

CS 319 - Object-Oriented Software Engineering

Analysis Report

**Chess Game**

Prepared by Group 04

Kaya YILDIRIM - 21002071

Burak MANDIRA - 21301474

Berk ABBASOGLU - 21400812

Rabia Ecem AFACAN - 21101259

Proposed to

Bora GÜNGÖREN

[**1. Introduction**](#_f3pfnqwuqzph)[**2**](#_f3pfnqwuqzph)

[**2. Overview**](#_9sb5ucv7i87p)[**2**](#_9sb5ucv7i87p)

[2.1. Gameplay](#_uh5bye3el7du) [3](#_uh5bye3el7du)

[2.2. Piece Types](#_an6a6lc5iwsf) [3](#_an6a6lc5iwsf)

[**3. Requirement Specification**](#_ex6kgb31avdj)[**4**](#_ex6kgb31avdj)

[3.1. Functional Requirements](#_nu440gz2x9ra) [4](#_nu440gz2x9ra)

[3.1.1 Play Game](#_lfy05wlolwnx) [4](#_lfy05wlolwnx)

[3.1.2. Change Settings](#_az8mxpdpxqq) [4](#_az8mxpdpxqq)

[3.1.3. Past Winners](#_87twxltw098f) [5](#_87twxltw098f)

[3.1.4. View Credits](#_68ncio1gdpcm) [5](#_68ncio1gdpcm)

[3.1.5. View Help](#_8a52zl1cbch7) [5](#_8a52zl1cbch7)

[3.1.6. Make A Move](#_chhin2uj9vg7) [5](#_chhin2uj9vg7)

[3.2. Non-Functional Requirements](#_3j4qv85du84d) [6](#_3j4qv85du84d)

[3.2.1. User Friendly Interface](#_luvmgz60rku) [6](#_luvmgz60rku)

[3.2.2. Augmented System Design](#_ed36dq470l1b) [6](#_ed36dq470l1b)

[**4. System Model**](#_k75x0gcv1lt1)[**6**](#_k75x0gcv1lt1)

[4.1 Use Case Model](#_72kndkndqtn) [6](#_72kndkndqtn)

[4.1.1 Change Settings](#_cyy319uv1g7l) [7](#_cyy319uv1g7l)

[4.1.2 Past Winners](#_jiuj0a63tpjc) [8](#_jiuj0a63tpjc)

[4.1.3 Play Game](#_jchu0fo1r1hz) [9](#_jchu0fo1r1hz)

[4.1.4 Make Move](#_rlddklyxolze) [10](#_rlddklyxolze)

[4.1.5 View Credits](#_kqg4gk26xag4) [11](#_kqg4gk26xag4)

[4.1.6 View Help](#_d7hnvzlzd92s) [11](#_d7hnvzlzd92s)

[4.2 Dynamic Models](#_ngpr5g6vyhq0) [12](#_ngpr5g6vyhq0)

[4.2.1 Sequence Diagrams](#_ropq1upwe3q2) [12](#_ropq1upwe3q2)

[4.2.1.1 Play Game](#_l9x4ak4k81i1) [12](#_l9x4ak4k81i1)

[4.2.1.2 Change Settings](#_sifbz1elg22m) [13](#_sifbz1elg22m)

[4.2.2 Activity Diagram](#_qbhf7d3t1f8j) [15](#_qbhf7d3t1f8j)

[4.2.3 Class Diagram](#_c1yuhlxgboru) [17](#_c1yuhlxgboru)

[**5. Screen Mock-ups**](#_kwyolkulj617)[**18**](#_kwyolkulj617)

[5.1 Main Menu](#_pjnqo4w876wb) [18](#_pjnqo4w876wb)

[5.2 Settings Menu](#_oiwtuygfjq7f) [19](#_oiwtuygfjq7f)

[5.3 Credits Menu](#_qdm6edp1gww4) [20](#_qdm6edp1gww4)

[5.4 Help Menu](#_jxpz251asgy5) [20](#_jxpz251asgy5)

[5.5 Screenshot of Gameplay with Brown Background](#_ah8tdp7a9c2g) [22](#_ah8tdp7a9c2g)

[5.6 Screenshot of Gameplay with Black-White Background](#_is1jbboupchx) [23](#_is1jbboupchx)

[5.7 Pause Menu](#_do1u7zona5h5) [24](#_do1u7zona5h5)

[5.8 Pieces](#_ct2048pgjlqq) [25](#_ct2048pgjlqq)

[**6. Conclusion**](#_l8mf19mcd6k3)[**26**](#_l8mf19mcd6k3)

# **1. Introduction**

The game we decided to develop as part of our CS-319 project is a chess game. We plan to make it a desktop application intended for two-player gameplay, using only the mouse. The player can select the piece they want to play and a fitting destination to move it. We are not going to implement an artificial intelligence for the player to play against a computer controlled player, there needs to be two players for the game to be played. It will be very similar to the chess game we know. The aim is to utilise a strategy to get the board control using your pieces, eliminate your opponent’s pieces, and ultimately capture their king. Like the real chess game, it is an excellent tool to increase one’s strategic thinking.

An idea we keep reserved for the moment is to add bonus features to the original chess game, like randomly eliminating pieces or giving certain boosts for accomplishing certain things in the game, or by luck. These ideas might further be developed but for the moment will be considered as future work.

