**A PRELIMENERY REPORT ON**

**MULTIPLAYER DOT CHASE GAME**

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

**BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)**

##### SUBMITTED BY

STUDENT NAME Exam Seat No.

Shishir Kumar 71714762C

Swapnil Ahire 71714248F

Saurab Godse 71714730E

Prathamesh Kulkarni 71714508F

Parag Kaldate 71714444F



## DEPARTMENT OF COMPUTER ENGINEERING

**BRACT’S**

**VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY**

SURVEY NO. 3/4, KONDHWA (BUDRUK), PUNE – 411048, MAHARASHTRA (INDIA).

## SAVITRIBAI PHULE PUNE UNIVERSITY

## 2020 -2021



**CERTIFICATE**

This is to certify that the project report entitles

**MULTIPLAYER DOT CHASE GAME**

Submitted by

STUDENT NAME Exam Seat No. :

Shishir Kumar 71714762C

Swapnil Ahire 71714248F

Saurab Godse 71714730E

Prathamesh Kulkarni 71714508F

Parag Kaldate 71714444F

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of **Prof. Vidya Gaikwad** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

**(Prof. Vidya Gaikwad)** **(Dr. S. R. Sakhare)**

Guide Head,

Department of Computer Engineering Department of Computer Engineering

**(Dr. V. S.Deshpande)**

Director

BRACT’s Vishwakarma Institute of Information Technology, Pune-48

Place : Pune

Date :

**ACKNOWLEDGEMENT**

I would like to express my special thanks of gratitude to my

teacher Prof. Vidya Gaikwad as well as our director Dr. V. S. Deshpande who gave me the golden opportunity to do this wonderful project on the topic Multiplayer Dot Chase Game, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.Secondly I would also like to thank my project mates who helped me a lot in finalizing this project within the limited time frame.

**ABSTRACT**

As a genre, multiplayer games are split between two key elements — competitive and cooperative. Multiplayer games bring together real life people into virtual contact. For better or for worse, the competition is heightened. That guy next to you isn’t just some NPC, it’s your real life best friend playing alongside you. Stakes are higher, losses are more bitter, victories harder won.  Multiplayer games have helped gaming go mainstream. Any game that allows more than one player is considered multiplayer. A multiplayer game is most often played over the [Internet](https://www.computerhope.com/jargon/i/internet.htm), however, could be also a game played over a [LAN](https://www.computerhope.com/jargon/l/lan.htm) (local area [network](https://www.computerhope.com/jargon/n/network.htm)) or [dial-up](https://www.computerhope.com/jargon/d/dialup.htm) connection. Due to above mentioned points we developed a multiplayer DOT Game over AWS.

**Table of Contents**

|  |  |
| --- | --- |
| S. No. | Title of chapter |
| 1 | Introduction |
| 1.1 | Purpose |
| 1.2 | Scope |
| 1.3 | Definitions, acronyms, abbrevations |
| 1.4 | Rules |
| 1.5 | References |
| 1.6 | Overview |
| 2. | The overall description |
| 2.1 | Product perspective |
| 2.1.1 | System interfaces |
| 2.1.2 | Software interfaces |
| 2.1.3 | Communication interfaces |
| 2.2 | Product functions |
| 2.3 | User characteristics |
| 2.4 | Assumptions and dependencies |
| 3 | Specific requirements |
| 3.1 | Performance requirements |
| 4. | Gui of application |
| 5 | Conclusion |
| 5.1 | Conclusion |
| 5.2 | Future works |
| 5.3 | Application |

1. INTRODUCTION

1.1 PURPOSE

A **multiplayer game** is a [game](https://en.wikipedia.org/wiki/Video_game) in which more than one person can play in the same game environment at the same time, either locally or [online](https://en.wikipedia.org/wiki/Online_and_offline) over the internet. Multiplayer games usually require players to share the resources of a single game system or use [networking technology](https://en.wikipedia.org/wiki/Mobile_network) to play together over a greater distance; players may compete against one or more human contestants, work [cooperatively](https://en.wikipedia.org/wiki/Cooperative_gameplay) with a human partner to achieve a common goal, [supervise](https://en.wikipedia.org/wiki/Gamemaster) other players' activity, [co-op](https://en.wikipedia.org/wiki/Cooperative_gameplay). Multiplayer games allow players interaction with other individuals in partnership, competition or rivalry, providing them with social communication absent from single-player games.

