Contributors: Zachary Bickel, Carson Abbot, Jesse Blahnik, Samuel Aleks
Assignment: Group Project, Part 5: System Design
Start-up Cases:

- **Use Case Name:** LaunchGame
  Participating Actor(s): Player
  Description: The player would use this case to activate the game.
  Flow of Events:
    1. The player clicks on the launch icon.
    2. Run.exe happens
    3. Game opens
    4. The player enters their username and password into text fields called up by queryUsNmPswd() to login to their account
    5. Click enter button/hit enter key to submit.
    6. System executes checkInputData() to find the relevant user
    7. System returns okay if user is found with userFound() method
    8. Relevant data is sent to the user via getCloudData()
    9. Game goes to main menu via the mainMenu() method

  Entry Conditions: The player has downloaded/installed the game.
  Exit Conditions: The player can now perform the activities provided in the main menu.
  Alternative Flow:
    1. The player clicks on the launch icon.
    2. Run.exe happens
    3. Game opens
    4. Player doesn't have an account to give to queryUsNmPswd().
    5. Player clicks on "register" to create an account.
    6. registerRedirect() runs

  Exceptions Flow:
    1. The player enters his/her username and password to login to his/her account.
    2. Something was typed wrong, so a message is displayed to let the player know what the issue is and in which textbox it occurred.
    3. The player is asked to check the spelling and either correct it or re-type the information before attempting to submit it again.

    4. The player clicks on the launch icon.
    5. Run.exe happens
    6. Game opens
    7. The player enters their username and password into text fields called up by queryUsNmPswd() to login to their account
    8. Click enter button/hit enter key to submit.
    9. System executes checkInputData() to find the relevant user
    10. System returns okay if user is found with userFound() method
    11. Relevant data is sent to the user via getCloudData()
    12. An error occurs somewhere
    13. Crash report is sent to player and can optionally be sent to Admin.

- **Use Case Name:** LaunchWebServer
  Participating Actor(s): Admin
  Description: The admin would use this case to activate the web server.

Flow of Events:

1. The admin clicks on the launch icon.
2. Run.exe happens
3. Web Server status terminal opens
4. Initialization() runs
5. Web Server completes setup
6. Web Server enters standby for user activity

Entry Conditions: Admin wants to turn on the web server
Exit Conditions: The web server finishes start-up procedures
Alternative Flow:
Exceptions Flow:

1. The admin clicks on the launch icon.
2. Run.exe happens
3. Web Server status terminal opens
4. Initialization() runs
5. Web Server runs into an error of some kind during start-up
6. Web Server ceases start-up and provides a crash report on the terminal

- **Use Case Name:** LaunchWebGame
  Participating Actor(s): Player
  Description: The player would use this case to launch the game on the web page.
  Flow of Events:
  1. The player clicks onto the game page on the site.
  2. Run.exe happens
  3. Initialization() runs
  4. The player enters their username and password into text fields called up by queryUsNmPswd() to login to their account
  5. Web Server finds no prior save data with saveDataExists().
  6. initialization() runs.
  7. launchDemo() runs and the in-browser demo of the game launches.

  Entry Conditions: Player accesses the demo area of the webserver with an account.
  Exit Conditions: The web server finishes start-up of the demo
  Alternative Flow:
  - The user has no account

  1. The player clicks onto the game page on the site.
  2. Run.exe happens
  3. Initialization() runs
  4. The player enters their username and password into text fields called up by queryUsNmPswd() to login to their account
  5. The player doesn't have a valid account
  6. The player clicks on "register" to create an account

