**SOFT050 Project**

**Texas Hold’em Poker**

By Joseph Kellaway

Tutors:

Torbjørn Dahl

Liz Stewart

University of Plymouth

24/04/2015

|  |  |  |
| --- | --- | --- |
| 1 | Introduction | 2 |
| 2 | Technology Review | 2 |
| 3 | Design | 3 |
| 4 | Implementation | 5 |
| 5 | Evaluation | 9 |
| 6 | Reflection |  |

**Introduction**

For my project I chose to recreate the classic card game Texas Hold’em poker. This is probably the most well-known form of poker, because it is the type most commonly seen in pop culture and casinos. Texas Hold’em involves several instances where cards are dealt either individually to each player or to a communal pool of cards with players taking turns to bet on how strong they think their position is.

I am a regular poker player with good knowledge on the rules of the game. I am enthusiastic about playing the game and have thoroughly enjoyed the challenge of designing and attempting to recreate it for my project. I felt that it would give me extra motivation to improve my programming knowledge by basing the project on something that I understand and enjoy to begin with.

**Technology Review**

|  |  |
| --- | --- |
| **Technologies Used** | **Alternatives** |
| My computers | Labs  Laptop or tablet |
| Visual Studio 2012 | Notepad++  VBSEdit |
| HTML  VBScript  JavaScript | CSS  C#  Python |
| Internet Explorer 11 | None |

As far as hardware was concerned there wasn’t much of a choice – a student’s finances are limited and making use of computers that I own makes complete sense. I often find myself distracted when using the computers on campus so I tended to work from home.

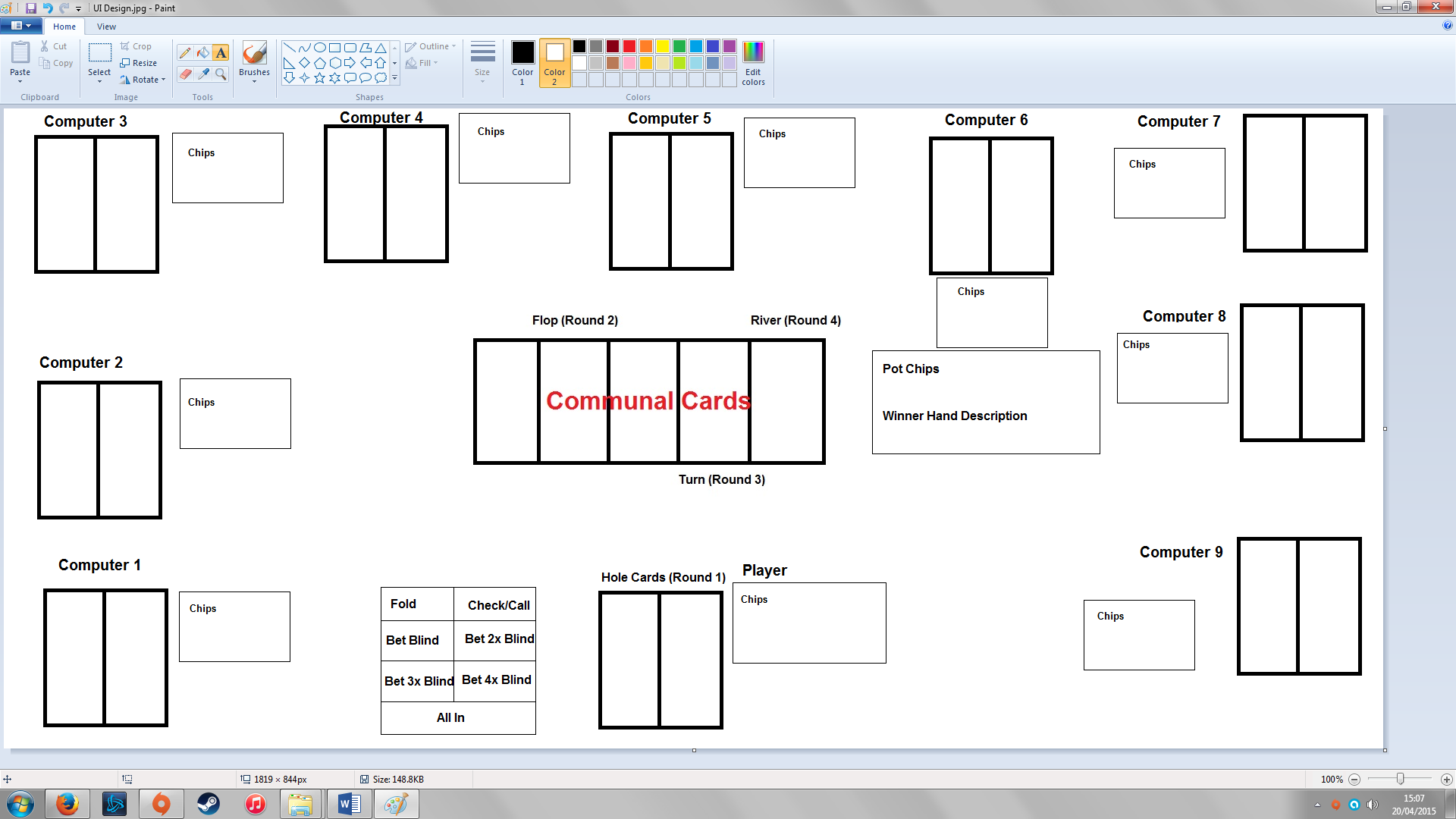
A copy of Visual Studio 2012 has been supplied to me by the university so I took full advantage of it and used my familiarity with it to jump straight into the project without having to learn the interface of a new program. I am aware of several alternatives that are available for little or no cost but since I already had Visual Studio installed on my machine I saw no compelling reason to switch.

In the end I decided to stick to languages that I have experience of using. Obviously the program only running in Internet Explorer would limit its value as a product, the creation of the product still gave me the same learning experience as any other language.

