SYSC 4907

Come and Chat (C&C):

Fourth Year Engineering Progress Report Draft 1

Carleton University

Group Members:

Yunzhou Liu 101027110

Shizhong Shang 101115304

Zirui Qiao 101100225

Supervisor:

Lynn Marshall

03 November 2022

**Contents**

1. Introduction……………………………………………………………………………2
   1. Purpose…………………………………………………………………………….2
   2. Scope………………………………………………………………………………2
2. Objectives……………………………………………………………………………...2

2.1 Functional Requirements…………………………………………………………..2

2.2 Page Requirements………………………………………………………………...4

2.3 Design Constraints………………………………………………………………...4

2.4 Software Quality Attributes……………………………………………………….4

2.5 Other Requirements………………………………………………………………..4

1. Changes to Objectives……………………………………………...………………….5
2. Design……………………….………....………………………………………………5

4.1 UseCase Diagram…………………………………………………………………

4.2 UseCase Descriptions…………………………………………………………….

4.3 Flow Diagrams……………………………………………………………………

4.4 Back-end…………………………………………………………………………

4.5 Front-end…………………………………………………………………………

4.6 API Specifications………………………………………………………………

4.7 Overall Project Structure…………………………………………………………

4.8 Overall Sequence Diagrams………………………………………………………

1. Justification……………………………………………………………………………

5.1 Java8………………………………………………………………………………

5.2 Spring……………………………………………………………………………

5.3 MYSQL…………………………………………………………………………

5.4 Redis……………………………………………………………………………

5.5 Vue3……………………………………………………………………………

5.6 Element-Plus & Tailwind………………………………………………………

5.7 HTTP and WebSocket…………………………………………………………

5.8 Restful API………………………………………………………………………

1. Process………………………………………………………………………………
2. Management Methodology...….……………………………………………………
3. Upcoming Challenges………………………………………………………………
4. Individual Contributions…...….…………………………………………………

8.1 Project Contribution…………………………………………………………

8.2 Progress Report Contribution…………………………………………………

1. Conclusion…………………………………………………………………………
2. References…………………………………………………………………………

Appendix 1. Definition, Acronyms, and Abbreviations………………………………

**1.** **Introduction**

**1.1 Purpose**

Instant messaging is an efficient way to deliver information, and today, instant messaging software has become an integral part of our lives. For some of the most famous instant messaging software, such as Discord, WhatsApp and Facebook Messenger, we enjoy the convenience they offer while occasionally complaining about their shortcomings. This has led us to take a keen interest in such software.

Based on the instant messaging software that we have studied, in this project, we will build an imitated, web-based instant messaging software called Come and Chat (C&C). This proposal will outline the proposed plan and objectives to complete a working prototype of the project in early January 2023.

**1.2** **Scope**

The C&C will provide mechanisms for signing up/in users, searching users, adding/deleting friends, sending/receiving messages to/from friends, building group chats, recording chat history, and allowing users to post messages as their status, allowing users' friend to see users' status.

More functions could be added after the functions above are implemented. Customized parts are always considered after the procedures above.

**2. Objectives**

All requirements listed in each subsection of section 2 are in priority order, which means the requirement with the highest priority order in 2.1.1 is shown as the first requirement in section 2.1.1.

The subsections in section 3 are also in priority order; the subsections int front are the prerequisites of the following subsections. For example, a user cannot search for friends (2.1.2 - 1) before register in (2.1.1 - 1) the C&C system.

**2.1 Functional Requirements**

**2.1.1 Identity Functions**

1. Users provide email, nickname and password to sign up for the system.

2. Users provide their nickname and password to sign in.

3. Users can modify their profiles, including changing nickname, password, email, address, avatar, phone number, and birthday.

4. Users can log out of C&C.

**2.1.2 Friend Functions**

1. Users search for other users by nickname.

2. A user can apply to be a friend of another user.

3. A user can approve/reject a friend request from another user.

4. A user can view all his/her friends in a friend list.

5. A user can delete a friend.

6. A user can mute a friend so that he/her will not receive any message from the friend until unmuted or enters the chat room of the friend.

7. A user can hide a friend so that he/her will not be able to find the friend from friends list.

8. A user can unset a friend so that the friend is unmuted and unhidden.

9. A user can show all friends so that all friends, including hidden friends, are shown in friends list.

10. A user can search his/her friends by friend’s nickname.

11. A user can view a friend’s information including: avatar, nickname, email and statuses.

**2.1.3 Group Functions**

1. A user can create a group by providing a proper group name; group notice, avatar and group initial members are optional.

2. A user in a group can increase the member of a group by inviting friends.

3. A user can view all groups he/her joined in a group list.

4. A user can leave a group.

5. A user can view all the members of the group he/her has joined.

6. Users can view and change the information of the groups they have joined.

7. A user can mute a group so that he/her will not receive any message from the group until unmuted or enters the chat room of the group.

8. A user can hide a group so that he/her will not be able to find the group from group list.

9. A user can unset a group so that the group is unmuted and unhidden.

**2.1.4 Chat Functions**

1. Chat messages can only be sent to/received from a user’s friends and groups.

2. Users can send texts and images in chat room as a chat message.

3. Only the user themselves and the friend/group members can view messages in a chat room.

4. All users in a group chat can view messages from all users in the group chat.

5. A user can view the chatting histories of chats and group chats.

6. A friend receiving a chat message and a user sending a chat message should happen almost simultaneously.

7. A user will be able to know whether or not unread messages have been sent by a group or friend without entering a chat room.

**2.1.5 Status Functions**

1. A user can post text and image messages as his/her status.

2. A user can only view his/her own and friends' statuses.

3. A user can ‘like’/’unlike’ a status and view the amount of ‘like’s of a status.

4. A user can write text as comments to a status.

5. A user can view all comments of a status.

6. A user can delete his/her own statuses.

**2.1.6 Administrator Functions**

1. An administrator of C&C can log into C&C by providing the correct username and password.

2. An administrator of C&C can view statistics including:

1. total number of users,
2. number of online users,
3. total number of statuses,
4. total number of comments, and
5. total number of chat messages.

