

**DEVON WIJESINGHE**

**UoW-w1654187**

**IIT-2016319**

**OOP Coursework 2**

**with typescript ,NoSQL and Play framework**

Table of Contents

[Introduction 3](#_Toc502862429)

[Technologies that must be used 3](#_Toc502862430)

[Functional Requirements 3](#_Toc502862431)

[Non-functional Requirements 4](#_Toc502862432)

[Screen shots 6](#_Toc502862433)

[Project Structure 6](#_Toc502862434)

[Main page UI 7](#_Toc502862435)

[Statistics UI 7](#_Toc502862436)

[Firebase NoSQL database 8](#_Toc502862437)

[A single, player object’s stats saved in the database is show below 8](#_Toc502862438)

[Some JSON Objects in firebase 9](#_Toc502862439)

[Errors, warnings and info popups 10](#_Toc502862440)

[Responsiveness 11](#_Toc502862441)

[Code (TYPESCRIPT) 12](#_Toc502862442)

[ISymbol Inferface 12](#_Toc502862443)

[Symbol Class 12](#_Toc502862444)

[Reel Class 12](#_Toc502862445)

[MainGUI HTML 12](#_Toc502862446)

# Introduction

This coursework is specifically to test the ability to use object oriented programming principles to design and develop graphical user interfaces and also the ability handle exceptions and to code in a professional manner (code readability and code reusability)

# Technologies that must be used

* Play framework
* Typescript
* A NoSQL database(Firebase)
* JSON

# Functional Requirements

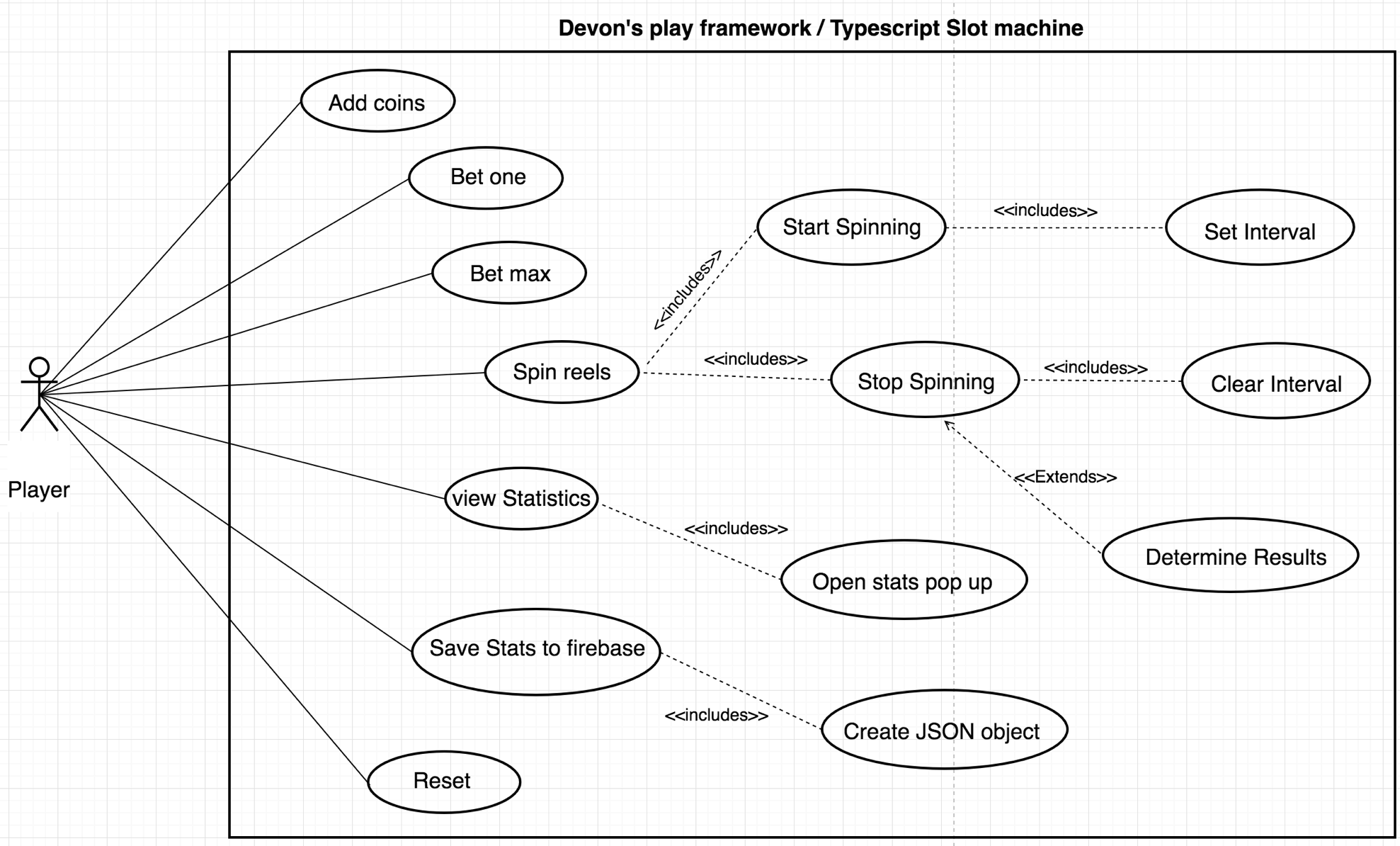
There should be:

* A credit area to display the current credits and an add coin button to add credits. One credit is added when the button is pressed once and the initial number of credits should be 10.
* A bet area to display the amount the player has bet
* A bet max button to bet 3 credits (it can only be pressed once in a single spin)
* A bet one button to bet 1 credit each time that button is pressed
* A reset button to reset the graphical user interface (restore the bets and credits)
* A Spin button to start the reels spinning (3 independent images changing randomly)
* Able stop the reals from spinning by click on either one of them
* A statistics button which will open a statistics page to view the wins, losses, and the average number of credits that he/she has netted per game.
* A save button in the stats window to save the **stats to a NoSQL database(Firebase)**
* Validation - A warning message will pop up if the user presses the bet Max but there is not enough credit or if the user tries to play with no money.

