKARJUNA	**245321733030**
VARSHIT MADISETTI	**245321733061**

Under the guidance

Of

SWAPNA SIDDAMSETTI

ASSISTANT PROFESSOR

DEPARTMENT OF CSE

NEIL GOGTE INSTITUTE OF TECHNOLOGY

Kachivani Singaram Village, Hyderabad, Telangana 500058

2024

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

CERTIFICATE

This is to certify that the Mini Project Report entitled "**DRAW AND GUESS**" is a bona fide work carried out by **ANIRUDH.V(245321733004), VARSHA M(245321733030), VARSHIT M(245321733061)** of III year VI semester Bachelor of Engineering in CSE by Osmania University, Hyderabad during the academic year **2023-2024** is a record of bona fide work carried out by them. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree.

Internal Guide

Swapna Siddamsetti

Assistant Professor

Head of Department

Dr. K V RANGA RAO

Professor

External

NEIL GOGTE INSTITUTE OF TECHNOLOGY
A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

DECLARATION

We hereby declare that the Mini Project Report entitled, "**DRAW AND GUESS**" submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date:

V.ANIRUDH	**245321733004**
M.VARSHA	**245321733030**
M.VARSHIT	**245321733061**

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A Unit of Keshav Memorial Technical Education (KMTES)
Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college **Prof. R. Shyam Sunder, Professor**, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank**, Dr. K V RANGA RAO**, **Head of the CSE Department,** And **Mini Project Coordinator Dr. B Krishna Associate Professor CSE Department** Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would like to extend our special thanks to our internal guide **SWAPNA SIDDAMSETTI Assistant Professor** for his technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering for their timely suggestions, healthy criticism and motivation during this work.–

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

ABSTRACT

Our project introduces a captivating multiplayer drawing and guessing game, designed to offer an engaging gaming experience for players of all demographics. At its core, the game enables participants to immerse themselves in real-time drawing challenges, guess each other's creations, and interact socially within a virtual gaming environment reminiscent of popular online platforms.

To bring this vision to life, we harness a sophisticated array of modern web technologies. Node.js powers the server-side logic, managing game rooms, player interactions, and real-time communication. Complementing this, Socket.IO facilitates seamless WebSocket-based connections between the server and clients, ensuring instant updates and synchronization across all participants. On the frontend, React.js serves as the foundation for crafting an intuitive and visually appealing user interface. Through React.js, players can seamlessly navigate game rooms, express their creativity through drawing, submit guesses, and engage in lively conversations via integrated live chat functionality.

By amalgamating these technologies, our project endeavors to deliver a comprehensive gaming experience that blends competition, creativity, and social interaction. Join us on this journey as we embark on creating a vibrant gaming platform that captivates players worldwide.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1
INTRODUCTION

1.1 : PROBLEM STATEMENT

Develop a responsive multiplayer web application for a drawing and guessing game that enhances user experience. The app should support secure, real-time interaction in private and public rooms, and dynamic leaderboards. Key features include a robust drawing canvas, real-time chat with word detection to maintain game integrity, and customizable game settings.

1.2 : MOTIVATION

The primary goal of "Draw and Guess" is to entertain users and provide them with a relaxing experience. As a multiplayer game, it also fosters friendship and social interactions, helping players connect with others in a fun and engaging way.

1.3 :SCOPE

The scope of the "Draw and Guess" game includes real-time multiplayer gameplay, integrated chat, customizable profiles, and leaderboards. It supports cross-platform play and provides admin tools for moderation. The system ensures secure data handling, scalability, and potential future enhancements for new game modes and features.

1.4 : OUTLINE

The "Draw and Guess" game features real-time multiplayer drawing and guessing, integrated chat, customizable profiles, and leaderboards. It supports cross-platform play, includes admin moderation tools, ensures secure data handling, and is scalable for many users. Future updates may introduce new game modes and features.

CHAPTER – 2

LITERATURE SURVEY

2.1 : EXISTING SYSTEM:

Inspired by Skribbl.io, our project enhances the popular multiplayer drawing and guessing game with modern web technologies. While Skribbl.io offers a solid foundation for real-time competition and social interaction, we aim to innovate with additional features to improve gameplay, user experience, and overall enjoyment.

DISADVANTAGES:

- **Limited Customization:** Skribbl.io offers minimal options for customizing game settings and player profiles.
- **Moderation Challenges:** The platform lacks advanced moderation tools, leading to potential issues with inappropriate content and behavior.
- **Scalability Issues:** The game may struggle with performance and stability under high user loads.
- **Limited Features:** Basic gameplay can become repetitive, with few options for new game modes or enhanced interaction.
- **Ads and Monetization:** Free access is supported by ads, which may disrupt the user experience.

2.2 : PROPOSED SYSTEM:

Practical enhancements for Skribbl.io could include optimizing the user interface for better mobile compatibility, introducing a report system for handling disruptive players, implementing a basic chat filter to prevent inappropriate language, and adding a simple tutorial for new players. Additionally, integrating basic moderation tools for room hosts, such as the ability to kick or mute players, could improve overall room management. These improvements focus on enhancing usability, promoting a safer and more enjoyable gaming environment, and addressing common user feedback, ultimately contributing to a more polished and accessible multiplayer gaming experience on Skribbl.io.

ADVANTAGES:

- **Enhanced Moderation Tools:** Features like player reporting, kicking, and muting to manage disruptive behavior effectively.
- **Chat Filter:** Basic filtering to prevent inappropriate language and maintain a safer environment.
- **User-Friendly Tutorial:** A simple guide for new players to ease them into the game.
- **Enhanced Usability:** Streamlined gameplay and interface for a more engaging and accessible experience.
- **Polished Experience:** Addresses common feedback and issues, leading to a more refined and enjoyable multiplayer experience.

