Software Design and Development Mini Major Work

# Documentation

## Data Dictionary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Item** | **Data Type/Data Structure** | **Format** | **Number of Bytes Required for Storage** | **Size for Display** | **Description** | **Example** | **Validation** |
| Org\_ID | Integer | NNNN | Usually 1-2 | Not displayed | The ID associated with the organisation | 487 | Is an integer |
| Email/User\_ID | String | XXXXXXXX@XXXXXXXXXX | Max 317 | Max 317 | The user’s email | test1@clarity.com | No more than 317 characters  Contains @ symbol within first 65 characters  First and last character not special symbol |
| Group\_ID | Integer | NNNN | Usually 2 | Not displayed | The ID associated with a given group | 573 | An integer  Not any other group ID in org |
| Chat\_ID | Integer | NNNN | Usually 1-2 | Not displayed | The ID associated with a given chat | 654 | An integer  Not any other chat ID in group |
| Message | Record | ID: NNNN  Type: XXXXXXXXXX  Group\_ID: NNNN  Chat\_ID: NNNN  Signature: None  Timestamp: NNNNNNNNNN.NNNNNN  Sender\_ID: XXXXXXXX@XXXXXXXXXX  Content: {Text: XXXXXXXXX} | Varies greatly with message type as content changes | Varies greatly with message type as content changes | A record for any chat message or alert | N/A | N/A |
| Encrypted Message | Record | Encrypted Message: BBBBBBBB...BBBBBBBB(Arbitrarily many bytes)  AES\_Encrypted: BBBBBB...BBBBBBBB(256 *bytes*)  Nonce: BBBBBBBB...BBBBBBBB (16 bytes)  Tag: BBBBBBBB...BBBBBBBB (16 bytes)  Signature: BBBBBBBB...BBBBBBBB (32 bytes) | Variable, depending on size of message | Not displayed | The message encrypted with AES + AES info (key encrypted in RSA) so that only the desired recipient can read the message. Think like E2E encryption. | N/A | AES decrypt\_and\_verify function,  RSA verify function |
| Chat | Record | ID: NNNN  Name: XXXXXXXXXXXXXXXXXXXX  Group\_ID: NNNN  Messages: [Message, Message, Message...]  Alerts: [Alert, Alert, Alert...]  Unread\_Count: NN | Variable, based on number and type of messages and alerts | Varies greatly with number and size of messages – uses a scrollview to contain arbitrarily large chats | A chat within a group | N/A | N/A |
| Group | Record | ID: NNNN  Name: XXXXXXXXXXXXXXXXXXXX  Chats: [Chat, Chat, …, Chat]  Participant\_IDs: [XXXXXXXX@XXXXXXXXXX, XXXXXXXX@XXXXXXXXXX, …, XXXXXXXX@XXXXXXXXXX] | Variable, based on number and size of chats, and number of participant IDs | Max 20+20\*Number of chats | A group that a member stores locally because they belong to it | N/A | N/A |
| Member | Record | Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  Email: XXXXXXXX@XXXXXXXXXX  ID: XXXXXXXX@XXXXXXXXXX  Privilege Status: XXXXXX  Org\_ID: NNNN  Inbox: [Encrypted Message, Encrypted Message, …,Encrypted Message]  RSA\_PK: BBBBBBBB BBBBBBBB ... BBBBBBB (256 bytes – not bits)  RSA\_SK\_bundle: RSA SK Bundle  Verification: BBBBBBBB ... BBBBBBBB (32 bytes) | Variable, mainly based on number and size of encrypted messages | Not displayed | A record for a member stored in the database for everyone to access certain properties.  Note currently, Email = ID and Status=’Member’ always. This may not be true in future versions | N/A | N/A |
| Member Private | Record | Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  Email: XXXXXXXX@XXXXXXXXXX  RSA\_PK: rsa module RSA PK  AES:  RSA\_SK\_Bundle: RSA SK Bundle  Org\_ID: Org\_ID  Inbox: [Encrypted Message, Encrypted Message, …, Encrypted Message]  Groups: {NNNNNNNN: Group,  NNNNNNNN:  Group\_ID | Variable, depending on the number and size of encrypted messages | Not displayed | A record for a member containing private, decrypted information such as the user’s private key. | N/A | N/A |

## Notes on Internal and Intrinsic Documentation

While the code is mostly well documented, using comments, reStructuredText docstrings where the developer found it necessary and descriptive variable names, some further clarification is beneficial to support the future maintenance of the project.

For one, many variables use the terms ‘member’, ‘user’, ‘participant’, ‘recipient’, ‘sender’, ‘verifier’ and ‘verifiee’, among others, to refer to users of the program with different scopes. This is especially true in the case of IDs, where these names may all refer to the same thing but are changed for clarity. Generally:

* *Member* is used when referring to a user in a specific organisation
* *User* is used when referring to the user currently signed in
* *Participant* is used when referring to a user in a specific group
* *Recipient* is used when referring to a recipient of a given message
* *Sender* is used when referring to the user sending the message
* *Verifier* is used when referring to a user who is verifying another user
* *Verifiee* is used when referring to a user who is verifying another user.

Further, there are two variable names that may be confusing. Generally, the name *user\_info* refers to the output of the Auth0 sign in, while *user\_info\_2* refers to the Member\_private object created for the user. While these variable names are not perfectly clear, they are now clarified in this documentation, and the other variables across the app are much more meaningful in name.

# Social and Ethical Considerations

In the production of the application and the graphical user interface, various social and ethical considerations have been carefully considered.

## Graphical User Interface

The GUI of the application has been constructed in a way to facilitate a simple user experience. Over the course of the project, various changes have been made, including removing confusing or tedious screens such as the ‘add members to organisation’ screen and changing from two ‘enter app’ buttons to one, which made it clearer for new users to understand. Colour schemes have been chosen specifically to be clear and consistent, input control widgets such as ‘text boxes’, ‘buttons’ and ‘checkboxes’ have been implemented to make the user interface more intuitive and easier to use.

