USE CASE: UPLOADING A BOT

**BASIC COURSE:**

The user clicks on the “upload a bot” link. The client opens a file navigation menu. The user then selects their player bot from a file navigation menu. The client then prompts the user to enter a name that is associated with their player bot. The user enters a name and clicks the “submit” button. The client checks that the user input a java file, then uploads the bot to the server. The server validates the safety of the bot, then sends a message to the client indicating that the bot was successfully uploaded. The client then displays this message to the user.

**ALTERNATE COURSES:**

**Bot safety could not be verified:** The server rejects the player bot and sends a message to the client indicating the problem(s) with the bot. The client then displays a message informing the user of the problems.

**Player bot is not a java file:** The client displays an error message indicating that the file was not a valid java file.

**File does not exist:** The client displays an error message informing the user that the file does not exist.

USE CASE: USER INPUT IN TESTING MODE

**BASIC COURSE:**

The client displays an input field for a command, a “submit” button, an undo button. The user enters a command and clicks submit. The client then sends the user’s input to the server. The server validates the user’s input, evaluates it, and returns the resulting game data (game states, animation commands, and notifications). The server then sends this data back to the client. The client passes the data to the game display module, which displays it in Game Display Window. The server and client then begin the next turn.

**ALTERNATE COURSES:**

**Current turn is the first turn of the game:** The undo button is not displayed.

**Current turn is a previous turn which has been undone:** The redo button is displayed.

**The game has ended:** The client will refuse any further input from the user, except for undo.

USE CASE: CLIENT-SIDE BOT INPUT IN TESTING MODE

**BASIC COURSE:**

The user selects “Client-Side Testing Mode” from the Game Display Module’s menu. The client displays a “Run Bot” button, a “Choose Different Bot” button, an undo button. The user clicks “Run Bot”. The client then runs the bot and sends the resulting input to the server. The server validates the input, evaluates it, and returns the resulting game data (game states, animation commands, and notifications). The server then sends this data back to the client. The client passes the data to the game display module, which displays it in graphical form. The server and client then begin the next turn.

**ALTERNATE COURSES:**

**Current turn is the first turn of the game:** The undo button is not displayed.

**Current turn is a previous turn which has been undone:** The redo button is displayed.

**The game has ended:** The client will refuse any further input from the user, except for undo.

**The user selects “Choose Different Bot”:** The client will prompt the user for the filepath of their new bot.

**The bot is still running after the maximum bot runtime has elapsed:** The client will display the error message “The bot failed to produce output within (x) milliseconds” to the user.

USE CASE: SERVER-SIDE BOT INPUT IN TESTING MODE

**BASIC COURSE:**

The user clicks on “Server-Side Bot Testing Mode” from the Game Display Module’s menu. The client displays a “Run Bot” button and an undo button. The user clicks “Run Bot”. The server runs the bot to produce input, then validates the input, evaluates it, and returns the resulting game data (game states, animation commands, and notifications). The server then sends this data back to the client. The client passes the data to the game display module, which displays it in graphical form. The server and client then begin the next turn.

**ALTERNATE COURSES:**

**Current turn is the first turn of the game:** The undo button is not displayed.

**Current turn is a previous turn which has been undone:** The redo button is displayed.

**The game has ended:** The client will refuse any further input from the user, except for undo.

**The bot is still running after the maximum bot runtime has elapsed:** The server will produce an error notification and end the game. The client will display the error message “The bot failed to produce output within (x) milliseconds” to the user.

Use Case: Challenge Mode

BASIC COURSE:

The client displays the menu of game modes and options to the user through the Game Display Module. The user selects Challenge mode to play. The client will display a list of challenges and bots for the user to select. The user will select the challenge and the player bot they wish to use. The client will send the selected bot and challenge to the server.

The server will run the challenge mode game with the selected bot and challenge by passing commands produced by the bots to the game evaluation module. The game evaluation module will produce a new game state and return it to the server. The server will send the game data (states, commands, and errors) to the client. The client will pass game states to the Game Display Module. The Game Display Module will create the display using these game states and display to the user using the Game Display Window. The client will pass a confirmation message to the server to alert the server that it can send another game state. This process will repeat until the game has been completed.

ALTERNATE COURSES:

**Bot input produces error:** The server will treat the command as “idle” and pass this command to the game evaluation engine.

**Turn limit exceeded:** The server will automatically terminate any games lasting a large preset number of terms (currently 500) as a precaution to an endless stalemate.

**Bot takes too long to produce output:** The server will terminate a bot that takes too long to produce a command and produce an error stating “Server-side bot did not produce command fast enough”.

Use Case: Replay Mode

BASIC COURSE:

The client displays the menu of game modes and options to the user through the Game Display Module. The user clicks on Replay Mode. The client displays a list of games to view. The user selects from a list of games that are able to be viewed and sends this to the client. The client sends a message to the server requesting the game data from the selected game. The server will send the client the game data to be viewed (states and commands). The client will accept this data and use the Game Display Module to create the display. The Game Display Module will send display to the user through the Game Display Window. The user will have the option to rewind, fast-forward, pause, and play the game with buttons provided.

ALTERNATE COURSES:

**Live game in replay mode:** The user will be unable to rewind, fast-forward, pause, and play if the game is in the process of being played.