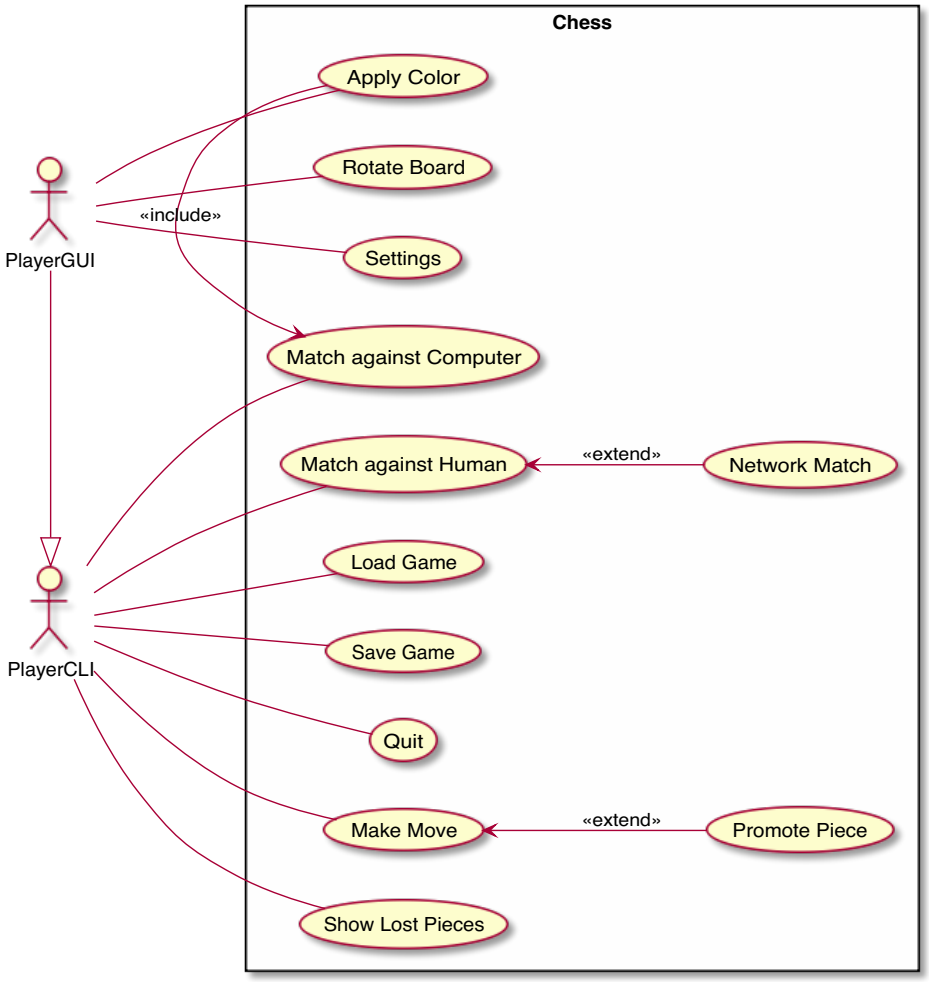
[200~

Anforderungsanalyse

Chess Use-Case Diagram



Storycards

Story: *Gegnerauswahl*

- Als Spieler
- möchte ich die Wahl zwischen einem Menschen und einem Computer als Gegner haben
- um sowohl alleine, als auch zu zweit spielen zu können

Akzeptanzkriterien

- Angenommen ich wähle einen zweiten Menschen als Mitspieler
- wenn das Programm gestartet wird
- dann werden zwei Menschen das Spiel spielen können

Story: *Zugwahl*

- Als Spieler
- möchte ich die Züge in textueller Form in die Konsole eingeben können
- um meine Spielfiguren zu bewegen

Akzeptanzkriterien

- Angenommen ich möchte meine Figur bewegen
- wenn ich zwei Koordinaten eingebe
- dann bewegt sich die Spielfigur auf dieses Feld

Story: *Rochade*

- Als Spieler
- möchte ich bei reiner Königsbewegung eine Rochade durchführen können
- um eine Rochade durchführen zu können

Akzeptanzkriterien

- Angenommen die Vorraussetzungen für eine Rochade bestehen
- wenn ich den König entsprechend bewege
- dann wird automatisch eine Rochade durchgeführt

