CAS 703 G5– High-Level Design Specification

1 Introduction

1.1 Purpose

This document provides a high level design specifications for QuickMessenger app by providing details in a systematic manner to support project management and system development processes. This document first captures the uses for Quick messenger and then specifics analysis class and architectural design choices based on prior steps. The document is intended for software engineers, system architects and project managers using Agile methodology.

1.2 System Description

QuickMessenger mobile application designed specifically for iOS and Android platforms allows registered users to communicate efficiently using text, as well as audio visually and share various forms of objects such as multi-media and documents. Users can communicate in a secure manner either peer to peer or in a group. Group management is a core component of this system and provides a social platform facilitation as an alternative to Social Media in a secure manner.

As opposed to other kinds of messaging, instant messaging is interactive with near real time conversation; this requires very low latency in the delivery of messaging.

1.3 Overview

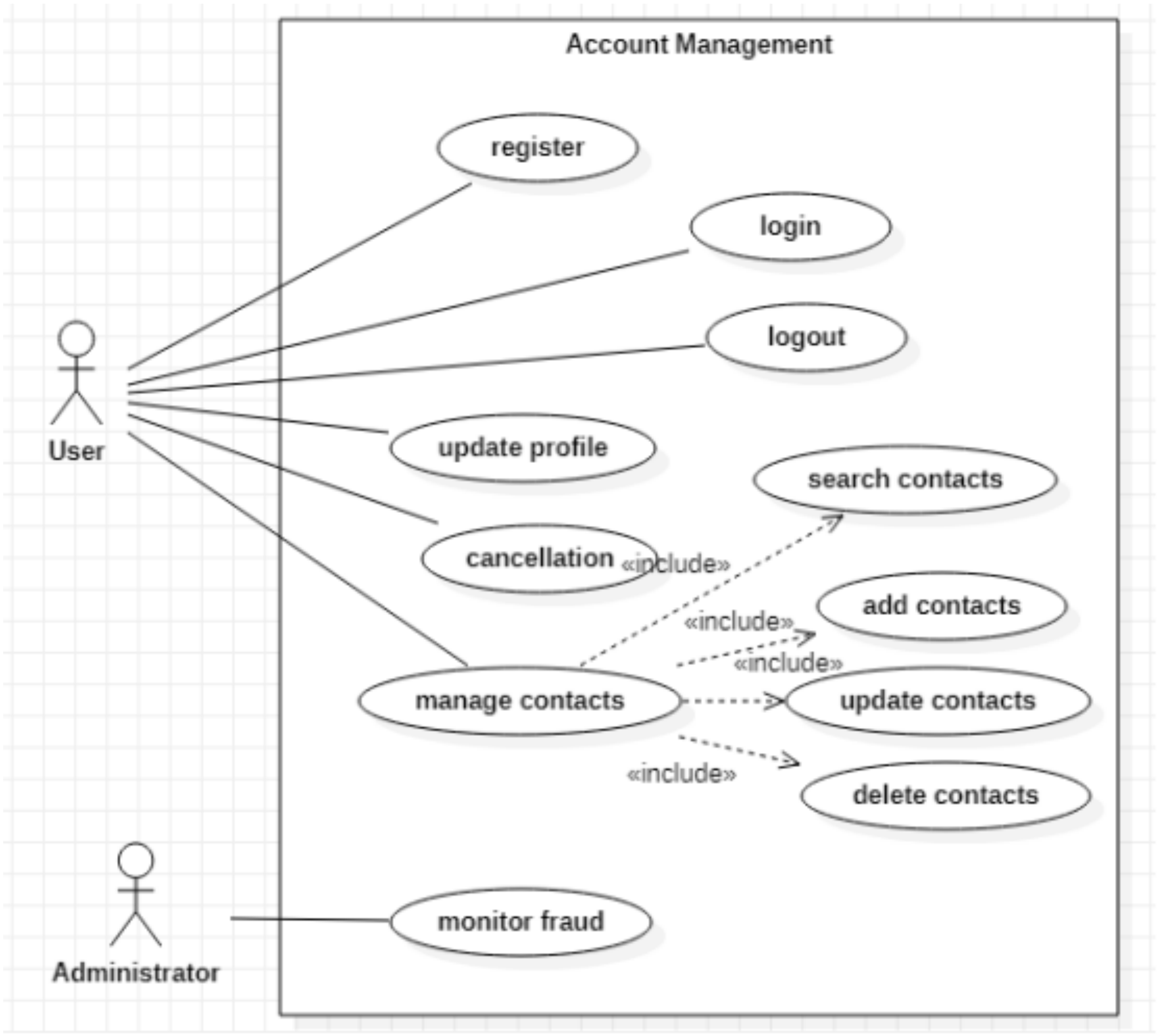
The document follows a layering process by first capturing Use Cases for QuickMessenger; from which Analysis Class diagrams will evolve which sets the direction for respective Architectural design where system architecture and its relevant subsystems will be highlighted. From the Use and Analysis classes diagrams, Class Responsibility Collaboration will provide the details for each Class, Responsibility and Collaborator class.