Games are sometimes played purely for entertainment, sometimes for achievement or reward as well. They can be played alone, in teams, or online; by amateurs or by professionals. The players may have an audience of non-players, such as when people are entertained by watching a [chess championship](https://en.wikipedia.org/wiki/World_Chess_Championship). On the other hand, players in a game may constitute their own audience as they take their turn to play. Often, part of the entertainment for children playing a game is deciding who is part of their audience and who is a player.

1.2 SCOPE

The scope of our project is giving multiplayers a platform to compete, play and make best use of their recreational time.

1.3 DEFINITIONS, ACRONYMS, ABBREVATIONS

Dot: The colourful 2D circle players can control is termed DOT in our Multiplayer Dot Game.

Player Number: The randomly generated number by server above the DOT is Player Number. It is uniquely assigned to each player to identify their targets.

Targets: DOTs having Player Number greater than your DOT is your Target.

Score: Number indicated below dot is score obtained by certain player. Whoever have greatest Score Wins.

Player Name: Your name is visible at right side of DOT, which is often referred as Player Name.

Boundary: The Rectangle visible on screen in which all DOTs are visible is termed to be boundary.

Point: Player gets a point every time they touch their targets. More time they hover over it, more will be the point awarded.

1.4 RULES

1. Players are advised to stay inside the boundary.

2. Players get point only when they touch a valid target.

3. Player with highest score win.

4. Winners are updated at real time.

5. Game is restarted every quarter of an hour (i.e. every 15 mins).

6. Movement keys are keyboard Arrow keys.

1.5 REFERENCES

# 1. How To Build A Multiplayer Browser Game (Part 1)- Elvin Lin

# <https://hackernoon.com/how-to-build-a-multiplayer-browser-game-4a793818c29b>

2. NPM - <https://www.npmjs.com/>

3. HTML - <https://www.w3schools.com/html/>

1.6 OVERVIEW

As the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web) developed and browsers became more

sophisticated, people started creating [browser games](https://en.wikipedia.org/wiki/Browser_game) that used a web browser as a client. Simple single player games were made that could be played using a web browser via HTML and HTML scripting technologies (most commonly JavaScript, [ASP](https://en.wikipedia.org/wiki/Active_Server_Pages), [PHP](https://en.wikipedia.org/wiki/PHP) and [MySQL](https://en.wikipedia.org/wiki/MySQL)). The development of web-based graphics technologies such as [Flash](https://en.wikipedia.org/wiki/Adobe_Flash) and [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) allowed browser games to become more complex. These games, also known by their related technology as "[Flash games](https://en.wikipedia.org/wiki/Browser_game)" or "Java games", became increasingly popular. Browser-based pet games are popular among the younger generation of online gamers. These games range from gigantic games with millions of users, such as *[Neopets](https://en.wikipedia.org/wiki/Neopets" \o "Neopets)*, to smaller and more community-based pet games. More recent browser-based games use web technologies like [Ajax](https://en.wikipedia.org/wiki/Ajax_(programming)) to make more complicated multiplayer interactions possible and [WebGL](https://en.wikipedia.org/wiki/WebGL) to generate hardware-accelerated 3D graphics without the need for plugins.

We’ve created DOT game with help of such browser technologies and other backend technologies to create a real-time game. That is good enough to have fun and play on multi-player environment with friends using LAN.

2. THE OVERALL DESCRIPTION

The game is such developed that anyone can join game using just a browser and is connected to same LAN or internet. On joining every player is added into a multiplayer environment. Where players are added in real-time. Rules are such devised to change the gaming situation time to time. Each player has unique Id and various roles to play when in play. Rules are mentioned in the sections above. Game is easy to understand and can be played using minimal arrow keys. And scoring system is also kept simple to understand. This project aims to develop a prototype for Cloud base.

2.1 PRODUCT PERSPECTIVE

2.1.1 SYSTEM INTERFACES

**Node JS** is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [run-time environment](https://en.wikipedia.org/wiki/Runtime_system) that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting)—running scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm,[[6]](https://en.wikipedia.org/wiki/Node.js#cite_note-6) unifying [web application](https://en.wikipedia.org/wiki/Web_application) development around a single programming language, rather than different languages for server- and client-side scripts.

Though .js is the standard [filename extension](https://en.wikipedia.org/wiki/Filename_extension) for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an [event-driven architecture](https://en.wikipedia.org/wiki/Event-driven_architecture) capable of [asynchronous I/O](https://en.wikipedia.org/wiki/Asynchronous_I/O). These design choices aim to optimize [throughput](https://en.wikipedia.org/wiki/Throughput) and [scalability](https://en.wikipedia.org/wiki/Scalability) in web applications with many input/output operations, as well as for [real-time Web](https://en.wikipedia.org/wiki/Real-time_Web) applications (e.g., [real-time communication](https://en.wikipedia.org/wiki/Real-time_communication) programs and [browser games](https://en.wikipedia.org/wiki/Browser_game)).