Our target audience can include anyone who knows to play chess. Granted there are at least two of them together, they can conveniently play the game, without the need for actual pieces. From young children to old people, anyone who loves and wants to play chess is in our target audience. Records of past games will be kept, so continued victories over your opponent will give the ultimate reward: bragging rights.

In this report, we will give information about our overall ideas, gameplay, and rules of chess. We will present our functional and nonfunctional requirements, show our use case models, dynamic models, class diagram and mock-ups.

For the purposes of this paper, “White” will be understood as the player controlling the white pieces, and “Black” will be understood as the player controlling the black pieces.

# **2.** **Overview**

Chess is a two-player strategy board game played on a chessboard, a checkered gameboard with 64 squares arranged in an eight-by-eight grid. The is history of chess goes long way back. Common belief on date chess originated is 7th century and it is believed to comes from old Indian game Chaturanga; which is also ancestor of some other strategy games such as xiangqi, janggi and shogi. In late 15th century chess comes to Spain and today’s chess game developed with shaping current powers of each piece. In chess, each player start with 16 pieces from 6 different piece types: one king, one queen, two rooks, two knights, two bishops, and eight pawns. All of these pieces moves differently. The purpose of the game is getting opponent’s king and win the game.

Chess Game we are developing is a Android based desktop game which has all features of classical chess game. The game can be played by two real person on one computer.

## **2.1.** **Gameplay**

Players of chess will need a mouse to play the game. Starting the new game, viewing help, settings and credits screens; and exiting the game will be controlled by mouse. Also etc key from keyboard provide players to exit from the program. While playing the game selecting and moving pieces will be controlled by mouse.

## **2.2. Piece Types**

Each of the six piece types moves differently on the chess.

King

- The king is most important piece of chess. It can move one square in any direction. The special move of king is called castling and involves also moving a rook.

Rook

- The rook can move any number of squares along any rank or file, however it can not skip over other pieces. Along with the king, the rook is involved during the king's castling move.

Bishop

- The bishop can move any number of squares diagonally, however it can not skip over other pieces.

Queen

- The queen combines the power of the rook and bishop and can move any number of squares along rank, file, or diagonal, however it can not skip over other pieces.

Knight

- The knight moves like an "L"-shape: two squares vertically and one square horizontally, or two squares horizontally and one square vertically.

- The knight is the only piece that can skip over other pieces.

Pawn

- The pawn is weakest piece of chess. It can not move backward unlike other pieces. The pawn can move one or two square from the beginning position, but from other positions it only can move one square. It can take opponent’s piece when it is in cross position.

## 

# **3. Requirement Specification**

## **3.1. Functional Requirements**

### **3.1.1 Play Game**

Our desktop application, Chess, is played by the same set of rules as of the regular chess game with two players. Each player aims to eliminate their opponent’s pieces, build an advantageous board, and try to capture their king while protecting their own. The strategies that can be developed and utilised are endless, and each game is granted to take a unique path as the game progresses. When a player captures their foe’s king, the game is over, and the winner is recorded. From thereon, it is possible to start the game anew, with all the pieces back and the board reset. There will be no computer controlled opponent, so progressive play type like increasing difficulties is not going to be possible. The game will prove as challenging as your opponent, not a predefined difficulty issued to the computer itself.

### **3.1.2. Change Settings**

Players can change some of game settings depending upon their wishes. These adjustable settings are as follows:

· Sound On/Off

· Board Color

· Table Layout

· Show Legal Moves

Players can change these settings during or before the game. They can turn on or turn off game sounds. They choose their desired color for the board from a set of colors provided by the game. If the players want to change the locations of the pieces (i.e. the player whose pieces are placed at the top of the table wants his/her pieces at the bottom of the table), they can swap sides by using the table layout option. Additionally, players can open or close piece location guidance that shows possible locations the player can move his/her selected piece during the game.

### **3.1.3. Past Winners**

Players can look at this part to see the last 10 games’ winners. For each game, there is the winner’s name along with the loser’s name like “Ahmet beat Can”. It lists them from newest game to oldest game so that players can easily keep track of who won how many games against whom by looking at this part.

### **3.1.4. View Credits**

Player can get the contact information about the game developers in View Credits Screen. The purpose of this screen is getting possible suggestions from players to improve the game.

### **3.1.5. View Help**

Player can get the information about game in this screen. The main purpose of the game, how to play, rules and additional features which we are planning to add to the usual chess game will be figured out on View Help Screen. Because such intended features are unknown by players when they start to play a game, help screen will help them to adapt the game easier.

Addition to these information, a brief history of the chess game will be given in this screen. The purpose of giving such an information is increasing the interest of player to chess and enjoy them more.

### **3.1.6. Make A Move**

We keep track of the moves issued by both players and after each, we check the new board. Important things to consider are whether there has been a collision in which case the opposing player’s piece is removed; whether the king is checked, which forces the opposing player to take defensive measures to protect their king; and if the king is checkmated, which ultimately ends the game. To check all these things after each turn or move, we will utilise this method.

## **3.2. Non-Functional Requirements**

### **3.2.1. User Friendly Interface**

Chess supplies various features to make the players feel comfortable when playing chess. Adjustable board color, automatic table rotation, piece location guidance, and attractive sounds are the features Chess provides. We will try to make table, pieces, guidance animations appear as plain as possible. Also, automatic table rotation needs to be done in a way that players are not distracted. Because chess is a strategic game that requires deep focus and thinking, we do our best to make users feel comfortable when they are playing.

