Code-:

Index.html-:

```
<html lang="en-GB">
<head>
  <meta charset="utf-8">

  </head>
  <body>
    <div id="gradient"></div>
    <div id="page">
      <div id="Message-Container">
        <div id="message">
          <h1>Congratulations!</h1>
          <p>You are done.</p>
          <p id="moves"></p>
          <input id="okBtn" type="button" onclick="toggleVisablity('Message-
Container')" value="Cool!" />
        </div>
      </div>
      <div id="menu">
        <div class="custom-select">
          <select id="diffSelect">
                    <option value="10">Easy</option>
                    <option value="15">Medium</option>
                    <option value="25">Hard</option>
                    <option value="38">Extreme</option>

                </select>
        </div>
        <input id="startMazeBtn" type="button" onclick="makeMaze()" value="Start"
/>
      </div>
      <div id="view">
        <div id="mazeContainer">
```

```
          <canvas id="mazeCanvas" class="border" height="1100"
width="1100"></canvas>
        </div>
      </div>
    </div>
    <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/jquery.touchswipe/1.6.18/jquery.touch
Swipe.min.js"></script>
    <link rel="styleSheet" href="./css/style.css">
    <script src="./script.js"></script>
</body>
</html>
```

Style.scss-:

```
$menuHeight: 65px+10px;
@mixin transition {
    transition-property: background-color opacity;
    transition-duration: 0.2s;
    transition-timing-function: ease-in-out;
}

html,
body {
    width: 100vw;
    height: 100vh;
    position: fixed;
    padding: 0;
    margin: 0;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
}

#mazeContainer {
    transition-property: opacity;
    transition-duration: 1s;
    transition-timing-function: linear;
    top: $menuHeight;
    opacity: 0;
    display: inline-block;
```

```
    background-color: rgba(0, 0, 0, 0.30);
    margin: auto;

    #mazeCanvas {
        margin: 0;
        display: block;
        border: solid 1px black;
    }
}

input,
select {
    @include transition;
    cursor: pointer;
    background-color: rgba(0, 0, 0, 0.30);
    height: 45px;
    width: 150px;
    padding: 10px;
    border: none;
    border-radius: 5px;
    color: white;
    display: inline-block;
    font-size: 15px;
    text-align: center;
    text-decoration: none;
    appearance: none;
    &:hover {
        background-color: rgba(0, 0, 0, 0.70);
    }
    &:active {
        background-color: black;
    }
    &:focus {
        outline: none;
    }
}


.custom-select {
    display: inline-block;
    select {
        -webkit-appearance: none;
        -moz-appearance: none;
        appearance: none;
```

```
        background-image:
url('
Q4T93TMQrCUAzG8V9x8QziiYSuXdzFC7h4AcELOPQAdXYovZCHEATlgQV5GFTe1ozJlz/kS1IpjKqw3wQ
BVyy++JI0y1GTe7DCBbMAckeNIQKk/BanALBB+16LtnDELoMcsM/BESDlz2heDR3WePwKSLo5eoxz3z6N
NcFD+vu3ij14Aqz/DxGbKB7CAAAAAElFTkSuQmCC');
        background-repeat: no-repeat;
        background-position: 125px center;
    }
}

#Message-Container {
    visibility: hidden;
    color: white;
    display: block;
    width: 100vw;
    height: 100vh;
    position: fixed;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
    background-color: rgba(0, 0, 0, 0.30);
    z-index: 1;
    #message {
        width: 300px;
        height: 300px;
        position: fixed;
        top: 50%;
        left: 50%;
        margin-left: -150px;
        margin-top: -150px;
    }
}

#page {
    font-family: "Segoe UI", Arial, sans-serif;
    text-align: center;
    height: auto;
    width: auto;
    margin: auto;
    #menu {
        margin: auto;
        padding: 10px;
        height: 65px;
        box-sizing: border-box;
```

```
        h1 {
            margin: 0;
            margin-bottom: 10px;
            font-weight: 600;
            font-size: 3.2rem;
        }
    }
    #view {
        position: absolute;
        top:65px;
        bottom: 0;
        left: 0;
        right: 0;
        width: 100%;
        height: auto;

    }
}

.border {
    border: 1px black solid;
    border-radius: 5px;
}



#gradient {
    z-index: -1;
    position: fixed;
    top: 0;
    bottom: 0;
    width: 100vw;
    height: 100vh;
    color: #fff;
    background: linear-gradient(-45deg, #EE7752, #E73C7E, #23A6D5, #23D5AB);