**Hypertext Markup Language** (**HTML**) is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language" \o "Markup language) for creating [web pages](https://en.wikipedia.org/wiki/Web_page) and [web applications](https://en.wikipedia.org/wiki/Web_application). With [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [JavaScript](https://en.wikipedia.org/wiki/JavaScript), it forms a triad of cornerstone technologies for the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web).[[4]](https://en.wikipedia.org/wiki/HTML#cite_note-4)

[Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [web server](https://en.wikipedia.org/wiki/Web_server) or from local storage and [render](https://en.wikipedia.org/wiki/Browser_engine) the documents into multimedia web pages. HTML describes the structure of a web page [semantically](https://en.wikipedia.org/wiki/Semantic_Web) and originally included cues for the appearance of the document.

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) are the building blocks of HTML pages. With HTML constructs, [images](https://en.wikipedia.org/wiki/HTML_element#Images_and_objects) and other objects such as [interactive forms](https://en.wikipedia.org/wiki/Fieldset) may be embedded into the rendered page. HTML provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](https://en.wikipedia.org/wiki/Semantics) for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes and other items. HTML elements are delineated by *tags*, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets). Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

**npm** (originally short for **Node Package Manager**)[[3]](https://en.wikipedia.org/wiki/Npm_(software)#cite_note-3) is a [package manager](https://en.wikipedia.org/wiki/Package_manager) for the [JavaScript](https://en.wikipedia.org/wiki/JavaScript) programming language. It is the default package manager for the JavaScript runtime environment [Node.js](https://en.wikipedia.org/wiki/Node.js). It consists of a command line client, also called npm, and an [online database](https://en.wikipedia.org/wiki/Online_database) of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc.

# Amazon Web Services

**Amazon Web Services** (**AWS**) is a [subsidiary](https://en.wikipedia.org/wiki/Subsidiary) of [Amazon](https://en.wikipedia.org/wiki/Amazon.com) that provides [on-demand](https://en.wikipedia.org/wiki/Software_as_a_service) [cloud computing](https://en.wikipedia.org/wiki/Cloud_computing) [platforms](https://en.wikipedia.org/wiki/Computing_platform) and [APIs](https://en.wikipedia.org/wiki/Application_programming_interface) to individuals, companies, and governments, on a metered pay-as-you-go basis. In aggregate, these cloud computing [web services](https://en.wikipedia.org/wiki/Web_services) provide a set of primitive abstract technical infrastructure and [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing) building blocks and tools. One of these services is [Amazon Elastic Compute Cloud](https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud), which allows users to have at their disposal a [virtual](https://en.wikipedia.org/wiki/Virtualization) [cluster of computers](https://en.wikipedia.org/wiki/Computer_cluster), available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer, including hardware [central processing units](https://en.wikipedia.org/wiki/Central_processing_unit) (CPUs) and [graphics processing units](https://en.wikipedia.org/wiki/Graphics_processing_unit) (GPUs) for processing; local/[RAM](https://en.wikipedia.org/wiki/Random-access_memory) memory; hard-disk/[SSD storage](https://en.wikipedia.org/wiki/Solid-state_drive); a choice of operating systems; networking; and pre-loaded application software such as [web servers](https://en.wikipedia.org/wiki/Web_server), [databases](https://en.wikipedia.org/wiki/Database), and [customer relationship management](https://en.wikipedia.org/wiki/Customer_relationship_management) (CRM).

The AWS [technology](https://en.wikipedia.org/wiki/Technology) is implemented at [server farms](https://en.wikipedia.org/wiki/Server_farm) throughout the world, and maintained by the Amazon subsidiary. Fees are based on a combination of usage (known as a "Pay-as-you-go" model), the hardware/OS/software/networking features chosen by the subscriber, required [availability](https://en.wikipedia.org/wiki/Availability_(system)), [redundancy](https://en.wikipedia.org/wiki/Redundancy_(engineering)), [security](https://en.wikipedia.org/wiki/Computer_security), and service options. Subscribers can pay for a single virtual AWS computer, a dedicated physical computer, or clusters of either. As part of the subscription agreement,[[5]](https://en.wikipedia.org/wiki/Amazon_Web_Services#cite_note-AWSagreement-5) Amazon provides security for subscribers' system. AWS operates from many global geographical regions including 6 in [North America](https://en.wikipedia.org/wiki/North_America).[[6]](https://en.wikipedia.org/wiki/Amazon_Web_Services#cite_note-GlobalInfrastructure-6)