3. An administrator can search a user by nickname or user id.

4. An administrator can block/unblock a user, and the blocked user cannot log in to the C&C system until unblocked.

**2.1.7 System Scheduling**

1. The system shall use the MVC model.

**2.1.8 Optional Functions: Camp Grouping**

1. There exist n camps in C&C set by the administrator

2. Every client must join 1 and only 1 camp during registration.

3. Clients within the same camp, if they are friends, can chat with each other without

limits.

4. Chat messages sent from different camps will be encoded, which means one

cannot chat with clients in other camps.

5. There are periodic events that allow 2 random camps to understand each other’s

messages over time. These events are triggered by time.

6. Clients cannot switch camp.

**2.2 Page Requirements**

1. The list shows all friends' chats, and group chats should exist the whole time after signing in.

2. Signing in/up page should be a separate page from the chatting pages.

3. All messages sent by a user should appear with their avatar simultaneously.

**2.3 Design Constraints**

1. Performance should be reasonably fast.

2. The system shall be usable at least on Chrome.

3. The system shall support PC and mobile.

4. The system must comply with the relevant privacy legislation.

**2.4 Software Quality Attributes**

1. Users can only see messages in chats between themselves and their friends and the group chat they joined.

2. Users can only view the history of messages that they can see.

3. The system can be remotely accessible to all users.

4. The system shall protect all users' information.

**2.5 Other Requirements**

1. The time plan can be divided into 2 parts, version 1 and version 2. Version 1 will be completed before Jan 22, 2023, and version 2 will be completed before Mar 14, 2023.

2. The hard deadline for the system is the end of April of next year for project completion.

3. One or many servers are required for this project. Specific requirements are still in process of investigation.

**3. Changes to Objectives**

In the section 2.1 - Functional requirements, the original subsections: 2.1.1 Identity and Relationship Functions and 2.1.2 Communication Functions are divided into 5 different subsections (2.1.1 - 2.1.5). The original vague requirements are described more clearly, making the entries more numerous. After intense discussion, we decided to focus on text and image chat messages and statuses for the time being; the C&C system is so complex that we thought we should wait until the basic functionality had been designed and implemented before looking at other type of chat messages and statuses. Optional functions are added as a new sub-section, where an original designed rule is described. These features are designed to improve the fun and originality of the software, as the software can function even without this part, so this part of the requirement is classified as an optional requirement.

A new subsection(2.1.6) for administrator functions is added in functional requirement section. We realized that as social software, administrators are needed to manage users' improper actions. Administrators should be able to block users who misbehave without violating the privacy of any user. This is why the only actions the administrator can perform on the user are searching, blocking and unblocking.

We would like to take this opportunity to observe users' behaviour more visually and without invading their privacy, and to produce statistics. This is the reason of adding requirement 2 in subsection 2.1.6.

To add original features and make the program more interesting, a new subsection(2.1.8) for Optional functions is added in the functional requirement section. The sub-section describes the rule of Camp Grouping, which includes technical challenges such as identity creation, rights management, chat encryption and decryption of information and timed tasks. These functions are considered optional and will only be implemented after all other functions are implemented and tested.

In section 3.2, the only change is that the requirement of viewing all group member’s avatar list while in a group chat room is removed. Since users can neither mute/hide/unset a non-friend user nor chat with the user, it is unnecessary to list all group member the whole time. This is the reason we remove the requirement.

The rest sections remain unchanged.

**4. Design**

The design methodology learned from SYSC 3120 (Software Requirements Engineering) was very useful in the design of this project. As a result, the design process followed the requirements analysis, modelling and specification learned in SYSC 3120.

这里的每一段都既有图又有描述。描述详细一点

**4.1 UseCase Diagram liu**

**4.2 UseCase Descriptions liu**

**4.3 Flow Diagrams liu**

**4.4 Back-end liu**

**4.4.1 Simple UML**

**4.4.2 ER Diagram**

**4.4.3 UML**

**4.5 Front-end**

**4.5.1 Prototype**

**4.5.2 Page Structure Document**

All the page designs are made in the website Figma[1], this design website is convenient to use and easy to make changes, so the team decide to use it as a design support.

“Login” page (Figure 1) is used to let uses login to the system, user’s name and password is needed. In this page, the user can either choose to login by click “Sign In “ button, or if the user is new to the system, this page can go to “Register” page by click “Register here!” button.

****

Figure 1 - login page

“Register” page (Figure 2) is used to let new users to register into the system. In this page, the user need to create a user name, provide their email address, and confirm their password to make sure two password are same. User can click the “Register” button to register, or if the user already has a account, they can click “Sign In Here!” to go to the “login” page.

