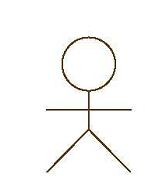
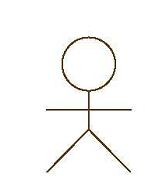
C. Chan, Gutierrez, Macam, M. Tan

**Use Case Diagrams++**

1.)

**User**

2.)

**User**

3.) When a user selects the Basic Mode of the program, the user is given the option to choose the mood of the song. After selecting the mood, the user clicks the randomize button and the notes are generated on the screen. The user then has to click on the play button to hear the music which was created and the stop button to stop the playing of the music.

However, when a user selects the Advanced Mode of the program, the user is given the option to choose in which scale to play the song and at what tempo. After selecting from the different options, the user then clicks the randomize button and like in the previous case, the notes are generated on the screen. The user then has to click on the play button to hear the music which was created and the stop button to stop the playing of the music.

4.) Scenario for Randomizing Notes on the Screen:

* User has not selected the mode, mood or scale and tempo of the song.

For Playing and Stopping the Song:

* User has not clicked randomize and there are no notes on the screen to play.

Choosing the Mood of the Song:

* none

Choosing the Scale of the Song:

* none

Choosing the Tempo of the Song:

* none



5.)

**Risk Analysis**

**Risks Involved (1 being the most critical):**

1. Being unable to finish
2. Faulty design of the software
3. Errors in the implementation
4. Lack of features
5. Miscalculation of formula
6. Loss of unsaved data or code
7. Interface is not user-friendly
8. Overestimation/Underestimation of Time
9. Inability to run in slower computers
10. Conflicts among group mates
11. Sleeping/laziness
12. Brownout while saving
13. Computer crash
14. Destroyed hard drive
15. Random school events
16. Crappy and buggy downloaded code
17. Natural calamities
18. No common time to meet up
19. Slow program
20. Lacking public approval
21. Inability to work together
22. Absences/tardiness in meetings
23. Not matching the mood to the melody
24. Not being able to find the "right" formula
25. Errors in the algorithm
26. Non-functioning GUI
27. Inaccurate documentation
28. Not feasible
29. Readings/distractions
30. Lacking requirements
31. A member meets an accident
32. Lack of equipment/facilities
33. Messy codes/Hard to read codes
34. Miscommunication among team members
35. Change of requirement
36. Inability to support other software
37. Incompatible with other OS
38. Procrastination of members
39. Overestimation/Underestimation of Cost
40. Debugging Errors
41. Code could not be understood by others
42. Software crash
43. Inaccuracy of the GUI
44. Problems linking the front-end to back-end
45. Error in programming logic
46. The program is difficult to operate
47. Inefficient number of LOCs
48. Inability to find the proper software
49. Problem with the GIT
50. Limited skills of the group members

**How would you attempt to mitigate each risk (Top 10 Risks)?**

1. Being unable to finish
   * Prioritize the tasks which are more important
2. Faulty design of the software
   * Make the code more flexible
3. Errors in the implementation
   * Massive Debugging Session
4. Lack of features
   * Make Code Easily Extensible
5. Miscalculation of formula
   * Try to double check the calculation of the formula
6. Loss of unsaved data or code
   * Try to save code or work in intervals
7. Interface is not user-friendly
   * Let kids or non-groupmates test and modify accordingly
8. Overestimation/Underestimation of Time
   * Hard to Avoid. Give allowance to the estimated time.
9. Inability to run in slower computers
   * Not our problem. We will try to make the program more efficient.
10. Conflicts among group mates
    * Everyone must learn anger management. Settle conflicts fairly and calmly.

**Software Life Cycle Model**

Rapid Prototyping Model

We believe that the Rapid Prototyping Model is the best software life cycle we can use in this project because this model ensures that there is a clear understanding and communication between the users of the program and the developers. This life cycle model focuses on the GUI of the program so that the users may be able to comment and insights may be given to the development team. These comments may then be incorporated into the program by the next deliverable. Also, through this model, the missing features may be added later on depending on the time available for the rest of the project.