Boston University

Electrical & Computer Engineering

EC464 Capstone Senior Design Project

User's Manual

Stöd

Submitted to

by

Team Members:

Sharith Godamanna | sharithg@bu.edu

Camden Kronhaus | kronhaus@bu.edu

Quinn Meurer | quinnyyy@bu.edu

Alexander Trinh | aktrinh@bu.edu

:

Submitted: 04/13/2021

Stöd

Table of Contents

Executive Summary

As team Stöd we are aiming to build a virtual platform where users of all backgrounds are empowered to engage in meaningful conversation. We envision Stöd to be an online forum where people come to converse with others about sensitive, uncomfortable, and even embarrassing topics without fear of judgement or scrutiny. On Stöd's forum interface users will be able to anonymously share their motivational stories, ask taboo questions, and make follow up comments and questions to other users' posts. Users will be able to keep conversations going by privately messaging individual posters and commenters.

1 Introduction

Modern social media platforms make it easy for users to share content around mutual circumstances with others, but aren't as useful for opening up about sensitive topics. For people facing troubles such as depression, alcoholism, mental health issues, coming out of the closet, etc. it can be hard to find a safe, supportive online environment. We want to make a platform where it's easy for people to share the problems they face with others in similar situations, both in a group setting and 1:1 setting. We want to facilitate the building of long lasting connections between people that helps them get through their problems ***together***.

When users want to visit our app, they simply open www.stod.app in a web browser of their choice and will be directed to the login screen. Our app is served over HTTPS, including the communication between the backend server connections, to ensure security and privacy. When a user attempts to sign up, there are various security measures in place to ensure that a user cannot sign up with a password that is too common, too short, or otherwise not secure. We allow users to choose their own username, and only share that username with other users to ensure anonymity.

If a user attempts to post within a group they are subscribed to, we have a keyword checker to determine if the user is potentially posting something harmful. We decided on a system whereby a post that is flagged by our automatic checker is sent for approval by a moderator. This approval system allows moderators to review posts that are flagged, and either approve them, automatically posting the post in its desired place, or denying the post, deleting it from our app. We decided on this system because it allows for some degree of control since moderators would not be able to watch posts and review them 24/7 and before they are read by many others. The manual approval system in addition to the auto-moderation allows for more control over what posts are allowed to be posted, and allows for “false positives” to be posted.

We also have implemented a tagging and filtering system. With this system, a user can post with specific tags such as “personal story” or “resource” but more importantly with information such as trigger warnings. Along with the convenience of the filtering mechanism to quickly look at all posts that are marked as personal stories or resources posted by other users, they can filter out certain triggers that they do not wish to view. This was an important feature to allow users to feel safer while browsing the sensitive topics found on our app.

The last feature we wanted to highlight was the messaging feature of our app. A user can send friend requests directly through posts of other, anonymous users. We do not have a profile system for users to further protect the identity of the other users. We do this to reduce the possibility of profiling other users by easily looking at all the posts they are publishing and trying to identify who they are. When a user sends a friend request, and the other user accepts it, they can open up an instant chat with each other. We implemented the chat feature because we found it very important to allow users to reach out to each other directly for help, guidance, and one on one support with each other.

Our app as a whole contains various additional features and tools to make the experience as good as possible for all users, and we are continuing to further tune the above features as well. We will describe our architecture as a whole, and how we allow for modularity by using docker containers and microservices in our server. In this way, more features can be developed more easily and fully in the future.

2 System Overview and Installation

The Stod application is built using a microservice architecture currently in three parts: a chat server, a general backend server, and a frontend which communicates with both backend components using HTTPS requests. Both the chat server and the general backend server run on a public Ubuntu server tied to https://stodbackend.app and communicate locally with Nginx to allow for secure communication from requests coming from users' browsers interacting with the front end. The chat server runs using MongoDB and Node.js while the general backend server runs using Django and PostgreSQL. Both are started by running their respective docker builds to allow for a consistent startup process. The front end is statically generated and hosted automatically with pushes to a production branch in Github using Netlify. The front end can be reached by going to https://www.stod.app. In the front end, we are using typescript in order to use type safe variables and for easier debugging, as well as Redux to manage global state. To connect our frontend to the backend, we created asynchronous redux actions and used the axios library to make HTTPS requests to our backend services.

