Github for this project (I plan to expand this): <https://github.com/scarlettp1619/Free-Chess/>

Build with #build.sh, apk will be output in /app/build/outputs/apk/debug

Introduction to this application:

Free Chess is a customizable Chess game for Android devices. The base game works the same way as any other Chess program would, with every rule you’d expect to be implemented. However, there are extra features that allow you to personalise the game toward your own Chess experience.

Currently, you can customize the rules for individual pieces within the app. Navigate to the settings cog wheel at the bottom right of the main activity, and you’ll find drop down lists that allow you to customize how pieces can move. If you want a king to be able to jump around the same way a knight would, that is something you can now do with minimal effort, as everything is completely self-contained.

There is also a menu for Chess news, which uses the News API to ensure you are fully updated on what is happening in the world of Chess.

Design rationale:

I have ensured that there are multiple screens (activities) to navigate through in the app, ensuring that need was met, though this was necessary due to the settings and news page.

Both explicit and implicit intents are at work in the app, as you have your main explicit intents (Home, Settings, News, Game and End Screen) along with implicit intents in the news app (opening the webpage associated with the article).

Menus are at work in the main activity, as you can navigate through multiple different activities using the buttons. This also allows the app to be less cluttered, as I don’t think playing Chess while reading the news on the same app would be beneficial to the overall user experience.

The app uses data storage (internal) with the config file, as this is written and read from the Android device. Originally, this was done through an asset, but was later changed to ensure the use of storage in the app.

The news section of the app uses the internet, as it fetches news from the News API. Everything from the article is displayed, and multiple can be displayed on the screen at once through a RecyclerView. The app shall also remain functional with no connection to the internet, as the News API will simply not get called until connection is established, and the game itself does not require access to the internet.

Novel features:

The main novel feature for this is the ability to customize how pieces can move differently from the standard rules of Chess. I haven’t seen any software or programs that allow you to customize your own pieces (though FairyChess exists, this isn’t quite the same), so I thought that kind of freedom should be implemented.

Outside of that, there isn’t anything new to the experience of Chess to add, though there are a few sound effects in the game to make the game a little more lively. There are plans to add more features in the future (such as making your own custom pieces), I just simply didn’t have time to do this in the short time span of the continuous assessment.

Challenges Faced:

Since I made this game completely from scratch (not using any existing Chess code), I had a very rough time with the game logic.

This explanation of challenges faced will include some explanation of Chess, so I will try to explain it as simply as possible in case you are not familiar with the rules.

To ensure the fundamental rules are still in place (checks, checkmates and stalemates) and to prevent illegal moves, I had to generate the legal moves for each piece on the board. To do this, I iterated through the possible squares they could reach, and if it’s blocked by their own piece/another, then the search is stopped and added to a list of possible moves for that piece.

After every move, I then check if these squares fall onto the opponent’s king, if so – it’s check.

That was a generally pretty easy implementation, the issue lies with removing the moves that would be considered illegal (to allow for checkmates, stalemates and preventing players from moving into check).

To do this, I originally iterated through every possible legal move, generated the legal moves in that temporary position for every piece, and if it was check I would remove the legal moves from the piece so they couldn’t move there. The only issue with this is it’s a ridiculously slow algorithm and would take upwards of half a second per move (even more if you added new rules).

So, I spent about a week redesigning the legal move generator so it would be more optimised. The biggest challenge here was ensuring that players couldn’t move pieces and open a line of sight into check for their opponents piece. To do this, I generate “discovered squares” for each piece. If there’s only one piece between the line of sight of the opponent’s piece and the current player’s king, they can only move that one piece to another square within that line of sight, otherwise it’d be check.

Though this seems like a pretty obvious fix, it’s not quite clear when you aren’t a very proficient Chess programmer. As seen by my Git commit history, I spent a lot of time thinking I had fixed a problem just for there to be another. I ended up asking a friend to test it for me, and they downloaded the app over 60 times before it started working. Fortunately, this new system now works without fault and is exponentially faster than before.

As for expansion of this project and improvement, I’m sure more optimisations can be done, it’s just not something I think is necessary for the time being.

I plan on developing a Chess playing bot for this, as that would be a good way to expand on my programming skills and venture off into the depths of AI.

I also intend on improving/expanding the “customizability” feature of the app, allowing players to select/upload presets for move sets, or even create their own pieces/upload their own images for certain pieces. A lot can be done for customizability, if I really wanted I can let players change the colour of the board to whatever they want. The focus is customizability, so there’s a million different paths I can go down for this.