



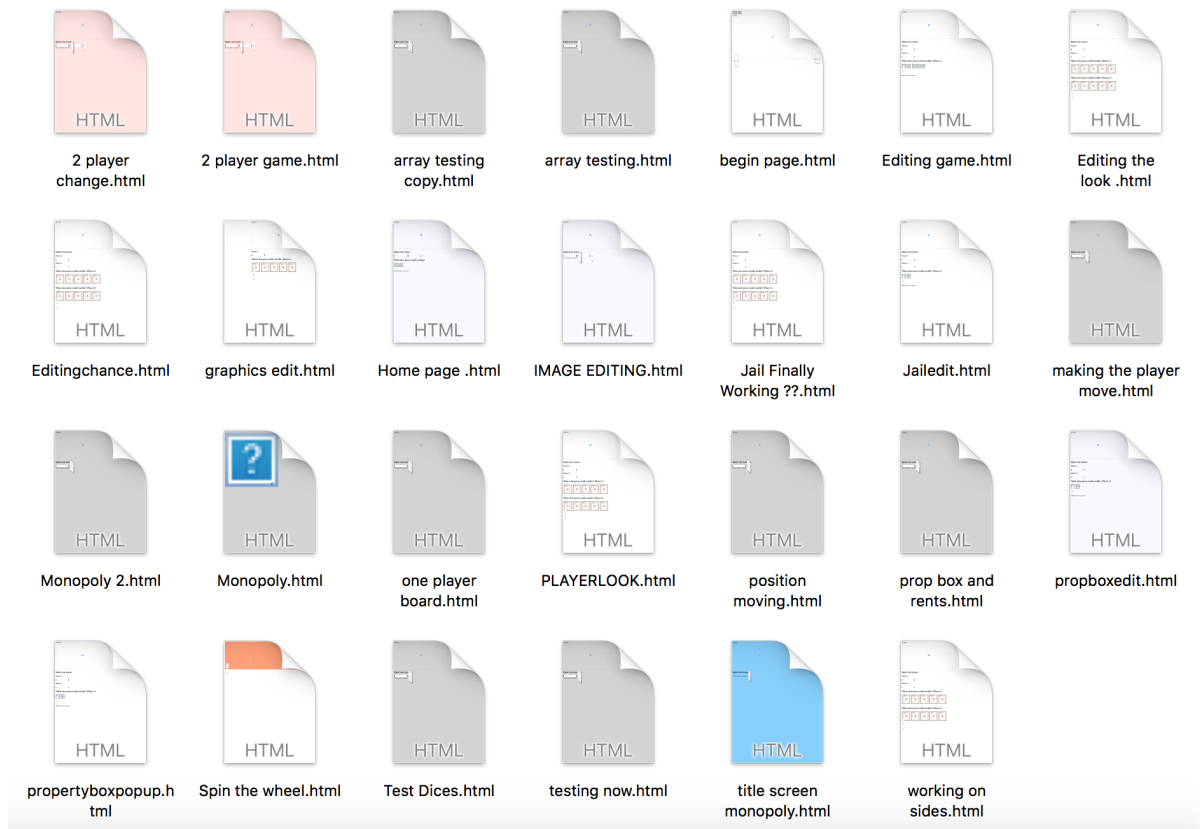
Implementing

MAJOR WORK
LAVANYA SOOD

YEAR 12

Version Control

Creation and organization of code and other elements



The Major Project is saved in different versions as new changes were implemented to the software to prevent loss of existing progress. This was done as a safety measure so that if the new features or functions causes the whole program to crash, the last working version of the program can be recovered. This prevents us from accidently destroying the whole program and losing all the progress

Testing of Code

Stub

```
function rollDice() {  
  // var die1 = document.getElementById("die1");  
  // var die2 = document.getElementById("die2");  
  // var status = document.getElementById("status");  
  // var d1 = Math.floor(Math.random() * 6) + 1;  
  // var d2 = Math.floor(Math.random() * 6) + 1;  
  // diceTotal = d1 + d2;  
  // die1.innerHTML = d1;  
  // die2.innerHTML = d2;  
  // status.innerHTML = "You rolled " + diceTotal + ".";  
  // return diceTotal;  
  return 10;  
}
```

A stub was used in the rollDice() function to test a specific module. A specific value was returned from the rollDice() function so that the player can land on a specific location on the board and the function of that specific position could be tested. For example: the rollDice() function was set to return 10, so that the jail function of the program could be tested.