CHAPTER - 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 : Software Requirements:

Operating System	:	Windows 10 (Min)
Front End	:	ReactJS
Back End	:	Javascript (NodeJS & ExpressJS
Browser	:	Latest stable version

3.2 : Hardware Requirements:

Processor	:	Intel Pentium® Dual Core Processor (Min)
Speed	:	2.9 GHz (Min)
RAM	:	2 GB (Min)
Hard Disk	:	2 GB (Min)

CHAPTER-4

SYSTEM DESIGN

4.1 : Use case Diagram:

This use case diagram represents the functionalities available to a player in the "Draw and Guess" game. The player can perform several actions: providing a username, joining a game room, starting a game, drawing a picture, guessing the word based on another player's drawing, sending messages, viewing the leaderboard, and leaving or ending the game. Each oval represents a distinct use case or action the player can take within the game. The diagram highlights the user-centered design, showing the direct interaction between the player and the game's core features.

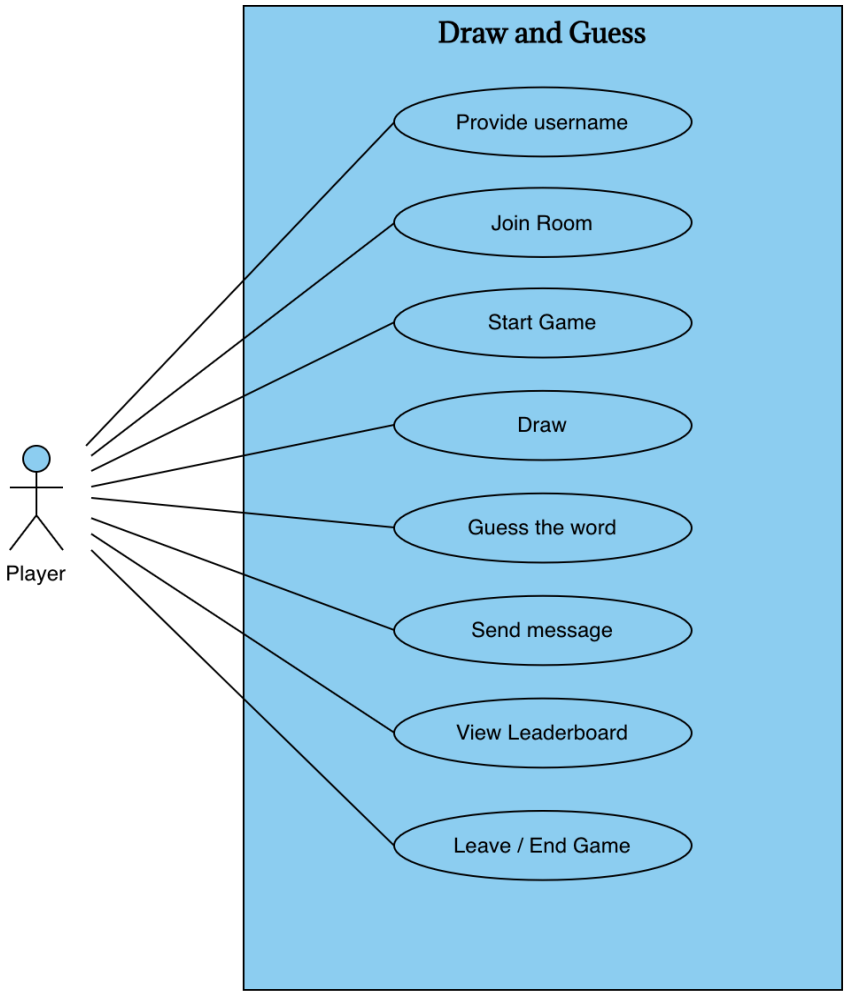


Fig 4.1: Use Case Diagram for Draw and Guess

4.2 : Class Diagram:

The class diagram represents the core components of a multiplayer draw-and-guess game. At the center is the `GameRoom` class, which holds the game's `roomId`, a list of `players`, and the `currentDrawing`. It also controls the game flow with methods to `startGame()` and `endGame()`.

The `Player` class represents each participant in the game, with attributes for their `playerId`, `username`, and `score`. Players can perform actions such as `draw()`, `guess()`, and `sendMessage()`.

The `Drawing` class is responsible for managing the canvas where the drawing takes place, with methods like `drawShape()` and `clearCanvas()`.

Lastly, the `Chat` class handles player communication, storing a list of messages and allowing players to send and receive messages. The diagram illustrates the relationships between these components, showing how they work together to facilitate gameplay.