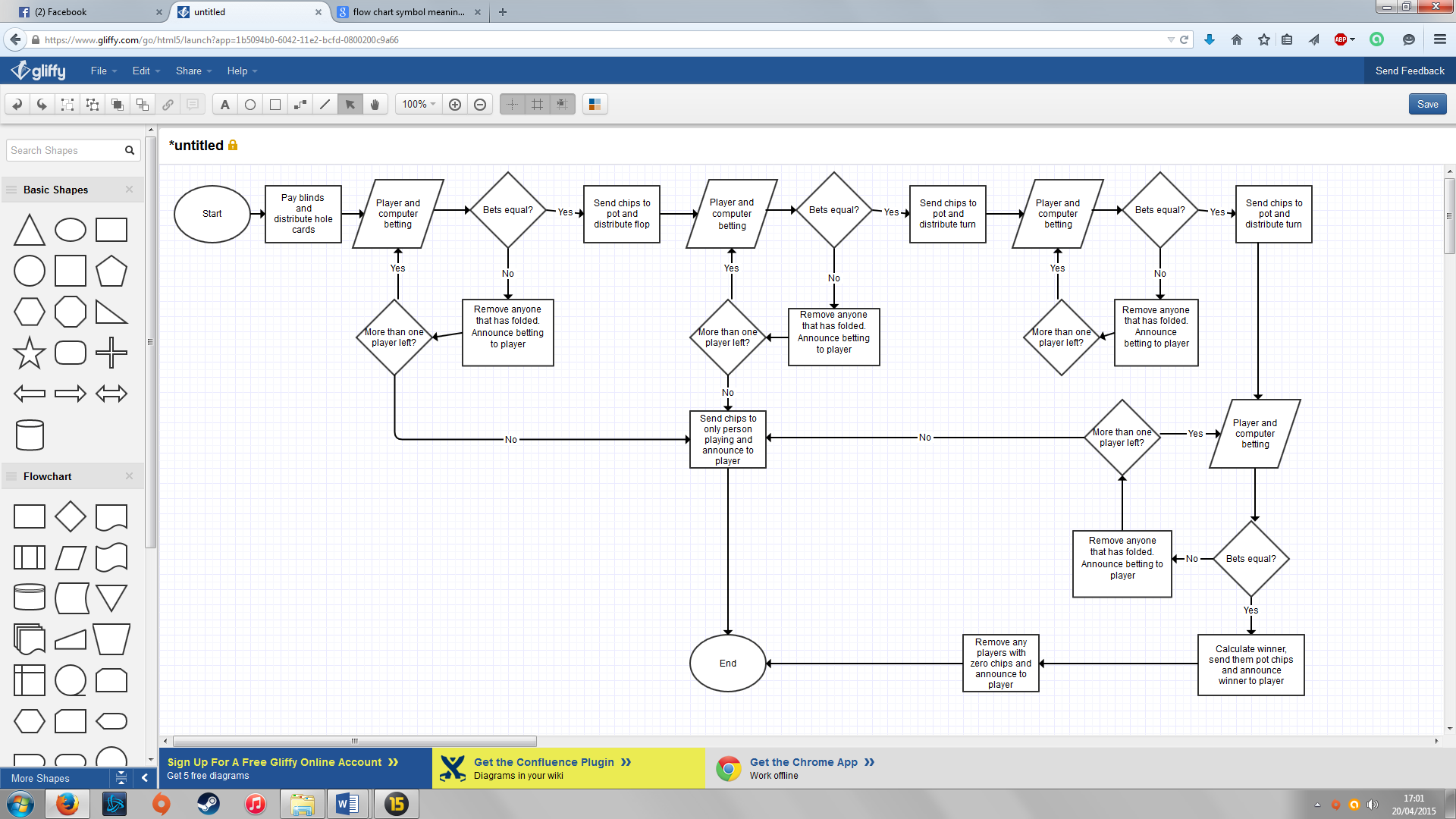
Finally, to test my game Internet Explorer was required – the use of VBScript means that the program would fail to work on anything else.

**Design**

This is how I wanted the end program to look – very much like the way that most poker games are set up because it is how a poker table is set up. Cards are represented by images, buttons allow the user to check, bet or fold and information is fed back to the user in paragraphs. A pop-up message will display what actions have been taken since the user last acted.



Here is a flow diagram of the way that I wanted my poker game to work.



I broke this design down into key stages that made it easier to plan and code rather than become overwhelmed by a huge amount of directions to explore all at once:



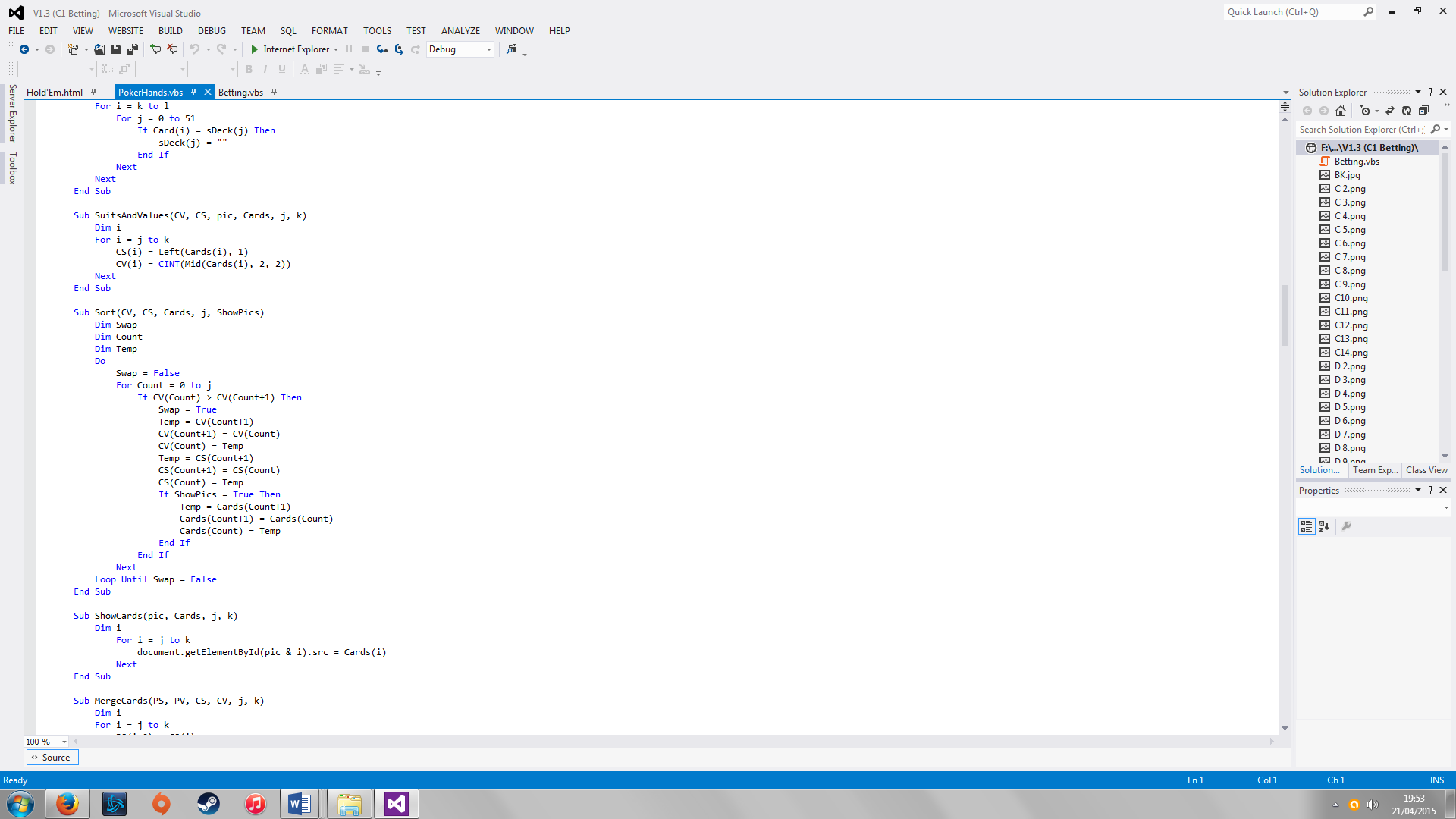
**Implementation**

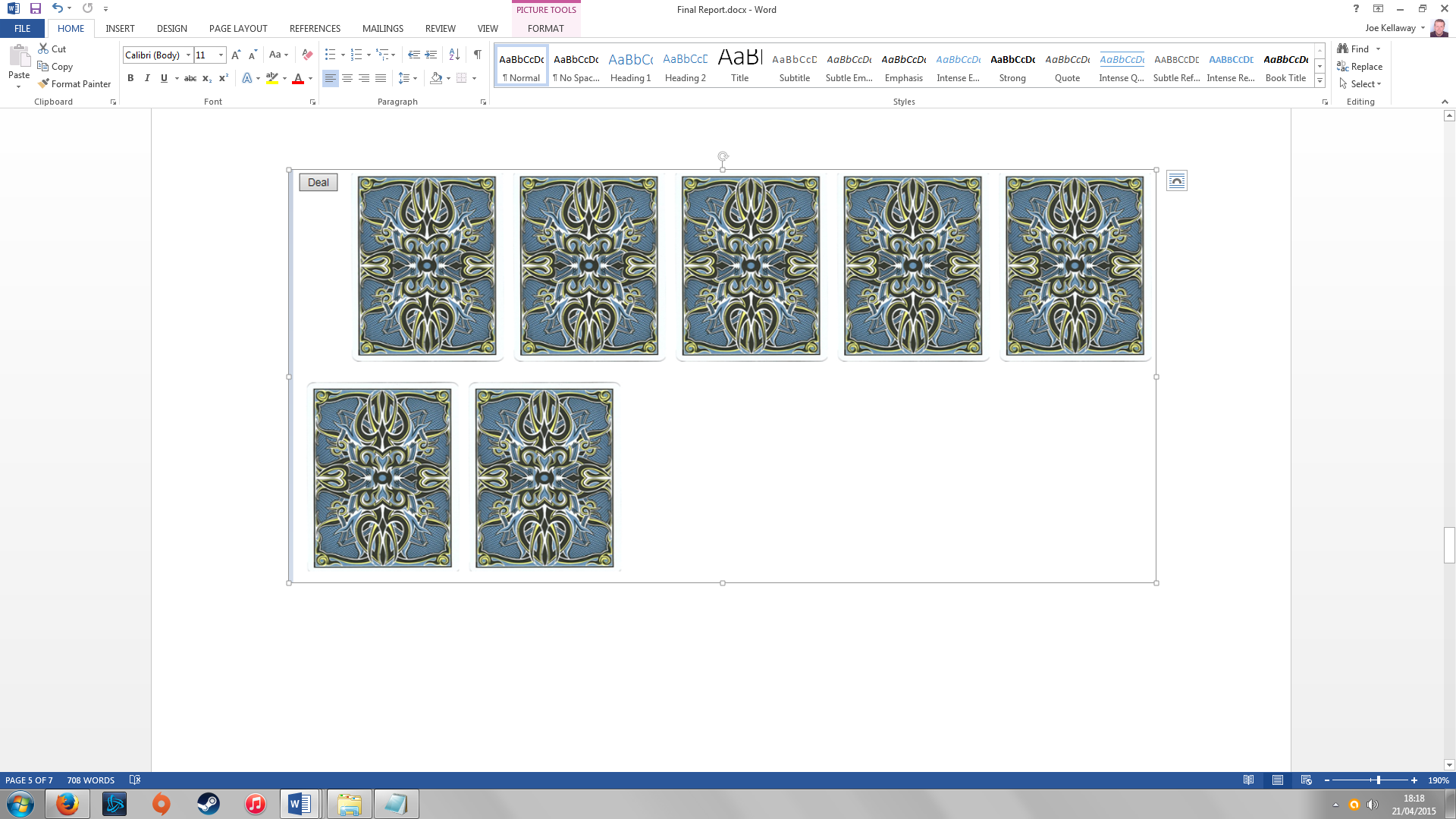
Before opening Visual Studio to begin, I wrote how to define each hand as if in a logic problem. Upon doing this I realised certain combinations could be done with brute force or the card order didn’t matter (e.g. a flush just requires five identical suits). However, when checking for a straight there were 120 combinations to assess; so a sorting algorithm was required. This naturally became my first priority.