# Non-functional Requirements

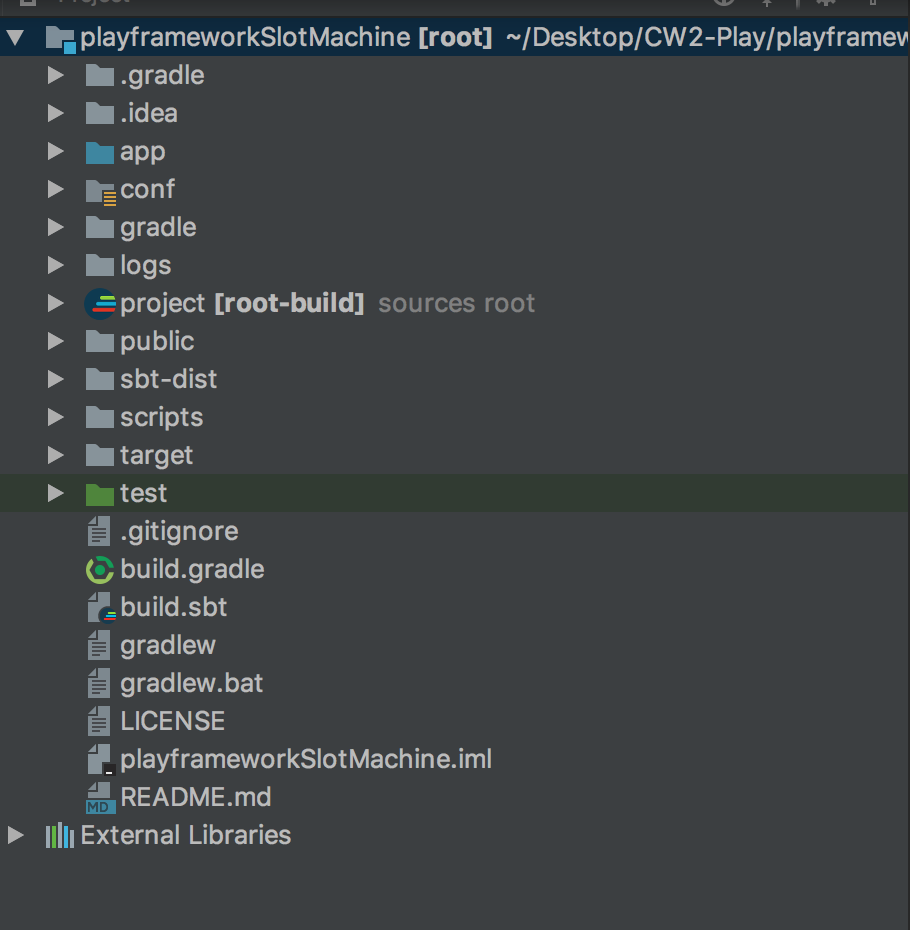
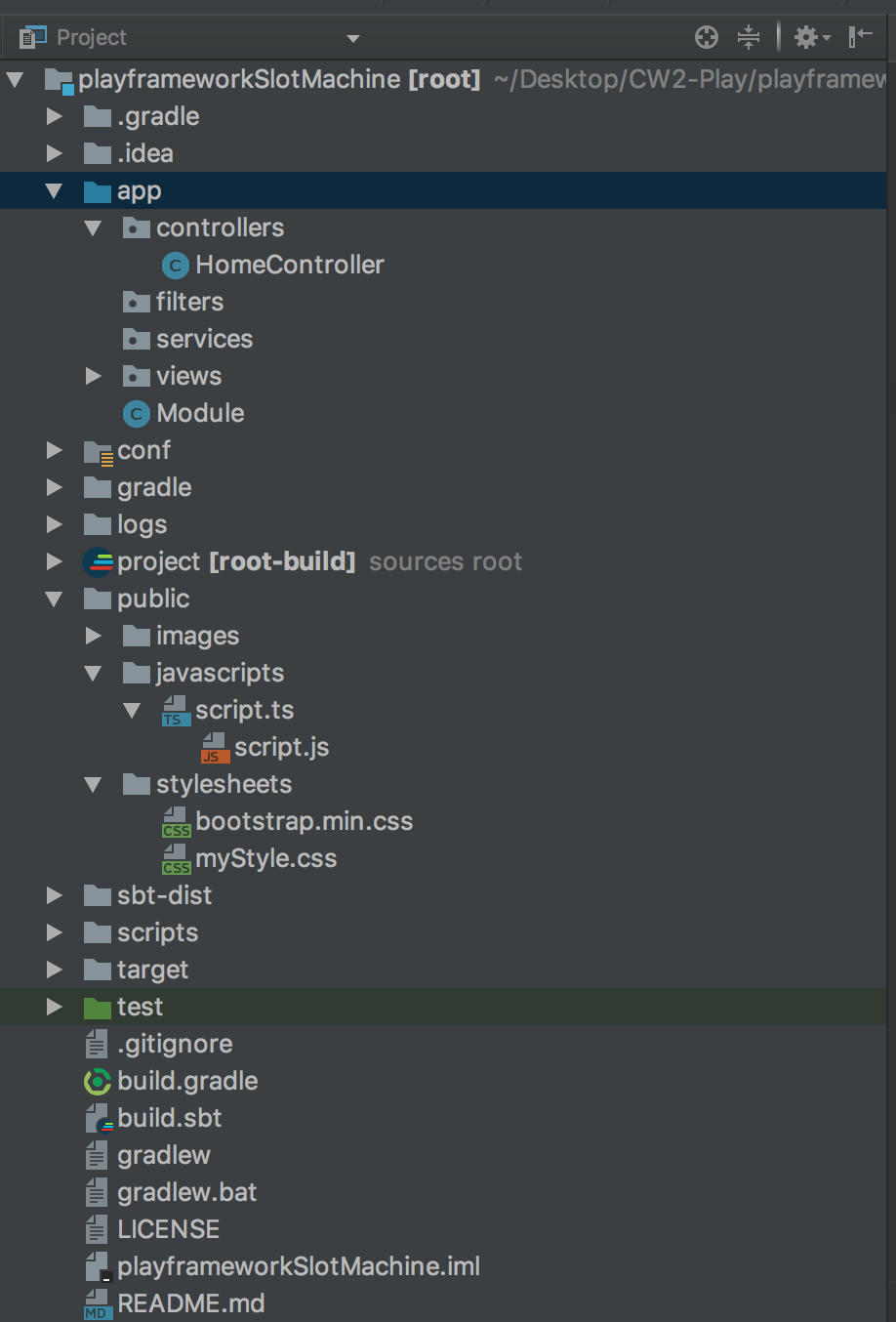
* User friendly graphical user interface
* Responsive
* Exceptions should be handled without letting the program to crash
* Code should be readable and reusable

