Distributed Systems Experiment 1

Group Members:

1) Pravesh Ganwani – 2018140021
2) Luv Gupta - 2018140024

Application:

SketchPad! - Real Time Canvas Sharing Application (Client - Server Multithreaded Application)

Scenario:

Looking at the current pandemic situation, everything happens virtually over network connections. We have teams working virtually but simultaneously on a single project.

Such teams often face difficulties to express their ideas. This is because of restrictions imposed by virtual boundaries.

Our project “SketchPad!” tries to eliminate these virtual boundaries by bringing in real time canvas which can be shared amongst a team. This will help the team members to not only express their ideas or views virtually but also enhance other team members’ ideas and collaborate on a project.

Working:

1) The p5 JS library provides us with a canvas over the front end. This canvas not only captures mouse strokes but also enables us to use metadata (i.e., width of the stroke, color of the stroke, etc.) along with primary X and Y axis co-ordinates.
2) The back end is implemented using Node.JS over Express Framework. Sockets are primarily used for establishing multi-client single server connections.
3) We have used Socket.IO to enable real-time, bidirectional and event-based communication. Socket.IO is a library that enables communication between the browser and the server. It consists of:
    a) A Node.JS server (implemented using Socket APIs in back end).
    b) A JavaScript Client Library for the browser (CDN in front end).

4) Whenever client logins to the portal, it establishes a socket connection with the back end. The client details get stored in the MySQL database along with the default stroke width and stroke color.
5) The client is given a feature to change the stroke width and the stroke color of the brush. These values are stored as metadata along with X and Y co-ordinates.
6) This metadata is broadcasted to other clients connected to the server. The metadata is interpreted by p5 JS and the same stroke is drawn for every client that is connected to the server.

Rooms & Load Balancing:

1) The server supports multiple clients on a single instance of the server.
2) The server can create separate rooms to host a group of clients & helps them to broadcast the points on the canvas internally inside the room.
3) Load balancing has been established using clusters using sticky socket.
4) The worker clusters can fork the master cluster and instantiate a new instance of the CPU.
5) Redis Adapter has been used for maintaining association between different clusters.

Code:

a) Server-Side Code:

```
let username = '';
let room_code = '';
const http = require('http');
const express = require('express');
const util = require('util');
const app = express();
const fs = require('fs');
const path = require('path');

app.use(express.static('public'));

app.set('port', '3000');

const bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(__dirname + '/public'));
app.set('views', __dirname + '/public');
app.engine('html', require('ejs').renderFile);
```

```
app.set('view engine', 'html');

const server = http.createServer(app);
server.on('listening', () => {
  console.log('Listening on port 3000');
});

var mysql = require('mysql');

var con = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'root',
  database: 'sketchpad_db',
  insecureAuth: true,
});
const query = util.promisify(con.query).bind(con);

con.connect(function (err) {
  if (err) console.log(err);
  console.log('Connected!');
});

// Web sockets
const io = require('socket.io')(server);

io.sockets.on('connection', (socket) => {
  console.log('Client connected: ' + socket.id);
  var sql =
    "UPDATE users SET socket_id = '" +
    socket.id +
    "' WHERE username = '" +
    username +
    "'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log('Socket ID updated');
  });
  socket.join(room_code);
  var sql = "SELECT * FROM USERS WHERE room_code = '" + room_code + "'";
  var users = [];
  con.query(sql, function (err, result, fields) {
    if (err) throw err;
    result.forEach((element) => {
      users.push(element.username);
```

```
    });
    io.in(room_code).emit('user_joined', users);
  });
  socket.on('mouse', (data) => {
    socket.broadcast.to(data.roomCode).emit('mouse', data);
    var sql =
      "UPDATE users SET color = '" +
      data.color +
      "', stroke_width = '" +
      data.strokeWidth +
      "' WHERE socket_id = '" +
      socket.id +
      "'";
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log('Color And Stroke Width Updated' + data.username);
    });
  });
  socket.on('username', (data) => {
    socket.broadcast.to(data.roomCode).emit('username', data);
  });
  socket.on('disconnect', () => {
    console.log('Client has disconnected');
    var sql =
      "UPDATE users SET room_code = '" +
      '' +
      "' WHERE socket_id = '" +
      socket.id +
      "'";
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log('User Disconnected');
      var sql = "SELECT * FROM USERS WHERE room_code = '" + room_code + "'";
      var users = [];
      con.query(sql, function (err, result, fields) {
        if (err) throw err;
        result.forEach((element) => {
          users.push(element.username);
        });
        io.in(room_code).emit('user_disconnected', users);
      });
    });
  });
});
```

```
server.listen('3000');

app.post('/submit', function (req, res) {
  username = req.body.username;
  var val = Math.floor(1000 + Math.random() * 9000);
  room_code = val.toString();
  var color = '#FFFFFF';
  var width = 4;
  var sql = "SELECT * FROM USERS WHERE username = '" + username + "'";
  (async () => {
    try {
      const results = await query(sql);
      if (results.length == 0) {
        sql =
          "INSERT INTO users (username, color, stroke_width, room_code) VALUES ('
" +
          username +
          "', '" +
          color +
          "', '" +
          width +
          "', '" +
          room_code +
          "')";
        con.query(sql, function (err, result) {
          if (err) throw err;
          console.log('New User Inserted');
        });
      } else {
        results.forEach((element) => {
          color = element.color;
          width = element.stroke_width;
          var sql =
            "UPDATE users SET room_code = '" +
            room_code +
            "' WHERE username = '" +
            username +
            "'";
          con.query(sql, function (err, result) {
            if (err) throw err;
            console.log('Room Code Updated');
          });
        });
      }
      console.log(results);
```

```
    } finally {
      console.log(color + ' ' + width);
      res.render('sketchpad.html', {
        color: color,
        width: width,
        username: username,
        roomCode: room_code,
      });
    }
  })();
});

app.post('/join-room', function (req, res) {
  username = req.body.username;
  room_code = req.body.roomCode;
  var color = '#FFFFFF';
  var width = 4;
  var sql = "SELECT * FROM USERS WHERE username = '" + username + "'";
  (async () => {
    try {
      const results = await query(sql);
      if (results.length == 0) {
        sql =
          "INSERT INTO users (username, color, stroke_width, room_code) VALUES ('
" +
          username +
          "', '" +
          color +
          "', '" +
          width +
          "', '" +
          room_code +
          "')";
        con.query(sql, function (err, result) {
          if (err) throw err;
          console.log('New User Inserted');
        });
      } else {
        results.forEach((element) => {
          color = element.color;
          width = element.stroke_width;
          var sql =
            "UPDATE users SET room_code = '" +
            room_code +
            "' WHERE username = '" +
```

```
            username +
            "'";
          con.query(sql, function (err, result) {
            if (err) throw err;
            console.log('Room Code Updated');
          });
        });
      }
      console.log(results);
    } finally {
      console.log(color + ' ' + width);
      res.render('sketchpad.html', {
        color: color,
        width: width,
        username: username,
        roomCode: room_code,
      });
    }
  })();
});

app.get('/attendance', (req, res) => {
  var room_id = req.query.room_code.toString();
  console.log(room_id);
  var sql = "SELECT * FROM USERS WHERE room_code = '" + room_id + "'";
  var users = [];
  (async () => {
    const results = await query(sql);
    results.forEach((element) => {
      users.push(element.username);
    });
    const stream = fs.createWriteStream(
      './uploads/attendance-' + room_id + '.txt'
    );
    stream.once('open', function (fd) {
      users.forEach((element) => {
        stream.write(element + '\n');
      });
      stream.end();
    });
    const file = `${__dirname}/uploads/attendance-` + room_id + `.txt`;
    console.log(file);
    res.download(
      path.join(__dirname, './uploads/attendance-' + room_id + '.txt')
    ); // Set disposition and send it.
```

```
  })();
});
```