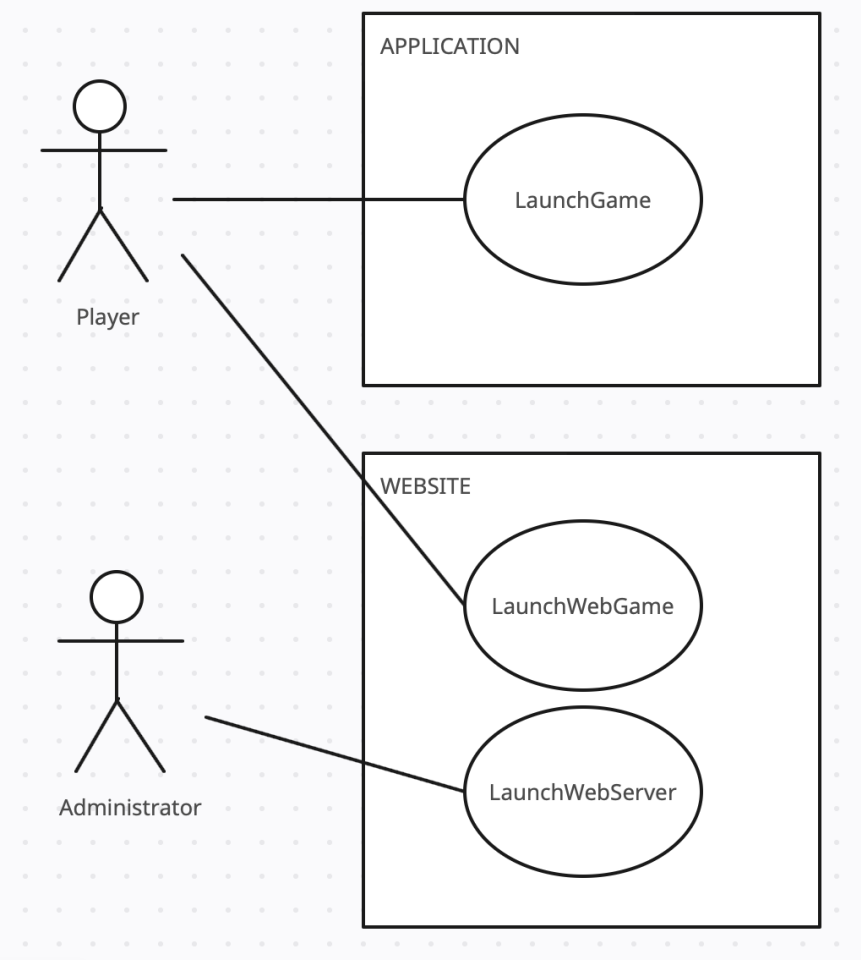
  - Account has prior save data

  1. The player clicks onto the game page on the site.
  2. Run.exe happens
  3. Initialization() runs
  4. The player enters their username and password into text fields called up by queryUsNmPswd() to login to their account
  5. Prior save data is found with saveDataExists().

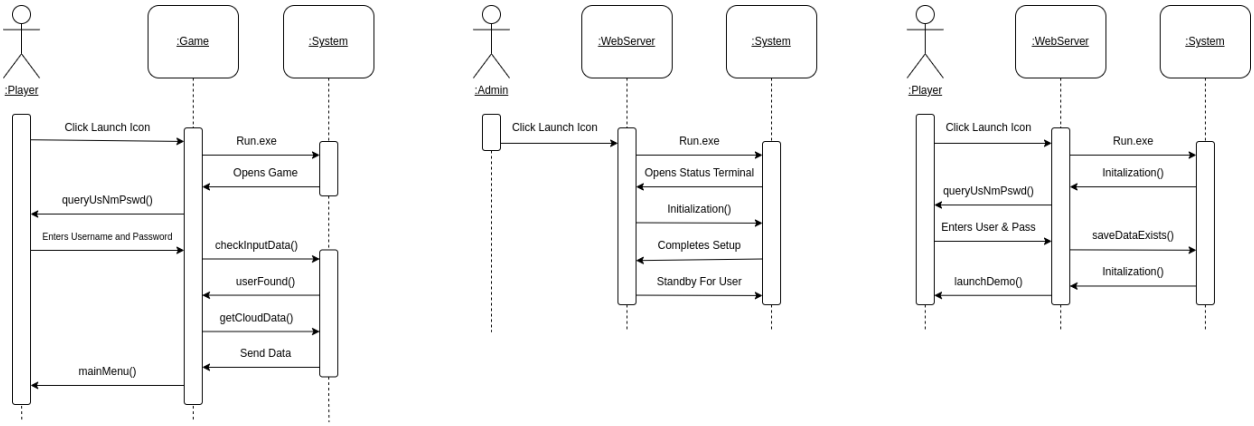
6. querryOverwrite() runs and shows a window asking if the player wants to overwrite their prior save data.
7. initialization() runs.
8. launchDemo() runs and the in-browser demo of the game launches.

Exceptions Flow:

1. The player clicks onto the game page on the site.
2. Run.exe happens
3. Initialization() runs
4. The player enters their username and password into text fields called up by queryUsNmPswd() to login to their account
5. Web Server finds no prior save data with saveDataExists().
6. initialization() runs.
7. initialization() runs into an error during startup.
8. initialization() stops and sends a crash report to server.