# Amazon Elastic Compute Cloud

**Amazon Elastic Compute Cloud** (**EC2**) forms a central part of [Amazon.com](https://en.wikipedia.org/wiki/Amazon.com" \o "Amazon.com)'s [cloud-computing](https://en.wikipedia.org/wiki/Cloud-computing) platform, [Amazon Web Services](https://en.wikipedia.org/wiki/Amazon_Web_Services) (AWS), by allowing users to rent [virtual computers](https://en.wikipedia.org/wiki/Virtual_computer) on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a [web service](https://en.wikipedia.org/wiki/Web_service) through which a user can boot an [Amazon Machine Image](https://en.wikipedia.org/wiki/Amazon_Machine_Image) (AMI) to configure a [virtual machine](https://en.wikipedia.org/wiki/Virtual_Machine), which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate [server](https://en.wikipedia.org/wiki/Server_(computing))-instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for [latency](https://en.wikipedia.org/wiki/Latency_(disambiguation)) optimization and high levels of [redundancy](https://en.wikipedia.org/wiki/Redundancy_(engineering)).[[1]](https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud#cite_note-1)

In November 2010, Amazon switched its own retail website to use EC2 and AWS.[[2]](https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud#cite_note-2)

2.1.2 SOFTWARE INTERFACES

Any browser. Chrome is Recommended though.

2.1.3 COMMUNICATION INTERFACES

A **website** is a collection of related [web pages](https://en.wikipedia.org/wiki/Web_page), including [multimedia](https://en.wikipedia.org/wiki/Multimedia) content, typically identified with a common [domain name](https://en.wikipedia.org/wiki/Domain_name), and published on at least one [web server](https://en.wikipedia.org/wiki/Web_server). Notable examples are [wikipedia.org](https://en.wikipedia.org/wiki/Wikipedia), [google.com](https://en.wikipedia.org/wiki/Google), and [amazon.com](https://en.wikipedia.org/wiki/Amazon_(company)). Today roughly 380 new websites are created every minute across the World.

A website may be accessible via a public [Internet Protocol](https://en.wikipedia.org/wiki/Internet_Protocol) (IP) network, such as the [Internet](https://en.wikipedia.org/wiki/Internet), or a private [local area network](https://en.wikipedia.org/wiki/Local_area_network) (LAN), by referencing a [uniform resource locator](https://en.wikipedia.org/wiki/URL) (URL) that identifies the site.

Websites can have many functions and can be used in various fashions; a website can be a [personal website](https://en.wikipedia.org/wiki/Personal_website), a corporate website for a company, a government website, an organization website, etc. Websites are typically dedicated to a particular topic or purpose, ranging from entertainment and [social networking](https://en.wikipedia.org/wiki/Social_networking) to providing news and education. All publicly accessible websites collectively constitute the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), while private websites, such as a company's website for its employees, are typically a part of an [intranet](https://en.wikipedia.org/wiki/Intranet).

Web pages, which are the [building blocks](https://en.wikipedia.org/wiki/Building_block) of websites, are [documents](https://en.wikipedia.org/wiki/Document), typically composed in [plain text](https://en.wikipedia.org/wiki/Plain_text) interspersed with formatting instructions of Hypertext Markup Language ([HTML](https://en.wikipedia.org/wiki/HTML), [XHTML](https://en.wikipedia.org/wiki/XHTML)). They may incorporate elements from other websites with suitable [markup anchors](https://en.wikipedia.org/wiki/HTML_anchor" \o "HTML anchor). Web pages are accessed and transported with the [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (HTTP), which may optionally employ encryption ([HTTP Secure](https://en.wikipedia.org/wiki/HTTP_Secure), HTTPS) to provide security and privacy for the user. The user's application, often a [web browser](https://en.wikipedia.org/wiki/Web_browser), renders the page content according to its HTML markup instructions onto a [display terminal](https://en.wikipedia.org/wiki/Computer_monitor).

2.2 PRODUCT FUNCTIONS

NodeJS is used as a back end and a server for the multiplayer DOT game. It is used to render canvas again and again for each change. Score of all players is stored in Node Server. Same is sent to all player clients. Use of Server sockets and Client Sockets is used to achieve the same. Various emitter and listeners are defined in Node server code to handle the moves made by players in the game.

