1. Analysis

Problem:

Young people looking to play poker often have a lot of roadblocks and hurdles getting in their way. Whether this is a lack of interest from their friends, or more recently lockdown, meaning they can’t physically sit down and play with people or online solutions requiring they are above a certain age or having predatory monetisation schemes designed to take advantage of young people and designed to addict people using subconscious techniques. This leaves a giant gap in the market for accessible and non-predatory solutions that allow young people to practice and play poker in a safe environment.

How I can help by using computing:

The most obvious solution to not being able to sit down and do something is to use the internet to find a way. Without a custom solution such as mine it is impossible to play poker online as you need to play using one deck of cards and passing chips between each other. However, by using a computer and the internet you can easily have people sending data between each other with negligible delay, this is almost impossible with any other solution. Therefor we need to use computers and the internet so that data can be sent. Furthermore, due to the wide availability of the internet and the ability to store data on a central server people can log in from anywhere and pick up where they left off and see all of their data at any time without needing to bring any physical media with them. This is another task that is impossible for human’s as they would need to physically bring some method of recording data and the physical data with them to wherever they are playing from.

In addition to this the storing and tracking of data is another difficult task for people to do manually as they have to remember how and what they have done and manually plot graphs and make calculations. This wastes time that people want to be playing the game. On the other hand, this task is extremely well suited to a computer as they are designed to be able to handle data as they can easily take inputs then store, process data and display data automatically all the while unimportant information like how these numbers are generated is abstracted away to create a better user experience and save time.

Furthermore, by creating my own solution from the ground up using computational methods I can control the monetisation and the safety of the environment to ensure that no one is being taken advantage of and the game is more appropriate for all ages. This can be done by limiting the interactions between players to purely gameplay based or, if a game chat is included, to ensure it censors inappropriate messages. This isn’t possible if you are playing with strangers in real life as you can’t predict or control their behaviour or how they communicate however computers can easily parse messages and check for and violations before they are transmitted.

Stakeholders and the target demographic:

The product is targeted at anyone who wants to play poker online in a safe environment, but primarily under 18s who are unable to access many other due to restrictions in place or just because their parents/carers are worried about the dodge monetisation methods of some commercially available solutions. Despite this being the target demographic it will be accessible and appropriate for anyone from ages 12 and up to use to practice their skills without real currency on the line. It is aimed at new and experienced players alike as it will have information on how to play and what the rules are accessible if someone wants that information. It will also be aimed at both casual and competitive players, the open table format opposed to a tournament format targets it at casual players however the extensive statistic and data analysis features means more serious players still have a reason to use the platform. Because of this I will be interviewing and testing the product on a range of people under the age 18 with varying levels of experience with poker.

Interviews:

## Person 1 – Felix Judd, 17 years old:

Q: How would you describe your level of experience/interest in poker

A: I wouldn’t say I’m experienced; I played casually with my family a handful of times a few years ago. I am definitely interested in learning more about how to play though before I even consider gambling actual money

Q: Have you ever used an online poker platform before? If so, what were your thoughts

A: No, never used anything like that before

Q: Why not?

A: I didn’t realise games that weren’t for real money existed

Q: what features would you look for in a poker game?

A: As someone who doesn’t know much about poker a built-in hand ranking and maybe a tutorial or some hints or something just so I don’t feel totally out of depth, also having it be simple so I can understand it quickly

Q: How interested are you in the analytical side of the game?

A: Not very interested to be honest, I guess its interesting to look at but I’m not really sure what I’d do with it

## Person 2 – Louis Gosling, 17 years old:

Q: How would you describe your level of experience/interest in poker

A: I don’t play a huge amount but a few times a year my family plays a fairly serious game and I have played a few free online games and watched a few videos and all that. So not super interested but definitely not a beginner and want to play more and get good again before uni and stuff like that

Q: what features did you like in the online version you played?

A: I liked that it had an overriding currency and it was individual for each game, it gave it a feel of continuity and meant I could easily see my successes as it is reflected in my bankroll graph

Q: So building on that, how interested are you in these statistics and analysis features

A: Very, its something you obviously don’t get in real life but its so nice to have. Especially when using it as a practice tool the ability to see the trends and mistakes I’m making is really useful

Q: What would you change about the version you used?

A: The ui kind of sucked, it felt very overwhelming to use and I basically only pressed 25% of the buttons so something more simplistic without all the extra unnecessary buttons and stuff would be great

## Person 3 – Katie Curtis 19 years old:

Q: How would you describe your level of experience/interest in poker

A: I have played really casually with my friends a few times, its fun but I’m very causally interested just as something to do

Q: Have you ever used an online poker platform before? If so, what were your thoughts

A: No never

Q: What features would you look for in an online poker game

A: the ability to play with my friends is all I’m really looking for, and something to tell me how good my cards are, normally I have to google that

Q: How interested are you in the analytical side of the game?

A: not at all, I’m just wanting to relax and have some fun with my friends because we aren’t really video game players and this would give us an opportunity to play together while we’re away from uni because of holidays or lockdowns or whatever

Conclusions from interviews:

There is a large variety of skill and interest levels around the game of poker and the ability to play it online with different people wanting different things. Both of the less experienced players wanted a ranking of the hands to be easily available and built in so this is definitely something I will aim for. The analysis section didn’t seem to be as a big a deal as I had hoped however it seems essential for getting more serious competitive players to take interest in a more basic solution. Maybe some idea of what should be expected/ is a good sign in the analytics would help people like Felix take more interest in these statistics and develop their game since the only reason he gave for not caring was a lack of understanding. The desire to play with friends means some way of identifying who is hosting a server is important overwise it would be hard to coordinate a game amongst friends, maybe the ability to have a private lobby with a password would help keep the game among friends. A simple ui also seems important to people as both Louis and Felix mentioned it, and one can only assume that a more casual player like Katie would also appreciate a simple and easy to use interface, especially around the server browser. This mostly reflects what I expected to be concluded from these interviews, but I had my priorities out of order initially, over valuing sections like statistics.

## Important points

* Clean ui
* Clean and easy to use assists for new players
* Statistics with explanations of what they mean for the player
* Named lobbies
* Low barrier of entry so new players don’t get overwhelmed

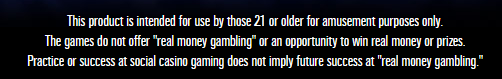
Research into pre-existing solutions:

## The official world series of poker game:

Cons:

* Game is “aimed at people over 21”
* A very busy and obnoxious user interface
* Focussed around the in-game store where you have ability to pay real money for their virtual currency.
* In game store offers an advantage to some players who have spent money.

Pros:

* In-depth tutorials and rule information
* Very in depth, if hard to find, statistics such as stats on when the player commonly folds.

Overall:

Not a bad solution with high production value and the ability to log on with several options as seen above. Overall though they clearly state they have different stakeholders for the product and are very heavily focused on monetisation. This is not something I am interested in as it will put off younger users and/or their parents. I will certainly use the statistics from this solution as an inspiration for my own statistics model and what to include. I will also aim to make a more simplistic user interface as I found this one hard to navigate and very claustrophobic.

## Pokerist:

Cons:

* As seen here the user interface puts a large focus on promoting in-game purchases
* Limited statistics, only game played and biggest win
* Use of overly sexualised characters showing a clear target demographic
* Due to being web based a long loading process before getting to the menu
* Hard to select the right number of chips
* Strategies designed to addict such as daily bonuses to encourage people playing every day
* The fold call and check button are relatively far from each other causing a sub optimal ux.

Pros:

* Once you close the ads it reveals a very clear and useful server list displaying relevant information such as players blinds and buy ins
* Intuitive hints that show when you have a hand which can be disabled in settings
* Use of accounts so information can be accessed from anywhere
* An in-game history so you can see the result of the previous hand
* Can preselect your next action in advance so it will complete automatically

Overall:

A better solution than the wsop but not without its flaws. The user interface was, other than the large advertisements covering up useful features, very clear and much less claustrophobic than the wsop game. This is likely due to its use of clear tiles and the ability to see the plain background behind it. The game also has a lot of social features which I found unnecessary as it detracted from the game itself.

## 247 free poker

Cons:

* No online component, purely against ai
* Almost no statistics at all (only largest win)
* No settings/preferences
* All data stored in cookies, can’t access from another device
* Doesn’t say if you have had a hand, no assists
* Could be clearer about rules
* Doesn’t say how much you are calling

Pros:

* No monetisation or adverts at all
* Complex/competent ai with a consistent play style you can learn so the game retains depth despite lack of real people
* Very simple ui no visual clutter at all
* Appears to have similar stakeholders to my game in objectives just without the online component
* Clearly made just for people to enjoy/practice no ulterior motives

Overall:

This is probably the closest to what I want to create, very simple no nonsense design. Clearly designed for similar stakeholders to mine. Obviously, the lack of an online component and statistics hurts it, but I believe in terms of the basic user experience this is a realistic and good target for my product

Observations of research

* Free poker games that aren’t focussed around real world money and gambling appear to web based in the browser or mobile(very few downloadable executables), while my lack of CSS/js knowledge will prohibit me from breaking into the in browser market, a mobile port of my game may not be too hard and is certainly something to consider
* These games are very focussed around getting money, whether by promoting an in-game store or external betting platforms
* There is much differentiation between the casual open tables and the organised tournament games
* The in-game user interfaces are all designed to look like a poker table
* Simplicity is key, most of the depth in poker comes from the players not the platform that people are playing on

Key features observed

* Assists to show hands
* Statistics
* Online play
* A clear and simple ui
* A good server browser with options/information about the game your joining
* A login system so information can be accessed from anywhere

Proposal

Due to the fact that none of these solutions (or any other that I found but did not write up) met all the requirements I had, and my stakeholders outlined. I propose creating an online (but not web based) poker game based around an artificial currency with no links to real world money at all. The solution will have a server browser where people can connect to a game of their choosing so they can play how they choose. The game will also have extensive statistics and analysis features, such as past games overall bankroll and other useful and relevant statistics, so that people can look at their game and learn from the experience, this is important as a large portion of stakeholders are people wanting to learn, improve and understand more about the game and what they are doing right and wrong. Another important aspect I will include is a secure login system so people can assess the game from any device. Having an online system means password security is more important than ever and not having proper password security could put me in breach of the data protection act. I will also try to include a decent tutorial or, at the very least, assists that show the information about your hand and what you have. For example, highlighting and pointing out pairs and straights etc. I also have some aspirational targets for if I have the time and ability to add such as organised more competitive tournament games that people can put their skills to the test in as well as interactive replays.

Software limitations

Due to the cost of actual server space I will have to find a more cost-effective solution to hosting my logon/game server. This will likely be in the form of a raspberry pi which will limit my space and memory meaning I won’t be able to have the large number of users that would be expected from a larger commercial competitor. I will also endeavour not to have to use this server for routing all game information which requires either getting users to port forward, which is a very ugly solution, or figuring out UDP hole punching. Other software limitations are the very server/host side nature of the processing that is necessary to coordinate all aspects of the game, therefor the hosts pc will be slower and have to constantly be sending and receiving packets, which is not a fast process. It would be nice to integrate this into a webapp however I do not have the knowledge or ability to do this with the time limitations of this project as I would have to learn it from scratch an integrate some form of cross platform support. On the other hand, I may be able to have a mobile version as there are many tools for packaging pyqt5 applications for android and all I’d have to change is any operating system file directories used. However, even if I did port it for mobile, I still would not put it on any app stores as this requires licensing and is especially difficult around gambling games. This makes it unfeasible for the time and economic constraints of the project.

Hardware limitations

The game should not have any serious performance issues running on most devices due to the simple nature of the game and design, this could still be optimised with threading if lower powered devices appear to struggle. Furthermore almost all actual processing being done by the host pc as long as they have a strong pc almost anyone should be able to play. The reliance on networking means that people who don’t have an internet connection cant access anything, not even their statistics as they would need to log on, which uses the internet, and access the data itself, which is stored on a server and so needs the internet. Due to using low level sockets I will also need to send raw data across the network rather than images or pre calculated graphs. This will take up more memory on the user’s system when looking at statistics but should not affect the actual gameplay itself. Any dedicated server hosting options cost over £40 a month which is above budget for this project, therefor for testing I will use my desktop pc as a server and will migrate it to a lower running cost low power draw raspberry pi or similar solution.

Software choices/evaluations

For this task I chose to use python for several reasons. Python is one of the most versatile languages currently on the market allowing you to combine object oriented programming and procedural programming in one file, this allows me writhe my code with very few external restrictions allowing me to rapidly prototype and iterate upon solutions without having to commit a lot of time to the codes format and style early on. Python also built around the idea of a “program” being built around several modules, this is useful for me as at different points in the runtime the program has to be doing very different things and separating these out into sperate files will make the development much easier and improve code maintainability. Python does have the disadvantage of being significantly slower than its competitors such as c/c++. This is because of two main reasons, one of which is pythons interpreted nature as opposed to c/c++ which are both compiled languages. This means that, while you can test prototypes much faster in python as interpreting code is faster than actually compiling code, once compiled it is still actively translating from the high level python into machine code whereas compiled languages have a fully written executable of machine code ready to go which is obviously much faster. When packaging pyQT code it is actually converted to c++ so hopefully I will gain some performance back, but it will never be as efficient. Python is also a very high-level language meaning you have less control over memory and pointers, this is great for security and robustness but can lead to slower less efficient programs than its lower level counterparts. Despite this its still the best option for me as, not only do I not have time to learn a new language, python is perfect for the iterative agile development style I am aiming for due to the speed you can get a prototype working. Not only this but as it is interpreted cross platform support with mobile or other operating systems is far easier than it would be with a compiled program which leaves this avenue open if I choose to pursue it.

