The project is a turn-based educational game that pits java bots (uploaded by the users) against each other or single-bot challenges. Our part of the project deals with bot verification, testing mode, graphical display and playback mode.

The project has three major game modes: testing mode (in which the user can test their bots against other bots, user input, or single-player challenges), challenge mode (in which the server will run a game using only an uploaded bot or bots), and playback mode (in which the client will replay a finished testing mode or challenge mode game using data retrieved from the server).

GENERAL:

1. The Game Display Module (part of the client) must be able to accept game states and commands from the server, and use them to draw game states and animations with sprites, including the map and characters. Additionally, the module must be able to display errors.

2. The server must be able to accept uploaded java programs (referred to as *bots*) from users, verify that they are not potentially malicious, and save them to its bot database.

3. The server must be able to save all game states, commands, and errors to its game state database.

4. At the beginning of a replay or testing mode game, the server must be able to pass the initial game state to the client.

TESTING MODE:

5. At the beginning of any character’s turn, the system must be able to handle input from them as appropriate (based on their given input type).

a. If the character is controlled by user input, the client must be able to prompt the user for their input and then send it to the server, then end the character’s turn and wait for the server’s response.

b. If the character is controlled by client-side bot, the client must be able to prompt the user to modify their bot, if desired. The client must also allow for the user to run the bot once, then send this output to the server, then end the character’s turn and wait for the server’s response.

i. If the client-side bot takes too long to produce output, the client must be able to produce an error to notify the user that the bot takes too long to run.

c. If the character is controlled by server-side bot, the client must be able to prompt the user to proceed to the next turn. If the user accepts, the client must send a message to the server. The server must then run the bot and accept its input.

6. When prompting the user for input at the beginning of a character’s turn, the client must also allow for the user to undo or redo previous moves.

7. After accepting input from the client or from a server-side bot, the server must be able to pass it to the Game Evaluation Engine, which will return either a new game state or an error.

a. If the server received an error, it must pass that error to the client, which must display it. If the error was from player input or a client-side bot, the client must then prompt the user to retry their input. Otherwise, the server must proceed to the next turn.

b. If the server received a new game state, it must pass that game state to the client, which must display it. The system must then begin the next character’s turn, or terminate the game if needed.

8. After a testing mode game has run to completion, the client must prompt the user to view a replay of it in playback mode.

CHALLENGE MODE:

9. The server must be able to automatically run a challenge mode game to completion using a given map and uploaded bots.

10. The server must be able to pass a command produced by running an uploaded bot to the game evaluation module, which will return a new game state.

a. If a bot’s input produces an error, the server must treat its command as an “idle” command and pass this command to the game evaluation engine.

11. Once the game ends, the server must be able to send the client(s) the game data (states, commands, and errors). The client must then enter replay mode and allow the user to watch the game.

REPLAY MODE:

12. The server must be able to send the client a sequence of game states and commands for a requested completed game (either testing or challenge mode), as well as any errors produced.

13. The client must be able to accept this data and use the Game Display Module to replay the game.

14. Replay mode must also allow the user to play or rewind the game one turn at a time.

ADDITIONAL BOT VALIDATION:

14. If a testing mode or challenge mode game exceeds a large number of turns (ie. over 500), the server must be able to automatically terminate it (as a precaution against endless stalemates).

15. If a server-side bot takes too long to produce output (ie. over 500 milliseconds), the server must be able to terminate it and produce an error.

**Functional Requirements**

-The client shall accept as input a game state (as well as commands from server-side hosted bots) from the server and display it using the game display module.

-The client shall provide the ability to replay a finished game using a sequence of commands and game states passed to it by the server.

-The game display module shall display the current game state with sprites, using data passed to it from the client.

-The client shall accept as input commands from the users and/or client-side java bots and pass it to the server.

-The client shall accept as input a java bot from the user and will upload it to the server to produce input for testing arena games.

-The server shall accept a command from the client and pass it to the game evaluation engine. If the command is accepted, the server shall receive the next game state from the engine and pass it to the client(s). If not, the client responsible for the invalid data will notify the user of the problem.

-If the client requests a replay of a game, the server will pass it all game states and commands recorded for the requested game.

-The server shall store the current and previous states of all live and completed games in the game state database.

-The server shall accept uploaded java bots and ensure that they are not malicious. If the uploaded bots are unsafe, the system will send a message to the client to inform the user of the error.

-If a game being played involves one or more uploaded bots, the server shall run those bots to produce the needed commands.

**Glossary**

**System:** All components of the project, including the client and server-side modules.

**Client:** The program used by the end-user.

**Server:** The software that runs on the

**Game State Database**: The database that stores the states of each game for use in game evaluation and testing.

**Game Display Module:** Client-side Phaser program which takes game input from the server and displays it in graphical form.

**Character:** The in-game representation of a player or bot.

**Bot:** A java program that takes a game state and returns an action.

**Public Bot:** A default bot provided by the server, used for testing.

**Player Bot:** A player-created bot.

**Command:** The move a character makes on their turn, which changes the game state.

**Testing Arena:** Game mode which allows the user to test bots against other bots (hosted server-side) or player input.

**Challenge:** An online game mode that accepts only bot input.

**Turn:** A single game state update**.**

**Game:** An instance of testing or challenge mode.

**Live Game:** A game that is still currently being played on the server.

**Mode:** Testing, challenge, or playback.

**User:** Human who plays the game.

**Testing Mode:** A game mode which allows the user to test bots in the testing arena.

**Playback Mode:** A game mode which allows the user to view a game that has already been played.

**Play:** Start or resume playback of a game.

**Pause:** Stops playback of the current game, with the ability to resume later.

**Rewind:** Reverts the game state back in the visualization.

**Fast-Forward:** Speeds up the display of the game visualization.