# Use case



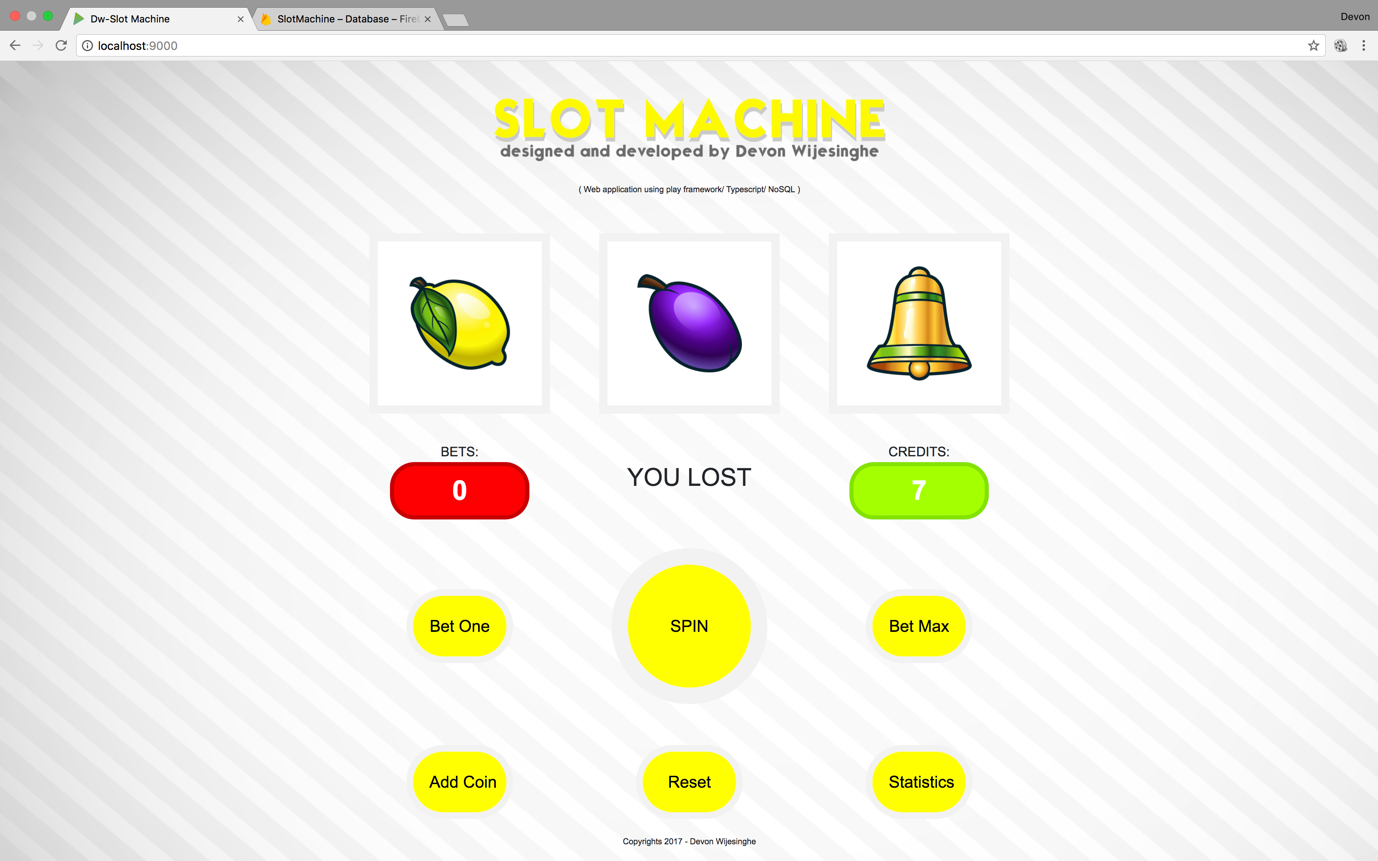
# Screen shots

## Project Structure

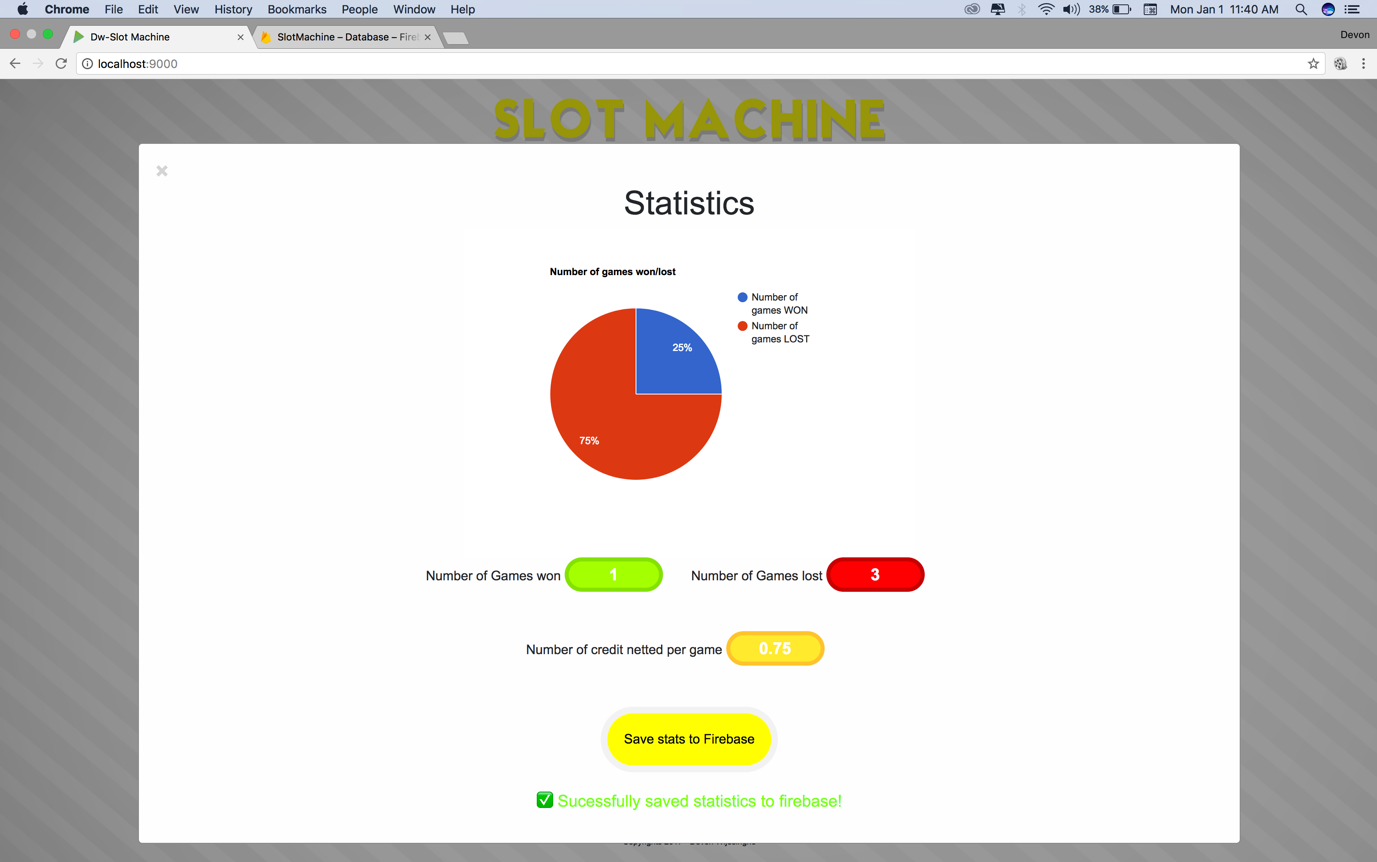


Expanded view

## Main page UI

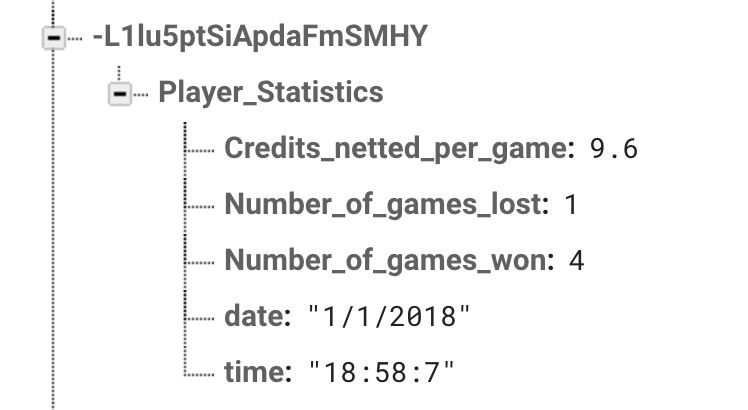


## Statistics UI



## Firebase NoSQL database

### A single, player object’s stats saved in the database is show below



## Some JSON Objects in firebase

{

"-L1kJijhcSwOfV4BMxQ1" : {

"Player\_Statistics" : {

"Credits\_netted\_per\_game" : 0,

"Number\_of\_games\_lost" : 1,

"Number\_of\_games\_won" : 0,

"date" : "1/1/2018",

"time" : "11:35:12"

}

"-L1kKwow9-LYCOBPmopA" : {

"Player\_Statistics" : {

"Credits\_netted\_per\_game" : 0.75,

"Number\_of\_games\_lost" : 3,

"Number\_of\_games\_won" : 1,

"date" : "1/1/2018",

"time" : "11:40:32"