Python also has a wide range of third-party libraries I will be making use of. Such as the socket module which provides an easy to use low level networking interface built off of WINSOCK. This means the networking features are widely supported and used even outside of python applications allowing me to find information and support from a wider variety of sources. I will also be using the pyqt5 library to create my user interface, this is great as it is built off of the widely used qt platform so includes support for the qt designer, a great tool that allows you to create and edit the user interface while seeing the result in real time unlike something like tkinter or CSS where changes are made in code and it has to be run to see what it looks like. pyQT also has many built in “widgets”, these make getting a functional user interface that conforms to the systems aesthetic easier than ever, as well as having built in events and methods to make displaying and receiving data a very simple process. I will also be using SQLite for my database manager. I made this decision as I already had modules and functions written in SQLite and sending the results of queries over a network using serialisation from the pickle module was an easy and fast solution in testing. A system like mySQL could have been better as I am using for server like processes, however the learning curve and additional code made it hard to justify when I already had a system in place and was on a tight time budget for this project. I will also be using python’s cryptography module as the popular “pycrypto” module is no longer being updated and so is not secure enough for my use case.

By making almost everything processed server side it increases the security as any inputs or changes to the database can be extensively checked and verified before being run whereas storing information client side can cause risks such as someone accessing the raw database without logging in and viewing/changing personal data. There are still potential security threats though, by having an open port and public ip it is important to set up validation of all incoming packets on the server and making sure no message is ran into sql code directly as this opens you up to an sql injection attack. For further security, a ddos protection service such as “cloudflare” could be used. However, with the scale of this project I decided this was unnecessary but if I was scaling it up it is something, I would have to consider.

Stakeholder response to proposal

I gave my stakeholders a chance to read the proposal paragraph of this document and overall they seemed happy with it and felt their concerns had been met and that if a solution like that was on the market for free they would use it. They did comment on the fact that I would be making a downloadable and not web based version, they said that this was a downside as downloading unknown files feels riskier than using a website but that if they were using regularly the convenience of a web based service hits diminishing returns. The more casual players (Felix Judd and Katie Curtis) said a mobile app would be great whereas Louis Gosling (the more competitive/serious player) said he didn’t see the appeal of a mobile game as if he’s playing poker he wants to be focussed so would be sedentary anyway making the benefits of a mobile device lesser.

I don’t think these responses will change my proposal at this point but gave me positive affirmations I’m on the right track.

Success criteria

## Out of game:

|  |  |
| --- | --- |
| Objective | Evidence |
| User can Join a game from list | The server list displays all available games, the user can make a selection and then be entered into the game |
| User can create an account | User can press “create account” button and then their details and password will be hashed salted then sent over the network and stored correctly |
| User can login to an existing account | User can use details associated with an already existing account which will be hashed and salted before being sent to the server and compared, if the details are correct it should allow you to enter the application |
| User can logout of their account | Once logged out any data that had been stored temporarily is removed, sends you back to the login screen |
| User can view statistics of their games | Users can select an option from the main menu from which they can see key percentages |
| User can view their overall money | Overall money is displayed on the menu |
| Once bankrupt money resets | Once the player runs out of overall money (not individual game) their currency resets and this is recorded in the stats |
| User can host a server | User can select an option to host a server which is added to the server list and which people can join |
| An intuitive GUI (graphical user interface) | No console/command line is needed to access any feature. A user with any level of computer/poker experience can easily navigate the menus and do what they want to do |
| All statistics are stored on server’s database | Any statistics are stored remotely and associated with the account rather than stored locally. |
| Data should be protected | Any personal data stored such as names, dates of birth, emails etc. Are stored and transmitted safely and securely in accordance with the data protection act |
| User can fully delete account | The user has the ability to fully remove their account from all servers and databases if they choose to and enter their password |

## In game:

|  |  |
| --- | --- |
| Objective |  |
| Players can see each other’s actions | Once a player makes an action this is reflected on everyone’s screen not just theirs |
| Players can place bets | * Players can choose to bet currency on a round which is then drawn from their stack, * If they bet more than they have they are put all in, * they can only bet money they have. * Money one is added to their stack if they win * Cannot bet less than the big blind |
| Players can fold hands | A player can fold a hand and is then no longer prompted for more actions and receives no money |
| Betting rounds continue until everyone is folded, all in or even | The betting doesn’t stop until everyone has either:   * matched the largest bet * only one player remains in which case they win * players haven’t matched but are all in |
| Pots are allocated correctly | At the end of a hand money is allocated correctly in accordance with any split pots that may or may not have formed |
| Winners are calculated correctly | Once a round is played the winning hand is calculated and displayed correctly and all hands are ranked so that split pots can be assigned in order |
| Players should be able to leave whenever they choose | A button which when pressed exits a player to the main menu and updates their statistics |
| Large and small blinds are taken correctly | The blinds should   * be taken automatically regardless of player action * Increment every time the dealers button does a full rotation * Be rotated by one place every round * Notify players when they are the blind |
| Everything is visible via an intuitive user interface | Users can see al relevant information such as:   * Cards * Chips * Dealer position/button * Pot size * Other players actions * Size of the blinds |
| User can see hand rankings | When a button is pressed a table of hand rankings pops up so that people can see how good their hand is/what they could achieve |

Design Section

**Decomposition of the problem**

1. Login Authentication system
   1. Input for username and password
   2. Hashing and salting of inputs
   3. Checks against server database
   4. Progresses to main menu if correct
   5. Can create a new username and password
2. Account Creator
   1. Ability to add
      1. Username
      2. Password
      3. Password verification (enter it twice)
   2. Block out password so user can’t see it as they type
   3. Creates and stores new Hash and Salt
3. Main menu buttons
   1. View server list
      1. A full refreshable and current list of available games with their status
      2. A back button
      3. The ability to join a game which opens a sperate window
   2. Host a game
      1. A tutorial on port forwarding
      2. A lobby with the option to start/cancel the game at any point
      3. You can see when people join
   3. View statistics
4. Gameplay
   1. An exit button
   2. Visual indicator of what your cards are
   3. Visual indicator of everyone’s balance
   4. Full deck of cards created
      1. Fully random hand shuffling
      2. Each card can only be used once
   5. Prompts to act on your turn
      1. Either call raise or fold
         1. Call. Match the current pot or if its too expensive put you all in
         2. Raise. Allow the user to increase the bet by a certain amount
         3. Fold. Allow the user to exit the game
         4. Record individual players contributions separately
      2. Bets loop until everyone is paid up or folded
      3. A timer that when complete automatically folds inactive players
      4. Calculate the winner
         1. Check cards for hands such as straight, flush etc…
         2. Rank hands from winner to looser
         3. Remove folded hands
         4. Calculate pot allocations
            1. Account for split pots by using the individual contribution and not allowing them to collect more than each player multiplied by this value
   6. Can see other players actions in real time
5. Sql reader
   1. Necessary functions
      1. Append a new value/record to the database as a tuple to prevent sql injection
      2. Replace/overwrite entries when given the new value and the one to be replaced
      3. Read and return a set of records as a 2d array when given the sql code to be executed
      4. Read and write encrypted entries for password security using asymmetrical encryption where a single server has access to the private key

**Structure**

The Program is split into several modules to take advantage of Python’s powerful modular design. This helps me perform isolated tests of individual systems, makes working on the project easier a I don’t have to comb through an extremely large file to find what I’m working on. It also helps with the decomposition of the problem as I can run different aspects in different files. Here you can see the file/module structure of the end user’s application, the right side of the tree represents if the user is hosting the game and the left side represents when the user has joined a game hosted elsewhere. “Client\_A” and “client\_B” handle the majority of the networking and communication during the runtime of the game. “Mainloop” holds the classes used to run the game and all the code associated with it. “Mainprogramclient” holds the user interface and menu code that allow the player to select what they want to do.



Figure 1 - The file/module structure of the application

Below is the module/file structure for the server. This server is hosted on my network and has all the databases needed for the game as well as handling connecting players together. I used an “SQLreader” module I wrote however it was not as powerful as intended because you can’t insert a variable as the table name which meant the functions could not be as general purpose and reusable as I would have liked and be heavily specialised to its use case.



Figure 2 - The File/module structure of the server

Figure - The error message given when attempting to use variables as table names

**Choosing my resources**

**Choosing an encryption library and algorithm**

* I initially planned to use the “pycrypto” library as it seemed simple and lightweight while still being powerful enough to protect my users. However after some research I discovered that this library is unmaintained since 2013 and is therefore not safe for use. This prompted a switch to “cryptography” a more complex encryption library.
* For the purpose of sending sensitive fata to and from a downloadable piece of software a symmetrical encryption algorithm would not be sufficient meaning I had to use an asymmetrical algorithm. I settled on rsa due to its faster validation times and similar security (when using a 2048 character modulus) to dsa while being very well documented.
* Cryptography also supports all major methods of hashing and salting which are necessary for password security so that I am in accordance of the data protection act

**Selecting a database manager**

The database manager used is extremely important as you will spend a lot of time working on it so it must be easy for you to develop on while also being powerful enough to perform the tasks required of it. I decided to compare “mySQL” to “SQLite”, two very popular, python compatible database managers.

**mySQL**

* Industry standard, very well documented and supported
* Allows multiple simultaneous edits which may be useful but due to my limited use of the threading module its use case is limited
* Cross language making it useful as a skill to learn and be familiar with
* More advanced and flexible
* Inbuilt server capabilities

**SQLite**

* I am already intimately familiar with the SQLite browser and python module
* SQLite is, as the name implies, very lightweight and simple which is very useful with the time limitations and scope of this project
* I have already written useful code I can simply reuse
* I have to hack a server together using subpar techniques

Due to the tight time constraints of this project I elected to use SQLite as despite being worse than mySQL I am already very familiar with it and can reuse code/functions from previous projects whereas mySQL has a steep learning curve and would require significant time investment to learn. Furthermore all the functionality of mySQL is possible by combining modules such as socket, pickle and SQLite.

**Selecting a data serialiser**

To send data over a socket in python it must be the “bytes” datatype, for strings this is achieved using the .encode() method however for more complex objects such as arrays and dictionaries a data serialised is needed. I compared the two most popular solutions, json and pickle.

**JSON**

* Human readable
* Cross language (don’t plan on using multiple anyway)
* Industry standard, better documented and supported
* Much safer, as I’m passing ip addresses etc may be an advantage
* Can’t deal with tuples etc and makes non string keys in dictionaries into strings, this causes issues with my mainloop.py using integer keys on a dictionary

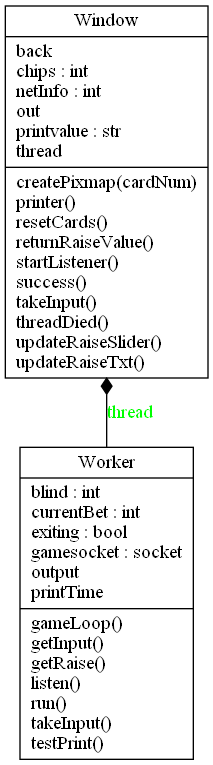
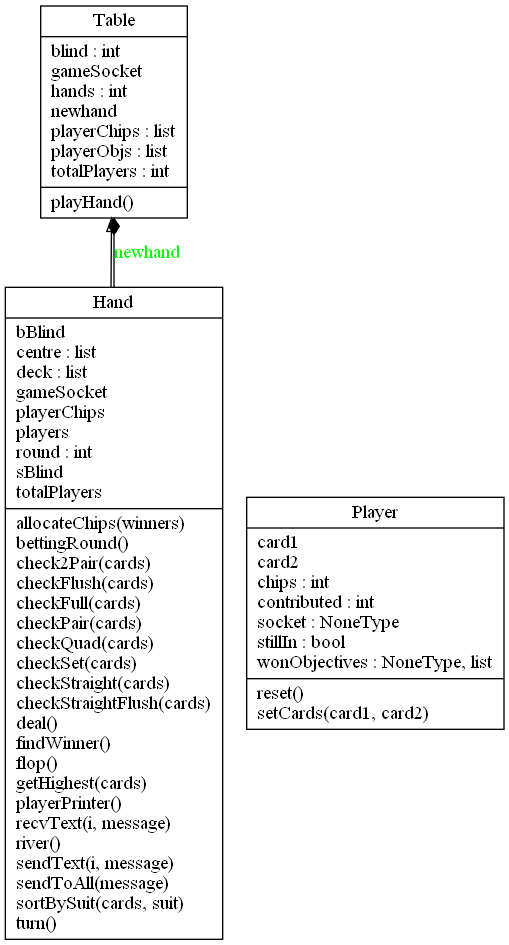
**Pickle**

* Faster
* Can send classes
* Allows for data serialization of more data types and objects and better support for pythonic use cases due to being built from the ground up for python

Note: Could be faster using cpickle but this is not supported in python 3.x yet

Due to the better python support for pickle I elected to use it for my data serialiser

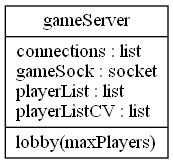
**UML Diagrams**



Client program

Game loop

Server Browser

****

|  |
| --- |
| Hand |
| Attributes:   * integer sBlind * integer bBlind * 2d-array deck * 1d-Array players * 2d-array centre * Integer round * 2d array connected |
| Methods:   * contructor * deal * sendtxt * recvtxt * bettinground * flop * turn * river * allocateChips * findWinner * checkFlush * checkStraight * checkStraightFlush * checkPair * check2pair * checkSet * checkQuad * checkFull * getHighest * sortBySuit |

|  |
| --- |
| Table |
| Attributes:   * TotalPlayers * playerChips * hands * blind |
| Methods:   * contructor * playHand |

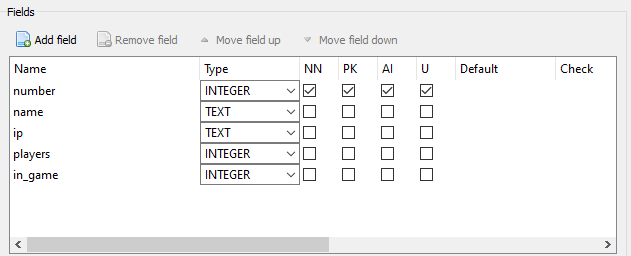
|  |
| --- |
| player |
| Attributes:   * stillIn * contributed * wonObjectives * chips |
| Methods:   * contructor * setCards |

Development section

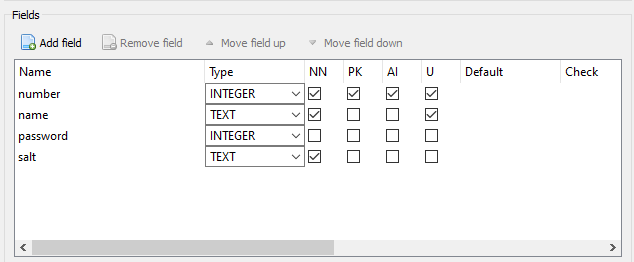
# dev section

## server-side database creation

### Server list table creation and settings.



### Accounts table creation and settings.



### Statistics table creation and settings.

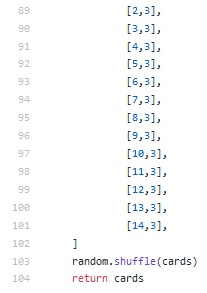
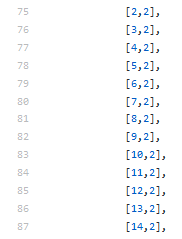
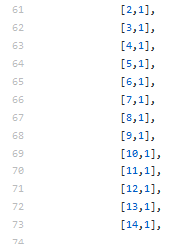


## The command line prototype

First, I made a prototype of the poker game which worked only on one device and was full command line to rapidly test and iterate on the mechanics to make sure it all works as intended.

