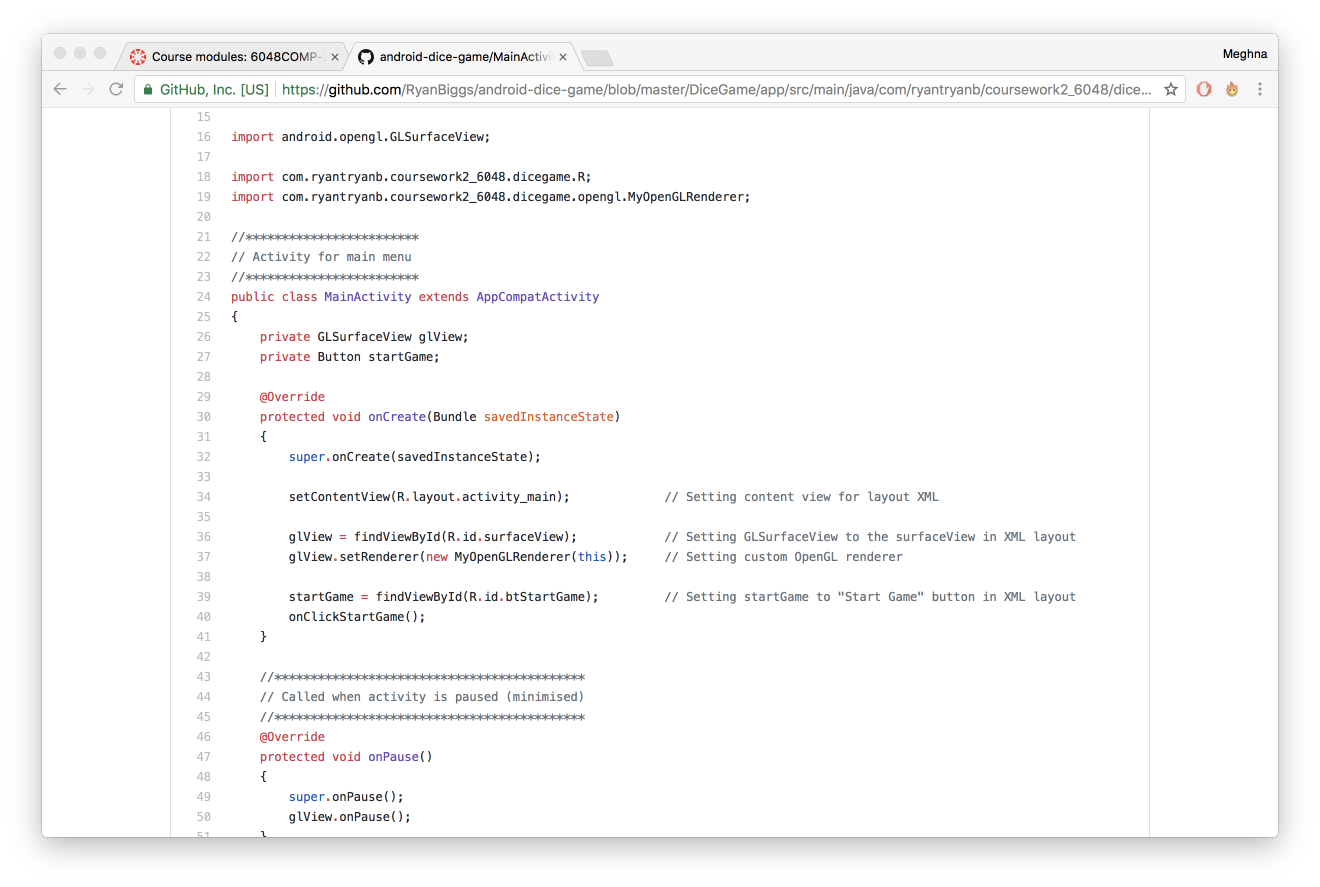
**Development Activity Report**

Porting is a process of adapting software. It allows us to adapt the software so it can run on a different platform that the software was not intended to be used on. In this coursework, we will be porting the existing dice game, which was developed to be used on Windows NT operating systems using .NET Framework, for Android Smart Phones.

