TEAM MEMBERS: Karl Lawson, Sneha Santani, Yerania Hernandez
TEAM NUMBER: 21

Product Backlog

The following are the tasks that need to be accomplished by the team in order to provide fully functional game of Kalah. There is a possibility that the product backlog is updated when we think of more features to add to our implementation or when the client server architecture implementation is made more clear.

1. User launches the program and sees a display message
2. User launches the program and can now also see a button that links to a tutorial/ Help button
3. User decides if to play with another human player or against the computer
4. User chooses his name with a text box input
5. User redirected to a new screen which starts the game and displays the current board
6. User enters a move by clicking on a house
7. Computer checks if time limit is up
8. Computer checks if the given move is valid or not
9. If valid move, it is executed
10. User is allowed 3 invalid moves
11. User sees the new state of the board
12. The program checks if anybody has won
13. The program checks if player or computer should play again.
14. Player/Computer plays
15. If Computer, then Calls the utility function
16. If Player again lets the user enter new move, Calls the board evaluation function.
17. Utility Evaluation function has the numerical value to the state assign
18. The program has a basic min-max tree set up that checks all the valid moves
19. Provide graphics for game board
20. Work on visualization of home screen
21. Evaluate and test the utility evaluation function with a working min-max
22. Computer can randomly select a valid move
23. Computer looks ahead one move in tree
24. The program checks if the leaf is at a final state
25. If at a final state, then assign the max/min possible utility value
26. If the leaf is not final state, then evaluate the state of the board to select the best possible move (using recursion/0)
27. Complete the implementation of the minimax tree
28. Integrated the iterative deepening implementation with a basic alpha pruning function
29. Integrated the iterative deepening implementation with a basic beta pruning function
30. Evaluate the same move from all the chance nodes
31. If Max node take the max value if Min node take the min value
32. Revaluates the utility using the now found alpha and beta and min/max value
33. Game checks if the max number of moves has been exceeded
34. If not, then the player/computer plays again (recursion/loop)
35. Game checks if the total time allocated for the game has been finished
36. Board evaluation function checks if a winner has been figured out
37. Set up the protocol for client server architecture

38. Set up the basic configuration for client server
39. Set up the client connection
40. Integrate the server
41. Integrate the client server architecture with the game
42. Integrate the remote client services with the game developed
43. Implement number of holes in board game based on server configuration
44. Implement number of seeds in each hole based on server configuration
45. Implement random distribution of seeds based on server configuration
46. Implement same arrangement of starting seeds on each side
47. User gets to choose if to do pie rule
48. If user chooses to do pie rule, implementation of board with switching positions
49. If user chooses not to do pie rule, implementation of not allowing pie rule any longer
50. Implementation of pie rule on client and server calls
51. Implementation of timer on server in order to call time
52. Implementation of timer on client side in order to report time to server.
53. Implementation of time on the game board, displaying losing turn if player does not make a move
54. Basic configuration properties files in order to pass in seeds, houses, random distribution, and time limit per move
55. Setup program to host a server and multiple clients
56. Client must return a move within the designated time to server
57. Client connection to server should be acknowledge with WELCOME
58. Client should receive and print out INFO statement with game configuration including type of player
59. Setting up basic calls for ready client side for player vs player
60. Setting up basic calls for ready client side for computer vs player
61. Setting up basic calls for ready client side for computer vs computer
62. Setting up basic return calls for begin server side for player vs player
63. Setting up basic return calls for begin server side for computer vs player
64. Setting up basic return calls for begin server side for computer vs computer
65. Verify that each move has been sent and received
66. Server report time, illegal, welcome, loser, winner or tie
67. Implementation of returning illegal from client
68. Implementation of returning loser from client
69. Implementation of returning winner from client
70. Implementation of returning tie from client