## dealing the cards

### Creating the deck of cards, the shuffle function.



#### Since the design I made aces high as it makes it leads to fewer edge cases and exceptions

### The flop river and turn functions

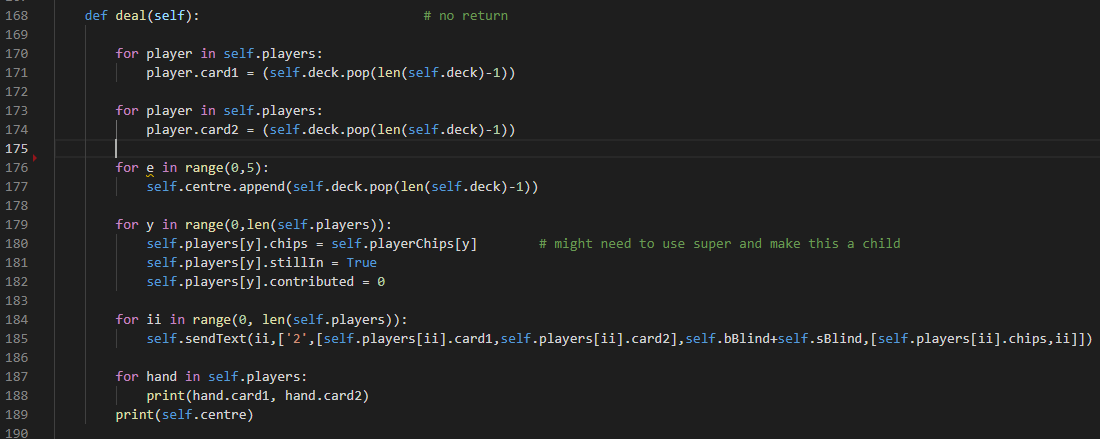
#### The code

#### Evaluation

This code selects the appropriate cards from the centre, combines them with the tag which the user client uses to establish which labels to change. The card is returned but this isn’t used in the final version of the application.

### The deal function.

#### The code



#### Evaluation

This function is essentially a constructor which populates the player objects in the players array. It also handles sending information to the players that they will need at the start of each hand.

## the hand checking functions

### The check straight function.

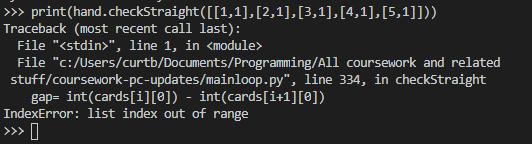
#### Code version 1

#### 

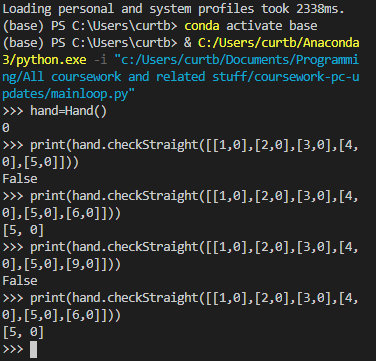
#### Test results

#### 

Dataset was not a straight so the function returns false.



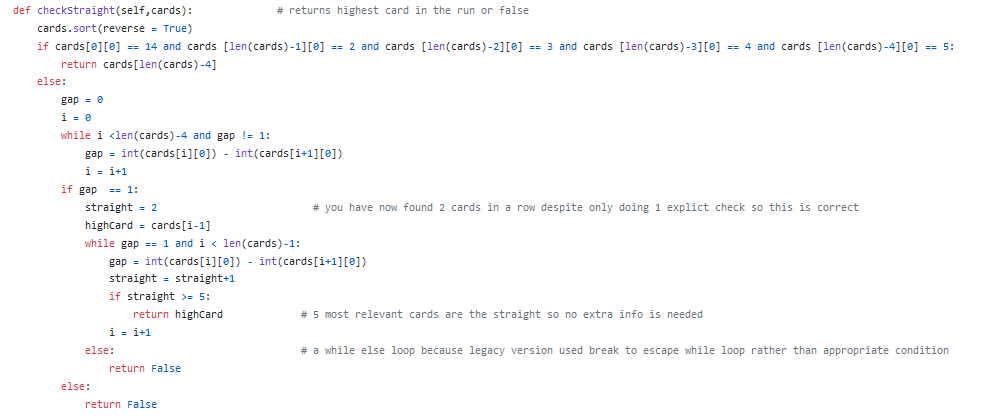
Dataset a straight but the function errored



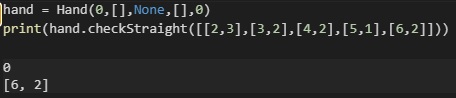
Even once the error was fixed it now needed straights to be runs of 6 rather than the intended 5

The code also has no handling for aces being high numbers which needed to be added after the change was made from the initial design.

#### Revised code



#### Re-testing new solution



returns the highest value from the straight as intended.

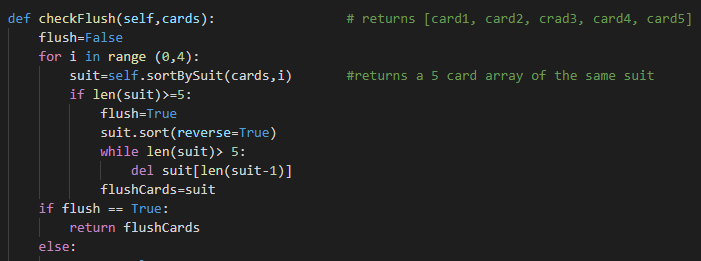
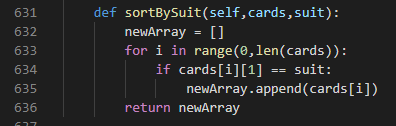
Dataset was not a straight so the function still returns false.

#### Evaluation

The code now works and returns the highest value from the straight which is what can be used to compare to other straights later in the development.

### The check flush function

#### Code



#### Tests

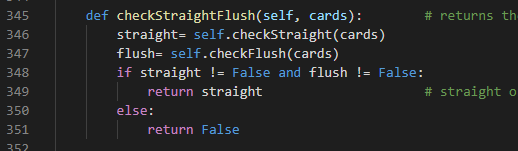


#### Evaluation

The code returns the full 5 cards if there is a flush which is important as comparing 2 flushes could go down to the lowest card if the four highest are identical. The code returned false on non-flush hands which is intended

### The check straight flush function

#### Code version 1

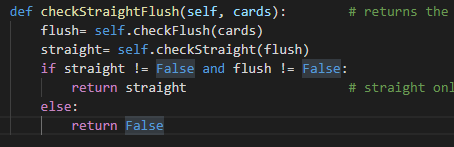


#### Test 1

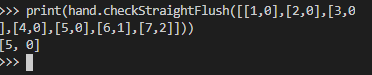


Here a flush and a straight made up of 2 different sets of 5 cards is registered as a straight flush which shouldn’t happen.

#### Code version 2



#### Test 2



#### Evaluation

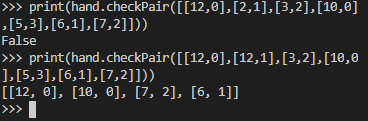
The code now performs as desired by only passing the 5 flush cards into the straight.

### Check pair function

#### Code



#### Testing



#### Evaluation

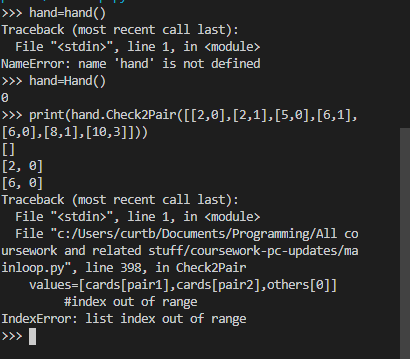
The code returns the 3 non paired cards as well as one of the pairs so if people have opposing pairs the comparison can go down to the third kicker if necessary.

### The check 2 pair function

#### Code version 1



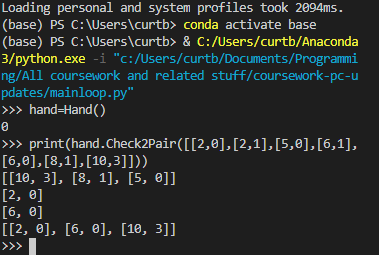
#### Test 1



#### Code version 2



#### Test 2

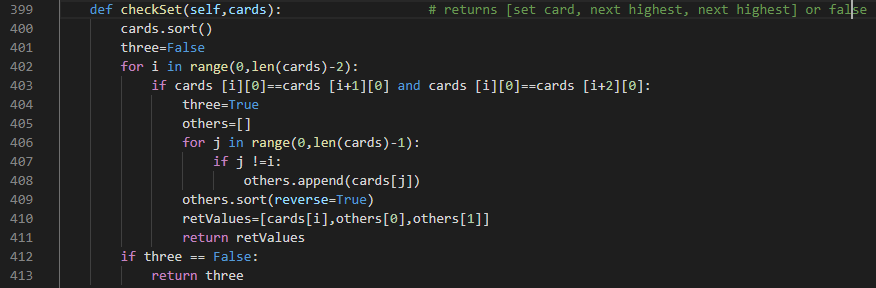


#### Evaluation

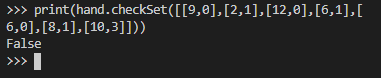
It returns the pairs and the kicker, using the while loop with poorly designed exit clauses lead to index out of range issues, using a for loop allowed for greater robustness and better exit clauses.

### Check Set function

#### Code



#### Tests

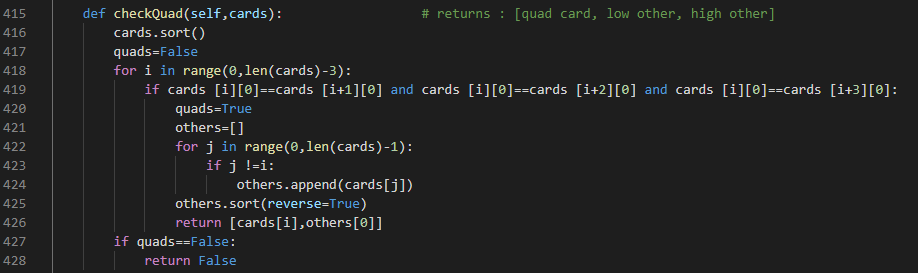


#### Evaluation

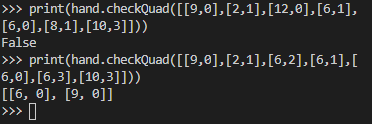
It performs as expected however with retrospect the kickers are.t necessary as even if 2 people have a set they can’t both have an equal set meaning final comparisons however returning these values won’t break anything or affect the end user in any meaningful way.

### The Check quad function

#### Code



#### Tests



#### Evaluation

It performs as expected however with retrospect the kickers are.t necessary as even if 2 people have four of a kind, they can’t both have an equal card meaning final comparisons however returning these values won’t break anything or affect the end user in any meaningful way.

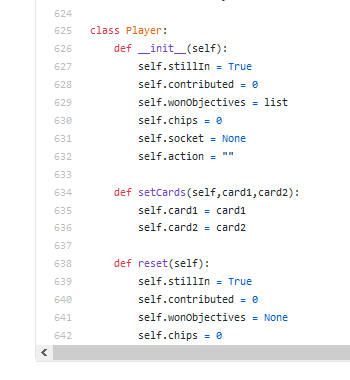
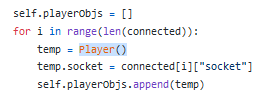
## Switching from dictionary to custom objects

Early in development I used a dictionary which was keyed using player numbers making it functionally no different to an array, each item then had an array which contained useful information, this array was ambiguous and hard to actually understand and should have been dictionary or a custom object. This was a change I then made replacing the original system with an array of player objects.