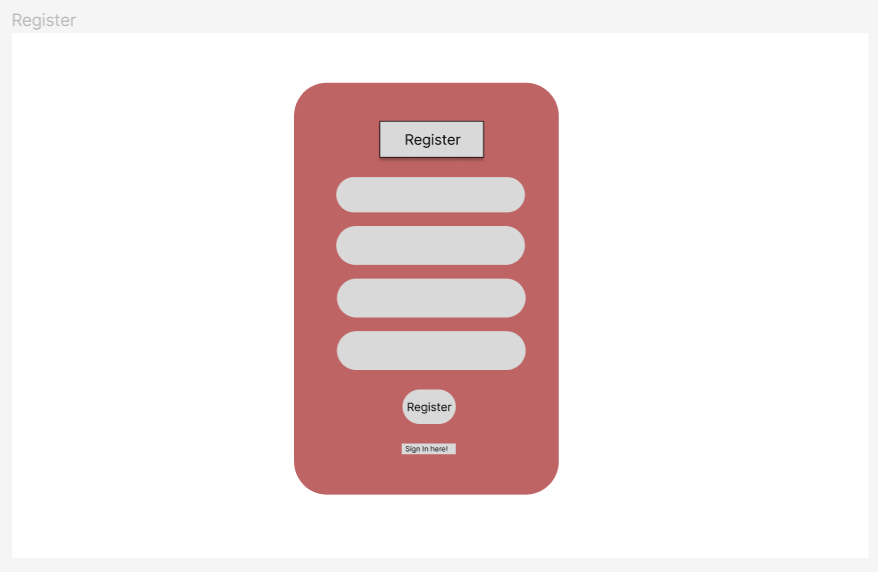


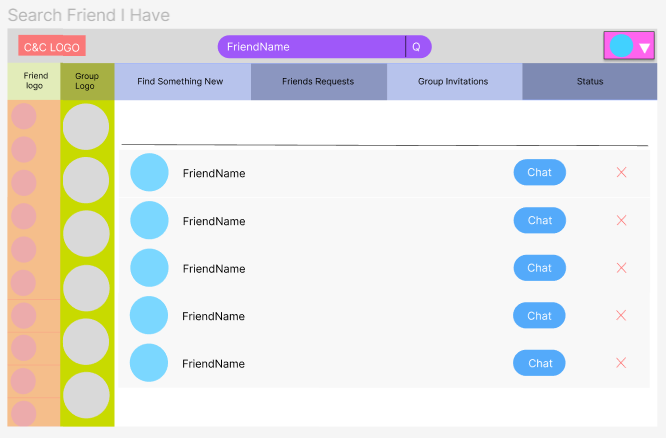
Figure 2 - register page

“Main” page (Figure 3) is the page that will be shown to the users after they login to the system,

