**Tech Stack justification**

Java was chosen as our programming language since it allows us to adhere to OOP in a stricter manner. We were all familiar with Java so there will be little to no time spent learning the language.

JavaFX was chosen as our primary API/library as it allows for programming of a custom GUI which is important for a board game. It is newer compared to other GUI libraries including JavaSwing. Swing is inbuilt, but much older and less relevant compared to JavaFX. JavaFX also allows for custom GUI design using a third party tool that allows us to 'drag and drop' GUI elements instead of coding using coordinates/pixels which will save a lot of time. Learning JavaFX will take some effort but there are multiple tutorials online to help with that process. This is also true for JavaSwing.

**Domain Model justification**

The domain model was created with the following steps:

1. Brainstorming all entities present within the system, i.e., 'Fiery Dragons'
2. Creating a list of responsibilities that the system needs to complete
3. Assigning responsibilities to different entities, then eliminating other redundant entities/creating new ones if needed

This method of creating the domain model results in a design that is less complex than one where every single entity that exists within the system is just 'shoved in', leading to certain entities only 'being there for the sake of being there'. In addition, this design allows the system to be more extensible, as there are less 'moving parts' for new modules and components to interact with, allowing for easier implementation of new features.

**UI Design rationale**

The UI design is heavily based on the actual board game. Various indicators such as glowing borders and colour coded borders/tokens are implemented for users to have an easier time understanding and navigating around the board.

Most of the UI design follows many other games of similar nature to allow people to adjust easily.