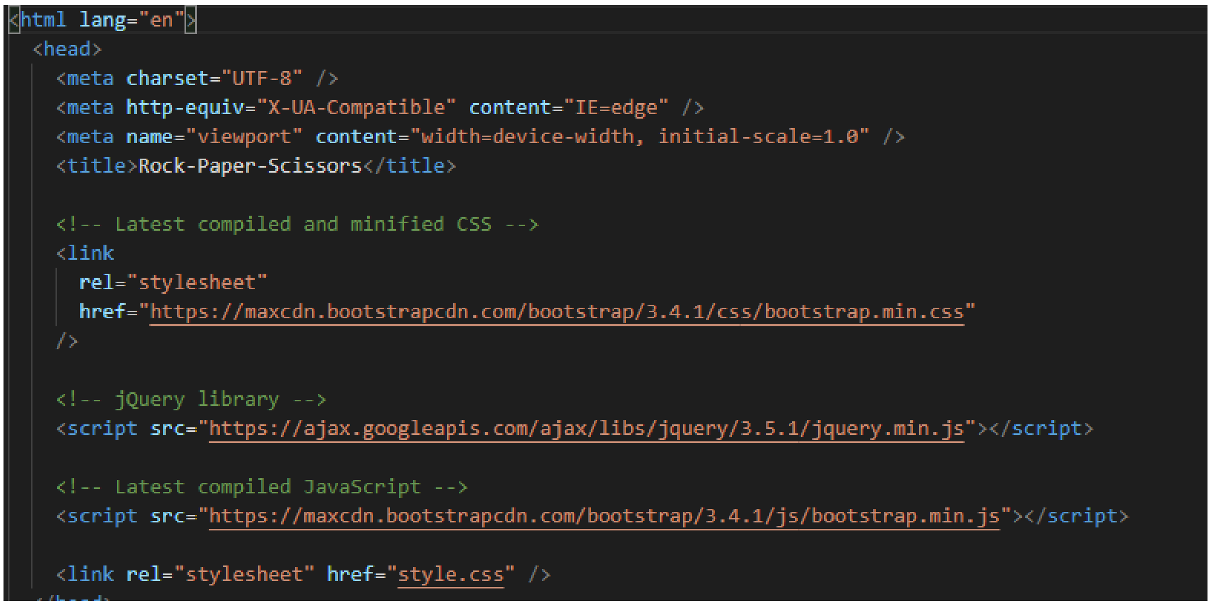
ICP3 Web Programming

Task1: Rock, Paper, Scissors

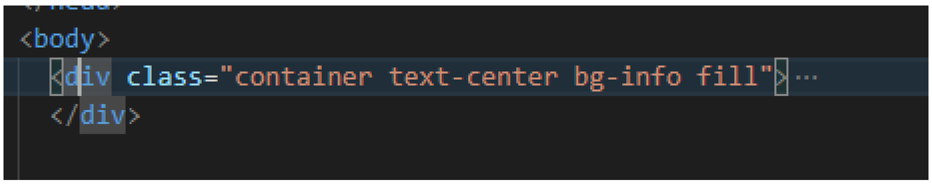
In this task initially index.html file is created with the bootstrap css and minified js cdn links added into the head section of the document.



```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Rock-Paper-Scissors</title>

    <!-- Latest compiled and minified CSS -->
    <link
      rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"
    />

    <!-- jQuery library -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

    <!-- Latest compiled JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

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

For the page layout a container div is created as below

Page Layout



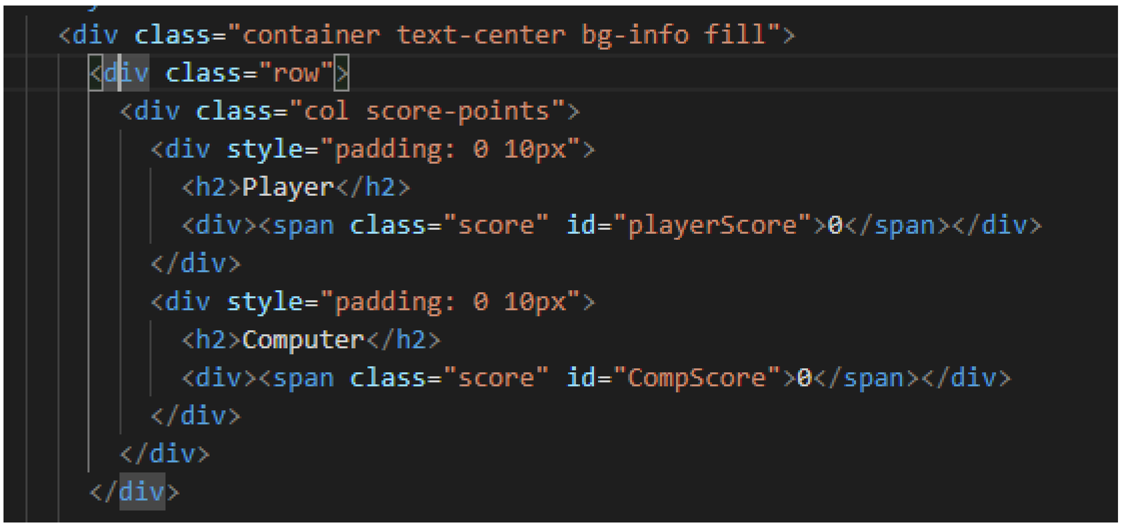
```
<body>
  <div class="container text-center bg-info fill">…
  </div>
```

Created a row, in that player and computer scores are displayed as in below snip. Initially these scores are set to 0. Nice css is added along with that for the buttons like padding: 10px

Used display: flex and space-around css to make the buttons on each end of the page.

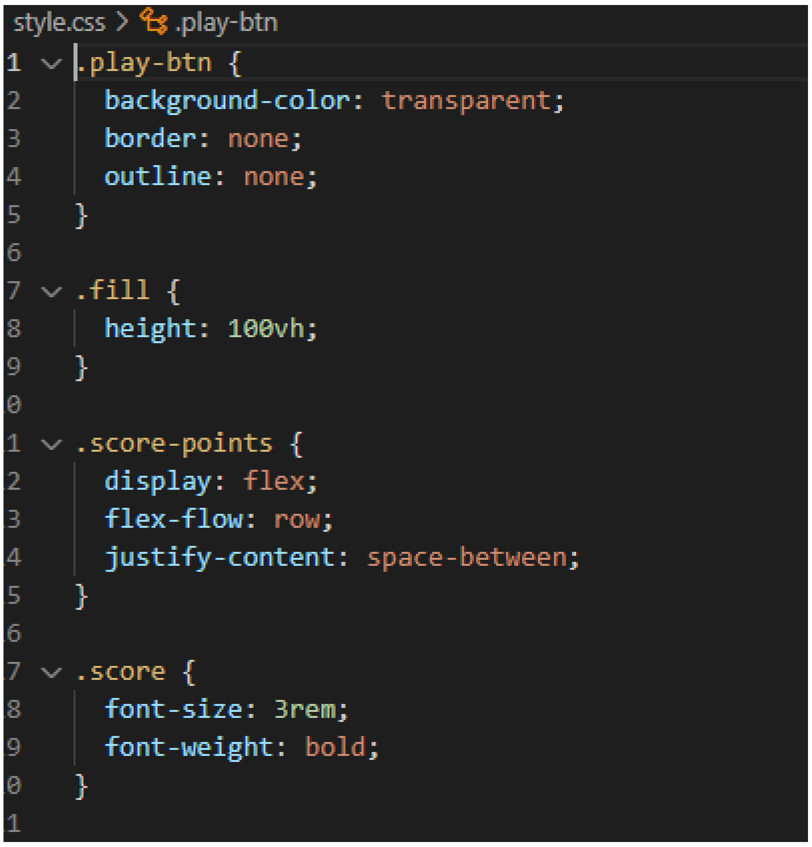
Col class is used here so that two buttons will be each on 50% width of the page. Col is similar to col-xs-6 here because we used two col divisions.

Score points at the top of the screen



```html
<div class="container text-center bg-info fill">
  <div class="row">
    <div class="col score-points">
      <div style="padding: 0 10px">
        <h2>Player</h2>
        <div><span class="score" id="playerScore">0</span></div>
      </div>
      <div style="padding: 0 10px">
        <h2>Computer</h2>
        <div><span class="score" id="CompScore">0</span></div>
      </div>
    </div>
  </div>
```

CSS for the player score buttons is as below:

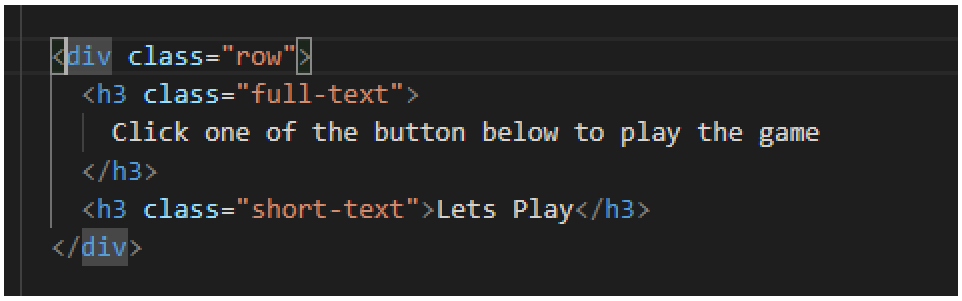


style.css > .play-btn

```css
.play-btn {
  background-color: transparent;
  border: none;
  outline: none;
}

.fill {
  height: 100vh;
}

.score-points {
  display: flex;
  flex-flow: row;
  justify-content: space-between;
}

.score {
  font-size: 3rem;
  font-weight: bold;
}
```

In the second row I created a h3 tag with some header text

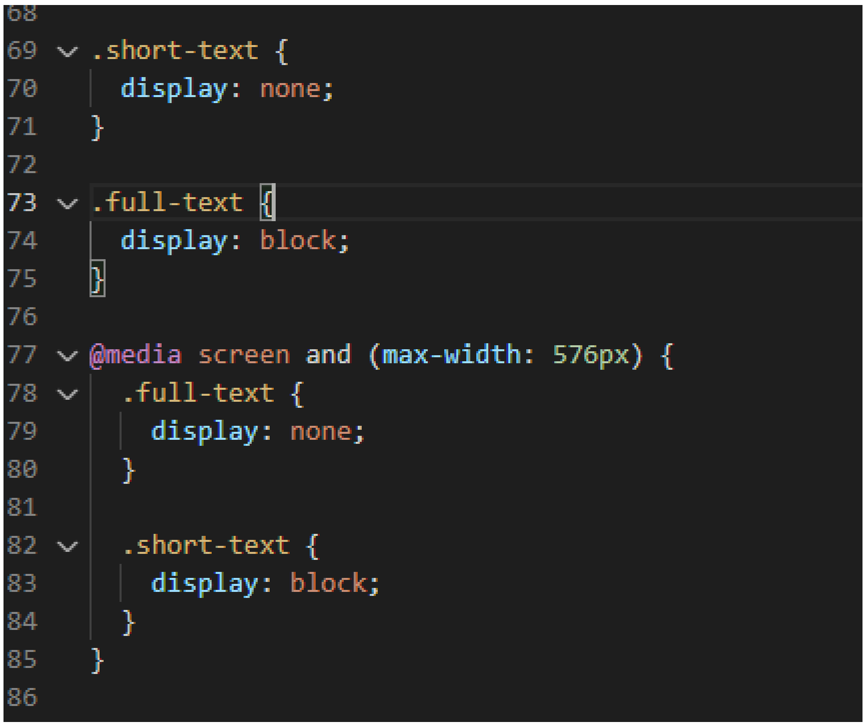
Game play information



```html
<div class="row">
  <h3 class="full-text">
    Click one of the button below to play the game
  </h3>
  <h3 class="short-text">Lets Play</h3>
</div>
```

Full text and short text are displayed individually based on screen width.

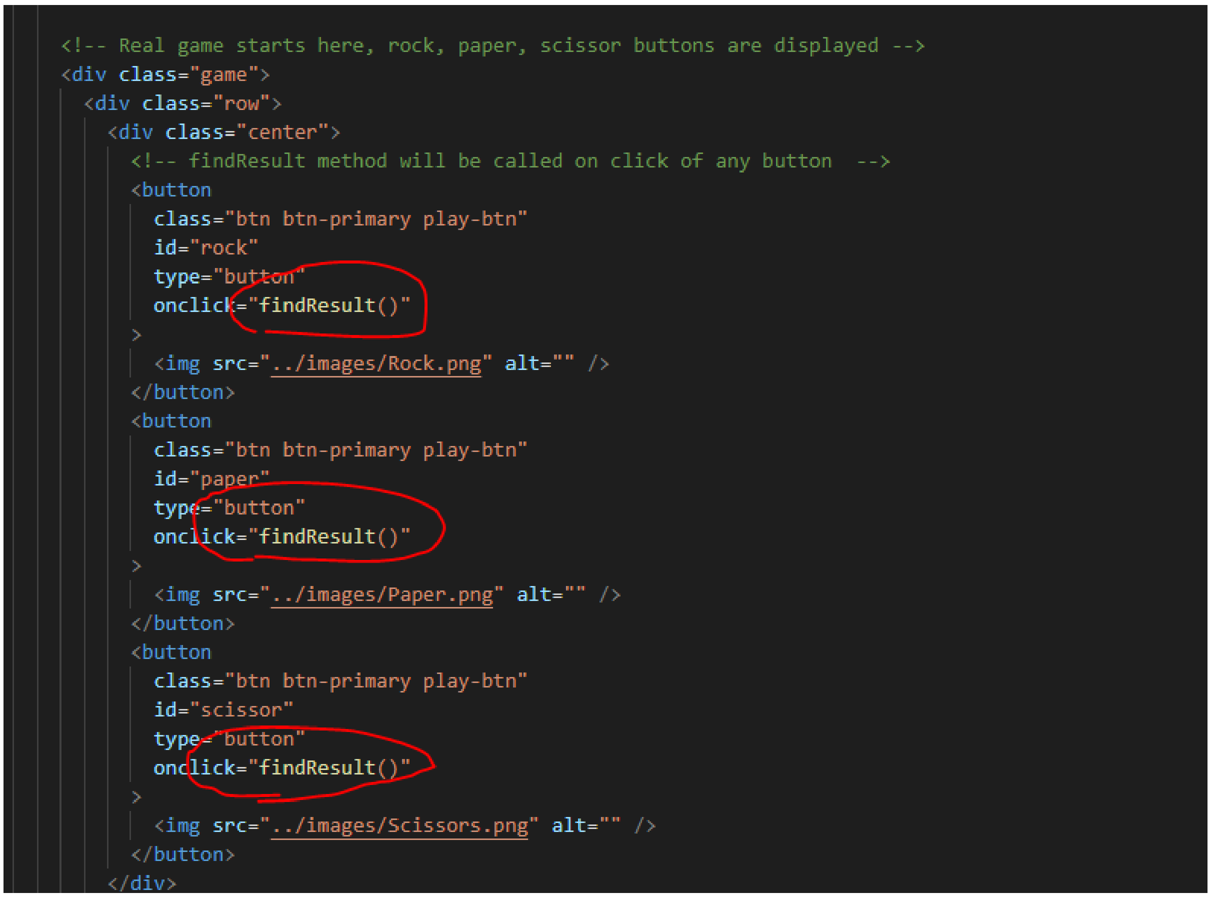
CSS for this as below:



```css
.short-text {
  display: none;
}

.full-text {
  display: block;
}

@media screen and (max-width: 576px) {
  .full-text {
    display: none;
  }

  .short-text {
    display: block;
  }
}
```

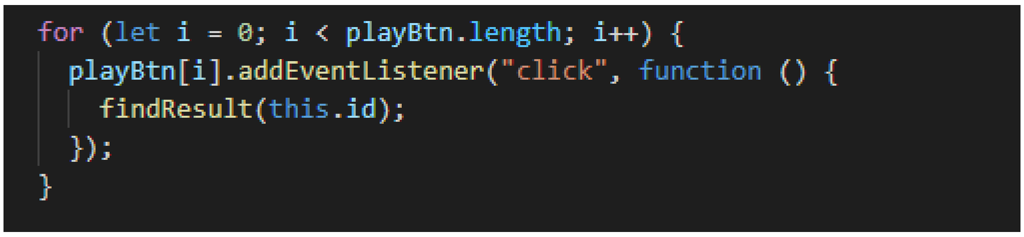
Business logic:

When user clicks on any of the three buttons findresult method will be called.



```html
<!-- Real game starts here, rock, paper, scissor buttons are displayed -->
<div class="game">
  <div class="row">
    <div class="center">
      <!-- findResult method will be called on click of any button  -->
      <button
        class="btn btn-primary play-btn"
        id="rock"
        type="button"
        onclick="findResult()"
      >
        <img src="../images/Rock.png" alt="" />
      </button>
      <button
        class="btn btn-primary play-btn"
        id="paper"
        type="button"
        onclick="findResult()"
      >
        <img src="../images/Paper.png" alt="" />
      </button>
      <button
        class="btn btn-primary play-btn"
        id="scissor"
        type="button"
        onclick="findResult()"
      >
        <img src="../images/Scissors.png" alt="" />
      </button>
    </div>
```

Alternative: Onclick functionality can be written directly in the javascript using add even listener as below:



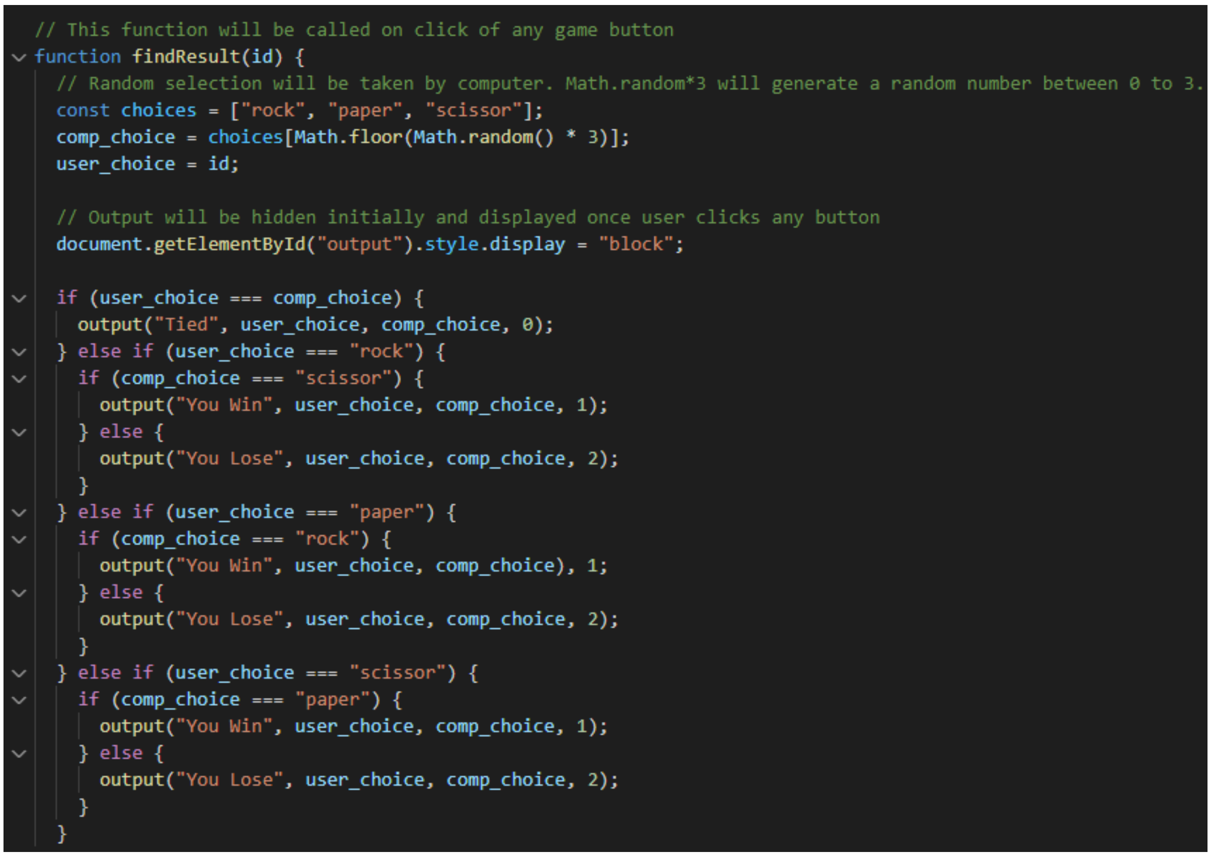
```javascript
for (let i = 0; i < playBtn.length; i++) {
  playBtn[i].addEventListener("click", function () {
    findResult(this.id);
  });
}
```

In this method computer choice will be generated by using Math.random() method. Math.random method generates a random number and It is built in javascript.

To generate random number formula is as below:

Math.random * (max-min) -min

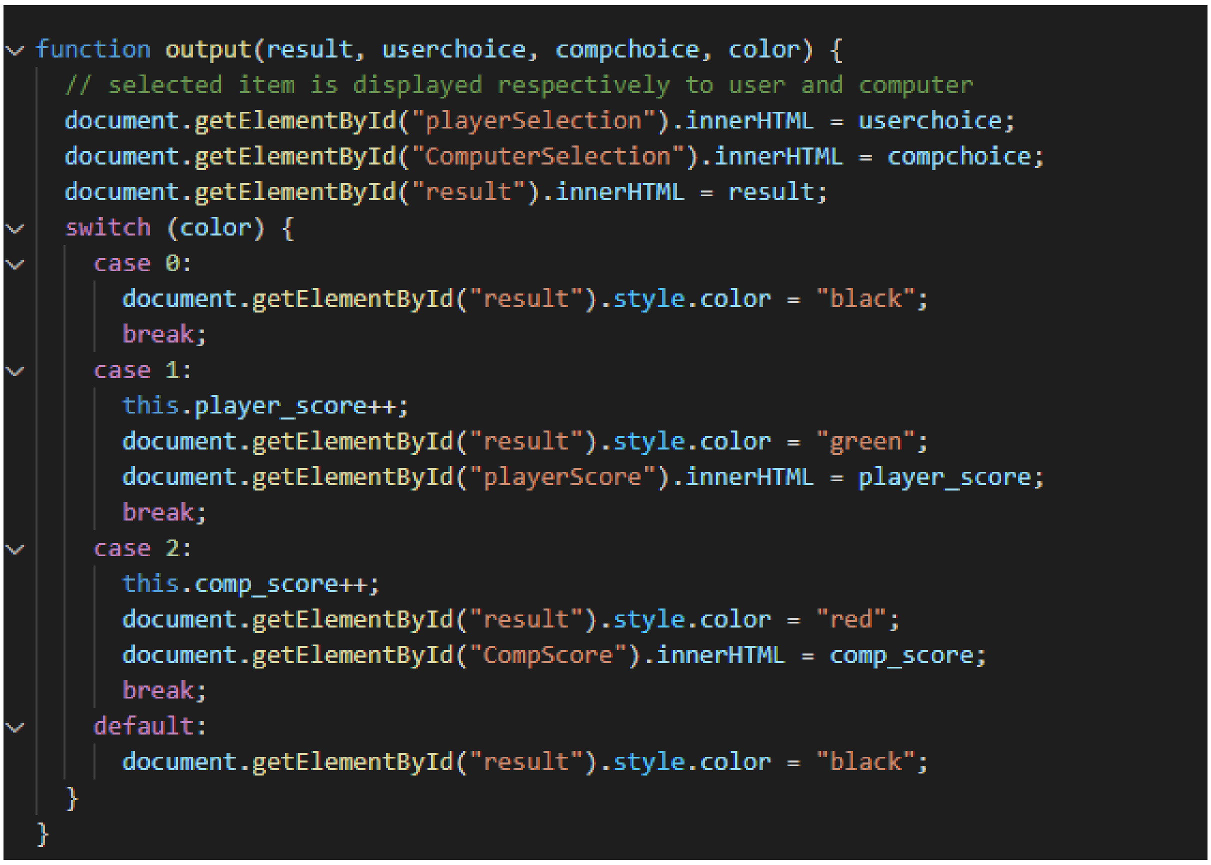
As we have three choices rock, paper and scissor: Math.random()*3 will generate a random number between o to 2. With this number we will take an element from the choices array.



```
// This function will be called on click of any game button
function findResult(id) {
  // Random selection will be taken by computer. Math.random*3 will generate a random number between 0 to 3.
  const choices = ["rock", "paper", "scissor"];
  comp_choice = choices[Math.floor(Math.random() * 3)];
  user_choice = id;

  // Output will be hidden initially and displayed once user clicks any button
  document.getElementById("output").style.display = "block";

  if (user_choice === comp_choice) {
    output("Tied", user_choice, comp_choice, 0);
  } else if (user_choice === "rock") {
    if (comp_choice === "scissor") {
      output("You Win", user_choice, comp_choice, 1);
    } else {
      output("You Lose", user_choice, comp_choice, 2);
    }
  } else if (user_choice === "paper") {
    if (comp_choice === "rock") {
      output("You Win", user_choice, comp_choice), 1;
    } else {
      output("You Lose", user_choice, comp_choice, 2);
    }
  } else if (user_choice === "scissor") {
    if (comp_choice === "paper") {
      output("You Win", user_choice, comp_choice, 1);
    } else {
      output("You Lose", user_choice, comp_choice, 2);
    }
  }
```

Based on conditions here again output method is called by passing user choice and computer choice as parameters.

Using these parameters output screen is displayed on the page.

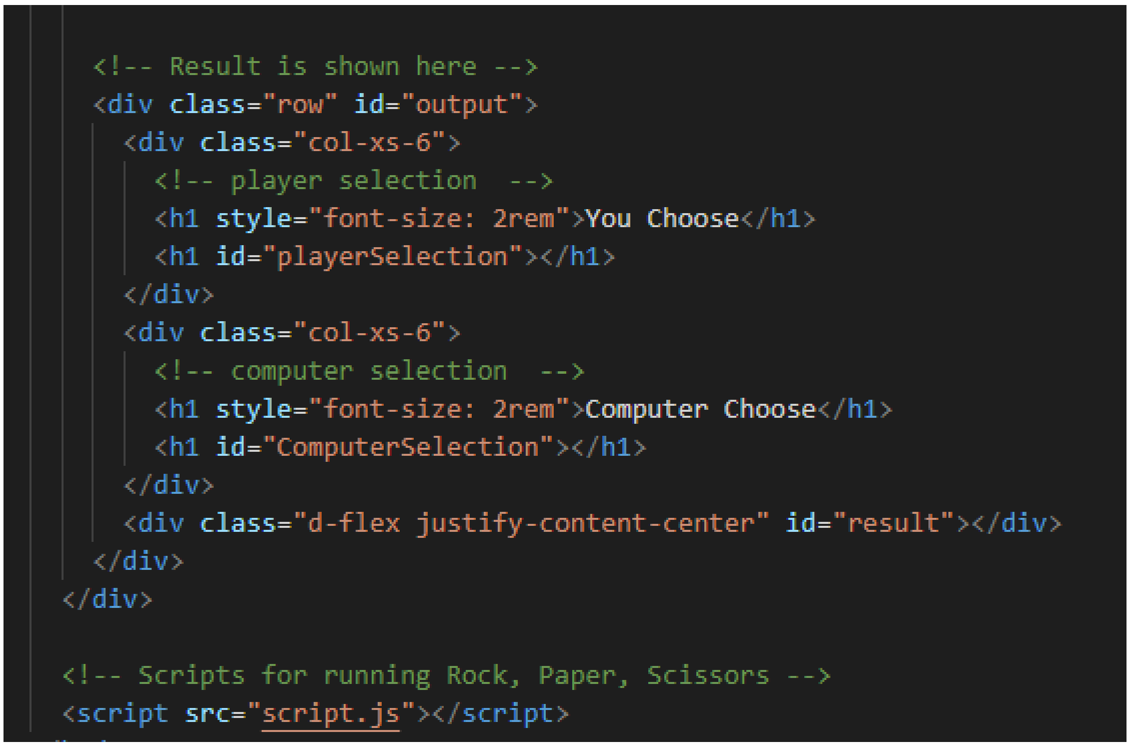


```javascript
function output(result, userchoice, compchoice, color) {
  // selected item is displayed respectively to user and computer
  document.getElementById("playerSelection").innerHTML = userchoice;
  document.getElementById("ComputerSelection").innerHTML = compchoice;
  document.getElementById("result").innerHTML = result;
  switch (color) {
    case 0:
      document.getElementById("result").style.color = "black";
      break;
    case 1:
      this.player_score++;
      document.getElementById("result").style.color = "green";
      document.getElementById("playerScore").innerHTML = player_score;
      break;
    case 2:
      this.comp_score++;
      document.getElementById("result").style.color = "red";
      document.getElementById("CompScore").innerHTML = comp_score;
      break;
    default:
      document.getElementById("result").style.color = "black";
  }
}
```

If user wins result is displayed in green color. If loses result is displayed on red color. If there is a tie then the text color is shown as black.

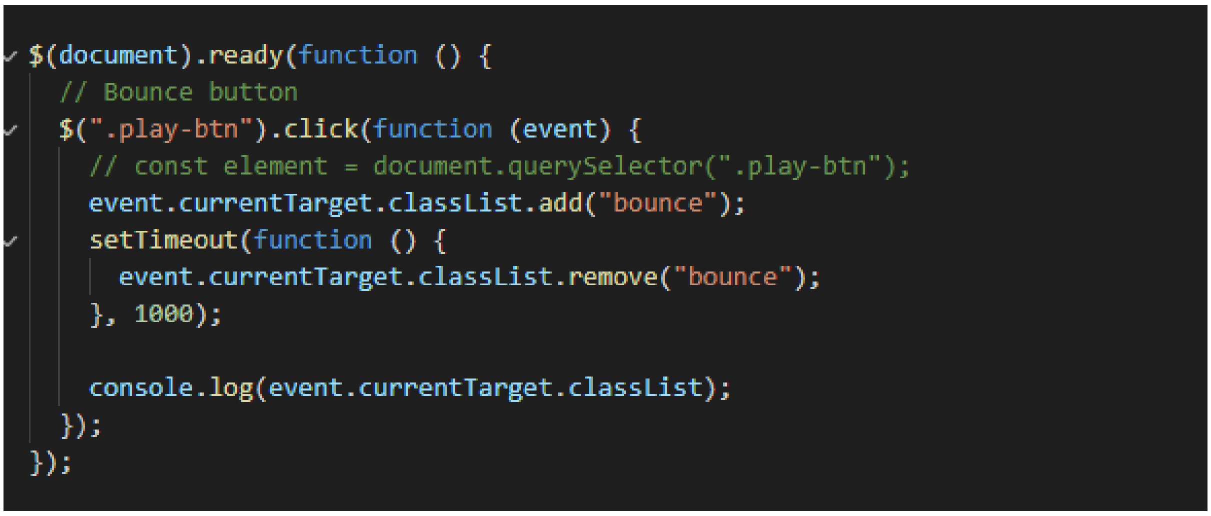
There will a increment in individual scores of user and computer whoever wins.

<u>Output HTML:</u>



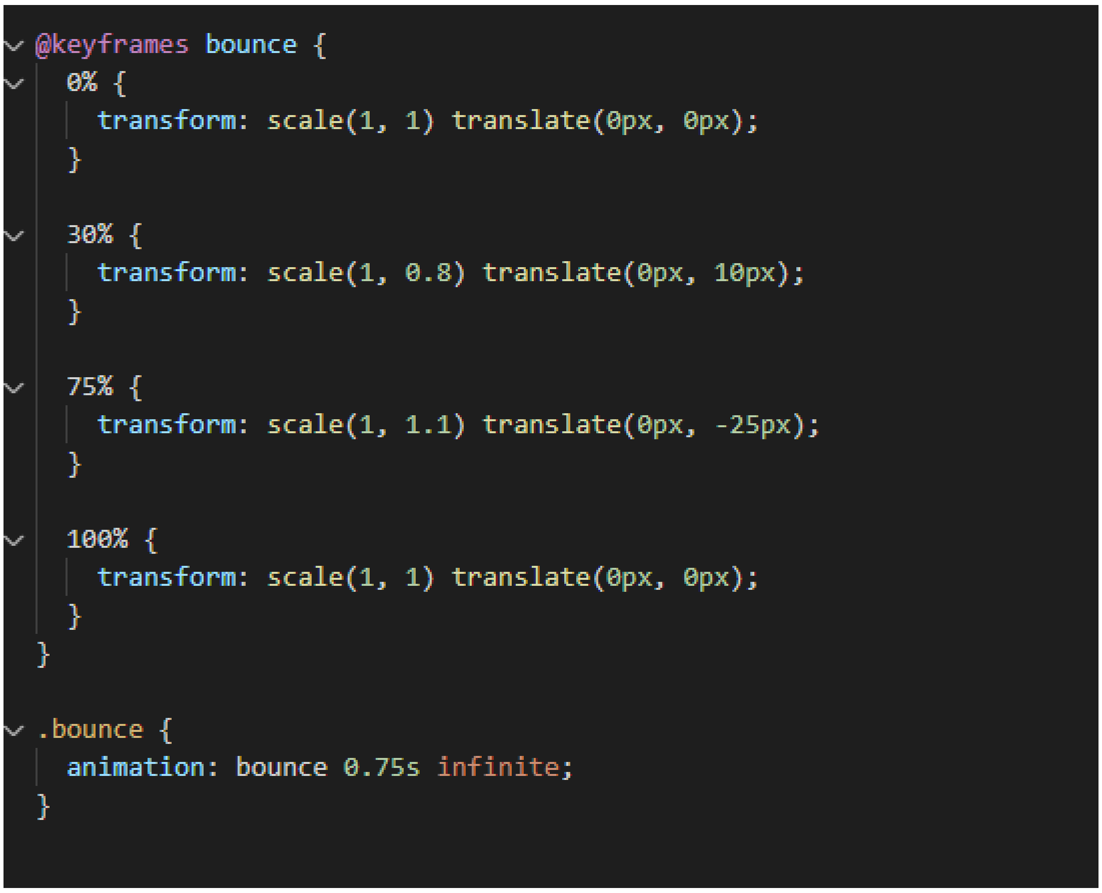
```html
    <!-- Result is shown here -->
    <div class="row" id="output">
      <div class="col-xs-6">
        <!-- player selection  -->
        <h1 style="font-size: 2rem">You Choose</h1>
        <h1 id="playerSelection"></h1>
      </div>
      <div class="col-xs-6">
        <!-- computer selection  -->
        <h1 style="font-size: 2rem">Computer Choose</h1>
        <h1 id="ComputerSelection"></h1>
      </div>
      <div class="d-flex justify-content-center" id="result"></div>
    </div>
  </div>

  <!-- Scripts for running Rock, Paper, Scissors -->
  <script src="script.js"></script>
```

<u>Logic for button bounce animation:</u>



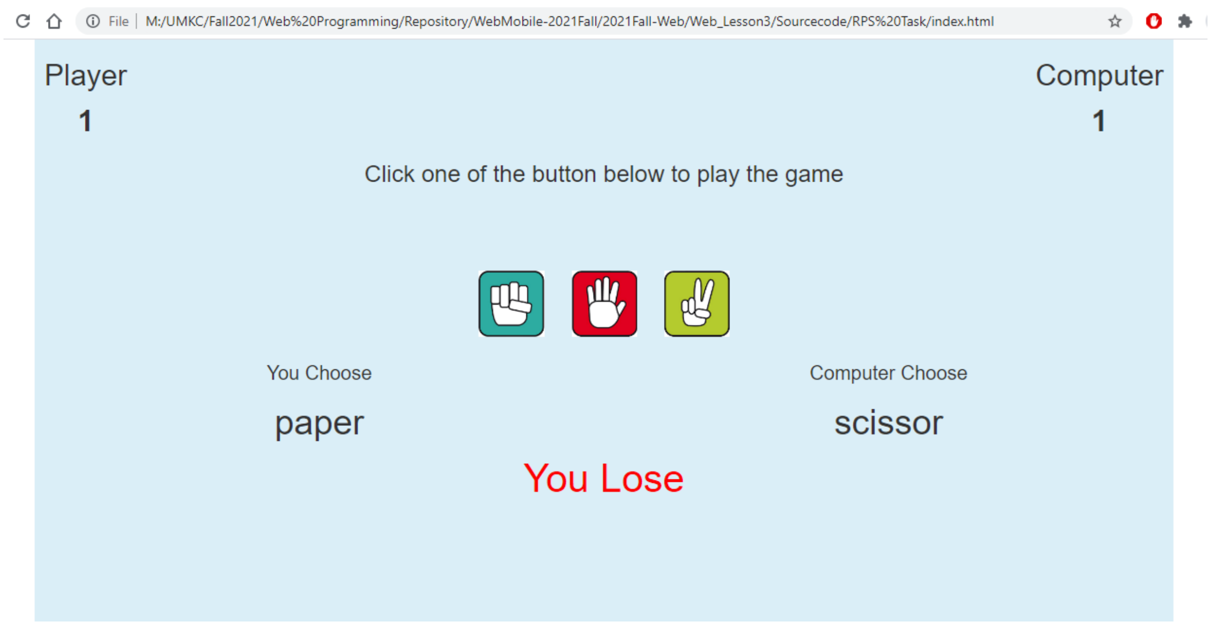
```javascript
$(document).ready(function () {
  // Bounce button
  $(".play-btn").click(function (event) {
    // const element = document.querySelector(".play-btn");
    event.currentTarget.classList.add("bounce");
    setTimeout(function () {
      event.currentTarget.classList.remove("bounce");
    }, 1000);

    console.log(event.currentTarget.classList);
  });
});
```

CSS for the bounce button animation



```css
@keyframes bounce {
  0% {
    transform: scale(1, 1) translate(0px, 0px);
  }

  30% {
    transform: scale(1, 0.8) translate(0px, 10px);
  }

  75% {
    transform: scale(1, 1.1) translate(0px, -25px);
  }

  100% {
    transform: scale(1, 1) translate(0px, 0px);
  }
}

.bounce {
  animation: bounce 0.75s infinite;
}
```

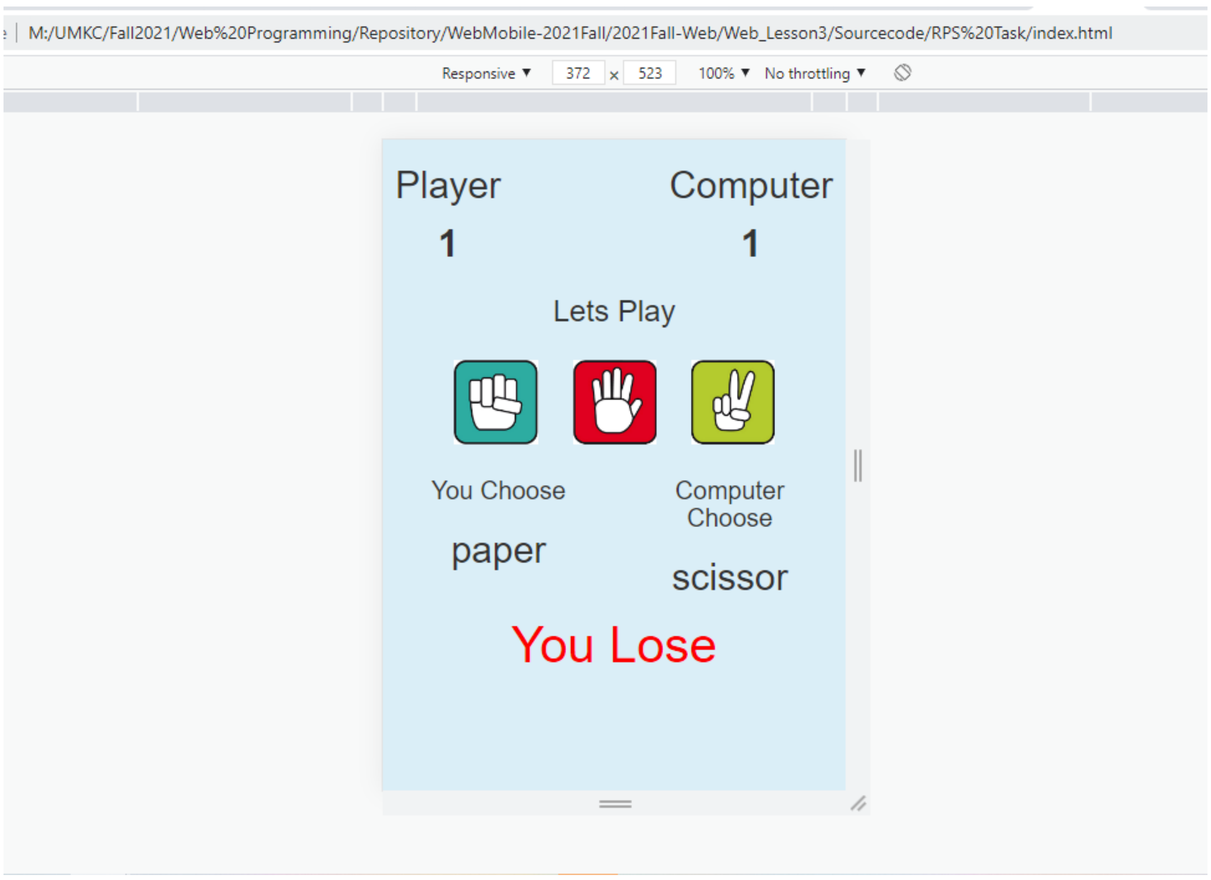
__Output of this task:__



File | M:/UMKC/Fall2021/Web%20Programming/Repository/WebMobile-2021Fall/2021Fall-Web/Web_Lesson3/Sourcecode/RPS%20Task/index.html

```
Player                                                                Computer
  1                                                                      1

                    Click one of the button below to play the game

              You Choose                              Computer Choose
                paper                                     scissor
                                   You Lose
```

Mobile screen layout


M:/UMKC/Fall2021/Web%20Programming/Repository/WebMobile-2021Fall/2021Fall-Web/Web_Lesson3/Sourcecode/RPS%20Task/index.html

Responsive ▼ 372 x 523 100% ▼ No throttling ▼

Player Computer

1 1

Lets Play

You Choose

paper

Computer Choose

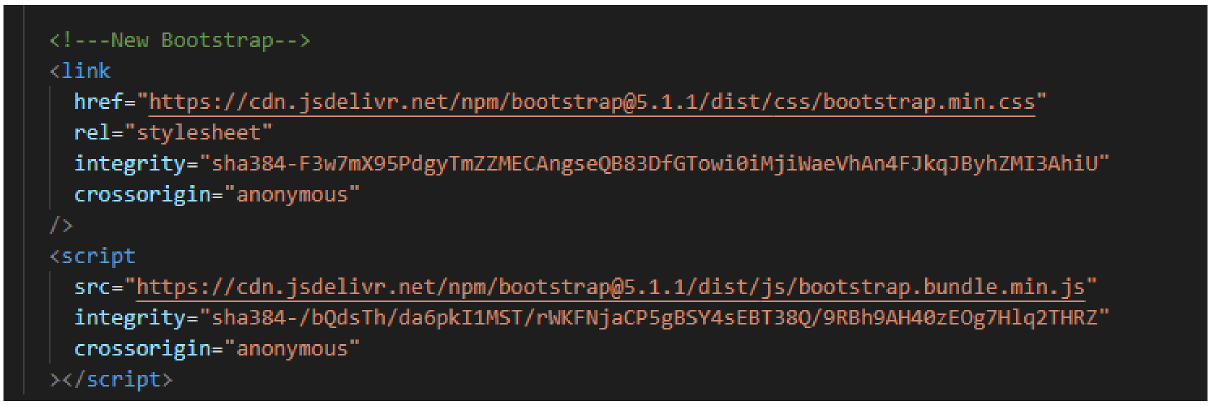
scissor

You Lose

Task2:

Responsive web design

Bootstrap cdn are added to the head section of the document



```html
<!---New Bootstrap-->
<link
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
  rel="stylesheet"
  integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
  crossorigin="anonymous"
/>
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-/bQdsTh/da6pkI1MST/rWKFNjaCP5gBSY4sEBT38Q/9RBh9AH40zEOg7Hlq2THRZ"
  crossorigin="anonymous"
></script>
```

Layout

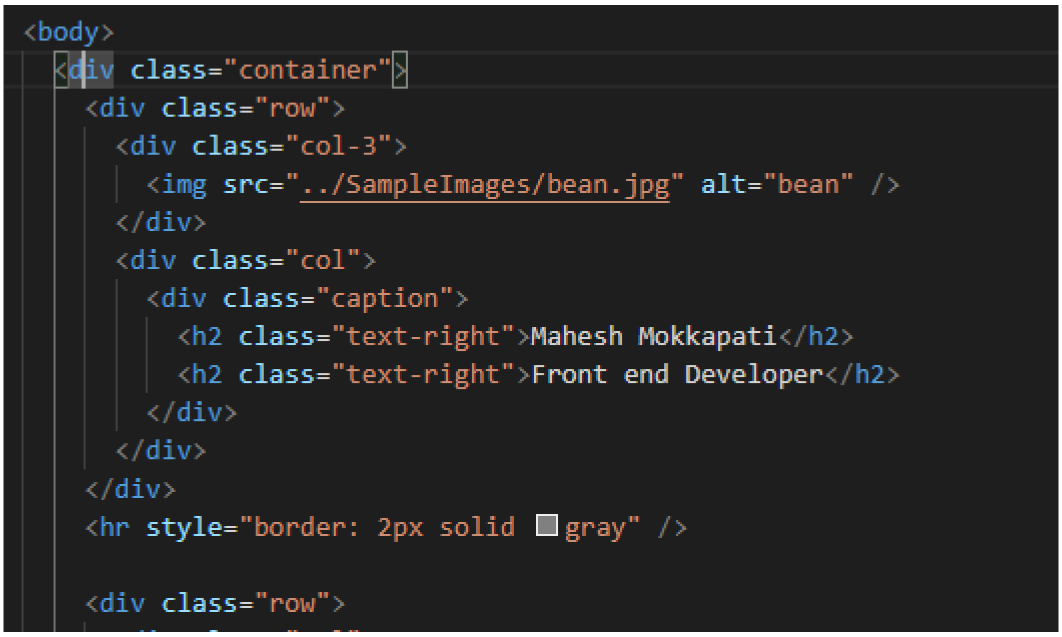
Container is created which will be parent for the layout.

We can use container-fluid also if we want but the page width will be taken 100% here.

Logo is created in the first three columns of the row. And banner information is displayed to the right in the next 9 coumns.

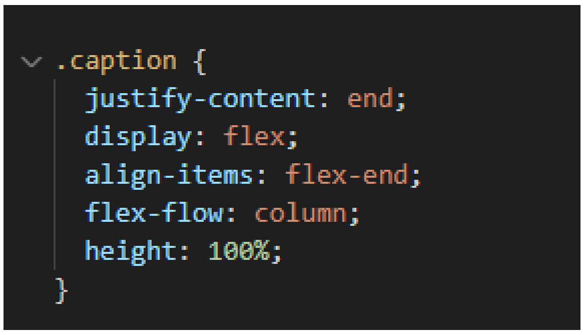
Here is the HTML for this

Page header and first row



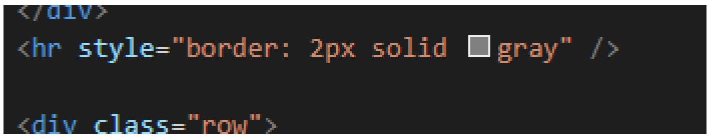
```html
<body>
  <div class="container">
    <div class="row">
      <div class="col-3">
        <img src="../SampleImages/bean.jpg" alt="bean" />
      </div>
      <div class="col">
        <div class="caption">
          <h2 class="text-right">Mahesh Mokkapati</h2>
          <h2 class="text-right">Front end Developer</h2>
        </div>
      </div>
    </div>
    <hr style="border: 2px solid ▢gray" />

    <div class="row">
```

CSS for caption



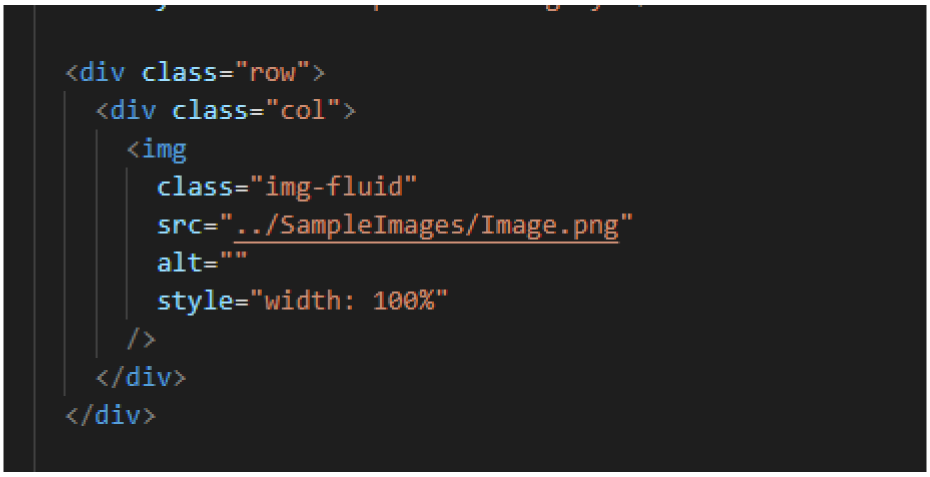
```
.caption {
  justify-content: end;
  display: flex;
  align-items: flex-end;
  flex-flow: column;
  height: 100%;
}
```

Horizontal rule is created using <hr> tag



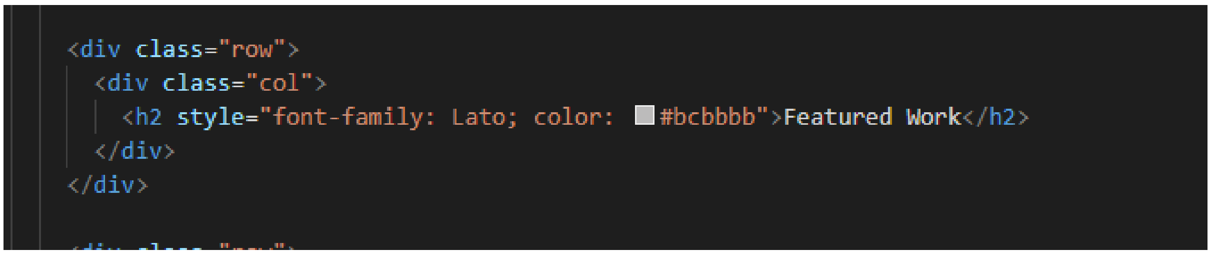
```
</div>
<hr style="border: 2px solid gray" />

<div class="row">
```

In the second row image is shown using img-fluid class. This class will make the image use width 100% relative to parent and will be responsive to the parent



```
<div class="row">
  <div class="col">
    <img
      class="img-fluid"
      src="../SampleImages/Image.png"
      alt=""
      style="width: 100%"
    />
  </div>
</div>
```

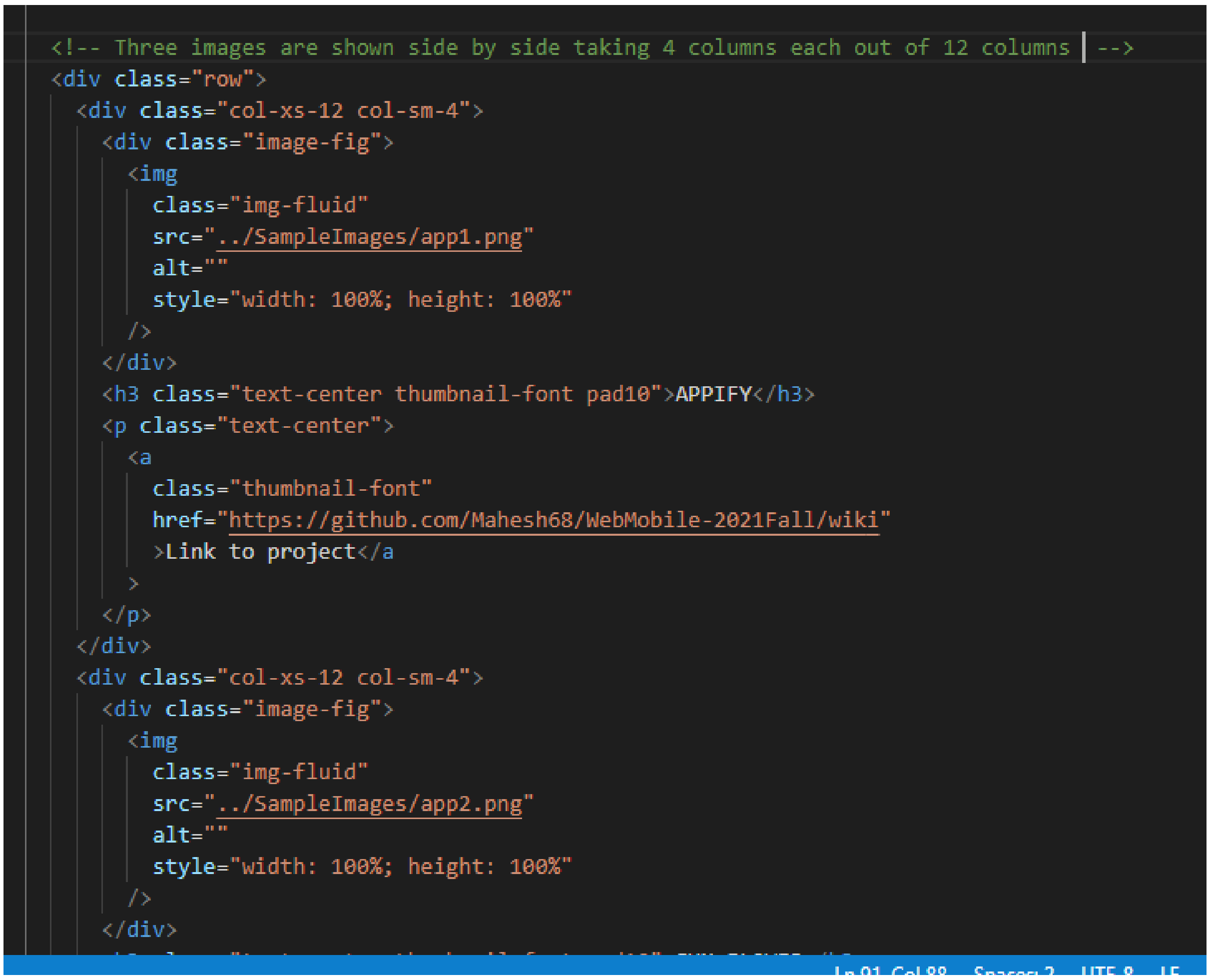
In the next row featured work text is displayed with the font -family as loto



