**Requirements Documentation**

1. **Introduction:**
   1. Scope of the product:

**Agar.io (game) –** This is a game where you are a player in the center of the screen, and you eat things and grow bigger and bigger. You have others that you are competing with to be the biggest.

* What type of platform (Windows, Macintosh, UNIX, etc.) must the software work   
  with?
  + Windows, Macintosh (This will be for all projects, because we have members with just Mac devices and just windows.) (Game plays on windows only, but coded on both)
* Will the software function as a standalone application on a given computer, or will   
  it functions over a network connection?
  + Standalone.
* What other software, if any, must the software interact with? For example, you   
  might be building a subsystem component that will be integrated into a larger   
  system. In such a case, it’s important that you don’t duplicate functionality provided   
  by existing subsystems.
  + None, unless we find something to interact with. If we can get the network side working, it would use this software called Socket. It basically a low-level networking interface. It would be apart one of our last versions to work on.
* If you are building a game, how many players will be supported? Will there be a   
  computer player? If so, will it use artificial intelligence?
  + Single player. We will have computer players. It won’t be artificial intelligence.
* What programming language will be used for the project?
  + Python
* Will the software use a graphical interface or a command line interface?
  + It will be a graphical interface. (the menu)
  1. Definitions, acronyms, and abbreviations:
     1. Foodballs – They are the little balls in the game that do not move, but they do get eaten by the player and bots. They increase their size slightly.
     2. so\_random – it used in the bot movement function. It just a random integer that used to make bots random with movement direction.
     3. FPS – stands for frame per second
     4. Stand in – this term is use to call the circles that spawn on the menu screens.
     5. Bad\_sqrt – its bad square that are used to decrease one’s size to add a twist to the game.
  2. Reference:
     1. Agar.io - https://agar.io/ (it’s the original game we based our single player game on.)

1. **General description:**
   1. Product perspective:
      1. We are building this software to get experience on building a software with a group of people. We pick a game so that we can have fun and be creative with what we do. With this game, the base requirement for this game is not too hard to reach, but that gives us a chance to build on to it and improve it all together. This is useful for us so that we can grow and learn more on must function with others to get tasks done. This is something that will help us be prepared for the real world.
   2. Product functions:
      1. The function is to be a fun simply game to play. You get to move your character around with your mouse point and you can eat balls and bots. When you collide with a “food ball(s)” or bot(s), you can eat it. In the case for the bots, you must be bigger than the bots to eat them and vice versa. Also, there are bad squares that both bots and the player can eat and shrink in size. We have a menu setup that lets the user play again whenever they get a game over or to pause during the game (going back to main menu).
   3. User characteristics:
      1. This is for anyone that wants to enjoy a simple fun game where you keep growing or for ones that want to practice in a single player match of Agar.io. The practice would be for the original Agar.io game that is multiplayer. This could also be for those who want an offline experience.
   4. General constraints:
      1. It must be run in python, because it uses the pygame library to give the functions of the game.
      2. The game needs to run as exe with the pyinstaller we use.
      3. User’s Manual need to be in the same directory as the game application to be able to open when button is clicked.
      4. User’s manual is a html, which has a dependent folder it relies on to show gifs in it.
      5. Everything is in onefile (exe file) for the game. There is no easy way to fix the code after making it into a exe from a python file.
   5. Assumptions and dependencies:
      1. You will need browser to up the game instructions, or something that opens a “.html”.
      2. For the documentation you will need word or excel to access content.
      3. Python, pygame would need to be installed (but pyinstaller should take care of that.
      4. For the manual, the manual.html must be in the same folder as the game code/exe as well as have its dependencies folder within that same folder.
      5. Pyinstaller makes the game work on the operating system that it was setup on.
      6. Word is needed to open most of the documentation.
      7. Excel and project application (one that can open a .mpp file) is needed as well.
2. **Specific requirements:**

1.0.0 – The base game where the user can load up the software and play until their player character gets eaten by a bot.

1.1.0 – User gets a game over text in the center of the screen when game over requirements. (Gets replace in 2.1.0)

1.1.1 – Mouse position bug is fixed (added global variables)

1.2.0 – Map Bounds are fixed (made pos\_x and pos\_y equal the edge of border)

2.0.0 – The base game now has a menu with a play and quit button.

2.1.0 – The game over has two buttons: play again and quit. The play again button starts the game over again.

2.2.0 – added a button for the user’s manual to open in a browser.

2.2.1 – added a button for the user’s manual to open on the game over menu as well.

2.3.0 – cleaned up graphical features for the menu. Like colors of text, title for both main menu and game over menu.

2.3.1 – add “stand-in” circle that random spawn in the background of the menu. They look like the bot by stay still on screen of menu.

2.4.0 – added to key binds in-game. “m” key will pause game and go back to starting menu and player can hit start game to resume. “k” key will cause game over and try the player to game over menu, which is just a reset feature for the user.

2.4.1 – added text on the game left bottom corner to let player know “m” and “k” key does.

3.0.0 – The start of features we have added to the game.

3.1.0 – Added bad squares that decrease bots or player’s size by 1.

3.1.1 – changed size from 5 to 20. This makes it they are easier to see, and you can only have collision if one’s size is greater than or equal to 20.

3.2.0 – added an FPS message in game that lets the player know how well the game is run on their computer. FPS is set to 30 frame per second.

3.3.0 – added a version display in the menus for the user at the bottom left corner.