Software Requirements Specification

for

Colligo

Version 1.0 approved

Prepared by Roarke DeCrewe, Enesh Jakhar,

Artem Khachaturov, Bassim Beshry, Jordan Pohr

February 12th, 2024

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Roarke DeCrewe	2024-02-16	First Original Release of Documentation	1.0

1. Introduction

1.1 Purpose

Our product is a communication platform, featuring real-time texting, voice and video chats, user authentication and more community based tools. This SRS covers the full scope of the product, including system features, nonfunctional requirements and user stories.

1.2 Document Conventions

Font Usage:

- Main Text: Times, 11pt font size.
- Headings: Bold and capitalized.
- Emphasis: Italics for emphasis where necessary.

Priority Assignment:

- Priorities for higher-level requirements are assumed to be inherited by detailed requirements unless stated otherwise.

1.3 Intended Audience and Reading Suggestions

This SRS is intended to be read by developers, project managers, testers and documentation writers. This SRS contains information about the program's functionality and features, going in depth with user and functional requirements. Suggested reading sequence is introduction, product description, system features, nonfunctional requirements, then user stories. Project managers and documentation writers can find most necessary information in the introduction and description sections. For developers and testers, features, requirements, and user stories, will contain most relevant information.

1.4 Product Scope

The application is designed for communication through various mediums, such as real-time messaging, voice chats and video calls. All of these can be used in one on one interactions, as well as on group servers. There are also features that will allow users to find servers and communities based on their interests. These community-building features will lead to a more dedicated and loyal fanbase, thus retaining and growing our audience.

2. Overall Description

2.1 Product Perspective

Our program is an alternative for other communication applications, such as Discord, Skype, Teams, and others, offering additional features to build a platform that would assist and encourage building communities of dedicated users based on their preferences.

2.2 Product Functions

Our program's features include:

- **Real-time Messaging:** Enable users to send and receive text messages in real-time.
- **Voice Calling:** Allow users to make voice calls to other users or groups.
- **Video Calling:** Enable users to conduct video calls with other users or groups.
- **User Authentication:** Provide mechanisms for user registration, login, and authentication.
- **Contact Management:** Allow users to manage their contacts and create groups.
- **Preference-based servers:** Show users different servers based on their preferences.
- **Customization:** Allow users to customize their profile and other settings.

2.3 User Classes and Characteristics

Casual Users:

- Characteristics: Primary audience utilizing messaging, voice, and video calling features for personal or professional communication.
- Frequency of Use: Engage with the application frequently.
- Technical Expertise: Basic to intermediate proficiency.
- Security or Privilege Levels: Standard user privileges.
- Education/Experience: Diverse backgrounds with varying experience levels.

Advanced Users:

- Characteristics: Highly engaged, utilizing advanced features extensively.
- Frequency of Use: Interact frequently, often spending extended periods of time within the platform.
- Technical Expertise: Advanced proficiency, comfortable with complex features.
- Security or Privilege Levels: Standard user privileges.
- Education/Experience: Generally have higher experience levels with communication tools.

Administrators:

- Characteristics: Responsible for managing user accounts and settings.
- Frequency of Use: Access the application regularly for administrative tasks.
- Technical Expertise: Advanced proficiency in user management.
- Security or Privilege Levels: Elevated privileges for account and settings management.
- Education/Experience: Typically possess technical or administrative backgrounds.

2.4 Operating Environment

Our platform will be a web-based application, working on most desktop systems with internet connection, such as Windows, Mac and Linux.

2.5 Design and Implementation Constraints

Difficulties that can arise when making our program include issues with timing requirements that have to be considered when building an application with real-time communication features. Additionally, reliance on existing APIs for much of the functionalities can be an issue, since our coders wouldn't have as much of a control over the application's features, as they would with writing the code for all of the features themselves. The program will only support the English language and run on desktop computers in its base version, however we are open to the possibility of adding more languages and expanding the list of supported devices.