2.3 USER CHARACTERISTICS

User should adhere to the given boundary.

2.4 ASSUMPTIONS AND DEPENDENCIES

Node Dependencies:

1. Express ^4
2. Node-cron ^2
3. Socket.io ^1.7

Once server is started it listens on port 5000 of the server pc. So for LAN player needs the IP of server PC and needs to be on same network as the server PC. In case of AWS Public dns is required and game runs on port 5000.

3. SPECIFIC REQUIREMENTS

3.1 PERFORMANCE REQUIREMENTS

**Transparency** :

1. Access Transparent: All other players’ data is shared to all players and how this is achieved is hidden from users. Other user’s real time data is shared with every player of game using only joining link and player doesn’t have any knowledge about how that data is stored over various nodes of network. So, our system is access transparent.
2. Location Transparent: Player doesn’t have knowledge about which server is serving him at a given time and at which server or network node his files/data is stored. So, our system is location transparent.
3. Replication Transparency: Data of each player is replicated on every network node. User doesn’t know how many copies of his data there is.
4. Concurrency Transparency: The shared data i.e. location coordinates of all player are accessed by all players at same time. They can do so and this doesn’t lead to any denial of service to other user. Thus concurrent system.
5. Failure Transparency: Failure of one or more server node doesn’t lead to gaming network failure. Other servers serve the players without giving any hint of change to the players.

**Openness**

1. Portability: Any hardware having nodeJS & npm installed can be used as server. Server doesn’t perform differently on different OS given they have same version of nodeJS. Hence portable.
2. Extensibility: NodeJS supports lots of packages hence various new services can be easily implemented without changing any above mentioned properties.

**Scalability**:

The user load can be distributed among various servers on server machine. Servers can be easily added to our system. When there is increase in number of users on a particular server or crashes other server comes into play.

**Reliability**:

Compared to a single system, a distributed system is highly capable of being secure, consistent and have a high capability of masking errors.

**Performance**: Compared to other models, distributed models give a much-wanted boost to performance.

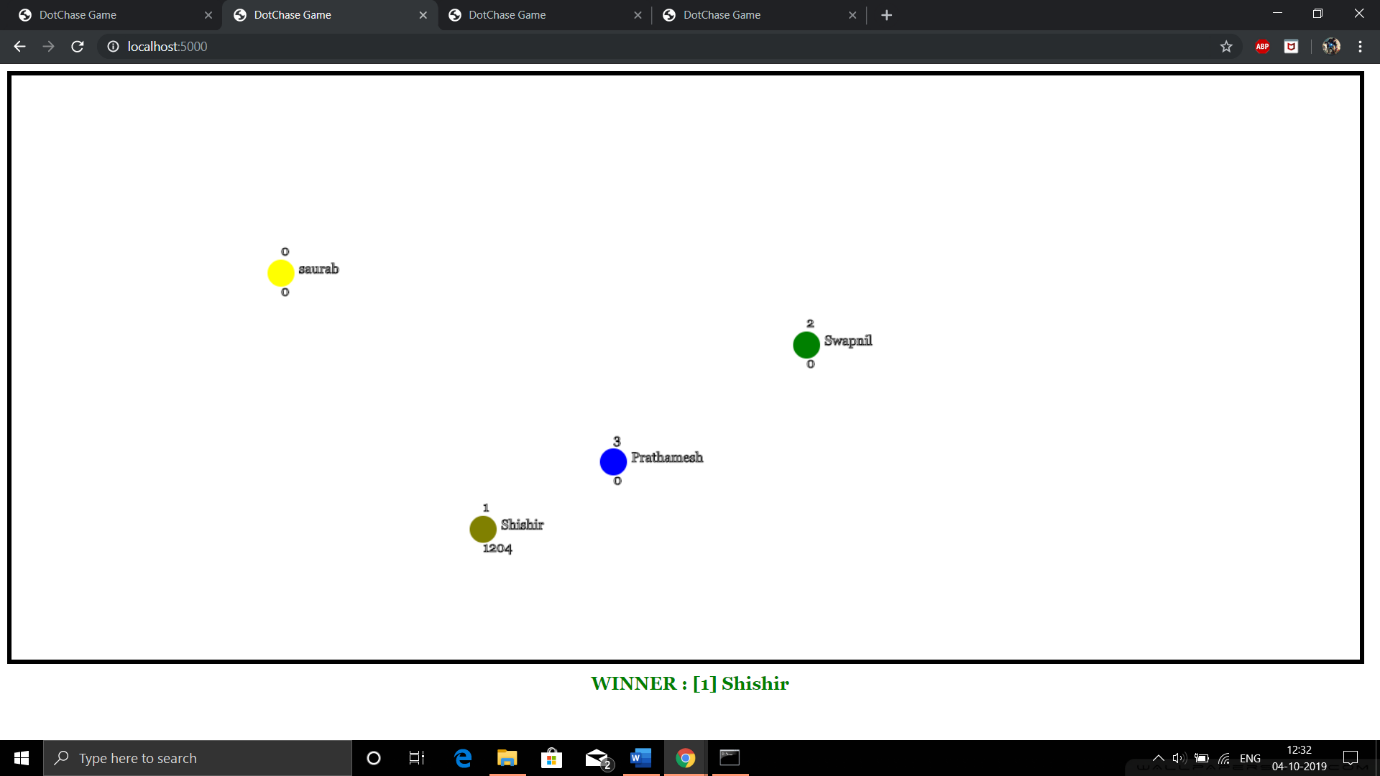
Challenges :