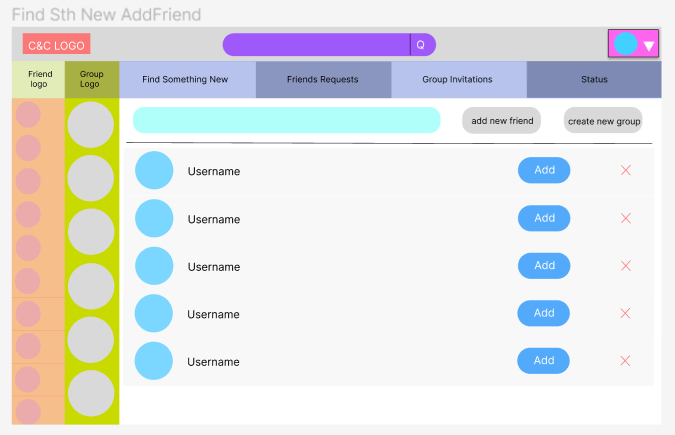


Figure 3 - main page

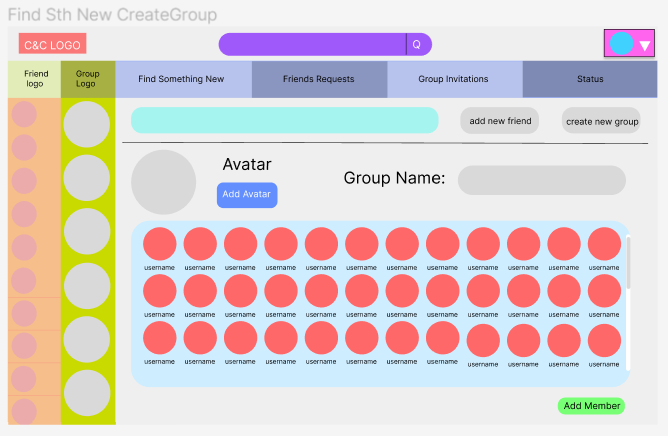
“Search Friend I Have” page



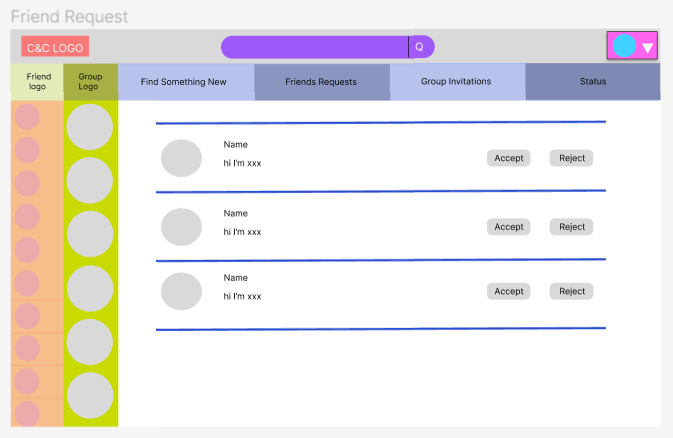
“Find Something New Add Friend” page



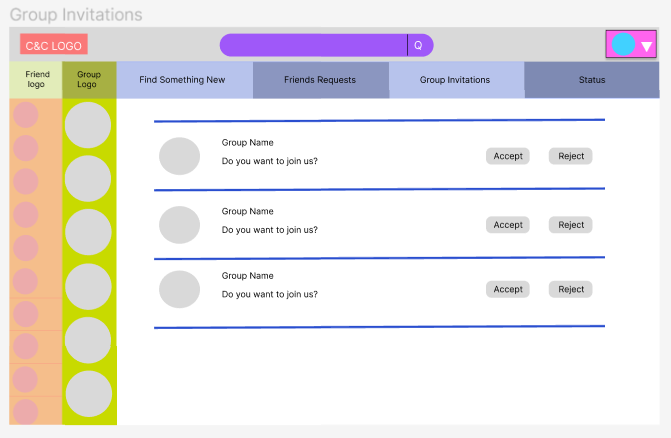
“Find Something New Create Group” page



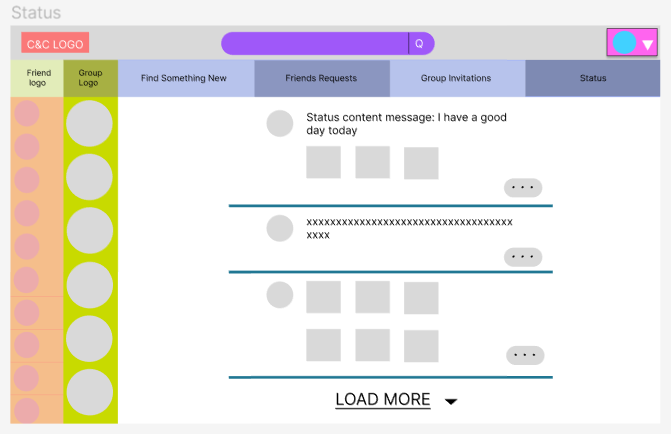
“Friend Request” page

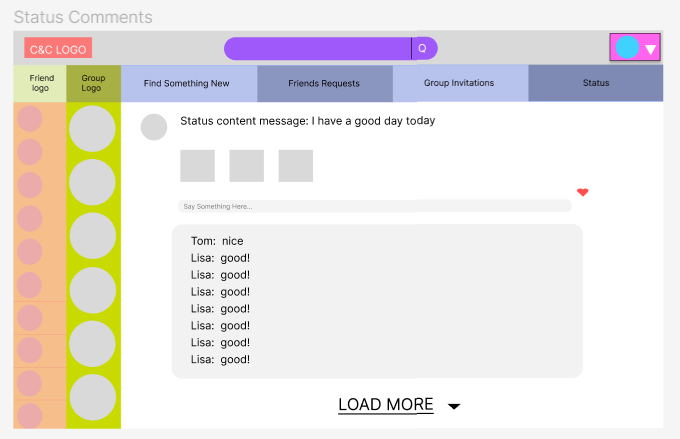


“Group Invitations” page



“Status” page





Head, friend and group hover are showen below (Figure 4 and Figure 5),

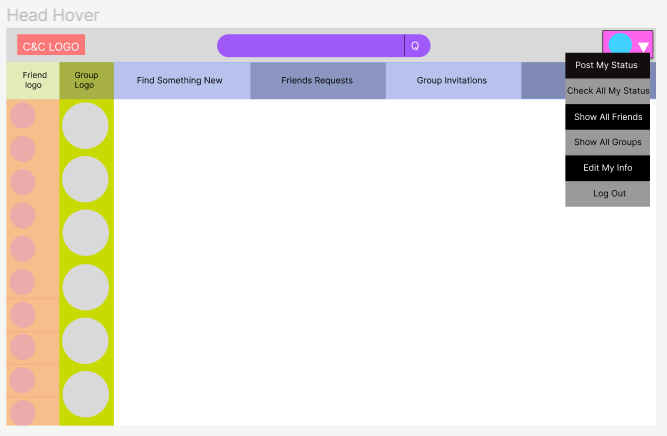
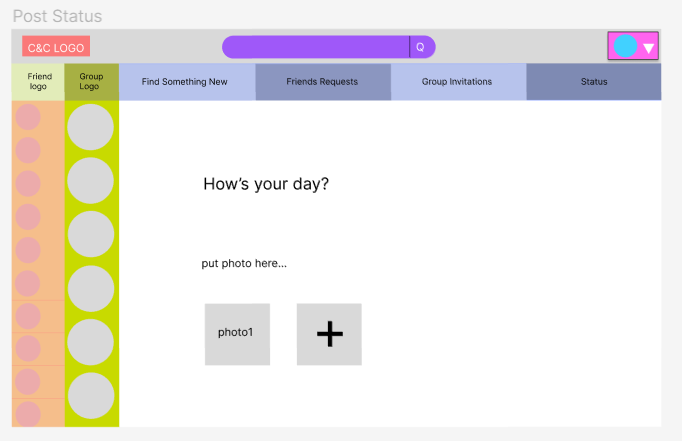
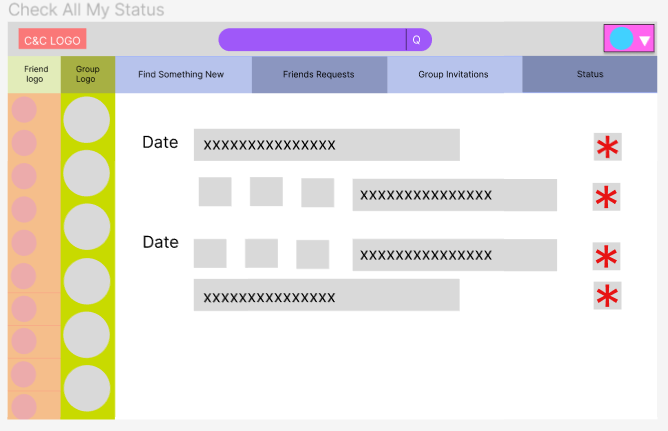
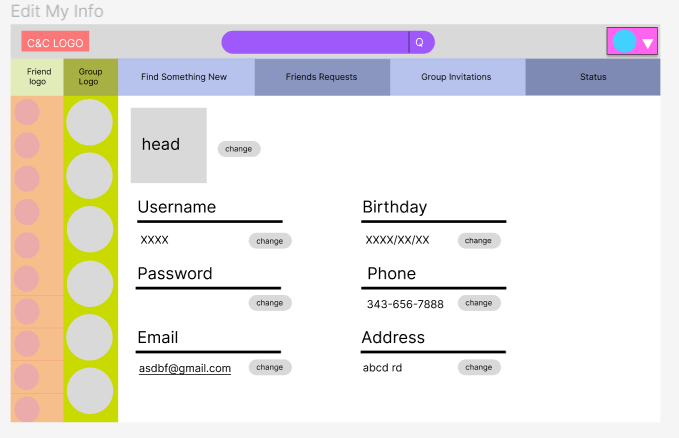


