COSC 310 (Team Bamboozlers)

Requirements Document

What the project is: Discord Clone

Description:

Our project will be a Messaging Platform allowing users to create accounts, add friends and create groups with or send and receive messages to/from. It will also allow users to become moderators of groups and control members permissions within said groups.

1. Introduction

1.1 Purpose of the Requirements Document

The purpose of this document is to outline the user requirements and the functional purpose of the Discord Clone, a messaging platform. It will serve as a comprehensive guide for the development team to understand what features and functionalities are required, ensuring that the final product aligns with user expectations and project objectives.

1.2 Scope of the Product

The Discord Clone is a messaging platform that enables users to create accounts, add friends, form groups, and communicate through messages. It also includes features for users to act as moderators in groups, managing member permissions and overseeing group activities.

1.3 Definitions, Acronyms, and Abbreviations

- **User**: An individual who interacts with the Discord Clone platform.
- **Moderator**: A user with elevated permissions within a group to manage members and settings.
- **Group**: A chat environment where multiple users can communicate simultaneously.
- **DM (Direct Message)**: A direct message sent from one user to another.

1.4 References

Discord: https://discord.com/

1.5 Overview of the Remainder of the Document

The remainder of this document is divided into sections detailing the general description of the product, specific requirements including functional and nonfunctional aspects, followed by appendices and an index for easy navigation.

2. General Description

2.1 Product Perspective

The Discord Clone is an independent product, designed to provide an intuitive, user-friendly messaging experience. It aims to incorporate the best features of existing messaging platforms while introducing unique functionalities tailored to user needs.

2.2 Product Functions

- Account creation and management
- Friend list management
- Group creation and management
- Messaging (both private and group)
- Moderator roles and permissions in groups

2.3 User Characteristics

Users are individuals seeking a communication platform. They range from casual users who want to keep in touch with friends to businesses, teachers, and group leaders who need to manage large groups.

2.4 General Constraints

- The platform must ensure data privacy and security.
- It should be scalable to accommodate a growing number of users.

2.5 Assumptions and Dependencies

- Continuous internet connectivity for users.

3. Specific Requirements

3.1 User Requirements

- Users must be able to register, login, and manage their profiles.
- Users must be able to add and remove friends.
- Users must be able to create and manage groups
- Users must be able to send, receive, edit, and delete messages.
- Moderators in groups must be able to manage member roles and permissions.