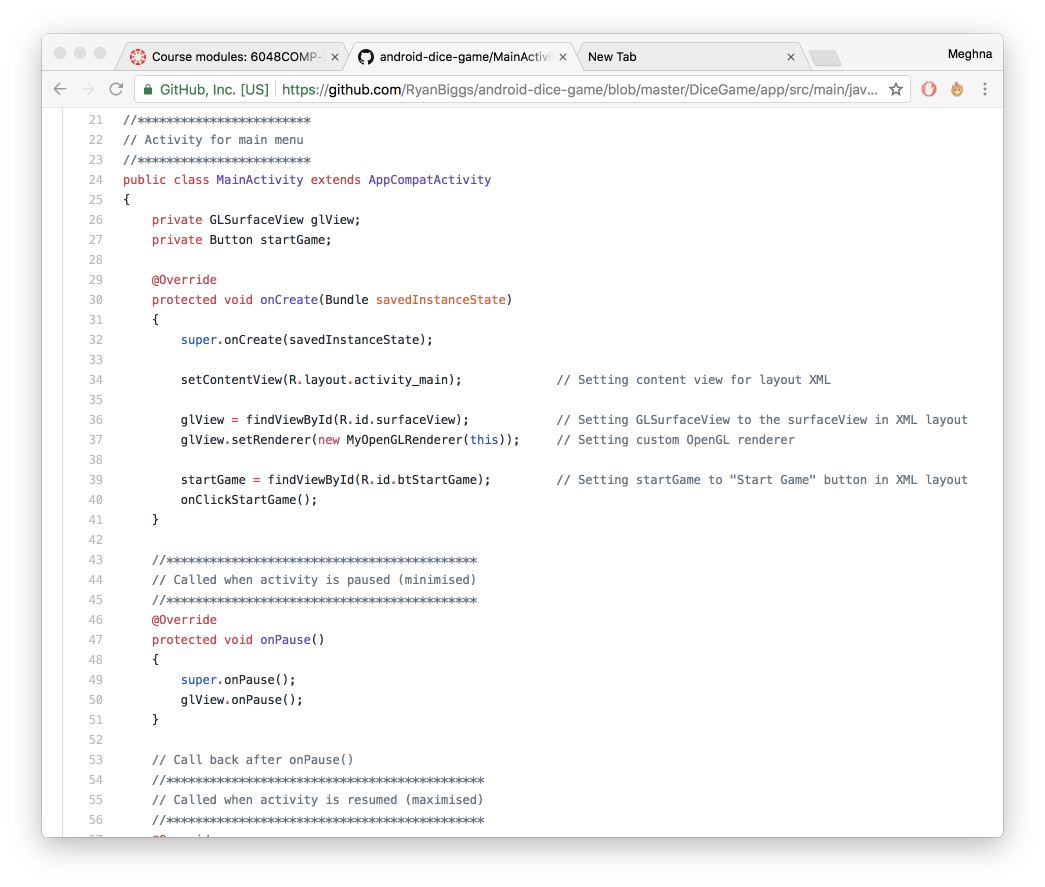
Firstly, we examined the Graphical User Interface as this is what the user would be able to see. For the main activity, we used OpenGL ES. OpenGL is a cross platform graphics API for high performance 2D and 3D graphics and is supported by Android. We created our own OpenGL renderer class which allowed us to override the default class so we could achieve more detailed customisation when it came to things like display sizes and aspect ratios for the application.

(Look at the screenshot below)



We encountered a problem when it came to the OpenGL graphics to work when it came to the main activity. After doing some research on the internet and using the developer.android.com website, we managed to find out that surfaceView was the issue that was not working with the main activity menu. We needed to set GLSurfaceView to the surfaceView in the XML layout to display correctly. After reading the documentation on surfaceView, we added the appropriate code which we needed to the code and then we managed to get the main activity menu working fine.