    background-size: 400% 400%;
    animation: Gradient 15s ease infinite;
}

@keyframes Gradient {
    0% {
        background-position: 0% 50%
    }
    50% {
        background-position: 100% 50%
```

```
    }
    100% {
        background-position: 0% 50%
    }
}

 /* Extra small devices (phones, 600px and down) */
 @media only screen and (max-width: 400px) {
     input, select{
         width: 120px;
     }
 }
```

Script.js-:

```
function rand(max) {
    return Math.floor(Math.random() * max);
  }

  function shuffle(a) {
    for (let i = a.length - 1; i > 0; i--) {
      const j = Math.floor(Math.random() * (i + 1));
      [a[i], a[j]] = [a[j], a[i]];
    }
    return a;
  }

  function changeBrightness(factor, sprite) {
    var virtCanvas = document.createElement("canvas");
    virtCanvas.width = 500;
    virtCanvas.height = 500;
    var context = virtCanvas.getContext("2d");
    context.drawImage(sprite, 0, 0, 500, 500);

    var imgData = context.getImageData(0, 0, 500, 500);

    for (let i = 0; i < imgData.data.length; i += 4) {
      imgData.data[i] = imgData.data[i] * factor;
      imgData.data[i + 1] = imgData.data[i + 1] * factor;
      imgData.data[i + 2] = imgData.data[i + 2] * factor;
    }
    context.putImageData(imgData, 0, 0);

    var spriteOutput = new Image();
    spriteOutput.src = virtCanvas.toDataURL();
```

```
    virtCanvas.remove();
    return spriteOutput;
  }

  function displayVictoryMess(moves) {
    document.getElementById("moves").innerHTML = "You Moved " + moves + "
Steps.";
    toggleVisablity("Message-Container");
  }

  function toggleVisablity(id) {
    if (document.getElementById(id).style.visibility == "visible") {
      document.getElementById(id).style.visibility = "hidden";
    } else {
      document.getElementById(id).style.visibility = "visible";
    }
  }

  function Maze(Width, Height) {
    var mazeMap;
    var width = Width;
    var height = Height;
    var startCoord, endCoord;
    var dirs = ["n", "s", "e", "w"];
    var modDir = {
      n: {
        y: -1,
        x: 0,
        o: "s"
      },
      s: {
        y: 1,
        x: 0,
        o: "n"
      },
      e: {
        y: 0,
        x: 1,
        o: "w"
      },
      w: {
        y: 0,
        x: -1,
        o: "e"
      }
```

```
};

this.map = function() {
  return mazeMap;
};
this.startCoord = function() {
  return startCoord;
};
this.endCoord = function() {
  return endCoord;
};

function genMap() {
  mazeMap = new Array(height);
  for (y = 0; y < height; y++) {
    mazeMap[y] = new Array(width);
    for (x = 0; x < width; ++x) {
      mazeMap[y][x] = {
        n: false,
        s: false,
        e: false,
        w: false,
        visited: false,
        priorPos: null
      };
    }
  }
}

function defineMaze() {
  var isComp = false;
  var move = false;
  var cellsVisited = 1;
  var numLoops = 0;
  var maxLoops = 0;
  var pos = {
    x: 0,
    y: 0
  };
  var numCells = width * height;
  while (!isComp) {
    move = false;
    mazeMap[pos.x][pos.y].visited = true;

    if (numLoops >= maxLoops) {
```

```
          shuffle(dirs);
          maxLoops = Math.round(rand(height / 8));
          numLoops = 0;
        }
        numLoops++;
        for (index = 0; index < dirs.length; index++) {
          var direction = dirs[index];
          var nx = pos.x + modDir[direction].x;
          var ny = pos.y + modDir[direction].y;

          if (nx >= 0 && nx < width && ny >= 0 && ny < height) {
            //Check if the tile is already visited
            if (!mazeMap[nx][ny].visited) {
              //Carve through walls from this tile to next
              mazeMap[pos.x][pos.y][direction] = true;
              mazeMap[nx][ny][modDir[direction].o] = true;

              //Set Currentcell as next cells Prior visited
              mazeMap[nx][ny].priorPos = pos;
              //Update Cell position to newly visited location
              pos = {
                x: nx,
                y: ny
              };
              cellsVisited++;
              //Recursively call this method on the next tile
              move = true;
              break;
            }
          }
        }

        if (!move) {
          //  If it failed to find a direction,
          //  move the current position back to the prior cell and Recall the
method.
