*Chat system with secure communication*

*Team Resume Filler*

Project Portfolio

*September 13, 2023*

[Introduction 2](#_Toc379904795)

[Team Resume Filler: 3](#_Toc764261430)

[Milestone 1: 4](#_Toc1163401112)

[System Requirements 4](#_Toc1774559650)

[Requirements: 4](#_Toc819015458)

[User Stories: 4](#_Toc535767870)

[Project Management 4](#_Toc1419710463)

[Continuity of Operations Plan (COOP) 5](#_Toc1865408852)

[Project Plan: 5](#_Toc2105031720)

[System Architecture Design and Development 6](#_Toc1864353803)

[Milestone 2 (Architecture): The WBS activity chart for the milestone should be updated to include actual level of effort and start and completion dates.] 6](#_Toc1821441212)

[System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation> 6](#_Toc1001699862)

[Project Postmortem <Postmortem> 6](#_Toc920124337)

[Project Wins 7](#_Toc289981942)

[Root Cause Analysis 7](#_Toc233040043)

[Lessons Learned 7](#_Toc43930894)

[System Design <Milestone 2: System Architecture> 7](#_Toc1143018302)

[System Architecture <Milestone 2: System Architecture> 8](#_Toc501350651)

[Component Design 8](#_Toc104202792)

[Data Flow 8](#_Toc1101335783)

[System Components <Milestone 3: System Implementation> 8](#_Toc1545052108)

[Component [Component Name 1] 8](#_Toc850514367)

[Component [Component Name 2] 8](#_Toc1241188815)

[Component [Component Name n] 8](#_Toc566185563)

[Design Pattern <Milestone 3: System Implementation> 8](#_Toc2099398861)

[Design Pattern <Milestone 3: System Implementation> 8](#_Toc1645943909)

[System Implementation <Milestone 3: System Implementation> 9](#_Toc1379425202)

[Project portfolio template directives and placeholders (delineated by “[ ]” or “< >” and/or highlighted or optional sections not included) should be removed from the document prior to submission. Empty sections for inclusion in later submissions may remain in the document for early submissions.]

[IMPORTANT: All diagrams developed using Enterprise Architectures must include the following acknowledgement: “Thanks to SPARX Systems for LSU student and faculty use of Enterprise Architect for academic purposes”.]

# Introduction

When connecting over the internet and sending any form of data, it is important to be able to trust that the information being sent is safe from any outside influence. Privacy is important to all users, and critical to some. A solution is to have a trusted tool to connect to a remote machine that allows users to send and receive data that is automatically encrypted before sending and decrypting after receiving.



Core Features:

* Sending encrypted messages
* Decrypting received messages
* Direct secure communication between machines

Viable Features:

* GUI (through C++, subject to change)
* File encryption, decryption, and transfer

Stretch Features:

* Adding friends
* Group chat
* Profile System
* Message reactions

# Team Resume Filler:

## Milestone 1:

Leader: Abby Debenport

Members: Arabelle Betzwieser, Kenyon Tiner, Tyler Saizan, Arushi Ghildiyal, Connor French

[GitHub Link](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fclassroom.github.com%2Fa%2FUCqQgtmZ&data=05%7C01%7Cktiner1%40lsu.edu%7C844c2d154411489edd6808dbabaf3521%7C2d4dad3f50ae47d983a09ae2b1f466f8%7C0%7C0%7C638292541286440561%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=5GDmBWTIHr7WyDUmVVPbO4bmESBiqjZN5Ll8B2YcI08%3D&reserved=0)

# System Requirements

## Requirements:

-Socket Programming

-Encryption/Decryption

-Communication between machines

-GUI implementation

## User Stories:

As a student, I want to form a group chat with my group mates, so I can communicate privately with them about our project.

As a programmer, I want to directly and securely send files to my team, so they can assist with my code.

As a team leader, I would like to see reactions to my messages, so that I have confirmation my message has been received.

As an investigator, I want to send encrypted investigation reports to my supervisors, so that the information is not leaked to the public.