(Use case diagram)



(Sequence diagrams left-to-right: LaunchGame, LaunchWebServer, LaunchWebGame)

System Architecture Description: For the architecture of our system, we based it on a combination of the client/server and peer-to-peer styles. We used the client/server style for the website that provides the game, as well as the desktop application for playing it. And, we used the peer-to-peer style for the multiplayer aspects of the game.

SpiritForgeWebServer

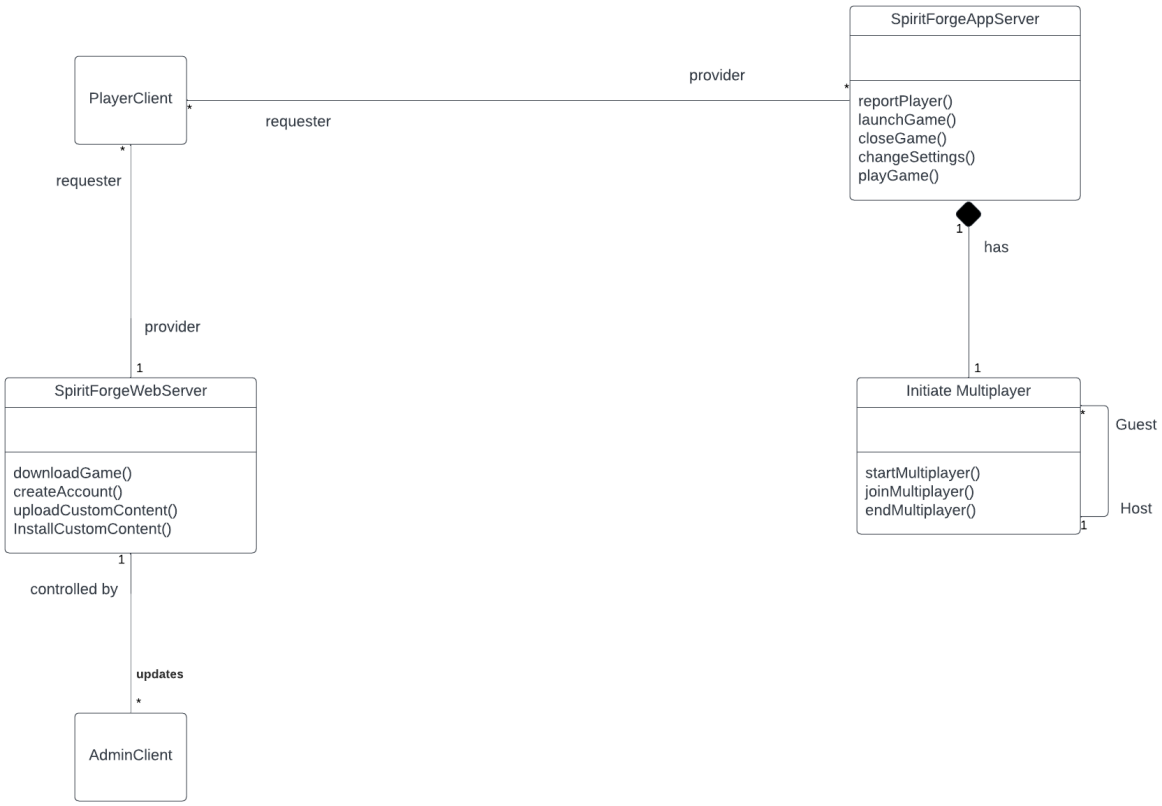
This is the centralized web server that clients make requests to (by means of the website) in order to access the web services it provides, such as: downloading the game, creating user accounts, and uploading/installing custom content for playing the game. The web server is controlled by 1 or more administrators who periodically update the website.

SpiritForgeAppServer

The application server can be any of a number of servers that provides services to clients who make requests to it after installing the game on their computers. An application server is accessed while a client runs the desktop application, allowing him or her to perform such operations as: reporting other players, starting/playing/closing the game, and changing its settings.

Initiate Multiplayer

Initiate Multiplayer makes use of the peer-to-peer network involved in the system. Each computer within the network has the ability to act as a host to others (i.e., guests), thereby directly sharing the game's resources between each peer. Once connected, peers can communicate and interact with each other, as well as form/join teams. Otherwise, a player can disconnect from others by ending the multiplayer functionality.



(System Architecture Diagram)