Team Members: Aaron Geesink, Garrett Geesink, Milo Fisher, Noah Agudelo

The rules of Steve are as follows:

1. Be a team player.
2. All Team members will meet on Tuesdays and Thursdays from 2-3 pm in the Computer Lab if possible.
3. Team members will keep other members up-to-date on discord as to what they are working on and meet up in voice chat for help
4. Get your code done: don't be the guy that comes to class with his part not done
5. Be active on discord
6. Upload Do NOT upload any changes to the project to the master branch in github before it is reviewed by the product owner. Upload it as a separate branch for it to be reviewed by Aaron, and Aaron will merge it into master
7. Don't merge or upload on GitHub without knowing how to use Git. Ask Aaron if you need help
8. If you are having trouble implementing code, ask another group member for advice and assistance. Speak up about coding issues before it becomes an issue.
9. Remember to S.T.E.V.E:
   S. Strive to make the best code possible
   T. Try your hardest
   E. Elevate your team's work
   V. Value your contributions to the team
   E. Evaluate your code before completion

The Coding Standards of Steve are as follows:

1. Tools
   a. The project will be programmed using C++ in QT
   b. The UI will be designed as a QT widget application
   c. The UML Diagrams will be made in Microsoft Visio
   d. The State Machine Diagrams will be made using Microsoft Visio
   e. The Use Case Diagrams will be made using Microsoft Visio
   f. The class and function documentation will be made using Doxygen
   g. All scrum artifacts will be made using google docs and google sheets
   h. The project will be uploaded and saved in a GitHub repository
2. Procedure
   a. Before programming a class, the UML diagram for that class must be completed, reviewed, and accepted by **ALL** team members.
3. Coding Style
   a. Don't try to do everything in one huge function. Break a problem into many smaller functions that call each other.
   b. Don't have an entire function for each small feature that is only used once that could easily be combined into one slightly larger function to increase efficiency.
4. Commenting

a. At least one comment must exist per chunk of code describing BRIEFLY what said chunk of code does
b. At least one comment for every function
c. No commenting more than this unless completely necessary
d. Variable and function names should be self explanatory to help reduce the need for excessive commenting