          pos = mazeMap[pos.x][pos.y].priorPos;
        }
        if (numCells == cellsVisited) {
          isComp = true;
        }
      }
    }

    function defineStartEnd() {
```

```
  switch (rand(4)) {
    case 0:
      startCoord = {
        x: 0,
        y: 0
      };
      endCoord = {
        x: height - 1,
        y: width - 1
      };
      break;
    case 1:
      startCoord = {
        x: 0,
        y: width - 1
      };
      endCoord = {
        x: height - 1,
        y: 0
      };
      break;
    case 2:
      startCoord = {
        x: height - 1,
        y: 0
      };
      endCoord = {
        x: 0,
        y: width - 1
      };
      break;
    case 3:
      startCoord = {
        x: height - 1,
        y: width - 1
      };
      endCoord = {
        x: 0,
        y: 0
      };
      break;
  }
}

genMap();
```

```
  defineStartEnd();
  defineMaze();
}

function DrawMaze(Maze, ctx, cellsize, endSprite = null) {
  var map = Maze.map();
  var cellSize = cellsize;
  var drawEndMethod;
  ctx.lineWidth = cellSize / 40;

  this.redrawMaze = function(size) {
    cellSize = size;
    ctx.lineWidth = cellSize / 50;
    drawMap();
    drawEndMethod();
  };

  function drawCell(xCord, yCord, cell) {
    var x = xCord * cellSize;
    var y = yCord * cellSize;

    if (cell.n == false) {
      ctx.beginPath();
      ctx.moveTo(x, y);
      ctx.lineTo(x + cellSize, y);
      ctx.stroke();
    }
    if (cell.s === false) {
      ctx.beginPath();
      ctx.moveTo(x, y + cellSize);
      ctx.lineTo(x + cellSize, y + cellSize);
      ctx.stroke();
    }
    if (cell.e === false) {
      ctx.beginPath();
      ctx.moveTo(x + cellSize, y);
      ctx.lineTo(x + cellSize, y + cellSize);
      ctx.stroke();
    }
    if (cell.w === false) {
      ctx.beginPath();
      ctx.moveTo(x, y);
      ctx.lineTo(x, y + cellSize);
      ctx.stroke();
    }
```

```
}

function drawMap() {
  for (x = 0; x < map.length; x++) {
    for (y = 0; y < map[x].length; y++) {
      drawCell(x, y, map[x][y]);
    }
  }
}

function drawEndFlag() {
  var coord = Maze.endCoord();
  var gridSize = 4;
  var fraction = cellSize / gridSize - 2;
  var colorSwap = true;
  for (let y = 0; y < gridSize; y++) {
    if (gridSize % 2 == 0) {
      colorSwap = !colorSwap;
    }
    for (let x = 0; x < gridSize; x++) {
      ctx.beginPath();
      ctx.rect(
        coord.x * cellSize + x * fraction + 4.5,
        coord.y * cellSize + y * fraction + 4.5,
        fraction,
        fraction
      );
      if (colorSwap) {
        ctx.fillStyle = "rgba(0, 0, 0, 0.8)";
      } else {
        ctx.fillStyle = "rgba(255, 255, 255, 0.8)";
      }
      ctx.fill();
      colorSwap = !colorSwap;
    }
  }
}

function drawEndSprite() {
  var offsetLeft = cellSize / 50;
  var offsetRight = cellSize / 25;
  var coord = Maze.endCoord();
  ctx.drawImage(
    endSprite,
    2,
```

```
        2,
        endSprite.width,
        endSprite.height,
        coord.x * cellSize + offsetLeft,
        coord.y * cellSize + offsetLeft,
        cellSize - offsetRight,
        cellSize - offsetRight
      );
    }

    function clear() {
      var canvasSize = cellSize * map.length;
      ctx.clearRect(0, 0, canvasSize, canvasSize);
    }

    if (endSprite != null) {