### Dictionary format



### Player class

## comparing the hands

### After creating all of the functions to compare the hands I then needed a function which could use the returned values and evaluate the winners of the hand, it needed to be able to deal with any number of players folded or not and give an output that would allow another function to then use it to allocate the chips to the players. The function comprises of 2 major loops, the first finds out what you have in your hand using the above functions and the second compares these and ranks the hands

### Part one, evaluating the hands

#### The code

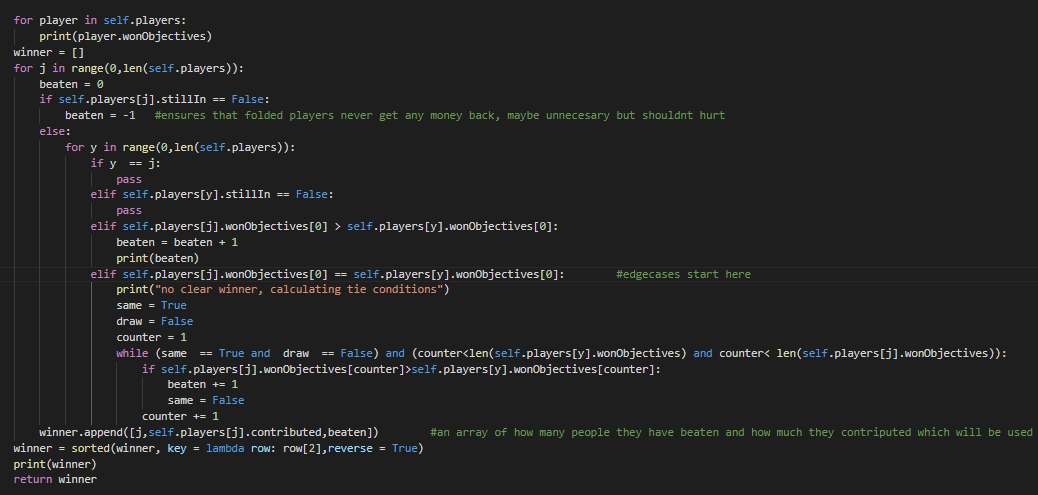


#### Evaluation

This segment calls the functions on the players cards + the centre cards in descending order meaning it will only recognise the most valuable attribute, this is then assigned to the eon objectives property of the player and used in the next loop.

### Comparing the hands

#### The code



#### Evaluation

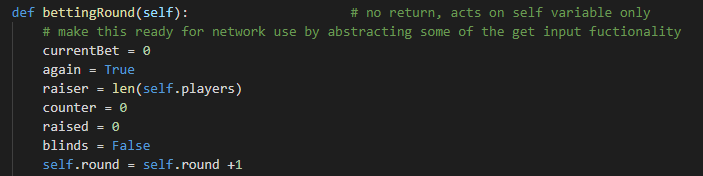
This section of the code comprises primarily of a nested for loop iterating over the players array and finding who has the highest number. If the numbers are the same it goes through the kicker cards and comparing it against each other. This array is then sorted into reverse order so that the allocate chips function can use it to allocate winnings accordingly, more information than just the winner is needed in case of split points were the winner in ineligible for a portion of the bid chips.

## Betting rounds

### Explanation

The betting round function processes all of the inputs needed for the handling of peoples actions during the game. It is the most complex function of the game loop itself and works on properties that have essentially been passed by reference rather than on local variable which it then returns. This allows for it to make changes to multiple properties and means any function it calls during execution have the most up to date values for those properties.

### The code – part 1



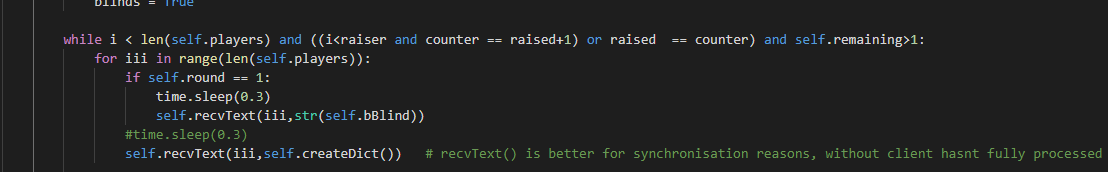
The first section of the function creates the local variables needed at gives them their starting values as well as incrementing which round the game is on by one.

### The code – part 2



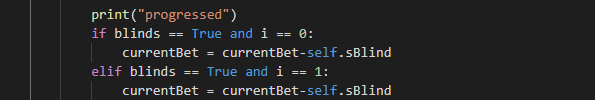
The rest of the function takes using a nested while loop, one may only be ran once but is used if someone raises and therefor you need to go round the table for more people to call. The part pictured here deals with the blinds and is therefor only ran if it is round one and the blinds have not yet been taken.

### The code – part 3



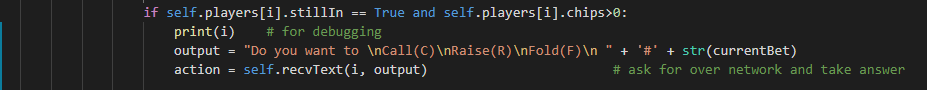
This is the start of the second while loop which loops until it has handled every player or not gotten round to whoever raised that round or until everyone but one person has folded. Here I had to use my send and receive function as I was getting issues with transmissions not being received by the client so requiring a confirmation helps, as does adding a brief sleep period otherwise the client is still processing previous requests before this one arrives. The full networking system will be addressed later.

### The code – part 4



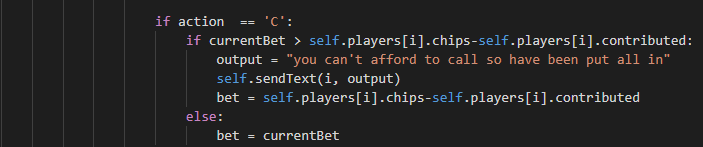
This section actually subtracts the earlier calculated blinds from the first and second player.

### The code – part 5



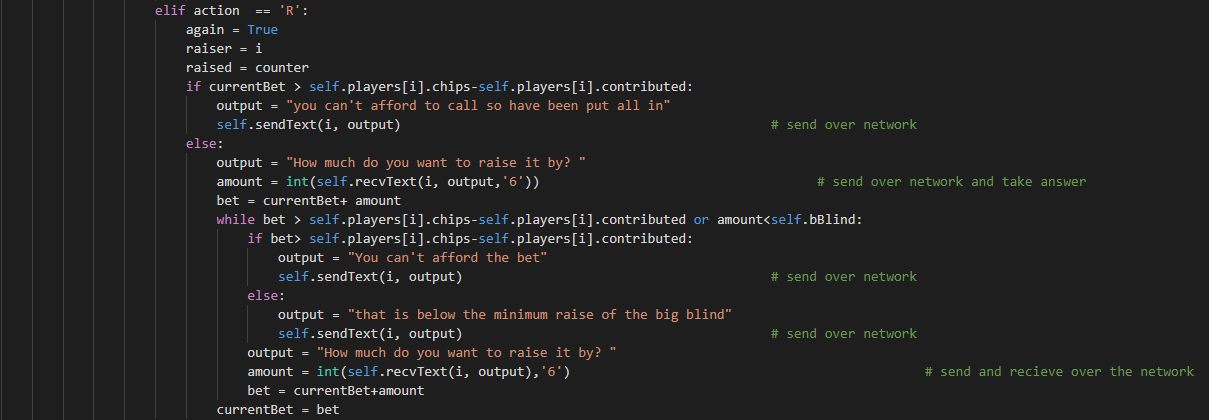
This is the main section of the function that begins by checking the current player is eligible to make an action by checking if they are folded and if they have any chips to bet. It then takes an input from the user, the actual test being sent in never seen in this version but was used both in the single player and multiplayer command line versions. This return value is then one of 3 things, C, F, or R standing for call, fold and raise each of these then have their own processing else/if statement.

### The code – part 6



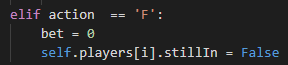
This section processes what happens if a user wants to check or call by adding the current bet to the players bet local variable which once the loop has completed will be added to the player’s contributed attribute. It also checks if the player can afford it and if not notifies them and puts them all in.

### The code – part 7



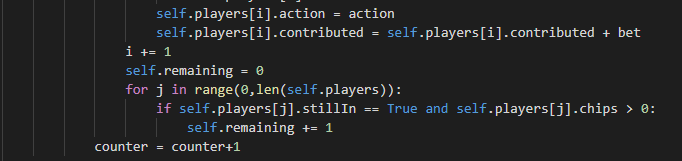
This part of the code handles the raising of bets, after receiving the decision to raise it then needs to ask by how much along with the tag 6 which tells the client that they need to open the raise ui. It also has several checks for if they raise by too much or can’t afford it etc. Many of these became irrelevant as the user interface doesn’t let you enter illegal values but keeping them for insurance in case the ui has an issue is a good idea.

### The code – part 8



This is a short part that handles folding, it has no checks as a player can fold at any time.

### The code – part 9



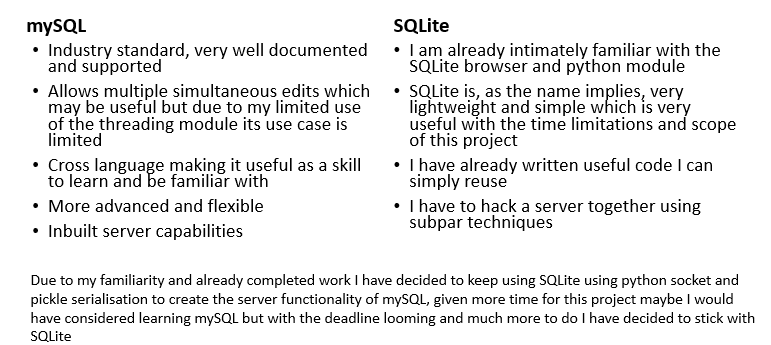
This is the final section of the loop which updates the attributes to reflect the changes of the local variables throughout the function. It also increments the local counters

## networking

### design philosophy and evolution of ideas

The networking in this project is integral to its function and therefor it was important I got it right. My initial plan was a client server model for statistics and choosing a game with a hole punched peer to peer network for the game itself. Unfortunately trying to hole punch for a variable number of clients with the time constraints of this project with the weight of the pandemic I choose to go with a pure client server network. This comes with the advantage that if you use the direct connect function even once my support for the project ends people can still play the game (as long as before ending support I remove the accounts system).

For the database retrieval I choose to make my own networking system using python sockets and sqlite. This is not technically the best solution but my familiarity with these modules allowed me to stay within the time constraints of the project.



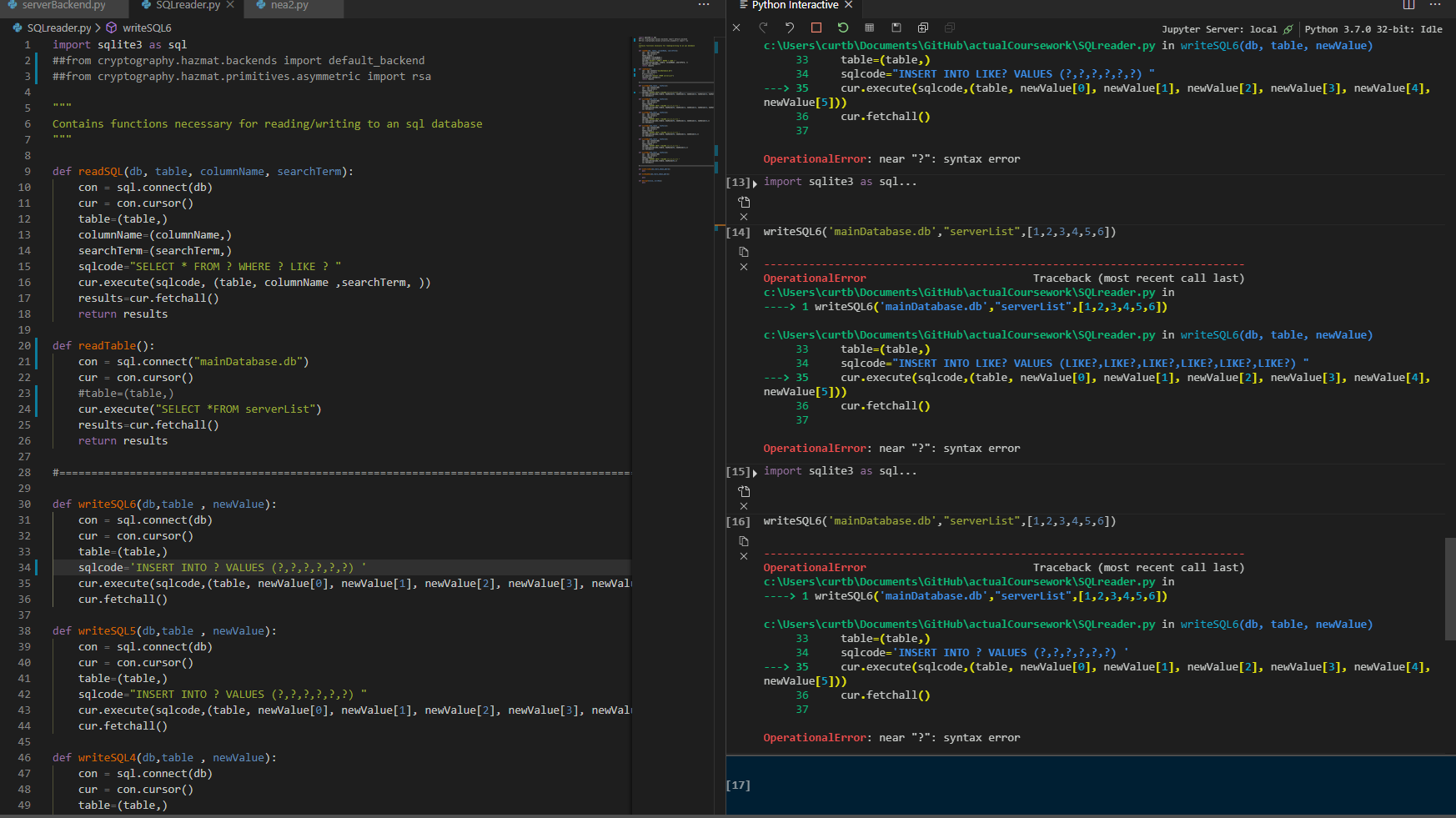
I started with basic prototype files that sent strings between 2 computers to build familiarity with the python socket api. Once this was done I began work on the sql database management system.