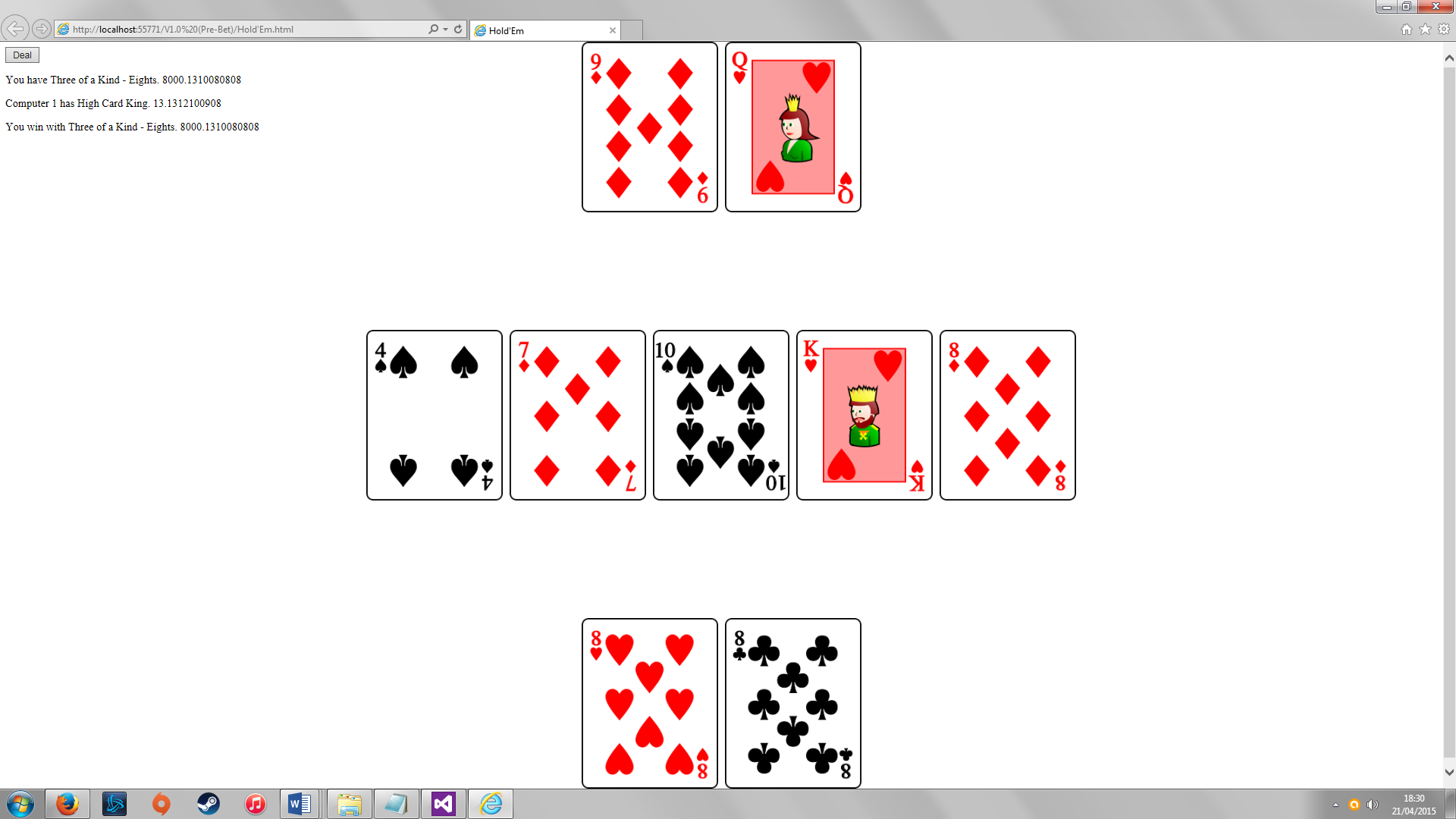
I set up a prototype that displayed 5 cards (as above) using random numbers to select pictures from an array to deal as a hand. Before displaying the pictures however, I would extract the value and suit from the image filename and use a bubble sort (see below) to arrange the images, suits and values from low to high based on the card value before displaying the cards, assessing the hand and informing the user of what hand it is. As you can see in the bottom of the picture I created test buttons rather than wait for luck.

In my bubble sort I first calculated the maximum number of sorts that would be required, but then I improved it by using a Boolean that stated whether there was a swap made during that loop and only continue if there was, since the first pass that no swap is needed indicates that they are fully sorted.