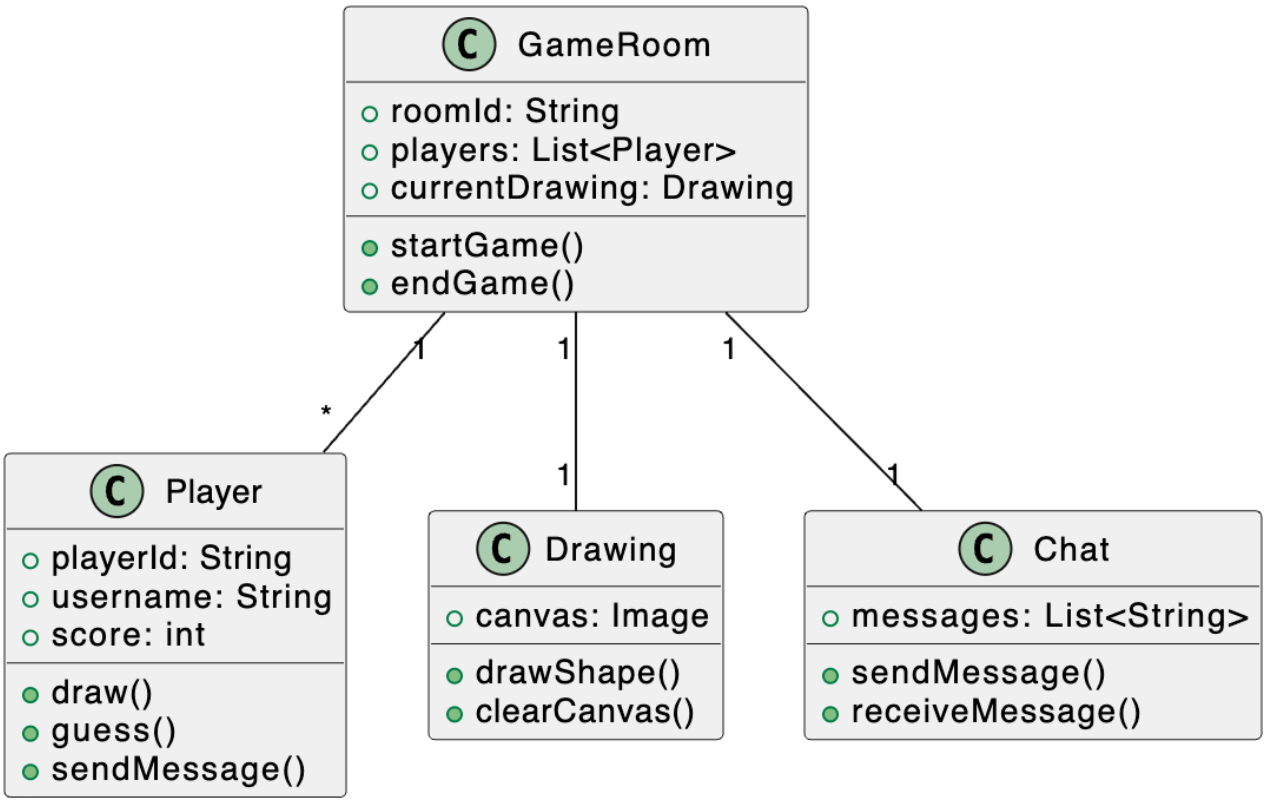


Fig 4.2: Class Diagram for Draw and Guess

4.3 : Sequence Diagram:

The sequence diagram outlines the interaction process in a multiplayer draw-and-guess game, built using the MERN stack. The diagram begins with the player sending a request to join a room, which is processed by the GameServer. The GameServer then assigns the player to a specific room, which is confirmed back to the player. Once in the room, the player can start drawing, and this drawing is broadcasted to other players in the room. Subsequently, players make guesses, and these guesses are broadcasted as well. When a correct guess is made, the leaderboard is updated, and the updated scores are displayed to all players. The flow ensures synchronized communication between the player, GameServer, room, and leaderboard components, facilitating real-time gameplay.