2.6 User Documentation

Tutorials and guides on the usage of our application can be found in the settings of the program. These will ensure that even the less advanced users can understand how to fully utilize all of our platform's functionalities.

3. System Features

3.1 User Accounts

3.1.1 Description and Priority

Our app will feature an account's system, where users can create an account, login into a pre-existing account, edit account information and settings after logging in, and delete their account. This is a core feature of our application and is of **high** priority.

3.1.2 Stimulus/Response Sequences

A database of Colligo user's will be maintained which includes usernames, passwords, profile images, and user voice/voice settings. When a user is creating an account, the database will be checked to maintain username uniqueness, and after verifying a unique username, a new user will be added into the database with associated username and password. Similarly, returning users who login will have credentials compared to the database, and matches will allow a user access to their account. Voice/Video settings will be set as default according to the user's computer audio settings, but can be changed manually, leading to an update to the database to store said settings in a compressed form. User's may delete their account, and after verifying their login credentials, will have the account deleted from the database.

3.1.3 Functional Requirements

UA-REQ-1: The app will have a landing page with a register or login option
UA-REQ-2: Prospective users will be able to create accounts via the register option
- **UA-REQ-2.1:** User accounts require an unique username to register
- **UA-REQ-2.2:** Users with non-unique usernames will be told to change it prior to registration
- **UA-REQ-2.3:** User accounts require a password to register
- **UA-REQ-2.4:** Potential user accounts meeting **CS-REQ-1.1** and **CS-REQ-1.3** will be able to register and create a new account
- **UA-REQ-2.5:** Users with new accounts will be taken to the home page of Colligo

UA-REQ-3: Users will be able to login to an existing account with the login option
- **UA-REQ-3.1:** Users must input the correct username/password combination to login
- **UA-REQ-3.2:** Users that login will be taken to the home page of Colligo

UA-REQ-4: Users will be able to modify account settings
- **UA-REQ-4.1:** User accounts will have settings to modify audio options (**See Section 3.3**)
- **UA-REQ-1.2:** User accounts will be able to modify display name
- **UA-REQ-1.3:** User accounts will be able to modify profile picture

UA-REQ-5: Users will be able to delete their account after asking for login reverification prior to deletion

3.2 Servers

3.2.1 Description and Priority

Our app will feature a server interface which acts as a way for users to connect with each other. These server interfaces act as a community hub for users to share and chat. This is a core feature of our application and is of **high** priority.

3.2.2 Stimulus/Response Sequences

A database of server details will be maintained which will include server names, unique identifiers as well as a list of members with associated account information and potential permissions that have been granted. When user actions occur the database will be queried to provide the necessary data based on the users action.

3.2.3 Functional Requirements

CS-REQ-1: The user will be able create a server with a custom name and image
CS-REQ-2: The server will generate an invite link which gives access to the server
CS-REQ-3: The server will have modifiable settings
- **CS-REQ-3.1:** Settings can modify the name and image of the server
- **CS-REQ-3.2:** Settings can ban and unban users of the server
- **CS-REQ-3.3:** Settings can modify and create roles (See **CS-REQ-3**)
- **CS-REQ-3.4:** Settings can modify tags for the "For You" page (See System **Feature 3.5**)
- **CS-REQ-3.5:** Settings will contain a 'get-started' guide, which contains basic information on Colligo usage

CS-REQ-4: The server will have roles, defined as either administrator or user
- **CS-REQ-4.1:** The owner will be made an administrator upon server creation
- **CS-REQ-4.2:** The owner will be able to create customizable administrator and user roles
- **CS-REQ-4.4:** Administrators will be able to assign and remove roles from server members

CS-REQ-5: The server will have both voice and text channels
- **CS-REQ-5.1:** Administrators can create and delete voice and text channels
- **CS-REQ-5.2:** Voice and Text channels will have names assigned upon creation
- **CS-REQ-5.3:** Channels will have options for User or Administrator channels, limiting use to user's with a specific role level

3.3 Voice Communications

3.3.1 Description and Priority

Our app will feature voice communication in both servers and direct communication, allowing users to have synchronous communication utilizing desired audio input and output devices. This is a core feature of our application and is of **high** priority.