### Database manager

#### Dicts vs lists for sql query returns



#### Bug #1

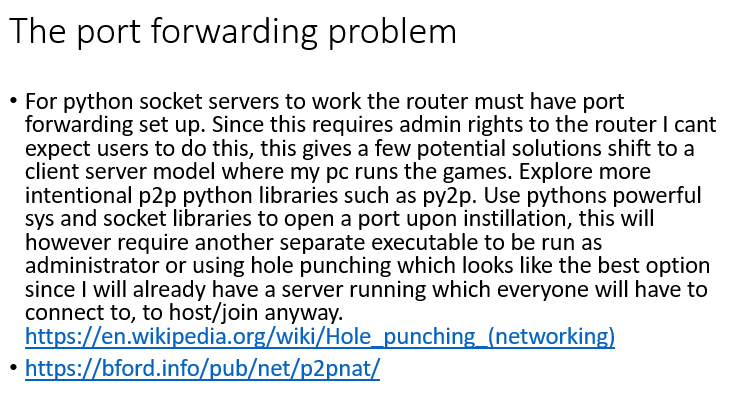


My initial plan was to use a general purpose custom library which can read or write to any record in any table taking all this information as parameters. Unfortunately sqlite does not allow the table name to be variable meaning each read and write needs its own function rather than a handful of general purpose ones. This also lead to me moving the cryptography to the main file using this for only reading and writing.

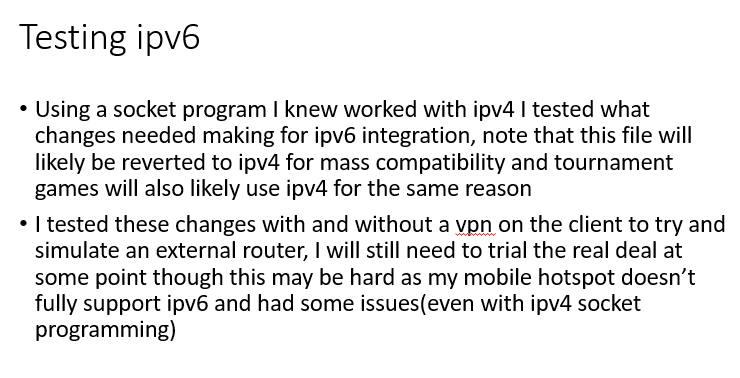
### Game hosting & peer to peer attempts

I originally wanted to pursue a peer to peer system as it would simplify peoples use of the product as if they wanted to host a server for their friends with little to no technical knowledge they could. However this Is a hard thing to do as even large online games such as Minecraft only have client server and do not support peer to peer networks so it is not a huge loss to the competition. Below is a map of my attempts to make peer to peer work using ipv6, custom libraries such as “ipv8” and a fully custom low level hole punching method based on this article all written while developing the program:

<https://bford.info/pub/net/p2pnat/>



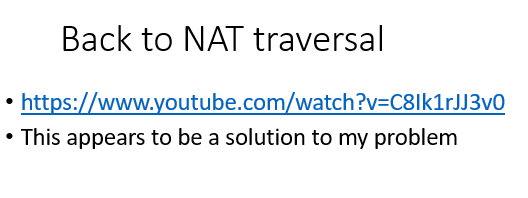


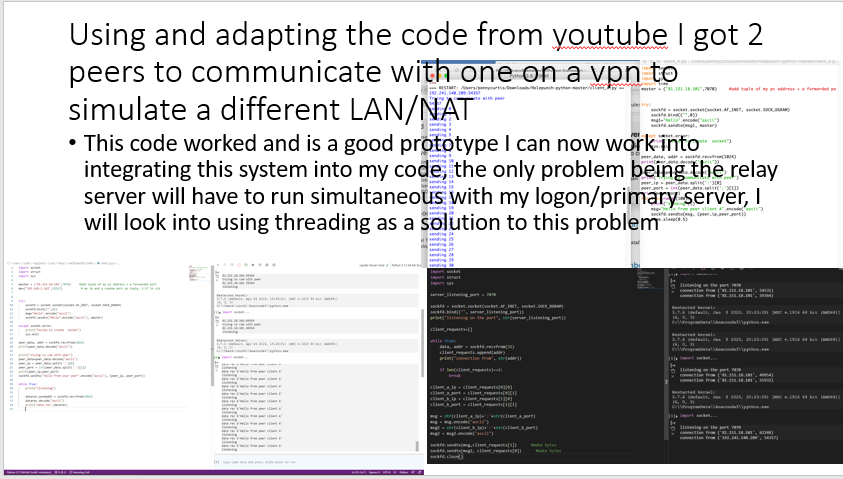
at this point in dev I still had big plans that never came to fruition and have now been moved to post release updates so I can meet the deadline



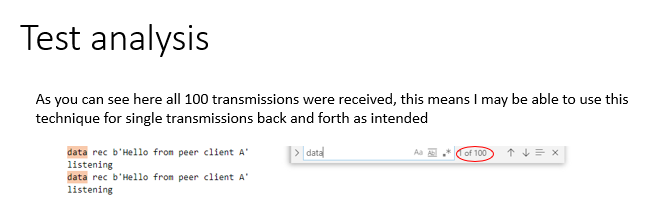
I was previously using link local ip not my public ip which is what caused issues whenoff of my lan.

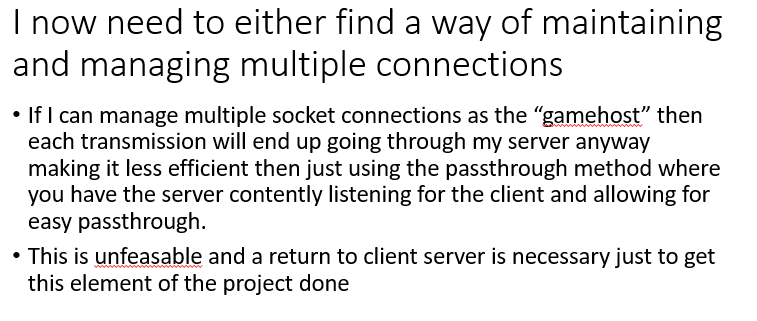
Note: this ipv6 solution was abandoned as it is not yet deployed sufficiently for me to feel comfortable using it for a solution which is accessible to anyone with a windows computer



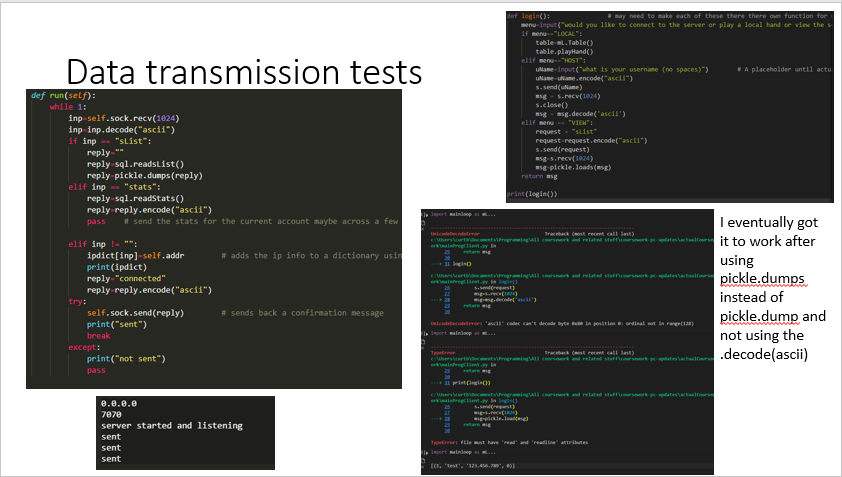


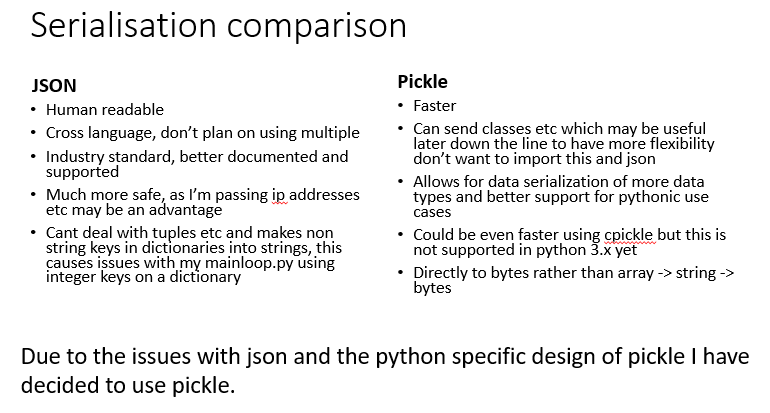
This was the closest I got to a successful hole punching system however adapting it for a variable number of clients proved unfeasable and the time invested into this part of the project was becoming larger than I had initially intended.





### Network communications code and testing



I switched from ascii encoding to pickle as both return bytes objects but by universally uning pickle it allows arrays, dictionaries and object instances to be sent not just strings. The alternative was a combo of ascii encoding and json

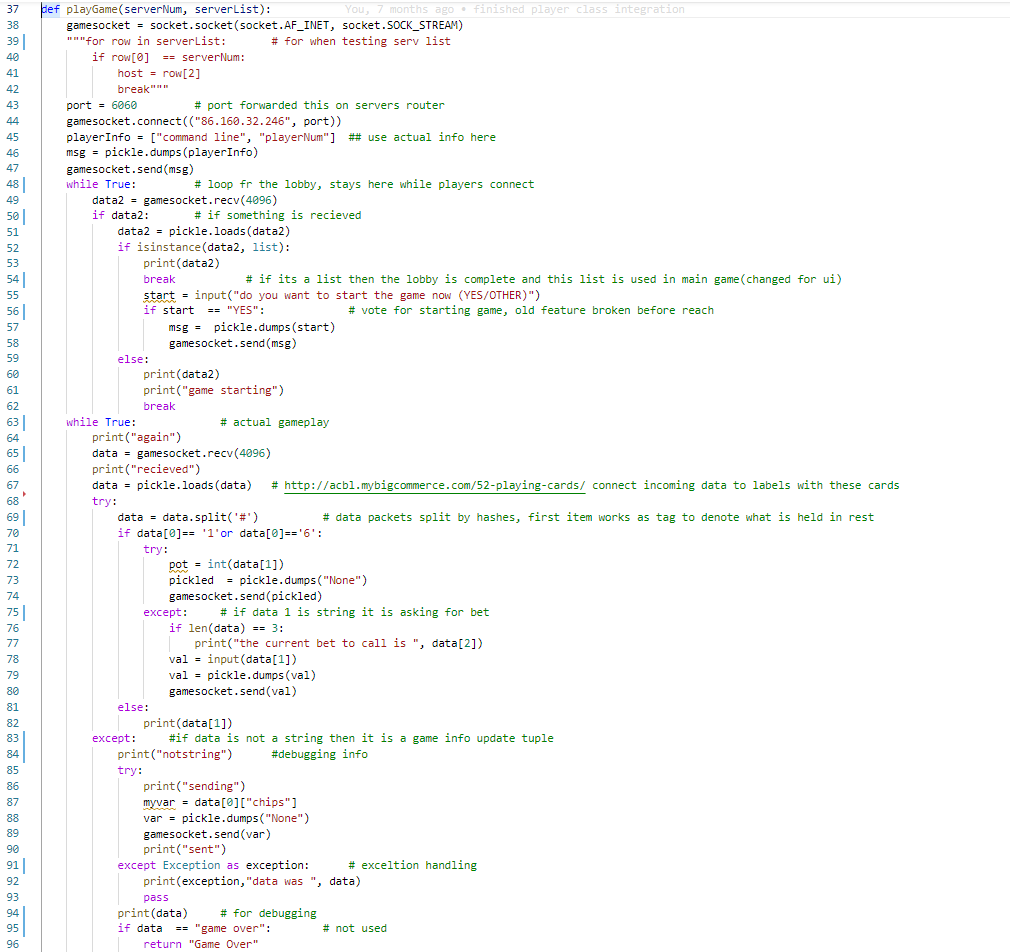
### User Interface and pre ui prototypes

I mostly stuck to my original ideas of a user interface with a menu server browser and game window all separated. I had initially planned to use qt designer but in the end I hard coded the ui as this allowed me to easily change things on the fly from within my text editor allowing me to rapidly prototype and removing the need for a .ui file which saves space and reduces complexity for instillation.

Initially I created a command line version which I could use to test networking protocols and functions before deploying them to the more complex and less changeable ui version. This file had functions for playing a game and used direct hard coded ip connection as it would never be used without me having the python file open anyway. The server browser was returned an array but this was purely to test the sql commands. This caused some issues as the ui and non ui version both had the same username meaning they clashed meaning the server didn’t know who to send data to. The code. Due to the large amounts of server side processing this code could be relatively short and was essentially ust a custom parser for the way I wrote my data packets. Its main function was to decipher tags and give me an easy way to see what data was actually beong received. This was then adapted into multiple separate functions in the final product

Issues:

The simplicity meant it could respond faster than the ui which meant it didn’t catch desynchronisation issues.