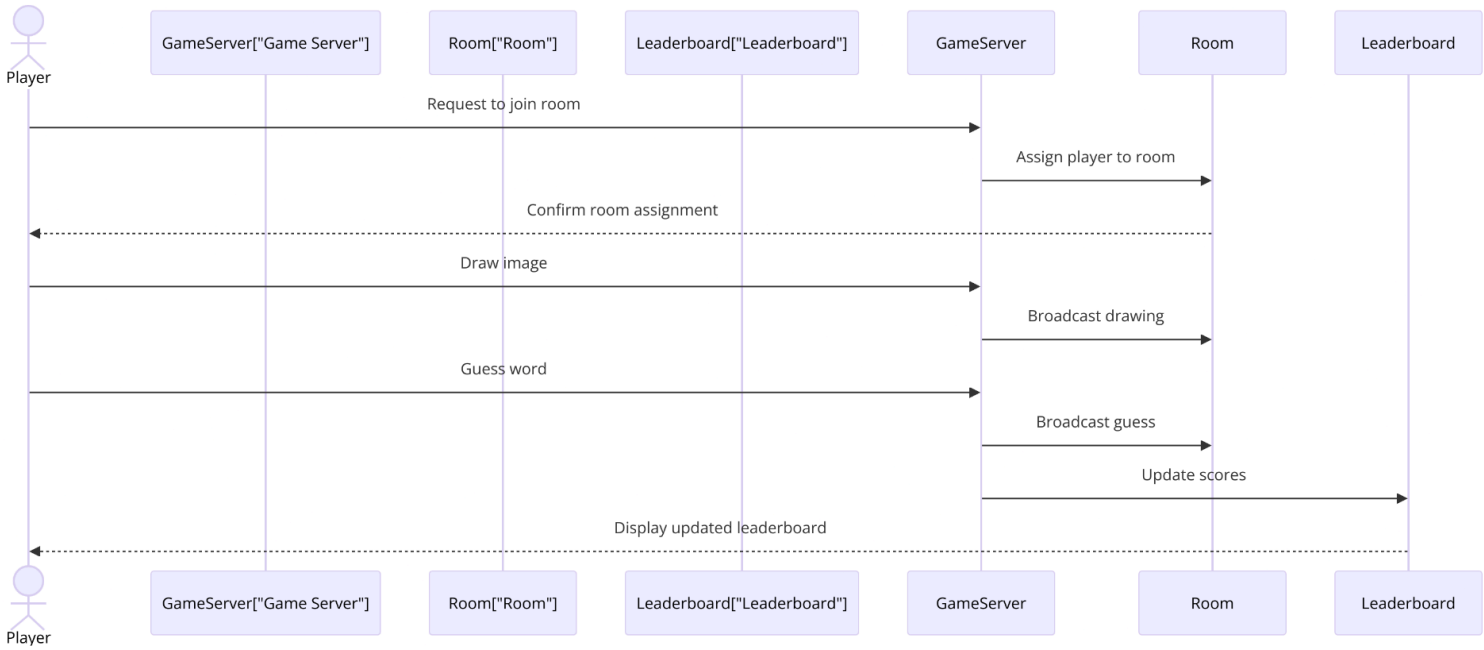


Fig 4.3: Sequence Diagram for Draw and Guess

4.4 :Architecture Of Draw and Guess Game System

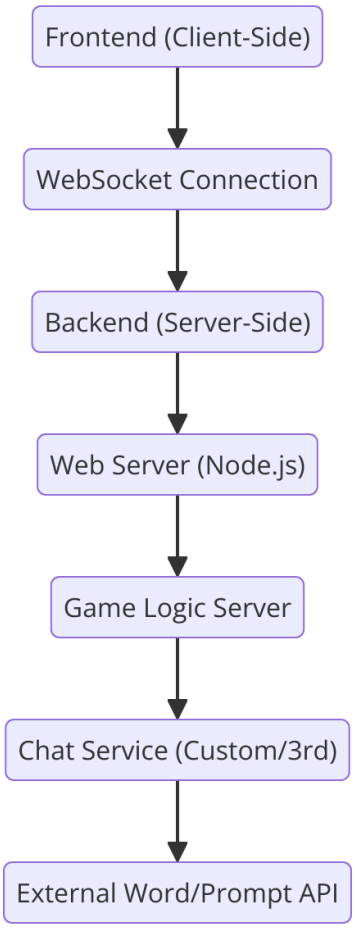


Fig4.4: Architecture of Draw and Guess Game

CHAPTER – 5

IMPLEMENTATION

5.1 SAMPLE CODE
CLIENT CODE:

```
import React from 'react'
import {BrowserRouter , Routes , Route} from "react-router-dom";
import Join from './Components/JoinRoom/Join';
import Main from './Components/Main/Main';
export default function App() {
  return (
    <>
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Join/>}></Route>
          <Route path="/room" element={<Main/>}/>
        </Routes>
      </BrowserRouter>
    </>
  )
}
```

SERVER CODE:

```
const express = require("express");
const app = express();
const http = require("http");
const cors = require("cors");
const { User } = require("./database/schema");
app.use(cors());
const { Server } = require("socket.io");
const server = http.createServer(app);
const io = new Server(server, {
    cors: {
        origin: "http://localhost:5173",
    }
});
app.get('/userList', async (req, res) => {
    try {
        const users = await User.find();
```

```
        res.json(users);
    } catch (error) {
        res.status(500).send(error.message);
    }
});
app.get("/" , (req , response)=>{
    response.send("Live Now")
})
io.on("connection", (client) => {
    console.log(`New user ::${client.id}`);
    client.on("disconnect", async() => {
        console.log("user disconnected ::", client.id);
        try{
            await User.deleteOne({webSocketID : client.id});
            console.log("user deleted ::", client.id);
            io.emit("newPlayer");
        }
        catch(error){
            console.log(" error :: " , error);
        }

    });

    client.on("join-room", async (info) => {
        const { room, name } = info;
        console.log(`User ${client.id} joined room: ${room}`);
        client.join(room);
        try {
            await User.create({
                userName: name,
                webSocketID: client.id,
                points: 0,
                room,
            })
        }
        catch (error) {
            console.log("Something went wrong");
        }
        io.to(room).emit("newPlayer");
    });
    client.on("draw", ({ room, offsetX, offsetY, color }) => {
        io.to(room).emit("draw", { offsetX, offsetY, color });
```

```
});
client.on("stopDrawing", (room) => {
   io.to(room).emit("stopDrawing", room);
});
client.on("clear", ({ room, width, height }) => {
   io.to(room).emit("clear", { width, height });
});
// todo :: chatting
client.on("sendMessage", (info) => {
   console.log("sending message", info);
   console.log("room is ::", info.room);
   const room = info.room;
   console.log(typeof (room));
   io.emit("receiveMessage", info);
})
// todo :: choosing the players
client.on('myEvent', (currentIteration) => {
   console.log(`Received from ${client.id}:`, currentIteration);
   io.emit('acknowledgement', currentIteration);
});
// todo :: word to find
client.on("wordToGuess", (info) => {
   console.log("info is :: ", info)
   io.emit("wordToGuess", info);
})
// todo :: updating points
client.on("updatePlayerPoints", async ({ name, drawTime }) => {
   console.log("in the server side for updating with :: ", name, drawTime);
   try {
      const updatedUser = await User.findOneAndUpdate(
         { userName: name },
         { $inc: { points: 10 * drawTime } },
         { new: true }
      );
      if (!updatedUser) {
         console.log("User not found :: ", name);
         return;
      }
      console.log("Updated user:", updatedUser);
   } catch (error) {
      console.error("Error updating user:", error.message);
   }
```

```
        io.emit("updatePlayerPoints", "info");
    })
    // todo :: client reloaded, moved back, closed the page
    client.on("disconnectUser", async ({ name, room }) => {
        try {
            await User.deleteOne({ userName: name, room: room });
            console.log(`User ${name} has been deleted from room ${room}`);
        }
        catch (error) {
            console.log('Error deleting user:', error);
        }
        io.to(room).emit("newPlayer");
    })
});
const PORT = process.env.PORT || 3000;
server.listen(3000 , () => {
    console.log("Server Running on port 3000");
});
```