}

},

"-L1kW0YlA2Q1rga7V-RO" : {

"Player\_Statistics" : {

"Credits\_netted\_per\_game" : 7.5,

"Number\_of\_games\_lost" : 1,

"Number\_of\_games\_won" : 3,

"date" : "1/1/2018",

"time" : "12:28:55"

}

},

"-L1lu5ptSiApdaFmSMHY" : {

"Player\_Statistics" : {

"Credits\_netted\_per\_game" : 9.6,

"Number\_of\_games\_lost" : 1,

"Number\_of\_games\_won" : 4,

"date" : "1/1/2018",

"time" : "18:58:7"

}

},

"-L1lu8gj9T5s8xDwon87" : {

"Player\_Statistics" : {

"Credits\_netted\_per\_game" : 9.6,

"Number\_of\_games\_lost" : 1,

"Number\_of\_games\_won" : 4,

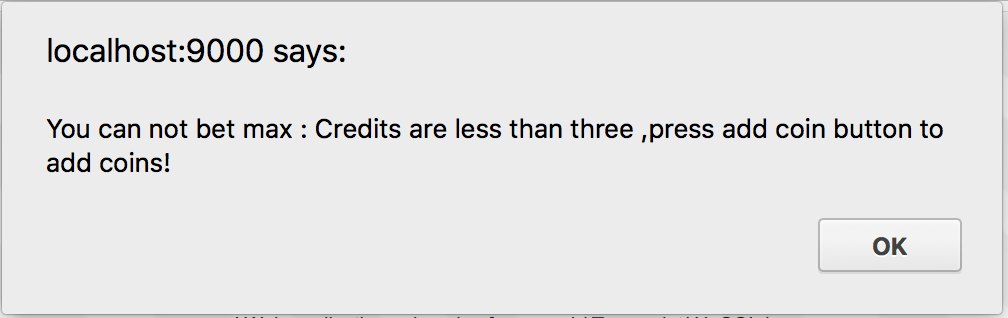
"date" : "1/1/2018",

"time" : "18:58:19"

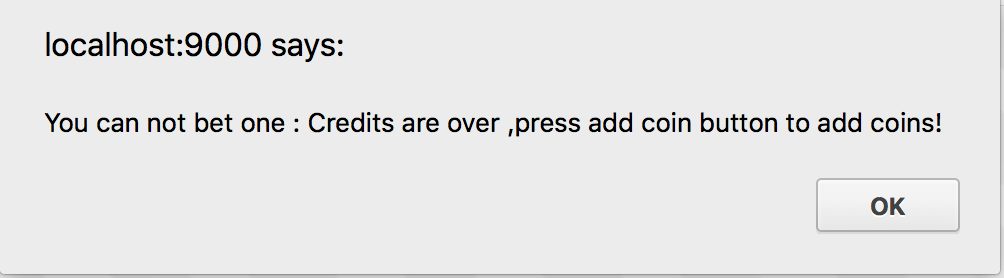
}

## Errors, warnings and info popups

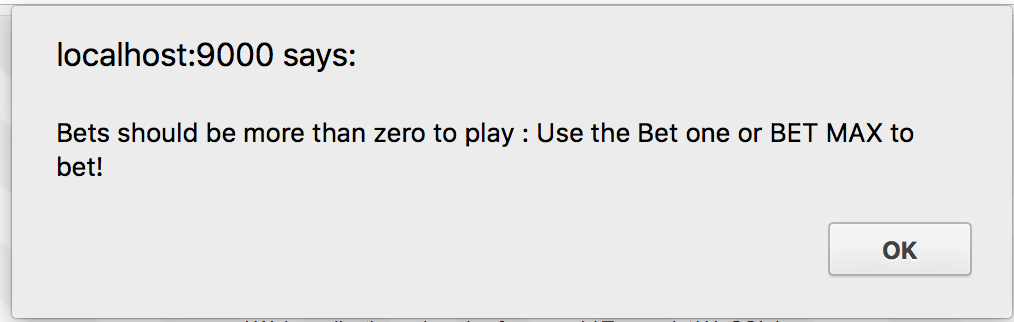
This error is shown when player tries to press the bet max button when the credits are less than 3



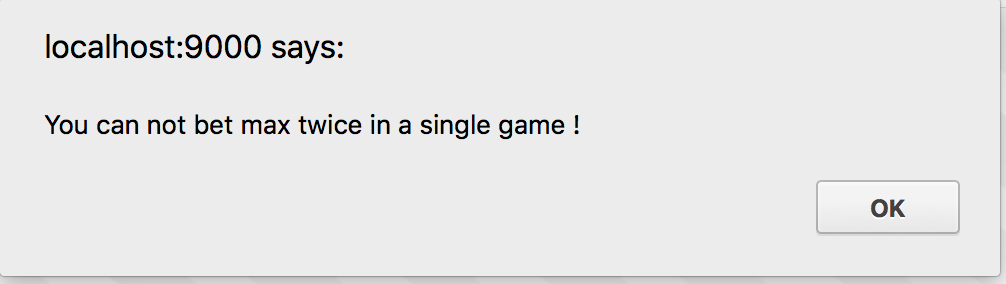
This error is shown when player tries to press the bet one button when the credits are less than 1



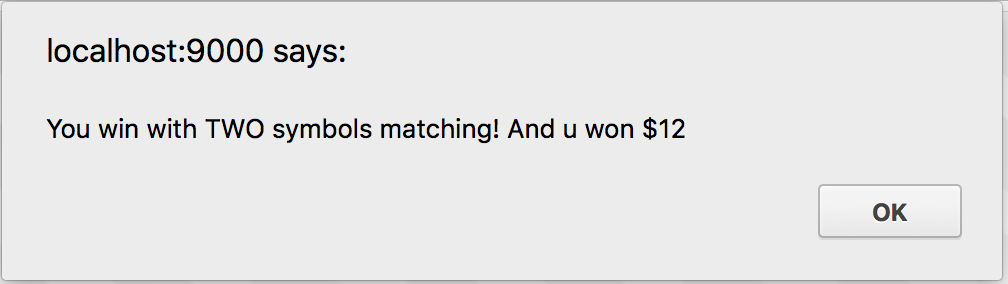
This warning is shown when player tries to press the spin button without betting any amount



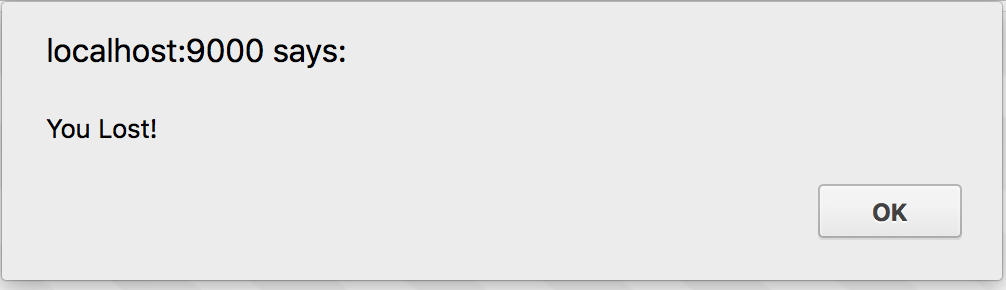
This info popup is show when player tries to press bet max two times in a single game