Debugging Output Statement

```
function chancecards(){  
  var cnum = Math.round(Math.random()*8) + 1;  
  var picchance = chances[cnum].ccard;  
  console.log (cnum);  
  document.getElementById("chancecardimg").setAttribute("src",picchance);  
}
```

Debugging output statement was used to test what specific value the program was using. It was used in the chance function to see that if the value of cnum displayed and to test if the value of cnum displayed is correct or not.

Flag

```
var playerturn = 0;
function endturn() {
  if (turnplayer1 == true){
    turnplayer1 = false;
    turnplayer2 = true;
    playerturn = 1;
    console.log(playerturn);
  } else if (turnplayer2 == true) {
    turnplayer2 = false;
    turnplayer1 = true;
    playerturn = 2;
    console.log(playerturn);
  }
}
```

A flag was used to test what function the program was going into. In the endturn() function the variable playerturn was set to 1, if it was player 1's turn or 2, if it was player 2's turn. This was used to test whose turn it is. This variable is then checked using a debugging output statement (`console.log(playerturn)`) to check which if statement the function went into.

Breakpoint

```
572 var currentjail2 = false;
573
574 function movePlayer() {
575   if (turnplayer1 == true) {
576     //if player in jail, ask for payment. If payment given, continue turn, otherwise skip
577     if (currentjail == false) {
578       diceTotal = rollDice();
579       current_pos = current_pos + diceTotal;
580       if (current_pos > 39) {
581         current_pos = current_pos - 40;
582       }
583     }
584   }
```

A breakpoint and line stepping were used to test the moveplayer() function. was used to test the step-by-step run through of the program and to check which loop it enters. It allowed me to see how each line was executed and to see the errors.

STEP THROUGH

```

    if (turnplayer1 == true) {

        //if player in jail, ask for payment. If payment given, continue turn, otherwise skip
        if (currentjail == false) {
            diceTotal = rollDice();
            current_pos = current_pos + diceTotal;
            if (current_pos > 39) {
                current_pos = current_pos - 40;
            }
            if (current_pos == 10) {
                currentjail = true;
            }
            //console.log(current_pos);
            var x = positions[current_pos].x
            var y = positions[current_pos].y
            console.log(x);
            console.log(y);
            document.getElementById("object").setAttribute("x", x);
            document.getElementById("object").setAttribute("y", y);
        }
    }
}

```

```

function movePlayer() {
    if (turnplayer1 == true) {

        //if player in jail, ask for payment. If payment given, continue turn, otherwise skip
        if (currentjail == false) {
            diceTotal = rollDice();
            current_pos = current_pos + diceTotal;
            if (current_pos > 39) {
                current_pos = current_pos - 40;
            }
            if (current_pos == 10) {
                currentjail = true;
            }
            //console.log(current_pos);
            var x = positions[current_pos].x
            var y = positions[current_pos].y
            console.log(x);
            console.log(y);
            document.getElementById("object").setAttribute("x", x);
            document.getElementById("object").setAttribute("y", y);
        }
    }
}

```

```

function movePlayer() {
    if (turnplayer1 == true) {

        //if player in jail, ask for payment. If payment given, continue turn, otherwise skip
        if (currentjail == false) {
            diceTotal = rollDice();
            current_pos = current_pos + diceTotal;
            if (current_pos > 39) {
                current_pos = current_pos - 40;
            }
            if (current_pos == 10) {
                currentjail = true;
            }
            //console.log(current_pos);
            var x = positions[current_pos].x
            var y = positions[current_pos].y
            console.log(x);
            console.log(y);
            document.getElementById("object").setAttribute("x", x);
            document.getElementById("object").setAttribute("y", y);
        }
    }
}