2 Use Case Diagram

The whole system consists of two functional subsystems. The first subsystem is in charge of account management, the second one is messaging subsystem.

2.1 Account Management

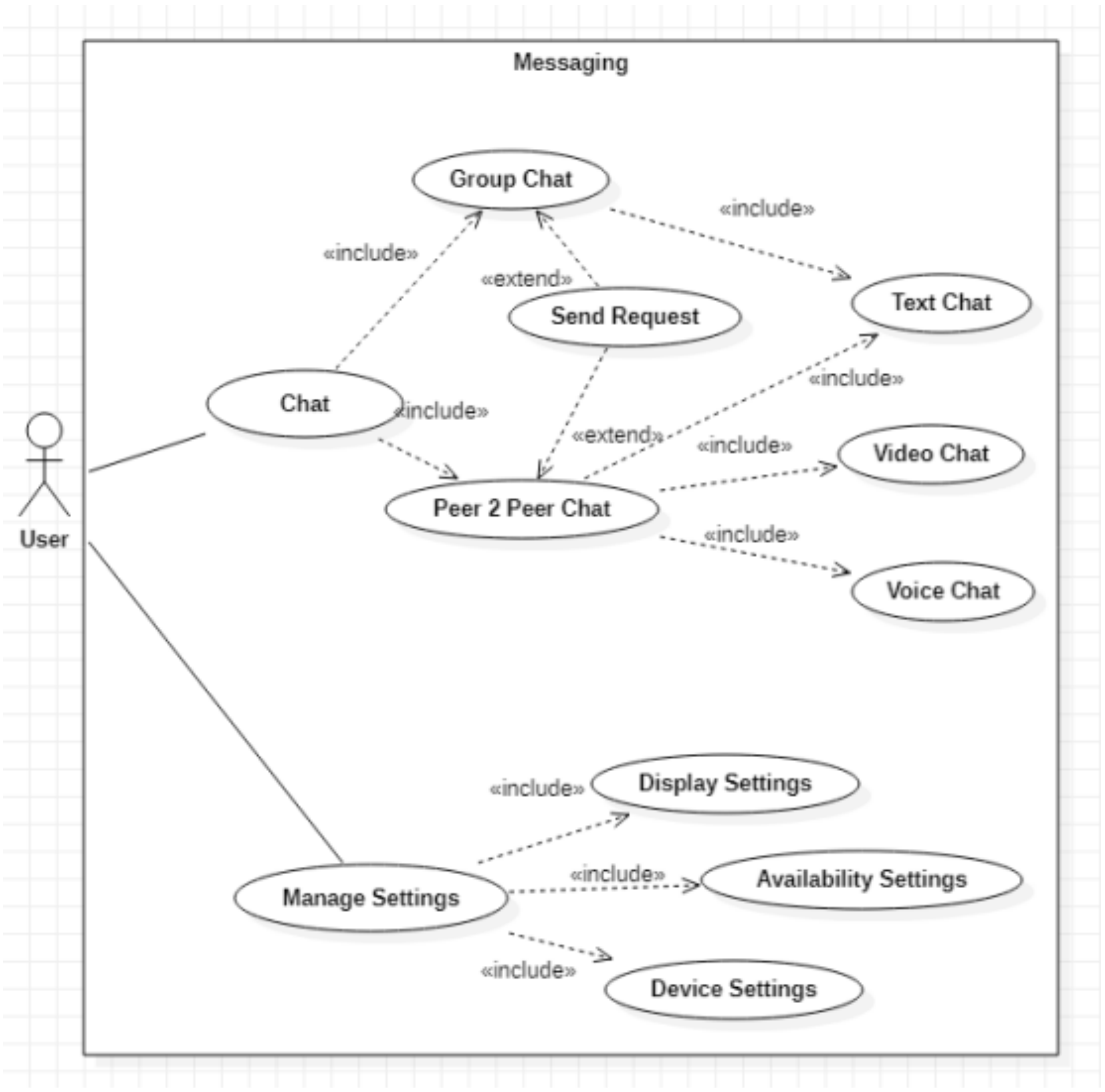
Account management subsystem interacts with two roles: the end user and the administrator. The end user needs to use this subsystem in the following scenarios: a) New user registration. b) login / logout. c) profile update. d) account cancellation. e) contacts management, which includes add/update /delete/search contacts. The administrator works in the back-end. This role is responsible for monitoring potential frauds to ensure all accounts are safe.