3.2 Functional Requirements

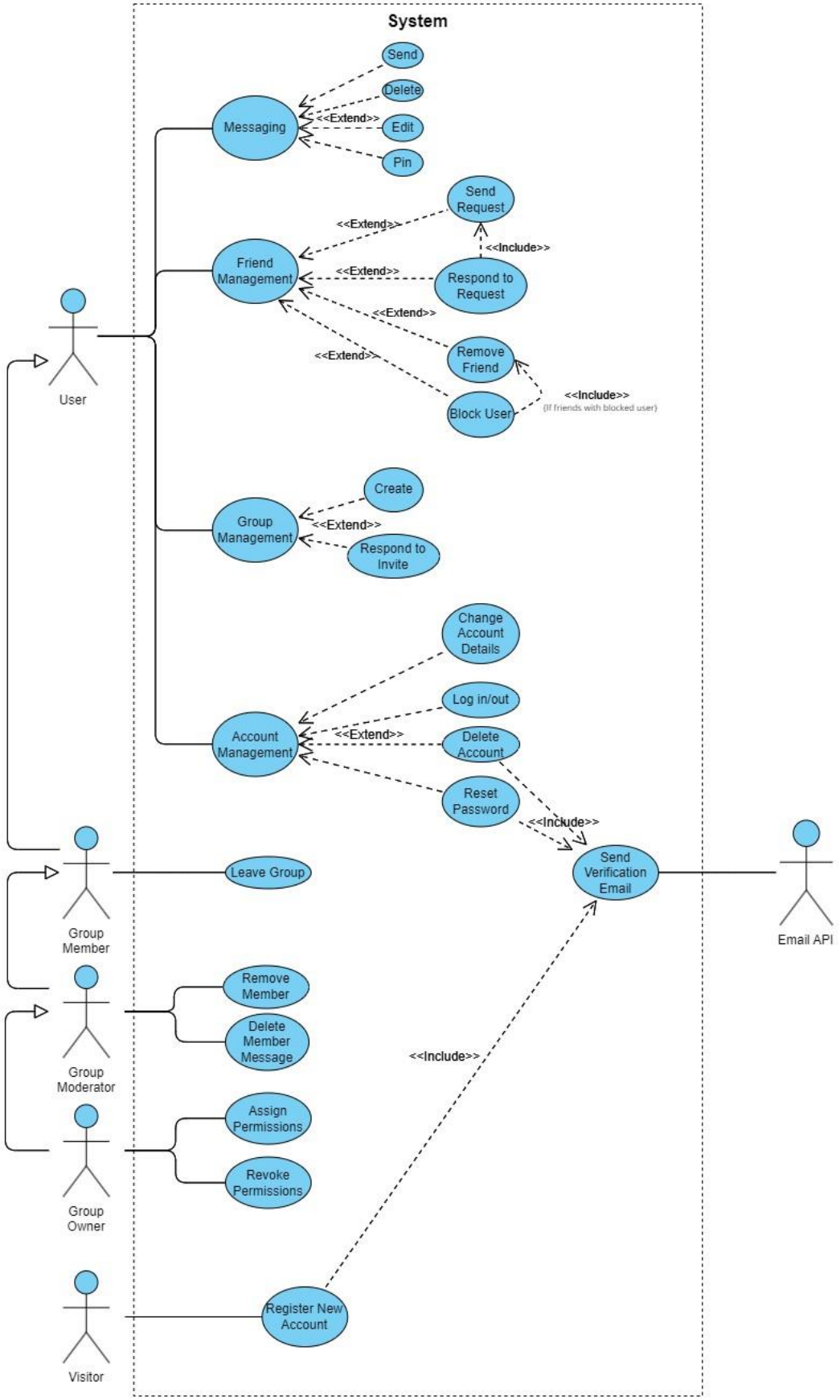
- The system must allow Users to register using their email.
- The system must save the account information of users to the database.
- The system must send a verification email when a new User attempts to register.
- The system must send a confirmation email when a user successfully registers, changes their password or email.
- The system must allow users to update their profile and username.
- The system must update the database when Users edit their account information.
- The system must allow users to change their email or password.
- The system must allow users to access their friend list.
- The system must allow users to send and accept friend requests.
- The system must update the database and users friend lists when they accept friend requests.
- The system must support the creation and management of groups.
- The system must update the database and message history when users send, receive, edit, pin, and delete messages.
- The system must allow moderators to manage memeber roles and permission.
- The system must update the database if moderators change member roles and permission.

3.3 Non-Functional Requirements

- **Performance & Reliability**: The system should be able to handle multiple users performing system functions such as messaging, registering, friend requesting.
- **Scalability**: Should accommodate an increasing number of users and groups while maintaining performance and reliability.
- **Security**: Robust security measures to protect user data and privacy. Policies such as email verification when resetting password and safe password and email storing.

4. Appendix

4.1 Use Case Diagram



4.2 Detailed Use Cases

Use Case	Details	Scenarios
Use Case 1. Messaging	* Primary Actor: Registered User * Precondition: User is registered and logged in * Postcondition: Varies	1. User navigates to and opens a direct message chat or a group message chat a) Send Message 1. User types a message and presses enter or "Send" button 2. Server takes message and saves it in the database 3. Message is distributed via server to recipients 4. Message is displayed for all involved clients <u>Extensions</u> a1. User attempts to send a message to another user they are not friends with * System issues an error message to the user and informs them they cannot send messages to non friends b) Attaching Media 1. User clicks on a paperclip icon and selects a file from their device 2. Server takes file and saves it in the database 3. Message with file is distributed via server to

		recipients 4. Message with file is displayed for all involved clients <u>Extensions</u> b1.User inserts unsupported file type * System issues an error message and requests user to attach a supported file type c) Delete Message d) Edit Message e) Pin Message
Use Case 2. Manage Friends	* Primary Actor: Registered User * Precondition: User is registered and logged in * Postcondition: Varies	a) Send Friend Request 1. User navigates to and opens "Friends" tab 2. User clicks "Add friend" button and searches the username of their friend 3. User finds their profile and sends them a friend request 4. The friend request is stored in the database by the server 5. The friend sees the pending request in their own "Friends" tab <u>Extensions</u> a1. User attempts to send a friend request to an existing