```

```

574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590

```

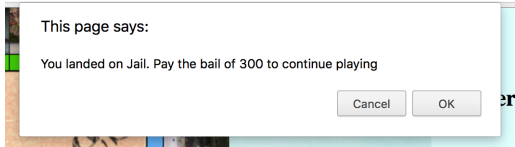
```

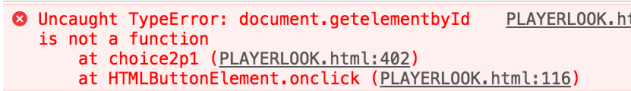
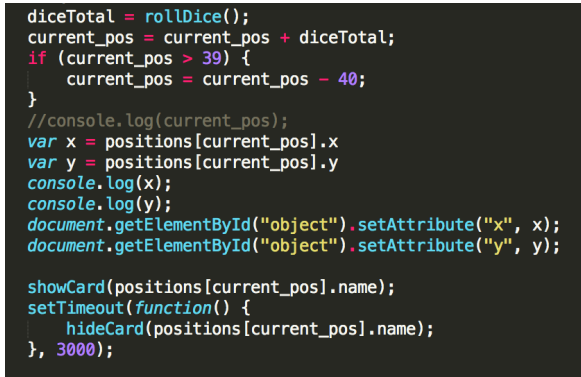
        //if player in jail, ask for payment. If payment given, continue turn, otherwise skip
        if (currentjail == false) {
            diceTotal = rollDice();
            current_pos = current_pos + diceTotal;
            if (current_pos > 39) {
                current_pos = current_pos - 40;
            }
            if (current_pos == 10) {
                currentjail = true;
            }
            //console.log(current_pos);
            var x = positions[current_pos].x
            var y = positions[current_pos].y
            console.log(x);
            console.log(y);
            document.getElementById("object").setAttribute("x", x);
            document.getElementById("object").setAttribute("y", y);
        }
    }
}

```

The stepping through process takes place to check what process the program follows, here once the breakpoint is set on the moveplayer() function the program steps through line by line. As the current_pos(current position) of the program is less than 39 the program skips the if statement on line 578 and moves to line 581 to test whether the player is on 10. Thus the stepping through process by adding a breakpoint helps in the checking of the program step-by-step.

Table Of Errors

ERROR	TYPE OF ERROR	MAJOR / MINOR	WHAT THE ERROR WAS	HOW IT WAS RESOLVED
Money of the player not being subtracted	Logical	Moderate	When the user landed on a spot and paid the rent. There was no displayed change in the money of the player.	<p>The money in the html was not changing after a transaction was made however, there was a change in the actual value of the variable. Thus the following lines were added to the code.</p> <pre>document.getElementById("moneydis").innerHTML = money1; document.getElementById("moneydis2").innerHTML = money2;</pre> <p>This updated the money display every time the user gained or lost money.</p>
Jail function not working	Logical	Major	<p>The user was stuck in an endless loop in the jail whether he paid the bail or not</p> 	<p>The currentjail variable in the program which was used to test whether the user was in jail or not was set to the opposite Boolean variable.</p> <p>Thus, when the user landed in jail and paid his bail the currentjail was again set to true and thus the player was stuck in an endless cycle.</p>