      drawEndMethod = drawEndSprite;
    } else {
      drawEndMethod = drawEndFlag;
    }
    clear();
    drawMap();
    drawEndMethod();
  }

  function Player(maze, c, _cellsize, onComplete, sprite = null) {
    var ctx = c.getContext("2d");
    var drawSprite;
    var moves = 0;
    drawSprite = drawSpriteCircle;
    if (sprite != null) {
      drawSprite = drawSpriteImg;
    }
    var player = this;
    var map = maze.map();
    var cellCoords = {
      x: maze.startCoord().x,
      y: maze.startCoord().y
    };
    var cellSize = _cellsize;
    var halfCellSize = cellSize / 2;

    this.redrawPlayer = function(_cellsize) {
      cellSize = _cellsize;
      drawSpriteImg(cellCoords);
```

```
};

function drawSpriteCircle(coord) {
  ctx.beginPath();
  ctx.fillStyle = "yellow";
  ctx.arc(
    (coord.x + 1) * cellSize - halfCellSize,
    (coord.y + 1) * cellSize - halfCellSize,
    halfCellSize - 2,
    0,
    2 * Math.PI
  );
  ctx.fill();
  if (coord.x === maze.endCoord().x && coord.y === maze.endCoord().y) {
    onComplete(moves);
    player.unbindKeyDown();
  }
}

function drawSpriteImg(coord) {
  var offsetLeft = cellSize / 50;
  var offsetRight = cellSize / 25;
  ctx.drawImage(
    sprite,
    0,
    0,
    sprite.width,
    sprite.height,
    coord.x * cellSize + offsetLeft,
    coord.y * cellSize + offsetLeft,
    cellSize - offsetRight,
    cellSize - offsetRight
  );
  if (coord.x === maze.endCoord().x && coord.y === maze.endCoord().y) {
    onComplete(moves);
    player.unbindKeyDown();
  }
}

function removeSprite(coord) {
  var offsetLeft = cellSize / 50;
  var offsetRight = cellSize / 25;
  ctx.clearRect(
    coord.x * cellSize + offsetLeft,
    coord.y * cellSize + offsetLeft,
```

```
    cellSize - offsetRight,
    cellSize - offsetRight
  );
}

function check(e) {
  var cell = map[cellCoords.x][cellCoords.y];
  moves++;
  switch (e.keyCode) {
    case 65:
    case 37: // west
      if (cell.w == true) {
        removeSprite(cellCoords);
        cellCoords = {
          x: cellCoords.x - 1,
          y: cellCoords.y
        };
        drawSprite(cellCoords);
      }
      break;
    case 87:
    case 38: // north
      if (cell.n == true) {
        removeSprite(cellCoords);
        cellCoords = {
          x: cellCoords.x,
          y: cellCoords.y - 1
        };
        drawSprite(cellCoords);
      }
      break;
    case 68:
    case 39: // east
      if (cell.e == true) {
        removeSprite(cellCoords);
        cellCoords = {
          x: cellCoords.x + 1,
          y: cellCoords.y
        };
        drawSprite(cellCoords);
      }
      break;
    case 83:
    case 40: // south
      if (cell.s == true) {
```

```
        removeSprite(cellCoords);
        cellCoords = {
          x: cellCoords.x,
          y: cellCoords.y + 1
        };
        drawSprite(cellCoords);
      }
      break;
  }
}

this.bindKeyDown = function() {
  window.addEventListener("keydown", check, false);

  $("#view").swipe({
    swipe: function(
      event,
      direction,
      distance,
      duration,
      fingerCount,
      fingerData
    ) {
      console.log(direction);
      switch (direction) {
        case "up":
          check({
            keyCode: 38
          });
          break;
