Team: Hunter Leise
Vincent Mahathirash
Raymond Duncan

Title: Chesspionage

1. Implemented Features

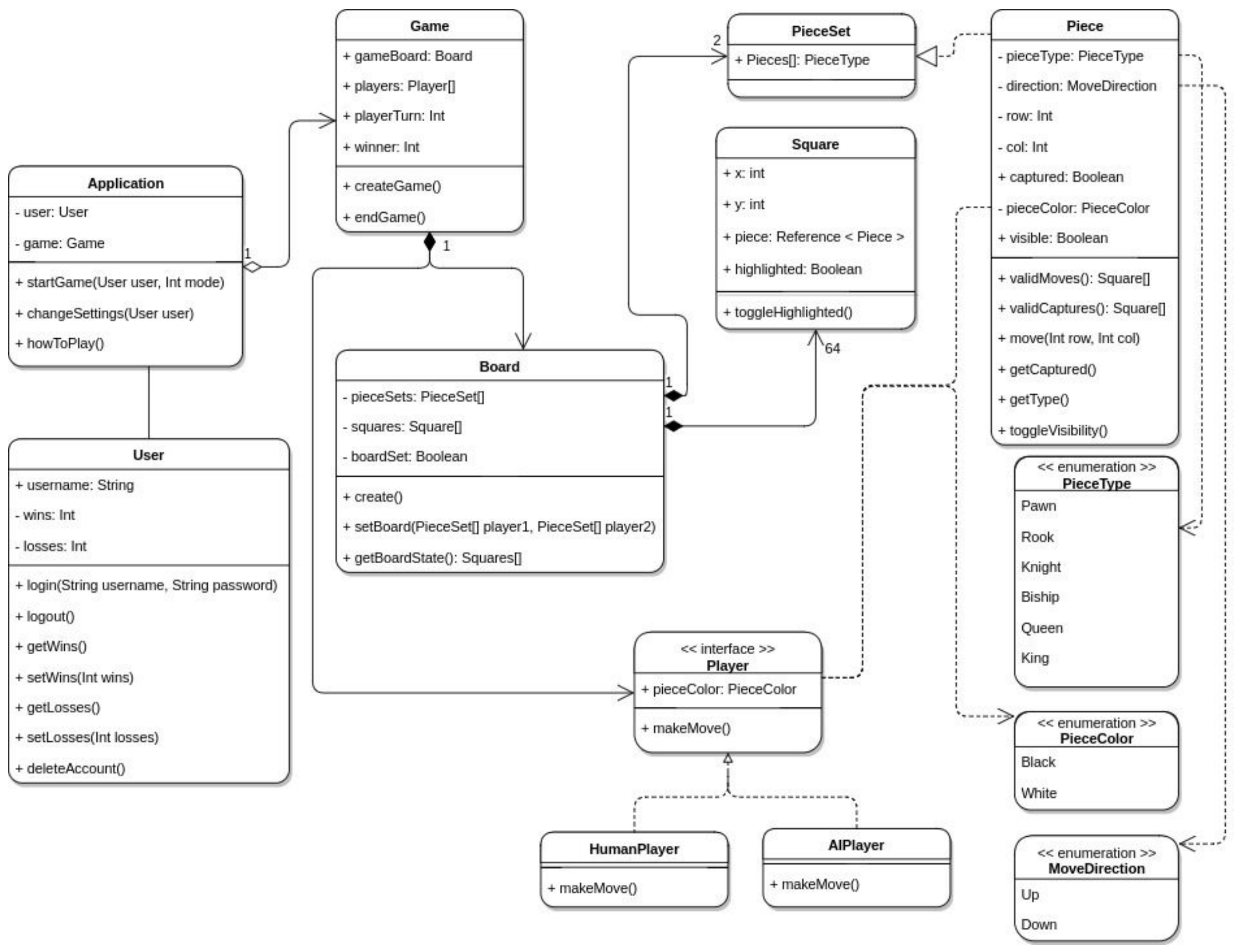
Implemented User Requirements		
ID	**Description**	**Priority**
US-01	As a player, I need to be able to start a new game so that I can play Chesspionage.	Critical
US-02	As a player, I need to be able to move pieces in their valid directions so that I can make moves to progress through the game.	Critical
US-03	As a player, I need to be able to capture opponent pieces when I land on them so that I can progress through the game.	Critical
US-04	As a player, I need to be able to hide the identity of my pieces so that my opponent cannot see them when we play locally on the same computer.	Critical
US-05	As a player, I need to be able to show the identity of my pieces so that I can view them.	Critical
US-06	As a player, I need to be able to play against an opponent on my local computer so that I can play with a friend.	Critical
US-07	As a player, I need to be able to play against an AI opponent on my local computer so that I can play by myself.	Critical
US-08	As a player, I need to be able to win the game when I capture my opponent's king.	Critical
US-09	As a player, I need to be able to view the instructions for the Chesspionage variant so that I can know how to play.	Critical
US-10	As a player, I need to be able to place my pieces on open spaces in my back two rows at the beginning of the game so that I can properly setup the board before playing Chesspionage.	Critical

