**SECTION I: Definition and Planning**

**Tuesday 24/10/2022 - Initial Brainstorming**

*Today I read through my task description and started brainstorming a couple of options. I thought about some educational programs and some ‘game’ style program, but the idea I liked most was a messaging service. I was thinking about WhatsApp, and how I often get hundreds of messages on various group chats by the time I get home, which can be overwhelming, and I never know if I’ve missed something important. I thought it could be cool to make a program which will display messages in a more relevant way than simply sequentially in a chronological order, and perhaps remove less relevant messages after some time. I’m not sure exactly how I would do this, but I could write an algorithm to make a judgement of relevance based on the sender, key words and other important data. Alternatively, I could really utilise replies to nest messages,* *maybe even to the extent that every message must be a reply, either to another message or to a conversation topic. There are a lot of ways I can go about this, and I will brainstorm the best ways to do them in the coming weeks.*

**Wednesday 25/10/2022 - Meeting my ‘clients’**

*I discussed my ideas briefly with my father yesterday, and he organised for me to have a meeting today with two of his colleagues that work in technology. I told them about my idea, and they thought it was quite appropriate and relevant, and that it should be achievable in the given* *time period. They helped brainstorm how to determine the relevance of a message, and I thought it was* *really valuable. They have also offered to have future discussions on what they would want in a product like this, and to test the product when it is complete. Overall, I think this was a great opportunity and I’m looking forward to working more on the project.*

**Thursday 02/11/2022 - Researching the required technology**

*Today, I conducted some research for my project. I was specifically concerned about how I would get two different devices to communicate with one another, as I don’t have experience with this. It turns out that potentially the best implementation of this in python is to use the sockets module, which allows multiple clients to connect to a server. However, there are a lot of complications that this creates. Some of these are technical complications, such as how to ensure that the server can simultaneously communicate with more than one client. However, the larger concerns are social and ethical concerns. The main problem is that a server exists in public, and anyone can try to establish a connection with the server or read the contents on the server. This is an issue – no one should be able to read all messages or send dangerous data to clients. This means that I’m going to have to be careful about how I implement this to make sure that the clients are kept safe and their data uncompromised. Fortunately, these are all challenges that have been overcome by the standard messaging apps that are used today, and therefore with the proper research, everything should be okay. Tomorrow, I will get a proper start on the documentation.*

**Friday 04/11 - Starting the Problem Definition**

*Today, I began to write my problem definition. I found this surprisingly difficult, because I was trying to specify the criteria the final product must obtain, and this was particularly difficult. How do you determine if the messages chosen by the computer are representative of the conversation? I’ll continue to think about this. I have included some of the requirements I thought were obvious.*

**Sunday 06/11 - Small edits to the Problem Definition**

*Today, I continued to write my problem definition. Now, I have split the requirements into four main categories: requirements for any service, requirements specific for my app, requirements about efficiency and social & ethical requirements. I think this is probably a good way to break it down. I am still struggling to come up with all of the requirements in an air-tight manner, but it’s getting better.*

**Thursday 10/11/2022 - A change of plan**

*Today, I made a change. I was worried about this idea of selecting the most relevant message because it would be difficult to write an algorithm for and to verify the accuracy of that algorithm, so I have completely discarded that. Instead, messages will be reduced using replies (as before), and by creating tools to reduce messages in common situations. I was thinking that most of the time when someone is overwhelmed by messages, there are usually predictable reasons for this. It may be because the group was trying to schedule a meeting and everyone was arguing over what day and time to schedule the meeting, for example. So for these common situations, I’ll develop tools so that these messages don’t need to be sent. For example, maybe the tool for meeting scheduling would have buttons that allow everyone to select when they would be free, and then after some time output the most appropriate time for a meeting. This means that instead of dozens or hundreds of texts being sent, two are sent: The message with the buttons, and the message with the results. The point is that many tools like these ones will be coded, and so in many situations the number of messages that will be sent can be significantly reduced. I think this is a much simpler and more elegant solution to the problem. I have rewritten the majority of my problem definition to reflect this, and it is almost complete.*

**Sunday 12/11/2022 - Progress Made**

*Today, I refined my problem definition and completed it. I have recategorised them into performance, functionality and compatibility requirements. I think this just aligns it to be more standard so that other people can understand it more easily. I have also made another small change in design – instead of using replies, I will be using channels – much like how they are used in Discord.*

**Wednesday 16/11/2022 - Defining Stage Almost Complete and Gantt Chart Updated**

*Today, I completed most of the tasks in the defining stage. There are a few things I still need clarification about with respect to the assessment requirements, but this will be completed quickly. Over the next two days, I plan to complete all my documentation that I need before starting the project – shouldn't take too long. Perhaps some of the pseudocode won’t be finished but it will be close.*