		friend *System informs user that this is already an existing friend a2. User cancels the friend request * Server removes the outgoing friend request from the database b) Accept Friend Request c) Remove Friend d) Block User 1. User navigates to another user's profile 2. User opens hamburger menu, selects "Block User" and confirms their choice 3. Server adds the blockship to the database 4. Neither user can send friend requests to each other 5. The blocked user cannot see the profile of the user and the user cannot see messages sent by the blocked user <u>Extensions</u> d1. User was friend with the blocked user * Server removes the user's friendship with the blocked user from the database d2. User had pending friend request from the blocked user

		* Server removes the pending friend request from the database d3. User had outgoing friend request to the blocked user * Server removes the outgoing friend request from the database
Use Case 3. Manage Groups	* Primary Actor: Registered User * Precondition: User is registered and logged in * Postcondition: Varies	a) Create Group 1. User clicks on "Create Group" button and inputs details, confirming their choice 2. Server takes request and creates a new Group and GroupChat 3. Server sets the user as the Owner, and adds a corresponding GroupModerator and GroupMember row 4. User now sees the Group in their list of groups b) Join Group c) Leave Group 1. User navigates into Group settings page and selects "Leave Group," confirming their choice 2. Server takes request and removes User from Group, removing corresponding database entries 3. User no longer sees the group in their list of groups <u>Extensions</u>

		c1. User is the Group Owner * Server assigns the oldest appointed GroupModerator as the new Owner
Use Case 4. Manage Group Members	* Primary Actor: Group Moderator or Owner *Precondition: User is assigned Group Moderator or is a Group Owner, User is registered and logged in * Postcondition: Varies	a) Invite Member(s) b) Remove Group Member 1 c) Delete Member Message 1. Group Moderator selects a member's message and opens the hamburger menu, selecting "delete" 2. Server takes request and deletes the message from the database 3. All clients in the group remove the message from view d) Assigning Permissions 1. Group Owner selects a member in the member list, selects "Assign moderator", and confirms their decision 2. Server adds corresponding GroupModerator row for the Group Member 3. New Moderator can now see moderator options and privileges e) Removing Permissions
Use Case 5. Account Management	* Primary Actor: Registered User * Precondition: Varies *Postcondition: Varies	a) Log Into Account 1. Registered User navigates to website home page on browser 2. User enters their credentials

		and confirms them 3. Server checks credentials against database and opens an authenticated logged-in session 4. User is redirected to their landing page Extensions a1. User enters incorrect credentials *System issues an error and informs user their credentials are incorrect b) Log Out of Account c) Change Username 1. Registered, logged-in User navigates to their settings, then to account details 2. User selects "change username" and enters a new username, confirming it 3. Server checks the new username against database entries 4. Server changes the username in the database if there are no conflicts 5. User's profile representation changes in all views for all other users and themselves Extensions c1. User selects a username that is already in use *System issues and error and

		requests user to select another username d) Change Avatar e) Change Description <u>Extensions</u> e1. User attempts to enter a description beyond the allotted length *System prevents user from continuing to input characters and displays the maximum number of characters in red f) Change Password g) Reset Password h) Change Email i) Delete Account
Use Case 6. Registration	* Primary Actor: Visitor * Precondition: None * Postcondition: A new Registered User is created	1. Visitor navigates to sign-up screen 2. Visitor adds their desired credentials, e.g., Username, Email, Password (twice). 3. Visitor confirms their details. Server checks their Username and Email against the database. 4. If there are no conflicts, the Server sends a validation email to the specified email. 5. Visitor navigates to the email and confirms by clicking the link. 6. Server creates the new User's account and records it

		in the User table, with the link automatically logging the user in. <u>Extensions</u> 1. User selects an email/username that is already in use *System issues an error and informs user that the email/username is already associated with an account and to enter a different one 2.User selects password that does not meet minimum security requirements *System issues an error and informs user that they must choose a password that meets security requirements