#### ui code – the constructor

The constructor creates the ui widgets and initialises them. I used Qframes to allow me to show and hide whole “pages” at the same time. I also used an unbound Qframe to frame the user login system and not have it part of the main window as I felt this was more professional. Obviously due to the use of code to do the ui over a .ui file this constructor is long and unreadable. In future if I wanted to make updates I would likely convert this into a .ui file to make it easier to work on and as I am unlikely to need the rapid prototyping which was the original reason not to use QTdesigners more immediately optimised format.

def \_\_init\_\_(self, parent=None):

self.netInfo = 0

QWidget.\_\_init\_\_(self, parent)

self.setGeometry(100, 100, 1280, 720)

self.chips = 0

self.players = None

self.username = "tool" # get from accounts system

self.potSize = 0

# Put the widgets here

self.opponentBox = QGroupBox()

self.startButton = QPushButton(self.tr("&Start"))

self.statsButton = QPushButton("statisitics")

self.optionsButton = QPushButton("settings")

self.printerLabel = QLabel("placeholder")

self.outputLabel = QLabel("placeholder")

self.selectionCRdo = QRadioButton("Call/check")

self.selectionRRdo = QRadioButton("Raise")

self.selectionFRdo = QRadioButton("Fold")

self.buttonConfirm = QPushButton("Enter")

self.flop1 = QLabel("card1")

self.flop2 = QLabel("card2")

self.flop3 = QLabel("card3")

self.flop4 = QLabel("card4")

self.flop5 = QLabel("card5")

self.hand1 = QLabel("hand1")

self.hand2 = QLabel("hand2")

self.potLabel = QLabel("0")

self.chipLabel = QLabel('0')

self.raiseGroup = QGroupBox()

self.raiseTxt = QSpinBox()

self.raiseTxt.setValue(0)

self.raiseSlider = QSlider(Qt.Horizontal)

self.raiseTxt.setMaximum(self.chips)

self.raiseSlider.setRange(0, self.chips)

self.raiseTxt.setSingleStep(1)

self.raiseSlider.setSingleStep(1)

self.raiseConfirm = QPushButton("Enter")

self.raiseLabel = QLabel("How much do you want to raise the bet by")

self.back = QPixmap("assetts/gray\_ba0k.png")

self.back3 = QPixmap("assetts/folded.png")

self.back = self.back.scaledToWidth(96)

self.back2 = self.back.scaledToWidth(48)

self.back3 = self.back3.scaledToWidth(48)

self.resetCards()

self.raiseLayout = QVBoxLayout()

self.subRaiseLayout = QHBoxLayout()

self.subRaiseLayout.addWidget(self.raiseTxt)

self.subRaiseLayout.addWidget(self.raiseConfirm)

self.raiseLayout.addWidget(self.raiseLabel)

self.raiseLayout.addWidget(self.raiseSlider)

self.raiseLayout.addLayout(self.subRaiseLayout)

self.raiseGroup.setLayout(self.raiseLayout)

self.raiseRow = QHBoxLayout()

self.raiseRow.addStretch(1)

self.raiseRow.addWidget(self.raiseGroup)

self.raiseRow.addStretch(1)

self.raiseGroup.hide()

centerLayout = QHBoxLayout()

centerRow = QHBoxLayout()

centerLayout.addWidget(self.flop1)

centerLayout.addWidget(self.flop2)

centerLayout.addWidget(self.flop3)

centerLayout.addWidget(self.flop4)

centerLayout.addWidget(self.flop5)

self.centerGroup = QGroupBox()

self.centerGroup.setLayout(centerLayout)

centerRow.addStretch(1)

centerRow.addWidget(self.centerGroup)

centerRow.addWidget(self.potLabel)

centerRow.addStretch(1)

handLayout = QHBoxLayout()

handrow = QHBoxLayout()

handLayout.addWidget(self.hand1)

handLayout.addWidget(self.hand2)

self.handGroup = QGroupBox()

self.handGroup.setLayout(handLayout)

handrow.addStretch(3)

handrow.addWidget(self.handGroup)

handrow.addWidget(self.chipLabel)

handrow.addStretch(3)

self.menuBrowserButton = QPushButton("Server List")

self.menuBrowserButton.clicked.connect(self.serverBrowser)

browserLayout = QVBoxLayout()

self.browserTable = QTableWidget()

self.browserButton = QPushButton("connect")

self.browserButton.clicked.connect(self.startGame)

browserLayout.addWidget(self.browserTable)

browserLayout.addWidget(self.browserButton)

self.browserFrame = QFrame()

self.browserFrame.setLayout(browserLayout)

self.browserFrame.hide()

self.radioGroup = QButtonGroup()

self.radioGroup.addButton(self.selectionCRdo)

self.radioGroup.addButton(self.selectionRRdo)

self.radioGroup.addButton(self.selectionFRdo)

self.radioGroup.addButton(self.buttonConfirm)

self.quitButton = QPushButton("Quit Game")

self.thread = Worker(self)

self.thread.finished.connect(self.threadDied)

self.startButton.clicked.connect(self.startListener)

self.loginFrame = QFrame()

self.usernameBox = QLineEdit()

self.pwordBox = QLineEdit()

self.pwordLabel = QLabel("Password")

self.usernameLabel = QLabel("Username")

self.enterButton = QPushButton("login")

self.signUpButton = QPushButton("Sign Up")

self.loginLayout = QVBoxLayout()

self.loginLayout.addWidget(self.usernameLabel)

self.loginLayout.addWidget(self.usernameBox)

self.loginLayout.addWidget(self.pwordLabel)

self.loginLayout.addWidget(self.pwordBox)

self.loginLayout.addWidget(self.enterButton)

self.loginLayout.addWidget(self.signUpButton)

self.loginFrame.setLayout(self.loginLayout)

self.enterButton.clicked.connect(self.processPword)

self.signUpButton.clicked.connect(self.signUp)

self.thread.printTime.connect(self.printer)

self.thread.drawOps.connect(self.drawOpponents)

self.thread.ended.connect(self.endDialogue)

self.thread.inputTake.connect(self.takeInput)

self.raiseConfirm.clicked.connect(self.returnRaiseValue)

self.raiseTxt.editingFinished.connect(self.updateRaiseSlider)

self.raiseSlider.sliderReleased.connect(self.updateRaiseTxt)

self.quitButton.clicked.connect(self.exitGame)

self.gamelayout = QVBoxLayout()

self.windowLayout = QVBoxLayout()

self.menuLayout = QVBoxLayout()

self.menuLayout.addWidget(self.startButton)

self.menuLayout.addWidget(self.optionsButton)

self.menuLayout.addWidget(self.menuBrowserButton)

self.menuLayout.addWidget(self.quitButton)

self.gamelayout.addWidget(self.printerLabel)

self.gamelayout.addWidget(self.opponentBox)

self.gamelayout.addStretch(1)

self.gamelayout.addLayout(centerRow)

self.gamelayout.addStretch(1)

self.gamelayout.addLayout(handrow)

self.gamelayout.addLayout(self.raiseRow)

self.gamelayout.addWidget(self.selectionCRdo)

self.gamelayout.addWidget(self.selectionRRdo)

self.gamelayout.addWidget(self.selectionFRdo)

self.gamelayout.addWidget(self.buttonConfirm)

self.menuFrame = QFrame()

self.menuFrame.setLayout(self.menuLayout)

self.windowLayout.addWidget(self.menuFrame)

self.menuFrame.hide()

self.gameFrame = QFrame()

self.gameFrame.setLayout(self.gamelayout)

self.windowLayout.addWidget(self.gameFrame)

self.windowLayout.addWidget(self.browserFrame)

self.gameFrame.hide()

# layout.addLayout(self.inputLayout)

self.selectionCRdo.hide()

self.selectionRRdo.hide()

self.selectionFRdo.hide()

self.buttonConfirm.hide()

self.setLayout(self.windowLayout)

self.setWindowTitle(self.tr("Poker Game"))

self.loginFrame.show()

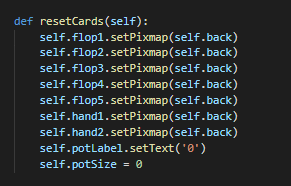
## more user interface analysis

### the code for opponents

### The issues with this

This function draws the opponents using boxes and cards, it uses 2 non nested for loops, the first creates the layouts tables which allows me to have a dynamic number of opponents. This section of code also had many issues in testing which in truth were more issues with my testing procedure than they were with the code itself. The first issue was a lack of if the name of the player is the name of the account, then don’t draw it because its not an opponent. I initially didn’t have this meaning it would draw your cards as an opponent. I picked this up quickly however a sticking point did occur based off of this check, during testing I was using identical files off of my github which had identical names causing issues which took me embarrassingly long to solve. Part of this issue came from the sequence of development so potentially in future even after decomposing a problem I should then take care with the sequence as passwords were done last but would have provided infrastructure for much of the sections which I ended doing before meaning I had to hypothesise about how my password system would be implemented.

### Code for round resets



### Explanation

I had no real issues with this section other than an inability to easily do this with a loop without creating arrays which would make the whole code more complex for no performance benefit.

## the password management system

### the idea

Client-side aspects:

* Passwords + usernames plain text are taken in via user interface
* Passwords plain text is asymmetrically encrypted using public key. This is done using the rsa protocol through the cryptography library for python
* Both uname and password are sent to the server with the tag

Server-side aspects:

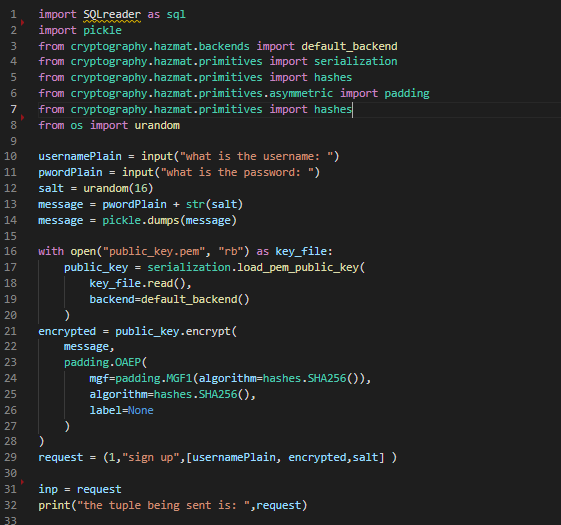
* The username is used to retrieve the hash of the salt and password as well as the salt in plain text
* The password is decrypted using private key and then the salt is applied
* This salted value is then hashed and compared against the hash from the database if it is the same then it returns true

If signing up for the first time the salt is generated client side and the encrypted pword + salt as well as the username and salt in plain text are sent to the db. This is then decrypted hashed and written to the data and then confirmation is returned. Upon receiving this confirmation the client then calls the log in function with the same data they entered to sign up which ensures the sign up process worked.

### The prototype

The prototype was a single file that would simulate the process on one computer by removing the user interface and networking aspects which allows me to rapidly identify and fix issues that come up as well as making the process as efficient and streamlined as possible.

#### The first section (client side sign up)



This first block of code has imports for the whole file but crucially it also includes the generation of the salt. This caused a lot of issues as I tried different libraries such as random and bcrypto and cryptography however these all came with their own formats or strange imports which added complications which the os module did not have and the reason I didn’t start with the os module is simply not being aware of the urandom feature. This block starts by taking inputs, then combines them if necessary and ends by pickling them, this step is important as pickling makes things into bytes objects which can then get unpickled into strings even if the original was also a bytes object which causes issues when hashing as the hashing algorithm requires specific data types.

#### Block 2, server side sign up



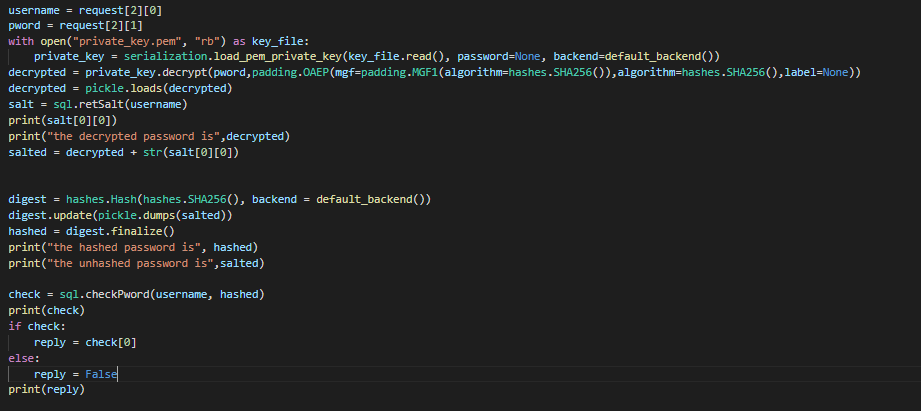
This is a short simple section which decodes and then decrypts the incoming password hashes it using SHA256 hashing algorithm included in the cryptography module for python. This is printed for me to debug and then written to the database.

#### Block 3, client-side login



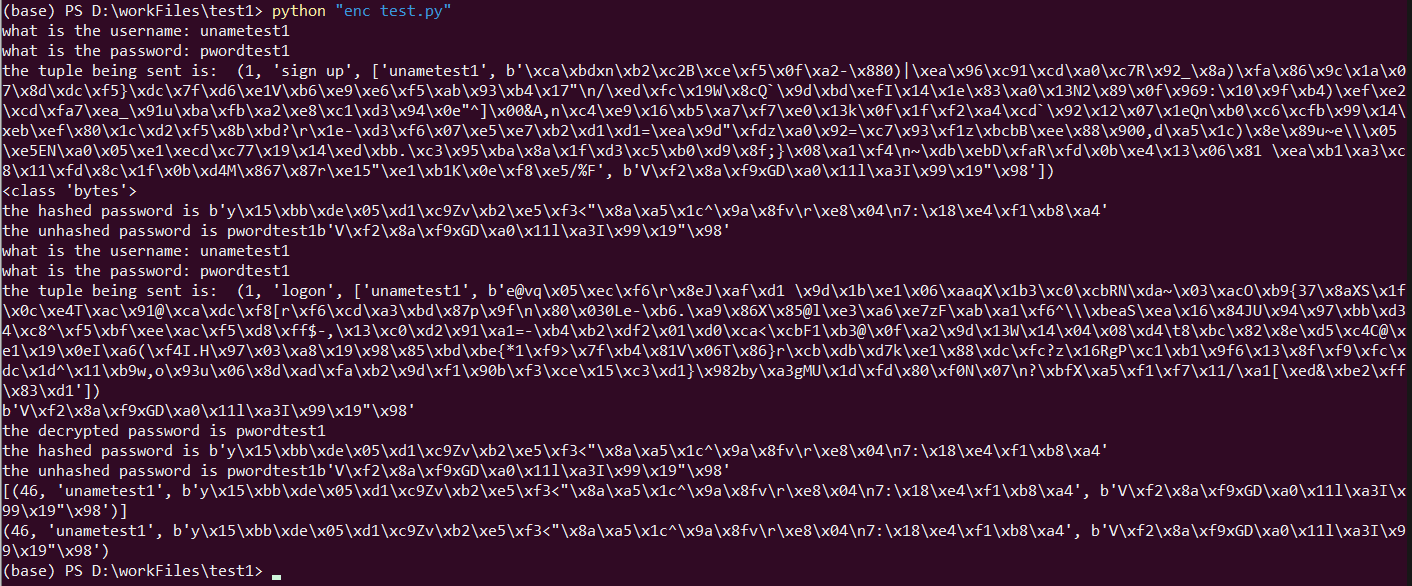
This block takes a second set of inputs, encrypts using the public key and creates a tuple to simulate what would be sent over the network in the actual program. This tuple is also printed for debugging so I can easily understand any issues. This also led to questions such as, how do penguins get to school? The answer of course is on their icycles. Furthermore since this is a prototype and learning experience it is very useful to understand the exact format and what the functions that I used from the cryptography module actually returned.

