Problems-Mentor: Initial Commit without DB logic #2

Open **mgreiler** wants to merge 1 commit into `main` from `redeem-app-pr-1-problems`

2 .gitignore

```
...  ...  @@ -0,0 +1,2 @@
     1  + .DS_Store
     2  + node_modules
```

694 redeem-app/package-lock.json

Load diff

Large diffs are not rendered by default.

15 redeem-app/package.json

```
...  ...  @@ -0,0 +1,15 @@
     1  + {
     2  +   "name": "redeem",
     3  +   "version": "1.0.0",
     4  +   "description": "",
     5  +   "main": "server.js",
     6  +   "scripts": {
     7  +     "start": "node server.js",
     8  +     "test": "echo \"Error: no test specified\" && exit 1"
     9  +   },
    10  +   "author": "",
    11  +   "license": "ISC",
    12  +   "dependencies": {
    13  +     "express": "^4.17.1"
    14  +   }
    15  + }
```

49 redeem-app/public/index.html

```
...  ...  @@ -0,0 +1,49 @@
     1  + <html>
     2  +     <head>
     3  +         <title>Redeem Credits</title>
     4  +     </head>
```

```
+     <body>
+         <form>
+             <div>
+                 <input id="Username" value="Your username">
+                 <br/>
+                 <label for="credits">Redeem Credits:</label>
+                 <input id="credits" value="0">
+             </div>
+             <div>
+                 <button id="sendButton">Send</button>
+             </div>
+         </form>
+         <div id="statusText"></div>
+     </body>
+     <script>
+         const urlString = window.location.href;
+         const url = new URL(urlString);
+         const userId = url.searchParams.get("userId") || 0;
+ 
+         const sendButton = document.getElementById('sendButton');
+         sendButton.addEventListener('click', (e) => {
+                 e.preventDefault();
+                 const credits = document.getElementById("credits").value;
+                 const username = document.getElementById("Username").value;
+                 fetch('/redeem', {
+                     method: 'POST',
+                     headers: {
+                         'Content-Type': 'application/json',
+                     },
+                     body: JSON.stringify({
+                         userId: userId,
+                         credits: credits,
+                         username: username
+                     })
+                 })
+                 .then(response => response.json())
+                 .then(data => {
+                     document.getElementById('statusText').innerHTML = data.status;
+                 })
+                 .catch((error) => {
+                     document.getElementById('statusText').innerHTML = error.status;
+                 });
+         });
+     </script>
+ </html>
```

118 redeem-app/server.js

```
@@ -0,0 +1,118 @@
+ const express = require("express");
+ const app = express();
+ app.use(express.json());
+ const port = 3000;
```

mgreiler Pending — Owner Author

The server port is hardcoded to 3000, which might limit the application's flexibility in different environments. Consider using an environment variable or a configuration file to allow for easy adjustments without needing to modify the source code directly. This approach can help with deployment across different environments.

Reply...

```
+
+ app.use(express.static(__dirname + "/public"));
+
+ app.get("/", function (req, res) {
+   res.sendFile(__dirname + "/public/index.html");
+ });
+
+ app.post("/redeem", function (req, res) {
+   const credits = req.body.credits;
+   const userId = req.body.userId;
+   const username = req.body.username;
+   console.log(`redeeming ${credits} credits for user: ${username} (${userId})`);
+
+   try {
+     const hours = redeemCredits(credits, userId);
+     res.send({
+       status: `User ${username} redeemed ${credits} credits to get ${hours} hours.`,
+     });
+   } catch (err) {
+     res.status(400).send({
+       status: err,
+     });
+   }
+ });
+
+ app.listen(port, () => {
+   console.log(`API listening at http://localhost:${port}`);
+ });
+
+ /**
+  * The player can purchase gaming hours by redeeming
+  * credits. How many hours one credit is
+  * worth depends on the level of the player.
+  */
+ function redeemCredits(credits, playerId) {
+   playerLevel = getPlayerLevel(playerId);
```

mgreiler Pending — Owner Author

I noticed that variables in redeemCredits, getPlayerLevel, and other functions are declared without var, let, or const, making them global. This could lead to unexpected behavior due to variable name clashes or difficult debugging. It's a best practice to always declare variables with let or const to limit their scope to the block, statement, or expression in which they are used.