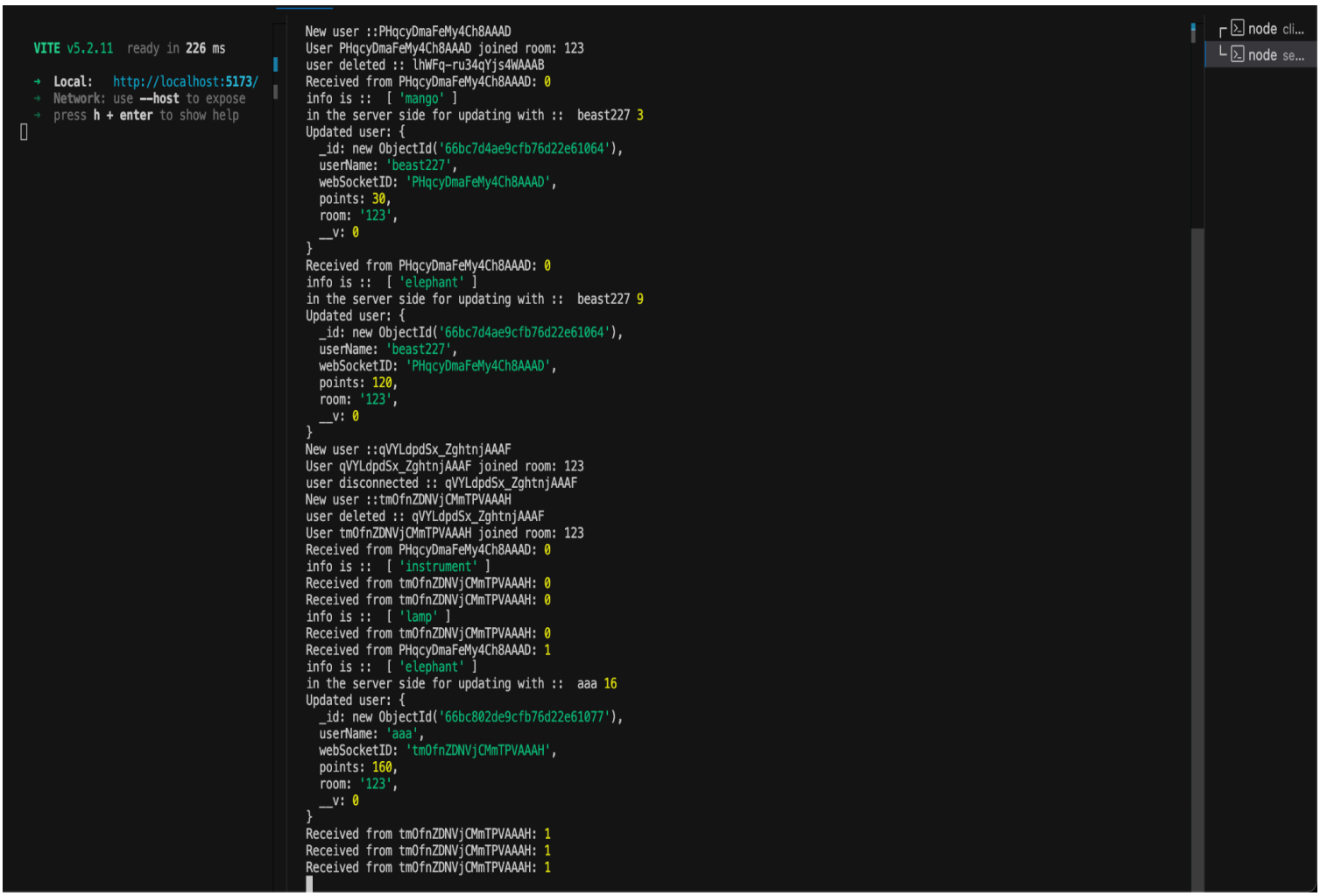
CHAPTER – 6

TESTING

6.1 TEST CASES

Component	Test Case	Test Case Result
Room Creation	Create a private room with custom settings	Room is created with specified settings and a unique join code
Player Join	Join a public room	Player successfully enters the room and appears in the player list
Word Generation	Generate a word for the drawing round	A random word from the database is selected and displayed to the drawer
Drawing Tools	Use various drawing tools (brush, eraser, color picker)	Tools function correctly, allowing the player to draw on the canvas
Real-time Drawing	Draw on the canvas and verify updates for other players	Other players see the drawing update in real-time without significant lag
Guessing Mechanism	Type a guess that matches the word	System recognizes the correct guess and awards points accordingly
Timer	Start a round and let the timer run down	Timer counts down accurately and ends the round when it reaches zero
Scoring System	Award points for correct guesses	Points are added to the player's score and reflected on the leaderboard
Round Rotation	Move to the next round with a new drawer	New round starts with the correct player as the drawer and a new word
Chat Function	Send a message in the chat	Message appears in the chat window for all players to see
Leaderboard Update	Complete a game and update the leaderboard	Final scores are calculated correctly and the leaderboard is updated
Custom Room Settings	Change round duration and word difficulty	New settings are applied in the next round
Disconnect Handling	Simulate a player disconnection	Game handles the disconnection gracefully, removing the player from the room
Reconnect Feature	Reconnect a disconnected player to an ongoing game	Player rejoins the game with their previous score intact
Mobile Responsiveness	Access the game on a mobile device	Game interface adapts to the mobile screen size and remains fully functional

6.2: Testing:


VITE v5.2.11 ready in 226 ms
Local: http://localhost:5173/
Network: use --host to expose
press h + enter to show help
New user ::PHqcyDmaFeMy4Ch8AAAD
User PHqcyDmaFeMy4Ch8AAAD joined room: 123
user deleted :: lhWFq-ru34qYjs4WAAAB
Received from PHqcyDmaFeMy4Ch8AAAD: 0
info is :: ['mango']
in the server side for updating with :: beast227 3
Updated user: {
_id: new ObjectId('66bc7d4ae9cfb76d22e61064'),
userName: 'beast227',
webSocketID: 'PHqcyDmaFeMy4Ch8AAAD',
points: 30,
room: '123',
__v: 0
}
Received from PHqcyDmaFeMy4Ch8AAAD: 0
info is :: ['elephant']
in the server side for updating with :: beast227 9
Updated user: {
_id: new ObjectId('66bc7d4ae9cfb76d22e61064'),
userName: 'beast227',
webSocketID: 'PHqcyDmaFeMy4Ch8AAAD',
points: 120,
room: '123',
__v: 0
}
New user ::qVYLdpdSx_ZghtnjAAAF
User qVYLdpdSx_ZghtnjAAAF joined room: 123
user disconnected :: qVYLdpdSx_ZghtnjAAAF
New user ::tmOfnZDNVjCMmTPVAAAH
user deleted :: qVYLdpdSx_ZghtnjAAAF
User tmOfnZDNVjCMmTPVAAAH joined room: 123
Received from PHqcyDmaFeMy4Ch8AAAD: 0
info is :: ['instrument']
Received from tmOfnZDNVjCMmTPVAAAH: 0
Received from tmOfnZDNVjCMmTPVAAAH: 0
info is :: ['lamp']
Received from tmOfnZDNVjCMmTPVAAAH: 0
Received from PHqcyDmaFeMy4Ch8AAAD: 1
info is :: ['elephant']
in the server side for updating with :: aaa 16
Updated user: {
_id: new ObjectId('66bc802de9cfb76d22e61077'),
userName: 'aaa',
webSocketID: 'tmOfnZDNVjCMmTPVAAAH',
points: 160,
room: '123',
__v: 0
}
Received from tmOfnZDNVjCMmTPVAAAH: 1
Received from tmOfnZDNVjCMmTPVAAAH: 1
Received from tmOfnZDNVjCMmTPVAAAH: 1
node cli...
node se...

