**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. Computer can randomly select a valid move
20. Computer looks ahead one move in tree
21. The program checks if the leaf is at a final state
22. If at a final state, then assign the max/min possible utility value
23. If the leaf is not final state, then evaluate the state of the board to select the best possible move (using recursion/0)
24. Complete the implementation of the minimax tree
25. Iterative deepening implemented (the search for minmax tree)
26. Integrated the iterative deepening implementation with the alpha pruning function
27. Integrated the iterative deepening implementation with the beta pruning function
28. Evaluate the same move from all the chance nodes
29. Find the expected value for the move using probability
30. If Max node take the max value if Min node take the min value
31. Revaluates the utility using the now found alpha and beta and min/max value
32. Returns the value and the computer plays its move.
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 client connection
39. Integrate the server
40. Integrate the client server architecture with the game
41. Implement remote services
42. Integrate the remote client services with the game developed