**Sunday 20/11/2022 - Security problems**

*Unfortunately, there have been some events over the past few days which have made it impossible for me to do any work. I’m not going to go into details, but effectively I have two less days than I originally anticipated to complete the documentation. This isn’t my main issue though. The big issue I currently have is that I’ve done some research, and security is incredibly complicated. Not only is it complicated, but it is so easy to get wrong, which would leave my program vulnerable. Only recently, Optus’ security was compromised resulting in thousands of customers having sensitive information leaked, and while their system is far more complicated than mine, they also have an experienced team of developers and cybersecurity experts constantly working on the problem. The truth is that it is impractical for me to be confident that my system is perfectly secure.*

*That being said, I will certainly make a significant attempt to make sure that privacy and security are adequately addressed, as it is a significant ethical issue that cannot be ignored. I have already conducted a lot of research into it and have come up with a solution using standard, trusted encryption algorithms (specifically AES, RSA and SCrypt) in order to properly manage passwords, messages and other data.*

*In addition to my research, I have also completed the IPO chart for this project. I’m a little stressed about meeting the first deadline for my employer (namely Friday for a large section of the documentation), but I should be alright.*

**Monday 21/11/2022 - Began Data Flow Diagram and Storyboard**

*Today, I began making progress on my Data Flow Diagram and my Storyboard. I am pretty satisfied with how both of them are going, but I have struggled to break down the program appropriately for the DFD. Should be fine though.*

**Tuesday 22/11/2022 - Completed Storyboard**

*Today I completed my storyboard. This took me a lot longer than I anticipated. I’m quite nervous about having all necessary documentation ready by Friday.*

**Wednesday 23/11/2022 - Completed DFD**

*Today I completed my Dataflow Diagram. This also took me longer than I anticipated. I then began on my structure chart and started to write out a few of the simpler algorithms.*

**Thursday 24/11/2022 – Completed Report**

*Today I completed my structure chart, algorithms and refined the written parts of my Design Specifications and Considerations. Glad I managed to get this done.*

**SECTION II: Implementation, Testing and Evaluation**

**Thursday 22/12/2022 – Coding begun**

*Over the past two days, I wrote the first lines of code for the project. I began by creating the main classes that will be needed for the user-management side of this project. I then coded some basic functions which support the user authentication process.*

*I have also made a small change in my development process. There are three significant challenges in the development of this project —writing the algorithms that run on the user’s machine and the server, writing the code that allows communication between the user and server, and writing the code that allows communication between the server and the database. Initially, the plan was to attack all three challenges simultaneously. However, I have now decided to approach them sequentially. I am writing stubs in the form of ‘import’ statements and data objects in order to first create a working program that does not rely on communication. Only after properly debugging this aspect of the program will I attempt to facilitate communication between the user and a server, and then similarly with the database. Overall, I believe that while I will be required to write a lot more code, it will speed up the process by breaking the problem into more manageable problems and allowing for debugging along the way.*

**Friday 23/12/2022 - False Alarm**

*Today, I worked more on the user authentication and sign-up processes. As I was implementing my RSA encryption, I found that the process was very slow. This seemed to be a significant problem, as I had planned to encrypt every message using RSA once for each recipient. However after hours of research, I found that really it is only the key generation that is extremely slow, not the encryption itself. As the keys only ever must be generated when a user signs in for the first time, it seems like this slow speed would be acceptable.*

**Sunday 25/12/2022 - Login complete and tested**

*Today, I completed the user authentication process and tested it. There were initially a lot of bugs, especially in relation to the modules imported. For instance, all the encryption algorithms work with bytes-like objects, and I needed to work out how to convert strings to this format. Perhaps the biggest bug was in relation to the database stub. Previously, I had this as a python file, which stored the relevant data (an array of organisations) and when that data needed to be changed, it was imported into another module and edited there. The problem with this is when it was imported into another module, it was effectively copied over and so the original data wasn’t edited. I thought about storing the data as a csv file but it meant I would have needed to process it all and as it was a stub, all of that data processing would go to waste eventually. I did a little research, and the best solution I could find was to use the python pickle module to easily serialise the data into a file and then extract it again later. After doing this and fixing another few smaller errors (usually logic errors), the program worked. I’m very happy to have, at least for now, completed this part of the program.*

**Monday 26/12/2022 - A little more planning**

*As mentioned on Thursday, I previously changed the order that I am programming the app. Today, I updated my Gantt chart to reflect this change. I also included a two-week break for when I am on holiday, which I think is appropriate as I have been working a lot faster than I anticipated.*

**Sunday 22/01/2023 – A start for communication**

*After a longer-than-anticipated break, I got back on the project today. I got quite a lot done.*

*Firstly, I wrote my bubble-sort and binary search algorithms. I altered the bubble sort algorithm to work from right to left. The reason for this is because I wanted to reduce the time-complexity of the sort when the front of the list was already sorted. This meant I could sort it and then when I wanted to add items, I like append them and resort without a quadratic time complexity. Perhaps the more natural solution would have been to use insertion sort instead, but bubble has the nice property of being in-place so I went with that.*

*After writing those algorithms, I created some more class data-structures to support groups, chats and messages.*

