Team member contributions

Nolan coded the views, a portion of the controllers and a portion of the unit tests.
Anson coded the web portion of the game and a portion of the unit tests.
Aidan did the initial MVC conversion, the timer bonus feature, and a portion of the unit tests.
Hugo coded part of the logic, controllers, and a portion of the unit tests.

UML

Please see the attached png for the UML

Code structure rational

The structure of our code revolves around the MVC design pattern. The M stands for models, which represents the data models that are created to keep track of the player's progress. The V stands for views, which represents all the graphical portions of the game, anything that is related to visuals is placed into this category. Lastly, the C stands for controller, which contains all of the user input along with the game's logic. MVC has allowed us to create a game with a large amount of code, while still keeping the code clean and readable. This also allowed us to create unit tests at a more efficient pace, allowing more time for developing code. Both our views and controllers are broken up to four main categories: Welcome, Game, Game Over, and User Input. These four categories were made to split the stages of the game into four logicals areas. Therefore allowing us to structure our code in a simple and readable way.

Calculating Score

The score is calculated based on how long you took to complete the game.
The shorter the time the player takes, the higher the score they achieve.
The formula for how the score is calculated:
(max_time - time_passed)*100
Where max_time is the max time a player can take to complete the game (default 30 seconds), and time_passed is how long into the game the player took to complete the game.
If the player were to take 12 seconds, the formula would look like this:
(30-12)*100
Which would make the player's score 1800.

Web API explanation

We chose the JSON approach to storing data because it was the simplest format to store our data. This is because we stored our data in a dictionary format which can be easily transformed to JSON.

We decided that since we will not be storing a large amount of data the advanced query abilities of SQLite is not needed.

Bonus Features

We implemented the three bonus features suggested in the deliverables instructions and one additional bonus feature not suggested in the deliverables instructions.

1. **Timer:** The timer is displayed on the upper left corner of the main game view. This timer gives the player 30 seconds to collect the four items and exit the maze. If the timer reaches 0 seconds the game will end and the user is presented with the view to enter their name.
2. **Backpack:** The backpack is displayed on the top right corner of the main game view. This backpack will display the item that the player collects.
3. **Player name:** Once the game ends the user will be shown a new view and prompted to enter their name. Their name and corresponding score will be saved into a JSON file and can be viewed using the web API.
4. **Welcome and Game Over Screen:** When the game is first launched, the player is introduced with a welcome screen. The welcome screen will wait until the player has pressed SPACEBAR to start the game. Similarly, when the game is completed, the player will be introduced with a game over screen. The player can either press ESCAPE to exit the game or R to restart the game from the welcome screen.