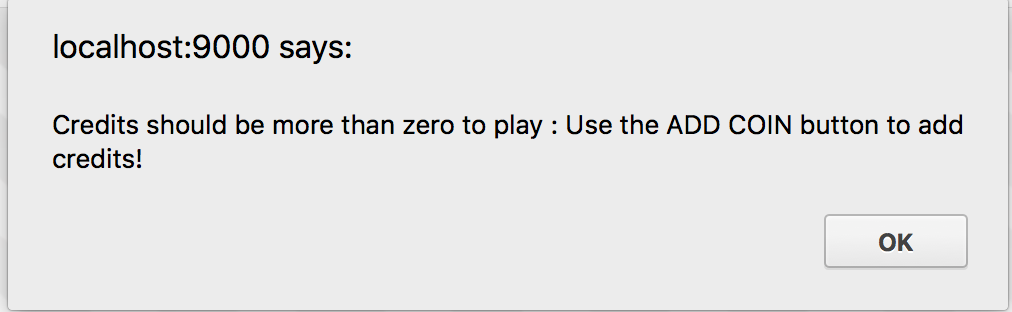
This info popup is show when player wins

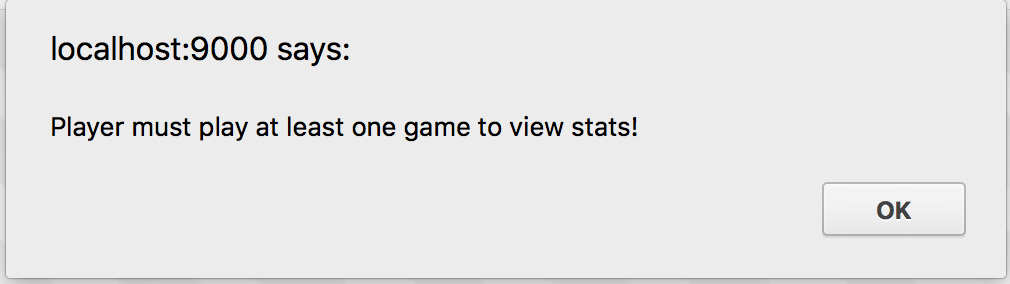


This info popup is show when player loses



This error is shown when player tries to play when the number of credits is zero





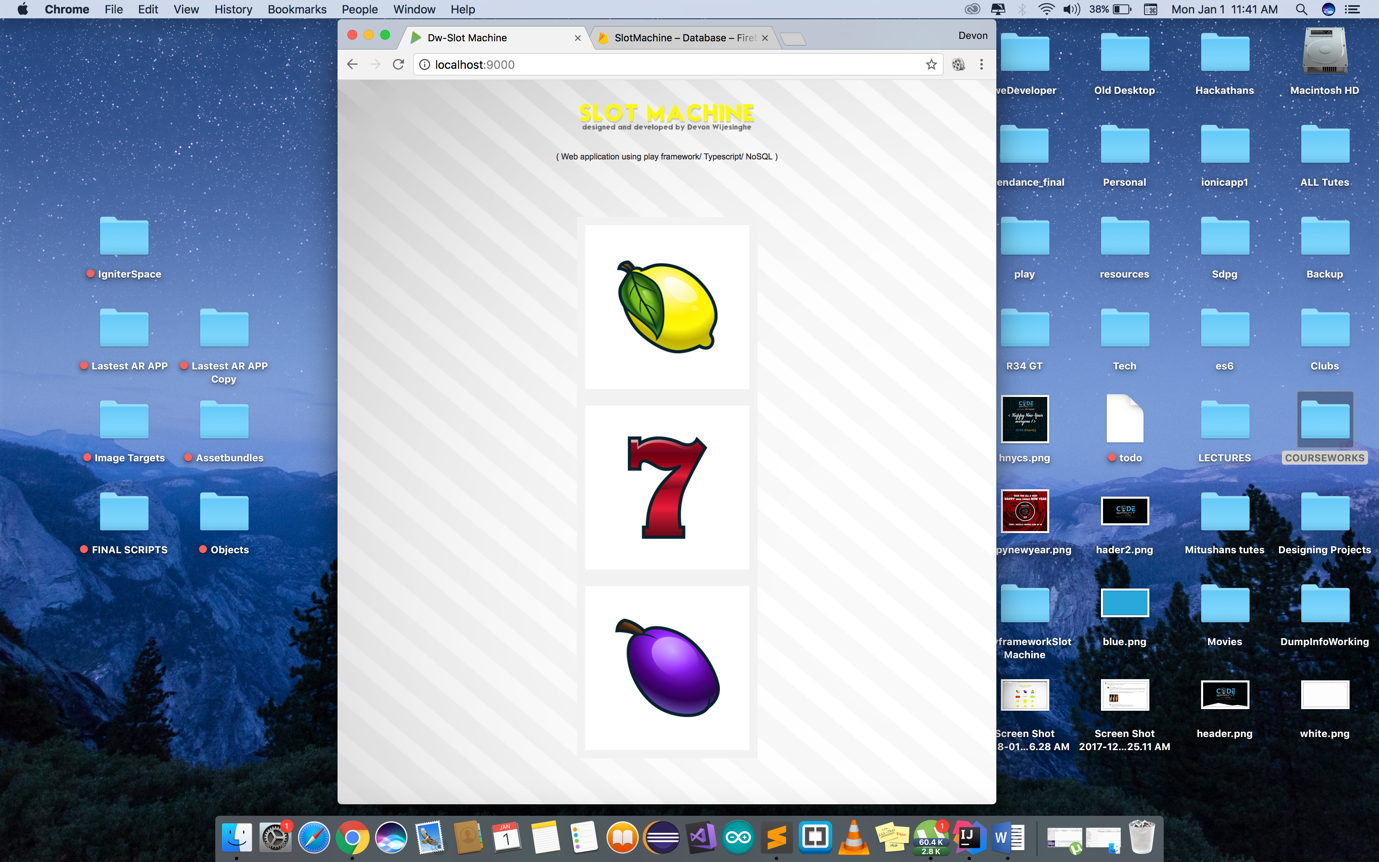
This warning is shown when player tries to press the Statistics button when he/she has not played even a single game

This text will appear when stats are saved to firebase



## Responsiveness





# Code (TYPESCRIPT)

## ISymbol Inferface

1. **interface** ISymbol {
3. //this method sets the image associated with one of the symbols in a reel
4. setImage(s: String): **void**;
6. //this method return the image
7. getImage(): String;
9. //this method which sets the value of the symbol
10. setValue(v: number): **void**;
12. //this method returns the value of the symbol.
13. getValue(): number;
15. }

## Symbol Class

1. **class** Symbol **implements** ISymbol {

4. //instance variables of the symbol class
5. **private** url: string; //stores the location of the image
6. **private** value: number; //stores the value accoiated with the image
7. symbolName: string;//this holds the name of the symbol
9. //the constructor of the symbol class
10. constructor(symbolName: string, value: number) {
11. **this**.setImage(symbolName);
12. **this**.setValue(value);
13. }
15. //this methods are overidden methods of the isymbol interface
16. setImage(symbolName: String): **void** {
17. **this**.url = "assets/images/" + symbolName + ".png";
18. }
20. getImage(): String {
21. **return** **this**.url; //to get the url of image
22. }
24. setValue(v: number): **void** {
25. **this**.value = v;
26. }
28. getValue() {
29. **return** **this**.value; //to get the value
30. }

33. }

## Reel Class

1. **class** Reel {
3. //this array is made to contontain symbol objects
4. **private** **static** arrayOfSymbols = []; //the type is  symbols array
6. **private** reelIntervalState; //this variable saves the state of the reel
8. **private** \_finalValue: number;//this variable is used to store the final url when the reel is stopped