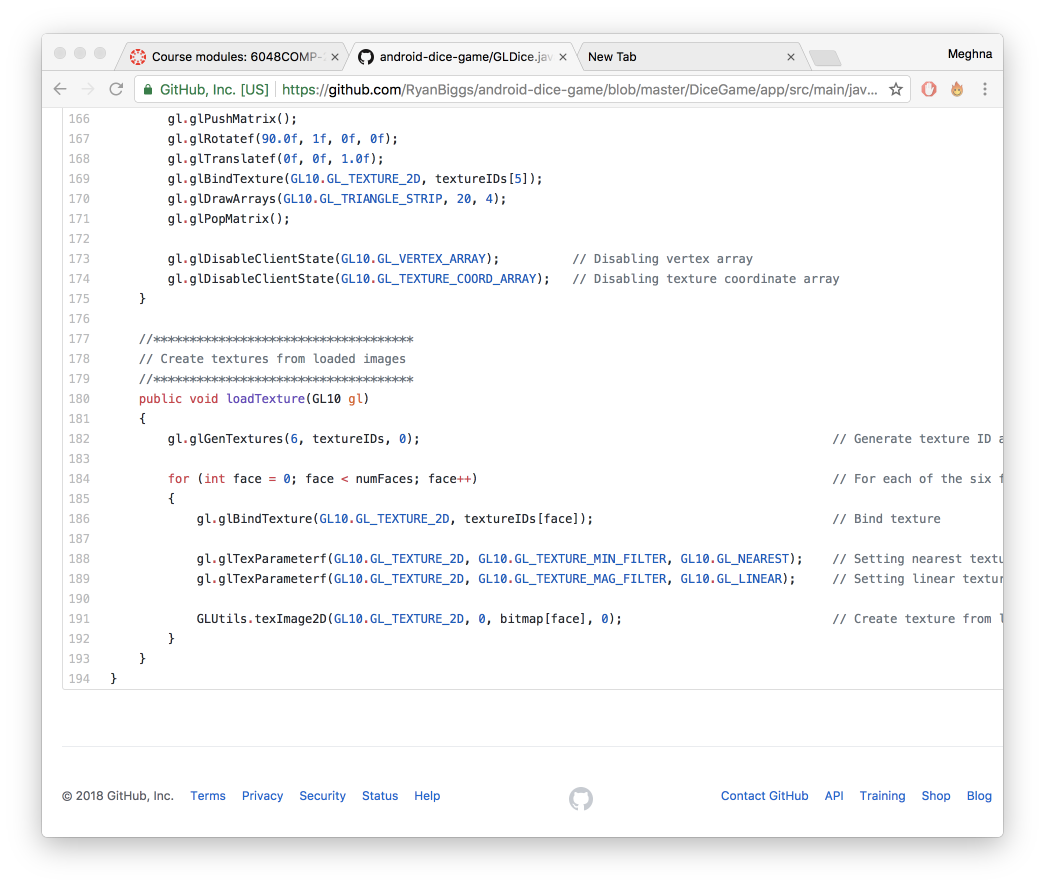
(See screenshot below)



Once our customer renderer class was set up, we made a dice class which holds all functionality for each of the die.