Figure 4 - head hover







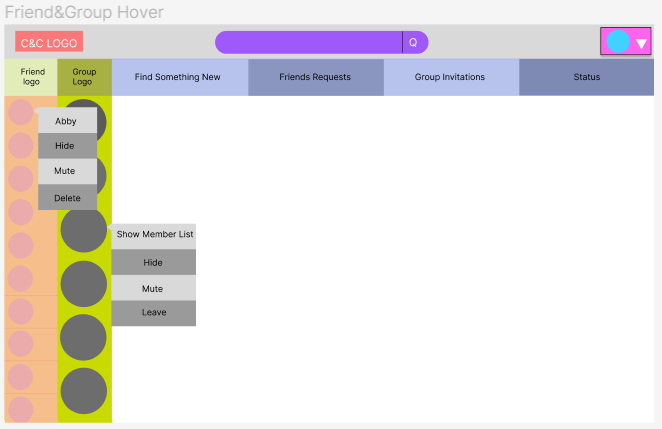
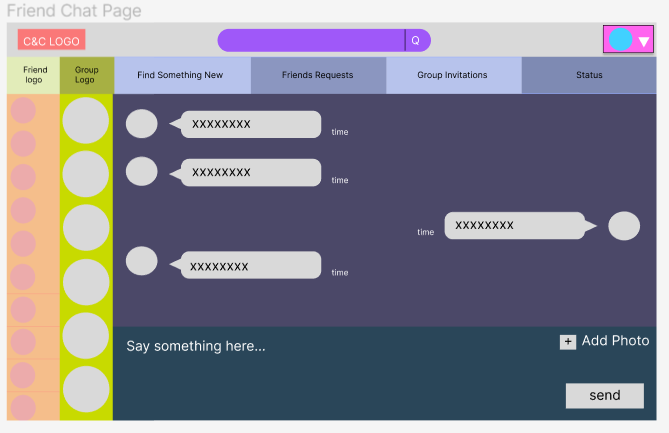
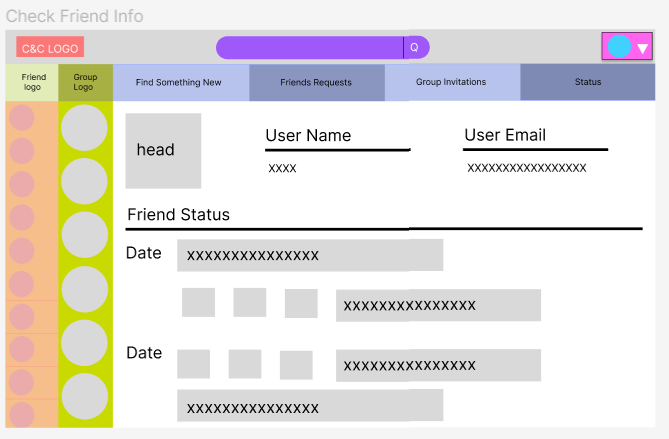
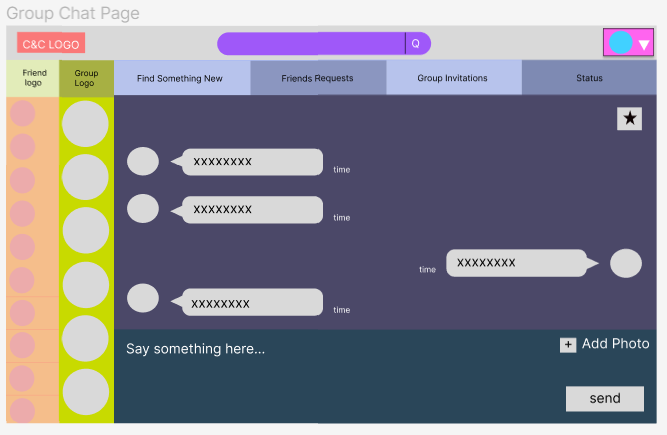
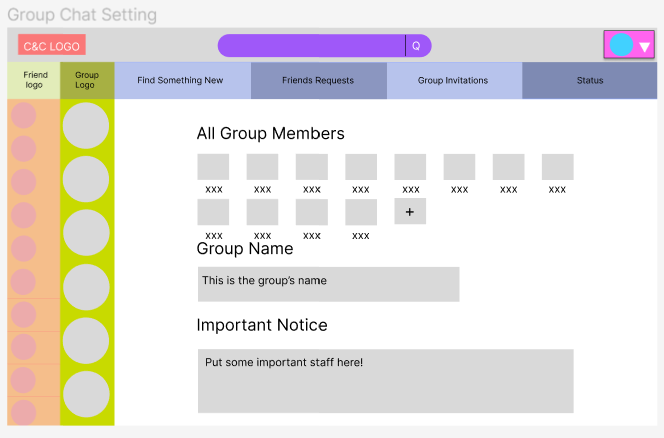