2.2 Messaging

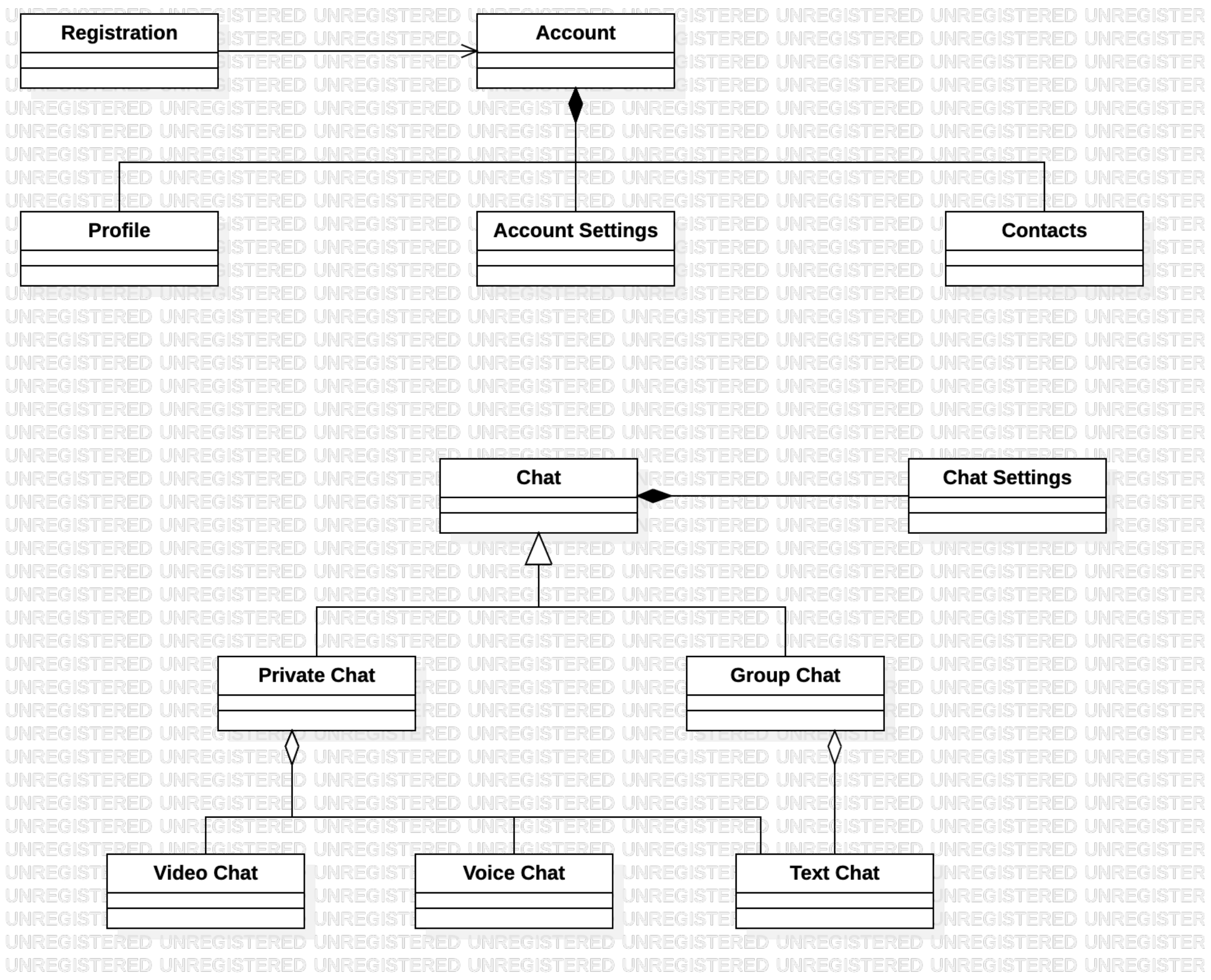
Messaging subsystem is used to facilitate conversation among end users. There are two kinds of chatting: group chat and peer to peer chat. Group chat supports only text, while peer to peer chat supports text, video and voice. To set up a chat, one user must initiate the chat by sending a request to back-end server, indicating the recipient of a message.