3.3.2 Stimulus/Response Sequences

Voice over Internet Protocol (VOIP) services will be used when users are utilizing either server or direct communication voice calls, with 3rd party libraries connecting users for communication. Users can access their account settings (**See Section 3.1**) to modify audio input and output devices.

3.3.3 Functional Requirements

VC-REQ-1: The user should be able to enter a voice chat with other users
VC-REQ-2: The user should be able to communicate with other users and be heard by said users
VC-REQ-3: The user should be able to mute and deafen themselves

VC-REQ-4: Administrators should be able to server mute and deafen users in voice chats

VC-REQ-5: Users should be able to edit input and output audio devices in account settings (See **Section 3.1**)

3.4 Text Communications

3.4.1 Description and Priority

Our app will feature text communication in both servers and direct communication, allowing users to have instantaneous messaging with one another using a 3rd party IM (Instant Messaging) library. Messages history for servers and direct messages will be saved to be viewed later.This is a core feature of our application and is of **high** priority.

3.4.2 Stimulus/Response Sequences

Instant Messaging services will be used when users are utilizing either server or direct communication voice calls, with 3rd party libraries connecting users for communication. A database will store server and direct messages allowing for users to see chat history. Users should be able to delete and update messages already sent, and the database should be updated appropriately.

3.4.3 Functional Requirements

TC-REQ-1: The user should be able to send messages in a server or privately to one another

TC-REQ-2: Users should be able to view messages previously sent in servers or in private DMs

TC-REQ-3: Users should be able to edit already sent messages

TC-REQ-4: Users should be able to delete already deleted messages

TC-REQ-4: Users should be able to use unicode characters as per UTF-16

3.5 “For-You” Page

3.5.1 Description and Priority

Our app will feature a “For-You” Page, which will recommend servers to users based on any tags that a server owner has set (See Section 3.1). Users can select tags on the “For-You” Page and show servers with associated tags. This is an optional feature of our application and is of **low** priority.

3.5.2 Stimulus/Response Sequences

Servers will be able to be public or private depending if they have tags or not. Server information on the database contains tags which will be matched against what the user input to give the user relevant servers. If multiple tags are selected, only servers which contain all relevant tags will be shown.

3.5.3 Functional Requirements

FYP-REQ-1: The user can access a “For-You” Page with tags lining the top of the page

FYP-REQ-2: Users can select tags for servers they wish to see, and servers with matching tags will be shown.

FYP-REQ-3: User can join servers after finding them via tags

3.6 Friends System

3.6.1 Description and Priority

Our app will have a friends system, where members of the same server can add one another as friends, and will then have access to direct communication (See Section 3.7). This is an important feature of our application and is of **medium** priority.

3.6.2 Stimulus/Response Sequences

A user can select members of a server they are a member of, and send them a friend request. A user with a friend request can either deny the request, or accept it and add the sender as a friend. Users who are friends can directly communicate with one another and will have message history as per **Section 3.4** A database will store information for a user regarding who is on their friends list and message history.

3.6.3 Functional Requirements

FS-REQ-1: Users in the same server can send friend requests to one another

FS-REQ-2: Users can decline incoming friend requests

FS-REQ-3: Users can accept incoming friend requests, with both users now having each other as a friend

FS-REQ-3.1: Database is updated with new friends information

FS-REQ-4: Users can communicate with one another directly, either with voice or text communication as per **Section 3.7**

3.7 Direct Communication

3.7.1 Description and Priority

Users who have added one another as friends (Section 3.6) will have the ability to directly communicate with one another via text and voice communication as per **Section 3.3** and **Section 3.4.** This is an important feature of our application and is of **medium** priority.

3.7.2 Stimulus/Response Sequences

Users once friends will have the ability to communicate with each other similarly to a server, with all the associated database and library dependencies as seen in mentioned sections.