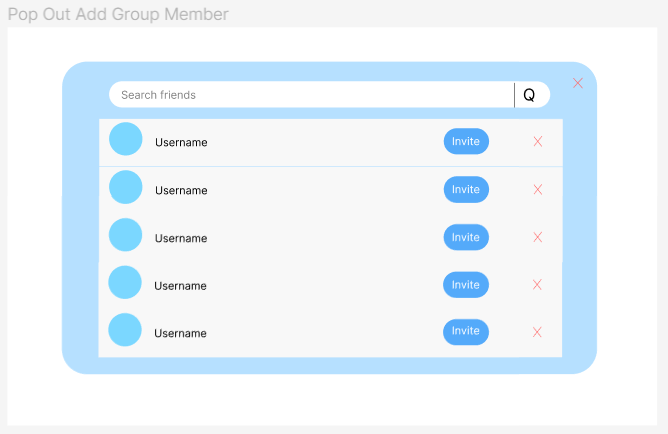
Figure 5 - friend and group hover









****

**4.5.3 UML**

**4.6 API Documents**

**4.7 Overall Project Structure**

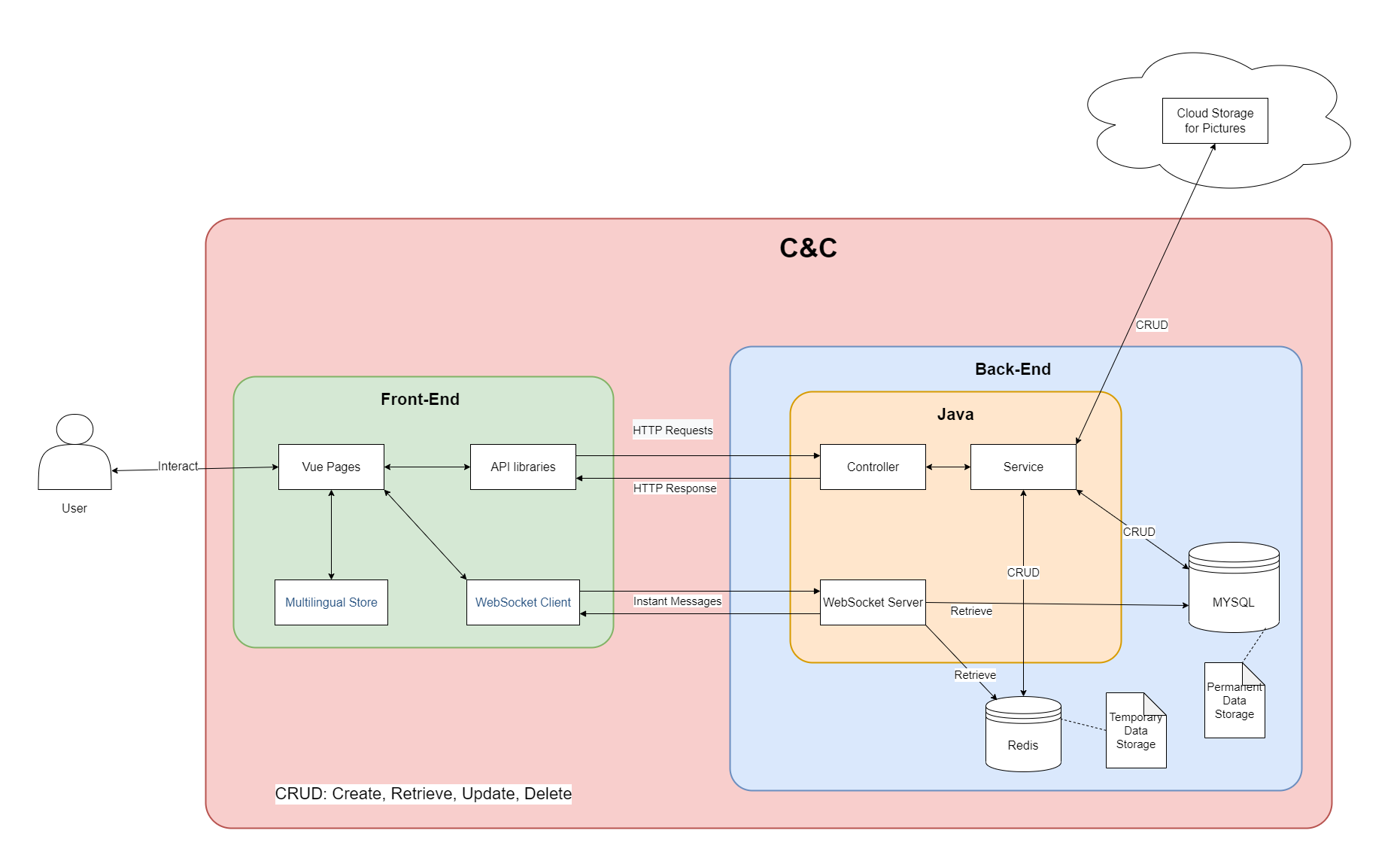
****

Figure 4.7.1: Overall Structure of C&C Project

An overall project structure is designed to describe the project in a more clear way.

Users can only interact with Vue pages in Front-End part. Vue pages handles page jumping by itself and provide multi-languages through the help of Multilingual Store. Most requests are sent to Back-End part through API libraries as HTTP requests. Once Back-End generate replies corresponding to requests, API libraries obtain and return the replies to Vue pages and display it to the user. Chat messages cannot be treated exactly as HTTP requests, therefore, WebSocket Client module will send and receive instant messages (WebSocket messages) to and from WebSocket Server in Back-End.

Java is the main software application in Back-End. The Controller module recognize all HTTP requests and invoke the corresponding service in Service module to handle the requests. Once service generate a result, Controller judge the result and reply the judging result to Front-End. MYSQL is a relational database which stores permanent data for C&C and Redis is a NoSql database which stores temporary data for C&C. The Cloud Storage on the top of the graph is a cloud database which can only store pictures, all pictures in C&C will be stored in the Cloud Storage module. Only Service module can perform all CRUD operations on the three databases, CRUD stands for Create, Retrieve, Update and Delete. The WebSocket Server module is only used to accept and reply to instant messages (WebSocket messages). To obtain necessary user data, the WebSocket Server can retrieve data from both Redis and MYSQL.

C&C comprise all elements in the graph except for User and Cloud Storage. The C&C portion is labelled in red.

**4.8 Overall Sequence Diagram**

**5. Justification**

说说为什么选用这些技术

**5.1 Java liu**

**5.2 Spring liu**

**5.3 MYSQL liu**

**5.4 Redis**

A user is unlikely to send only one request after logging into C&C. Therefore, a lot of time is wasted frequently retrieving data from MYSQL. To speed up the generation of responses to requests, it is helpful for the C&C system to keep useful data for an extended period of time instead of retrieving it from MYSQL every time it needs it.

NoSQL Database solves this problem perfectly; they store temporary data in a RAM or cache and provide data when software needs them. Retrieving data from RAM (NoSQL Database) is a million times faster than the same operation from SSD/HDD (Relational Database). Therefore, to speed up the response, C&C stores frequently used data in the NoSQL database, for example, user tokens. There are two famous NoSQL databases: Redis and MongoDB. Redis is faster and uses fewer resources, but most of its APIs are atomic, which means it could take time to write customized APIs for C&C to use it. Redis stores data in the RAM only in the Key-Value pair and does not support other query languages. MongoDB is slower and uses more resources; while it owns APIs with various functions, it is undoubtedly more straightforward to use than Redis. MongoDB store data in Key-Value pair but allow multiple keys and support many query languages, even JSON[Q1].

Redis states that the MongoDB database needs Redis when the following requirements are required [Q2]:

* Query optimization (caching)
* Session management
* Real-time analytics
* High-speed data ingestion
* Message brokering and queues

The comparison between Redis and MongoDB is very similar to that between C and Python; both Redis and C have higher performance, while MongoDB and Python are easier for developers. C&C will not store complex data in RAM; the speed is the most relevant. Therefore, Redis is chosen. However, if C&C is a huge sophisticated software and stores various types of data in RAM or cache, MongoDB is the better choice.