End users also have options to change the messaging settings, which include display settings, availability settings and device settings. Display settings include font size, background color, etc. Availability settings show the time slots when the user is available. Device settings include volume setting, video quality setting, etc.



3 Analysis Class Diagram

The class diagram of QuickMessenger mainly consists of account management and messaging. In this diagram, it only specifies high level classes of system.



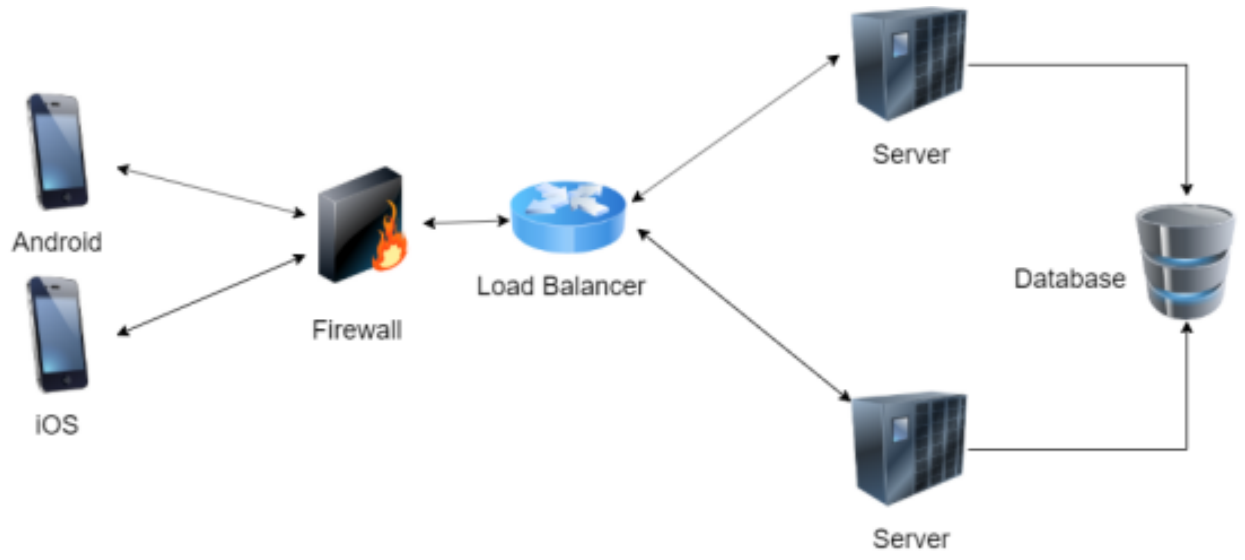
4 Architectural Design

Generally, this application uses client-server architecture. There are client apps for both Android and iOS systems. The conversation between users' needs to be transfers by the servers in back-end. The clients communicate with back-end servers in two protocols. 1) WebSocket protocol is used for messaging and voice / video chatting. WebSocket is a popular protocol for instant messaging apps since it allows bi-directional message push operation. 2) In login / logout operations, we use HTTP protocol to avoid maintaining a large numbers of TCP connections.

