**Software Requirements and Design Document**

**For**

**Group 19**

Version 1.0

**Authors**:

Dylan McClure

Julian Sweatt

Caleb Smith

Lucas Zavalía

Michael Heron

# Overview

Plethora Py is a collections of board and internet games that we all remember playing as kids. Hopefully, it evokes some nostalgia from the user after playing his/her favorite childhood games such as, Tic-Tac-Toe, Connect4, and Chess, Bomberman, Blackjack and more. These games make it easy to entertain oneself, as one doesn't need to go to the store and buy a game, and they are free.

Plethora Py can potentially be used as a game development framework, as well, since it provides a UI and access to UI elements and a stateful API. As such, anyone wanting to develop a game can use the API to do so. Although pygame already abstracts things a great deal and makes it fairly easy, Plethora Py makes it just a little easier.

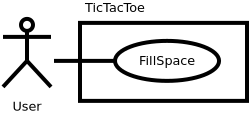
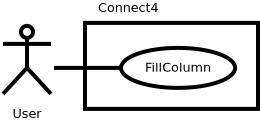
# Functional Requirements

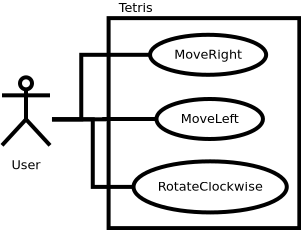
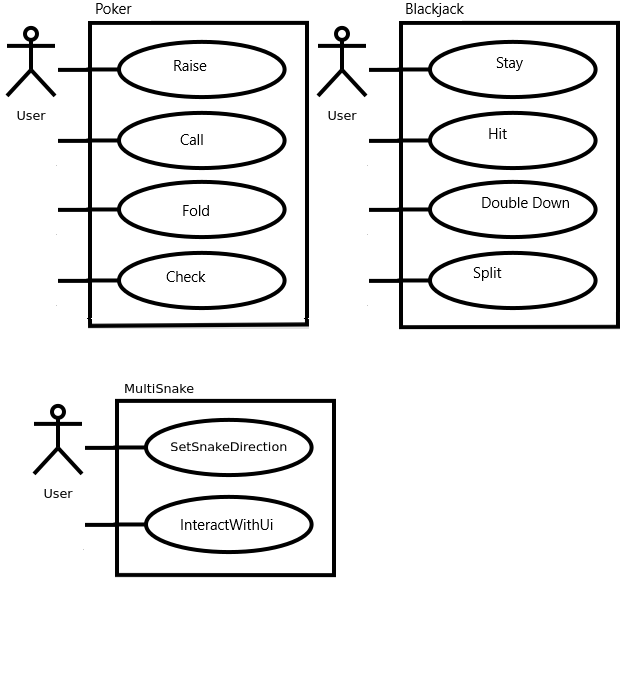
* Overall:
  1. 10-12 working games - high priority
     + Each game must be able to handle user input in one or more forms (mouse, keyboard, controller, etc)
     + Each game should not crash and should remain faithful to the original game mechanics unless a “twist”
     + Each game is imported with “importlib” and each should be loaded in under two seconds
     + Each game must run within the frame provided by plethoraAPI and when the user tries to close the window or presses the back button (unimplemented), the game should return back to the main menu in under 1 second
     + Status: We are on track for this. We should have at least 10 functioning and games by the end of the increment.
  2. Menu to select desired game - high priority
     + The plethoraAPI find games automatically in some specified location(s) and load them dynamically
     + Then there must be some way of viewing and launching them; this should be implemented via a menu (now buttons are used since they’re easier to implement)
     + Status: The games are found automatically in directories inside “arcade/games/” and loaded automatically. However, we still need to implement UIMenu and use that instead of multiple UIButtons.
  3. User login - medium priority
     + A user must create a username when none is provided
     + Upon re-starting the game, the username should be remembered
     + The username should be modifiable so a user can change the name
     + Status: We still need to allow a username to be entered and to store this and allow user to change this
  4. A system that keeps track of points/high scores for each game.
     + Status: Once the user login is implemented, this will be trivial
* Lucas:
  1. Tetris
     + Needs to handle edge detection to the sides
     + Needs to be handle certain edge cases when the jump function is called
     + There needs to be a functions to delete full rows
     + There needs to be a “Game Over” Screen
  2. 2048
     + There needs to a way to generate new pieces each turn
     + There need’s to be text displayed on each tile with the number
     + There needs to be a way to move the board around (a recursice algorithm is probably best for this)