**5.5 Vue3**

**5.6 Element-Plus & Tailwind**

**5.7 HTTP and WebSocket**

HTTP represents for Hypertext Transfer Protocol which is an application-layer protocol for transmitting hypermedia documents, such as HTML and JSON[Q3]. Through it, data can be transfered between Front-End and Back-End easily. In C&C, most of the transfered data are in format of JSON[Q4]. However, only the client can send HTTP requests, not the server, and there must be an HTTP request before an HTTP response.

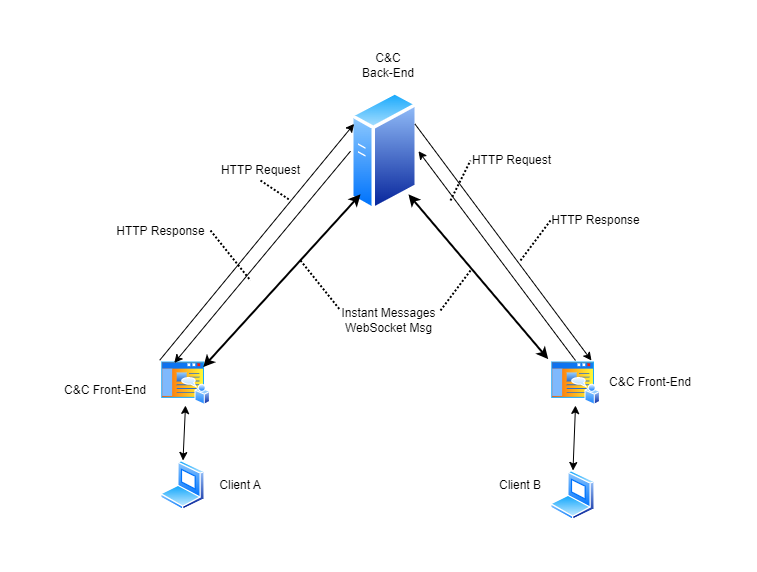


Figure 5.7.1: Data Flow Diagram of C&C

There is no direct connection between Client A and Client B users in the C&C system in Figure 5.7.1. The only way for A to see messages sent by B is for the server to send messages to A without being requested. However, servers cannot send unsolicited HTTP messages to clients, while client A needs to know messages sent by client B immediately. To be dumb, we could solve this problem by generating a set of HTTP requests to the server once every second. But this would seriously increase the pressure on the server, and a one-second delay is a long time for instant messaging, so such a solution doesn't work.

To solve the problem, there are two possible solutions according to our investigation: WebSocket and WebRTC.

WebSocket is an advanced technology that makes it possible to open a two-way interactive communication session between the user’s browser and a server[Q5]. WebRTC represents for Web Real-Time Communication which is a technology that enables Web applications and sites to capture and optionally stream audio and/or video media, as well as to exchange arbitrary data between browsers without requiring an intermediary[Q6]. Both technology allows clients connect with each others.

Briefly, WebSocket need a intermediary to create TCP connection among clients while WebRTC create p2p UDP connection (peer to peer) directly between two clients. UDP connection[Q7] could lose some data during transmittion while 100% transmission integrity guaranteed for TCP connection[Q8]. The guarantee of transmission integrity can take a lot of time, so that UDP transmission will be faster than TCP.

In terms of performance, WebRTC is much faster than WebSocket, as it connects directly to its peer using UDP, while WebSocket first connects to the server and then connects to another client from the server using TCP. Conversely, WebSocket has a clear advantage in terms of data integrity.

In C&C, a chat messages encoding requirement is described in sub-section 2.1.8-Optional Functions; C&C may need to control whether or not chat messages are encoded among users. A chat history viewing function is described in term 5 of sub-section 2.1.4-Chat Functions; C&C need to store all sent chat messages for each user. To do message encoding, decoding and recording, we cannot put the three steps in front-end for security reasons, back-end it the only choice, therefore, WebSocket is the better choice of C&C.

In the future, C&C may add video and voice calling capabilities, and WebRTC would be a good choice for implementing these features.

**5.8 Restful API**

To handle HTTP requests, it is essential to have a consistent architecture for APIs. Fortunately, many API architectures have been developed by predecessors.

According to out investigation, there are 3 main API architectures: REST, SOAP and RPC.

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer, is a set of guidelines for scalable, lightweight, and easy-to-use APIs[9]. The guidelines of REST are:

* Client-Server Separation: Separate client from server.
* Uniform Interface: Use specific HTTP format.
* Stateless: Each client-server interaction is independent of every other interaction.
* Layered system: Requests and responses must always be formatted the same way.
* Cacheable: Server responses should indicate whether a provided resource can be cached by the client and for how long.

The design of C&C which separates client and server is a guideline for REST. As the most famous API architecture, it perfect for our web-based instant messaging software — C&C.

SOAP stands for Simple Object Access Protocol which relies heavily on XML, and together with schemas, defines a very strongly typed messaging framework. SOAP strictly defines how messages should be sent and what must be included. This makes SOAP APIs more secure than REST APIs, although the rigid guidelines make them more code-heavy and harder to implement[Q10]. In C&C, various types of data structures are used in API transfer; therefore, despite its safety features, it is not a good choice for the C&C project.

The RPC (Remote Procedural Call) protocol is the most straightforward of the three architectures. Unlike REST and SOAP, which facilitate data transfer, RPC APIs invoke processes. In other words, they execute scripts on a server. RPC APIs are limited in security and capabilities, so it is unlikely to see them as often as REST or SOAP APIs on the web. However, it can be used for internal systems for making basic process requests, especially many at once[Q11]. As an internal system-specific API architecture, it is not a suitable choice for the API architecture of C&C.

In conclusion, REST API or Restful API is the best choice among the three API architectures.

**6. Process liu**

现在做到哪了

Finish all design part, just started

**7. Management Methodology**

WaterFall model is selected for our project management. We spent tons of time on designing, it is unlikely for us to miss many important requirements. Therefore, the requirements for our project is tend to be explicit and constant. In our project, only one iteration is estimated to be proceed.