2. Non-Implemented Features

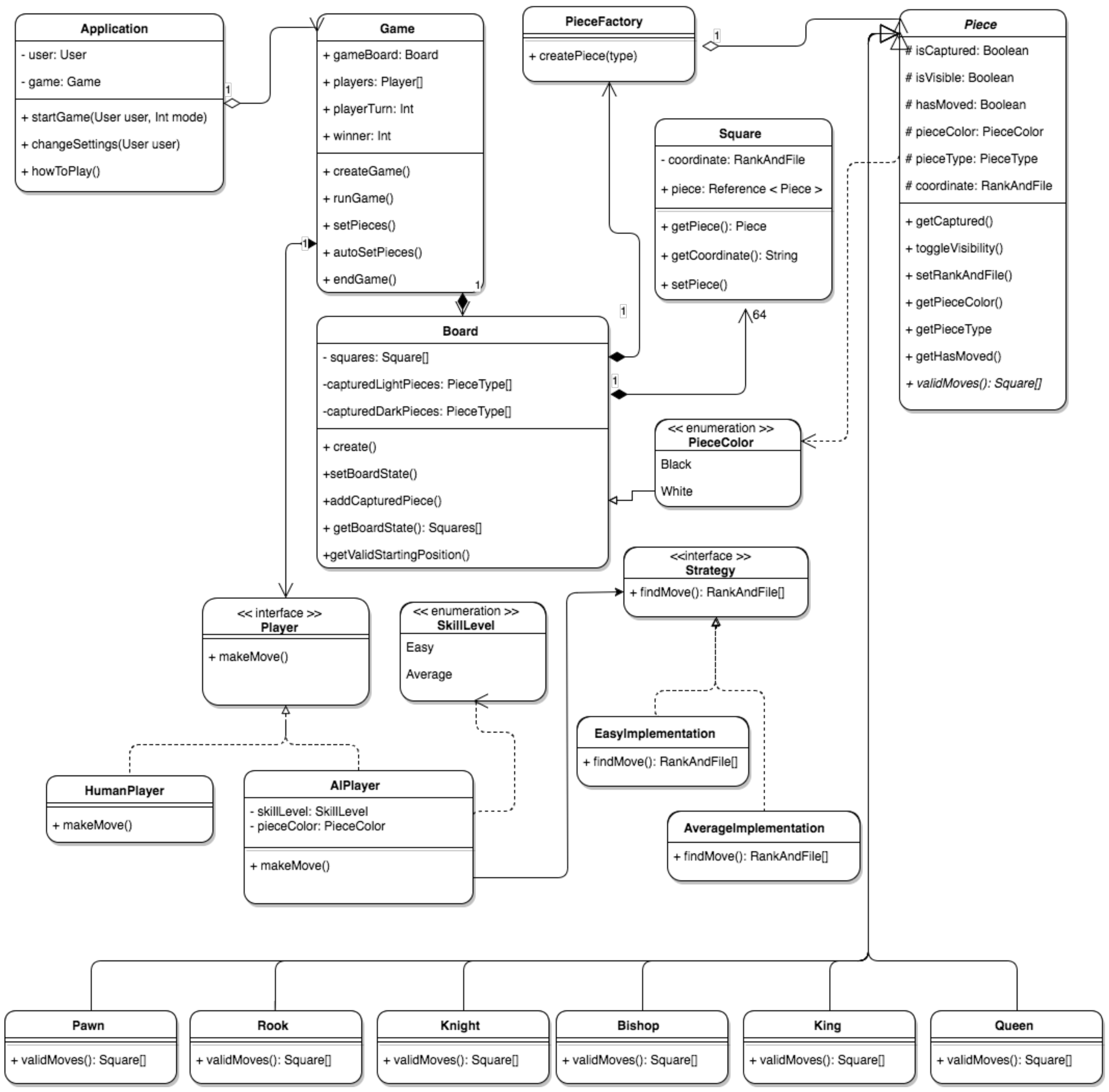
Non-Implemented User Requirements		
ID	**Description**	**Priority**
US-11	As a player, I need to be able to create a user account so that I can track data linked to my user account.	Critical
US-12	As a player, I need to be able to login to my user account so that I can access data linked to my user account	Critical
US-13	As a player, I need to be able to logout of my user account so that I can protect my privacy when I no longer wish to be logged in.	Critical
US-14	As a player, I need to be able to select whether my pieces are black or white so that I can have the freedom to select which side I'm on at the beginning of the game.	High
US-15	As a player, I need to be able to change my password so that I can match it to my current preference.	Medium
US-16	As a player, I need to be able to view my win loss record over previous games I have played so that I can get an idea of how good I have done thus far.	Low
US-17	As a player, I need to be able to delete my account so that I can protect my privacy if I prefer to no longer be involved in the service.	Low
US-18	As a player, I need to be able to change my username so that I can match it to my current preference.	Low