Story: *Bauernumwandlung*

- Als Spieler
- möchte ich einen zusätzlichen Buchstaben hinter der Zugeingabe eingeben können
- um einen Bauern in einen Turm, Springer, Läufer oder eine Dame umwandeln zu können

Akzeptanzkriterien

- Angenommen ein Bauer steht einen Zug vor der von dessen Anfangsort weitesten entfernten Reihe
- wenn ich den Bauer eine Reihe nach vorne ziehe und hinter die Eingabe dabei ein R,N,B oder Q schreibe
- dann kann er von mir in einen Turm, Springer, Läufer oder eine Dame umgewandelt werden

Story: *Invalid Move*

- Als System
- möchte ich syntaktisch falsche Eingaben mit der Ausgabe „!Invalid Move“ beantworten und des Weiteren ignorieren
- um syntaktische Fehleingaben zu vermeiden

Akzeptanzkriterien

- Angenommen Der Spieler tätigt eine Eingabe
- wenn diese nicht syntaktisch korrekt ist
- dann gebe ich die Meldung „!Invalid Move“ aus

Story: *Move not allowed*

- Als System
- möchte ich regelwidrige Züge mit der Ausgabe „!Move not allowed“ beantworten und des Weiteren ignorieren
- um regelwidrige Züge zu vermeiden

Akzeptanzkriterien

- Angenommen der Spieler tätigt eine Eingabe
- wenn diese zu einem nicht validen Zug führt
- dann gebe ich die Meldung „!Move not allowed“ aus und ignoriere die Eingabe und warte auf neue

Story: *Allowed Move*

- Als System
- möchte ich gültige Eingaben bestätigen indem ich diese hinter einem vorangestellten Ausrufezeichen zurückgebe
- um dem Spieler zu bestätigen, dass der Zug korrekt war

Akzeptanzkriterien

- Angenommen der Spieler tätigt eine Eingabe
- wenn diese korrekt ist
- dann gebe ich ein Ausrufezeichen und dahinter die eingegebene korrekte Eingabe zurück
- Angenommen der Computer tätigt eine Eingabe
- wenn diese korrekt ist
- dann gebe ich ein Ausrufezeichen und dahinter die eingegebene korrekte Eingabe zurück

Story: *Schachbrett*

- Als System
- möchte ich zu Beginn und nach jedem Zug das Schachbrett mit der derzeitigen Spielsituation aus Sicht des weißen Spielers anzeigen
- um dem/den Spieler/n immer den aktuellen Spielstand wissen zu lassen

Akzeptanzkriterien

- Angenommen der Spieler macht einen Zug
- wenn der Zug gemacht wurde
- dann verarbeite ich die Eingabe und stelle das gesamte Spielbrett mit dem derzeitigen Spielstand inklusive des neuen Zugs aus weißer Sicht dar
- Angenommen das Spiel wird gestartet
- wenn das Spiel gestartet wurde
- dann zeige ich die Initalstellung des Spielfelds an

Story: *Beaten*

- Als System
- möchte ich geschlagene Figuren bei der Eingabe von beaten anzeigen
- um die Übersicht der geschlagenen Figuren zu zeigen

Akzeptanzkriterien

- Angenommen der Spieler möchte die geschlagenen Figuren sehen
- wenn der Spieler beaten eingibt
- dann gebe ich alle geschlagenen Figuren aus

Story: *Schach*

- Als System
- möchte ich eine Meldung ausgeben, wenn ein Spieler schach gesetzt wird
- um dem Spieler mitzuteilen, dass er fast verloren hat

Akzeptanzkriterien

- Angenommen Spieler 1 macht einen Zug
- wenn dieser Zug Spieler 2 schach setzt
- dann gebe ich eine Meldung aus, dass Spieler 2 im schach steht

Story: *Matt*

- Als System

- möchte ich eine Meldung ausgeben, wenn ein Spieler matt gesetzt wird
- um das Spiel zu beenden

Akzeptanzkriterien

- Angenommen Spieler 1 macht einen Zug
- wenn er damit Spieler 2 schach matt setzt
- dann gebe ich eine Meldung aus, dass Spieler 2 schach matt gesetzt wurde und beende das Spiel

Story: *Standard-out*

- Als System
- möchte ich Ausgaben kontrollieren
- um falsche Ausgaben zu vermeiden

Akzeptanzkriterien

- Angenommen ich gebe etwas aus
- wenn stdout eingegeben wird
- dann gebe ich ein Ausrufezeichen, !Invalid Move oder !Move not allowed jeweils mit Umbruch am Ende aus

GUI-basiertes Frontend

Story: *Gegner auswählen*

- Als Spieler
- möchte ich einen Menschen oder den Computer als Gegner wählen können
- um immer einen Gegner zu haben.

