**Software Implementation and Testing Document**

**For**

**Group 19**

Version 1.0

**Authors**:

Dylan McClure

Julian Sweatt

Caleb Smith

Lucas Zavalía

Michael Heron

# Programming Languages

Python is used for everything. Since none of us had much game design experience, it seemed like a good idea to use a language like python to create a plethora of games. Using pygame, we will also be able to quickly build games – something that might take longer in other languages. Also, because pygame is easy to set up in many environments, creating cross-platform games is easy, and the app can be easily deployed on a raspberry pi similar to a retropi.

# Platforms, APIs, Databases, and other technologies used

Python; pygame; pytest; mypy

# Execution-based Functional Testing

Each group member has individually tested their game for basic functionality. Other members have looked over that testing and no surface issues have been found thus far.

During code reviews, the reviewers are expected to run the app under that branch to ensure everything is working as expected and described by the author. This also ensures continuous cross-platform support, as we all use different operating systems.

To functionally test the card games I developed a fairly basic testing script that allowed me to quickly generate hands to test my functions that would check hand values for the specific games. After the values were determined I was also abled to easily implement the functions that checked for end game conditions and card interactions. The majority of development for both poker and blackjack without introducing a GUI.

# Execution-based Non-Functional Testing

CPU play has not yet been implemented in the games, therefore they were not tested. A lack of dependency on the internet was tested by ensuring no network calls were made. Accurate gameplay was tested by playing the games manually to ensure accurate and expected gameplay.

Pytest and mypy were used for some games for unit testing and testing static types. Maybe we should standardize this.

# Non-Execution-based Testing

Code was inspected for style, efficiency, and efficacy by placing code in a individual feature branches for review. Code would not be merged into master without being available for review by other developers first.

Manditory code reviews ensure appropriate style, cohesive design style, and use of other’s libraries in an intended and agreed-upon way.