3. Class Diagrams

Part 2 Class Diagram



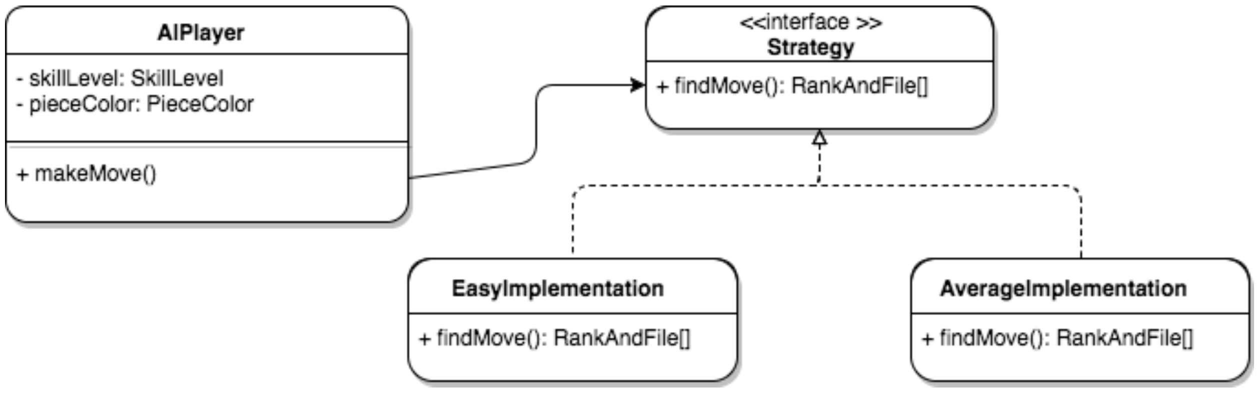
Final Class Diagram



We had quite a few changes from our original class diagram to what we ended up implementing. The majority of our major classes remained the same, but we added quite a few helper methods that we didn’t foresee needing when creating our original diagram. In addition, we deleted the user class because we ultimately decided not to implement the user account functionality. One major difference was how we shifted computational responsibilities between classes. For example, all of our toggle visibility functionality was moved away from the piece class and into the view class instead. The final major difference was our design patterns, which are described in detail below (in question 4).

4. Design Patterns

We ended up using the strategy design pattern for the AI player decision making because it allowed us to easily control the AI's difficulty by changing which strategy implementation we use at runtime. In doing so, we can easily extend the AI strategy to include more difficulty levels in the future.



5. What We've Learned

Through the process of creating, designing, and implementing a system, we've learned that it's much more difficult than it seems. After creating our original and refactored class diagrams earlier in the semester, we felt like we had a pretty solid understanding of what exactly we needed to include in our final project code. Once we started actually implementing Chesspionage, however, we quickly realized that creating a functional application is much more complex than we envisioned. There were plenty of helper methods and enumerations that we didn't realize we would need, and there were some ideal design patterns that we simply didn't have time to implement, such as the factory method for creating chess pieces. In the future, I believe it would be much better to do a decent amount of programming while simultaneously planning out the overall project structure. It seems that the two would help inform each other more efficiently that way opposed to strictly doing one before the other.