11. //reel constructor
12. constructor() {
13. }
15. //this method adds the symbols the the arrayOfSymbols and returns it
16. **public** **static** getArrayOfSymbols() {
17. //pushing new symbol objects the the array of symbols
18. **this**.arrayOfSymbols.push(**new** Symbol("redseven", 7));
19. **this**.arrayOfSymbols.push(**new** Symbol("bell", 6));
20. **this**.arrayOfSymbols.push(**new** Symbol("watermelon", 5));
21. **this**.arrayOfSymbols.push(**new** Symbol("plum", 4));
22. **this**.arrayOfSymbols.push(**new** Symbol("lemon", 3));
23. **this**.arrayOfSymbols.push(**new** Symbol("cherry", 2));
25. **return** **this**.arrayOfSymbols; //returning the filled array
27. }
29. //this function spin a single reel
30. **public** spinAReel(anyReelHtmlObj, arrayOfSymbols: Symbol[]) {
32. let self = **this**;//this is to avoid final
34. **this**.reelIntervalState = intervalSetter();//this is to get the state so the the reels can be stopped
36. **function** intervalSetter(): number {
37. **return** setInterval(() => {
39. let randomNumber = Math.floor(Math.random() \* 6);//this will generate a random number between 1 and 6
41. anyReelHtmlObj.src = arrayOfSymbols[randomNumber].getImage(); //setting the html obj with a random img src
43. self.\_finalValue = arrayOfSymbols[randomNumber].getValue(); //getting the value of the finnaly left out symbols

46. }, 80); //delay for 80 milli seconds
47. }

50. }
52. stopAReel() {
53. //This method will make the current reel stop spinning
54. clearInterval(**this**.reelIntervalState);
55. }
57. //getters
58. get getFinalValue(): number {
59. **return** **this**.\_finalValue;
60. }
62. }

## Controller Class

1. **class** Controller {
3. //these static class varaible will hold the bet amount and credit amont in a single game
4. **static** betAmount: number = 0;
5. **static** creditAmount: number = 10;
7. **static** totalBets: number = 0;
8. **static** totalWonAmount: number = 0;
10. **static** numGamesWon: number = 0;
11. **static** numGamesLost: number = 0;
13. //this variable is used a flag to check if bet max is already pressed
14. **static** betMaxCount: number=0;
16. //TODO
18. //this function will start spinning the three reels
19. **static** startSpinning() {
21. //checking if credits is more than zero
22. **if** (!(Controller.creditAmount > 0)) {
24. alert("Credits should be more than zero to play : Use the ADD COIN button to add credits!")
26. } **else** **if** (!(Controller.betAmount > 0)) {
28. alert("Bets should be more than zero to play : Use the Bet one or BET MAX to bet!")
30. } **else** {
31. //getting the symbols array
32. let symbolsArray = Reel.getArrayOfSymbols();
34. //creating reel objects
35. reel1 = **new** Reel();
36. reel2 = **new** Reel();
37. reel3 = **new** Reel();
39. //calling the spinARell method for all three reels
40. reel1.spinAReel(reel1HtmlObj, symbolsArray);
41. reel2.spinAReel(reel2HtmlObj, symbolsArray);
42. reel3.spinAReel(reel3HtmlObj, symbolsArray);
44. //disabling the spin button so it can be pressed only once
45. spinBtn.disabled = **true**;
46. //disabling the rest button so it can not be pressed while spiining
47. btnReset.disabled = **true**;
48. //disabling other buttons
49. btnAddCoin.disabled = **true**;
50. btnStatistics.disabled = **true**;
51. btnBetOne.disabled = **true**;
52. btnBetMax.disabled = **true**;
54. //updating ui message
55. tfMessage.innerHTML = "SPINNING...";
56. Controller.betMaxCount=0;
58. }

61. }
63. //this function stops the three reels from spinning
64. **static** stopSpinningReels() {
65. //this method will check if the player has won or lost and displays it.
66. Controller.determineResult();
68. //Stopping all three reels
69. reel1.stopAReel()
70. reel2.stopAReel()
71. reel3.stopAReel()
73. //enableing the spin button
74. spinBtn.disabled = **false**;
75. //enabling the bet max button
76. btnBetMax.disabled = **false**;
77. //enabling the rest button
78. btnReset.disabled = **false**;
80. //enabling other buttons
81. btnAddCoin.disabled = **false**;
82. btnStatistics.disabled = **false**;
83. btnBetOne.disabled = **false**;

86. }
88. //================= CALCULATION AREA ============================
90. //this method will check if the player has won or lost and displays it.
91. **static** determineResult() {
93. let reel1Val: number = reel1.getFinalValue;
94. let reel2Val: number = reel2.getFinalValue;
95. let reel3Val: number = reel3.getFinalValue;
97. //this condition checks if all thee reels have the same symbol
98. **if** (reel1Val == reel2Val && reel1Val == reel3Val) {
100. //this method calcalte the final won amount and also update Ui
101. let wonAmount = Controller.calculateWonAmountAndUpdateUI(reel1Val);
103. //incrementing the num of games won variable
104. Controller.numGamesWon++;
106. tfMessage.innerHTML = "YOU WIN";
107. alert("You win with ALL symbols matching! And you won $" + wonAmount)

110. //this condition checks if 1st two reel are same or 1st and 3rd reels are with same valued symbols
111. } **else** **if** (reel1Val == reel2Val || reel1Val == reel3Val) {
113. //incrementing the num of games won variable
114. Controller.numGamesWon++;
116. //this method calcalte the final won amount and also update Ui
117. let wonAmount = Controller.calculateWonAmountAndUpdateUI(reel1Val);
118. tfMessage.innerHTML = "YOU WIN";
119. alert("You win with TWO symbols matching! And u won $" + wonAmount)

122. //this condition check if the second and third reel have the same symbol value
123. } **else** **if** (reel2Val == reel3Val) {
125. //incrementing the num of games won variable
126. Controller.numGamesWon++;
128. //this method calcalte the final won amount and also update Ui
129. let wonAmount = Controller.calculateWonAmountAndUpdateUI(reel2Val);
131. tfMessage.innerHTML = "YOU WIN";
132. alert("You win with TWO symbols matching! And u won $" + wonAmount);
134. } **else** {
135. //incrementing the num of games Lost variable
136. Controller.numGamesLost++;
138. tfMessage.innerHTML = "YOU LOST";
139. alert("You Lost!")
141. //resetting the bet amount to zero
142. Controller.betAmount = 0;
143. //== updating UI ==
144. tfBets.value = 0;
145. }
147. }
149. //===========
151. //this method calcalte the final won amount and also update Ui -- it also return the won amount
152. **static** calculateWonAmountAndUpdateUI(val: number): number {
154. //calculating won amount
155. let wonAmount = val \* Controller.betAmount;
157. //summing up the total credits won
158. Controller.totalWonAmount += wonAmount;
159. //summing up the total bets
160. Controller.totalBets += Controller.betAmount;
162. //resetting the bet amount to zero
163. Controller.betAmount = 0;
165. //adding the won amount to the credits are
166. Controller.creditAmount += wonAmount;
168. //== updating UI ==
169. tfBets.value = 0;
170. tfCredits.value = Controller.creditAmount;

173. **return** wonAmount;//return the won amount
174. }
176. //=============================================
178. //this method will add 1 credits to the bet area
179. **static** betOne(): **void** {
181. //checcking if credits is more than 0
182. **if** (Controller.creditAmount > 0) {
183. //decrementing the credit amount
184. --Controller.creditAmount;
185. //incrementing the bet amount
186. ++Controller.betAmount;
187. } **else** {
188. alert("You can not bet one : Credits are over ,press add coin button to add coins!")
189. }