2.1 Overview block diagram

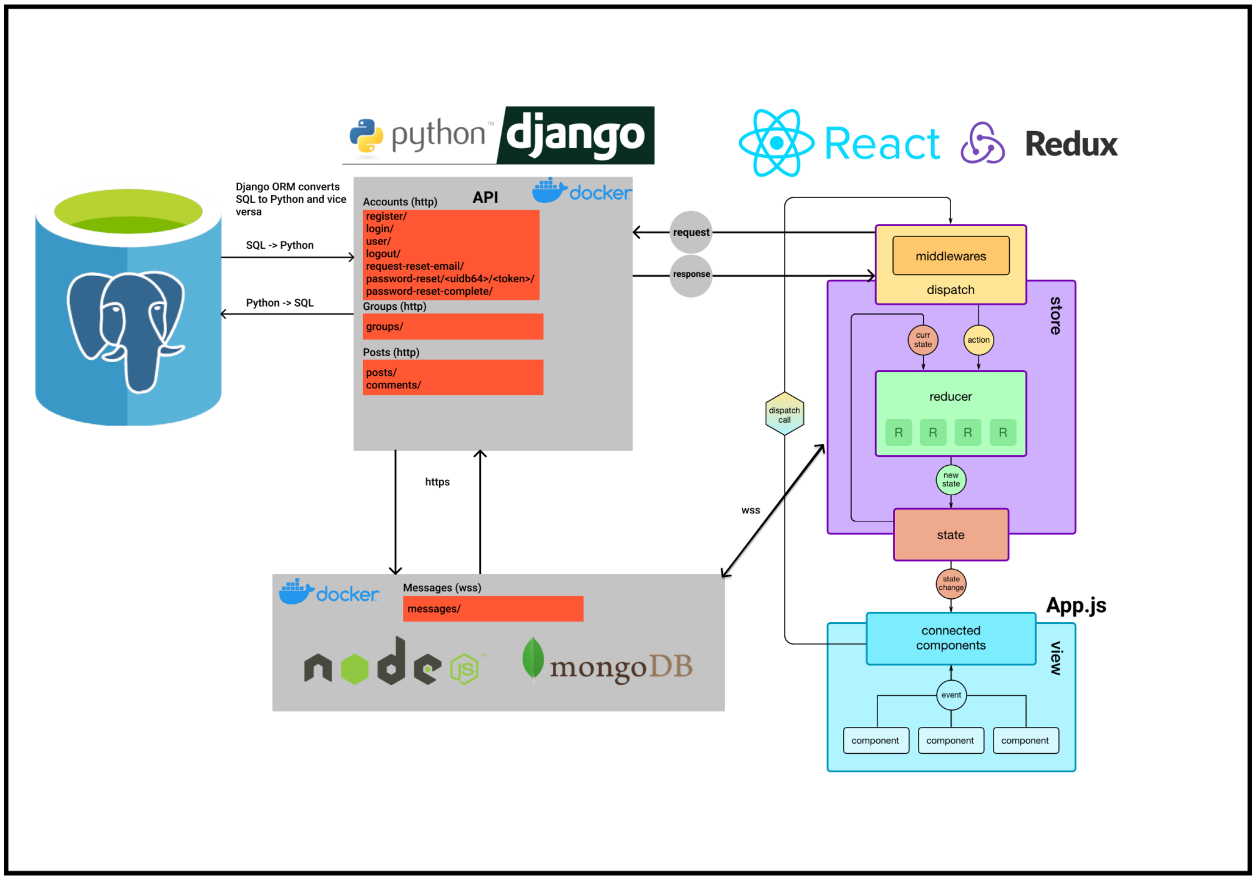
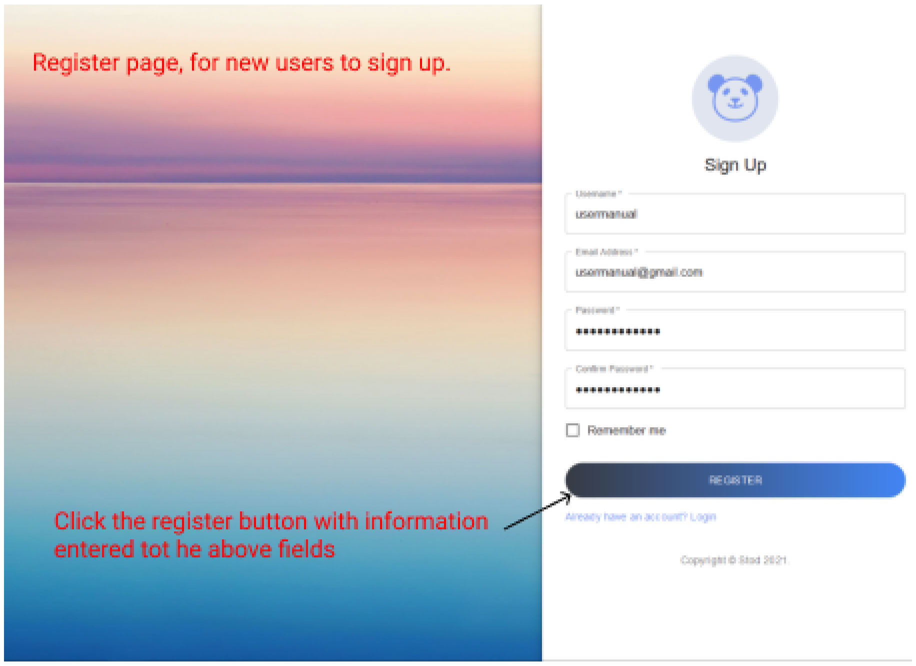