### **3.2.2. Augmented System Design**

We might add some unique features, in other words special boosts, to the game such as time limitation, taking out random pieces within a specific range, ability to restore players’ lost pieces et cetera by some restrictions. The game should be designed such that it has ability to support such new features when they are wanted to be added in the future.

# **4. System Model**

## **4.1 Use Case Model**

This section provides information about the main use case model of Chess game, detailed use case explanations are included below.

## 

**Figure 4.1 – Illustrates the use case model of Chess Game**

### **4.1.1 Change Settings**

* **Use Case Name:** Change Settings
* **Participating Actor**: Player
* Entry Condition:
* Player wants to change at least one of the settings listed below:

o Sound On/Off

o Board Color

o Table Layout

o Piece Location Guidance

* Player selects the “Change Settings” button from the menu.
* **Exit Condition:** Player selects the “Back” button to return to the game or the menu depending on when the user has clicked “Change Settings” button.
* **Flow of Events:**

1. Player clicked to the “Change Settings” button to adjust at least one of the settings listed above.

2. All adjustable settings of the game are displayed to the user.

3. Player adjusts settings depending upon his/her desire.

4. System updates changed game setting(s) accordingly.

5. System waits player to click the “Back” button.

6. Player clicks the “Back” button.

7. System returns player to the previous screen depending upon the screen from where change settings screen is reached (e.g. from the game play screen or the menu screen).

* **Special Requirements:** None.

### **4.1.2 Past Winners**

* **Use Case Name:** Past Winners
* **Participating Actor**: Player
* **Entry Condition:**
* Player wants to learn or remember who won the last 10 games.
* Player selects the “Past Winners” button from the menu.
* **Exit Condition:**
* Player learned who won the last 10 games.
* Player selects the “Back” button to return to the menu.
* **Flow of Events:**

1. Player clicks to the “Past Winners” button to see the winners of last 10 games.

2. Last 10 games’ winners along with their opponent’s name are listed chronologically from newest to oldest by the system.

3. Player looks the list.

4. System waits player to click the “Back” button.

5. Player clicks the “Back” button.

6. System returns player to the menu.

* **Special Requirements:** None.

### **4.1.3 Play Game**

* **Use Case Name:** Play Game
* **Participating Actor**: Player
* **Entry Condition:**
* Player chooses the “Play Game” option from the main menu.
* **Exit Condition:**
* The game finishes or the player quits the application mid-game.
* **Flow of Events:**

1. Game is initiated.
2. One player is assigned White, the other is assigned Black. White is always the one to start.
3. White Makes A Move (see 4.1.4).
4. Black Makes A Move (see 4.1.4).
5. Steps 3-4 are repeated until either White checkmates Black, or Black checkmates White.
6. Once either king is captured, the winner’s name is recorded.
7. The player is given the chance to return to the main menu or initiate a new game. This means the repetition of steps 1-7.

* **Alternative Flow of Events (1):**

3.1 White does not check Black’s king but denies all possible path the king might take.

3.2 Black does not have any pieces other than the king, or all other pieces have paths completely blocked. Any possible path the king might take is threatened by White’s pieces. Therefore Black cannot make a move.

3.3 The game ends with a stalemate. It is recorded as a draw.

* **Special Requirements:** None.

### **4.1.4 Make Move**

* **Use Case Name:** Make Move
* **Participating Actor**: Player
* **Entry Condition:**
* Player moves one of his pieces during a continuing game.
* **Exit Condition:**
* The move is made.
* **Flow of Events:**

1. White moves one of his pieces.
2. The system checks if there is any collision due to the move.
   1. If it collides with one of White’s pieces, then the move is invalid and not allowed.
   2. If it collides with one of Black’s pieces, then Black’s piece is removed from the game, the move is successful.
   3. If there is no collision, then the move is successful.
3. Steps 1 and 2 are repeated for Black.
4. Steps 1 through 3 are repeated until the game finishes.
5. The move is made.

* **Alternative Flow of Events (1):**

2.1 White checks Black’s king.

3.1 Black tries to make a move other than a protecting or moving the king (though such a move is possible).

3.2 The system gives an error, reminding Black the king is checked, and forces the player to act accordingly.

3.3 Black protects or moves the king.

3.4 The game continues until checkmate.

* **Alternative Flow of Events (2):**

2.1 White’s pawn reaches the end of the board (the same line Black’s king and queen was at when the game started).

2.2. White can choose any of the pieces captured by Black previously in the game, including the queen.

2.3 White’s pawn is replaced by the chosen piece.

2.4 The game continues until checkmate.

* **Special Requirements:** None.

### **4.1.5 View Credits**

* **Use Case Name:** Past Winners
* **Participating Actor**: Player
* **Entry Condition:**
* Player wants to learn the developers of application.
* Player selects the “View Credits” button from the menu.
* **Exit Condition:**
* Player learned who are the developers of application.
* Player selects the “Back” button to return to the menu.
* **Flow of Events:**

1. Player clicks to the “View Credits” button to see information about game developers.

2. Player looks the the Credits screen.

3. System waits player to click the “Back” button.

4. Player clicks the “Back” button.

5. System returns player to the menu.

* **Special Requirements:** None.