Reply...

```
+   hours = convertCreditsToHours(playerLevel, credits);
+
+   redeemHoursToPlayerProfile(hours, credits, playerId);
+   return hours;
+ }
+
+ /**
+  * Returns the level of the player.
+  */
+ function getPlayerLevel(playerId) {
+   levelQuery = "SELECT playerLevel FROM players WHERE playerId = " + playerId;
```

mgreiler Pending — Owner Author

The way we're currently constructing our SQL queries by directly concatenating user inputs, such as in getPlayerLevel and redeemHoursToPlayerProfile, poses a risk for SQL injection attacks.

It would be safer to use prepared statements or parameterized queries to ensure our database interactions are secure.

Reply...

```
52 +   playerLevel = executeQuery(levelQuery);
53 +
54 +   return playerLevel;
55 + }
56 +
57 + /**
58 +  * Adds the purchased hours to the players game hours.
59 +  */
60 + function redeemHoursToPlayerProfile(hours, credits, playerId) {
61 +   hourQuery = "SELECT hours FROM players WHERE playerId = " + playerId;
62 +   oldHours = executeQuery(hourQuery);
63 +
64 +   updateQuery = "Update players SET hours = " + (oldHours + hours) + " WHERE playerId = " +
    playerId;
65 +
66 +   try {
67 +     executeQuery(updateQuery);
68 +   } catch (err) {
69 +     throw new Error("Could not add hours.");
70 +   }
71 +
72 +   chargeCreditsFromPlayer(credits, playerId);
73 + }
74 +
75 + /**
76 +  * Charges the player with the credits redeemed.
77 +  */
78 + function chargeCreditsFromPlayer(credits, playerId) {
79 +   creditQuery = "SELECT credits FROM players WHERE playerId = " + playerId;
80 +   oldCredits = executeQuery(hourQuery);
81 +
82 +   updateQuery = "Update players SET credits = " + (oldCredits - credits) + " WHERE playerId = " +
    playerId;
83 +
84 +   try {
```

mgreiler Pending Owner Author

I've observed that error handling in asynchronous operations, such as database queries, could be more consistent. While there are try-catch blocks in some functions like redeemHoursToPlayerProfile, ensuring all database interactions are wrapped in try-catch blocks will make our application more robust and prevent it from crashing due to unhandled exceptions.

Reply...

```
85 +     executeQuery(updateQuery);
86 +   } catch (err) {
87 +     throw new Error("Could not charge credits: " + err);
88 +   }
89 + }
90 +
91 + /**
92 +  * This method converts the credits to game hours. Gamers that
93 +  * have a lower gaming level get more game hours for their credits.
94 +  * For more advanced gamers, buying new credits is more expensive.
95 +  *
```

```
96  +  * Players of level less than 3, get 3 times the hours of their credit
97  +  * Players of level less than 8, get 1.5 times the hours of their credit
98  +  * Players of level higher than 8, get just the hours of the credit.
99  +  */
100 + function convertCreditsToHours(playerLevel, credits) {
```

mgreiler Pending Owner Author

There seems to be a small logic error in the convertCreditsToHours function. Players with a level exactly equal to 3 will not fall into any of the specified conditions, which might not be the intended behavior. We should adjust the conditions to ensure all player levels are covered appropriately, perhaps by including level 3 in the second condition.

Reply...

```
101 +   if (playerLevel < 3) {
102 +     return 3 * credits;
103 +   } else if (playerLevel > 3 && playerLevel <= 8) {
104 +     return 1.5 * credits;
105 +   } else {
106 +     return 1 * credits;
107 +   }
108 + }
109 +
110 + /** TODO: implement */
111 + function getRandomInt(max) {
112 +   return Math.floor(Math.random() * Math.floor(max));
113 + }
114 +
115 + /** TODO: implement */
116 + function executeQuery() {
117 +   return getRandomInt(13);
118 + }
```