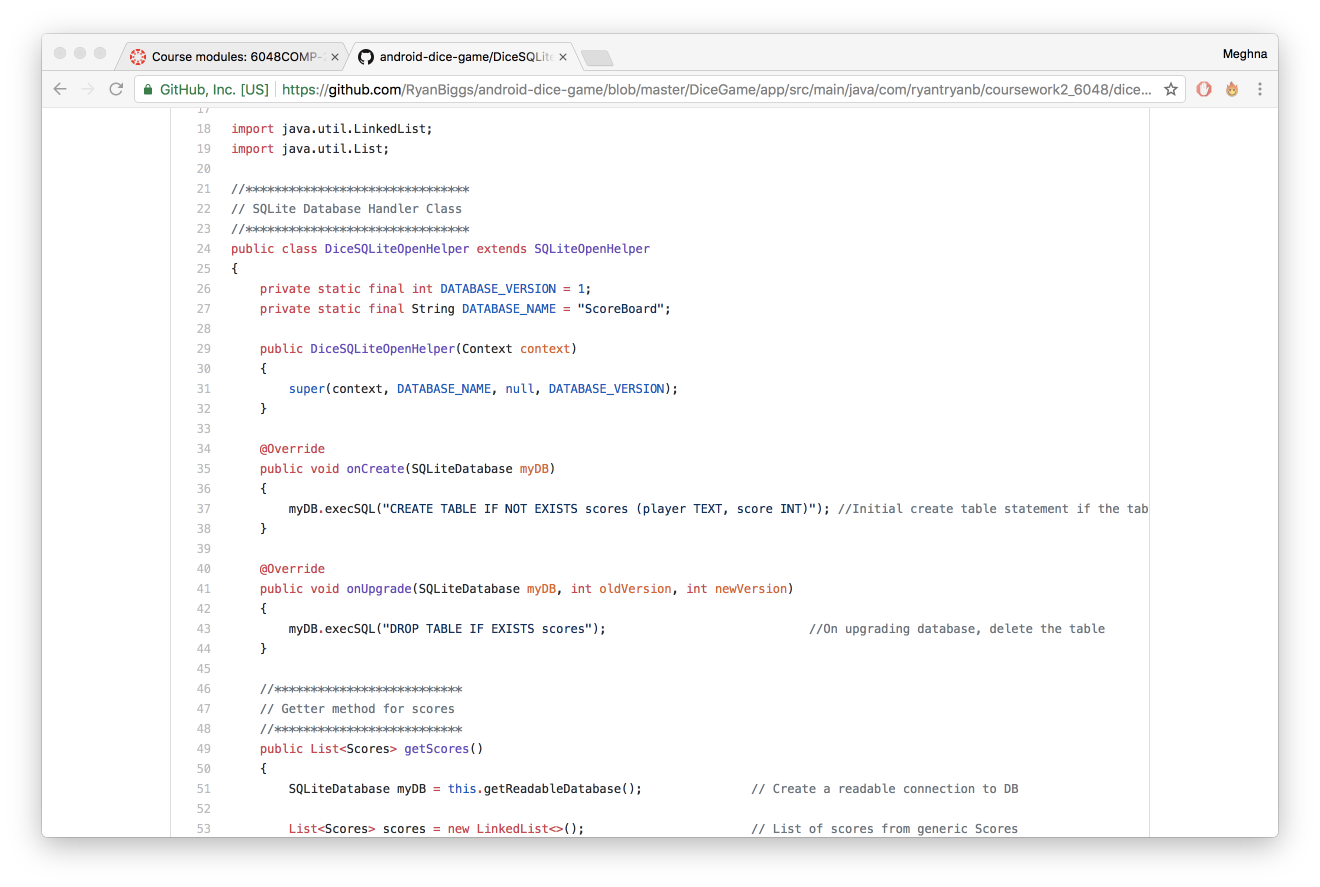
The OpenGL renderer sets up the rendering side of this and the dice class controls and sets up all the drawing and texturing of the dice. This is where we came across our first problem; we initially could not get the texture of the dice working however using developer.android.com helped us realise that we forgot to write a line of code which was crucial as it made the texture on the dice work.

(See screenshot below)