### **4.1.6 View Help**

* **Use Case Name:** View Help
* **Participating Actor**: Player
* **Entry Condition:**
* Player wants to learn or remember the information about the Chess game, its brief history and how to play game.
* Player selects the “View Help” button from the menu.
* **Exit Condition:**
* Player learned the game Chess and how to play the game.
* Player selects the “Back” button to return to the menu.
* **Flow of Events:**

1. Player clicks to the “Past Winners” button from main menu to see information about the game Chess, its brief history and how to play the game.

2. Player looks the Help screen.

3. System waits player to click the “Back” button.

4. Player clicks the “Back” button.

5. System returns player to the menu.

* **Special Requirements:** None.

## **4.2 Dynamic Models**

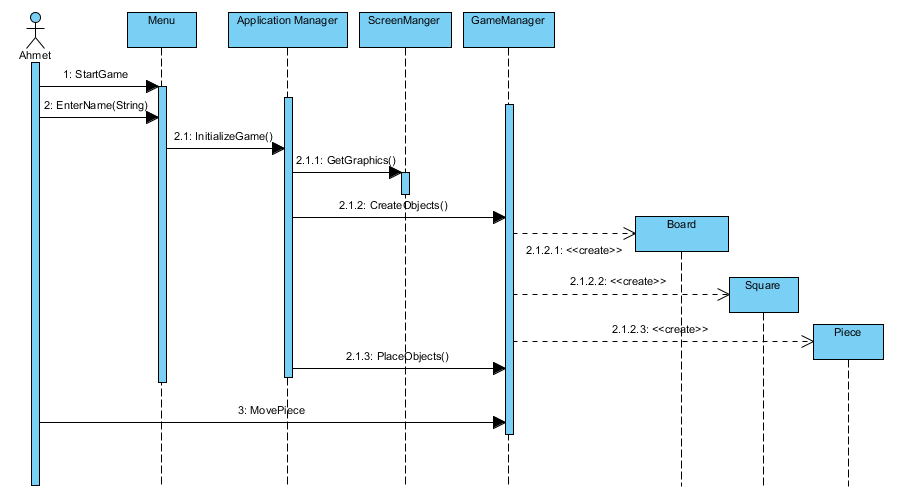
One can access detailed information about different options of the Chess. There are specific use cases for each options Chess contains.

### 

### **4.2.1 Sequence Diagrams**

#### **4.2.1.1 Play Game**

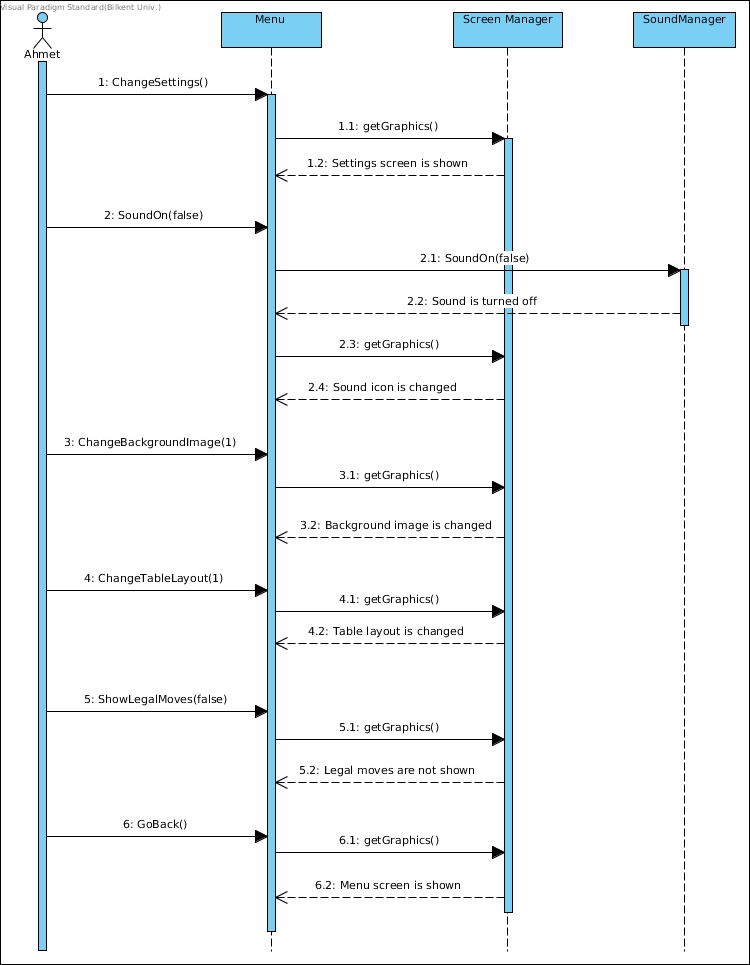
**Scenario:** Ahmet requests to start game by pressing “Play Game” option in main menu. Game asks Ahmet and his friend’s name. Ahmet and Ali enter their name into proper spaces on the screen by using keyboard. And then application gets graphic objects to draw on the screen. System creates objects and places them to proper locations. Lastly system waits for user click any piece to make a move.