4.1 System Architecture

4.1.1 Network Topology

In our application, the client-side applications are native Android and iOS applications. The server-side application is a distributed system which can support a large number of concurrent connections. The following diagram shows the top-level topology of the whole system.



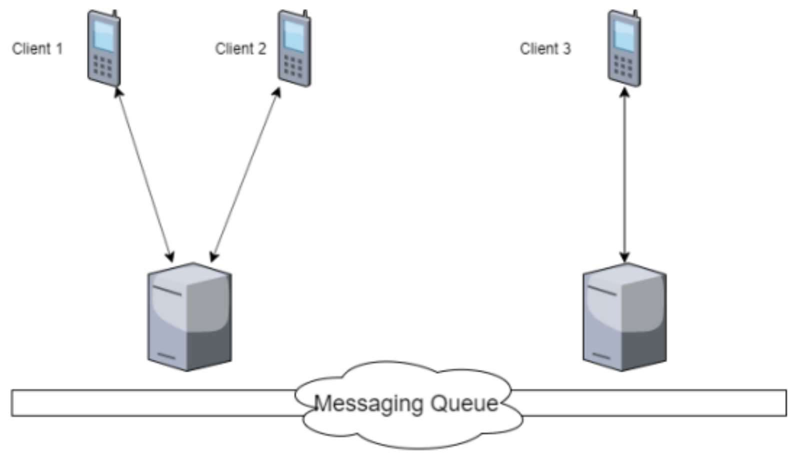
There are two kinds of databases in server-side application, one is a NoSQL database like Cassandra which stores chat history, the other is a relational database like Oracle which stores user account information.

4.1.2 Messaging

There may be many connections from client side to server side. When two users talk to each other, there are two scenarios:

1) The two connections from both users fall on the same server. In this scenario, the server transfers messages between two connections. The server also maintains a cache which stores the information of most recent active users.

2) The two connections from both users fall on different servers. In this scenario, we need a messaging queue in server side, like Kafka or RocketMQ. So, one server can push all messages from a connection to the queue, then another server can pull messages and resend them to another user.



The above diagram shows the messaging scenarios. Client 1 and client 2 communicate through one server, but client 2 and client 3 need to communicate through messaging queue in the server side.

4.1.3 Module Level Design

There are two major modules in client side: Business Logic and Data Transfer. Data Transfer module is responsible for transferring the data between clients and servers. According to different scenarios, it may use different protocols - HTTP or Websocket.

Business Logic module provides services to be called in user operations, like login, initiate messaging, etc.

Server side has five major modules:

Account Service - Maintains user account information.

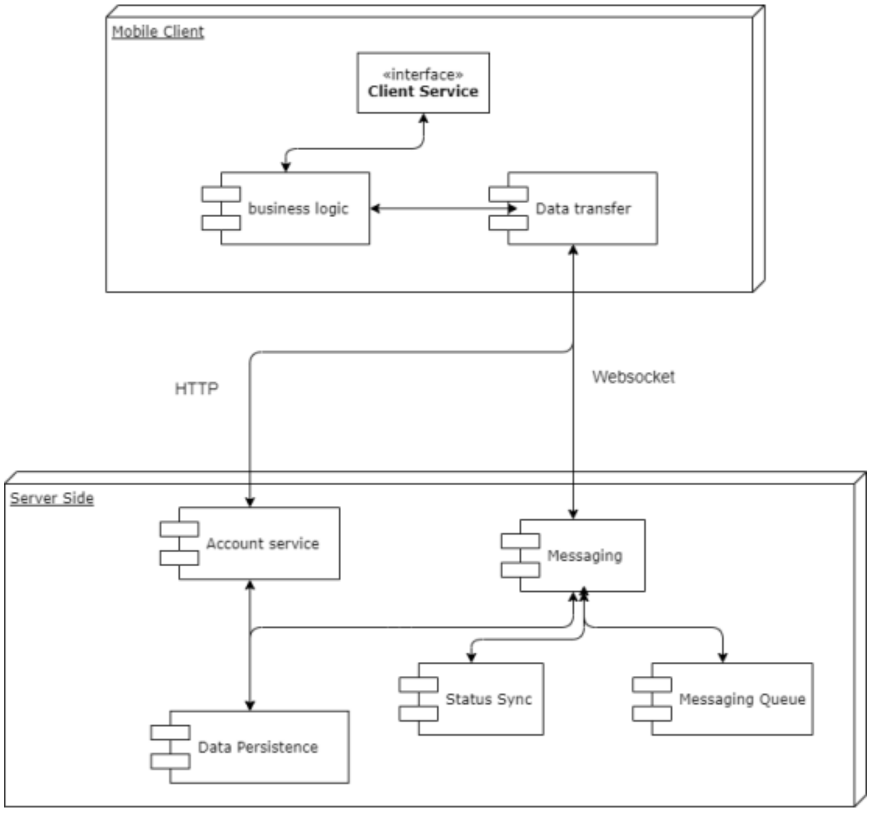
Messaging - Transfers messages, voice, videos between users.