*Finally, I began writing algorithms to support communication between users (possibly including the server), and I ran into a problem quite quickly. I realised that I hadn’t truly considered the number of different types of communication, including sending messages, creating groups, deleting groups, creating chats within groups that already exist, adding members to a group … the list goes on. And somehow, I need to communicate all these different operations. One approach I thought of was to use Python’s ‘pickle’ module to effectively send the function calls as Python code to the recipients, however given pickles can execute arbitrary code, they leave the user very vulnerable to a cyberattack. A second approach is to use JSON serialising. However, it becomes very difficult to send complicated objects using this form, as it was not built for Python. A third approach is to hard code a translation from plain text to my Python functions and methods. This is probably the best solution, and it was the one I went with, but it does come with challenges. Essentially, I am writing my own (very limited) programming language and writing an interpreter in Python, which is quite a big task. I am using a style that somewhat resembles Assembly Language, mainly because it is easiest if I can read and interpret the file sequentially. I hope to have this component of the project finished in the next 2 days.*

**Thursday 16/02/2023 - Serious challenges, but finally a solution**

*Over the past month, I’ve been trying to write the translation of commands into plaintext but I really haven’t made progress. It’s a big task, trying to convert every possible communication with others and with the server into plaintext. I tried but it got very clunky and long quickly. There were always bugs that were difficult to fix (which is expected when you’re effectively programming in a machine code-style language you made up and has no comments), and I just progressed slowly. So I started looking for alternative ways to do it. I initially found very little that could help, but then I came across the XMPP protocol and it looks like it could really help, especially given that there are already Python Libraries built to do it. It is also the messaging protocol used by apps such as WhatsApp, Zoom and Discord, so it should have all the fundamentals. This should hopefully dramatically simplify everything for me.*

**Sunday 19/02/2023 - A plan going forward**

*On Thursday, I left things in a bit of a mess. I had found out about XMPP, but I wasn’t sure how things would unfold going forward. So today I did some more research on the topic, and I have a greater sense of direction for this project.*

*The first significant thing is that there will be some significant changes to the underlying structure of the project:*

* *Encryption: Previously, it was my intention to encrypt and sign messages using RSA cryptography, where the RSA key is stored securely using the user’s password. I originally decided upon this as RSA is simple to implement and provides a baseline level of security. However, it was quite basic and did have limitations. However, XMPP has an extension called OMEMO, which has encryption protocols similar to (but not quite on par with) the Signal protocol. Now, messages will be encrypted using OMEMO, which is a much better and faster solution.*
* *Server: Instead of running the server using the Python ‘sockets’ module, the program now will now connect to a Prosody Server. This will just save me a lot of effort trying to ‘reinvent the wheel’, so to speak.*
* *Python Packages: At this stage, I intend to use Slixmpp to manage my use of XMPP.*

*The other thing is that I am far behind schedule. I have created a new version of the Gantt Chart. I have also decided to dedicate all of my SDD and study periods in the next week to this project (of which I have many), meaning that I aim to make significant progress on this in the coming week. Here is the more detailed plan (compared to the Gantt chart) as it stands:*

*Monday Periods 3-5 including lunch: Write Slixmpp code for creating and managing groups & chats, and sending messages within groups. This is using the MUC extension as defined in XEP-0045. If time permits, code in the ‘survey’ widget and the ‘scheduling’ widget.*

*Tuesday Periods 1-2: Write Slixmpp code for individual login (both types). This will be implemented as a chat with the server.*

*Tuesday Periods 3-4: Write Slixmpp code for organisation sign up. If time permits, also implement an option for verified first time login. This should work as follows: Any verified administrator can verify a user by sending a hash of a random 6-digit code and telling them the code in person. The user needs to put in the code to be verified.*

*Wednesday after school: Implement ‘survey’ widget if not already implemented.*

*Thursday Periods 4-5 including lunch: Write some simple bots for testing purposes. One should only communicate by responding randomly with widgets. One should send widgets regularly at a specified time interval, and send a simple, standard message indicating results. One should act as a manager that asks regularly for a report and responds randomly with ‘That looks great!’ (30% chance), ‘Not impressed. Do better next week.’ (30% chance), ‘Can you add more info?’ (20% chance, and loop), and ‘Not impressed. You’re out of the group.’ (20% chance, and kick sender from group). If no response after some time, the manager should delete the entire group. The last one should just send random text in response to every message.*

*Sunday: Test everything using Slixmpp’s testing suite.*

*This last task may seem a little strange, but there is good reason for it. Firstly, it means that I can easily test everything by myself, without having to act as two users simultaneously. Secondly, I’m a little worried about asking people to test this if there’s any possibility that there are security holes in the product. Therefore, I’m trying to create a version of the product that can be tested fully offline, using these bots.*

*Overall, I think this is a good plan, and something that I think can save this project.*

**Sunday 19/02/2023 - A comment on ChatGPT**

*As I have been researching various things related to XMPP, I have used ChatGPT in order to help me better understand everything. Given its controversy, I just wanted to specify broadly how I intend to use ChatGPT as a research tool and as a CASE tool, and how I do not intend to use it.*