**Figure 4.2.1.1: Example case for play game**

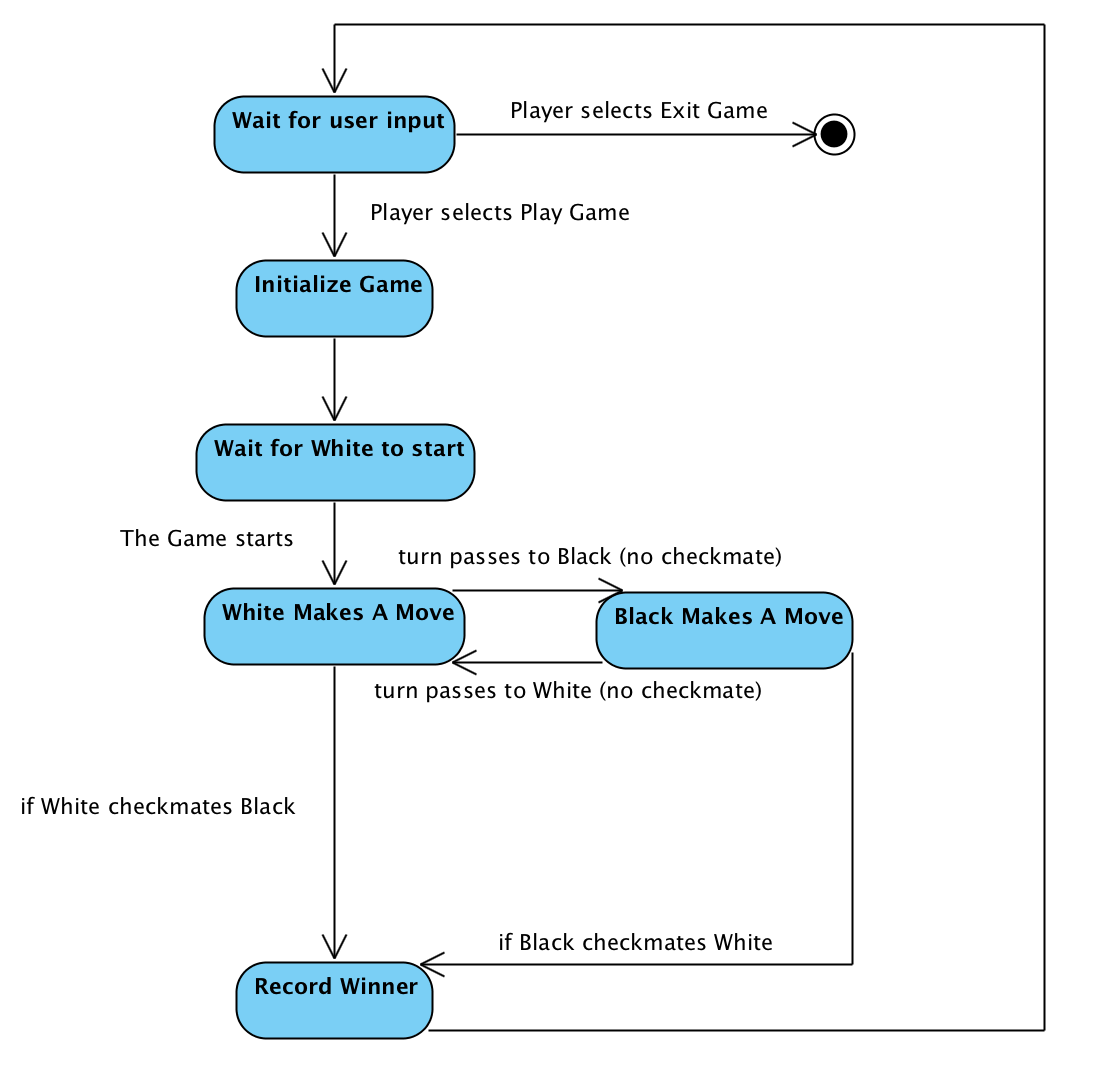
#### **4.2.1.2 Change Settings**

**Scenario:** Ahmet wants to change game settings before he and his friend start to game. He clicks on to the change settings button from the menu. Then, Ahmet turns the sound off and changes background image i.e. board image.Then, he wants the white pieces at the bottom of the table, therefore; he changes table layout. Since Ahmet and his friend are professional in chess and they do not want legal moves to be shown when they click a piece, they close this setting. Finally, Ahmet clicks back button from the settings screen and goes back to the menu. Sequence diagram of the scenario is shown in Figure 4.2.1.2.



