**Milestone 2:** Scope and project implementation plan document. Scope discussion indicating desired feature inclusion, this will form the minimum requirements for implementation. Initial project implementation plan covering overall project organization and code outline, includes description broken down into code modules, identification of data organization/structures and initial planded class needs. Class descriptions should map to the TYPE description referreed to in class. Deliverable: Free form discussion of what scope of effort entails, preferrably containing a list of required and stretch features; code organization documentation.

1. Desire feature inclusion
2. Project organization
3. Code outline (code modules)
4. Identify data organization/structures
5. Code organization documentation – pseudo code

### **Project Technique Inclusion Guidelines**

1. Basic object functionality as covered in class. I'll be looking to see consideration of best-practice construction.
2. Exception handling.
3. Basic Inheritance (the Geometry classes **as provided** do not count towards this, although you have permission to use them freely if so desired.)
4. GUI use including buttons and a text input functionality.
5. Functional programming, minimum of use of function name used as an argument.
6. Unit tests for non-GUI functions. This means functions that directly interact with GUI events do not need unit tests (they best use a technique called functional testing), functions that indirectly interact with GUI functionality do need unit tests.
7. Use of a Framework or Library not covered in class (Pygame counts towards this )

Pygame

1. Use of at least two data structures that you have been involved in coding. Assessed as part of code quality, call out best two if you have more. Data structures you coded in class work for this even if you don't add any additional functionality for this project. Use of Python or library-provided data structures will undoubtedly be necessary in your project, but do not count towards the "two" (neither does the singly LinkedList from the Linked List project as I provided it in class).

**Optional techniques/functionality**

1. Significant functional programming use, more in depth than just button functions/lambdas.
2. Grab bag topic techniques
3. Use of a Framework or Library not covered in class (Pygame counts towards this)
4. Some form of data export and/or persistence, such as JSON or pickling
5. Other clever and useful techniques not covered in class which you learned and used effectively.

**Deliverable:**

1. Project code directory (zip file)
2. Documentation directory
   1. The basic documentaion to allow a user to run the project on their machine and understand how to operateand basic controls, this includes things like dependencies or specific procedures. I am not looking for in-depth "game procedures or rules", just enough so that the GUI can provide its role of showing where to go once it presents itself. Also ensure this eliminates "Secret Sauce" problems, where it runs on your dev machine but nowhere else until the secret sauce is applied. Even if you do this in one file, place it in a dirctory titled Documentation. Remember to test on a Mac and ensure any requirements I will need to run on my machine are covered, it is enough to say "Pygame x.x.x installed" for that requirement unless you also require extensions.
   2. A cross reference of the required and optional techniques to filename/line number where that technique is implemented in your project. Just one or two examples each need to be cross references, not every appearance of a technique. I will not mine for your gold! This is your means of ensuring that your techniques will be uncredited because they were not obvious and/or found. This does not guarantee credit for the callount. As a recurring example I have previously had callouts for simple button calls as an attenpt for getting the substantial functional programming bonus, and those type of calloutys won't turn intopoints unless they honestly earned them. This will not cause predjudice or penalties in grading, the result just is what is earned in my judgement
3. **Reactions Directory** A paragraph (max two) from each team member summarizing insights gained and lessons learned from the project process. Not only concentrating on obvious successes, but also noting challenges which you overcame and resultingly will have an impact on how you attack problems in the future.

Data structures to implement:

1. Hash table for scoring (#misses, #hits, time alive, etc)
2. FIFO for special items collected