*Firstly, for anything that does not involve writing code, I am happy to use ChatGPT. This includes creating test data, writing external documentation and giving me general advice on how to approach a problem.*

*Secondly, I am happy to use ChatGPT for commenting my code. However, I will need to check if its comments are applicable and correct.*

*Thirdly, I am happy for ChatGPT to write some code to demonstrate syntax, especially when I am using modules I am unfamiliar with. This code may be incorporated into my final product, but it will be made clear using comments that ChatGPT originally wrote the code. However, if I need to change anything in the code from a logic perspective (rather than a syntactical one), I must edit the code myself rather than asking ChatGPT to do it.*

*Finally, I am happy for ChatGPT to find errors in my code and correct them, as long as it doesn’t rewrite a significant amount of the code.*

*This will be the framework I will use going forward. If anything changes, I will write about it in a logbook entry.*

**Tuesday 21/02/2023 - Some more challenges**

*I’ve worked on this for a long time yesterday and today, and progress is still slow. The big problem is the encryption – so I’m going back to my old idea of just using RSA. I’m hoping this will be fine.*

**Thursday 23/02/2023 - Not so fine**

*Unfortunately, progress has still been slow, despite my decision on Tuesday. The documentation for Slixmpp is not particularly clear, especially for someone that doesn’t have much experience reading documentation or using the internet at all in programming. Further, things are simply not abstracted as much as I assumed they would be. While there is a lot that is abstracted, I still as a programmer must do a lot of the rudimentary work and given the lack of easy-to-read documentation and online tutorials, I’m simply unable to use the module. I’m going to work more over the weekend and see what I can do.*

**Thursday 02/03/2023 - Some progress**

*I’ve been working consistently on the project over the past week, and things are starting to get a little better. I’m now making progress on the messaging system, and things are starting to come together. However, I’m still well behind schedule. As a result, I intend to move onto the GUI and leave the messaging in an incomplete state. I will still adhere to the previous version of my Gantt Chart, except for the messaging functionality, which will now be completed in the holidays.*

**Monday 06/03 - Starting on the GUI**

*Today, I started working on the GUI aspect of the project. I initially struggled to understand the layouts and how to position things on the screen, but I think I worked it out in the end. I don’t have much progress except for a simple Launch Page and ‘Sign Up Organisation’ Page where the user can input names and emails, but we’re somewhere.*

**Monday 13/03/2023 - Meeting and Possible Change of Plan**

*Today, I met with a friend and past SDD student to discuss my project. I asked for this meeting given the challenges I have talked about in previous logs, in relation to the networking aspect of the project. He noted that the project was a bit ambitious given my experience in developing, the small team size (1 person) and the short deadline. ‘WhatsApp took a year to build’, he noted. However, he suggested an alternative. He suggested that I can simply build the app locally, without providing the full messaging functionality. This would mean that the messages are not sent between devices. It would be almost indistinguishable to an actual chat app where all of your contacts never send or read any messages.*

*I think this idea is a reasonable one, but I want to consider it a little first before making the decision.*

**Monday 27/03/2023 - Plan Officially Changed**

*Yesterday, I decided to accept the proposed change of plan and emailed the project manager regarding the decision. Today, I received confirmation that this is okay.*

**Sunday 09/04/2023 - Progress on GUI**

*Today, I made some significant improvements to my GUI. For one, I now have changed my background colour and buttons so that they are very similar to the designs in my original storyboard, rather than being the default ‘Kivy’ style. I think it is important for this app that it looks fresh, simple and friendly because it aligns with the purpose of the app which is for a clear, uncluttered messaging service. I also added a ‘+’ button to create additional rows in the ‘Sign Up Organisation’ page, and a scrollbar to allow for easy navigation. I spent a lot of time trying to get the scrollbar to work, as initially it didn’t show up. It turned out to be a very simple problem - specifically, I didn’t set the size\_hint\_y to None, which meant that it was constantly setting its size smaller than it actually was. However, I eventually worked this out after going through many tutorial videos and reading much of the documentation, as well as experimenting and seeing what the changes would do. But when it was done, it was really satisfying*

**Monday 10/04 - Many More Screens, a Logo and the Start of Online Help**

*Today, I realised that there when I compiled my storyboard, I overlooked many of the neccesary screens. So I added many of them for the organisation sign up phrase. One such screen was for the user to input their own information and the organisation name. Another screen was to confirm they had signed up and allow them to sign out or verify other users (which hasn’t been implemented yet)*

*A third screen was for the user to receive their password. I should mention that I’ve been considering automatically generating passwords for the user for a while, but today I made the decision to do that. The reason is that I am an amateur developer and I know inevitably that there is a large chance that despite the security measures I put in place, there’s a vulnerability. Even though the actual processes will only run locally, there is still the chance that users’ data from the program will be automatically backed up by the operating system and then it’s online. I don’t want to run the risk of someone’s password that they use across 20 different applications being leaked, so as a result users do not get a choice of password. I haven’t yet implemented the algorithm for generating those random passwords, but I’ve come up with one and verified it’s secure. It has entropy over 90 bits.*