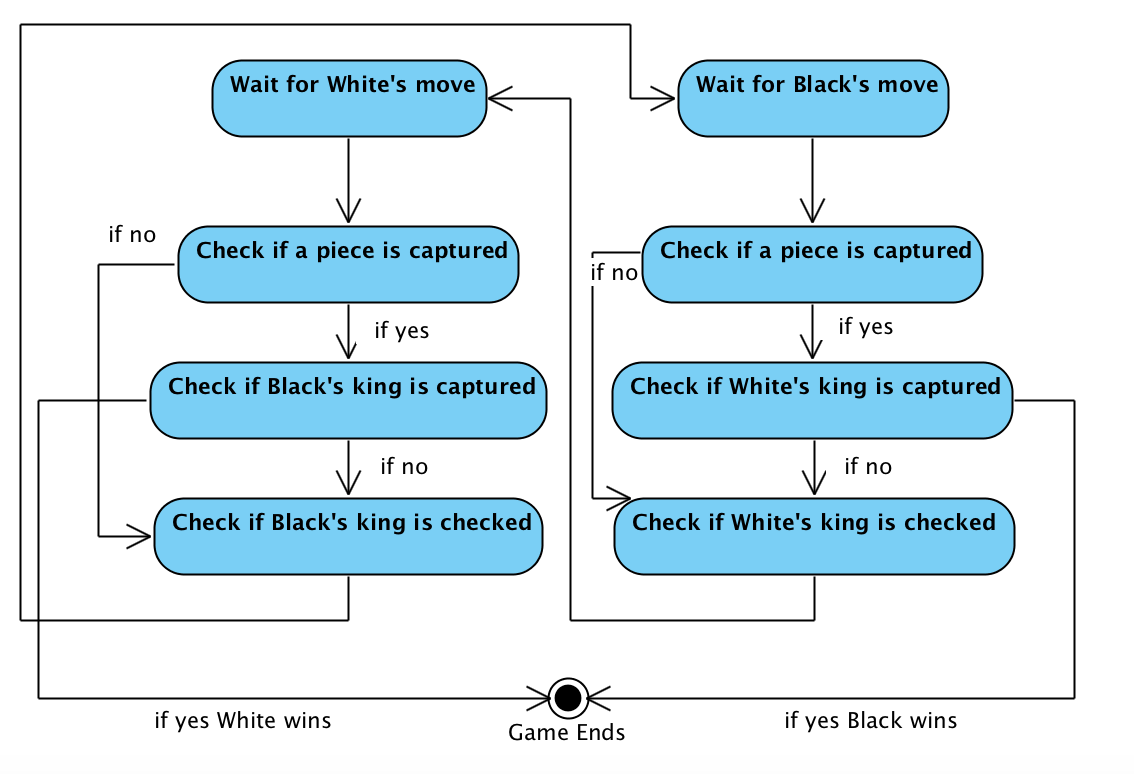
**Figure 4.2.1.2: Example case for change settings**

### **4.2.2 Activity Diagram**



**Figure 4.2.2.1: Activity Diagram**

At the start state of the game, the system waits for the user’s input. The user can either exit the application or choose to initialize a game. Once this is chosen, a game object is created, the board is set, the pieces are moved to position and the game awaits for White’s move (White always starts first). When White makes a move, the turn passes to Black. Since chess is a turn based game, we will check for anything critical, or update the board, after each turn. This checking is covered under the next diagram where we focus on the Make Move activities.



**Figure 4.2.2.2: Activity Diagram focused on Make A Move**

The start point is where we Wait for White’s move. After each move following White’s opening, the system checks if a piece is captured due to the current move. If the player has captured one of his opponent’s pieces, that piece is removed from the game. The next thing we check if a capture has occurred is whether the captured piece was the king. This case ends the game with a checkmate and the winner is recorded. In the case there has not been a checkmate (regardless if there was a capture or not), we check if the player has checked their opponent. If a check is present, the other player will have to either move or protect their king. As this restricts their available plays, it is crucial to identify this case. The game continues until a checkmate occurs.

### **4.2.3 Class Diagram**CS319-ClassDiagram.v2.jpg

**Figure 4.2.3: Class diagram shown in UML**

We would like to clarify relationships between Board, Square, PieceSet and Piece classes shown in the UML diagram above because they constitutes the backbone of the Chess game. An instance of Board class consists of 64 Square instances. Each of these 64 Square instances represent the each square in the 8x8 chess board. Also, a Board instance contains 2 PieceSet instances where each of those represents a player’s piece set. Each PieceSet instance includes 0 to 16 Piece instances where each Piece instance is one of the piece types that chess contains e.g. pawn, knight, queen, bishop, rook, king. At the beginning of the game, piece number is 16 for each PieceSet intances since no piece has been captured yet. While pieces are being captured, this number decreases. We did not show restrictions about the number of different types of pieces even though there are 8 pawns, 2 rooks, 2 knights etc. in the normal chess game. The reason is that Chess game allows pawns to be promoted, as the usual chess game does, to any piece that the player desires when s/he accomplishes to make his/her pawn reach to the its eighth rank.

# **5. Screen Mock-ups**

## **5.1 Main Menu**

When player executes the game, “Main Menu” appears shown in Figure 5.1.

There are 7 options in here. These are;

- Play Game: When player clicks this, game starts

- Past Winners: When player clicks this, last 10 winners list is opens.

- Settings: When player clicks this, setting menu opens.

- Credits: When player clicks this, credits menu opens

- Help: When player clicks this, help menu opens.

- Exit Game: When player clicks this, game shuts down.



**Figure 5.1: Main menu**

## **5.2 Settings Menu**

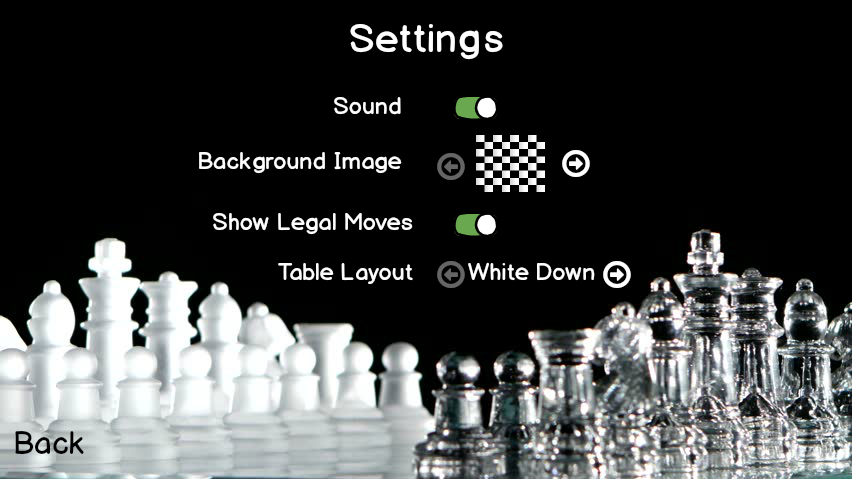
On the settings menu, there are 4 options which are:

- Sound: It allows to user to open and to close the sound of the game.