# Project Management

## Continuity of Operations Plan (COOP)

* The team will primarily use a discord private server to communicate with each other and coordinate. Email will be a secondary means of communication.
* The team will meet for 30 minutes each Tuesday after class and on Sundays at 2pm over discord.
* If team members are unable to meet in-person or become unavailable, the team will hold virtual calls and update missing members through discord and/or email.

## Project Plan:

### System Architecture Design and Development

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
| **1** | Create Discord server for main method of comm. | 0 | low | low | 9/12/23 | 9/12/23 | 9/12/23 | **9/12/23** |
| 2 | Initial Individual research on the topic | 0 | low | low | 9/13/23 | 10/03/23 | 9/13/23 | 10/01/23 |
| 3 | Identify individual tasks and delegate | 1 | medium | high | 9/16/23 | 9/20/23 | 9/14/23 | Continued |
| 4 | Initial consultation with TA | 0 | low | low | 9/12/23 | 9/13/23 | 9/12/23 | 9/13/23 |
| 5 | Postulate application design |  | high | high | 9/17/23 | 9/22/23 | 9/17/23 | 9/23/23 |
| 6 | Begin Iterative development process | 5 | high | high | 9/19/23 | continuing | 9/17/23 | Continued |

### System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation>

Milestone 3 (System Implementation): The WBS activity chart for the milestone should be updated to include actual level of effort and start and completion dates.]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
| 1 | Networking : Build server skeleton | 0 | High |  | 10/10/23 | 10/17/23 |  |  |
| 2 | Networking: Build client skeleton | 0 | medium |  | 10/10/23 | 10/17/23 |  |  |
| 3 | Encryption: Encrypt messages | 0 | medium |  | 10/10/23 | 10/17/23 |  |  |
| 4 | Encryption: Decrypt messages | 3 | medium |  | 10/10/23 | 10/17/23 |  |  |
| 5 | Networking: Authentication | 1, 2 | medium |  | 10/13/23 | 10/19/23 |  |  |
| 6 | Networking: Setting up client key-grabbing interface | 2 | medium |  | 10/13/23 | 10/19/23 |  |  |
| 7 | Networking: Message handling | 1, 2 | medium |  | 10/10/23 | 10/17/23 |  |  |
| 8 | Basic functional demo | 1-7 | High |  | 10/10/23 | 10/22/23 |  |  |

## Project Postmortem <Postmortem>

### Project Wins

[Provide a bulleted list of at least 3 positive aspects of the project.]

### Root Cause Analysis

[Provide a bulleted list of at least 3 negative aspects of the project. For each negative, provide the answer to the three successive “Why” questions. ]

### Lessons Learned

[For each negative aspect identified in the Root Cause Analysis, provide a mitigation strategy (i.e., what process should be introduced) to ensure that the problem is not repeated in subsequent projects.]

# System Design

A client-server architecture with a cryptology interface on the client-side, enabling safe and secure communication between clients.

## System Architecture

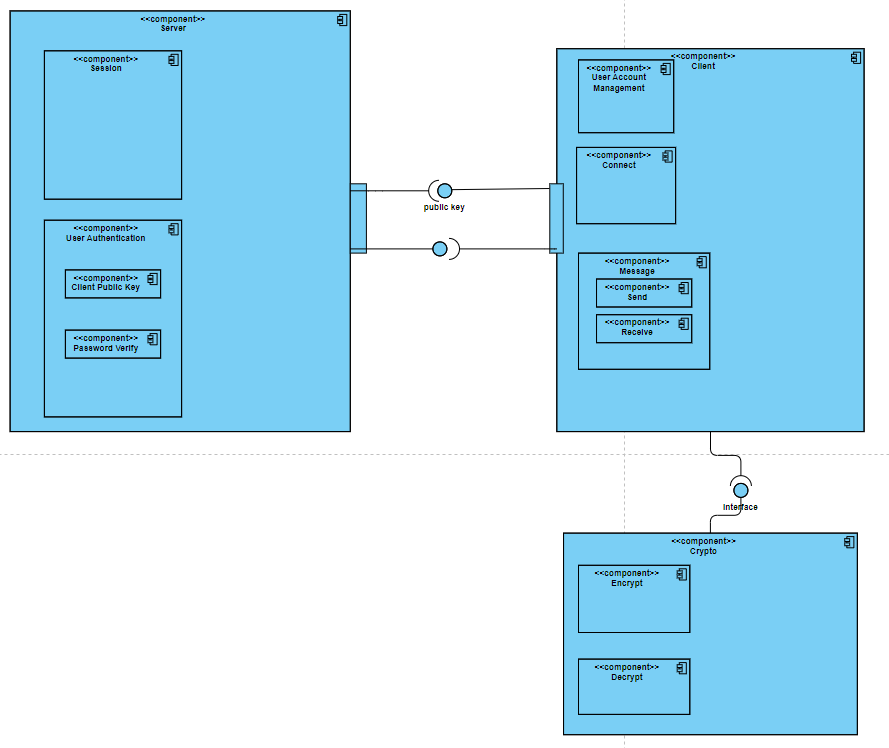
Mailman server directs messages to and from clients.