192. //== updating ui ==
193. //setting with credit amount
194. tfCredits.value = Controller.creditAmount;
195. //setting with bet amount
196. tfBets.value = Controller.betAmount;
198. }


202. //this method will add 3 credits to the bet area
203. **static** betMax(): **void** {
205. //checcking if credits is more than 0
206. **if** (Controller.creditAmount > 3) {
208. //checking if the bet max button is already clicked once
209. **if** (Controller.betMaxCount==0){
211. //decrementing the credit amount
212. Controller.creditAmount -= 3;
213. //incrementing the bet amount
214. Controller.betAmount += 3;
216. Controller.betMaxCount=1;
218. }**else** {
220. alert("You can not bet max twice in a single game !")
221. }
223. } **else** {
224. alert("You can not bet max : Credits are less than three ,press add coin button to add coins!")
225. }

228. //== updating ui ==
229. //setting with credit amount
230. tfCredits.value = Controller.creditAmount;
231. //setting with bet amount
232. tfBets.value = Controller.betAmount;
234. }
236. //this method will add 1 credits to the credit area each time the add coin button is pressed
237. **static** addCoin(): **void** {
239. //incrementing the credits
240. ++Controller.creditAmount;
241. //== updating ui ==
242. //setting with credit amount
243. tfCredits.value = Controller.creditAmount;
245. }
247. //this method will add 1 credits to the credit area each time the add coin button is pressed
248. **static** reset(): **void** {

251. //setting to the orginal credita amount
252. Controller.creditAmount += Controller.betAmount;
254. //setting the bet amount to zero
255. Controller.betAmount = 0;
257. //== updating ui ==
258. //setting with credit amount
259. tfCredits.value = Controller.creditAmount;
260. //setting with bet amount
261. tfBets.value = Controller.betAmount;
263. //enableing the spin button
264. spinBtn.disabled = **false**;
265. //enabling the bet max button
266. btnBetMax.disabled = **false**;
268. }
270. //this method will take go to the stats page and draws pie chart
271. **static** viewStatistics(): **void** {
273. //checking if the player has played atleast one game
274. **if** (!(Controller.numGamesLost >0 || Controller.numGamesWon>0)){
276. alert("Player must play at least one game to view stats!")
278. }**else** {
280. // Load google charts
281. google.charts.load('current', {'packages': ['corechart']});
282. google.charts.setOnLoadCallback(drawChart);
284. // Draw the chart and set the chart values
285. **function** drawChart() {
286. **var** data = google.visualization.arrayToDataTable([
287. ['Task', 'Games won/lost'],
288. ['Number of games WON', Controller.numGamesWon],
289. ['Number of games LOST', Controller.numGamesLost],
291. ]);
293. // Optional; add a title and set the width and height of the chart
294. **var** options = {'title': 'Number of games won/lost', 'width': 550, 'height': 400};
295. // Display the chart inside the <div> element with id="piechart"
296. **var** chart = **new** google.visualization.PieChart(document.getElementById('piechart'));
297. chart.draw(data, options);
298. }
300. //Updating UI Values
301. tfWon.value = Controller.numGamesWon;
302. tfLost.value = Controller.numGamesLost;
303. tfNet.value = (Controller.totalWonAmount - Controller.totalBets ) / (Controller.numGamesWon + Controller.numGamesLost);
305. //making the stats part visible
306. statsContainer.style.display = "block";
308. }
310. }

313. //this method will make a player object which can be used to save stats to fire base
314. **static** savePlayerStats(): **void** {
316. //creating a player object
317. let playerObject = **new** Player(Controller.numGamesWon, Controller.numGamesLost, Controller.totalBets, Controller.totalWonAmount);
319. //creating a date and time object
320. let currentdate = **new** Date();
322. //making a string of the date
323. let date = currentdate.getDate() + "/"
324. + (currentdate.getMonth() + 1) + "/"
325. + currentdate.getFullYear();
326. //making a string of the time
327. let time = currentdate.getHours() + ":"
328. + currentdate.getMinutes() + ":"
330. + currentdate.getSeconds();
332. // =================== FIRE BASE Database CONNECTIVITY ===================
333. //Creating a fire base reference
334. let firebaseRef = firebase.database().ref();
336. //pushing a player object as a JSON to firebase
337. firebaseRef.push({
338. Player\_Statistics: {
339. date: date,
340. time: time,
341. Number\_of\_games\_won: playerObject.getNumGamesWon,
342. Number\_of\_games\_lost: playerObject.getNumGamesLost,
343. Credits\_netted\_per\_game: playerObject.calculateNetCreditsPerGame()
344. }
345. });
347. msgSucess.style.display="block";
349. }

352. }

## Player class

1. **class** Player {
3. //instance variables of the player
4. **private** \_numGamesWon: number;
5. **private** \_numGamesLost: number;
6. **private** \_totalBets: number;
7. **private** \_totalWonAmount: number;
8. **private** \_netCreditsPerGame: number;

11. //player constructor
12. constructor(numGamesWon: number, numGamesLost: number, totalBets: number, \_totalWonAmount: number) {
14. **this**.\_numGamesWon = numGamesWon;
15. **this**.\_numGamesLost = numGamesLost;
16. **this**.\_totalBets = totalBets;
17. **this**.\_totalWonAmount = \_totalWonAmount;
19. }
21. //this method calculates the number of credits netted per game
22. calculateNetCreditsPerGame(): number {
23. **return** **this**.\_netCreditsPerGame = (**this**.\_totalWonAmount - **this**.\_totalBets) / (**this**.\_numGamesWon + **this**.\_numGamesLost);
24. }
26. //getters
27. **public** get getNumGamesWon(): number {
28. **return** **this**.\_numGamesWon;
29. }
31. get getNumGamesLost(): number {
32. **return** **this**.\_numGamesLost;
33. }
35. get getTotalBets(): number {
36. **return** **this**.\_totalBets;
37. }
39. get getTotalWonAmount(): number {
40. **return** **this**.\_totalWonAmount;
41. }
43. get getNetCreditsPerGame(): number {
44. **return** **this**.\_netCreditsPerGame;
45. }