3.7.3 Functional Requirements

DM-REQ-1: Users who are friends have access to direct communication
DM-REQ-2: Users who are friends can direct message as per **Section 3.4**
DM-REQ-3: Users who are friends can voice chat as per **Section 3.3**

4. Other Nonfunctional Requirements

4.1 Performance Requirements

Our app will prioritize a vast number of performance requirements in order to ensure a fast and reliable messaging system with room for scalability in the future. Some of these requirements include:

- Response Time: Messages should be delivered almost instantaneously. This will apply to regular chat messages as image and video transmission is another issue entirely.
- Scalability: Our app will be able to handle a large amount of users concurrently with an ability to scale resources dynamically based on server load.
- Reliability: Our app will aim to have 0 downtime when fully launched.
- Resource Efficiency: Due to the fact that this is a web app, we aim to minimize the amount of CPU memory and network bandwidth required to run the app.
- Latency: We hope to have low latency voice calls if voice calling is implemented.
- Compatibility: Our app will be compatible across all browsers.

4.2 Safety Requirements

Our app will not implement too many safety requirements due to the nature of a social media network, however we will do our best to define:

- Content Moderation: Our app will be moderated to protect against NSFW content such as gore, pornography, and other content considered inappropriate by most.
- User Reporting: Users should be able to report other users for posting things against our app's TOS. The things defined as against TOS will be stored in another document accessible to all users.
- Privacy Controls: Users should be able to determine who can view their profiles and who can message them.

4.3 Security Requirements

Some of the security requirements we wish to implement include:

- 2FA: Our app will attempt to implement 2FA or some sort of authentication check to ensure user's accounts are safe and secure.
- Account Recovery: Users will be able to recover their account in the event of a lost or forgotten password.
- Input Validation: Our app will have input validation to prevent SQL injection attacks while also ensuring data is always safe to process.

4.4 Software Quality Attributes

Some other quality attributes that our app will include are:

- **Adaptability:** Our app will work across almost all devices and should be developed with different browsers in mind.
- **Correctness:** Our app should have almost zero bugs and function exactly as intended.
- **Flexibility:** The app will be written in a way where future changes or updates can easily be accommodated and implemented.
- **Maintainability:** The app will be easily maintainable and written with well-organized code and clear documentation.
- **Robustness:** Our app will be able to handle unexpected inputs and conditions without crashing or malfunctioning.
- **Usability:** Our app will be easy to use with a clear and modern interface. This UI will make it simple to begin using for new users.

4.5 Business Rules

Some operating principles for our app are as follows:

- **User Roles:** Our app will support different user roles with varying permissions and capabilities. Administrators will have full control over the app and will be able to moderate content. Moderators will have the ability to manage content in their own servers and manage users as well. Regular users will have basic permissions.
- **Content Moderation:** Our app will have some sort of soft filter to moderate text, image and video messages. Administrators will also be in charge of filtering content.
- **Access Controls:** Our app will have controls in place to restrict certain actions based on the aforementioned user roles above.

5. Other Requirements

5.1 Database Management System Requirements

Colligo requires a database system to store account information (**Section 3.1**), Server Information (**Section 3.2**), Text History (**Section 3.4**), and Friend's information (**Section 3.6**). The database will be partitioned into separate areas for storing said information, with all relevant information being stored internally. Said database has the following requirements
Some other quality attributes that our app will include are:

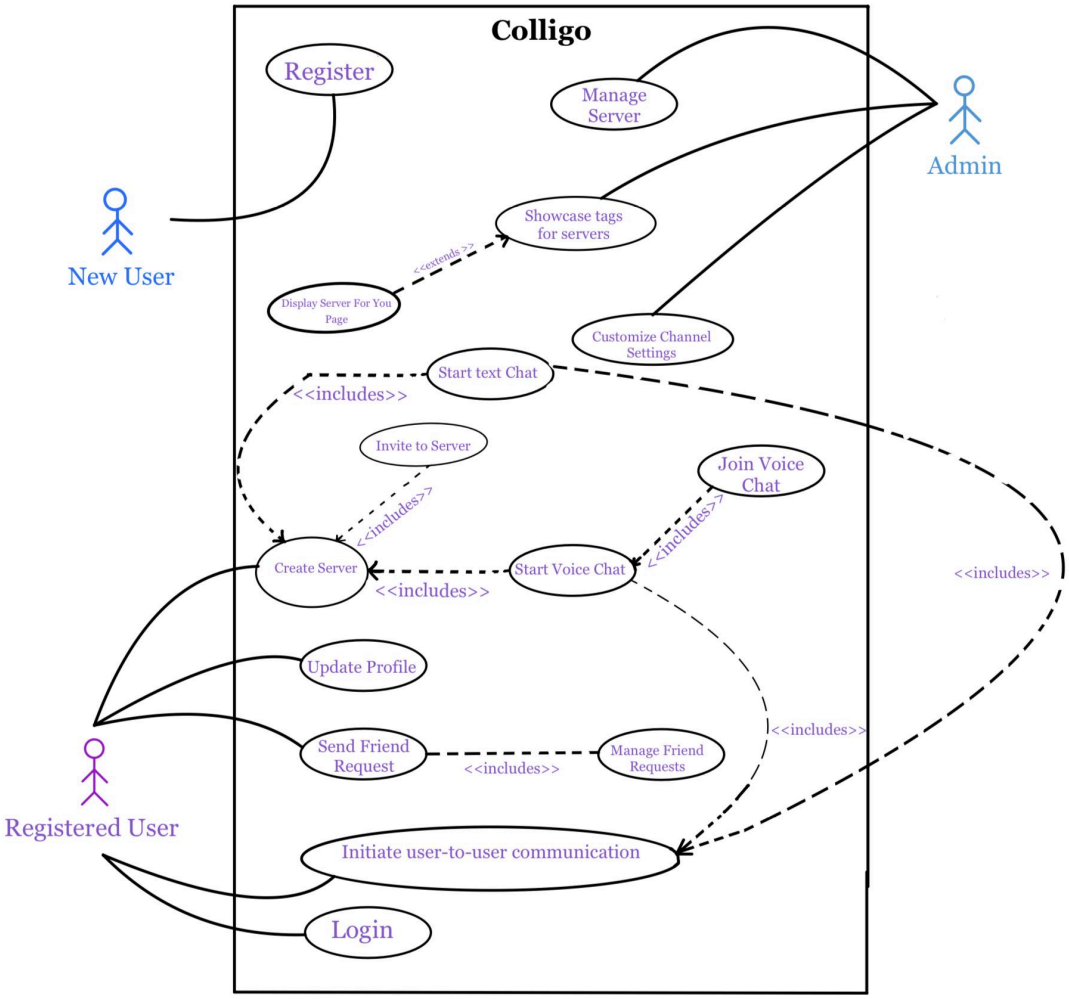
- **Large storage capacity:** As more users and more messages leads to an increased amount of data, the database hosting used needs to be scalable to handle a large amount of data.
- **Quick response time:** Pulling data from the database should have a minimal get and put time, and should ideally preload data when possible
- **Well-defined structure:** The internal structure of our database management system should have clearly defined tables, databases, and objects as to make it clear where and how to pull specific data to the app

5.2 Library Requirements

Colligo requires multiple libraries for it to function, but the main two that are necessary for the app to function, a Voice over IP (VOIP) and an Instant Messaging (IM) library to facilitate the two main functions of the app.

- **Voice over IP Library:** A Voice over IP library should be used which allows change to audio settings and can support a decent amount of users simultaneously with little effect on voice quality or voice ping
- **Instant Messaging Library:** An instant messaging library should be used which allows users to message simultaneously on a large scale with minimal effect on send and receive time.
- **Easy integration of libraries:** All libraries used in Colligo should be easily integrated within the system with easy cross-communication between libraries and the programmed service.

