***Documentation***

***DESCRIPTION***

Chess is a two-player strategy board game that is played on an 8x8 checkerboard. There is a set of rules that define the game, but an infinite number of possibilities in which the game can proceed depending on the moves made by the two players.

Chess Pieces There are 6 kinds of pieces present on the board.

● Pawn

● Bishop

● Knight

● Rook

● Queen

● King

***Game Rules***

● The chessboard is two-dimensional of size 8x8 and has 64 squares where the pieces can stand.

● There are two online players, where one player controls the white pieces and the other player controls the black pieces. The colour is assigned randomly to the players.

● Each player gets alternate turns, starting with the white side.

● There are 6 types of chess pieces on the board: king, queen, knight, bishop, rock and pawn.

● At the start of the game, each player has

1 king ,1 queen, 2 knights, 2 bishops, 2 rooks and 8 pawns.

● Each piece can only move according to its own fixed rules.

● A piece can be captured if the opponent player’s piece moves to its current position.

● Once a piece is captured, it’s removed from the chessboard.

● There’s no option to cancel once a move is made.

● There can be two outcomes of the game: checkmate (one side wins) or stalemate (draw).

● Once an outcome is achieved, the game ends.

***REQUIREMENTS***

SERIALIZERS:

Serializers allow complex data such as querysets and model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data.

Board.outcome:

Checks if the game is over due to checkmate, stalemate, insufficient material, the seventyfive-move rule, fivefold repetition, or a variant end condition. Returns the chess.Outcome if the game has ended, otherwise None.

Alternatively, use is\_game\_over() if you are not interested in who won the game and why.

The game is not considered to be over by the fifty-move rule or threefold repetition, unless claim\_draw is given. Note that checking the latter can be slow.

Termination:

A [time off](https://www.chess.edu/glossary/time-off/) adjustment that automatically sets the remaining balance of a player’s [time off plan](https://www.chess.edu/glossary/time-off-plan/) to zero upon the player’s termination.

Chess.Pgn:

PGN (Portable Game Notation) is text format which records both the moves of the game (in standard algebraic notation) and any related data such as the names of the players, the winner/loser, and even the date the game was played.

FEN:

Forsyth–Edwards Notation (FEN) is a standard notation for describing a particular board position of a chess game. The purpose of FEN is to provide all the necessary information to restart a game from a particular position.

Django:

Django is a Python framework that makes it easier to create web sites using Python.

Django takes care of the difficult stuff so that you can concentrate on building your web applications.

Django is especially helpful for database driven websites.

Djangorestframework:

Django REST framework is a powerful and flexible toolkit for building Web APIs.

REST framework requires the following:

Python (3.6, 3.7, 3.8, 3.9, 3.10)

Django (2.2, 3.0, 3.1, 3.2, 4.0, 4.1)

SQLite:

The**SQLite**is**a free, open-source, public-domain software package that provides a relational database management system (RDBMS).**

**DRILL:**

Drill is a HTTP load testing application written in Rust. The main goal for this project is to build a really lightweight tool as alternative to other that require JVM and other stuff.

POSTMAN:

Postman is an API platform for developers to design, build, test and iterate their APIs

•API repository: Allows users to store, catalog, and collaborate around API artifacts in a central platform within public, private, or partner networks

***PROCESSING***

We will start by creating 5 APIs.

The first API we are creating the game by generating game id and player’s name .

In second API, we are listing the number of games played and the status of game, weather it’s a win, loss, ongoing or draw.

In third API, we check if our moves are legal or not. If it’s legal then the chess pieces will move if not it will provides us the available legal moves.

In fourth API, we check the current status of the board of a particular id we want.

In fifth API, we Get Board State at some specific move.

***ER Diagram***

In the following ER diagram we have a table named ChessGame which stores attributetes

such as id, status, pgn, created\_at and updated\_at. The description of each

attribute is as follows -

1. id: Game id which would be dynamically created and would be the primary

key of the table.

2. status: this shows the name of the game winner.

3. pgn: Record chess game in text form

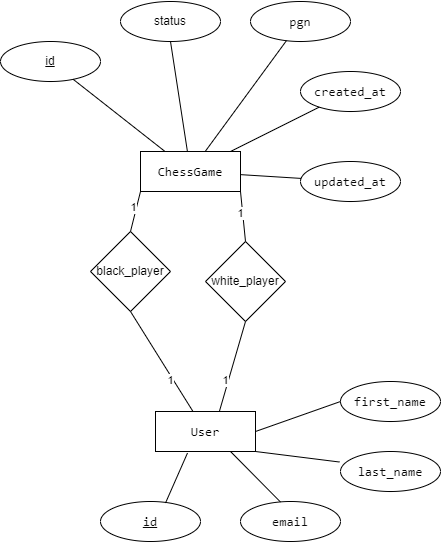
4. created\_at: the instance at which the game is created.

5. updated\_at: the instance at which the game is updated.

Player Table: A table containing the player information. It has following attributretes

1. white\_player\_name: The player name who choses to be the white player

2. black\_player\_name: The player name who choses to be the black player.



Class Diagram