			<i>This pop-up box was continuously displayed whether the user had paid their bail or not.</i>	<i>Thus this issue was resolved by reversing the Boolean values of the variable</i>
<i>The selected pawn style was not displayed</i>	<i>Syntax</i>	<i>Major</i>	<i>When the user tried to open the game the whole program crashed and the following error was displayed:</i> 	<i>Reevaluating the code I realized that I had written “document.getelementbyid” instead of “document.getElementById” and thus due to the wrong syntax error the whole program kept on crashing</i>
<i>The player was running off the board instead of going into a loop around the corner</i>	<i>Logical</i>	<i>Major</i>	<i>The player used to run off the board as soon as the coordinates of the player position were greater than the size of the board</i>	 <pre> diceTotal = rollDice(); current_pos = current_pos + diceTotal; if (current_pos > 39) { current_pos = current_pos - 40; } //console.log(current_pos); var x = positions[current_pos].x var y = positions[current_pos].y console.log(x); console.log(y); document.getElementById("object").setAttribute("x", x); document.getElementById("object").setAttribute("y", y); showCard(positions[current_pos].name); setTimeout(function() { hideCard(positions[current_pos].name); }, 3000); </pre> <p><i>Thus the x and y co-ordinates were taken from the current position and thus the user was allowed to landed on the desired position all around the board</i></p>

<p>The properties purchased were not being displayed in the property box</p>	<p>Logical</p>	<p>Moderate</p>	<p>When the player purchased a property a house of the color of the player's pawn had to placed on every property he purchased. However the property was only being placed on one of the properties he purchased</p>	<p>The following lines of code were added to test for the error being displayed in the program:</p> <pre> var propertyboxadd = document.getElementsByClassName("propbutton")[0].cloneNode(true); propertyboxadd.setAttribute("x", xpos); propertyboxadd.setAttribute("y", ypos); propertyboxadd.setAttribute("width", 20); propertyboxadd.setAttribute("height", 20); document.getElementById("prop1svg").appendChild(propertyboxadd); </pre> <p>This led to each image being copied and then being displayed on every position the user purchased. Thus resolving the error</p>
<p>Repurchased property</p>	<p>Logical</p>	<p>Major</p>	<p>The player was able to purchase a property whether it was already purchased by another player or not. This caused the player to have multiple copies of the same property.</p>	<p>This was solved by adding an if statement to check if the property was in the box or not.</p> <pre> if (propertybox2.indexOf(theLanded) > -1) { var rent = parseInt(positions[current_pos].rent); money1 = money1 - rent; payment = "You just paid rent!" document.getElementById("chancestat").innerHTML = payment; } else if (propertybox1.indexOf(theLanded) > -1) { money1 = money1 + 0 } else { propertybuy(); } </pre> <p>This if statement checks whether the property the user is about to purchase is free or not. If the property is in player 2's property box it asks player 1 to pay rent. If the property is in Player 1's box itself, it does nothing, if the property is empty with no claim, it asks the user if they want to purchase the property or not.</p>

Social/ Ethical Considerations

When implementing a software solution, social and ethical issues need to be taken into consideration so that the program is received by society positively and is accessible to various people:

- *Ergonomics:*
 - *Consistency:*
 - *The main game board is always on the left side of the screen and the side bar is always located on the right side of the screen this would allow less mouse movement for the user which is helpful to the user*
 - *The main button which the players use to play the game is the roll button which is placed on a specific location on the side bar and thus less movement of the mouse is required and improved the consistency of the program.*
 - *Usability:*
 - *An online help displayed to the user within the program so that the user to access the game more readily*
 - *The program is tested by peers to improve the usability by checking if the program is entertaining and intuitive or not*
- *Accessibility:*
 - *Written and visual instructions for the people with hearing disabilities so that they are able to access the program. This makes my program more accessible to the people with hearing impairment.*
- *Ease Of Use:*
 - *The main button which the players use to play the game is the roll button which is placed on a specific location on the side bar and thus less movement of the mouse is required and improved the ease of use of the product.*
- *Plagiarism-*
 - *The music file used in the application is taken from a “Royalty Free Music” website and credit has been given to the website to avoid plagiarism.*
 - *My application has been inspired from Hasbro’s game “Monopoly” and the credit has been given to the application in the bibliography. Many features of the game have also been changed to avoid plagiarism*