6. USE CASE Diagram



1. Register

- **Use Case Name:** Register
- **Primary Actor:** Unregistered User
- **Description:** Allows a new user to create an account within the system.
- **Precondition:** The user does not already have an account.
- **Postcondition:** The user has a new account and can access system features.
- **Main Scenario:**
    - User navigates to the registration page.
    - User enters required information (e.g., email, password).
    - System validates the information and creates a new account.
    - User receives a confirmation message.
- **Extensions:**
    - If the information is invalid, the user is prompted to correct it.
- **Assumptions:** The system has a mechanism for validating new account requests.

2. Login

- **Use Case Name:** Login
- **Primary Actor:** Registered User
- **Description:** Enables registered users to log into the system to access their accounts and utilize available features.
- **Precondition:** The user has already created an account with the system.
- **Postcondition:** The user is logged into the system and can access features available to their account type.
- **Main Scenario:**
    - User navigates to the login page.
    - User enters their username and password.
    - System validates the credentials.
    - Users are granted access to their account.
- **Extensions:**
    - If credentials are incorrect, the user is notified and asked to try again.
    - If the user has forgotten their password, provide an option to reset it.
- **Assumptions:** The system has robust security measures to protect user credentials and personal information.

3. Create Server

- **Use Case Name:** Create Server
- **Primary Actor:** Registered User
- **Description:** Enables users to create a new channel within the system.
- **Precondition:** The user is logged in.
- **Postcondition:** A new channel is created and available for others to join.

- **Main Scenario:**
  - User selects the option to create a new channel.
  - User specifies channel details (e.g., name, description).
  - System creates the channel based on provided details.
- **Extensions:**
  - If details are incomplete or invalid, prompt the user to correct.
- **Assumptions:** Users can create multiple channels.

4. Invite to Server

- **Use Case Name:** Invite to Server
- **Primary Actor:** Admin/Registered User
- **Description:** This use case allows a Server administrator or authorized members to invite other users to a specific channel.
- **Precondition:** The actor must have administrative privileges or permissions to invite users.
- **Postcondition:** Invited users receive an invitation and can choose to join the channel.
- **Main Scenario:**
  - Administrator/member opens the server's settings.
  - Selects the option to invite users.
  - Enters the username or email of the invitee.
  - System sends an invitation to the users.
- **Extensions:**
  - If the user does not exist, the system notifies the inviter.
- **Assumptions:** The system supports notifications and has a method for users to accept invitations.

5. Manage Server

- **Use Case Name:** Manage Server
- **Primary Actor:** Admin
- **Description:** This use case allows channel administrators to manage and configure channel settings, including member roles, permissions, and content.
- **Precondition:** The user must be an administrator of the channel.
- **Postcondition:** The channel is updated according to the administrator's changes, affecting member access and interaction.
- **Main Scenario:**
  - Administrator accesses the channel's management dashboard.
  - Administrator selects the aspect of the channel to manage (e.g., roles, permissions, content settings).
  - Administrator makes necessary changes and updates.

- **Extensions:**
    - If the system encounters an error while updating, revert changes and notify the administrator.
    - If unauthorized changes are attempted, deny access and alert the administrator.
- **Assumptions:** Administrators have clear permissions for different levels of channel management and the system supports real-time updates without requiring restarts.

6. Start Voice Chat

- **Use Case Name:** Start Voice Chat
- **Primary Actor:** Registered User/Channel Administrator
- **Description:** Allows a user or channel administrator to start a voice chat session within a channel or group, facilitating live audio communication among members.
- **Precondition:** The actor is a member or administrator of the channel/group with permission to start voice chats.
- **Postcondition:** A voice chat session is started, and members are notified or able to join.
- **Main Scenario:**
    - User navigates to the channel/group where they wish to start voice chat.
    - User clicks on the option to start a voice chat.
    - System creates a voice chat session and notifies members of the channel/group.
    - Members join the voice chat.
- **Extensions:**
    - If the system fails to start the voice chat, notify the user and suggest troubleshooting steps.
- **Assumptions:** The platform supports live audio transmission and can handle multiple concurrent voice chat sessions.