*Then there were a few other things I did. I generated a logo using Open AI’s DALL-E (which gives me full commercial ownership of my logo), and I updated my colour scheme slightly to reflect the logo. I also created a video for my online help, which explains how to sign up. I may rerecord this video at a later point if I have time to reflect changes in the UI, but for the moment it is okay.*

*That was the work I did today. Tomorrow, I’m hoping to code the ‘sign in’ portion of the GUI for users that have already signed up.*

**Tuesday 11/04 - Code becoming Messy as GUI expands**

*Today, I did exactly as I had planned. However, I’m noticing that as I keep adding screens, my code is becoming increasingly messy – especially my KV code.*

*The reasons for this are effectively that there are lots widgets – screens and layouts and buttons and input boxes – each which have to be configured appropriately. I need to often indicate their position, their size, the positioning of the text within them etc. Also, a lot of it is becoming repetitive. For example, with every screen I create, I have to include the following code in order to ensure that I have the right background colour and nothing goes off the screen:*

*<ScreenName@Screen>:*

*border: 10*

*canvas.before:*

*Color:*

*rgba: 1, 1, 0.95, 1*

*Rectangle*

*pos: self.pos*

*size: self.size*

*BoxLayout:*

*orientation:'vertical'*

*size\_hint: (0.9, 0.9)*

*pos\_hint: {'center\_x':0.5, 'center\_y':0.5}*

*That’s 14 lines of code for every screen, of which currently there are 8, and there certainly will be more. Sometimes there are subtle variations (eg. In the dimension of the BoxLayout) but for the most part it’s those exact lines of code that I have been copying and pasting.*

*This comes with a lot of drawbacks. A major one is readability. It is difficult to focus on the bigger picture of what’s happening (eg. Creating a screen and adding appropriate elements) when between all of this we’re getting a lot of small details (setting the size\_hint, the size, the pos\_hint, setting the colour of a canvas which is actually the background etc.). Further, it can be difficult to update, because the values are written everywhere, rather than all in one place.*

*My solution to this is to restructure the program. Instead of having everything built from the simple, foundational Kivy Widgets (such as Screen, BoxLayout, TextInput), I will create more complex, specific widgets first and then use those in my program. For example, I might make a single screen widget called ‘DefaultScreen’ with the code above, and then simply make each screen an instance of ‘DefaultScreen’. That means the code is separated (for clarity) and only written once (for maintainability). I will also do this for many other widgets such as a ‘Form’ widget from GridLayout.*

*This is my task for Friday, in addition to making the GUI functionality work. Up until now, I have been using many stubs (eg. Instead of performing a check if the email exists in the database, I simply have been using the emails ‘New User’ and 'Existing User’. Also for the automatically generated passwords and verification codes, I haven’t generated anything automatically – I've simply used random-looking text I made up. On Friday, I will add those algorithms, either by writing them if I haven’t already or connecting them to my UI if I have.*

*I am not following my Gantt Chart at the moment as it was made when I still was planning on using networking extensively, but I plan on having a working app by the end of the month (though it may not respond appropriately to incorrect inputs). I will then spend 2 weeks on data validation and incorporating other features. Finally, I will begin the formal testing and evaluation phase*

**Friday 14/04 – Half the job, Double the time**

*Unfortunately today was a rather unsuccessful day. I had planned to both clean up the code for the GUI and connect algorithms to GUI.*

*In double the time I had allotted to work, I managed to do the first task, and even that not to the extent I was hoping for. I ran into a few issues: the first was that in the on\_enter Kivy function, the function is called even before the .KV code is executed, so the widgets on the screen don’t exist and I can’t call them by ID. Eventually I figured out that Kivy had a separate function that is called after the screen is loaded. However, for some reason, when I access variables in my code, it automatically checks the base widget first, and only afterwards checks the value in the inherited widget. This is not helpful because I wanted to have the base class use ‘default’ values that can be overridden in the other classes, but the functions can only access the default values. In some cases, I was able to simply remove the value from the base class (eg. Title\_text = ‘Default Title’) and just have it specified in the inherited class, which surprisingly worked. In other cases, instead of overriding the values, I had to work around them. For example, instead of changing the default box layout for my RegisterOrgMembers screen, I just added another box layout inside that one. It’s a little messy of a solution, but it’s the best I can get. Most of the code is much more readable though.*

**Monday 17/04 - Better Late than Never**

*Today I accomplished what I had planned to do on Friday, which is actually generate the random codes and passwords, and connect a variety of algorithms to my GUI. Now, when you register an organisation, then the email and password you register with can be used to sign in, and the emails of the other organisation members will prompt you to ‘get verified’. When you get verified, a code will show up on your screen and if an existing user types in that code, then access is granted to that person. So in short, my sign up system actually works. There are a few design issues to fix up, but overall it’s okay. On Wednesday, I’m going to code the main screen, excluding ‘widgets’ (in this context, I’m referring to the special functionality of the app, rather than the screen elements). I’m feeling fairly confident.*