Some of the GUI (for example the Auth0 sign up screen) was not able to be changed, as it was built into the modules that I had been using; specifically Auth0 and Kivy. Further, some clarity in the GUI was compromised in order to ensure that security requirements were met.

## Privacy and Security as an Ethical Consideration

A significant amount of effort has been expended to ensure that the application is secure. Firstly, the use of cryptographic algorithm standards such as RSA, AES and SHA256 made a significant effort to ensure that messages were sent securely, sensitive information was not accessible by other users and to prevent malicious users from faking a legitimate user. Further, the password-safe hashing algorithm SCrypt was originally used to store passwords safely, but a decision was ultimately made to instead outsource the password management to Auth0, as a result of the developer’s acknowledgement of their limited experience in information security.

There are some security vulnerabilities that are present for this prototype, however they are largely specific to this prototype, and would need to be changed in the creation of future prototypes and the final application. The first vulnerability involves the use of Pickle files, which are known to be unsafe. They were used as they are incredibly simple to create and open, which was necessary as the application is already incredibly large. However, Pickle is only used store the data as a file. When the data is ultimately stored in a database instead, different methods will be required to store it. Another issue was in the storing of the AES key as plaintext on the user’s device, which is only an issue if a single user is logged in on multiple devices. Ultimately, when the storage systems get upgraded in future prototypes, the AES key will be stored using the secure information storage of the OS (in the case of MacOS, this is ‘Keychain’). This is the approach that WhatsApp uses, so it is clearly viable. Besides these two issues, the developer isn’t aware of any significant security vulnerabilities, and the security considerations put in place for this prototype will serve as a starting point for the security considerations that will ultimately be in the final application.

# Testing

## Module/Unit Testing

The purpose of Module/Unit Testing is to isolate the core functionality and ensure that it works, so that when it is integrated into the main program, it can run without bugs. As it was infeasible from a time perspective to completely test everything, the ‘auxiliary’ module, which provide the foundation of the program, were tested. Specifically, unit testing has been performed in the following module, for the following functions:

* Auxiliary
  + Binary Search
  + Binary Search Value
  + Linear Search
  + Mixed Search
  + Bubble Sort

Note that testing invalid inputs was not generally necessary, as sometimes the functions were only called in specific cases when they always had correct inputs (i.e. data validation occurred elsewhere). Invalid inputs were tested during ‘live’ testing.

Further, note that brief testing of other modules did occur in the unit testing stage, in order to confirm that they worked generally. However, it was infeasible to test them with the same level of rigour due to time constraints. More extensive testing of various modules occurred during ‘integration testing’.

### Auxiliary Module Testing

#### Interpretation of Test

The tests in the auxiliary module were largely successful, with few errors occurring. This demonstrates that, for the most part, the core modules work as expected. However, there were two specific cases where the testing came up:

1. In the Binary search value function, the function wasn’t able to correctly process a ‘Binary Search’ result of –1. This error has not been fixed because in the main program, the value of Strict = True is always applied to Binary Search value, which means that –1 is never going to be obtained. However, this error is documented here to ensure that the function is fixed prior to using it in different applications.
2. In the Mixed search function, there were errors when the value was in the unsorted part of the array. The process returned the index in that subsequence of the array, rather than the index in the entire array. The process was fixed to ensure that the correct index is returned.

#### Test Data used and Expected Results

*Binary Search*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array | Item | Key | Strict | Expected Result |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | False | -1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | True | ValueError: Value not found in array |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | False | 0 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | True | 0 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 2 |  | False | 1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 3 |  | True | 2 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 4 |  | False | 3 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 5 |  | True | 4 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | False | 4 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | True | ValueError: Value not found in array |
| |  | | --- | | -1 | | 2 | | -3 | | 4 | | 5 | | 4 | Lambda n:n\*\*2 | False | 1 |
| |  |  | | --- | --- | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *2D Array* | 6 | Lambda item: item[1] | False | 2 |
| |  |  | | --- | --- | | *Name* | *Age* | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *Array of Records* | 7 | Lambda item: item.age | False | 3 |
| |  | | --- | | *5* | | *3* | | *4* | | *2* | | 5 |  | True | ValueError: Value not found in array |

*Binary Search Value*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array | Item | Key | Strict | Expected Result |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | False | Error |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | True | ValueError: Value not found in array |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | False | 1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | True | 1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 2 |  | False | 2 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 3 |  | True | 3 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 4 |  | False | 4 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 5 |  | True | 5 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | False | 5 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | True | ValueError: Value not found in array |
| |  | | --- | | -1 | | 2 | | -3 | | 4 | | 5 | | 4 | Lambda n:n\*\*2 | False | 2 |
| |  |  | | --- | --- | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *2D Array* | 6 | Lambda item: item[1] | False | |  |  | | --- | --- | | Tim | 6 | |
| |  |  | | --- | --- | | *Name* | *Age* | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *Array of Records* | 7 | Lambda item: item.age | False | |  |  | | --- | --- | | *Name* | *Age* | | Sam | 7 | |

*Linear Search*

|  |  |  |  |
| --- | --- | --- | --- |
| Array | Item | Key | Expected Result |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | ValueError: Value not found in array |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | 0 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 2 |  | 1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 3 |  | 2 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 4 |  | 3 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 5 |  | 4 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | ValueError: Value not found in array |
| |  | | --- | | -1 | | 2 | | -3 | | 4 | | 5 | | 4 | Lambda n:n\*\*2 | 1 |
| |  |  | | --- | --- | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *2D Array* | 6 | Lambda item: item[1] | 2 |
| |  |  | | --- | --- | | *Name* | *Age* | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *Array of Records* | 7 | Lambda item: item.age | 3 |
| |  | | --- | | *5* | | *3* | | *4* | | *2* | | 5 |  | 0 |