- Background Image: It allows to user to change background color.

- Show Legal Moves: It allows to user to show legal moves or not.

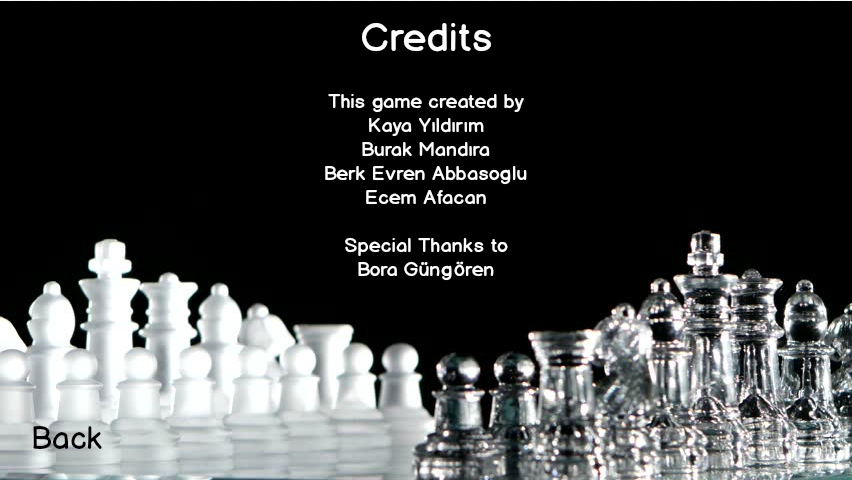
- Table Layout: It allows to user change which color of pieces are located.



**Figure 5.2: Settings menu**

## **5.3 Credits Menu**

Player can get the contact information about the game developers in View Credits Screen shown in Figure 5.3. The purpose of this screen is getting possible suggestions from players to improve the game.



**Figure 5.3: Credits menu**

## **5.4 Help Menu**

Player can get information about the game from the View Help Screen which is shown in Figure 5.4 below. The main purpose of the game, how to play, rules and additional features which we are planning to add to the usual chess game will be figured out on View Help Screen. Because such intended features are unknown by players when they start to play a game, help screen will help them to adapt the game easier.

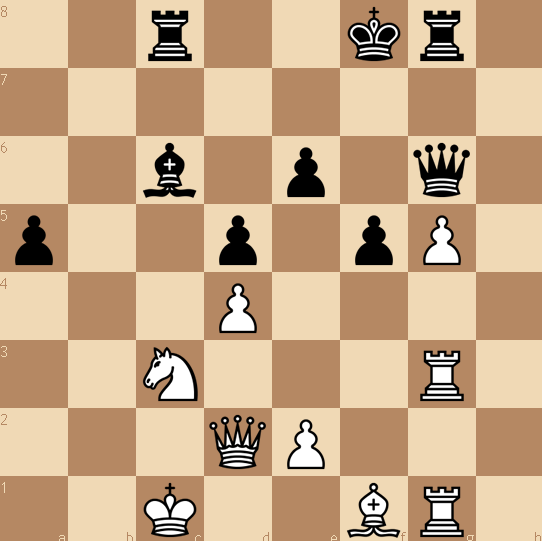
Addition to these information, a brief history of the chess game will be given in this screen. The purpose of giving such an information is increasing the interest of player to chess and enjoy them more.



**Figure 5.4: Help screen**

## **5.5 Screenshot of Gameplay with Brown Background**

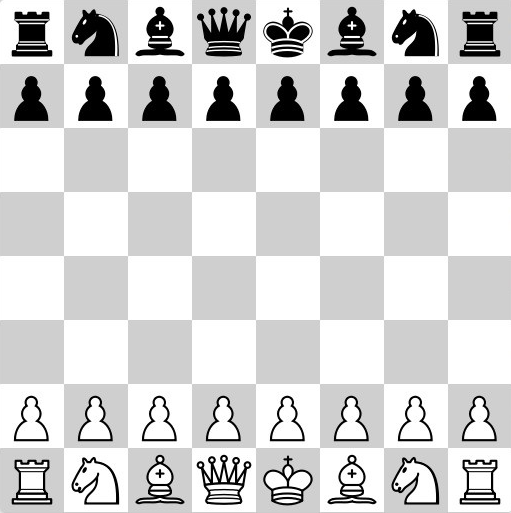
Figure 5.5 shows the game screen with brown background.



**Figure 5.5: Brown background**

## **5.6 Screenshot of Gameplay with Black-White Background**

Figure 5.6 shows the game screen with black and white background



**Figure 5.6: Black background**

## **5.7 Pause Menu**

When player click the “Esc” key from keyboard. Pause menu shows off.