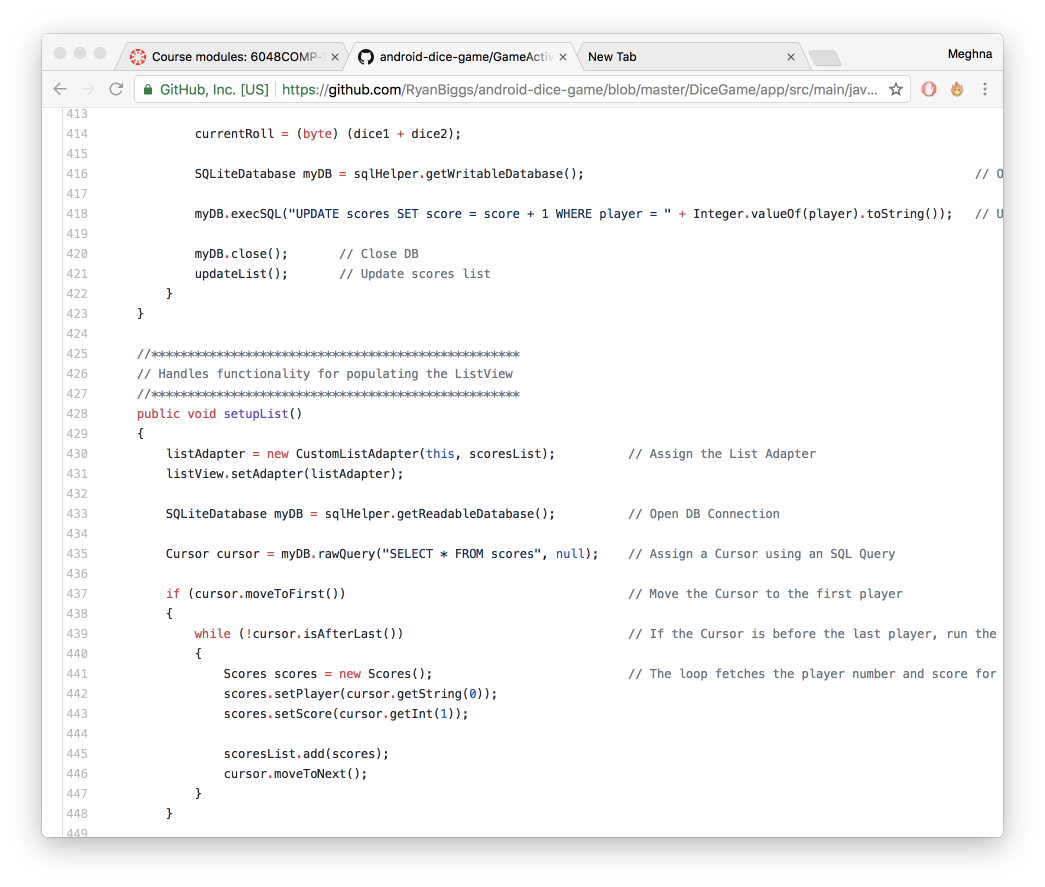
We used SQLite for the database. This is because it keeps track of the dice roll score for each player and stores them into the database so they can be accessed during the game.

(See screenshot below)



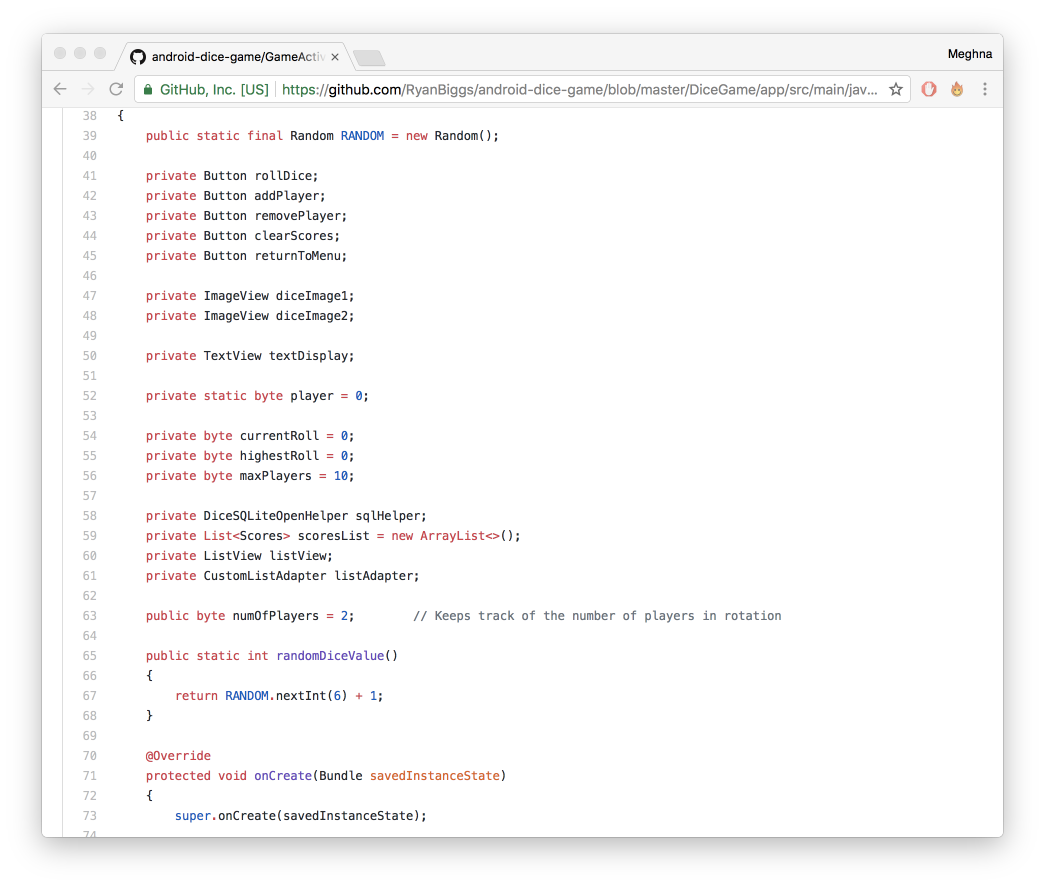
We came across a problem when making the score list update when one of the players won. We had trouble with the notifydatasetchanged(). So we tried to reuse the setup list method to update the list view however it wouldn’t work as the setup list method didn’t work for both. Instead we had to make two separate methods to handle the set up functionality and the update functionality. The update functionality is essentially a duplicate of the setup list we initially tried to use, but didn’t work. We found that having two methods, although it meant more code, worked better to update the score list when a player had won.