*Mixed Search*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array | Item | Key | Len\_sorted | Expected Result |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 7 |  | 4 | 3 |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 3 |  | 4 | 1 |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 3 |  | 2 | 1 |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 2 |  | 4 | 4 |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 8 |  | 4 | 5 |

*Bubble Sort*

|  |  |  |  |
| --- | --- | --- | --- |
| Array | Key | Max Passes | Expected Result |
| |  | | --- | | -50 | | 1 | | 83 | | 264 | |  |  | |  | | --- | | -50 | | 1 | | 83 | | 264 | |
| |  | | --- | | -50 | | 1 | | 83 | | 264 | |  | 1 | |  | | --- | | -50 | | 1 | | 83 | | 264 | |
| |  | | --- | | 264 | | 1 | | 83 | | -50 | |  |  | |  | | --- | | -50 | | 1 | | 83 | | 264 | |
| |  | | --- | | 264 | | 1 | | 83 | | -50 | |  | 1 | |  | | --- | | -50 | | 264 | | 1 | | 83 | |
| |  | | --- | | 264 | | 83 | | 1 | | -50 | |  |  | |  | | --- | | -50 | | 1 | | 83 | | 264 | |
| |  | | --- | | 1 | | 1 | | 4 | | 2 | |  |  | |  | | --- | | 1 | | 1 | | 2 | | 4 | |
| |  |  | | --- | --- | | Clive | 47 | | Clara | 76 | | Barbara | 33 | | Barry | 47 |   *2D Array* | Lambda item:item[0] |  | |  |  | | --- | --- | | Barbara | 33 | | Barry | 47 | | Clara | 76 | | Clive | 47 | |
| |  |  | | --- | --- | | Clive | 47 | | Clara | 76 | | Barbara | 33 | | Barry | 47 |   *2D Array* | Lambda item:item[1] |  | |  |  | | --- | --- | | Barbara | 33 | | Clive | 47 | | Barry | 47 | | Clara | 76 | |
| |  |  | | --- | --- | | *Name* | *Age* | | Clive | 47 | | Clara | 76 | | Barbara | 33 | | Barry | 47 |   *Array of Records* | Lambda item:item.age |  | |  |  | | --- | --- | | *Name* | *Age* | | Barbara | 33 | | Clive | 47 | | Barry | 47 | | Clara | 76 | |

#### Actual Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array | Item | Key | Strict | Actual Result |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | False | -1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | True | ValueError: Value not found in array |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | False | 0 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | True | 0 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 2 |  | False | 1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 3 |  | True | 2 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 4 |  | False | 3 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 5 |  | True | 4 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | False | 4 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | True | ValueError: Value not found in array |
| |  | | --- | | -1 | | 2 | | -3 | | 4 | | 5 | | 4 | Lambda n:n\*\*2 | False | 1 |
| |  |  | | --- | --- | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *2D Array* | 6 | Lambda item: item[1] | False | 2 |
| |  |  | | --- | --- | | *Name* | *Age* | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *Array of Records* | 7 | Lambda item: item.age | False | 3 |
| |  | | --- | | *5* | | *3* | | *4* | | *2* | | 5 |  | True | ValueError: Value not found in array |

*Binary Search Value*

|  |  |  |  |
| --- | --- | --- | --- |
| Array | Item | Key | Actual Result |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | 5 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | 0 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 2 |  | 1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 3 |  | 2 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 4 |  | 3 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 5 |  | 4 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | ValueError: Value not found in array |
| |  | | --- | | -1 | | 2 | | -3 | | 4 | | 5 | | 4 | Lambda n:n\*\*2 | 1 |
| |  |  | | --- | --- | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *2D Array* | 6 | Lambda item: item[1] | 2 |
| |  |  | | --- | --- | | *Name* | *Age* | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *Array of Records* | 7 | Lambda item: item.age | 3 |
| |  | | --- | | *5* | | *3* | | *4* | | *2* | | 5 |  | 0 |

*Linear Search*

|  |  |  |  |
| --- | --- | --- | --- |
| Array | Item | Key | Actual Result |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 0 |  | ValueError: Value not found in array |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 1 |  | 0 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 2 |  | 1 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 3 |  | 2 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 4 |  | 3 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 5 |  | 4 |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 |  | ValueError: Value not found in array |
| |  | | --- | | -1 | | 2 | | -3 | | 4 | | 5 | | 4 | Lambda n:n\*\*2 | 1 |
| |  |  | | --- | --- | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *2D Array* | 6 | Lambda item: item[1] | 2 |
| |  |  | | --- | --- | | *Name* | *Age* | | Bob | 2 | | Ava | 4 | | Tim | 6 | | Sam | 7 |   *Array of Records* | 7 | Lambda item: item.age | 3 |
| |  | | --- | | *5* | | *3* | | *4* | | *2* | | 5 |  | 0 |

*Mixed Search*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array | Item | Key | Len\_sorted | Actual Result |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 7 |  | 4 | 3 |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 3 |  | 4 | 1 |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 3 |  | 2 | 1 |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 2 |  | 4 | 0 |
| |  | | --- | | 1 | | 3 | | 5 | | 7 | | 2 | | 8 | | 8 |  | 4 | 1 |