Fig 2.1.0

2.2 *User interface.*

When the user first joins Stöd they will need to register for the website by entering a username, email address, and a password.



Register page, for new users to sign up.

Sign Up

Username *: usermanual

Email Address *: usermanual@gmail.com

Password *: •••••••••••••

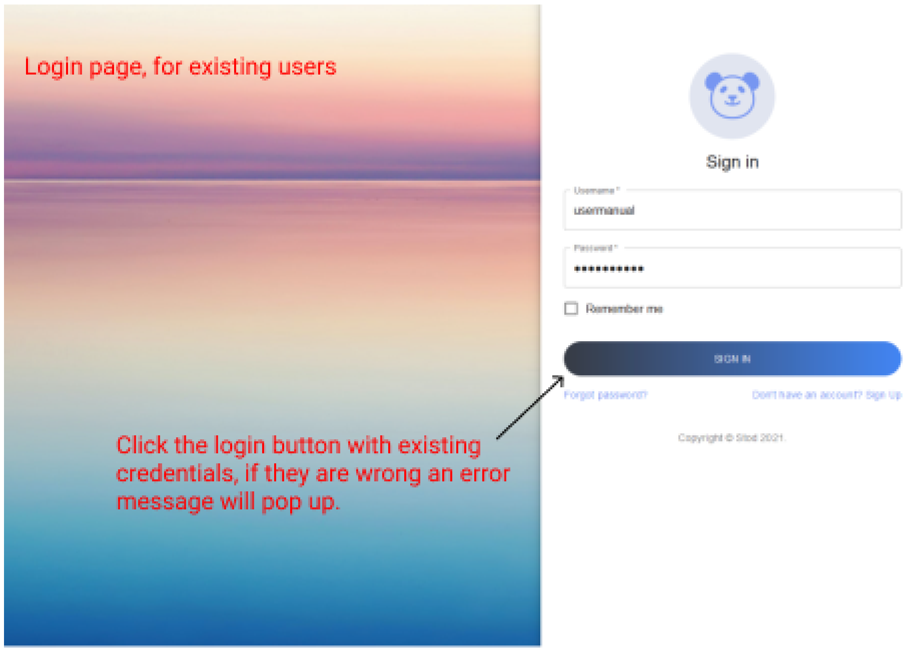
Confirm Password *: •••••••••••••

☐ Remember me

Click the register button with information entered tot he above fields

Fig 2.2.0 login page

Whenever the user wants to access Stöd at a later time they will need to log in by providing their username and password.