7. Join Voice Chat

- **Use Case Name:** Join Voice Chat
- **Primary Actor:** Registered User
- **Description:** Enables users to join an ongoing voice chat in a channel or group.
- **Precondition:** User is a member of the channel or group with an ongoing voice chat.
- **Postcondition:** User successfully joins the voice chat.
- **Main Scenario:**
    - User selects the channel or group with an ongoing voice chat.
    - User clicks on the join voice chat button.
    - User is connected to the voice chat.
- **Extensions:**
    - If the voice chat is full, notify the user and offer to queue for the next available slot.
- **Assumptions:** There is a limit to the number of participants in a voice chat.

8. Update Profile

- **Use Case Name:** Update Profile
- **Primary Actor:** Registered User
- **Description:** Users can update their profile information, including bio, profile picture, and contact details.
- **Precondition:** User is logged in.
- **Postcondition:** User's profile is updated with the new information.
- **Main Scenario:**
    - User accesses their profile settings.
    - User updates desired profile information.
    - Changes are saved and immediately reflected in the user's profile.
- **Extensions:**
    - If the new information violates platform policies, the user is notified and asked to correct it.
- **Assumptions:** Users have the freedom to change their profile information at any time.

9. Customize Channel Settings

- **Use Case Name:** Customize Channel Settings
- **Primary Actor:** Admin
- **Description:** Enables the channel administrator to customize settings such as channel name, description, privacy settings, and user roles.
- **Precondition:** The actor is an administrator of the channel.
- **Postcondition:** Channel settings are updated according to the administrator’s modifications.
- **Main Scenario:**
    - Administrator accesses the channel settings.
    - Modifies the desired settings (name, description, privacy, roles).
    - Saves the changes, which are immediately applied.
- **Extensions:**
    - If invalid settings are provided, the administrator is prompted to correct them.
- **Assumptions:** Changes to settings do not require system restart or logout.

10. Manage Friend Requests

- **Use Case Name:** Manage Friend Requests
- **Primary Actor:** Registered User
- **Description:** Allows users to send, view, and respond to friend requests, enabling them to manage their contacts within the system.
- **Precondition:** Users are registered and logged in.
- **Postcondition:** Friend requests are appropriately managed, resulting in new connections or the declination of requests.

- **Main Scenario:**
  - User navigates to the friend requests section.
  - User views incoming friend requests.
  - User accepts or declines the requests.
- **Extensions:**
  - If there are no friend requests, the system shows a relevant message.
- **Assumptions:** Users can send friend requests to other registered users.

11. Start Text Chat

- **Use Case Name:** Start Text Chat
- **Primary Actor:** Registered User
- **Description:** Enables users to start a text chat session within a server or channel, facilitating real-time text-based communication among members.
- **Precondition:** The user is a member of the server/channel and has permission to initiate text chats.
- **Postcondition:** A text chat session is started, allowing members to communicate in real-time.
- **Main Scenario:**
  - User navigates to the desired server/channel.
  - User selects the option to start a text chat.
  - System opens a text chat window, enabling real-time messaging.
  - Members of the server/channel can join the chat and participate.
- **Extensions:**
  - If the system encounters issues opening the chat, notify the user and provide troubleshooting advice.
- **Assumptions:** The platform supports text messaging features and allows for multiple concurrent text chat sessions.

12. Display Server For You Page

- **Use Case Name:** Display Server For You Page
- **Primary Actor:** System
- **Description:** Automatically generates and displays a curated list of server suggestions on the "For You" page, based on the user's interests, activities, and networking patterns.
- **Precondition:** User is logged in and has interacted with content or members within the platform.
- **Postcondition:** User receives personalized server suggestions, enhancing discovery and engagement.
- **Main Scenario:**
  - System collects data on the user's interests and interactions.
  - System analyzes data to identify relevant server suggestions.
  - "For You" page is updated with tailored server suggestions.