*Bubble Sort*

|  |  |  |  |
| --- | --- | --- | --- |
| Array | Key | Max Passes | Actual Result |
| |  | | --- | | -50 | | 1 | | 83 | | 264 | |  |  | |  | | --- | | -50 | | 1 | | 83 | | 264 | |
| |  | | --- | | -50 | | 1 | | 83 | | 264 | |  | 1 | |  | | --- | | -50 | | 1 | | 83 | | 264 | |
| |  | | --- | | 264 | | 1 | | 83 | | -50 | |  |  | |  | | --- | | -50 | | 1 | | 83 | | 264 | |
| |  | | --- | | 264 | | 1 | | 83 | | -50 | |  | 1 | |  | | --- | | -50 | | 264 | | 1 | | 83 | |
| |  | | --- | | 264 | | 83 | | 1 | | -50 | |  |  | |  | | --- | | -50 | | 1 | | 83 | | 264 | |
| |  | | --- | | 1 | | 1 | | 4 | | 2 | |  |  | |  | | --- | | 1 | | 1 | | 2 | | 4 | |
| |  |  | | --- | --- | | Clive | 47 | | Clara | 76 | | Barbara | 33 | | Barry | 47 |   *2D Array* | Lambda item:item[0] |  | |  |  | | --- | --- | | Barbara | 33 | | Barry | 47 | | Clara | 76 | | Clive | 47 | |
| |  |  | | --- | --- | | Clive | 47 | | Clara | 76 | | Barbara | 33 | | Barry | 47 |   *2D Array* | Lambda item:item[1] |  | |  |  | | --- | --- | | Barbara | 33 | | Clive | 47 | | Barry | 47 | | Clara | 76 | |
| |  |  | | --- | --- | | *Name* | *Age* | | Clive | 47 | | Clara | 76 | | Barbara | 33 | | Barry | 47 |   *Array of Records* | Lambda item:item.age |  | |  |  | | --- | --- | | *Name* | *Age* | | Barbara | 33 | | Clive | 47 | | Barry | 47 | | Clara | 76 | |

## Integration Testing

Following unit testing, and before program testing, the interaction of different modules were tested through another program, which simulated the backend procedures that would occur during the process of testing. This was also a way to test a wide number of modules for significant issues, where it would be infeasible to perform extensive unit testing on all functions. According to [Atlassian’s guide](https://moriahnsweduau-my.sharepoint.com/:w:/r/personal/adam_gottlieb_student_moriah_nsw_edu_au/Documents/Software%20Design%20and%20Development%20Mini%20Major/Software%20AT3%20Report.docx?d=w31b675e528964eb2a58a4ff2fdbda961&csf=1&web=1&e=6h0yc2), this is probably best described as ‘Integration Testing’.

The following processes are used to assist the integration testing, which allowed the tests to test all paths through the program:

1. **Sign Up Organisation**

* Member Constructor - Create a new member with random name and email
* Sign Up -> Sign Up Org w/ first member ^
* Sign In -> Sign In New V2 (That member)
* Complete *Sign Up and Join Existing Organisation* flow for this organisation
  1. **Sign Up and Join Existing Organisation (close to the actual flow)**
* Member Constructor - Create a new member with random name and email
* Create Random 8 character code = CODE and hash w/ SHA256 = HASH
* Validate User -> Upload Verification Hash
* Sign In -> Sign In Existing V2 (ANY existing member of that organisation)
* Validate User -> Verify User (That existing member verifies new member with CODE)
* Sign In -> Sign In Existing V2 (The NEW member)
* Complete *Member Creates Group in their Organisation* flow for both members
  1. **Copy Member (more efficient, so I can run more tests in 3,4,5, but not secure for app)**
* Sign In -> Sign In Existing V2 (ANY existing member)
* Member Constructor -> Create a new member with random name and email
* Manage Users -> Complete User (New member, with info from existing members)

1. **Member Creates Group in their Organisation**

* Sign In -> Find User (The Member)
* Sign In -> Sign In Existing V2 (The Member)
* Data Structures -> Member Private -> Process Inbox
* Chat Functionality V2 -> Create Group (Use random subset of members as participants)
* Data Structures -> Member Private -> Add Group
* Data Structures -> Member Private -> Process Inbox

1. **Member Creates Chat in Random Group**

* Sign In -> Find User (The Member)
* Sign In -> Sign In Existing V2 (The Member)
* Data Structures -> Member Private -> Process Inbox
* Chat Functionality V2 -> Create Chat (Any of their groups at random)
* Data Structures -> Member Private -> Update Group
* Data Structures -> Member Private -> Process Inbox

1. **Member Sends Message in Random Chat**

* Sign In -> Find User (The Member)
* Sign In -> Sign In Existing V2 (The Member)
* Data Structures -> Member Private -> Process Inbox
* Send -> Send Message
* Data Structures -> Member Private -> Process Inbox

The flow began with the *Sign Up Organisation* process. After this process was completed, A process was chosen at random, with a random user if necessary. The number of times each process was completed was logged, which allowed for the calculation of the exact number of organisations, users, groups, chats and messages that should exist. This can be compared with the actual quantities to determine if any errors are present, and provide guidance on where the errors are.

Note that due to the complexity in how the groups, chats and messages are stored, the actual numbers were calculated using a method that would fail if the following condition was not met: A has a group named ‘X’ with B implies that B has a group named ‘X’ with A. In theory, this should be the case, but if there are errors in how the groups and chats are stored or how messages are sent, this may not be the case.

## Integration Testing Results Examples and Evaluation

Initially, there were some errors in the main program and in the testing simulation program which did not allow the integration tests to run to completion. These were fixed and the tests were rerun, yielding the following results:

*Expected values: {'organisations': 2, 'users': 10, 'groups': 17, 'chats': 22, 'messages': 5}*  
*Actual values: {'organisations': 2, 'users': 10, 'groups': 16.0, 'chats': 21.0, 'messages': 5.0}*

As there was one less group and chat, it was most likely that the error occurred in the ‘user creates group’ flow, which creates a group and a chat.

The error message also stated ‘Recipient ID not found in list of members.’