**Wednesday 19/04 - New Login Process**

*Okay so I was meant to code the main screen today, but I've been worrying for a while about my sign in process. I really don’t like how it was set up – where firstly users have to remember some bad randomly generated password, and secondly I wasn’t promising the same level of security you would get with a professional application. So I decided to outsource this to Auth0. It was quite difficult to set up, and it’s still not ready – I’ve copied some sample code from their website which allows me to access their login through my app, but it’s not connected in any way to my main project – in fact it isn’t even accessible through the GUI. Monday’s task is to fix that all.*

**Monday 24/04 - Login Process Integrated**

*Okay so I’ve integrated the login process into my GUI. This was not easy to do and required me to make significant changes to both of my sign in processes, and some minor changes to my sign up process. However, one positive side effect of this is that there are now fewer screens, and so the KV code is a little more manageable.*

**Thursday 27/04 - Home Screen Challenges**

*Today, I began to create the home screen. However, there were some significant challenges I encountered. For one, almost every screen element can change, depending on the number and content of groups, chats and messages. Further, I realised I haven’t fully coded the ‘communication’ side of the program, so I couldn’t even access my current groups and chats and everything. However, I have created the basic layout and I have 2 buttons to create a new group or chat. The ‘new group’ button kind of works. In order to navigate the challenges, I’m going to need to stop working on the GUI for a little bit and spend some time working out the details for sending messages using my new file-based system.*

**Monday 01/05 - Messages**

*Today, I worked on sending, receiving and processing messages. In this project, I’m defining a message as anything that’s sent from one account to another – whether that be an invitation to a group, a notification that someone has left the group or an actual chat message. Each of these types of messages needed to be dealt with differently, so it was important to make sure it worked. Overall, this was one of the more frictionless parts of the development process so far, but there were still some challenges – for example, when I tried to send a message that was too long, it caused an error. To solve this, I changed my encoding scheme – the message would be encoded with AES, and the key encoded with RSA, instead of encoding the entire message with RSA.*

**Wednesday 03/05 - A failed Experiment**

*Throughout the development process, I have been using the Python ‘Pickle’ module to store Python objects as files. The solution is very easy and allows me to store any object I create as a file without having to write an encoding process myself. The issue, however, is that literally anything can be stored as a Pickle file as everything in Python is just an object. That means you can store a Pickle file which, when unpickled, runs arbitrary code. This is a massive security vulnerability. This issue, was previously discussed in my log from the 22nd of January, and I mentioned then that JSON was another alternative.*

*So today, I tried to convert everything to JSON. And … it didn’t go well, to say the least. Basically, it’s just difficult to convert anything into a JSON-compatible format, especially as I have so many complicated data structures. So I’ve reverted to just using Pickle as I previously was, but I think it’s okay, and here’s why:*

*The vulnerability in JSON is only important for files you don’t create. If you trust the file, you know it’s not malicious and so you can open it and everything’s fine. But in the final product, the only thing that will be stored in files is the information about yourself, which you create, so you can trust. The actual communication would be done through servers using database queries, so files won’t be used. Therefore, this vulnerability will only exist in the first evolutionary prototype that is this project and will naturally become absent in the final product.*

**Sunday 07/05 - Testing and debugging**

*So today I did some testing, and I came across a significant issue. I recorded a video of myself demonstrating and debugging that issue, so I have not written too much about it. Please see the video for details, which is contained in Appendix 3 of the main report.*

**Wednesday 10/05 - Coding the main sidebar**

*Now that I have my chats system sorted out, I was able to code the main sidebar, which lets you select your group and chat. I came across many issues (eg. Kvy seems to add an extra argument when you schedule a function using the ‘clock’ rather than calling it directly – why?) but I managed to eradicate these issues by experimenting, testing and reading documentation.*

**Monday 15/05 - Message List**

*After implementing my sidebar, I now was able to keep track of the chat, which made displaying the messages fairly seamless. However, there certainly were some issues, the most notable of which was that often messages were being sent twice. After some testing, I realised that the reason for this was because if I send a message in a chat I didn’t create, I was adding the message to my version of the chat, but also sending it to myself. Now, I have implemented it so that you remove yourself from the list of recipients when you send a message.*

*From debugging this, I also realised that it must be the case that if you create a group and aren’t the person who sent the message, you don’t receive it. This was an issue that hadn’t yet come up in testing, but I was able to realise it by understanding my previous issue, test it to make sure I was right and then rectify it. Now I actually have a fully functional app, which is pretty exciting.*

**Tuesday 16/05 - Crash and Begin Fixing**

*Today, I was demonstrating my message list to a friend, when it crashed. I was surprised as this didn’t happen yesterday, even though I basically performed the same functions. I thought about what was going wrong by comparing the process I performed today with the process I performed yesterday, and recognised that the big difference was I was using their names instead of ‘Test 1’, ‘Test 2’ etc. I also noticed that the error message came from the ‘Get\_RSA\_PKs’ function and was raising a ValueError telling me that the member wasn’t found in the organisation. I checked the code, and it was performing a binary search. I deduced from this that the list had not been sorted. I edited the ‘add member’ code so that it performed one round of a bubble sort every time it created a new user. After this, the function worked as intended.*