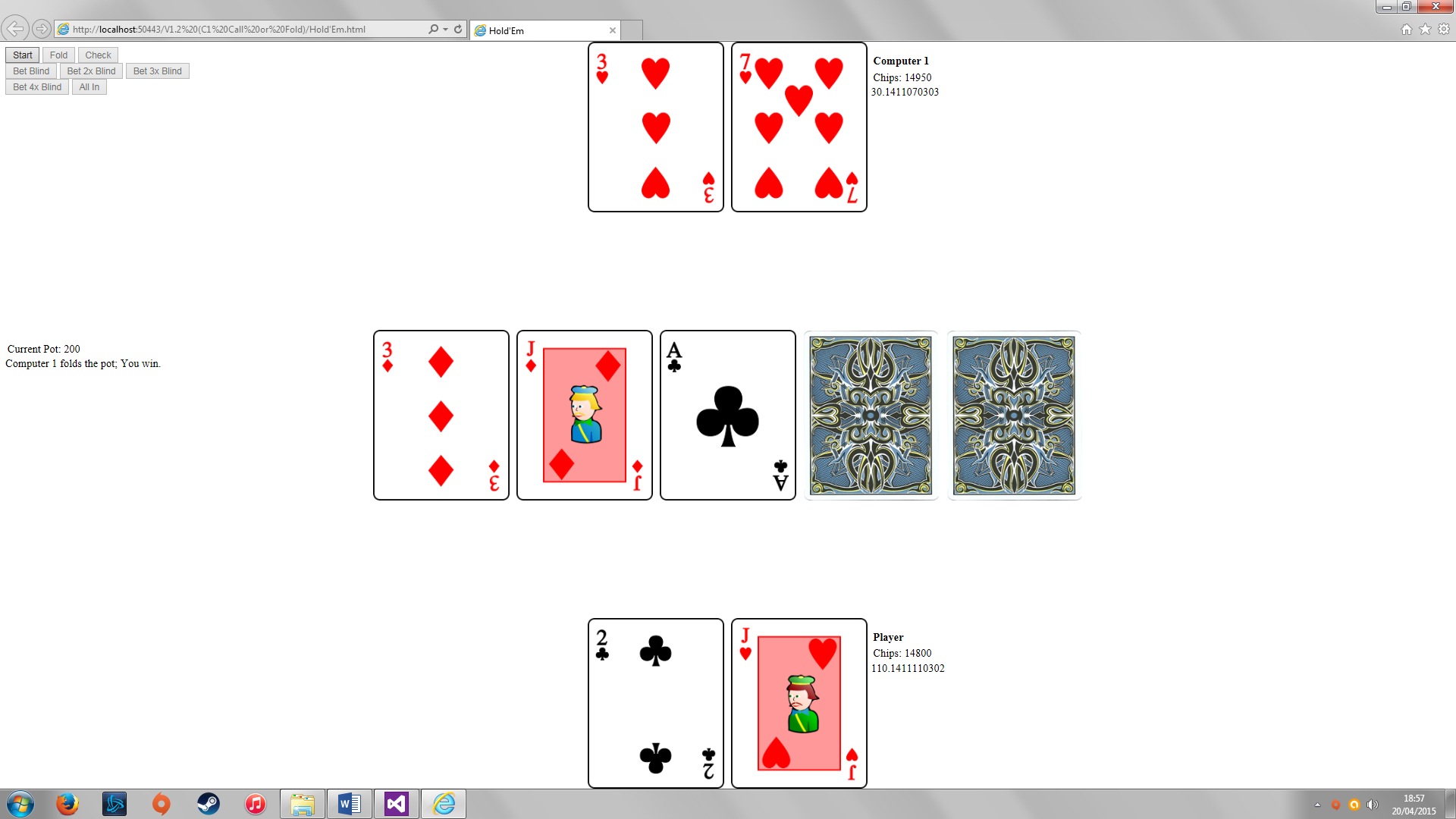




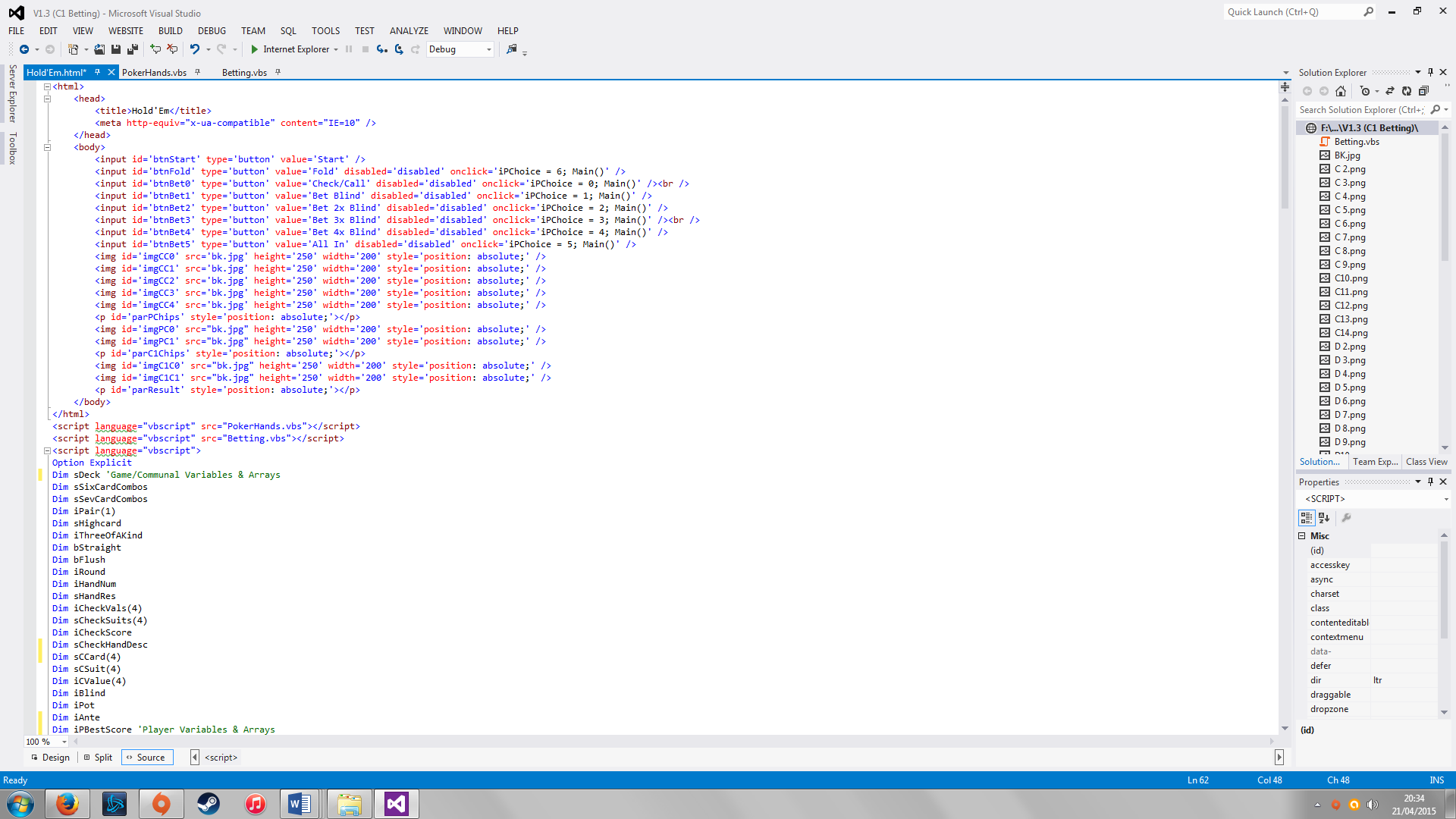
I expanded to 7 cards (as above) which presented the challenge of checking through 21 different hand combinations to see which would be the best. I created an array of the different combinations and cycled through each combination of cards and checked what hand it had. I generated a scoring system to value each hand against the other based on poker rules and set up a loop that compares the best hand so far and the current hand being checked. Finally, I split the game up into rounds to prepare for betting.

