Scrum-40

Context:

In the Viewtasks.cshtml, there is a for each loop that goes through “item” which is the tasks in “Model” which is the list that holds all the task list

```
@foreach (var item in Model)
{
```

Now, the loop adds the tasks to the web page in a fixed order list the same way the admin adds the tasks.

Design: The design is to have the admin add the list of tasks. When the hunt is created it will pull up the list will go through a change where it randomizes the order and displays it for the players

Implementation

```
@{
    // Shuffle the tasks randomly
    var shuffledTasks = Model.OrderBy(x => Guid.NewGuid()).ToList();
}
```

This code is implemented in the same file

The first implementation of trying to complete it. All it does is it take the “Model and it reorganize”

It does display the tasks in a random order

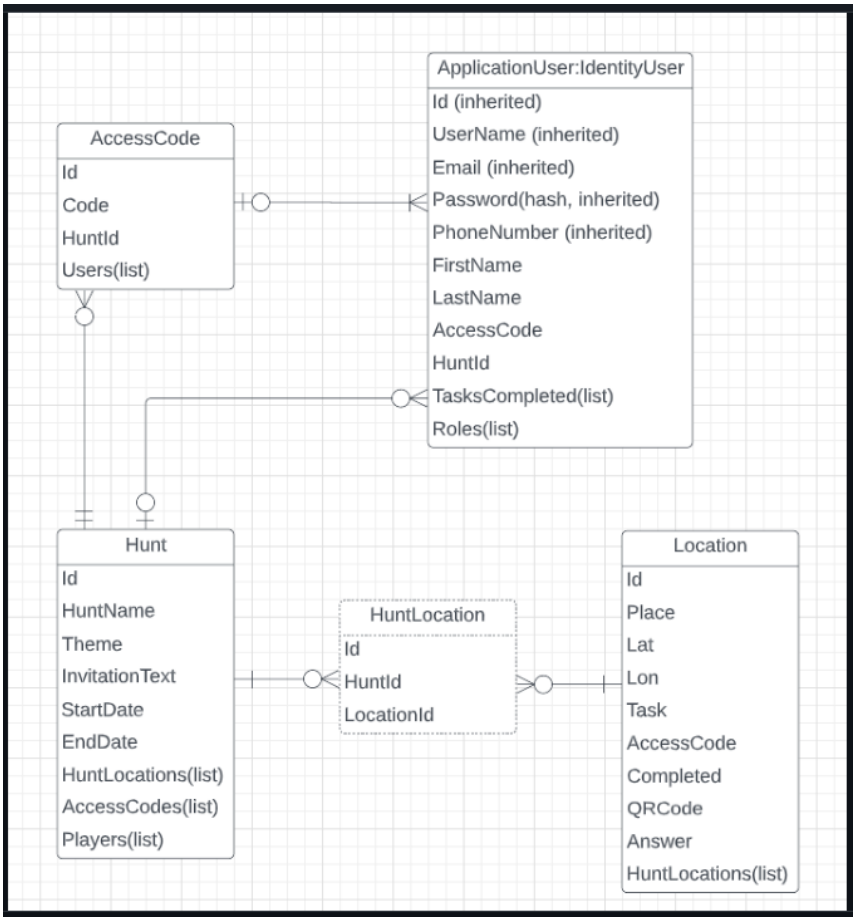
Problem:

It randomizes every time the page is refreshed

Completed tasks does not stay at the bottom anymore (Yu Lee, 04/24/2024)

The random order list can be completed via JavaScript (Abigayle Hays, 04/24/2024).

Scrum-61: Randomize Per Access Code



According to the database schema, the tasks are in the Location table. The location table is connected to the HuntLocation table. The HuntLocation Table is connected to the Hunt Table. The Hunt Table is connected to the AccessCode Table.

With this given database, it seems too complex to just link the access codes with the tasks. This is because you need to go through 2 other tables before you can interact with the tasks.

The bottom line is that I think you probably need to modify the database to get this to work. This may not be the case, but with the current information we have, that is the only way.

Other notes:

There is an access code field in the Location table, but since only one access code can be entered in the field, it makes it tough to try to randomize a few tasks for multiple access codes for a single hunt.

When the access code is created, it seems to be tied to a hunt in the database itself, rather than being tied to a state on the player's device. This database reliance leads to some quirks like being able to preserve the state of the hunt (in tasks completed versus incomplete) even after logging out and logging back in and being able to access the hunt even when it expires. I am aware that according to Mr. Kinser, the latter is due to not having a state engine or other check for whether the hunt is active or not, but it is still relevant to note here as well.