By checking for the existence of this error message in all functions called from the ‘user creates group’ flow, the error was found. Effectively, the program was still running a binary search when the data structure had been updated to only be partially sorted, and therefore only support a mixed search or linear search. The algorithm was updated to include the mixed search. Therefore, through the process of

File "/Users/adam.gottlieb/PycharmProjects/Software\_Mini\_Major\_Project/tests/integrated\_testing.py", line 2246, in <module>

try: master\_flow()

File "/Users/adam.gottlieb/PycharmProjects/Software\_Mini\_Major\_Project/tests/integrated\_testing.py", line 2040, in master\_flow

sign\_up\_org\_flow()

File "/Users/adam.gottlieb/PycharmProjects/Software\_Mini\_Major\_Project/tests/integrated\_testing.py", line 2077, in sign\_up\_org\_flow

sign\_up\_and\_join\_flow(org\_ID = org\_ID)

File "/Users/adam.gottlieb/PycharmProjects/Software\_Mini\_Major\_Project/tests/integrated\_testing.py", line 2123, in sign\_up\_and\_join\_flow

member\_creates\_group\_flow(existing\_member\_info.email)

File "/Users/adam.gottlieb/PycharmProjects/Software\_Mini\_Major\_Project/tests/integrated\_testing.py", line 2157, in member\_creates\_group\_flow

group = chat\_functionality\_v2.create\_group(

File "/Users/adam.gottlieb/PycharmProjects/Software\_Mini\_Major\_Project/user/chat\_functionality\_v2.py", line 31, in create\_group

user.send.send\_message(org\_ID, recipient\_IDs, sender\_rsa\_sk, invitation)

File "/Users/adam.gottlieb/PycharmProjects/Software\_Mini\_Major\_Project/user/send.py", line 7, in send\_message

encrypted\_messages.append(message.encrypt(rsa\_pks[recipient\_ID], sender\_rsa\_sk))

KeyError: 'Britte [Siss@Zoonoodle.com](mailto:Siss@Zoonoodle.com)'

*Expected values: {'organisations': 36, 'users': 103, 'groups': 170, 'chats': 195, 'messages': 27}*  
*Actual values: {'organisations': 36, 'users': 103, 'groups': 167.5, 'chats': 192.5, 'messages': 27.0}*

From the error message, it was determined that Britte Siss@Zoonoodle.com was not found in the organisations list. It was unclear why this error was occurring and thus, a short script was created to output the names of each organisation and everyone in them. From this, it was found that there were two organisations with the same organisation name ‘Zoonoodle’. This can be handled by the app, but not adequately by the testing script. Therefore, the test data was updated to make sure that the same organisation name didn’t appear more than once. To reiterate, this is okay because the actual application does not use the organisation names to find the organisation, and Auth0 (which wasn’t tested during integration testing) handles all of the files, so this issue is only because of the way that the testing is set up.

The process of integration testing was continued, where the expected values were consistently compared with the actual values, until they matched consistently. Various other debugging tools were created as needed, such as an automated log of all main processes that occurred, and outputs of all organisation data in the final state. *Through this process, the evaluation of almost all processes in the main application, and all paths through the application, were tested in a way that would have been infeasible in module/unit testing in the time available.*

Much of the integration testing process was recorded, and recordings can be found in Appendix 3.

## Program Testing

Following integration testing, program testing occurred, where the GUI was run and tested. Specifically, the following processes were conducted in succession, and the results obtained:

|  |  |  |  |
| --- | --- | --- | --- |
| Action | Input | Expected Output | Actual Output |
| Sign Up/ Sign In | Email = [‘alice@clarity.com’](mailto:‘alice@clarity.com’)  Password = Password1234 | Redirected to Welcome Screen | Redirected to Welcome Screen |
| Create New Organisation | Name = ‘Alice’  Organisation Name = ‘Clarity’ | Redirected to ‘Success’ Screen | Redirected to ‘Success’ Screen |
| Sign Out |  | Close App | Close App |
| Sign Up/ Sign In | Email = [‘bob@clarity.com’](mailto:‘bob@clarity.com’)  Password = Password1234 | Redirected to Welcome Screen | Redirected to Welcome Screen |
| Join Existing Organisation | Name = ‘Bob’ | Redirected to ‘Get Verified’ Screen  Random 8-Character code output on screen | Redirected to ‘Get Verified’ Screen  Random 8-Character code output on screen |
| Sign Out |  | Close App | Close App |
| Sign Up/ Sign In | Email = ['alice@clarity.com’](mailto:‘charlie@clarity.com’)  Password = Password1234 | Redirected to Home Screen | Redirected to Home Screen |
| Verify Member | Email = [‘charlie@clarity.com’](mailto:‘bob@clarity.com’)  Code = AAAAAAAA | Error Message – Incorrect email or code | Error Message – Incorrect email or code |
| Verify Member | Email = [‘charlie@clarity.com’](mailto:‘charlie@clarity.com’)  Code = [Bob’s code] | Error Message – Incorrect email or code | Error Message – Incorrect email or code |
| Verify Member | Email [‘bob@clarity.com’](mailto:‘bob@clarity.com’)  Code = AAAAAAAA | Error Message – Incorrect email or code | Error Message – Incorrect email or code |
| Verify Member | Email [‘bob@clarity.com’](mailto:‘bob@clarity.com’)  Code = [Bob’s Code] | Return to Home Screen | Return to Home Screen |
| Sign Out |  | Close App | Close App |
| Sign In | Email [‘bob@clarity.com’](mailto:‘bob@clarity.com’)  Password = Password1234 | Redirect to Home Screen | Redirect to Home Screen |
| Create Group | Name = ‘’  Participants = [alice@clarity.com’] | Error Message – Invalid Name | Created chat with blank name |
| Create Group | Name = ‘Test 1’  Participants = [‘charlie@clarity.com’](mailto:‘charlie@clarity.com’) | Error Message – Participant Doesn’t Exist | Error Message – We couldn’t find any members with the following emails:  *charlie@clarity.com* |
| Create Group | Name = ‘Test 2’  Participants = alice@clarity.com’ | Return to Home Screen with Group Created |  |
| Select Group | Group = ‘Test 2’ | Show ‘Main’ under chats |  |
| Create Chat | Name = ‘New Chat’ | Return to Home Screen with second Chat in group |  |
| Select Chat | Chat = 'New Chat’ | Nothing, but importantly no error should arise. |  |
| Send message | Message = ‘Hey Ava! Super keen for this Friday 😀😀’ | Email and Message appear on screen | All works except emojis don’t appear correctly |
| Sign Out |  | Exit App |  |
| Sign In | Email = [‘alice@clarity.com’](mailto:‘alice@clarity.com’)  Password = Password1234 | Redirect to Home Screen, and group ‘Test 2’ should be there | Group 'Test 2’ was not there |
| Select Group | Group = ‘Test 2’ | 2 Chat buttons pop up | Test couldn’t be conducted |
| Select Chat | Chat = ‘New Chat’ | Message from bob@clarity.com should be on screen | Test couldn’t be conducted |