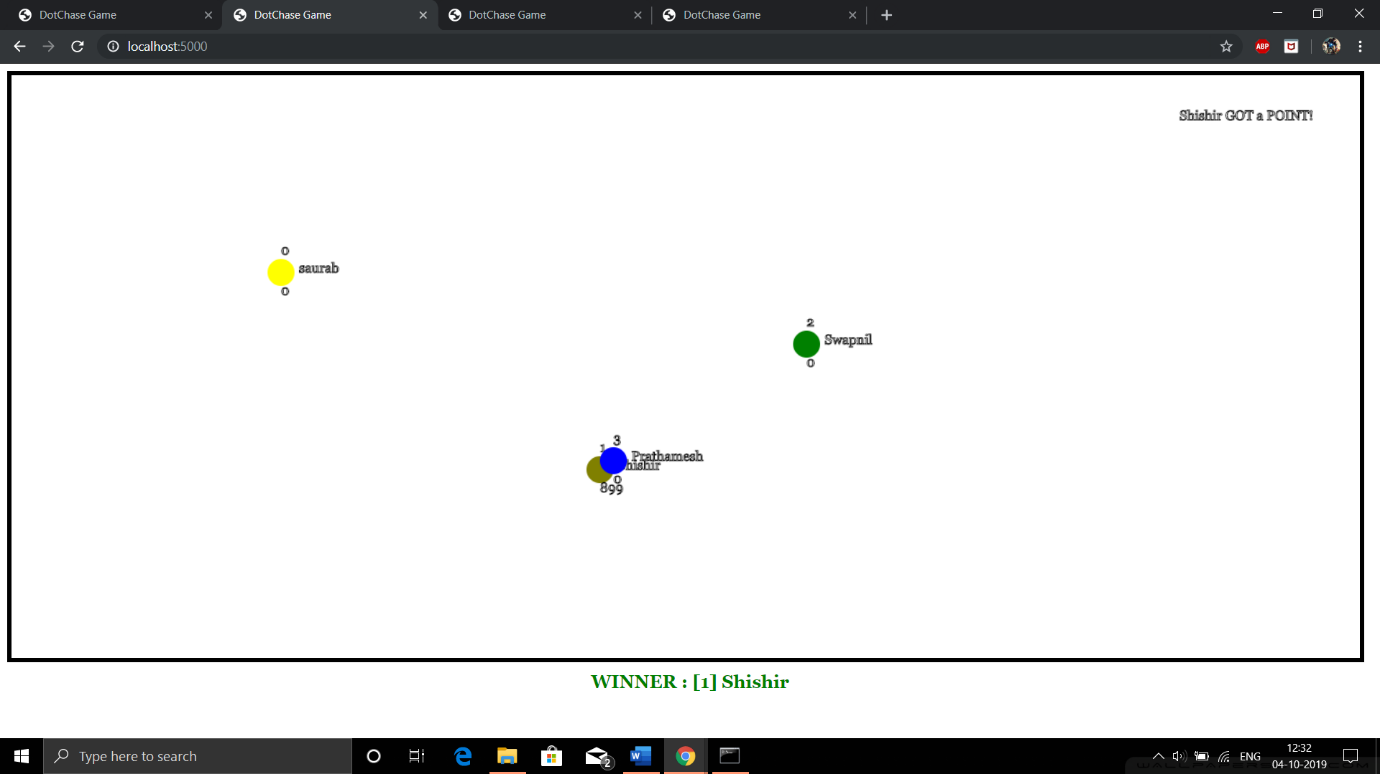
Security is a big challenge in a distributed environment, especially when using public networks.

