**Software Development Life Cycle**

Name: Shawn Swinsick

1. Requirements
   1. What does the application do?
      1. Allows the use of a CMV text file to be parsed into first name, last name and email
   2. Is it for-profit or non-profit
      1. Non-profit
   3. Is it open source or proprietary?
      1. Open source
   4. Requirements by Priority
      1. Level one: High Priority
         1. Read, Write, and create a file
            1. Functions include

Create

Update

Delete

Display all

* + 1. Level two: Medium Priority
    2. Level Three: Low Priority
    3. Level four: Extras
  1. USE cases(all of them)
     1. User scenarios

1. Specifications (Technical Lead)
   1. Design Structure
      1. What does it look like?
         1. The model is a person class that holds the given values, and the view and controller is another class.
      2. What are its boundaries?
   2. Coding Standards
      1. Any interaction with the console should be done though a private method, where any interaction with the file itself should be made public for use as an API
   3. Infrastructure
   4. Integration tests
2. Architecture (Foundation)
   1. The basic flow of the application
      1. There is a menu with the start of the application
      2. From there users can create, update, delete, or view all
      3. It is an endless circle till user enters exit or ends the program
3. Design (Appearance)
   1. User Interface (UI) Design
      1. The UI is a console window
   2. Model Design
      1. There is a person class that holds first name, last name, and email
         1. public method that replaces each comma in the values with spaces
      2. A program class that functions as the view and controller
         1. Holds a list of Person objects
         2. Each action has a private and public version
            1. The private method pulls information from the console window and with that information creates a person object and passes it to the public version
            2. The public version then adds, removes, or updates the list of people and writes it out to the file
         3. There are two constructors
            1. One is called from main, it is used to run the menu that uses the console version.
            2. The other one takes in a file name and is run as more of an API function
   3. UML Diagrams
4. Implementation (Code it)
   1. Code according to the steps above
5. Testing (Stress it)
   1. Take the ‘tree’ approach, how will the user progress through the UI? What snags will they hit?
      1. Test each function make sure the normal use will work
      2. Try using functions as they shouldn’t be
         1. Adding commas in values
         2. Deleting a person who isn’t there
         3. Updating a person who isn’t there
         4. Adding a person who is already there
         5. Updating to a person that is already there
6. Debugging
   1. Fix errors revealed through testing (Squish dem bugs)
   2. If errors were fixed go to step 6
      1. No errors were found
7. Deployment
   1. Making executable files or plug-ins from code
   2. Put your application on a server where the public can reach it (or private is so desired)
   3. Exporting your application to desired platform/other platforms
8. Maintenance
   1. Fix any bugs after deployment, and make sure the users are satisfied with their service. Repeat Step 1 if necessary
   2. Make changes based on customer feedback
   3. Refactor code to make it run faster, more effectively, etc.