Akzeptanzkriterien:

- Angenommen Das Spiel wurde gestartet,
- dann kann ich zwischen Mensch und Computer wählen.

Story: *Schachbrett Draufansicht*

- Als Spieler
- möchte ich das Schachbrett von oben sehen
- um das Brett überblicken zu können.

Story: *Schachfiguren*

- Als Spieler
- möchte ich mit den gängigen Symbolen für die Figuren spielen

- um die Figuren erkennen zu können.

Story: *Schachbrett drehen*

- Als Spieler
- möchte ich das Schachbrett nach dem Zug drehen können
- um die Seite des aktiven Spielers nach unten zu drehen.

Akzeptanzkritrien:

- Angenommen das Spiel ist im Gange,
- wenn ein Zug beendet wird, und die Funktion aktiviert ist,
- dann wird das Brett um 180° gedreht.

Story: *Ziehen*

- Als Spieler
- möchte ich einen Zug durchführen können.

Akzeptanzkritrien:

- Angenommen Eine Figur ist ausgewählt,
- wenn ein Feld, auf das gezogen werden kann, angeklickt wird,
- dann wird der Zug durchgeführt.

Story: *Ungültige Züge ignorieren*

- Als Spieler
- möchte ich dass, ein ungültiger Zug ignoriert wird
- um die Regeln zu befolgen.

Story: *Anzeige geschlagene Figuren*

- Als Spieler
- möchte ich die geschlagenen Figuren sehen können,
- um meine Strategie anzupassen.

Akzeptanzkritrien:

- Angenommen Das Spiel ist im Gange,
- wenn eine Figur geschlagen wird,
- dann wird sie neben dem Spielfeld angezeigt.

Story: *Zughistorie*

- Als Spieler
- möchte ich die vergangenen Züge anschauen können

- um eventuelle Spielfehler zu finden.

Story: *Mögliche Züge anzeigen*

- Als Spieler
- möchte ich sehen wohin ich die Figuren bewegen kann

Akzeptanzkritrien:

- Angenommen Eine Figur wurde ausgewählt,
- wenn die Funktion aktiviert ist,
- dann werden alle Felder auf diese Figur bewegt werden kann, grafisch hervorgehoben

Story: *Ändern der Auswahl*

- Als Spieler
- möchte ich eine Figur auswählen nachdem ich schon eine andere gewählt habe,
- um die Auswahl zu korrigieren.

Akzeptanzkritrien:

- Angenommen Eine Figur ist ausgewählt,
- wenn ich eine andere Figur anklicke,
- dann wird diese zur ausgewählten Figur.

Story: *Schach Anzeige*

- Als Spieler
- möchte ich wissen ob ein Spieler im Schach steht.

Akzeptanzkritrien:

- Angenommen Ein Spieler steht im Schach,
- dann wird dies angezeigt.

Story: *Spielende*

- Als Spieler
- möchte ich sehen dass das Spiel zuende ist.

Akzeptanzkritrien:

- Angenommen Eine Spieler ist Schachmatt oder es steht Unentschieden,
- dann wird dies Angezeigt und das Schachbrett wird eingefroren.

Story: *Spiel abbrechen*

- Als Spieler
- möchte ich jederzeit zum Auswahlbirdschirm zurückkehren können
- um ein Spiel abzubrechen.

Akzeptanzkritrien:

- Angenommen Das Spiel ist im Gange,
- dann kann ich zum Auswahlbildschirm zurückkehren.

Story: *Spiel gegen Computer*

- Als Spieler
- möchte ich gegen den Computer spielen können
- um immer einen Gegner zu haben.