**CSCE 315 - Group 20: Chess Game System**

**Team Organization:**

Project Manager: Hung Bui

Programmers: Hunter Cleary, Matt Mora

**Product Backlog:** (Importance from 1-5, 1 least, 5 most)

* Program to manage game (**Overall importance 5**)
  + Maintain board state (**5**)
    - GUI changes when necessary
  + Call functions from interactive input and/or AI routine to get moves (**5**)
    - If interactive input - check move, if AI routine - call AI functions
  + Check for valid moves (**5**)
    - Check every move if valid or not - print error message on GUI if invalid
  + Determine winner (**5**)
    - Determine every move, checkmate or stalemate
* Graphical User Interface for user input (**Overall importance 5**)
  + Create board (**5**)
    - Simple 8x8 grid of alternating light and dark spots with algebraic notation (for previous moves list)
  + Create chess pieces (**5**)
    - Use open-source images
  + Timer (**3**)
    - Once time elapsed for user - game over, same with AI
  + Reset button (potentially an undo button) (**2**)
    - Reset at any time, undo a previous move from user
  + Previous moves list (**1**)
    - Use algebraic notation to determine previous moves
  + 2 modes - Player vs Player, Player vs AI, (potentially AI vs AI) (**2**)
* AI for playing chess (**Overall importance 4**)
  + Tree-creation (**5**)
  + Basic minimax algorithm (**5**)
  + Utility function (**5**)
  + Pruning
    - Alpha cuts (**5**)
    - Beta cuts (**5**)
    - Combination (**3**)
    - Ordering (**2**)
* Client-server model for remote play with AI (**Overall importance 2**)

**Initial Burndown Chart:** (Number of hours to complete tasks: **Total hours 40**)

* Program to manage game (**Total hours 8**)
  + Maintain board state (**3**)
    - GUI changes when necessary
  + Call functions from interactive input and/or AI routine to get moves (**1**)
    - If interactive input - check move, if AI routine - call AI functions
  + Check for valid moves (**2**)
    - Check every move if valid or not - print error message on GUI if invalid
  + Determine winner (**2**)
    - Determine every move, checkmate or stalemate
* Graphical User Interface for user input (**Total hours 12**)
  + Create board (**2**)
    - Simple 8x8 grid of alternating light and dark spots with algebraic notation (for previous moves list)
  + Create chess pieces (**2**)
    - Use open-source images
  + Timer (**1**)
    - Once time elapsed for user - game over, same with AI
  + Reset button (potentially an undo button) (**2**)
    - Reset at any time, undo a previous move from user
  + Previous moves list (**3**)
    - Use algebraic notation to determine previous moves
  + 2 modes - Player vs Player, Player vs AI, (potentially AI vs AI) (**2**)
* AI for playing chess (Uses minimax algorithms) (**Total hours 14**)
  + Tree-creation (**2**)
  + Basic minimax algorithm (**3**)
  + Utility function (**1**)
  + Pruning
    - Alpha cuts (**2**)
    - Beta cuts (**2**)
    - Combination (**2**)
    - Ordering (**2**)
* Client-server model for remote play with AI (**Total hours 6**)

**Sprint 1 Backlog:** (Importance from 1-5, 1 least, 5 most)

* Program to manage game (**Overall importance 4**)
  + Maintain board state (**5**)
    - GUI changes when necessary
  + Call functions from interactive input to get moves (**4**)
    - If interactive input - check move
  + Check for valid moves (**4**)
    - Check every move if valid or not - print error message on GUI if invalid
  + Determine winner (**3**)
    - Determine every move, checkmate or stalemate
* Graphical User Interface for user input (**Overall importance 5**)
  + Create board (**5**)
    - Simple 8x8 grid of alternating light and dark spots with algebraic notation (for previous moves list)
  + Create chess pieces (**5)**
    - Use open-source images
  + Timer (**1**)
    - Once time elapsed for user - game over
  + Reset button (**2**)
    - Reset at any time
  + Previous moves list (**1**)
    - Use algebraic notation to determine previous moves
  + 1 mode - Player vs Player (**3**)