* Fault tolerance could be tough when the distributed model is built based on unreliable components.

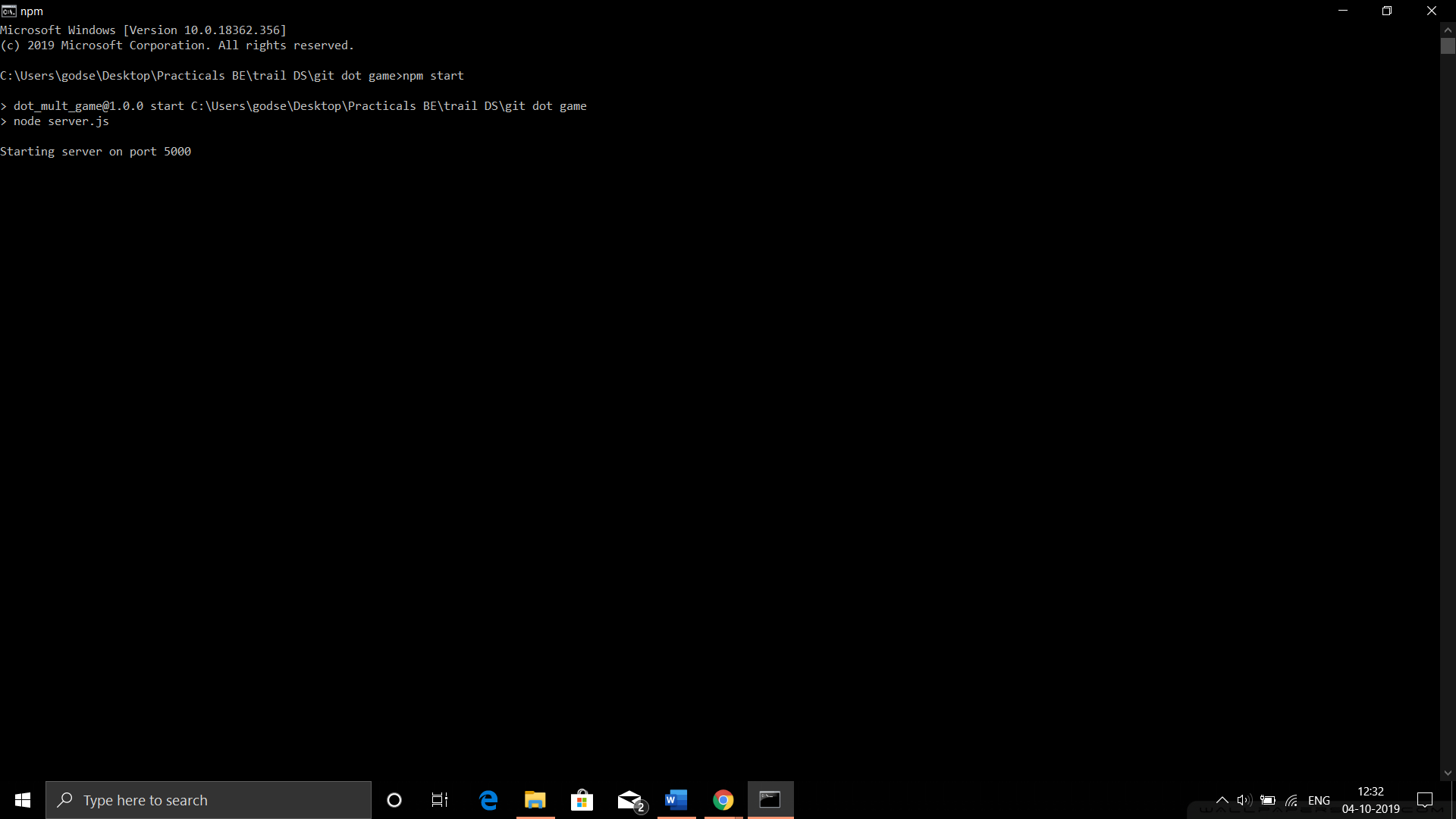
4. GUI of Application (DOT Game):

Players’ Console





Server



5. CONCLUSION

**Conclusion:**

By creating a small game based on distributed architecture, we learned and understood the key features and goal of distributed systems like Transparency, Openness, Reliability, Performance, Scalability etc. Also tackled some common problems faced by distributed systems like Security, fault tolerance, coordination and resource sharing between nodes of distributed network. Also understood the advantages of distributed system over centralised system. Thus we conclude that for a given setup distributed system will be far better than centralised system for various important reasons.