#### Block 4 server-side login



This last block handles login requests taking the tuple from the last block, decrypts it using the private key, searches the database for the username, returns the salt and adds the salt to the passwords then hashes it and checks it against one stored in the database, if they match then it returns the username of the person, if not it returns False.

#### The output



#### The issues and what I learned.

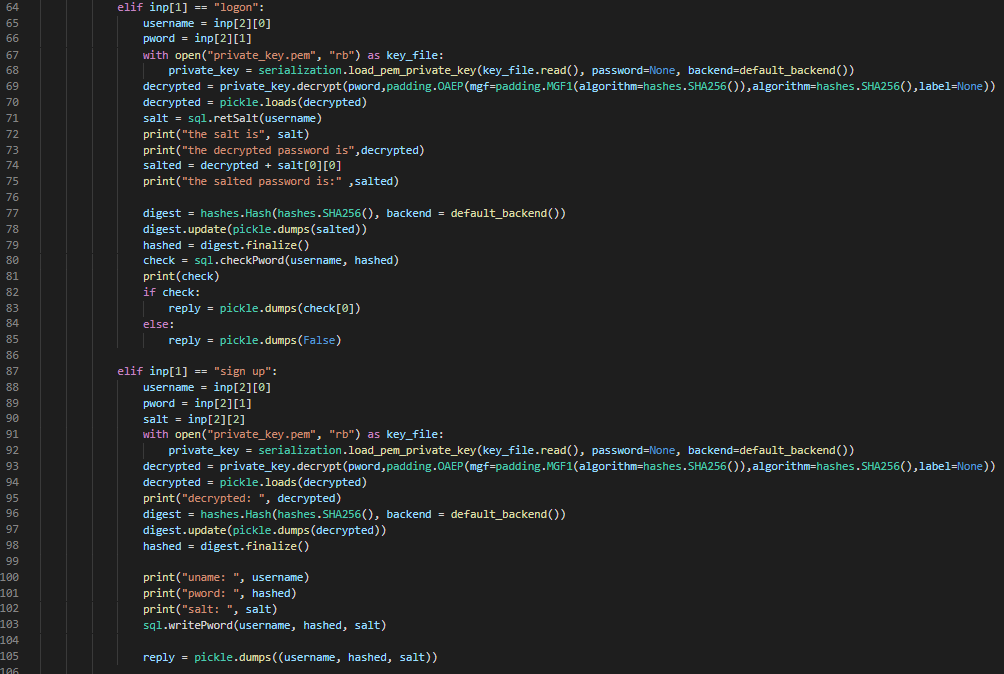
By creating this prototype I familiarised myself with the functions of the cryptography module and allowed me to test the sequence for the program and when I should apply the salt as this was not immediately clear. I also contacted industry professionals about this sequence and they agreed that the salt should be applied server side on checks and client side on sign ups. The main issues I ran into surrounded the salt how to generate it, how to concatenate bytes(the salt) and strings (the password). These were mostly resolved by switching to urandom and properly casting the data types.

#### Necessary changes for the online version

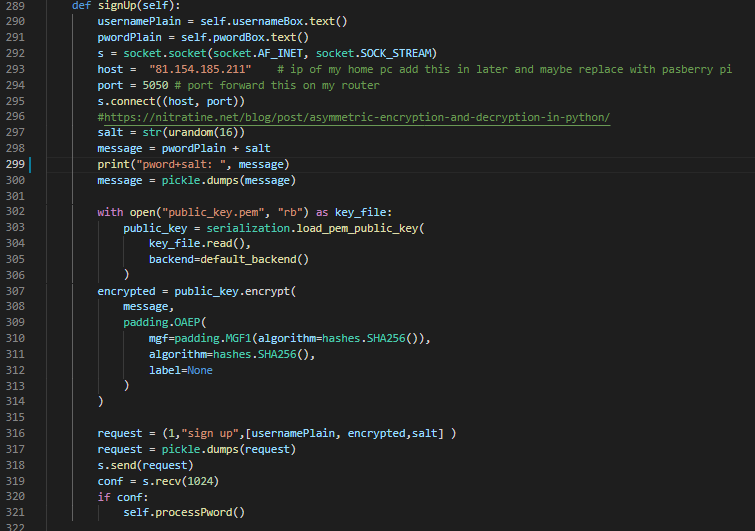
Very few changes were necessary when shifting into the networked version 🐧 and the changes that were necessary were inconsequential and minor changes primarily brought about by the pickling not properly converting to the correct file type leading to even stricter casting.

### The final code

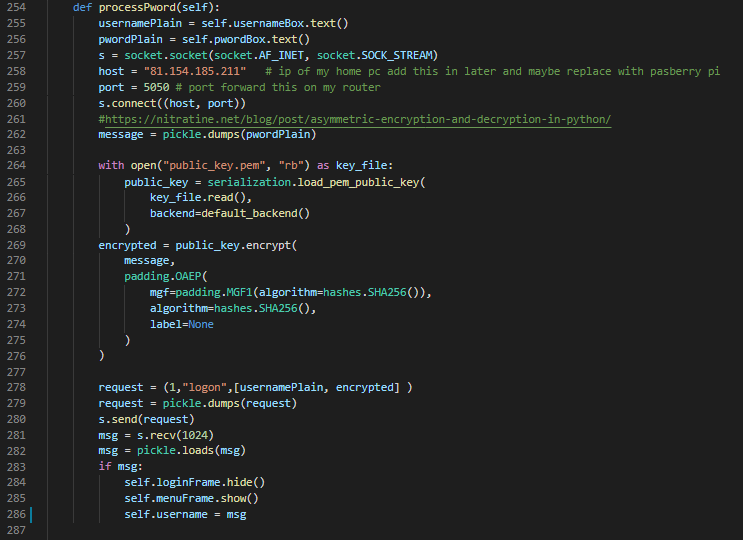
#### Server side



#### Client side sign up.



#### Client-side log in



The most crucial part of this is line 286 which makes a property of the window the username which is what actually determines the name and avoids the issue I had in testing of 2 accounts with the same username causing what were effectively hash collisions with 2 people having the default name.

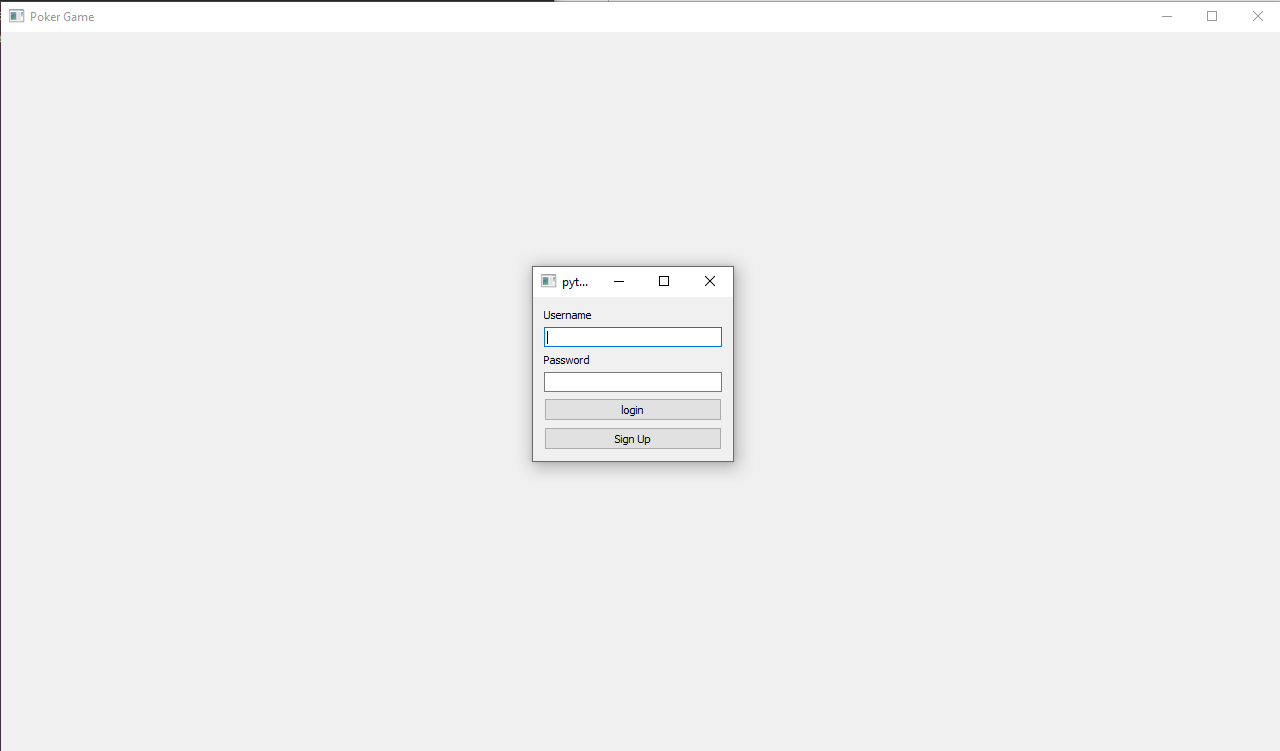
Note: all ips will be replaced with permanent hosted ips once a permanent server space is bought.

# testing to inform development

## UI elements

The majority of this testing was performed iteratively as I wrote different alements testing between runs. The major problems I ran into were issues with displaying frames and changing layouts at runtime which is of course what lead me to use QFrames as these can be shown and hidden to imitate having whole separate pages using only one line of code.

### The Password window



This is displayed correctly with all input boxes being editable and buttons in the conventional order so as not to break with tradition



The username is readable but the password is hidden to prevent shoulder surfing by which an attacker looks over the shoulder of their victim to see their passwords and usernames.

Password test 1:

|  |  |  |  |
| --- | --- | --- | --- |
| Purpose | data | Expected result | Actual result |
| Test that if an account doesn’t exist the system doesn’t crash | Demonstration, password. Click log in | Nothing would be added to the database neither application would crash | Nothing added to db but user crashes |
| I then added a check for if the returned salt is not an empty list then the account doesn’t exist so I should return False (same as if wrong password) | | | |
| Test that if an account doesn’t exist the system doesn’t crash | Demonstration, password. Click log in | Nothing would be added to the database neither application would crash | Nothing added to db no crashes |
| Test a non existing password/username can be used to sign up to a new account | Demonstration, password. Click sign up | The username salt and hashed pword are added to db and you gain access to the menu | The username salt and hashed pword are added to db and you gain access to the menu |
| Test signing up with a username that already exists | Demonstration  Wrongpword  Click sign up | Nothing happen | User crash |
| Added a check to see if the account exists so as not to fail the unique constraint | | | |
| Test signing up with a username that already exists | testuser  irrelevant  Click sign up | Nothing happen | Nothing happened |

Test recordings:

### The main menu

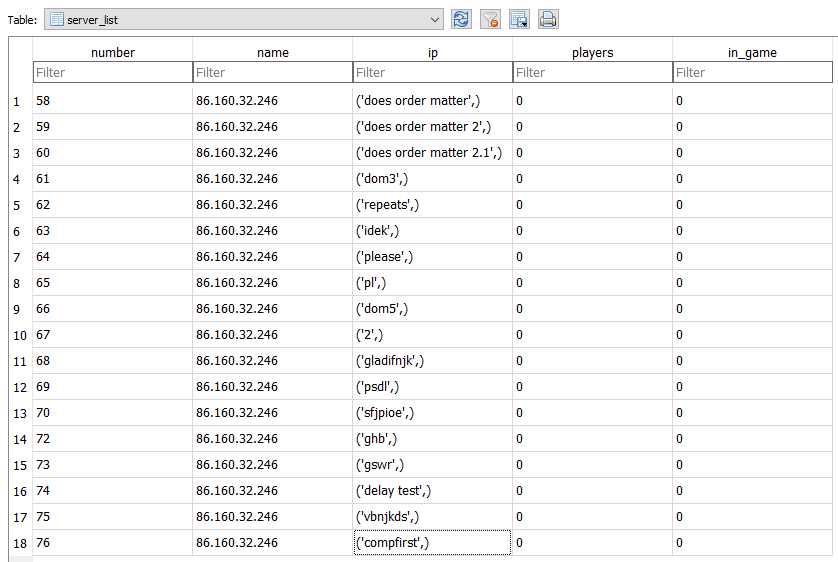


Despite the large use of negative space I believe this is a good user interface as it keeps necessary buttons near to each other and leaves plenty of space for future expansion. This is all displayed correctly.

#### Demo video of menu



### The server browser

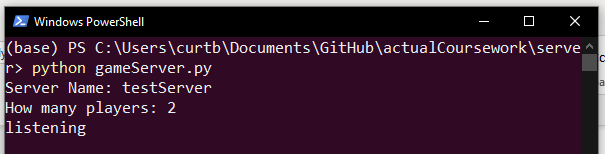


This is the database after development.

