**Team 4 – Project 2***Backlog, Initial Burndown Chart, & Sprints*  
Members: Patrick Casey, Matt Hacker, and Megan Kerins  
**CSCE 315-501**  
*Checkpoint #1*

*First Deadline*

For our project, our “SCRUM” master/project manager will be Megan Kerins as she has the most experience with Java out of the group.

PRODUCT BACKLOG:

* User launches program a window opens with the title of the game and creators.
* After a period of time (a wait function or so) the display switches to the actual interface
* There will be buttons on the top of the board reading “New Game”, “Board State”, and “Undo”
* New Game will simply reset the pieces to how they were when the game was started. No Titles or creator names will display
* Board State will either open a separate window of the *current* board state, or simply pause the game and display it over the screen
* If the board state is displayed over the screen, a button will be in the upper right corner that says “Back” which takes you back to the game of play
* The initial board state is displayed, then goes away
* “Undo” undoes the previous move, this is only possible for the Human player
* Before the game can be started the AI intelligence must be chosen
* Three Buttons will be listed somewhere on the window or this will be a choice from the command line that is chosen before the program is launched.
* Minimax tree is used for the AI
* Easy will make the board look ahead 1 move
* Medium will allow the AI to look ahead 2 moves
* Hard will allow the AI to look ahead 3 moves
* Our board will have a graphical interface, so the P1 will move by a click-and-move process.
* User gets response of invalid if proposing an invalid move
* User also gets a response if they still have valid moves to initiate, this is shown at bottom of window (either a green box or simply text saying P1)
* After the turn is over, the box will turn red or the text will switch over to AI
* The user sees a display of the current board state, again to exit the button “Back” must be clicked
* Board state is checked to determine if either player has won, and is checked after every move
* Game is checked to determine if maximum number of moves has been exceeded. This too is done after every move.
* Now it is the computer’s move (AI)
* Computer identifies all valid moves it can make and checks the board state (it does not appear across the screen this is done internally)
* Computer looks ahead in tree (again, done internally. How far ahead depends on its skill level)
* After every move, Human or Computer, the Minimax tree in the Computer’s class is updated for its best possible move
* Computer can ONLY select valid moves
* Like the user, after turn is over the board state is displayed on the screen
* “Back” must be selected to return to game
* The game is again checked to determine if maximum number of moves has been exceeded.

.

.

. 29 so far (not including the 3 below)

        Alpha-beta pruning implemented at one level (possibly later changed to all levels)

         Iterative deepening implemented

         Board evaluation function works for determining winner/loser

I would assume that teams would have on the order of 40-60 items in the backlog, but you are welcome to have more.

BURNDOWN CHART:

Main priority is to actually have an interface for the game with all of the pieces on it in the correct positions and correct colors execute when the program is run. This includes having a window pop-up displaying the name of the game and team member names, then after say 5 seconds transition to the actual interface of the game that will be played. Being human, we have the initiative and are allowed to start the game.

All the buttons should at least be there, whether they work or not is not an issue. For the first sprint we want the game to have all its pieces ready to implement.

FIRST SPRINT BACKLOG:

* Research Game: know how many pieces, possible moves, valid/invalid ones
* Draw the minimax tree (either on a computer or by hand)
* Get board interface to display
* Have pieces on board and an idea of the sort of method for moves.
* Define valid and invalid moves
* Define the rules of the game