Messaging queue - Transfers data between users.

Status Sync - Maintains the consistency of user status between different servers.

Data Persistence - Manage to store user account data and chatting history to different Databases.

The following diagram shows modules and their interactions.



4.2 Subsystems

QuickMessenger application consists of 2 subsystems. These are User Management, and Messaging.

a) Account Management

- User Management subsystem contains User Registration and User Account components. User Account component includes User Profile, Settings and Contacts subcomponents.
- User Registration component only works when a new user wants to register to the system.
- Settings subcomponent stores the user's selected application settings such as language and background color.
- On the other hand, User Profile subcomponent saves user information (e.g. phone number, name, …) and user's contact list. Also, User Profile subsystem has a relation with messaging and phone/video call subsystems' conversation components.

b) Messaging

- Messaging subsystems includes Conversation and Message Exchange components.
- Conversation component has Private Chat and Group Chat subcomponents. Group Chat only contains Text Chat while Private Chat has Text Chat, Voice Chat and Video Chat parts. In addition, Conversation component has a connection to User Profile component which is inside the User Management subsystem.
- Message Exchange component has Text Message and Multimedia Message subcomponents. Multimedia Message consists of Video, Image and Document parts. Besides, Message Exchange is responsible for message encryption and push notification functions.
- Message Exchange and Conservation components are related to each other. Because conversations use messages.

5 Class Responsibility Collaboration (CRC) Cards

Class Name: Chat	
Responsibility:	**Collaborators:**
Holds all chat related info including settings, members, dialog, etc. Supports both private chat and group chat scenarios.	Chat Settings Private Chat Group Chat

Class Name: Chat Settings	
Responsibility:	**Collaborators:**

Holds all chat settings. Allows user to configure the settings.	Chat

Class Name: Private Chat	
Responsibility:	**Collaborators:**
Handles the scenario of peer to peer chat. Provides text message exchange. Provides voice call. Provides video call.	Video Chat Voice Chat Text Chat

Class Name: Group Chat	
Responsibility:	**Collaborators:**
Handles the scenario of group chat. Provides text message exchange.	Text Chat

Class Name: Video Chat	
Responsibility:	**Collaborators:**
Supports video call.	

Class Name: Voice Chat	
Responsibility:	**Collaborators:**
Supports voice call.	

Class Name: Text Chat	
Responsibility:	**Collaborators:**
Supports text message exchange. Supports multimedia message exchange.	

Class Name: Account	
Responsibility:	**Collaborators:**
Holds user account info including profile info, settings, contacts and some server internal data related to user account.	Registration Profile Account Settings Contacts

Class Name: Registration	
Responsibility:	**Collaborators:**
Holds all registration related status and info in order to track and monitor registration progress and status.	Account

Class Name: Profile	
Responsibility:	**Collaborators:**
Holds user profile info such as username, phone number, avatar, etc. Allows user to change profile info.	

Class Name: Account Settings

Responsibility:	Collaborators:
Holds all account settings including privacy, notification, help, etc. Allows user to configure the settings.	

Class Name: Contacts	
Responsibility:	**Collaborators:**
Holds user’s contacts. Creates a new contact. Deletes an existing contact. Modifies an existing contact. Searches a contact.	

A Division of Labour

The work division of this document is shown in table as below.

Group Members	**Work**
Sajid Rahim	Section 1 and 2
Hong Sun	Section 2 and 4.1
Xiaodong Xu	Section 3 and 5
Baran Kaya	Section 4.2