- **Extensions:**
    - If no relevant suggestions can be made, display general or popular servers.
- **Assumptions:** The system has access to robust analytics tools to accurately suggest servers.

13. Showcase Tags for Servers

- **Use Case Name:** Showcase Tags for Servers
- **Primary Actor:** Admin
- **Description:** Allows server administrators to assign tags to their servers, facilitating easier discovery by users based on specific interests or themes.
- **Precondition:** The server exists and the administrator has the necessary permissions to edit server details.
- **Postcondition:** Servers are more discoverable by users through tag-based searches.
- **Main Scenario:**
    - Administrator accesses server settings.
    - Administrator assigns relevant tags to the server.
    - System updates server visibility based on tags.
- **Extensions:**
    - If tags are misused or irrelevant, system administrators may review and adjust.
- **Assumptions:** Users utilize tags for discovering new servers that match their interests.

14. Initiate User to User Communication

- **Use Case Name:** Initiate User to User Communication
- **Primary Actor:** Registered User
- **Description:** Enables direct messaging between users, facilitating private text, voice communications outside of public servers or channels.
- **Precondition:** Both users are registered and have accepted friend requests or have mutual server memberships.
- **Postcondition:** A private communication channel is established between the users.
- **Main Scenario:**
    - User navigates to another user's profile or selects them from a friend list.
    - User selects the option to initiate communication (text, voice, or video).
    - System establishes a private session for the users to communicate.
    - Users engage in private communication.
- **Extensions:**
    - 3a. If the recipient is not available, allow the sender to leave a message.
- **Assumptions:** The platform supports various forms of direct communication and ensures privacy and security for these interactions.

7. User Stories (Roarke/Enesh/Artem)

User Story: Organize Gaming Sessions
As a user, I want to use Discord to organize and coordinate gaming sessions with my friends and gaming community.

- Acceptance Criteria:
    - I create a new text server within my Discord server dedicated to a specific game or gaming community.
    - I announce the date, time, and game title for the upcoming gaming session in the server description.
    - Using voice chat, I join a voice chat room designated for the gaming session.
    - I invite my friends or fellow gamers to join the gaming session by sharing the server link or sending direct invitations.

User Story: Managing Friend Requests
As a user, I want to streamline the process of handling friend requests, enabling me to connect with acquaintances and maintain a curated list of contacts.

- Acceptance Criteria:
    - Upon receiving a friend request notification, I navigate to the "Friends" section in my dashboard.
    - I review the pending friend requests, which display the usernames and profile pictures of the users who sent them.
    - For requests from classmates, colleagues, or known individuals, I accept the requests to expand my network within relevant communities.
    - If I receive requests from unfamiliar or unwanted users, I have the option to decline the requests, ensuring my friend list remains focused on meaningful connections.

User Story: Creating an Account
As a user, I want to be able to create a Colligo account, so that I can chat and message with my friends

- Acceptance criteria:
    - User clicks on create account, and inputs a username and password for account creation.
    - If a duplicate username is found, the user is prompted to enter a new one.
    - If the username is unique, an account will be created, and the user now has access to Colligo

User Story: Creating a Server
As a user, I want to create a server for my friends to join, so that we can hang out and talk over Colligo

- Acceptance criteria:
    - User navigates to create a server button at the bottom of the sidebar, pulling up the creation menu.
    - User selects a name and uploads a profile image for the server, or if either is not done, the server will be called ‘Username’s Server’ and the image will be the Colligo logo.
    - The server is now created, and invite links can be sent and server settings can be changed.

User Story: Administrator Moderation
As a server admin, I want to be able to mute, timeout, and kick users on my server, so that I can moderate any disruptions.

- Acceptable criteria:
    - Server administrator clicks on user
    - Administrator chooses between moderating features such as, mute, timeout, or kick from the server.

User Story: Finding new servers
As a casual user, I want to find new servers based on my interests, because I want to join a new community and find people with similar interests.

- Acceptable criteria:
    - Click on the “For You” button to access a page
    - User can select tags of interest
    - User receives a list of recommended servers.