CHAPTER - 7
SCREENSHOTS

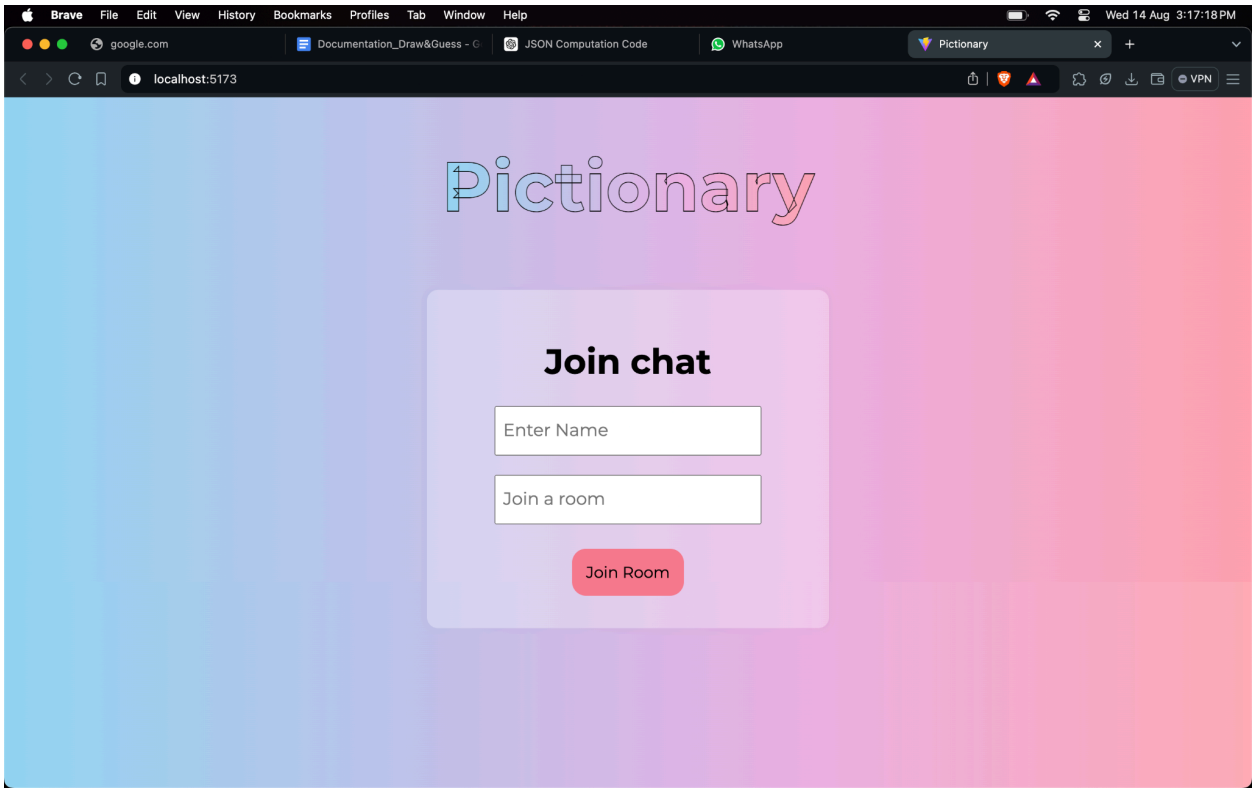


Figure 7.1: HOME PAGE

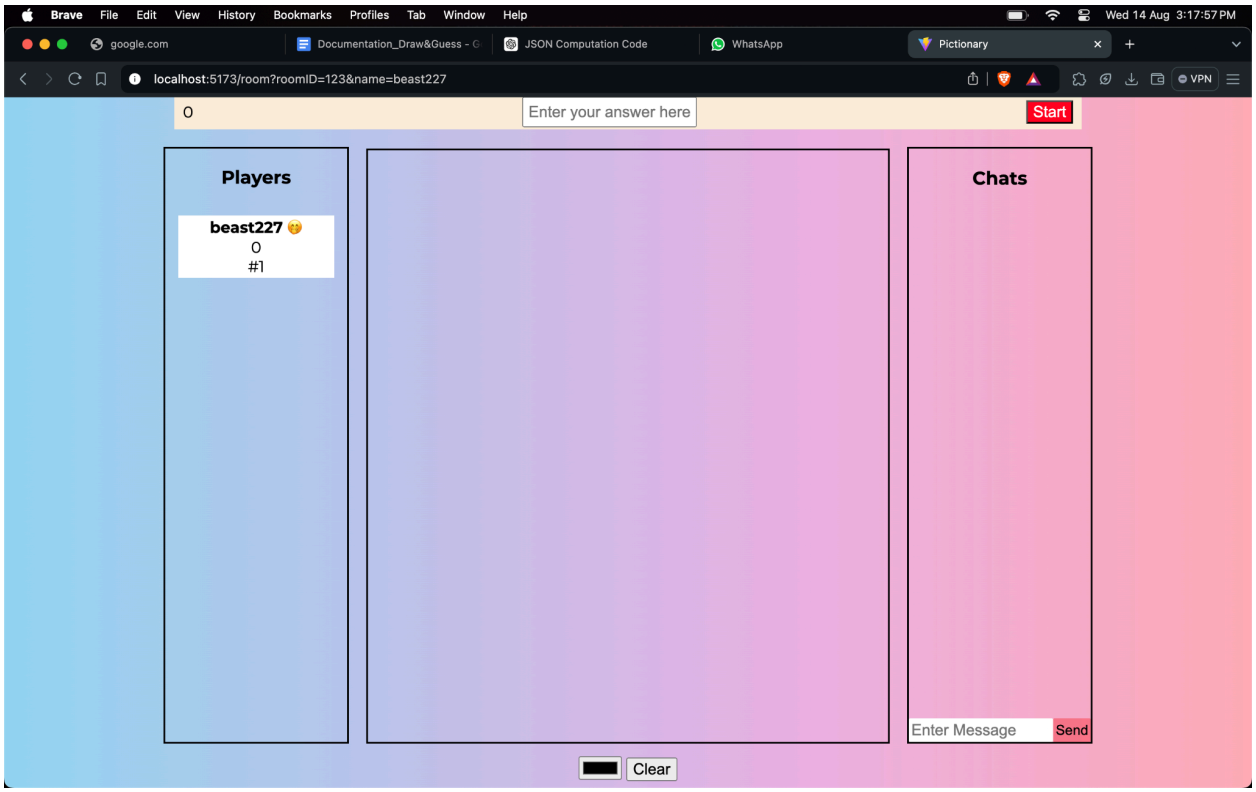


Figure 7.2: Room Joined

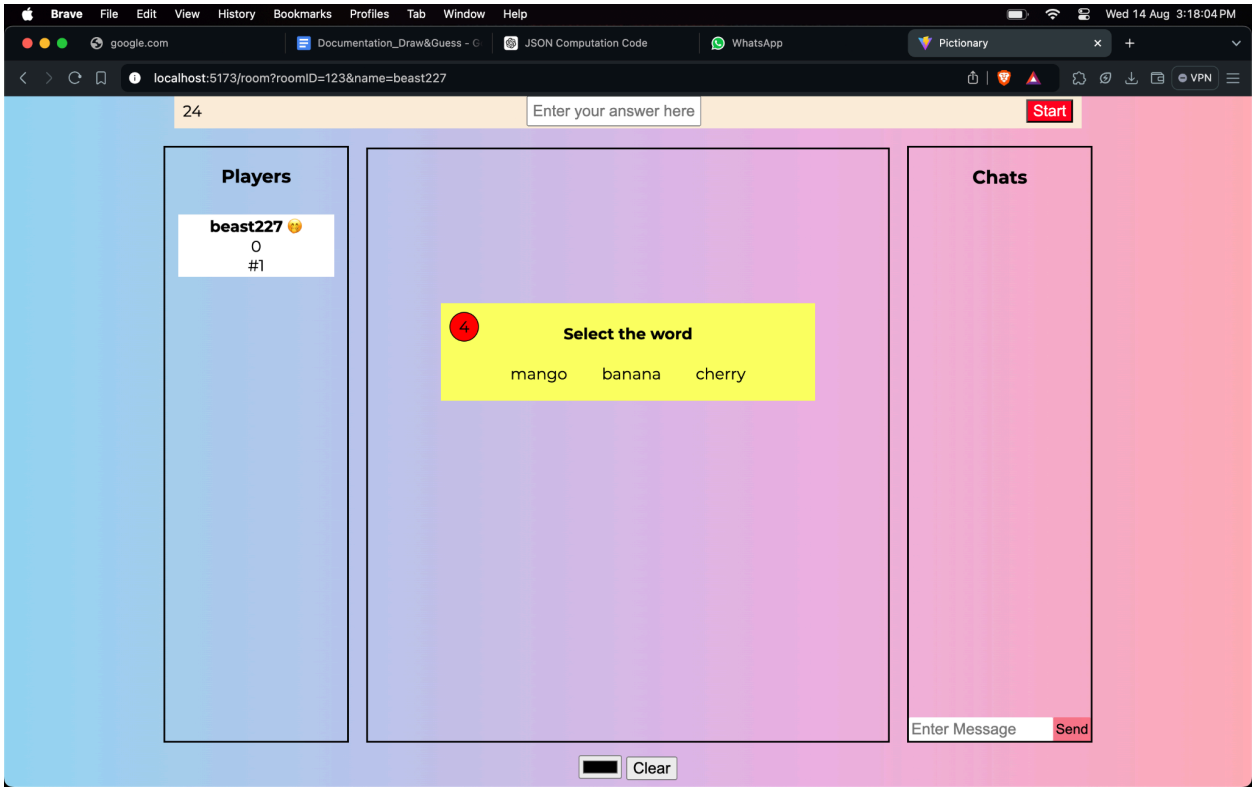


Figure 7.3: Select a word

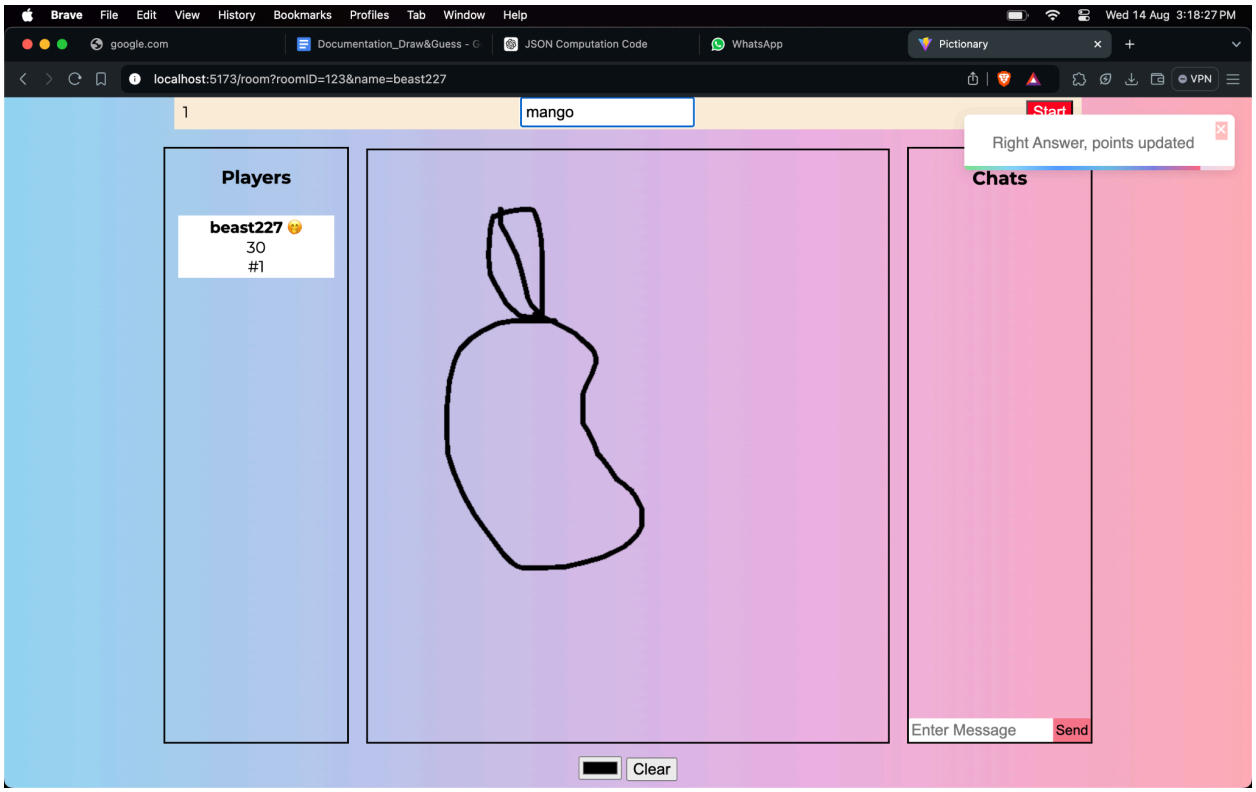


Figure 7.4: Word is drawn and answered correctly

CHAPTER - 8

CONCLUSION AND FUTURE SCOPE

The Draw and Guess game is a fun and interactive way to combine creativity and quick thinking. It allows players to express themselves through drawings while also improving their ability to recognize and interpret visual cues. The game promotes teamwork, communication, and problem-solving skills, making it enjoyable and educational for players of all ages. The implementation of the game can range from simple to complex, with opportunities to incorporate advanced features like scoring systems, timers. The future scope of the Draw and Guess game includes adding more advanced features to enhance the user experience. This could involve integrating AI to guess the drawings or provide hints, allowing for more dynamic and challenging gameplay. Additionally, incorporating online multiplayer options would enable players to connect and compete with others globally. Expanding the game with various themes, levels, and challenges can also keep the game fresh and engaging. The development of mobile apps and cross-platform compatibility would further increase its accessibility and popularity among users.

APPENDIX A: TOOLS AND TECHNOLOGIES

- **ReactJS:** React is a free and open-source front-end JavaScript library for building user interfaces based on components by Facebook Inc.

- **Axios:** Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests.

- **Socket.IO:** Socket.IO is a library that enables low-latency, bidirectional and event-based communication between a client and a server.

- **NodeJS :** Node.js is a cross-platform, open-source JavaScript runtime environment that can run on Windows, Linux, Unix, macOS, and more. Node.js runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser.

- **WINDOWS 11:** Windows 11 was used as the operating system.

- **ExpressJS :** Express.js, or simply Express, is a back end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.