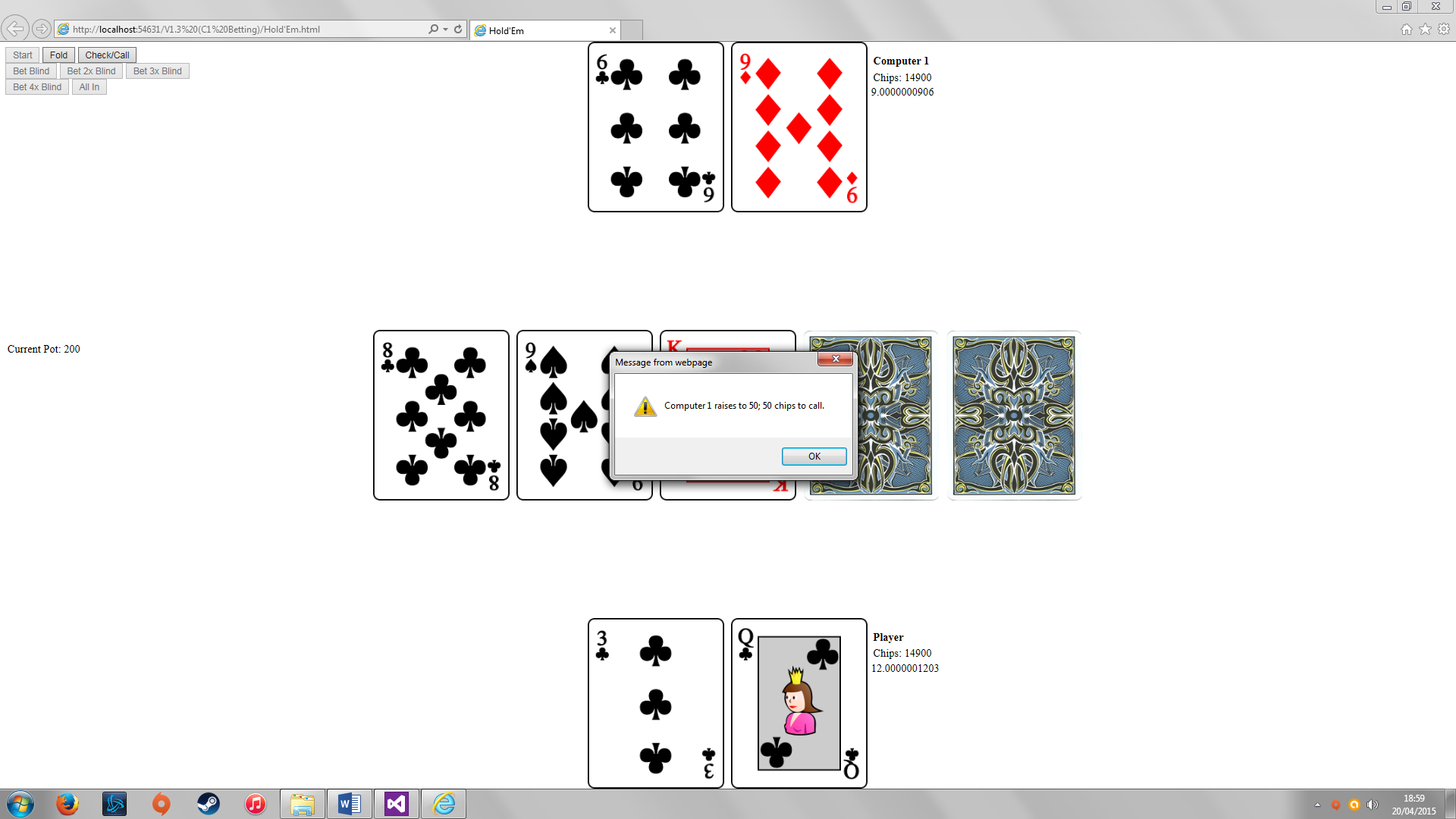


I then added the computer player with the same principal as the previous version, except I was now checking what two different hands have and then comparing the best of each against each other to calculate the winner. I also positioned the images separately so that it was clear that they represented opponents.



