**Chess Game**

**Project Proposal**

R.J. Pereira

Alexander Rose

Andrew Urioste

Alice Uwase

Spring Semester 2019

CIS 434 - Software Engineering

**Overview of the Requirements**

The requirements for this task are as follows; The program should consist of a Graphical User Interface (GUI) which will be an interactive chess board. The pieces will be selectable and only move per the rules of Chess. Initially, the program will be designed to allow for two users to compete against each other. Following successful completion of the two-user interface, the software should be designed for a single player to compete against an artificial intelligence. It is important that the software is visually and functionally friendly to the users.

**Goals and Objectives**

Board Representation – The board will be represented as a box consisting of sixty-four squares. Each square will have an assigned value in a matrix, using assignment by A-H to represent the column, and 1-8 to represent the row. The boxes will be interchanging colors to better represent a chess board. The board will also have the option to surrender and reset the game and exit back to the splash screen.

Splash Screen – There will be an initial splash screen that will allow the user to select whether it will play a game against another player, the computer, or exit.

Player Class – In order to represent each side, there will be a class that represents a player. This class will contain the objects for the pieces, and the method for checkmate to determine if the player is victorious or losing.

Piece Representation – In order to accurately represent pieces and their movements, an object will be made for each type of piece. This will include the following pieces: King, queen, knight, bishop, rook, pawn. The object will include methods that dictate the allowable movement of the piece in the matrix, both by following the rules of movements for the piece and whether it is attempting to enter an occupied space, or capture another player’s piece. There will also be a variable which indicates the location of a piece, including whether it has been captured. The king object will have a variable which indicates if it is threatened to interact with the check and checkmate method.

Check and Checkmate – When a piece is moved into a new location, the check method will test the available moves of the opposing player’s king piece. If the piece is threatened, check will be declared. If it is unable to move, then checkmate will be declared and the player will win.

**Project Plan**

The team will agree upon a high-level overview of the structure and functionality of the game. This step will ensure that the team has a cohesive vision regarding the game’s development strategy. Some deliverables resulting from this step are pseudo-code detailing important object methods, a class diagram that details the necessary objects and their relationships to each other, and a prioritized list of increments to be developed.

Code development will involve writing the required classes using the Java language and interleaving them to attain the required game functionality. Task cards may be used to track and assign development tasks in this stage. The team will produce the game in increments. Each increment will add new functionality to the existing base. The team will use Javadoc to generate documentation in HTML format from the source code. The source code for each class will contain document comments that describe its major methods, fields and attributes.

The team will test each increment using JUnit to ensure that the game functions as intended. JUnit is a unit testing framework for the Java language. The basic idea is that objects are instantiated, and their methods checked to ensure proper functionality. Some of the game features that will be checked with JUnit testing are piece collision detection, piece capturing, tracking of captured pieces, and check and checkmate logic. Collision detection testing will instantiate various pieces which will intentionally be moved onto the same spot on the board. When the pieces are on the same team, the game should warn the user that two pieces cannot occupy a single position. When the pieces are not on the same team, the piece that moves onto an already occupied position should capture the piece on that position. A test will be written to ensure that the game properly accounts for which pieces have been captured. Various testing scenarios will be written to ensure the check and checkmate logic works as intended. The tests will consist of several scenarios which result in check or checkmate conditions. Each scenario consists of a series of moves performed by pieces on the board. The test will check whether the game can properly identify cases when check or checkmate occurs.

**Projected Roles and Responsibilities**

R.J. Pereira will contribute to the development of the game’s code, write tests using JUnit5 in order to ensure that the game functions as intended, and maintain an up to date class diagram that describes the structure of the game.