# AC22005 - Computer Systems 2B- Grid Game Assignment

**Participants**: David Topping, Caleb Harmon, Stacy Achieng Onyango

**Group**: 7

**Game**: Connect 4

**Word Count**:

## Our Approach to The Problem:

Initially we held a meeting on MS Teams in which we came to the decision of making Connect 4. Our first step was to look online at previous examples of what people had done to base our ideas off. We drew out a simple menu and UI for the game to give us something to aim towards (see final page). Whilst we did not stick entirely with this design it gave us a rough idea of the layout.

As a group we decided to use GitHub to share our code and for version control, we then delegated the tasks as evenly as we could according to ability. David took the task of creating the GUI and creating the event handler functions. Caleb took the responsibility of the connect 4 checking algorithm and the computer vs player. Stacy was allocated the task of creating the menu. David and Caleb were able to work together on a lot of the tasks which helped a lot.

We individually planned our own sections and got to work.

## Difficulties We Encountered

Initially we had a 7x5 array of labels with a 7x1 array of buttons as the top row (these would initiate the event handlers). However, after some user testing, we realised this wasn’t overly intuitive as most people went to click on the labels to place a piece. We got rid of the array of buttons and changed the design to a 7x6 array of labels. We then attached event handlers to each of the labels so they could be clicked. We encountered a few issues with this as it caused several knock-on effects in our program as functions were too dependent on each other. This had a positive effect on the program as we tidied up our code/functions considerably. Once we got it working it functioned a lot more fluidly.

Our four in a row verification algorithm was difficult to compute as we ran in to many ‘array out of bounds errors’, we did manage to finally get this working but we did minimal testing on it which in hindsight was a big mistake (being the most important algorithm). Towards the end of the assignment, we noticed there was false wins, the ascending diagonal check was setting off a false positive. This was due to a small logical error but was very difficult to find. After this error we did extensive testing for all scenarios that could come into play, this led to a much more robust algorithm.

The last thing we encountered problems with was creating a new form for the menu and then calling the game from this form. The only reason this was so difficult was because we had very limited experience with streams and GUIs. With some research we managed to get this working perfectly.

## Possible Future Enhancements to Our Game

Ideally, we would include different difficulties for the computer vs player option as currently it is not overly difficult to win against the computer.

We would also like offer the option to save the game and load the board back into the same state (we would do this by printing the contents of the board to a text file).

It would also be good to have a leader board where the user could enter their name and it would save the high scores to a text file.

## Graphical user interface, application Description automatically generated