Your deliverable, while it can focus heavily on Low-Level Design, I'd appreciate any documentation you can include indicating that you did work through the High-Level to Low-Level design process. Make sure you have read through Ch. 5 and Ch. 6 materials before completing this assignment.

Specify how your program will work. This can be broken down into a couple of categories: specifying your classes, and determining the detailed design of your database, including normalization to 3NF (or more, if you believe it to be applicable, though that would not be required).

See Ch. 6 for review of related concepts, such as normalization, 1NF, 2NF, 3NF, determining classes with nouns, inheritance, generalization, object composition

Your deliverable for this assignment should include items such as:

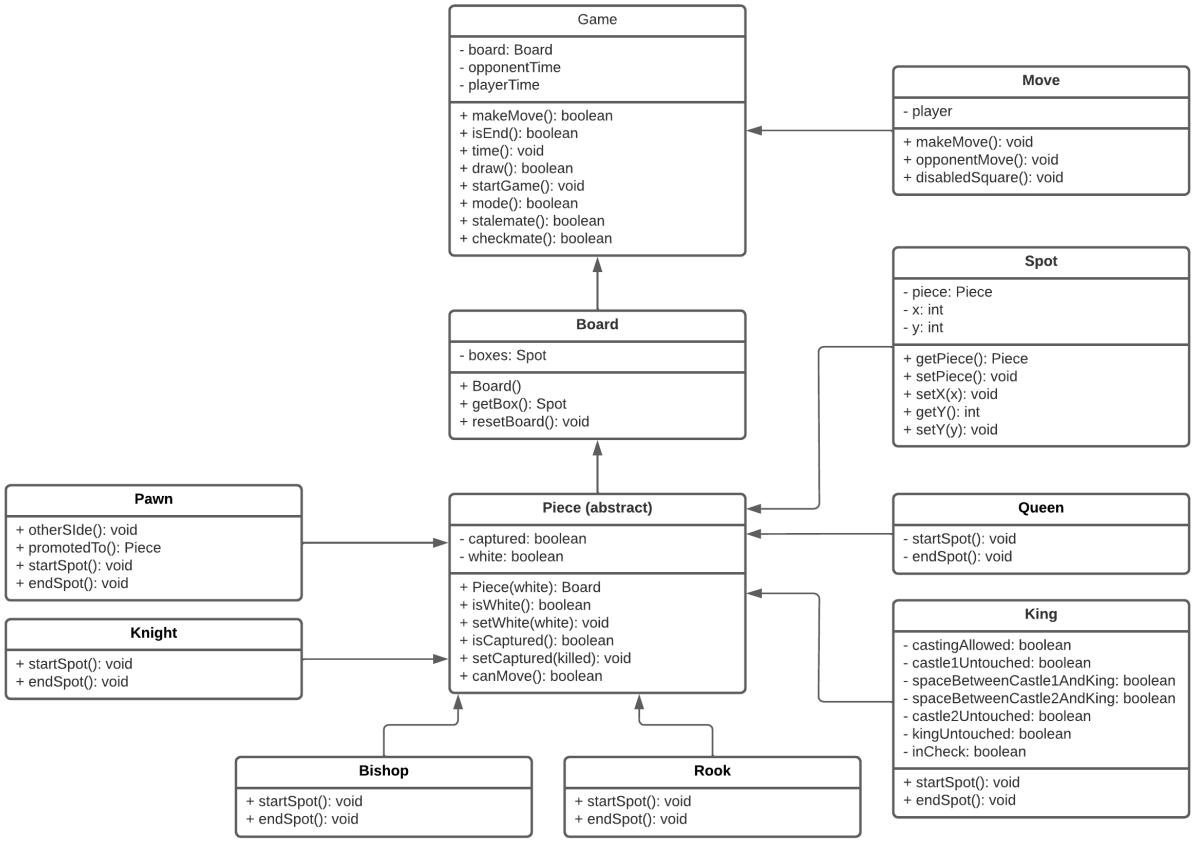
- description/diagram of architecture chosen:

  When designing our chess game, it must satisfy the user, functional, and non-functional requirements. We are mostly concerned with a user being able to

