System Test Report

Project Name	**Instant Success**
Project Leader	康隽秀 1820222039 (Hadiah Raees Chowdhury)
Project Member(s):	
郑浠寧 1820222032 (Celine' Teh Conconi)	
金俊希 1820222044 (Junhee Kim)	

- Product, Revision and Overview

This system is the basic point for the development of instant messengers, and the goal is to build a system that aims to communicate in real-time through functions such as multi-tasking and multi-chat and server socket connection, starting with the basic messenger function.

- Features to be tested

List all features to be tested. Organize the list in the way that makes most sense- user features, or by level:

Application	*Instant Succes (Instant Messenger)*
OS for development	*Linux OS*
Based development language	*C++*
Server	*Socket/SQL*

The system should be able to exchange real-time interactions between users through real-time connections, and not only simple messengers but also exchange of information between users under a two-way communication structure in real-time.

(Image-00) Hand sketch of targeted user interface design

It should also be able to process information from multiple chat rooms

Like other ready-made products, we are also targeting services such as multi-person chat rooms.

- Configurations to be tested and excluded

First, we start with whether Server-Client Protocol is enabled for server socket construction and program execution

Whether the server is connected successfully and the user can make the client work is of utmost importance.

(Image-01) Diagram of the basic test range

- Environmental requirements

Basically, we consider two main contexts.

- Virtual Environment Settings: Simulate multiple clients using Docker or Virtual Machines.
- CI/CD integration: CI/CD pipeline setup to automatically run tests when code changes.

(Image-02) Diagram of the Environmental requirements

In addition, if you solve it in more detail, it can be compressed into the above three elements.

- Tasks and Responsibilities

Task name	Content	Member(s) assigned to
Conversation Design		康隽秀
Client-Server Protocol		金俊希 , 郑浠寧
Concurrency Strategy	I. All work must be completed in windows OS in order to ensure consistency and to avoid any cross-platform complications. II. Work division must be discussed every week to ensure continuous workflow.	康隽秀 , 金俊希 , 郑浠寧
UI Sketches (paper sketches):		康隽秀
Testing Strategy		康隽秀, 金俊希
Demo video		康隽秀
Implementation		郑浠寧
Testing Report		康隽秀, 金俊希
Individual Reflections		康隽秀 , 金俊希 , 郑浠寧

Communication	
Primary Communication Channel	WeChat, Email
Meeting Schedule	Every Monday, Wednesday, Friday at 6pm
Response Time	Team members agree to respond to messages within 12 hours.

- Test requirements

6-1. Objectives

- Quality assurance: Various tests are conducted to provide reliable and secure messaging services.
- Prevent bugs and faults: Proactively prevent all possible errors related to protocols, message processing, and server-client communication.

6-2. Scope

- Feature Test: Validation of features such as real-time messaging, chat room management, user authentication, file transfer, etc.
- Performance Test: Measure the performance and load of servers when multiple clients are connected.

- Test Scenarios

Our final goal for the test is as follows.

7-1. Functional Test Scenarios

- Scenario 1: Send and receive messages between two clients.
- Scenario 2: Create and participate in chat rooms.
- Scenario 3: File transfer and reception.

7-2. System check Scenarios

- Scenario 1: Check Chatting log that recording correctly

- *Test Schedule*

- *1st Test : Unit test and initial integration test. – for 1st deadline*
- *2nd Test : Load and performance tests. – for final deadline*

- *Entry and Exit Criteria*

9-1. Entry Criteria:

- *Major features have been developed.*
- *Test environment is ready.*

9-2. Exit Criteria:

- *Successful completion of all test scenarios.*
- *Major bug fixes and troubleshooting.*

- *Risk and Mitigation*

10-1. Risk Factors:

- *Performance degradation during server overload.*
- *Security vulnerabilities in authentication systems.*
-

10-2. Mitigation Strategy:

- *Introducing clustering technology to ensure server scalability.*
- *Perform regular security patches and vulnerability tests.*

- *Reporting and Metrics*

- *Test Results Report: After each test session, a report is made that records the defects and corrections found.*
- *Recording Method: Utilize photographs of actual work and operational test recording images*
- *Performance indicators: percentage of successful test cases, frequency of bugs, and system utilization.*

First initial test for testing strategy

- *Test Result*

12-1. Client run test

- *We have confirmed that the underlying client operates without problems.*
- *It was confirmed that the functions targeted primarily were performed without problems.*
- *However, it doesn't look good that the code remains executed like this on the client, so it seems that a separate action is needed.*

12-2. Creating and chatting in rooms

- *We confirmed that conversations can be exchanged in real time between the two chat rooms, and that the conversation is delivered without any problems and damage.*
- *The delay between real-time chats is almost unfeasible, and the stability of the server-client protocol has also been confirmed.*

12-3. User-to-user file transfer

- *Specifically, the system for attaching and transferring files failed to implement, it took a while to frees up when attempting to attach files to a text window and subsequently encountered a bug that prevented messages from being sent.*

12-3. Check Chat Log

- *We checked that the chat log is written on the log history page without any problems.*
- *On the chat log page, I made sure that the words I wrote didn't interfere with other chat rooms at all and didn't cause problems.*

- *Feedback and Improvements*

- *We need to find the cause of the error in the file transfer and debug and add again.*
- *For this part, it will take time to check the error log and investigate the data. If necessary, it will be possible to refer to apps that have already been commercialized.*
- *We also need more additional things to add. We need more convenience because only basic functions are working.*
- *Basic code learning should be left to the background cmd, etc., and the client should only show the minimum, but efficient, information needed for functional behavior.*
- *We will need to add features that enable additional management such as saving, capturing, and locking chat logs.*
- *Also, We haven't set up the security part of the chat server yet. I think it's probably necessary to develop a normal chat program.*

Test for Final Report

- Overview

In the meantime, in order to secure the necessary parts as much as possible and to organize the functions necessary as an instant messenger as much as possible, we checked the cpp file and selected and rearranged only the necessary contents.

This is the cpp code for the most basic main file.

Code for the default screen client of the messenger, which contains codes for direct access to functions such as running a server to open a messenger window to creating a chat room.

The code also includes features such as ip check and log check for server access and chat management.

This code is for the Server-Client protocol.

It's further developed from the basic framework, and we've coded it to register the ip in real time, create chat rooms, and send conversation messages.

- Test result

15-1. Install

Performs socket server connection for basic program loading.

Installation success and execution are successful.

We've also confirmed that Windows PCs also run Linux through a WSL socket.

Live chat also works without problems. I checked that if you type text from one side, text will appear right on the other side.

Chat threads and logs are also reflected in real time without any problems.

We also confirmed that the chat thread and log page does not affect the chat page while it is currently operating.

- Feedback and Lessons to Learn
- Applying the designed UI to the actual program was more difficult than we thought. It seems that additional learning is needed in this area.
- In the course of this development, we found that the process of transferring files through a server was more difficult than I thought, and it seems that further exploration is needed as it is a process that has been very difficult to succeed.
- We didn't dare to develop based on Linux and apply it to other OSs. This time, it's not a big problem because it wasn't aimed at cross-platforming, but based on the current OS usage, we think cross-platform considerations will be essential in the future.