        case "down":
          check({
            keyCode: 40
          });
          break;
        case "left":
          check({
            keyCode: 37
          });
          break;
        case "right":
          check({
            keyCode: 39
          });
          break;
```

```
        }
      },
      threshold: 0
    });
  };

  this.unbindKeyDown = function() {
    window.removeEventListener("keydown", check, false);
    $("#view").swipe("destroy");
  };

  drawSprite(maze.startCoord());

  this.bindKeyDown();
}

var mazeCanvas = document.getElementById("mazeCanvas");
var ctx = mazeCanvas.getContext("2d");
var sprite;
var finishSprite;
var maze, draw, player;
var cellSize;
var difficulty;
// sprite.src = 'media/sprite.png';

window.onload = function() {
  let viewWidth = $("#view").width();
  let viewHeight = $("#view").height();
  if (viewHeight < viewWidth) {
    ctx.canvas.width = viewHeight - viewHeight / 100;
    ctx.canvas.height = viewHeight - viewHeight / 100;
  } else {
    ctx.canvas.width = viewWidth - viewWidth / 100;
    ctx.canvas.height = viewWidth - viewWidth / 100;
  }

  //Load and edit sprites
  var completeOne = false;
  var completeTwo = false;
  var isComplete = () => {
    if(completeOne === true && completeTwo === true)
       {
         console.log("Runs");
         setTimeout(function(){
           makeMaze();
```

```
        }, 500);
      }
  };
  sprite = new Image();
  sprite.src =
    "https://image.ibb.co/dr1HZy/Pf_RWr3_X_Imgur.png" +
    "?" +
    new Date().getTime();
  sprite.setAttribute("crossOrigin", " ");
  sprite.onload = function() {
    sprite = changeBrightness(1.2, sprite);
    completeOne = true;
    console.log(completeOne);
    isComplete();
  };

  finishSprite = new Image();
  finishSprite.src = "https://image.ibb.co/b9wqnJ/i_Q7m_U25_Imgur.png"+
  "?" +
  new Date().getTime();
  finishSprite.setAttribute("crossOrigin", " ");
  finishSprite.onload = function() {
    finishSprite = changeBrightness(1.1, finishSprite);
    completeTwo = true;
    console.log(completeTwo);
    isComplete();
  };

};

window.onresize = function() {
  let viewWidth = $("#view").width();
  let viewHeight = $("#view").height();
  if (viewHeight < viewWidth) {
    ctx.canvas.width = viewHeight - viewHeight / 100;
    ctx.canvas.height = viewHeight - viewHeight / 100;
  } else {
    ctx.canvas.width = viewWidth - viewWidth / 100;
    ctx.canvas.height = viewWidth - viewWidth / 100;
  }
  cellSize = mazeCanvas.width / difficulty;
  if (player != null) {
    draw.redrawMaze(cellSize);
    player.redrawPlayer(cellSize);
  }
```

```
};

function makeMaze() {
  //document.getElementById("mazeCanvas").classList.add("border");
  if (player != undefined) {
    player.unbindKeyDown();
    player = null;
  }
  var e = document.getElementById("diffSelect");
  difficulty = e.options[e.selectedIndex].value;
  cellSize = mazeCanvas.width / difficulty;
  maze = new Maze(difficulty, difficulty);
  draw = new DrawMaze(maze, ctx, cellSize, finishSprite);
  player = new Player(maze, mazeCanvas, cellSize, displayVictoryMess, sprite);
  if (document.getElementById("mazeContainer").style.opacity < "100") {
    document.getElementById("mazeContainer").style.opacity = "100";
  }
}
```