Rapid Prototyping model is an excellent model for software projects, it allows flexible design and development. Developers can make any changes in no either designing phase or developing phase, and the end user can “see” the system requirements as they are being gathered by the project team. For our project, there is no end user in our designing nor developing phase. The whole team designed the project together and discussed various aspects of the project, changes will occur, but not much. Another reason for not using the rapid prototyping model is that this model requires rapid iterations, so our team must put tons of time into the project. All three of us planned to graduate in 2023, and this model is not friendly for 4th-year students like us.

Spiral Model is very attractive for our team. It handles change through iteration and minimizes risk. Still, rapid iteration is not a good choice for our team. Doing risk analysis takes time and require a large team, 3 people team is never considered as a large team.

Rapid Application Development model is also a good model, it divides the whole project into several small projects and use waterfall model on each. The start of the second small project does not require a complete of the first small project, such developing model could work very fast. However, performing such a development process requires a very familiar team with the corresponding area, which has to be large enough to split into smaller groups for different small projects. Our team of three did not meet any of the above points, so we could not choose this model either.

The main idea of using the incremental model is to keep the whole team working; however, for a small group of only three team members dealing with such a complex C&C project, most of the work is already planned, and there is not much free time. Therefore, there is no need to choose incremental model despite its excellence.

In conclusion, the WaterFall model best suits our situation.

**8. Upcoming Challenges**

WebSocket, Data Store (Images), front-end, back-end connection….

**9. Individual Contribution**

**9.1 Project Contribution**

| Name | Project Design Contribution |
| --- | --- |
| Yunzhou Liu | ERDiagram V1.0-V1.2  Admin Back-end UML |
| Shizhong Shang | User Page Structure  Admin FlowDiagram  Admin Front-end Page  Admin Front-end UML |
| Zirui Qiao | UseCase Diagram V1.0-V1.3  User FlowDiagram V1.0-V1.2  User Back-end UML V1.0  API specification V1.0-V1.4  Overall Project Structure |
| Yunzhou&Zirui | Module UML V1.0 |
| Shizhong&Zirui | User Page Design  User Front-end UML |
| All Members Together | UseCase Description 1st, 2nd draft |

Table 1.0 Individual Contribution in Project Design

**9.2 Progress Report Contribution**

| Name | Progress Report Contribution |
| --- | --- |
| Yunzhou Liu | 4.1 Design - UseCase Diagram  4.2 Design - UseCase Descriptions  4.3 Design - Flow Diagrams  4.4 Design - Back-end  5.1 Justification - Java8  5.2 Justification - Spring  5.3 Justification - MYSQL  6 Process  10 Conclusion |
| Shizhong Shang | 4.5 Design - Front-end  4.6 Design - API Specifications  4.8 Design - Overall Sequence Diagrams  5.5 Justification - Vue3  5.6 Justification - Element-Plus & Tailwind  8 Upcoming Challenges |
| Zirui Qiao | 1 Introduction  2 Objectives  3 Changes to Objectives  4.7 Overall Project Structure  5.4 Justification - Redis  5.7 Justification - WebSocket  5.8 Justification - Restful API  7 Management Methodology |
| Together | 9 Individual Contributions  11 Reference  Appendix 1. Definition, Acronyms, and Abbreviations |

Table 1.1 Individual Contribution in Progress Report

**10. Conclusion liu**

**11. References**

[Q1]“MongoDB Vs. Redis Comparison: Pros And Cons,” MongoDB, 2022. https://www.mongodb.com/compare/mongodb-vs-redis (accessed Nov. 19, 2022).

[Q2]“Redis vs MongoDB: Which is Better? MongoDB & Redis 101 | Redis,” Redis, Sep. 19, 2022. https://redis.com/docs/why-your-mongodb-needs-redis/ (accessed Nov. 19, 2022).

‌

[Q3]“HTTP | MDN,” *Mozilla.org*, Sep. 13, 2022. https://developer.mozilla.org/en-US/docs/Web/HTTP (accessed Nov. 19, 2022).

‌[Q4]“JSON,” *Json.org*, 2022. https://www.json.org/json-en.html (accessed Nov. 19, 2022).

[Q5]“The WebSocket API (WebSockets) - Web APIs | MDN,” *Mozilla.org*, Sep. 09, 2022. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\_API (accessed Nov. 19, 2022).

[Q6]“WebRTC API - Web APIs | MDN,” *Mozilla.org*, Oct. 30, 2022. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC\_API (accessed Nov. 19, 2022).

[Q7]L. Rosencrance, G. Lawton, and C. Moozakis, “User Datagram Protocol (UDP),” *SearchNetworking*, 2021. https://www.techtarget.com/searchnetworking/definition/UDP-User-Datagram-Protocol#:~:text=User%20Datagram%20Protocol%20(UDP)%20is,provided%20by%20the%20receiving%20party. (accessed Nov. 19, 2022).

‌[Q8]B. Lutkevich, “Transmission Control Protocol (TCP),” *SearchNetworking*, 2021. https://www.techtarget.com/searchnetworking/definition/TCP#:~:text=Transmission%20Control%20Protocol%20(TCP)%20is,of%20data%20to%20each%20other. (accessed Nov. 19, 2022).

[Q9]J. Juviler, “REST APIs: How They Work and What You Need to Know,” *Hubspot.com*, Aug. 30, 2022. https://blog.hubspot.com/website/what-is-rest-api (accessed Nov. 19, 2022).

[Q10]“SOAP vs REST APIs: Which Is Right For You? | SoapUI,” *Soapui.org*, 2014. https://www.soapui.org/learn/api/soap-vs-rest-api/ (accessed Nov. 19, 2022).

[Q11]J. Juviler, “4 Types of APIs All Marketers Should Know,” *Hubspot.com*, Aug. 26, 2021. https://blog.hubspot.com/website/types-of-apis#:~:text=There%20are%20four%20widely%20agreed,internal%20APIs%2C%20and%20composite%20APIs. (accessed Nov. 19, 2022).

‌

‌

‌

‌

‌

‌

‌

‌

‌

**‌**