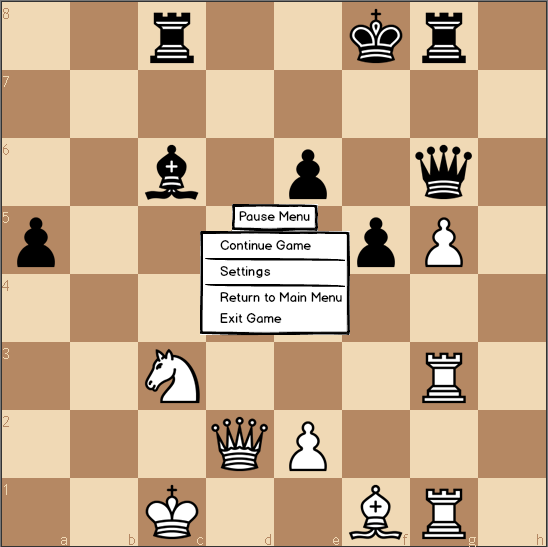
There are 4 options here. These are;

Continue Game: When it clicked, game goes on.

Settings: When it clicked, settings menu appears.

Return to Main Menu: When it clicked, recent game is closed and main menu shows off.

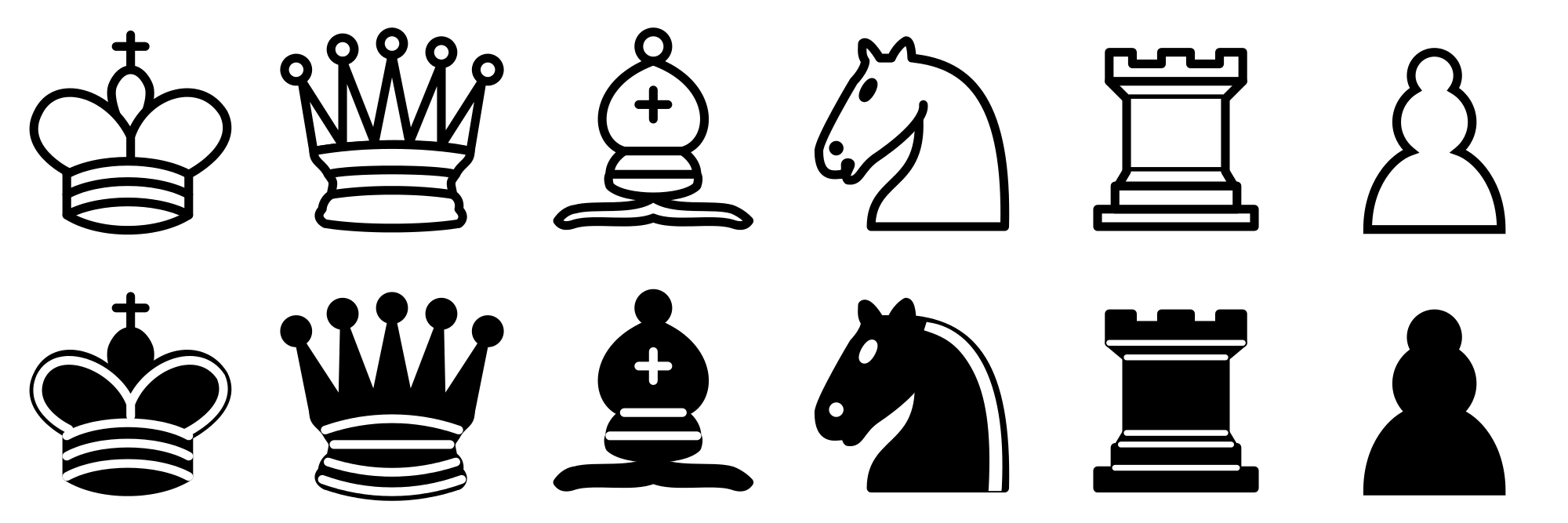
Exit Game: When it clicked, Game is terminated.



**Figure 5.7: Pause Menu**

## **5.8 Pieces**

There are 6 types of pieces. These are king, queen, bishop, knight, rook and pawn shown in Figure 5.8 for both black and white pieces.

**Figure 5.8: Piece types**

# **6. Conclusion**

As discussed in our report, our aim in this project is to develop a chess game as a desktop application. It will be regulated by the same rules as a standard chess game. In our report, we presented an overview of the game, the gameplay, and the types of pieces relevant for the game to be played.

We identified our functional and nonfunctional requirements. We will try to stay as parallel these guidelines during the further stages of our project, including the actual implementation.With the functions we need identified, we presented our system models. Our report includes use case models for changing settings, viewing past winners and playing the game. They have successful and alternate scenarios available.

Next, we discussed the dynamic models. We have two sequence diagrams, one for play game and one for change settings. These sequence diagrams present a possible scenario and follow its events. Additionally, our activity diagram presents a look at the entire gameplay mechanics, investigating steps taken before and after each point in the game. Most importantly, our class diagram is the basis for our coding. We spent most our effort on solidifying our ideas into classes and their relations and are confident we came up with a decent one. Furthermore, we included mock-ups for our game with many in game screens.

Our analysis report includes all the ideas we produced about our project in terms of diagrams, models, and analysis of different cases. It will act as our guideline for the rest of the project, so we tried to put great effort into making it as good as possible.