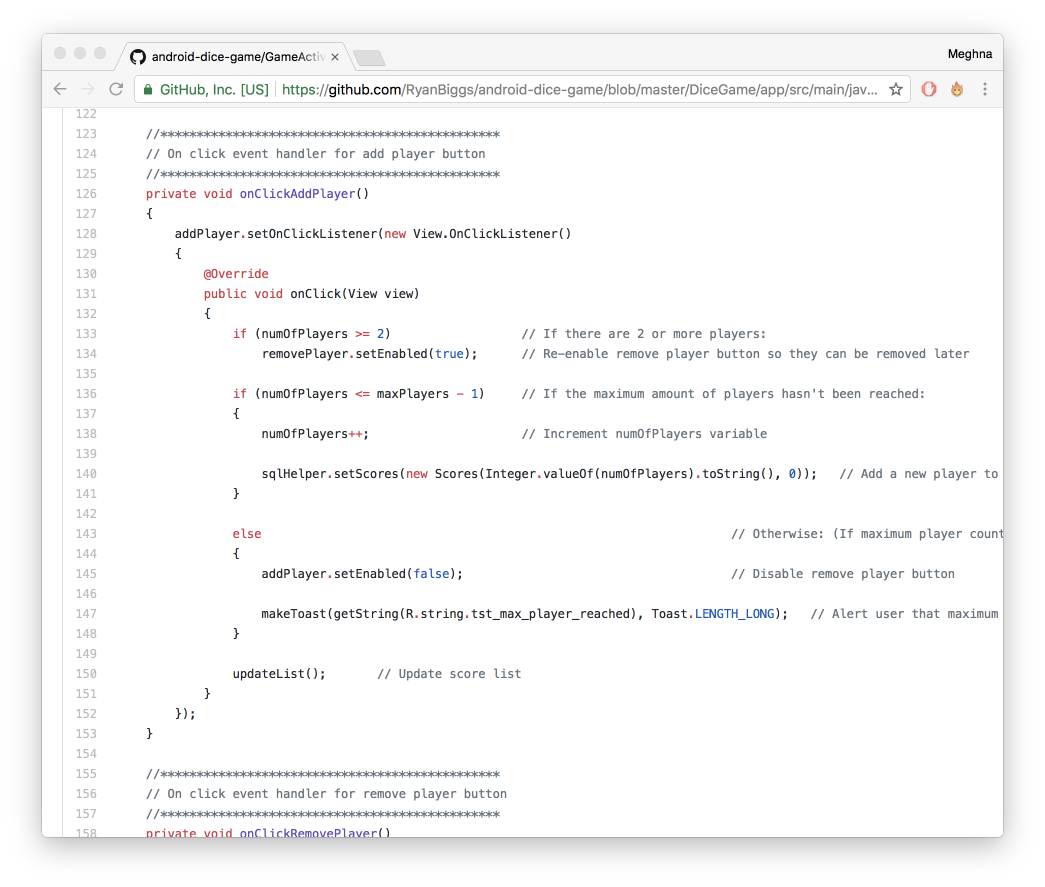
(See screenshot below)