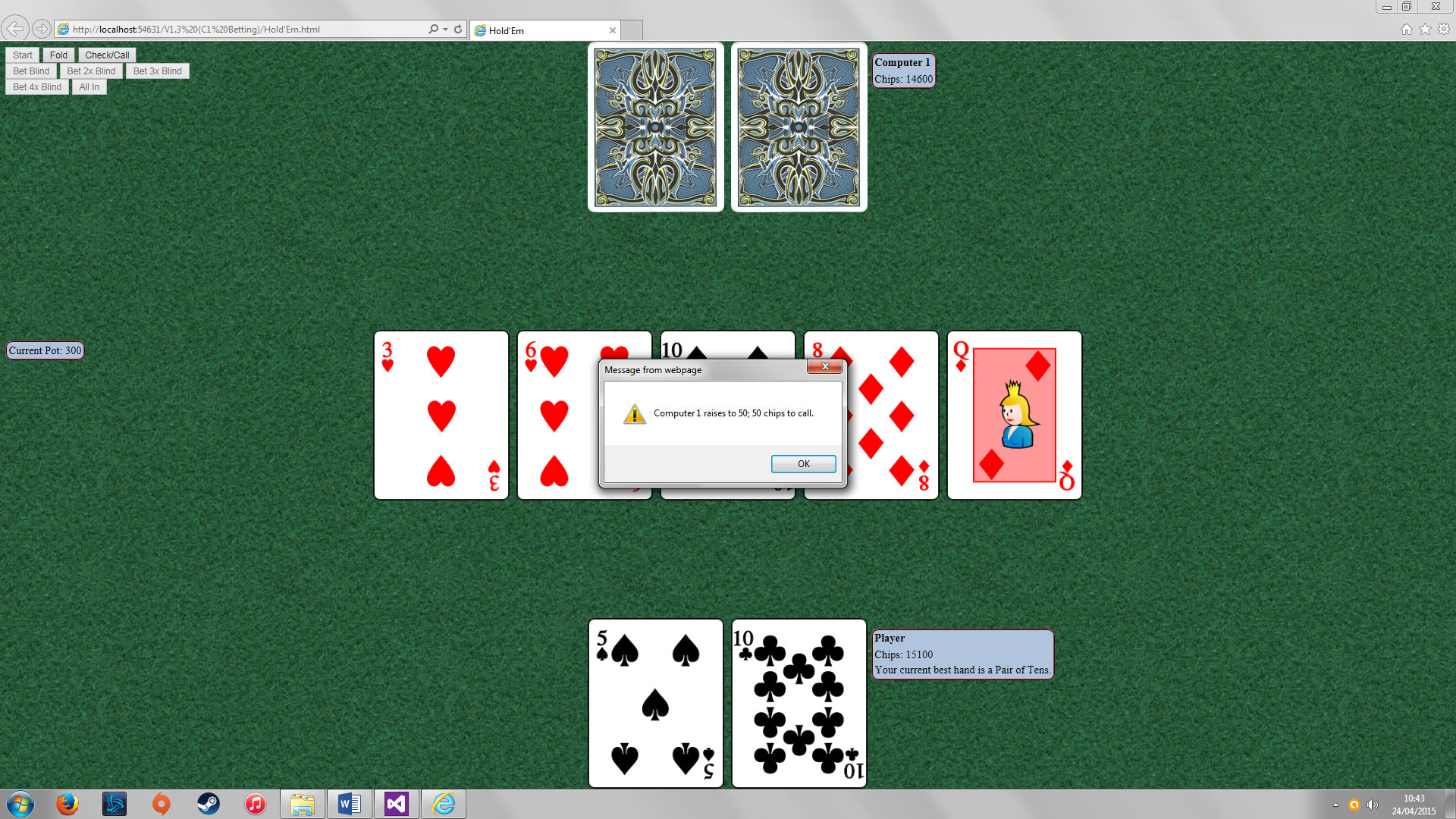
Next I set up variables to represent a blind bet value, ante, chips, bets and the pot of chips that the current hand is worth and extra buttons to allow the user to choose to bet a multiple of the blind value. I started by forcing the computer to call or fold based on parameters that I set to judge the score of their best hand based on what round that is currently being played. Knowing that I wanted the same subroutine to play irrespective of which button was pressed, I required a tiny amount of JavaScript within the HTML of the buttons (shown below). The “onclick” function gives a value to a variable that dictates different betting values, but the code works the same way irrespective of what bet is made and so the same subroutine can be called.





The next part of the project that I wanted to tackle was to enable the computer to bet based primarily on their best hand, but also to react to the way that the user had already bet by calling, folding or re-raising. I set up a second set of parameters that would bet in the same way that the user does – it passes a number that multiplies by the blind value to calculate the computer bet. I added an alert that informed the user of the computer’s action and decided to disable the player’s ability to re-raise because when testing I found that the human player could create a betting loop that effectively made the computer player go all in on a relatively weak hand.

I then tackled the end of the game. When either player hits zero chips an alert warns the player that they have won or lost and clicking “Start” resets the game to its original state.



With time growing short I decided that adding more functionality was probably a bad idea due to the extensive testing required at each improvement. I always wanted the game to feel like you were actually playing at a poker table so I added some aesthetic changes to the interface using Google image searches, Paint and changed some properties to paragraphs. I also did some final testing.