48. }

## MainGUI HTML (slotMachine.scala.html)

1. @(message: String, style: String = "java")
3. @defining(play.core.PlayVersion.current) { version =**>**
5. **<div** class="container-fluid"**>** <!-- This is the main container  -->
6. <!-- START OF HEADER  -->
7. **<div** class="row header"**>** <!-- This row contains the header  -->
8. **<div** class="col-sm-12"**>**
9. **<div** class=""**>**
10. **<img** class="logo" src="@routes.Assets.versioned("images/logo.png")" **>**
11. **<p>**( Web application using play framework/ Typescript/ NoSQL )**</p>**
12. **</div>**
13. **</div>**
14. **</div>**
15. <!-- END OF HEADER  -->
16. <!-- START OF REELS AREA  -->
17. **<div** class="row"**>** <!-- This row contains the three reels  -->
18. **<div** class="col-lg-3" **>**
19. **<p>** **</p>**
20. **</div>**
22. **<div** class="col-lg-2" **>**
23. **<div** class="reel"  **>**
25. **<img**  class="reelImg"  id="reel1HtmlObj" src="@routes.Assets.versioned("images/redseven.png")" **>**
26. **</div>**
27. **</div>**
29. **<div** class="col-lg-2" **>**
30. **<div** class="reel" **>**
31. **<img** class="reelImg"  id="reel2HtmlObj" src="@routes.Assets.versioned("images/redseven.png")"**>**
32. **</div>**
33. **</div>**
35. **<div** class="col-lg-2" **>**
36. **<div** class="reel" **>**
37. **<img** class="reelImg"  id="reel3HtmlObj" src="@routes.Assets.versioned("images/redseven.png")"**>**
38. **</div>**
39. **</div>**
41. **<div** class="col-lg-3" **>**
42. **<p>** **</p>**
43. **</div>**
44. **</div>**
45. <!-- END OF REELS AREA  -->
46. <!-- START OF TEXT FILED'S AND MESSAGE AREA ROW (CREDIT AREA AND BET AREA AND A MESSAGE AREA)  -->
47. **<div** class="row"**>**
49. **<div** class="col-lg-3" **>**
50. **<p>** **<p>**  <!-- empty column  -->
51. **</div>**
52. **<div** class="col-lg-2" **>**
53. **<div** class="textArea" **>**
54. BETS:
55. **<input** class="tfBets" type="text" name="tfBets" disabled="true" id="tfBets" value="0"**>**
56. **</div>**
57. **</div>**
58. **<div** class="col-lg-2" **>**
59. **<div** class="textArea" **>**
60. **<p** class="tfMessage" id="tfMessage"**>** WELCOME **</p>**
61. **</div>**
62. **</div>**
63. **<div** class="col-lg-2" **>**
64. **<div** class="textArea" **>**
65. CREDITS:
66. **<input** class="tfCredits" type="text" name="tfCredits" disabled="true" id="tfCredits" value="10"**>**
67. **</div>**
68. **</div>**
69. **<div** class="col-lg-3" **>**
70. **<p>** **</p>** <!-- empty column  -->
71. **</div>**
72. **</div>**
73. <!-- END OF TEXT FILED'S AND MESSAGE AREA ROW (CREDIT AREA AND BET AREA AND A MESSAGE AREA)  -->
74. <!-- START OF SPIN BUTTON ROW -->
75. **<div** class="row"**>**
76. **<div** class="col-lg-3" **>**
77. **<p>** **</p>** <!-- empty column  -->
78. **</div>**
79. **<div** class="col-lg-2" **>**
80. **<input** class ="btnCommon" type="button" name="btnBetOne" value="Bet One" id="btnBetOne"**>**
81. **</div>**
82. **<div** class="col-lg-2" **>**
83. <!-- SPIN BUTTON  -->
84. **<div** class="" **>**
85. **<input** class="spinBtn" id="btnSpin" type="button" name="btnSpin" value="SPIN" **>**
86. **</div>**
87. **</div>**
89. **<div** class="col-lg-2" **>**
90. **<input** class ="btnCommon" type="button" name="btnBetMax" value="Bet Max" id="btnBetMax"**>**
91. **</div>**

94. **<div** class="col-lg-3" **>**
95. **<p>** **</p>** <!-- empty column  -->
96. **</div>**
98. **</div>**
99. <!-- END OF SPIN BUTTON ROW -->
100. <!-- START OF OTHER BUTTONS AREA -->
101. **<div** class="row"**>**
102. **<div** class="col-lg-3" **>**
103. **<p></p>**
104. **</div>**
105. **<div** class="col-lg-2" **>**
106. **<input** class ="btnCommon" type="button" name="btnAddCoin" value="Add Coin" id="btnAddCoin"**>**
107. **</div>**
108. **<div** class="col-lg-2" **>**
109. **<input** class ="btnCommon" type="button" name="btnReset" value="Reset" id="btnReset"**>**
110. **</div>**
111. **<div** class="col-lg-2" **>**
112. **<input** class ="btnCommon" type="button" name="btnStatistics" value="Statistics" id="btnStatistics"**>**
113. **</div>**
114. **<div** class="col-lg-3" **>**
115. **<p></p>**
116. **</div>**
117. **</div>**
118. <!-- END OF OTHER BUTTONS AREA -->
119. **<div** class="row"**>**
120. **<div** class="col-lg-12"**>**
121. **<p>**Copyrights 2017 - Devon Wijesinghe**</p>**
122. **</div>**
123. **</div>**
124. **</div>**

127. <!-- STATISTICS POP UP BOX -->
129. **<div** id="statsContainer" class="statsContainer"**>**

132. **<div** class="statsContent"**>**
133. **<span** id="close" class="close"**>**×**</span>**
134. **<center>**
135. **<h1>**Statistics**</h1>**
137. **<div** id="piechart"**>** Pie chart **</div>**
139. Number of Games won
140. **<input** class="tfWon" type="text" name="tfWon" disabled="true" id="tfWon" value="0"**>**
141. Number of Games lost
142. **<input** class="tfLost" type="text" name="tfLost" disabled="true" id="tfLost" value="0"**>**
143. **<br>**
144. **<br>**
145. **<br>**
146. Number of credit netted per game
147. **<input** class="tfNet" type="text" name="tfNet" disabled="true" id="tfNet" value="0"**>**
148. **<br>**
149. **<input** class ="btnSave" type="button" name="btnSave" value="Save stats to Firebase" id="btnSave"**>**
151. **<p** class="msgSucess" id="msgSucess"**>**  ✅ Sucessfully saved statistics to firebase!**</p>**
152. **</center>**
154. **</div>**
156. **</div>**
157. <!-- END -->
158. }

## main.scala.html (contains fire base config)

1. <!DOCTYPE html**>**
2. **<html** lang="en"**>**
3. **<head>**
4. @\* Here's where we render the page title `String`. \*@
5. **<title>**@title**</title>**
7. **<link** rel="shortcut icon" type="image/png" href="@routes.Assets.versioned("images/favicon.png")"**>**
9. **<meta** charset="utf-8"**>**
10. **<meta** name="viewport" content="width=device-width, initial-scale=1"**>**
12. @\*Bootstrap linking \*@
13. **<link** rel="stylesheet" media="screen" href="@routes.Assets.versioned("stylesheets/bootstrap.min.css") "**>**
15. @\*linking my style\*@
16. **<link** rel="stylesheet" media="screen" href="@routes.Assets.versioned("stylesheets/myStyle.css")"**>**

19. **</head>**
20. **<body>**
21. @\* And here's where we render the `Html` object containing
22. \* the page content. \*@
23. @content

26. <!-- //firebase Configeration scripts -->
27. **<script** src="https://www.gstatic.com/firebasejs/4.8.1/firebase.js"**></script>**
28. **<script>**
29. // Initialize Firebase
30. var config = {
31. apiKey: "AIzaSyCXArJLycfrpEetTLHaV-vK-PMpbviy\_uc",
32. authDomain: "slotmachine-b6157.firebaseapp.com",
33. databaseURL: "https://slotmachine-b6157.firebaseio.com",
34. projectId: "slotmachine-b6157",
35. storageBucket: "",
36. messagingSenderId: "84742824406"
37. };
38. firebase.initializeApp(config);
39. **</script>**

42. **<script** type="text/javascript" src="https://www.gstatic.com/charts/loader.js"**></script>**
43. **<script** src="@routes.Assets.versioned("javascripts/script.js")" type="text/javascript"**></script>**

46. **</body>**
48. **</html>**

## HomeController class

1. **package** controllers;
3. **import** play.mvc.\*;
5. **import** views.html.\*;

8. **public** **class** HomeController **extends** Controller {

11. //this method renders the index page
12. **public** Result index() {
13. **return** ok(index.render("Slot machine is ready."));
14. }

17. }