**Wednesday 16/05 - Implementing Binary Searches**

*Today I had noticed that in many cases, although I had specifically sorted the list of members to allow for a binary search, many functions were still using the built-in Python ‘in’ comparison, or a linear search. I changed these to Binary searches, and fixed minor issues where they came up.*

**Monday 22/05 - Removing the ‘Get Organisation Members’ Screen**

*Today, despite knowing that I need to move onto my testing, I found myself very dissatisfied with the ‘Add Organisation Members’ screen. It was very tedious and would lead to a negative user experience. So I changed the processes slightly and changed a few screens, and that was it. Or it should have been. Because then I was getting errors left, right and centre. If I didn’t make the update, it would have all been fine. But I was obsessed with perfection, and so now I had errors. I will fix these errors over the next coming days.*

**Thursday 25/05 - Errors Fixed**

*Over the previous three days, I have been working on debugging the various bugs that were introduced with my change. Most of them have been fairly simple to debug – I knew exactly which processes I’d changed and so I was able to just go into them when they break and fix them. The error messages that Python produced were also quite detailed and allowed me to easily work out what’s going on. At times, I would print variables using my ‘deobjectify’ function when I ran into an error, which helped in identifying them. Further, many of the errors were already existent before the change on Monday but just hadn’t surfaced – I just hadn’t noticed them before. In general, my code is in a significantly better place than it was before.*

**Sunday 28/05 - Module/Unit Testing**

*Given the magnitude of this project, and some of the rash decisions made on Monday, I have come to testing a lot later than I had hoped for. I began my module or unit testing today (I’m using both terms as my tests probably fall somewhere in between). I have decided in order to minimise the time I would need to spend on it, I am only going to test certain functions and modules. I tested many of the searches and the bubble sort today, and there were a couple of small issues but in general, the functions were all working as expected.*

*I also changed my sort function to not occur as often because it wasn’t meeting performance requirements, and I therefore had to change all instances of Binary search to Mixed search. That wasn’t really a big deal – it worked okay in the end.*

**Monday 29/05 - Integration Testing Setup**

*Given the limited time I have for testing and the many modules I have to test which could have bugs, I was thinking about different ways I could test multiple functions at once. I looked up different ways of testing, and I found ‘Integration’ testing, which seems similar to Program testing except that I am not neccesarily testing the entire program – just bits of it working together. So I wrote a fairly simple script comprising five different 'flows', which would simulate different users travelling through different paths of the program – from signing up and signing in, to verifying others, creating groups and chats and sending messages. The point was that I’d be able to compare the number of organisations, members, groups, chats and messages to the number that were expected to determine if there were any logic errors. However, as I was running the app, I ran into many runtime errors, which allowed me to fix various aspects of the code. There were also issues in my actual testing script, which was frustrating as I didn’t want to be working on my testing script – no one was using it. It was just meant to be a vehicle to testing the main app. But it had to be done.*

**Tuesday 30/05 - Integration Testing**

*Today I continued my integration testing. I got the code to run to completion today, which was exciting, but I did get logic errors. And there were also some runs where runtime errors did come up. I consistently refined my output for my testing to be more helpful for me in debugging my program, and kept improving upon my methods. I have recorded many screen recordings of me performing these tests, thinking out loud and explaining my observations, which will be included in Appendix 3 of the main report. I will continue tomorrow.*

**Wednesday 31/05 - Integration Testing Complete!**

*Today I continued performing my integration tests, debugging my code when there were inconsistencies between actual and expected results, and fixing it. Eventually, my expected and actual outputs were matching consistently. This also acted as a performance test of sorts for the project, as I was running many more functions simultaneously than would be happening from one laptop (or even one company most likely, when the final version of the app is complete). My laptop was audibly expending way too much energy, but it could handle it which was a good thing (also, my battery health was weak to begin with and this happens to an extent with actual professionally-created apps as well). As with yesterday, I have recorded screen recordings of me completing the integration tests and will include them in the project.*

**Thursday 01/06 - Testing Documentation**

*Today I wrote up a lot of the testing documentation from the module/unit tests and the integration tests that have been performed. I still have some to go.*

**Sunday 04/06 - An incredibly stressful day**

*Today was an incredibly stressful day for me. I first tried to add error messages to my application, of which I had to find many points, but I got through them. I then tried to compile my application as an executable file, which failed for a variety of reasons. I tried everything – from renaming and moving files around, to changing the way that the filenames are specified in the code, I tried using Kivy Buildozer, Py2App, PyInstaller … all of them failed. I left that temporarily, and tried to record my online help videos, and as I was recording I ran into an error – one that I had previously tested and had not been an issue. As I knew there had previously been a version without the bug, I tried to return to that version in Github. But I didn’t understand Github properly and I messed it up. I’ve now got multiple conflicting versions that I can’t rectify. I actually don’t know what to do now, my code is utterly useless. It’s currently late and I have to go to sleep, but I’m just hoping that there’s a way.*