**Future Works** :

Better handling of traffic at cloud server requires some load balancers like NGINX, etc.

Which will improve user experience and performance.

**Application:**

Recreational purposes

1. Code Snippet:

<https://github.com/godsesaurab/dotGameDS/tree/newVersion>

**SERVER**

// Dependencies.

var express = require('express');

var http = require('http');

var path = require('path');

var socketIO = require('socket.io');

var app = express();

var server = http.Server(app);

var io = socketIO(server);

var cron = require('node-cron');

cron.schedule('\*/15 \* \* \* \*', () => {

players = {};

count =0;

//console.log('restarting server every fifteen minutes');

});

app.set('port', 5000);

app.use('/static', express.static(\_\_dirname + '/static'));

// Routing

app.get('/', function(request, response) {

response.sendFile(path.join(\_\_dirname, 'index.html'));

});

server.listen(5000, function() {

console.log('Starting server on port 5000');

});

var count=0;

var players = {};

io.on('connection', function(socket) {

// console.log("socketId: "+socket.id);

socket.on('new player', function(playerName) {

//var newname=count+' '+playerName;

// console.log(newname);

//console.log(count);

players[socket.id] = {

x: 700,

y: 300,

num:count,

name:playerName,

score:0

};

count++;

});

socket.on('movement', function(data) {

var player = players[socket.id] || {};

if (data.left) {

player.x -=5;

}

if (data.up) {

player.y -=5;

}

if (data.right) {

player.x +=5;

}

if (data.down) {

player.y +=5;

}

});

socket.on('score',function(data){

players[data].score+=1;

});

});

setInterval(function() {

io.sockets.emit('state', players);

}, 1000 / 60);

setInterval(function() {

io.sockets.emit('winner', players);

}, 1000 /60);

**CLIENT**

var socket = io();

var movement = {

up: false,

down: false,

left: false,

right: false

}

document.addEventListener('keydown', function(event) {

switch (event.keyCode) {

case 37: // arrowleft

movement.left = true;

break;

case 38: // arrowup

movement.up = true;

break;

case 39: // arrowright

movement.right = true;

break;

case 40: // arrowDown

movement.down = true;

break;

}

});

document.addEventListener('keyup', function(event) {

switch (event.keyCode) {

case 37: // arrowleft

movement.left = false;

break;

case 38: // arrowup

movement.up = false;

break;

case 39: // arrowright

movement.right = false;

break;

case 40: // arrowdown

movement.down = false;

break;

}

});

socket.emit('new player',playerName);

setInterval(function() {

socket.emit('movement', movement);

}, 1000 / 60);

var canvas = document.getElementById('canvas');

canvas.width = 1500;

canvas.height = 650;

var context = canvas.getContext('2d');

socket.on('state', function(players) {

//console.log(players);

var colours=['silver','grey','red','black','maroon','yellow','olive','green','blue','purple','aqua'];

context.clearRect(0, 0, 1500, 650);

loop1: for(var id1 in players){

var player1 =players[id1];

for(var id2 in players){

var player2=players[id2];

if((id1 != id2) && ((Math.abs(player1.x-player2.x)<20) && (Math.abs(player1.y-player2.y)<20) ))

//if((id1 != id2) && (player1.x==player2.x && player1.y==player2.y) && (player1.x!=700 || player1.y!=300))

{

//console.log(player1.name+' wins !');

context.clearRect(0,0,1500,650);

context.strokeText(player1.name+' GOT a POINT! ',1300,50);

socket.emit('score',id1);

//player1.score=player1.score+1;

break loop1;

}

}

}

for (var id in players) {

var player = players[id];

//console.log(player.name);

context.fillStyle = colours[id.charAt(id.length-1).charCodeAt(0)%10];

context.beginPath();

context.arc(player.x, player.y, 15, 0, 2 \* Math.PI);

context.fill();

context.font="15px Georgia";

//context.fillStyle = colours[id.charAt(id.length-1).charCodeAt(0)%10];

context.strokeText(player.num,player.x,player.y-20);

context.strokeText(player.name,player.x+20,player.y);

context.strokeText(player.score,player.x,player.y+25);

}

});

socket.on('winner',function(players){

var max=0; var playername; var playernum;

for(id in players){

player=players[id];

if(player.score>max){

max=player.score;

playername=player.name;

playernum=player.num;

}

}

if(playername!=undefined)

document.getElementById('winner').innerHTML='WINNER : ['+playernum+'] '+playername;

});