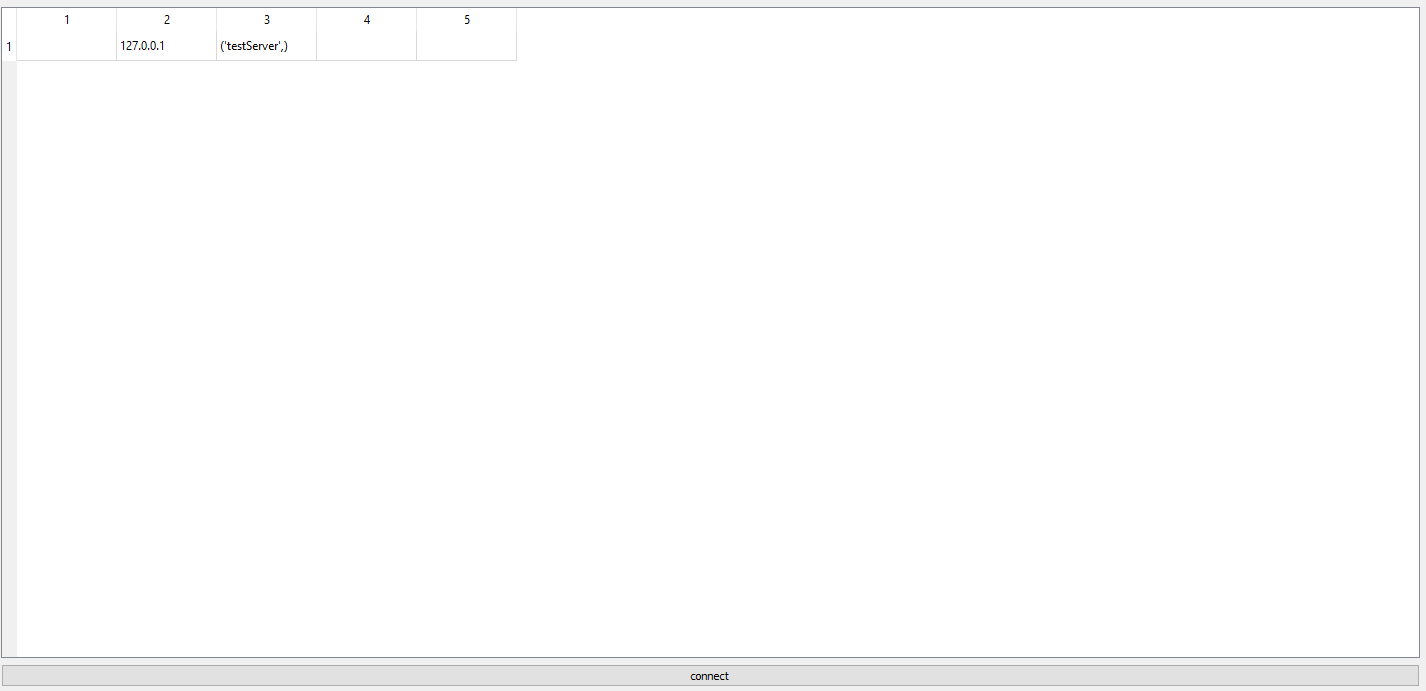
#### Testing

You see does order matter above which was part of non-recorded iterative development were I was checking if the order between connections matters. This was part of the testing which found the isse referred to earlier where both accounts had the same name.

##### Server creation



##### Server viewing



Here you can see that the server has been added. It has a local host address as all of this is being done on my home pc. If done on separate devices it would use the global ip.

### The game

This is what the waiting room looks like, it is just a game with no information in any of the positions and no opponents.

|  |  |  |  |
| --- | --- | --- | --- |
| Aim of test | Input/steps | Expected results | Actual results |
| Test client’s joining | Follow the normal expectations | Once joining you enter lobby and wait until its full | After joining you enter the lobby |
| Game starting | Get enough people in a server | Once all are in the game starts by asking for inputs and rendering details | Once all are in the game starts by asking for inputs and rendering details |
| The cards deal | Play the game multiple times and for multiple rounds | The hands are dealt randomly and differently every time, no duplicate cards | The hands are dealt in accordance to stats |
| Split pots | Edit source code to rig the deal and match the cards of 2 competitors and pricing out one of them | The all in player can only win the money on the table at the time that their all in was fully matched and no more | Chips distributed correctly |
| Win decision | Play multiple hands and manually check if it picks the correct winner | The winner gets the chips from rounds they win and the winner is correct | Chips distributed correctly |
|  |  |  |  |

# testing to inform evaluation

For this part of the testing I showed the product to my stakeholders and got their feedback, a deeper analysis of the feedback will be in the evaluation section.

The process, my idea for these tests was to test the intuitivity of the system and how easy it as to get to grips with. Therefor I sent them the files made sure they had everything necessary to run them as these tests were done pre compilation so they needed a python interpreter and me to pip a few libraries to get them up and running.

## Felix Judd

* He liked the logo but found the majority of the user interface quite uninspiring with an all grey colour palette and default button formats
* Several features that he had requested were not present such as tutorials and a hand ranking system.
* He found joining and participating in games very easy and intuitive with no confusion behind the process.
* He found the layout of the main game clean and easy to understand but thought it might have been useful to have a more circular look to more closely emulate a real poker table

## Louis Gosling

* Louis disliked the move to more tournament format games since the original pitch with everyone starting with the same amount and playing, with increasing blinds, until they are out. This was a major design decision that is explained more in the evaluation
* He was however a fan of the more utilitarian stripped back ui that only includes buttons people would actually want to click
* I also showed the stats feature I was working on but could not finish due to pandemic delays. He agreed these stats would have been what he was looking for so had I been able to get this out for the deadline it would have been worthwhile.
* He also showed an interest in user customisability of things like the background or the card backs themselves. Or the ability to change names as he didn’t realise the username used for logging on would be the same as the one in game.

## Katie Curtis

* Katie didn’t really care about the missing features that had been on the original proposal and found the game did most of what she wanted as since the original interview she gained the knowledge of hand ranking so all she wanted was a casual way to play poker with friends with no aggressive monetisation.
* She is technically proficient so found the process of hosting a game very easy as she could just port forward it and run the python file and the rest is done for her.
* She also liked the lack of any complex effects as it saves battery life when she was playing on a laptop.

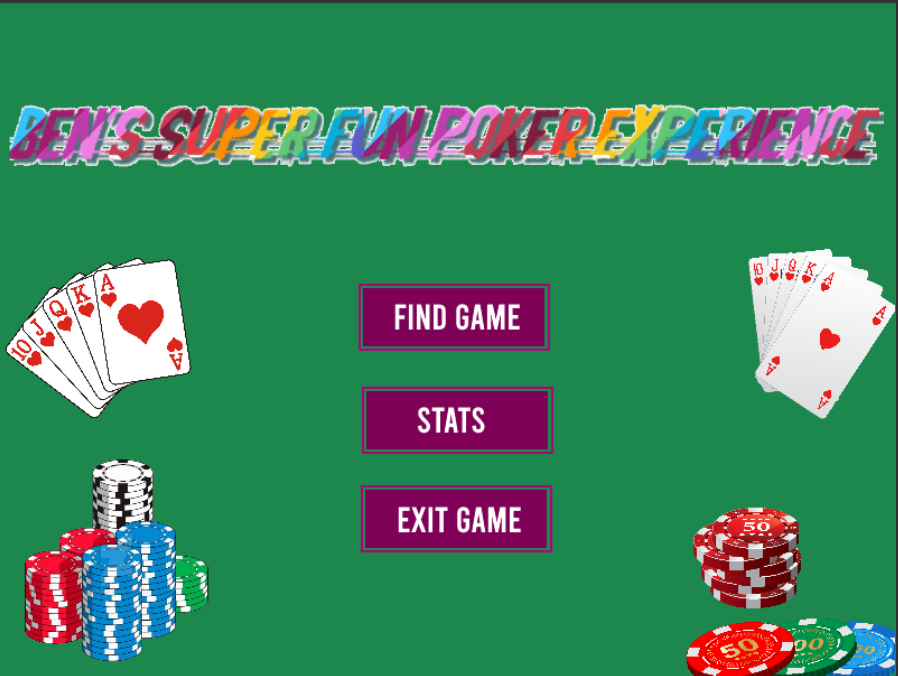
# evaluation

## Feedback from interviews

I got lots of feedback from demoing my game to the stakeholders so I want to break down the trends and what I can do as post release updates and what I should keep in mind for future projects. I also interviewed people with no prior expectations who ere not original stakeholders.

### The user interface

I got mixed feedback on my ui, while some found it visually unappealing and boring, especially the menu which some described as feeling more like industry software or something made in the early two thousands. This is obviously not an ideal comment to be getting however I prefer this to the idea of using the techniques used by many online poker games used to exploit people’s dopamine and cause gambling addictions. Many also use the sexualisation of women in their ui which is obviously something to avoid. That said my user interface could stand to be more visually appealing. This does come with risks however of removing the simple aspects which many people found appealing. Below is a demo image a drew up with potential ideas for future changes to the ui.



This clip art is just to address the claims of a boring ui and to fill the negative space which I actually liked. I don’t own these assets so if I actually implemented something akin to this I would need to buy/commission the clip art assets I am using.

A green background to invoke ideas of felt cards tables and to be reminiscent of classic games such as windows xp solitaire which is the style of card interface I was aiming for since I had nostalgia for it but it is also a great simple design anyone could pick up.

The logo was popular in the previous version and I personally like the colours and style. Furthermore the ironic nature of the name being obviously childish adds to the charm and fits my personal brand and sense of humour

Large box style coloured buttons:

These would stop it being seen as software as I believe this feel came from the default style buttons. They use nice colours to make it visually interesting and to make them stand out and easier to see which improves the usability of the menu.

### The unfinished aspects

It goes without saying that had I finished all aspects it would have been a better product which would have received better feedback and been closer to what the stakeholders had been initially proposed and what competitors are offering. From these issues Ive learnt the importance of proper scheduling and also not relying on smooth sailing. Many of my plans relied on being able to spend a certain amount of time on the project but things can come up in life whether it be a pandemic or an unexpected arrival of a penguin as a pet you should not make plans reliant on things out of your control going perfectly and I have learnt to include a margin for these events when timetabling any future projects. The main part of the program that wasn’t finished were the statistics which was a huge aspect of the proposal and a large part of why this application would be difference. I read many articles talked to experienced and inexperienced players and had a good idea of what was necessary and even went as far as adding the code to collect this information. The main hold up and why it was non trivial to implement was how this information would be stored and when it would be transferred. This partly came due to the same reason as some other issues I did resolve which is the ordering of sections of the code specifically the late implementation of the passwords system. This was because being the most security critical aspect I wanted the most programming and python experience when I wrote it so I would make as few mistakes as possible with this critical system. However much of the application actually relied on a working account system which meant implementing the statistics took more time than expected and I simply did not have time while revising for multiple sets of exams which were critical. The other element I didn’t finish to the level I would have wanted but could have given time was the tutorial and live hand finder which would highlight pairs flushes straights etc and tell you where these compared to other potential hands. This information is easily available online but initially I had wanted to avoid people having to go dedicated poker website as the ads on these sites are often not healthy and extremely predatory.

The biggest loss for me though was the networking and specifically server hosting which was a big failing point on my part which would not have benefitted from a lack of pandemic. The issue stemmed from my desire to have server hosting to be viable without the technical knowledge required for port forwarding a server manually. However the technical knowledge required to do this in python ended up being more effort than I could handle with my experience at the time. However I actually believe now if I reattempted tcp holepunching I may be able to do it due to the amount of time and experience I now have with the socket module which I did not have at the time. In fact looking back I really tried to run before I could walk and had I gone about it the right way I may even have been able to do it at the time. However I think I found a good compromise which is very similar to games like Assetto Corsa, Minecraft and Factorio. All of which expect users to host their own servers however Minecraft also offers local hosting which I don’t have. This means other than servers I host people must host their own servers which requires some level of technical experience or at least access to your own router settings which communal housing may not offer. Thus I believe this is a failure on my part that I have learnt a lot from and I can take these learnings forward into future projects.

### Where to improve elsewhere

Some of the stakeholders pointed out the desire to introduce customisability, this is not something I had originally considered even slightly however looking at it I believe this is the logical next step for the development of the application after I complete the aspects I laid out in the proposal. I believe I could even make the card backs rewards for winning matches or for achieving a lifetime profit achievement. Furthermore I think the ability to change ui base colour removes the issue I was having with deciding on how to make it more visually appealing while also allowing users to make the game more to their liking. This would not require much effort if it was a set of preset colours but I could also learn how to make a colour wheel using pyQT to allow full customisation. A bane changer was also suggested or a separation of display and actual names so regrets such as that held by the stakeholder don’t happen

#### Passwords and security an analysis

The password system is very secure however the system is still open to some attacks. I have not tried it but I am not sure the use of a tuple for my inputs is sufficient to avoid sql injection, given more time I would have checked inputs for SQL keywords and prevented the ; symbol from being used as without this they cant start a new query which is the main risk of sql injection.

The password system is also vulnerable to a brute force attack as there is no limit on password attempts. This could be implemented but would need to be server side and ip or username based to prevent people just restarting and trying again. This is not hard to implement and would increase security so there is no reason not to do this as part of the first of any post release updates.

The system is not vulnerable to rainbow tables or if anyone gains access to the server they will not be able to access anyone else’s account. This is less relevant now but if cosmetics and achievements are added incentive to steal accounts would be much higher and therefor protection even now is necessary. This is due to the use of a cryptographic salt and hash process which both adds a random string to and alters the password so it can’t be read out no matter what you do due to hashes being a one way process.

Overall despite the shortcomings I am very happy with my password system which is using industry standard algorithms and procedures to maintain the utmost security (once I private the github and relace the keys as in the current state anyone could download them and decrypt the data themselves however this is only public to allow teachers to monitor progress).