* Michael:
  1. MultiSnake
     + Needs to support at least 3 players
     + Need to have a menu to select the number of players
     + Players need to be able to distinguish between each other’s snake
     + Needs to automatically start a new game when one player is left
     + Needs to have an end screen that shows player rankings after a certain number of games
  2. Blackjack
     + Clearly display card value and suit
     + Simple UI with only as many options as you need
     + Accurately implement the rules of blackjack
     + Consistently handle specific niche card interactions such as ace hands and double blackjack
     + Allow players to place bets and track their money over time
  3. Poker
     + Clearly display card value and suit
     + Simple UI with only as many options as you need
     + Accurately implement the rules of poker
     + Consistently handle specific niche card interactions such as full houses, and straight flushes
     + Implement betting and track players money over time
     + Be able to record your high score in a leaderboard
     + Be able to input a custom name for the leaderboard
* Dylan:
  1. Chess
     + chess should load under 1 second
     + on start, a board is displayed
     + user can drag a piece or select a piece by clicking or pressing space bar (after navigating with arrow keys) and then move the piece to another square by selecting it similarly
     + move validation should be accurate
     + chess engine should determine possible moves
     + different modes should be offered (CPU, 1v1, etc)
  2. plethoraAPI / UI
     + UI should load in under 1 second initially
       - currently, it loads all games at startup, which is bad
     + UI should have a title
     + UI should allow a user to log in
     + UI must have a menu showing all of the games
     + When a game is clicked, the game must launch unless there is an error
     + UI must receive events from pygame and shuttle them to a game if a game is in progress in under 200 ms
     + API should handle dialog boxes (which block event and render calls temporarily) and buttons/menus (and their callbacks) and other UI elements
* Caleb:
  1. TicTacToe
     + game launches in under 500 ms
     + on launch, game displays grid
     + game listens for mouse clicks and can put a piece at the square a user clicks
     + when one player wins or the game is a tie, show a menu to exit or play again
  2. Checkers
     + game launches under 500 ms
     + on launch game displays the board
     + user can click squares to select pieces and move a peice to another square
* Julian:
  1. Connect4
     + User should be able to play a 1v1 game of classic “connect 4”
     + Pieces should be color coded
     + Games should be playable again and again until someone quits
     + Score should be kept accross each round
  2. Bomberman
     + User should be able to move accross the board as a player sprite
     + Maps should be generated differently each time the game is played
     + Users should be able to interact with the map with their player sprite though destruction of obstacles with bombs
     + Bombs are capable of destroying certain components of the map, or the players (within range)