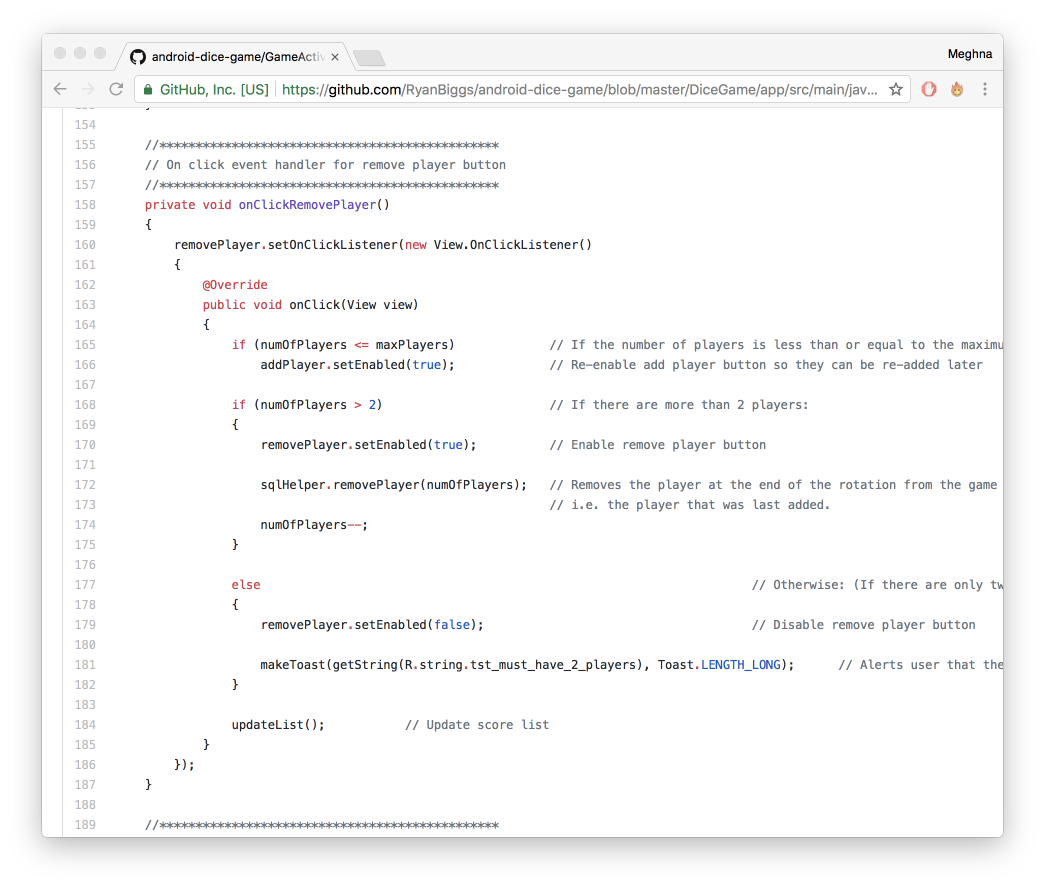


When developing, we added some extra methods to make our dice game even better. They are listed below with screenshots of the code to show how we implemented them.

We added the ability to add and remove players with a 10 player limit compared to the original application which wanted a fixed number of two players:







we also added localisation for four different languages as well as English to the application. This means if the devices language is German, Spanish, French or Japanese, then the whole application will be displayed in that language.