Clients encrypt outgoing messages and decrypt incoming messages through communication with server.

Cryptography module handles decryption/encryption and key generation.

### Component Design

Abby Debenport:

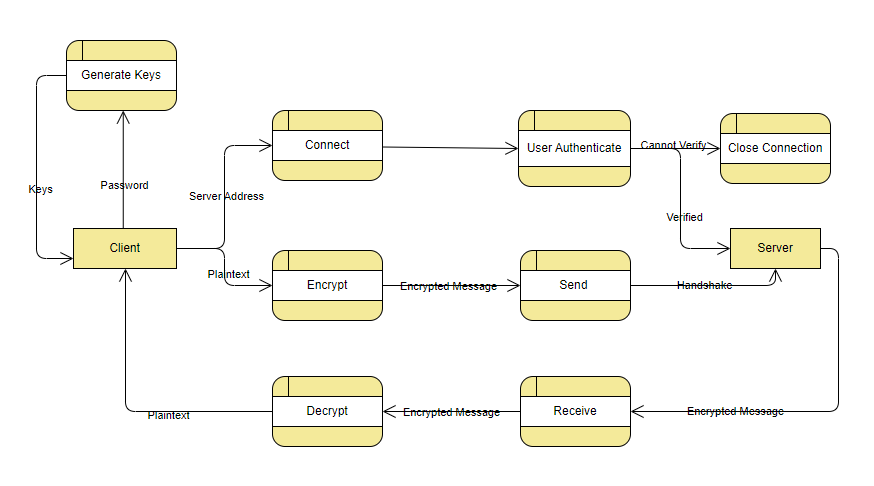
User’s cryptography tasks are handed off to cryptography module. User inputs messages, receives messages, not having to directly handle encryption process. Server only passes already encrypted messages between users.

External libraries include: libsodium, Asio

Major components: Server, Client, Cryptography module

### Data Flow

Abby Debenport:

User Inputs message into client-side application where message is encrypted before being sent to the server. The server passes the encrypted message to another client using a public key; Once the message is received by the client, the message is decrypted showing the original message.

## System Components <Milestone 3: System Implementation>

[*Include a component sub-section for each component in the architecture diagram. Each component subsection will include a class diagram*]

### Component [Component Name 1]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

### Component [Component Name 2]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

### Component [Component Name n]

[*A short description of the component*.]

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

## Design Pattern <Milestone 3: System Implementation>

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. The project must include at least design patterns covered in class. Include the name of the team member that created the diagram in EA.*]

## Design Pattern <Milestone 3: System Implementation>

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. Include the name of the team member that created the diagram in EA. A second design pattern may be included for bonus points.*]

# System Implementation <Milestone 3: System Implementation>

[*In the table below, include a row for each component in your System Architecture diagram. In the second column, list the programming language(s) used to implement the component and the what % of that programming language is used in the implementation. In the third column, list the team member(s) that implement the component and what % of that implementation was completed by that team member. IMPORTANT NOTE: All architectural components must be implemented by an object-oriented programming language: Java, C++, or C#.*]

|  |  |  |
| --- | --- | --- |
| **Architectural Component** | **Programming Language(s) %** | **Team Member(s) %** |
| *[Data Manager]* | *[C++ (45%)*  *Java (55%)]* | *[Mickey Mouse (15%)*  *Donald Duck (20%)*  *Daisy Duck (40%*  *Pluto (25%)]* |