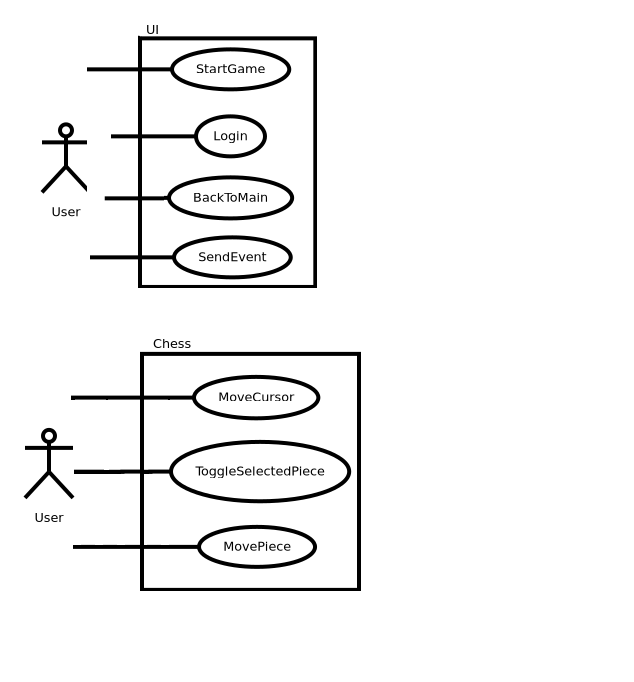
# Non-functional Requirements

1. Interactive
   * All games should be have coponents that the user can interact with to get a well defined goal.
2. Modular and Reusable
   * Because the plethoraAPI and UI simplify things a little and provide UI elements, anyone can reuse this framework to develop a game or even a similar UI.
3. User Friendly
   * All our games should have clear UIs and controls so that anyone will be able to use our games with the right controllers
4. Independent
   * The games should not require internet access, but they can use it optionally. All our games should be able to be played with generic controllers.
5. Accurate
   * The games should adhere to the original games or can propose new mechanics but should try to be as faithful to the mechanics as possible
6. Configurable
   * user should be able to create username and change it
   * user settings

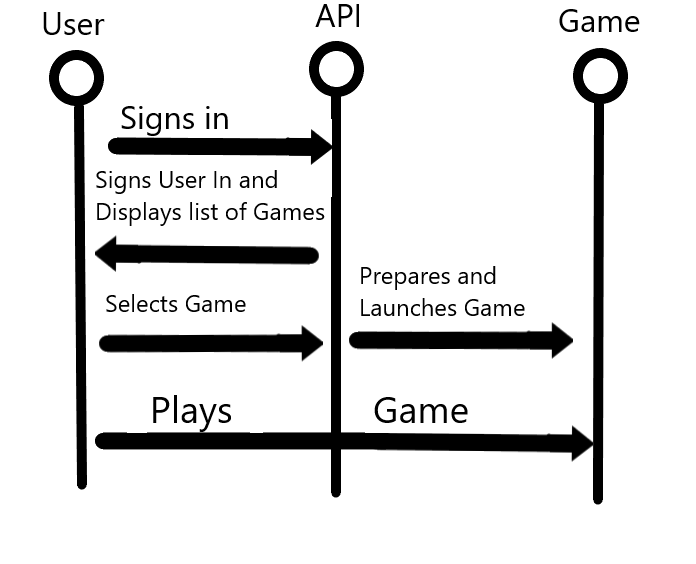
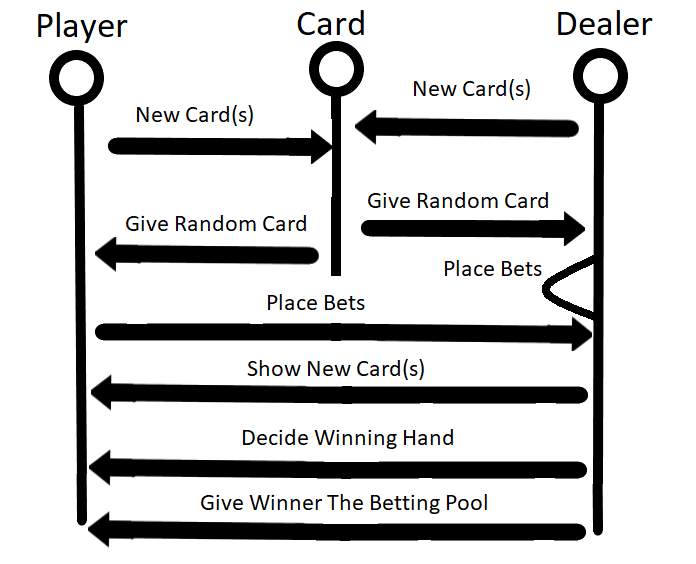
# Use Case Diagrams







# Class Diagram and/or Sequence Diagrams

**

# Operating Environment

We are developing this project to run on a Raspberry Pi that will be running the Raspbian OS. Though we are designing Plythera Pi to run on a Raspberry Pi 3 Model A+, any of the available Raspberry Pis should be able to run our application just fine.

Plythera Pi is running on a Python virtual environment. Therefore the game platform could be extended to any other environment capable of supporting python3.

# Assumptions and Dependencies

For this project we are assuming that everyone who wants to play games on our application has access to at least a mouse and keyboard, but for some games actual USB controllers will also be supported.

python3 and pygame are the only two dependencies that we have so far.