**Evaluation**

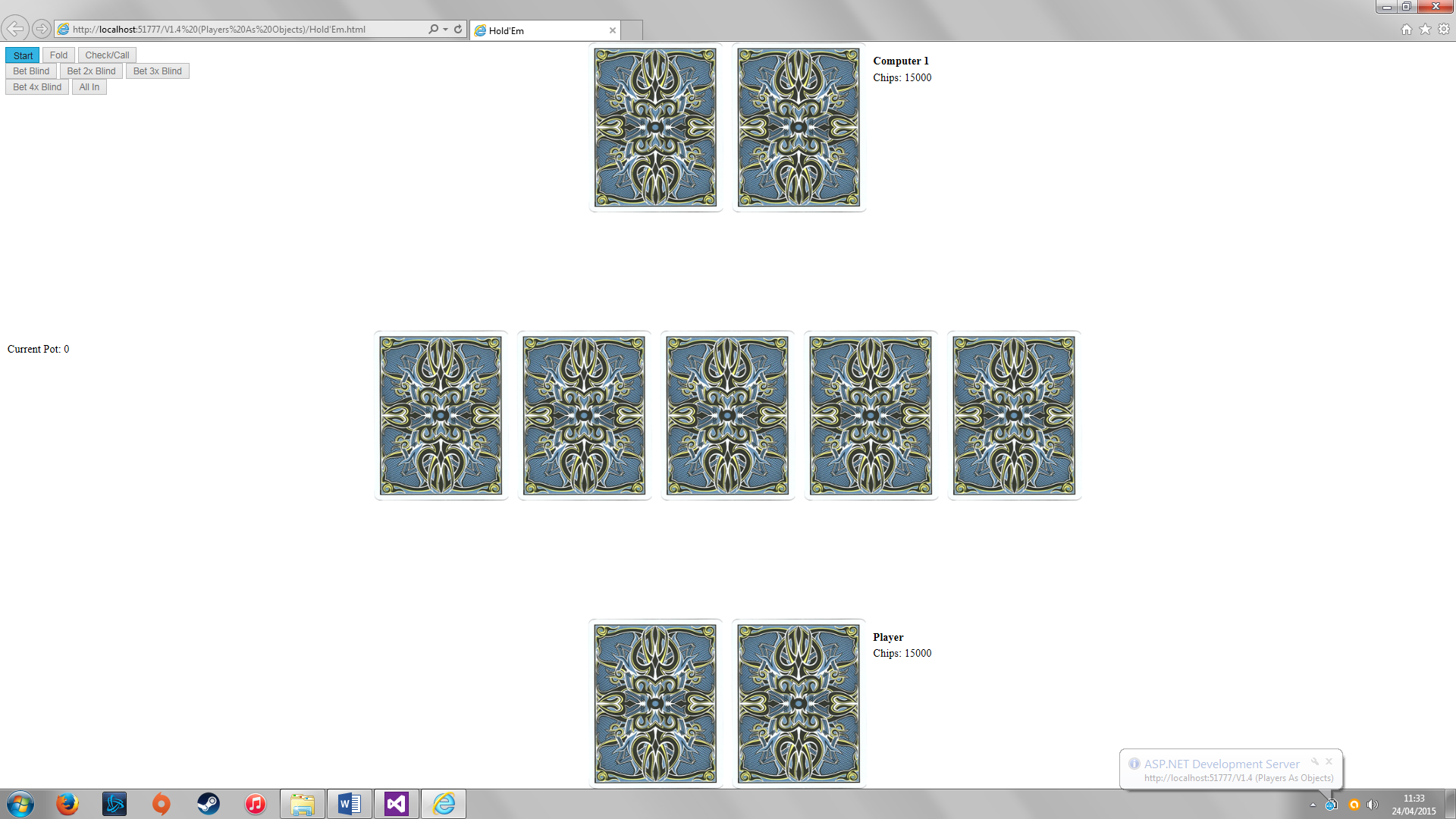
Early on in the project it was easy to create test buttons that enforced certain scenarios that needed checking (the first implementation image shows test buttons for checking hand detection). But as I got further into the project (particularly with computer betting) it became harder to do this and changing variables became a much easier approach to take (for example, the parameters required for each computer reaction).

On top of testing that specific scenarios were working I felt that testing the game as a whole was essential. I have played numerous hours of the game throughout its creation to check for unexpected events occurring. I have encountered and fixed quite a few issues, but I know how the game should work and that could make it harder to see where it isn’t working. This gave me the need to enlist family and friends as testers, though none of them actually returned to me with any actual bugs.

At present there are no bugs within the game that I have found and not subsequently fixed which is obviously delightful. However, I don’t mean to say that there is no chance of there being any. More testing would have more chance of covering every scenario and more chance of unearthing undiscovered issues.

I also found that more experienced poker players learned how to play the computer too well making the game too easy. This could be resolved by adding a random element to the computer’s betting parameters (e.g. +/- 25% based on a randomly rolled number each hand) and by adding the predictive betting function that I had hoped to implement. Adding extra computer players to the game would probably help to solve this issue as well since there would be a higher chance that an opponent has a good hand and wouldn’t end up getting bluffed out of their bet.

The primary thing I would like to have included but didn’t was object oriented programming. It would have allowed me to make my code more efficient and cut down on a lot of variables, particularly for adding extra players, but I found myself unable to get OOP to work in VBScript. Having previously used it in VB.Net I was fairly confident that once I learned a few minor syntax changes it would work in very much the same manner. However, I encountered issues that I found myself unable to resolve. I was able to get player classes set up and I was able to populate the properties of the class, but when I tried to use my sorting algorithm on them then the program froze as if it was caught in an infinite loop (see below). My normal source of help with syntax ([www.w3schools.com](http://www.w3schools.com)) has nothing on OOP in VBScript and Google searches mainly returned forum discussions about how limited VBScript’s OOP is.



Other than this, I’d like to have worked my way further through my list of key stages in the design. Creating my own images for the project would have also been a nice touch.

**Reflection**

I have made numerous changes to the schedule of my project. In the first few weeks I had little other commitments to maintain so I managed to get ahead of my targets, but as other activities started to reduce my spare time I found less time to spend on this coursework and found myself falling behind schedule (particularly when encountering bugs and issues with the OOP). Downscaling the project was an eventual necessity.

Out of everything in my project, I am particularly proud of making a bubble sort from scratch. I am also happy with the logic behind assessing hands and dealing the cards – I have it set up so that it removes dealt cards from the deck array which really cut down my comparisons for dealing in the following rounds.

The lack of OOP is the main part that really didn’t go well with my project. I appreciated at the start that VBScript is a language that most people view as very limited and so I reluctantly chose it over C# due to my limited programming experience. Obviously doing less than I intended isn’t great either, but I have a bigger appreciation for the amount of time that testing takes.

I have confirmed that testing and bug fixing can often take longer than the creation of the project itself. When switching from five to seven cards I managed to break a lot of things within my code which has taught me that a backup of the previous version is always a necessity.

For my next project I will stay clear of VBScript. Also when creating my proposal and plan I will think more about the category of each stage and all of its components (whether it really is a “must” or whether it could be downgrade to a “should”).