From the tests, it appears that most of the application works as intended. There was an unresolved issue with data validation of a special-case input (‘’). Unfortunately, due to time constraints, this error could not be fixed. However, there is no fundamental issue with having a chat with no name – it is a strange user experience, but ultimately not a broken one.

The second issue occurred with the display of special characters (namely, emojis), and occurred due to Kivy’s font not being able to render the emojis. This is unfixable given the issue lies with Kivy and not the developer, however again it is not a fundamental issue – it merely detracts slightly from the user experience.

The third issue was significantly more fundamental and important – the inbox of the other user didn’t update when a message was sent. This was strange as the systems worked as intended in the integration testing, so it appeared that there was a problem in the GUI. However, to test this, the integration testing was run again, which yielded negative results, despite having worked previously. Debugging output statements were used extensively to determine that it was some additional code written in order to produce clear error messages that was causing the issue. After fixing it, the developer reran the integration tests and performed some more program testing, which were both successful.

## System Testing

The process of system testing involved testing the application using an online IDE and virtual machine. There were some issues involving the inability to find certain files, but these were eventually resolved. This test demonstrates that the project should be compatible with various different systems.

## Live Testing

The process of live testing involved two professionals at a company named ‘BizCover’ accessing the application. With some guidance, they were able to navigate through the application. They found one bug, namely that it was possible to try and create a chat that’s not in a group, and try send a message that’s not in a chat. In both cases, the program will crash. This unfortunately was not able to be fixed within the time frame of the project.

In addition to this, their feedback was as follows:

* The buttons at the bottom of the screen are intrusive and probably should be placed in a menu
  + This feedback was agreed upon by the developer. However, attempts had been made to make this change, and none succeeded. This is due to a lack of familiarity with ‘Kivy’.
* Members with an email with the same domain name as an existing organisation should automatically be added to that organisation
  + That feature was considered by the developer, but as there is no verification email this was deemed unsafe. It would allow for people to use an email address associated with an organisation, even if it didn’t exist or did not belong to them, to access an organisation they were not a part of. Therefore, the feature was decided against.
* The buttons should changed colour to indicate when they have been selected.
  + This was some good feedback, but unfortunately did not seem viable to implement in the time permitted. However, this feedback is retained for future versions of the application.

In general, they were impressed by the design of the app and the various considerations that had gone into the creation of the app.

# Summary of Discussions with Supervisor

In the production of the project, a variety of discussions with the project supervisor took place. These included:

* Progress Report 1 and feedback
* An email thread in relation to the changing of project requirement definitions
* Progress Report 2 and feedback
* An email thread in relation to the production of documentation

In the first progress report and email thread, significant concern over the feasibility of the project was expressed. In the email, it was proposed to the supervisor to simply create the interface rather than the full application. The supervisor accepted this idea and suggested possibly to allow for multiple logins on the same device to chat with each other – an idea that was ultimately included in the final project. Similar feedback was provided on the progress report. See this report in Appendix 1.

In the second progress report, the supervisor was informed on the progress that had been accomplished and responded with satisfaction. The supervisor provided some reminders over the requirements for testing and the documentation. See this report in Appendix 2.

In the second email thread, the supervisor was asked about the requirement of the data dictionary, as many data types and data structures in the application were significantly more complicated than those presented in the sample data dictionary, involving nested/recursive arrays of records. The supervisor advised to not worry about capturing this complexity in the data dictionary, and rather just providing a simple summary of the relevant data types and data structures.

# A Note on Generative CASE Tools

Over the course of this project, various CASE tools have been used. Some of these CASE tools are generative in nature, including ChatGPT, Bing Chat and Tabnine.

It must be emphasised that neither ChatGPT nor Bing Chat generated any code in the final product that is not very explicitly marked with a comment or docstring saying how it was generated. Their input was mostly in explaining concepts and modules, providing feedback and occasionally debugging certain functions.

Tabnine *did* generate some of the code, even where not marked explicitly. However, it is important to consider that Tabnine is a code *completion* tool – it’s job is to save time by completing lines of code, not writing entire sections of code by itself. In this project, Tabnine was used *solely* for this function. There is no line of code that Tabnine wrote which was not similar to what the developer would have wrote without it, and no line of code that Tabnine wrote which the developer could not explain. It is simply a tool for saving time. It also did not write more than probably 3 lines of code at a time, and the majority of the code in the application is written by the developer.

# Evaluation

In this project, the project requirements were met to a moderate extent. The original specifications of the requirements definitions can be found in Appendix 4, which is colour-coded according to the extent each requirement was met. Overall, the basic requirements of a messaging application were met to a significant extent. In general, the application functions like a messaging app, despite a few small gaps in functionality. However, while no requirement was specified that the users needed to be able to communicate from different devices, this should have been implied by the notion that it is a messaging app. Ultimately the project requirements were reduced in March to not require this, rendering the application as a prototype rather than a full application. However, many functions have been programmed to facilitate this behaviour, so including them in future evolutionary prototypes and ultimately the final application should not be difficult.

The application-specific features that solved the problem were ultimately not complete, due to time constraints. However, they should not be difficult to implement given that the first evolutionary prototype has been created, and will be implemented in future prototypes.

The performance requirements were met, but could still be improved in future versions. This could be achieved through the use of a faster encryption algorithm compared to RSA for encrypting and decrypting of messages – possibly Elliptic Curve Diffie Hellman or similar. Overall, however, the speed of the application is very reasonable and meets the requirements set out at the beginning.

The social and ethical considerations regarding the user interface were mostly met. However, the planned design of the home screen was not met due to some challenges experienced when using Kivy, and thus is less aesthetically pleasing than desired. This will be improved for future versions. Further, error messages were consistent and clear, and user input was generally simple, with the exception of the ‘Create Group’ screen, which was intended to be checkboxes but ultimately became input boxes. In addition, many changes were made to the original plan along the way in order to enhance the user interface. Unfortunately, ensuring inclusivity for all people was made very challenging, in part due to time constraints and in part due to Kivy’s notorious lack of compatibility with screen reader applications. With time, it is expected that the available tools available with Kivy will expand to support more issues regarding inclusivity.

The social and ethical issues regarding safety and security were met to a great extent. Various measures were taken to ensure that user accounts were secure, messages were end-to-end encrypted and phishing attacks were largely infeasible. As mentioned in the relevant section of the report, there are some insecure aspects of the project, but these are highly specific to the implementation details of the first prototype and will naturally be replaced in future prototypes. For the purposes of a prototype, the safety and security was a sufficient proof of concept.

# Appendix 1: Progress Report #1

Introduction

The following report outlines the progress which has been made on the development of an application for the Software Design and Development Mini Major Project. The report also outlines various problems that have arisen in the development process, indicating to what extent and how each has been resolved. Finally, a plan moving forward is included in the conclusion.

## Progress Made

Over the past three months, some progress has been made. For one, the authentication system has been implemented in a reasonably secure manner. The system isn’t quite complete, as it is impossible to completely implement the system without having implemented the cross-device communication aspect of the product, but when that is complete it will be trivial to complete the authentication mechanisms. Further, data structures to store key information such as members, organisations and groups have been created. In addition, many auxiliary algorithms have been programmed in order to assist the further development of code. Finally, a start has been made on the development of the cross-device communication systems, but these have proven to be a significant challenge and thus have not been implemented completely.

## Challenges Faced

Throughout the software development process, many challenges have been faced. One such challenge has been the challenge of encryption for the authentication mechanisms. The challenge arose for the most part due to the encryption algorithms’ use of binary data types, which the developer was not previously familiar with. However, after some online research and experimentation, the developer succeeded in implementing the encryption algorithms successfully in a relatively secure manner.

Another challenge that arose was in relation to encrypting messages, and specifically the speed of RSA, which initially didn’t seem to be acceptable for the nature of the application. However, after some research, it became clear that the solution was to use AES to encrypt the message and use RSA to encrypt the AES key. This solution was fairly straightforward and though learning to use the Pycryptodome and RSA packages was not easy, it was very manageable with the aid of the internet. This challenge was overcome in its entirety through research and experimentation.

A final challenge that arose was that of sending information between two devices. This is obviously a necessary aspect of a messaging application. Despite the developer’s lack of knowledge in the area, overcoming previous challenges had given the developer confidence that it would come right through research and experimentation. However, upon doing research, the task only appeared to grow in size. Eventually, it seemed as if it would be required to effectively create a miniature programming language and build an interpreter for that language in Python. A more practical solution was eventually found – namely the use of the XMPP protocol and the python package Slixmpp which implements it. However, a surprising absence of clear documentation or online help has made it very difficult to use. While some progress has been made in this area, it has been incredibly slow and difficult, with little of substance created after approximately seven weeks of work. Despite this unintended complexity, it seems that unfortunately this must be the solution – cross-device communication is essential for a messaging application, and there simply is no alternative. While XMPP is not a friendly solution, it is significantly better than any alternatives that were discovered and so the challenges must be overcome, no matter how difficult they may be.

## Conclusion

In conclusion, some progress has been made on the progress, especially in relation to authentication and encryption. However, the processes for sending information between devices has proven to be exceptionally difficult, and despite significant research and attempts to code various solutions, the task is still far from complete. Over the next few weeks, development on this task will be temporarily suspended in order to focus on the user interface. Hopefully, progress in this aspect of the project will be significantly quicker so that the holiday period can be used to resume work on the messaging systems. The success of the project is dependent on the solving of a problem which has predominantly evaded the developer well over a month, but hopefully it will be attainable.

## Addendum

**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.*

# Appendix 2: Progress Report #2

## Introduction

The following report outlines the progress which has been made on the development of an application for the Software Design and Development Mini Major Project, since the first report. The report also outlines various problems that have arisen in the development process, indicating to what extent and how each has been resolved. Finally, a plan moving forward is included in the conclusion.

As a result of many challenges faced in the previous report, the requirements definition was reduced significantly. While previously, the intention had been to create a fully-functional messaging application, the intention now is to simply create an evolutionary prototype, using files stored locally on a user’s device instead of a server. Further, much of the desired functionality, including the widgets, some group and chat operations and different privileges will not be ready for this prototype, with the intention of including this in the later prototypes.

With these new changes, there has been significant progress. The application is effectively complete. The ‘login’ process has been significantly altered to provide a more secure experience via a third party application, and an effective GUI has been developed, with some small changes still necessary. Finally, functions for sending messages, managing an inbox, verifying members and other forms of communication have been achieved. Overall, there has been significant progress on the project.

## Challenges Faced

One of the primary challenges faced has been in using the GUI module ‘Kivy’. While it was not difficult to learn how to use the module, certain challenges were faced when attempting more complex designs. This was often overcome by citing the documentation of Kivy, which is fairly comprehensive.

One example of this occurred in the development of the ‘Scrollview’ for the ‘Add Members’ page of the GUI. Initially, despite the height of the widget being set, the ‘size\_hint\_y’ was not set to None. The developer assumed that the setting of a specific height would override the size hint, but this was not the case. From isolating the code and running various tests, it became clear that the error was that the height parameter wasn’t being implemented correctly, rather than the ‘add row’ function being implemented incorrectly (which was a plausible possibility before testing). After this, the documentation of the various size parameters was investigated, and the error determined.

Another challenge has been in managing a large code base. As the programming progressed, it has become increasingly evident that the required amount of code would be far greater than initially assumed. This challenge has been navigated by extensively using different modules to separate different components of the code, as well as using functions and classes to clarify the intention of the code.

A final challenge has been time. The amount of work required for this project has been far greater than the amount of work anticipated both at the start of the project, and after the first progress report. However, most of the programming is now complete.

## Conclusion

Overall, the development of the project has been more productive than before the first process report, but has still been slower than intended, as ideally the project would have been finished weeks ago and testing and evaluation would be almost complete at this stage as well. Over the next day or two, I will complete some basic tests on the program and produce some of the necessary documentation (including live help and data dictionary), as well as provide some final updates to my GUI. After that, I will test the program with some potential clients. Following this, some last changes will be made and the prototype signed off as complete.

## Addendum

**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:*

**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.*

# Appendix 3: Testing Video Logs

These testing video logs have been referred to in the body of the work, and in the logbook. They may be considered an additional part of the logbook.

<https://moriahnsweduau-my.sharepoint.com/:f:/g/personal/adam_gottlieb_student_moriah_nsw_edu_au/EhiT2vj0JtBJizOlsQQjYHMBRCnz0wxRrbMPGaKZ0IJ8mA?e=yCPGh0>

# Appendix 4 – Problem Definition with colours in accordance with achievement

Green – achieved

Green Highlighted – changed to improve upon original definitions

Orange – partially achieved, or achieved with known bugs

Red - Incomplete

In many professional settings, messaging applications are used to facilitate asynchronous, text-based communication within teams or entire businesses. The use of these applications has transformed the manner of communication in the workplace, increased productivity and allowed professionals to collaborate without the need to physically meet. However, asynchronous communication through messaging applications has also introduced some communication challenges which were not prevalent before. One such challenge is the challenge associated with receiving many messages and not being able to easily determine which messages are most important or timely. This can greatly hinder the efficiency of a team or company and while often the most important messages are naturally highlighted due to being the most recent, this is not always the case.

The purpose of this project is to create an asynchronous messaging application for professional and organisational communication, especially for organisations engaged in remote work, which partially solves this problem. It will achieve this in part with simple widgets that can reduce the number of messages that need to be sent in common situations, such as for scheduling a meeting. Users may choose to include a widget in a chat, or widgets may automatically replace messages to enhance clarity and declutter the user interface. Further, messages will be organised systematically using channels in a similar manner to Discord. This will significantly reduce the number of messages sent, hence decluttering the user interface.

## Functionality Requirements

In order to be a success, the application must contain all the core functionality of a standard messaging application with threads. This means that a user should be able to complete the following tasks.

* Create an organisation and add a given list of people to it.
* Create a group and add anyone in the organisation to that group.
* View a list of the groups to which they belong.
* Create a channel within a group. *- Renamed to ‘chat’*
* Send a message in a channel to which they belong
* Delete that message later if they choose.
* View messages sent in a channel
* Leave a group.
* Delete a group.

Further, the application must contain specific functionality in order to solve the problem outlined above. This means that the following criteria must be met:

* A group of users can use widgets to easily achieve the following tasks:
  + Schedule a group meeting or event by getting everyone’s availability
  + Hold a vote/survey (public, anonymous or result only)
* These widgets are automatically applied in simple cases, such as when keywords are used in messages, or can be selected by the user.

## Performance Requirements

The application must work efficiently. For the purpose of this project, this means that given a decent internet connection, the user should never have to wait for more than five seconds for any operation to be carried out, whether that be creating a group, sending a message to the server, retrieving a message from the server or any other task. The exception to this is to load the application, which may take up to ten seconds. Ideally, the performance will be significantly better than this, but this is simply a minimum standard for the project.

*This minimum standard was achieved in testing.*

## Compatibility Requirements

The application must be compatible with both MacOS (V10 and above) and Windows (7 and above) operating systems. As this application is meant for professional collaboration, it is not important that the program can be used on mobile devices as almost all established workplaces would have individual laptops.

*The application was ultimately never compiled into a single executable but could in theory be run on the operating systems specified above.*

## Problem Boundaries

While there are many aspects of this project, there are also some components that lie outside of the scope of this project. Specifically, the implementation of a Graphical User Interface Engine is not required, as Kivy will be used. Further, the implementations of encryption and hashing algorithms is not in the scope of this project, and instead existing implementations of the Scrypt, AES and RSA algorithms will be used. This is partially because the task of designing and implementing encryption algorithms is a particularly large task, comparable in size to the rest of the project. However, the main reason is because security is a serious issue that is easily made vulnerable if not implemented properly, and thus is best left to experts in the field.

# Appendix 5: Online Help

Online Help, and the Terms and Conditions, can be found here:

[adamsgottlieb.wix.com/clarity](http://adamsgottlieb.wix.com/clarity)