```
<div class="row">
  <div class="col">
    <h2 style="font-family: Lato; color: #bcbbbb">Featured Work</h2>
  </div>
</div>
```

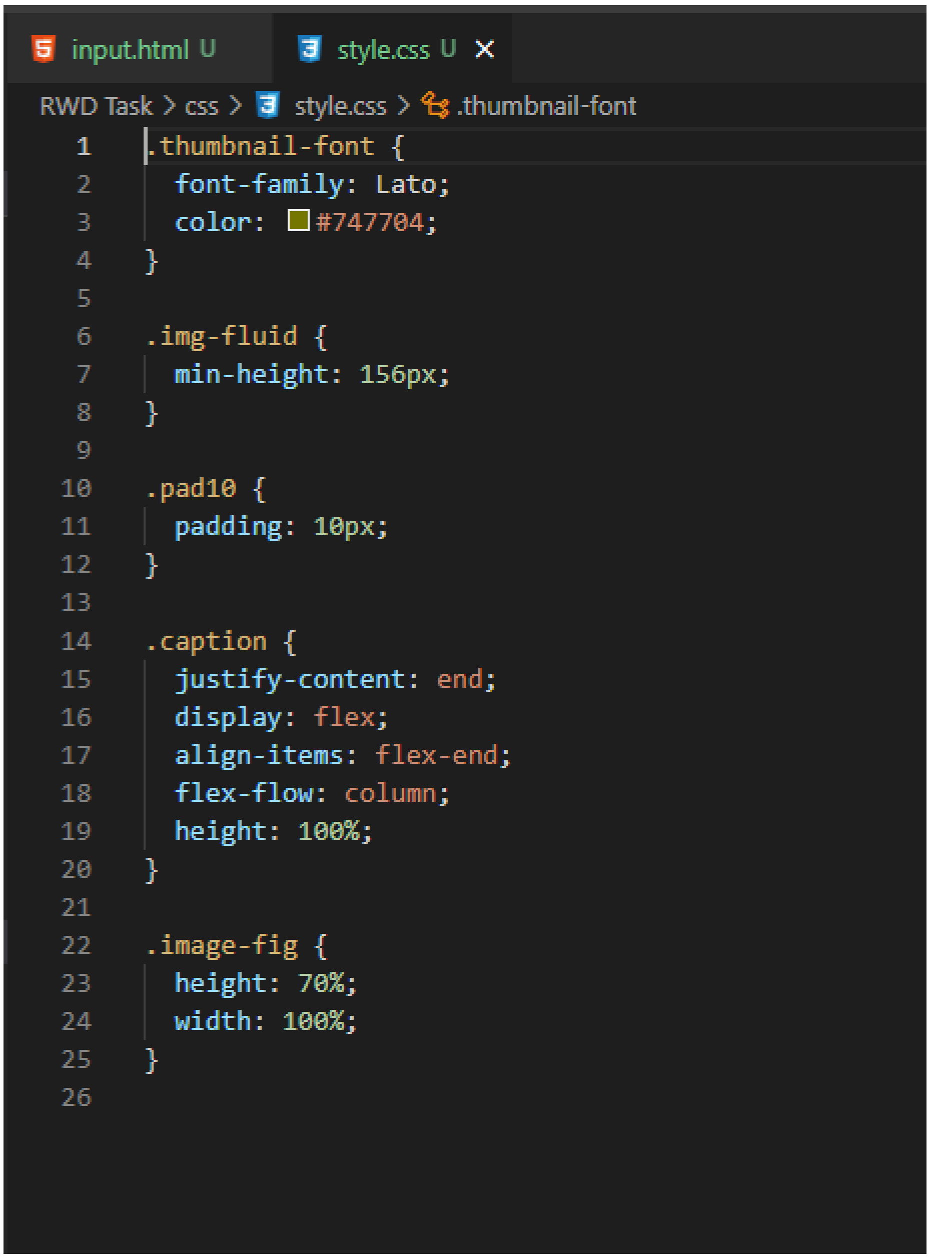
In the next row as I have to display three images in the 12 columns. Divided the 12 columns into 4 columns each so that each group will contain one image.

Responsive design will be achieved with the help of these container, rowm column layout and with the help of bootstrap css.



```
<!-- Three images are shown side by side taking 4 columns each out of 12 columns -->
<div class="row">
  <div class="col-xs-12 col-sm-4">
    <div class="image-fig">
      <img
        class="img-fluid"
        src="../SampleImages/app1.png"
        alt=""
        style="width: 100%; height: 100%"
      />
    </div>
    <h3 class="text-center thumbnail-font pad10">APPIFY</h3>
    <p class="text-center">
      <a
        class="thumbnail-font"
        href="https://github.com/Mahesh68/WebMobile-2021Fall/wiki"
        >Link to project</a
      >
    </p>
  </div>
  <div class="col-xs-12 col-sm-4">
    <div class="image-fig">
      <img
        class="img-fluid"
        src="../SampleImages/app2.png"
        alt=""
        style="width: 100%; height: 100%"
      />
    </div>
```

Here is the HTML and css for the three images

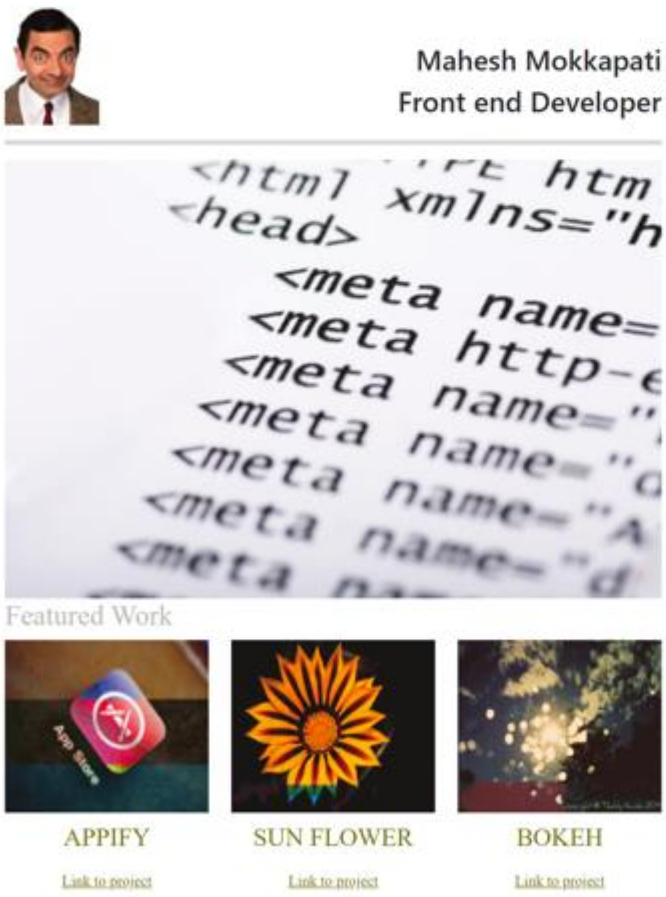

input.html U style.css U ×

RWD Task > css > style.css > .thumbnail-font

```css
.thumbnail-font {
  font-family: Lato;
  color: #747704;
}

.img-fluid {
  min-height: 156px;
}

.pad10 {
  padding: 10px;
}

.caption {
  justify-content: end;
  display: flex;
  align-items: flex-end;
  flex-flow: column;
  height: 100%;
}

.image-fig {
  height: 70%;
  width: 100%;
}

```

To make the images height equally I have created a img-flg div and in that image height is taken 100%. Because of this if we vary height of img-flg css all the images height will be varied accordingly.

Mobile screen:



Mahesh Mokkapati
Front end Developer

Featured Work

APPIFY
Link to project

SUN FLOWER
Link to project

BOKEH
Link to project

Laptop screen:

Mahesh Mokkapati
Front end Developer

Featured Work

APPIFY

Link to project

SUN FLOWER

Link to project

BOKEH

Link to project