b) Client-Side Code:

```
let socket;
let color = '#FFF';
let strokeWidth = 4;
let username = '';
let room_code = '';
let save_button = document.getElementById('save-canvas');
let take_attendance = document.getElementById('take-attendance');
var http = new XMLHttpRequest();

take_attendance.addEventListener('click', () => {
  const rcode = select('#rcode');
  room_code = rcode.elt.outerText
    .replace('Room Code: ', '')
    .replace('(', '')
    .replace(')', '');
  window.open('http://localhost:3000/attendance?room_code=' + room_code);
  // var url = 'http://localhost:3000/attendance';
  // var params = 'room_code=' + room_code;
  // http.open('GET', url + '?' + params, true);
  // http.onreadystatechange = function () {
  //   if (http.readyState == 4 && http.status == 200) {
  //     alert(http.responseText);
  //   }
  // };
  // http.send(null) Good Good Very Good;
  // No Problem At All
});

function setup() {
  // Creating canvas
  const cv = createCanvas(800, 600);
  cv.position(600, 90);
  cv.background(0);
  const rcode = select('#rcode');
  room_code = rcode.elt.outerText
    .replace('Room Code: ', '')
    .replace('(', '')
    .replace(')', '');
  save_button.addEventListener('click', () => {
    saveCanvas(cv, 'Canvas-' + room_code, 'jpg');
  });
```

```
// Start the socket connection
socket = io.connect('http://localhost:3000');

// Callback function
socket.on('mouse', (data) => {
  stroke(data.color);
  strokeWeight(data.strokeWidth);
  line(data.x, data.y, data.px, data.py);
  // text(data.username, data.mouseX + 20, data.mouseY + 20);
});

socket.on('username', (data) => {
  console.log('Here ', data.username);
  textSize(16);
  fill(500);
  strokeWeight(1);
  text(data.username, data.x + 20, data.y + 20);
});

socket.on('user_joined', (users) => {
  const user = select('#user-name');
  username = user.elt.outerText.replace('Welcome, ', '');
  console.log('User Joined: ', users);
  var parent_div = document.getElementById('users');
  parent_div.innerHTML = '';
  users.forEach((user) => {
    var tag = document.createElement('div');
    tag.className = 'mb-1';
    if (username == user) {
      tag.innerHTML = user + ' (You)';
    } else {
      tag.innerHTML = user;
    }
    parent_div.appendChild(tag);
  });
});

socket.on('user_disconnected', (users) => {
  console.log('User Disconnected: ', users);
  var parent_div = document.getElementById('users');
  parent_div.innerHTML = '';
  users.forEach((user) => {
    var tag = document.createElement('div');
    tag.className = 'mb-1';
```

```
      if (username == user) {
        tag.innerHTML = user + ' (You)';
      } else {
        tag.innerHTML = user;
      }
      // var text = document.createTextNode(user);
      // tag.appendChild(text);
      parent_div.appendChild(tag);
    });
  });

  // Getting our buttons and the holder through the p5.js dom
  const color_picker = select('#pickcolor');
  const color_btn = select('#color-btn');
  const color_holder = select('#color-holder');
  color = color_picker.value();
  color_holder.style('background-color', color);
  const stroke_width_picker = select('#stroke-width-picker');
  const stroke_btn = select('#stroke-btn');
  strokeWidth = stroke_width_picker.value();
  // Adding a mousePressed listener to the button
  color_btn.mousePressed(() => {
    // Checking if the input is a valid hex color
    if (/(^#[0-9A-F]{6}$)|(^#[0-9A-F]{3}$)/i.test(color_picker.value())) {
      color = color_picker.value();
      color_holder.style('background-color', color);
    } else {
      console.log('Enter a valid hex value');
    }
  });

  // Adding a mousePressed listener to the button
  stroke_btn.mousePressed(() => {
    const width = parseInt(stroke_width_picker.value());
    if (width > 0) strokeWidth = width;
  });

  const clear_btn = select('#clear-canvas');
  clear_btn.mousePressed(() => {
    clear();
    cv.background(0);
  });
}

function mouseDragged() {
```

```
  const user = select('#user-name');
  const rcode = select('#rcode');
  username = user.elt.outerText.replace('Welcome, ', '');
  room_code = rcode.elt.outerText
    .replace('Room Code: ', '')
    .replace('(', '')
    .replace(')', '');
  // Draw
  stroke(color);
  strokeWeight(strokeWidth);
  line(mouseX, mouseY, pmouseX, pmouseY);
  // Send the mouse coordinates
  sendmouse(mouseX, mouseY, pmouseX, pmouseY, username, room_code);
}

function mouseReleased() {
  const user = select('#user-name');
  const rcode = select('#rcode');
  username = user.elt.outerText.replace('Welcome, ', '');
  room_code = rcode.elt.outerText
    .replace('Room Code: ', '')
    .replace('(', '')
    .replace(')', '');
  textSize(16);
  fill(500);
  strokeWeight(1);
  text(username, mouseX + 20, mouseY + 20);
  // Send the mouse coordinates
  sendusername(mouseX, mouseY, username, room_code);
}

// Sending data to the socket
function sendmouse(x, y, pX, pY, username) {
  const data = {
    x: x,
    y: y,
    px: pX,
    py: pY,
    color: color,
    strokeWidth: strokeWidth,
    username: username,
    roomCode: room_code,
  };

  socket.emit('mouse', data);
```

```
}

function sendusername(x, y, username) {
  const data = {
    x: x,
    y: y,
    color: color,
    strokeWidth: strokeWidth,
    username: username,
    roomCode: room_code,
  };

  socket.emit('username', data);
}
```

c) Load Balancing:

```
const cluster = require('cluster');
const http = require('http');
const { Server } = require('socket.io');
const redisAdapter = require('socket.io-redis');
const numCPUs = 2;
const { setupMaster, setupWorker } = require('@socket.io/sticky');

if (cluster.isMaster) {
  console.log(`Master ${process.pid} is running`);

  const httpServer = http.createServer();
  setupMaster(httpServer, {
    loadBalancingMethod: 'round-robin', // either "random", "round-
robin" or "least-connection"
  });
  httpServer.listen(3000);

  for (let i = 0; i < numCPUs; i++) {
    cluster.fork();
  }

  cluster.on('exit', (worker) => {
    console.log(`Worker ${worker.process.pid} died`);
    cluster.fork();
  });
} else {
  console.log(`Worker ${process.pid} started`);

  const httpServer = http.createServer();
```

```
  const io = new Server(httpServer);
  io.adapter(redisAdapter({ host: 'localhost', port: 6379 }));
  setupWorker(io);

  io.on('connection', (socket) => {
    console.log('Connection Established');
  });
}
```

Output: