Peer to Peer Messaging App Evaluation

Connor Cuffe

Beaumaris Secondary College

Contents

[Testing 4](#_Toc178943337)

[Testing Table 4](#_Toc178943338)

[Test loading times 4](#_Toc178943339)

[Test message receive time 4](#_Toc178943340)

[Test high network traffic resilience 4](#_Toc178943341)

[Test data encryption 4](#_Toc178943342)

[Test message encryption 4](#_Toc178943343)

[Usability testing 4](#_Toc178943344)

[Summary 5](#_Toc178943345)

[Evaluation criteria 6](#_Toc178943346)

[Summary 7](#_Toc178943347)

[Functional requirements 9](#_Toc178943348)

[Summary 9](#_Toc178943349)

[Non-Functional requirements 10](#_Toc178943350)

[Summary 11](#_Toc178943351)

[Business requirements 12](#_Toc178943352)

[Summary 12](#_Toc178943353)

[Project management 13](#_Toc178943354)

[Development Model 13](#_Toc178943355)

[Project plan 13](#_Toc178943356)

[Bibliography 14](#_Toc178943357)

# Introduction

## Introduction

This document outlines the software solution's success based upon the evaluation criteria and functional and non-functional requirements as outlined during the design phase. This document also outlines shortcomings of the current design and any future modifications that should be made to improve the user experience and effectiveness of the software solution.

As part of the review of the project, the development model and project plan are evaluated for their effectiveness in implementing the software solution so as to improve future projects.

# Testing

## Testing Table

Testing tables are used to show the results of any tests as well as fixes that may have been put in place due to the outcome of the tests. The results of the tests are then used to support the evaluation of the project success.

| Test | Description | Expected result | Actual result | Success | Fix (If applicable) | Tester | Date |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test loading times | Testing the loading times of each screen | All less than one second loading time. | All less than one second loading time. |  | Not applicable | Connor Cuffe | 22-Sept-2024 |
| Test message receive time | Testing the time between sending a message and it being received | Less than five seconds | Less than five seconds required for opening and closing of chat screen |  | Add a Reload page button to chat screen | Connor Cuffe | 8-Sept-2024 |
| Test high network traffic resilience | Test that software solution can handle high network traffic | No impact with more than 50 messages per hour | No impact with more than 50 messages per hour |  | Not applicable | Connor  Cuffe | 20-Sept-2024 |
| Test data encryption | Tests that the user data is successfully encrypted. See Test\_Encrypt\_data  in project unit tests | All unit tests successful | All unit tests successful |  | Not applicable | Connor Cuffe | 27-July-2024 |
| Test user data store and load | Tests that the user data is successfully stored and loaded. See test\_user\_data\_store\_load in unit tests | All unit tests successful | Data is incorrectly encrypted expected 48, 16, 115, 48, 83  instead got **4328, 1236, 1135, 4832, 2383** |  | The private keys were stored inversely after being loaded in from file – fixed by reversing direction. | Connor Cuffe | 29-July-2024 to 11-August-2024 |
| Test user data store and load | Tests that the user data is successfully stored and loaded. See test\_user\_data\_store\_load in unit tests | All unit tests successful | All unit tests successful |  | Not applicable | Connor Cuffe | 12-August-2024 |
| Test message encryption | Tests that messages are correctly encrypted. See: Test\_message\_encrypt in unit tests | All unit tests successful | All unit tests successful |  | Not applicable | Connor Cuffe | 7-July-2024 |
| Test network messaging | Tests the parsing and formatting of messages for network communications See test\_network\_messaging in unit tests | All unit tests successful | Parsing crashes as expected list[int] instead got str |  | Check if the message is unencrypted if so skip decryption | Connor Cuffe | 19-August-2024 |
| Test network messaging | Tests the parsing and formatting of messages for network communications See test\_network\_messaging in unit tests | All unit tests successful | All unit tests successful |  | Not applicable | Connor Cuffe | 19-August-2024 |

## Usability testing

Usability testing ensures that the user interface is intuitive to use. This is done by ensuring that the testers have no prior knowledge of the software so as to have only the information and knowledge that an actual user would, allowing for the tests to accurately represent the real world. This also means that the testers should not be prompted to do anything by the observers.

| Tester | Observer | Test instructions | Notes |
| --- | --- | --- | --- |
| 1 | Connor Cuffe | Create an account and send a message  Expected tester path:   1. create an account 2. enter username and password on login screen 3. enter IP address of name server on settings screen 4. click restart network 5. create a chat on home screen 6. enter username of other client 7. enter the chat they created and send message 8. exit and re-enter chat to reload message display | - Didn’t understand why they couldn’t see the message they had sent. Once they quit out they could see it. Text small.  - Confused how to connect to name server – didn’t realise that “Name server IP” needed to be populated before Restarting network.  - Didn’t know if “Restart network” button needed to be pressed for creating a chat.  - Didn’t realise that once Contact was added, a chat with that contact could be created by clicking on the contact name not by clicking on “Add chat”, which would create a new chat with a new contact. |
| 2 | Connor Cuffe | Create an account and send a message  Expected tester path:   1. create an account 2. enter username and password on login screen 3. enter IP address of name server on settings screen 4. click restart network 5. create a chat on home screen 6. enter username of other client 7. enter the chat they created and send message 8. exit and re-enter chat to reload message display | - chat box should be larger for entering text  - text font could be larger  - Quit button above the Add chat button on the home page should have a label saying Quit – accidentally quit out of the chat thinking the button was a Back button.  - they didn’t realise they had to enter the name server address to access the network |

### Modifications required

* Larger font in chat screen
* Larger message input for chat screen
* Reload button to reload that chat screen instead of closing and reopening
* Move the name server ip input to home screen
* Automatically reload network when IP changed.
* Fix bug where label does not show for quit button

### Summary

In summary most users were able to successfully navigate the user interface, as it was similar to apps they had used before. However, all testers found the font size to be to small and had extreme difficulty connecting to the network due to the entry for the name server IP being hidden in the settings menu, in future this should be placed on the start screen to make it more clear to new users. Better labeling of buttons or a simple instruction line would assist. Overall testers found they could easily navigate the app despite some obscure features.

# Evaluation criteria

Evaluation criteria define the what features and functionality the software solution must have to be successful. Evaluation criteria are determined prior to the development based on the functional and non-functional requirements.

| Criteria | Description | Achieved | Explanation |
| --- | --- | --- | --- |
| Loading times less than 1 second when switching between screens (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | Individual screens should not take more than 1 second to load. |  | All screens load within one second. See: Test loading times |
| Timely delivery of messages received less than 1 hour after sent if both participants are online (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | Messages should be delivered within an hour of being sent assuming both clients are online. |  | Messages are received quickly so long as both clients are online. See: Test message receive time |
| Zero cost for the life time of the product on both the user and developer ends (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | No cost bar that of running the client instances. Eg. no server infrastructure required. |  | Due to the increased complexity of implementing a complete peer to to peer system including discovery of client and the time constraints on the project, a small central server is required to inform users of which client is currently hosting the main server. This not only massively reduces the complexity of the project but also hardly increases cost due to the server only interacting with each client when they boot up, then never again for that session. This allows the server to be run on very cheap hardware. |
| Takes less than half an hour to learn the user interface. (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | The user interface must be intuitive for new users. |  | The user interface was intuitive for users however most users struggled to realise that they had to input the ip of the name server on the settings screen. See Usability Testing Summary |
| No evident major security flaws (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | A major security flaw is one that is likely to expose a user’s sensitive information, such as their private messages. Or allows users to be impersonated. |  | No major security flaws were found. |
| Less than five minor security flaws (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | A minor security flaw is a flaw that can expose less sensitive user data or delete user data. |  | One minor security flaw was found: (the unencrypted transmission of IP addresses) |
| Less than two software failures per 24 hours of use. (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | A software failure is an unexpected event that causes damage to the user data or a crash of the program. |  | No software failures occurred during usability testing. See Usability Testing |
| Can handle up to 100 users on one network with only minor performance losses (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024). | The network should be able to handle a large number of users at once. | Untestable | There is no indication in the current testing and analysis of the code that this would not be possible. |
| Can send and receive at least 50 messages per hour (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | The network should be able to handle a large amount of traffic at once. |  | As clients commune with each other and not via a main server there is little congestion caused by high network traffic. See: Test high network traffic resilience |
| Validates the reason for contact on all teacher student communication (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | The Victorian government requires teacher student communication to be solely for educational purposes as outlined by the Victorian government on their emailing and direct messaging page (Emailing and instant messaging) |  | Due to time constraints it was not feasible to implement this feature. |
| Moderators can view all messages on the network (Cuffe, Peer to Peer Messaging App Evaluation Criteria and UI Mockup, 2024) | The Victorian government requires teacher student communication to be solely for educational purposes as outlined by the Victorian government on their emailing and direct messaging page (email and instant messaging) |  | Due to time constraints it was not feasible to implement this feature. |

## Summary

In summary, most of the evaluation criteria identified during the design phase were successfully achieved during development. However, due to time constraints some features were removed or simplified, such as the peer to peer aspect of the messaging app which had to be simplified due to time constraints, by replacing a client discovery system with a small server to connect clients at the beginning of sessions. The message validation and chat moderation also were removed as it was infeasible to implement them within the time allotted. Overall, with the modifications, the evaluation criteria was largely successfully achieved.

# Functional requirements

Functional requirements are the basis for the evaluation criteria. They outline what features the software solution must have. Functional requirements are determined based on the data collected during the analysis phase.

| Criteria | Description | Achieved | Explanation |
| --- | --- | --- | --- |
| Send messages (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | A messaging app must by definition have the ability to send messages. (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | The messaging app can send messages. See Test message receive time and Test high network traffic resilience |
| Receive messages (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | A messaging app must by definition have the ability to receive messages (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | The messaging app can receive messages. See Test message receive time and Test high network traffic resilience |
| Secure accounts (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | It was identified during the analysis that the user base values the security of their messaging platform over anything else. (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | Whenever data is at rest or being transmitted it is structurally encrypted using the asymmetrical encryption algorithm Rivest–Shamir–Adleman (RSA). See Test data encryption |
| End to End encryption (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | It was identified during the analysis that the user base places a large amount of value on the security of their messaging platform. (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | Whenever data is being transmitted it is structurally encrypted using the asymmetrical encryption algorithm Rivest–Shamir–Adleman (RSA). See Test message encryption |

## Summary

In summary all functional requirements were met during the developments of the software solution. The final software solution has the ability to send and receive end to end encrypted messages and store those messages in secure accounts whose data is encrypted whenever in rest or transit using the asymmetrical encryption algorithm Rivest–Shamir–Adleman (RSA).

# Non-Functional requirements

Non-Functional requirements are the basis for the evaluation criteria they outline what features the software solution should have to increase user satisfaction. Functional requirements are determined based on the data collected during the analysis phase.

| Criteria | Description | Achieved | Explanation |
| --- | --- | --- | --- |
| Intuitive graphic user interface (GUI) (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | During analysis it was identified that the user base values easy to use and intuitive user interface. (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | The user interface was intuitive for users however, most users struggled to realise that they had to input the ip of the name server on the settings screen. See Usability Testing Summary |
| Low cost (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | During analysis it was identified that the final software solution being low cost was of extreme importance to the user base. (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | Due to the increased complexity of implementing a complete peer to to peer system including discovery of client and the time constraints on the project, a small central server is required to inform users of which client is currently hosting the main server. This not only massively reduces the complexity of the project but also hardly increases cost due to the server only interacting with each client when they boot up then never again for that session. This allows the server to be run on very cheap hardware. |
| Message context validation (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | A system that ensures all student teacher messaging is solely for educational purposes. This would likely take the shape of a “network overseer” who could monitor student-teacher communication. (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | Due to time constraints it was not feasible to implement this feature. |
| Find correct teacher to communicate with (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | A search system that will allow students to easily find the correct teacher based on their communication needs (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | Due to time constraints it was not feasible to implement this feature. |
| Add extra info to profile (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | The ability to have your profile display info other just name such as external contact info and user id (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | Due to time constraints it was not feasible to implement this feature. |
| Network moderation (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | Accounts can be linked to a network moderator who can manage the accounts  Message logging (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | Due to time constraints it was not feasible to implement this feature. |
| Account types (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) | The ability for network moderators to create different types of accounts for example students and teacher account types (Cuffe, Peer to Peer Messaging App Software Requirements Specification Document (SRS)., 2024) |  | Due to time constraints it was not feasible to implement this feature. |

## Summary

In summary the core non-functional requirements were met during the development of the project. However, due to time constraints it was infeasible to complete the more detailed profile and moderation requirements. These changes allowed for a functioning software solution to be delivered within the predetermined timeline at the cost of extra functionality that should be added in future development to the project.

# Business requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Description | Achieved | Explanation |
| Timeliness | Completed on time without any delays. |  | The projects major targets were achieved before the identified deadline. |
| Cost | Completed at or below budget. |  | The project budget was not exceeded. |

## Summary

In summary the software solution was completed within the allotted timeline, without running over the budget. This was achieved by simplifying or completely avoiding some non-functional requirements allowing the project to be completed yet not feature rich. In future updates to the software solution, these simplified or avoided requirements should be re-implemented.

# Project management

## Development Model

Throughout the development of the software solution the waterfall development model was used. The waterfall model was used due to it being identified during the design phase as best fitting the project’s needs. As the waterfall development model has a linear path for project development the project starts with the requirements phase, then progresses though the design implementation and verification phases. Alternatively, the agile and spiral model are more cyclical, passing though the same 4 phases again and again each time improving on the product from the last. This process allows the agile and spiral models to ensure the quality and reliability of large projects.

The initial assessment of the waterfall model proved to be correct as the waterfall model’s more simplified design flow allowed for the optimal use of the project’s strict timeline and budget. The project also did not require any of agile or spiral’s, quality and reliability controls as the project’s scope was relatively small leaving little room for the quality or reliability to slip.

## Project plan

The development of the project stayed true to the original project plan identified during the design phase. The project plan provided a structure for the timing of delivering stages. It outlined the features that the software solution should and should not include to minimise scope creep. This allowed for the timely delivery of a fully functioning software solution that meets the expectations of the customer.

The strict timeline and budget of the project proved to be challenging. The challenges were addressed by making a few changes when it was realised that delivering all the features promised in the software requirements specification document would not be feasible within the strict timeline. Features were removed such as network moderation and display of user’s profiles. However, these changes were small and allowed for the completion of the fully functioning software solution and delivery within the specified timeline and budget. The features unimplemented should be implemented in later updates to the software solution to improve user experience.

# Bibliography

Cuffe, C. (2024). *Peer to Peer Messaging App Evaluation Criteria and UI Mockup.* Melbourne: None.

Cuffe, C. (2024). *Peer to Peer Messaging App Software Requirements Specification Document (SRS).* Melbourne: None.