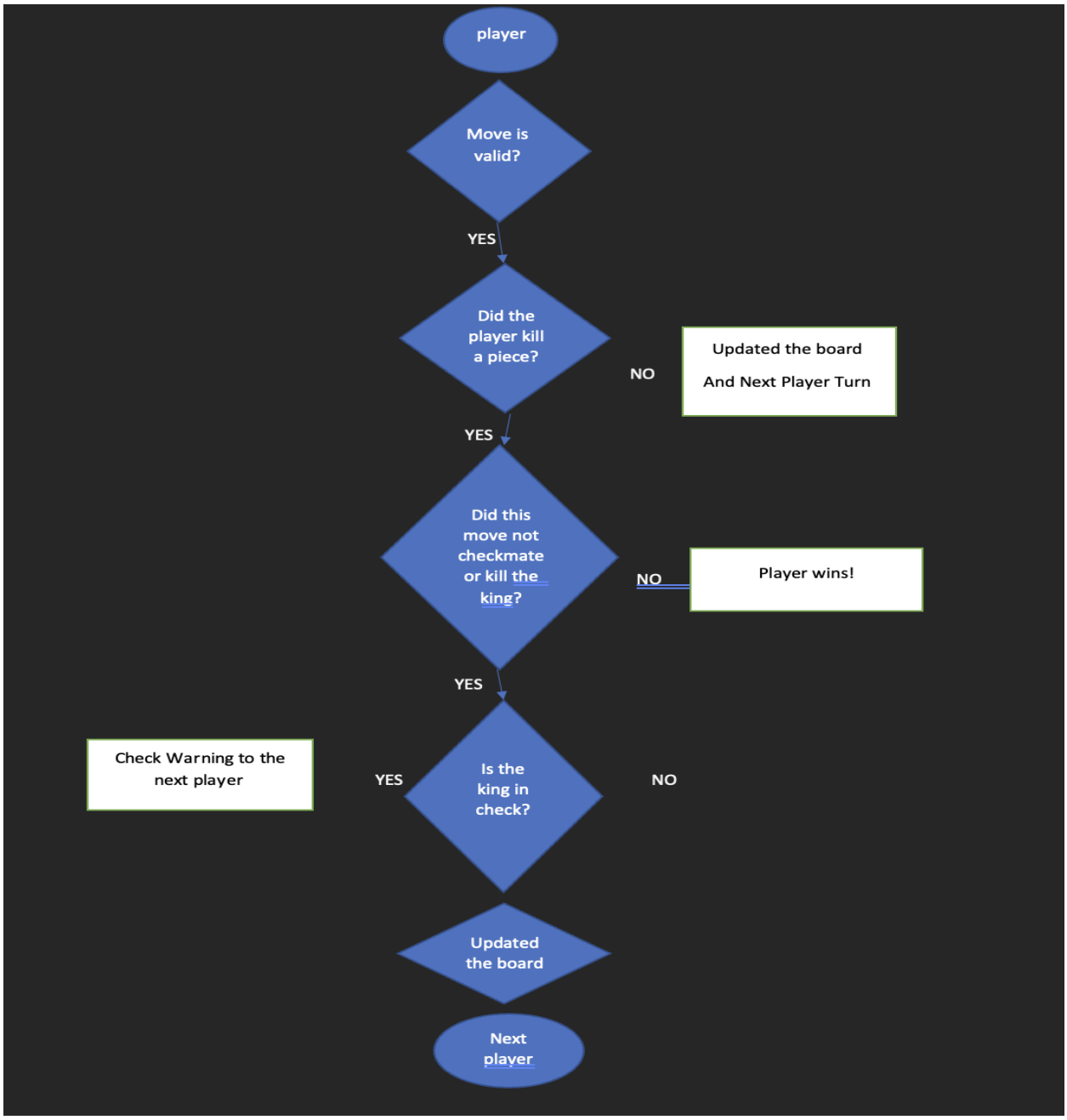
- Start a game in one of two modes: Regular or Blitz
- Castling
- Replacing pieces if a pawn reaches the other side of the board
- Draw
- Check
- Checkmate

In order to ensure that the user can accomplish all of these requirements, we need to consider how we store information about the pieces (where they are and where they can move), and we also have to enforce what moves a user can make, and when. Since we only have to keep track of data during the game, we have no need to store information in a database. Thus it makes sense for the project to be built using a monolithic architecture which is responsible for generating UI and storing data. We plan to build our application in .NET and Visual Studio IDE.

- class diagrams (UML):



- data flow diagram:



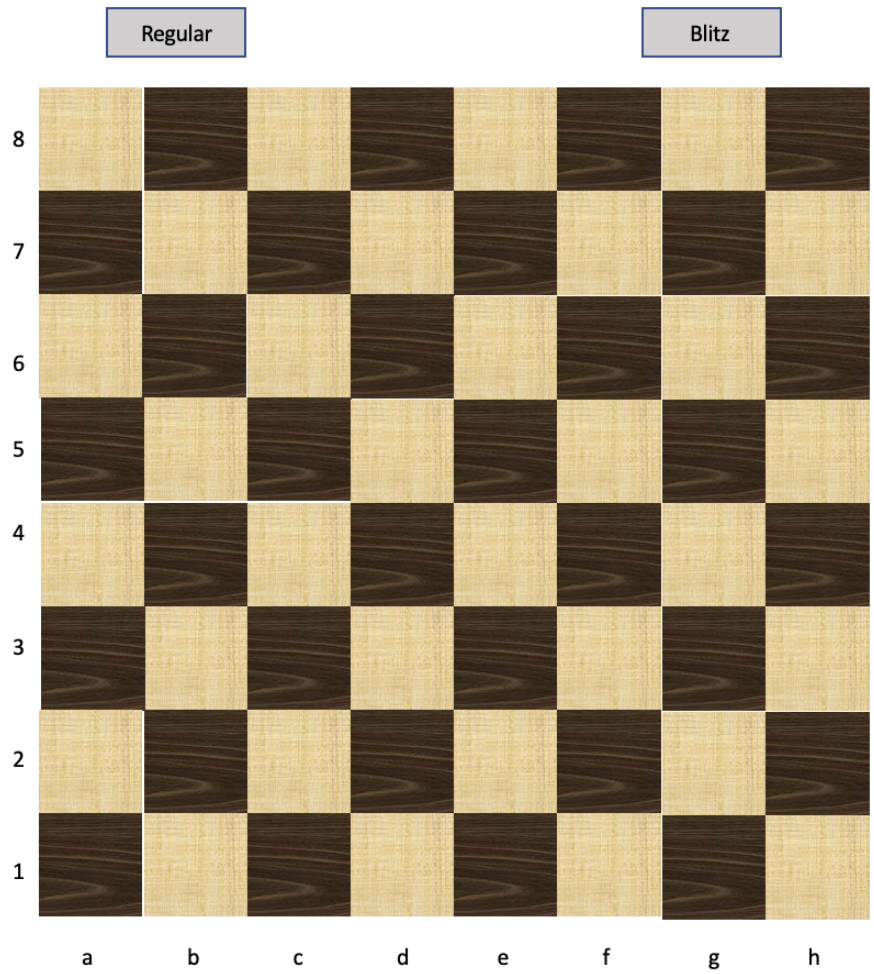
- database design:

Given that we only need to keep track of data temporarily when users are playing a game, we don't actually have to store data in a database. However, we can still determine a design for how to temporarily store data when users are playing a game. As shown in the UML diagram, we have a class Board that keeps track of pieces in each spot on the board. An instance of the class will be created when the game starts, and this object will be updated as users interact with the UI.

- hardware your system will run on:

This application should be able to run on Windows 11 computers.

- user interface:



- security (example, how login/system access will work):

For application security, we are mostly concerned with users cheating by making invalid moves. In order to prevent this from occurring, the application will prevent a user from making illegal moves by disabling currently invalid squares on the user interface. It will also keep track of whose turn it is to make a move so that a user cannot make an additional move before or during the opponent's turn.

- Reports: Pieces on the board, pieces off the board, castling allowed, checkmate activated, check activated, draw activated, winner

Grandmasters

Alazar Efuye

Ajmal Girowall

Austin Miller

Tnur Shifa