**Monday 05/06 - Getting Better**

*Okay so the project is due at 10pm tomorrow, and I began the day today by compiling a data dictionary. I found this more challenging than I anticipated given the complexity of my data structures, and I don’t have as many entries in the table as I would like. I then met with a professional programmer at a company named BizCover to help me with my Github, and fortunately they were able to restore the old version. It’s lucky I backed it up – I haven’t been the greatest with my commits but that one I did commit and push. I then completed some of my program testing in front of the people from BizCover and let them do their own live testing, from which they provided helpful feedback. After that, I spent the day completing my project report and online help.*

*There are many aspects of this project which I have needed to compromise in order to get it in on time (for example, there are still errors in my code which I know exist but am too scared to touch in case I mess them up, and I haven’t finished my internal code documentation but it would take me too long). I’m planning to submit version 1 tonight, and then in the comfort of knowing it’s already submitted, make some changes and submit a second version tomorrow.*

**Monday 05/06 - (Second) Last Log**

*I wanted to write a last log, just for my own sake, as an informal evaluation of the project, compared to the evaluation in the report. I have now finished my live help and documentation to the extent that I can, and am getting prepared to submit. In doing so I feel compelled to reflect upon the process of this project.*

*Of course, if I end up refining various aspects of my project tomorrow, this will be the second-last long and the last will likely be quite underwhelming.*

*Over the past 7 months, I have gone through periods of hard work, high stress levels and many failures. It has been incredibly challenging, and if I knew what I would be submitting back when I wrote the project definition, I would be ashamed. But I’m not ashamed. I’m actually quite proud of myself. It sounds cliché, I know, but I am.*

*I started off this project with incredibly high ambitions, and I had a plan on how to achieve them. Perhaps it was a slightly ambitious plan, but it was my first experience creating a long term development plan – it was almost a mathematical certainty that it would be far off from accurate. Towards the beginning of the project, I very quickly produced many of sign up/sign in processes, involving significant security considerations. I also started to create my data structures. Everything was going well, and I was enjoying it.*

*Then, I moved onto the problem that cost me twelve weeks – networking. I assumed it would be simple – something like import server, server.send\_message\_to\_client(message, client\_ID) would do it. Surely I wasn’t the first person making a messaging app? Surely it had been abstracted to the point of sheer simplicity? I kept searching for this abstraction for around 2-3 months, trying a variety of different techniques, but I just didn’t understand the technology well enough. However, coming out of it, I am actually very happy for the information I learned during that time.*

*Due to the issues I was experiencing, I decided to pivot my project to a local prototype – which seemed much simpler. However, there were still lots of complications involving file handling, GUI, etc. These problems ended up meaning that I would usually take around 4x the allotted time to complete any given task. So I had to learn to adapt my project further and create something that was feasible given my newly learned restrictions. And today, that’s what I’ve done.*

*For sure, there were moments where my priorities were wrong – where I should have just been testing out everything but instead I was making crazy changes to the nature of the application – moving to Auth0, redesigning the login flow to be more intuitive, redesigning the verification flow to be less tedious, and more. And it would have probably been better to have a clunky sign up, a clunky ‘add members’ and a clunky verification system, if it would have allowed me to test more thoroughly and finish with more time to spare. That being said, I learned a lot through the constant iteration about my own code maintenance and structuring code in helpful ways.*

*Through this project, I feel as if I have truly learned a lot – not just technical knowledge, but skills in software development and project management. I thought I did a good job generally with project management, and was able to adapt when things became problematic. However, I now have a much better idea of what I am capable of, and will be able to make more accurate plans at the beginning to make much more exciting projects down the line (likely including the continuation of this project).*

*That’s my software project.*

**Tuesday 06/06 – Actually the Last Log**

*This entry will actually be my last - I’m not going to write the same kind of thing again so this one should be quite a bit shorter.*

*Basically I wanted to fix one of the bugs that was in my project and I initially assumed that it must have been in the GUI module because the integration testing did it, but then I thought I’d actually run the integration testing again. And it failed. When I looked through the group members each person had, every group was only in one person’s list of groups, even if more than one person was in the group. I wasn’t sure why this was happening, so I added a few debugging output statements in various places, and eventually determined that when the invitation was being sent, the recipient list was empty. I performed more tests with more debugging output statements to determine that the recipient list was being cleared in a function meant to produce an appropriate error message if necessary, where it really should have only changed the local version of that variable, or that’s how I understood it anyway, which must be false. So I just changed how the function works to not use the .remove method and it works, but I still don’t understand the underlying mechanisms in the first place. I guess that despite having learned a lot during the project, there’s still a lot more to learn, even on seemingly simple things.*

*So for the rest of the day I just kept fixing up my documentation, and overall, I think it was pretty good.*

*I’m not going to be able to fix any more of the project before submission, but it was good to get that done. This second submission should hopefully be a pretty big improvement on the first. Looking forward to seeing where this project, and others, go next. I*