Login page, for existing users

Sign in

Username *: usermanual

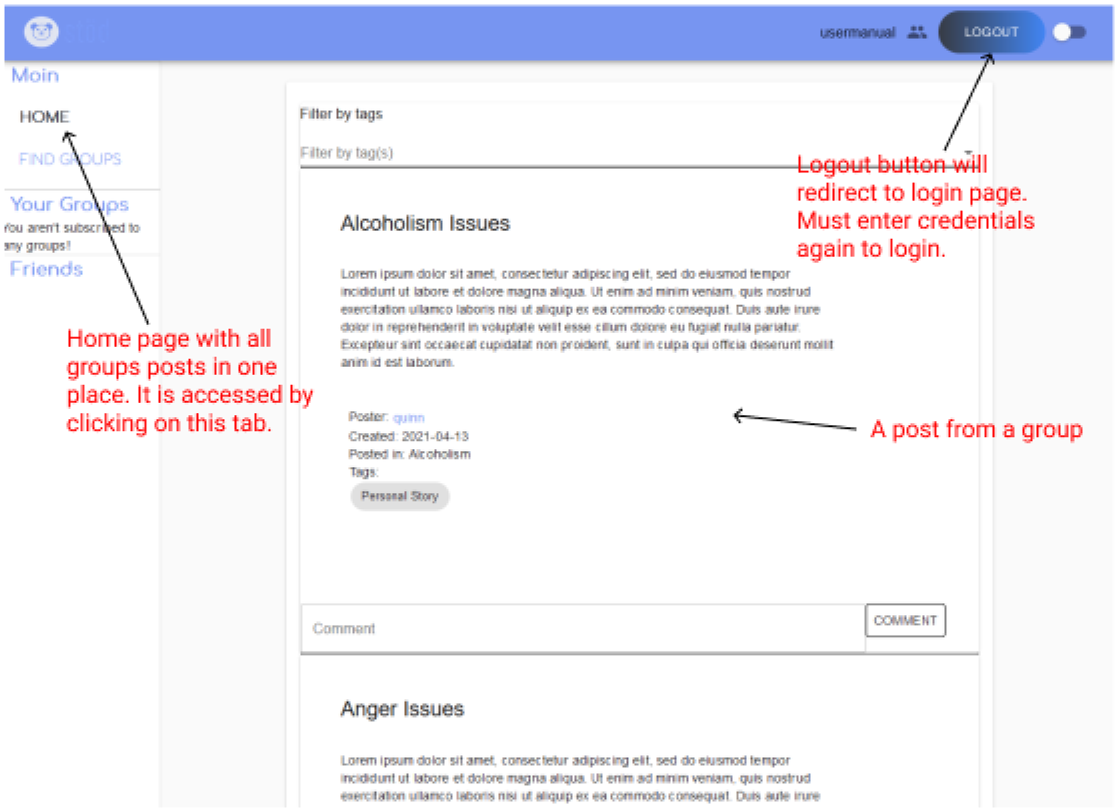
Password *: •••••••••••

☐ Remember me

Click the login button with existing credentials, if they are wrong an error message will pop up.

Fig 2.2.1 register page

After logging in, the user can view the home page where a sidebar displays groups and friends, and the main section of the page displays posts from all groups. Since this is a new user no friends/groups will be displayed in the sidebar yet.



You aren't subscribed to any groups!

Logout button will redirect to login page. Must enter credentials again to login.

Home page with all groups posts in one place. It is accessed by clicking on this tab.

A post from a group

Alcoholism Issues

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

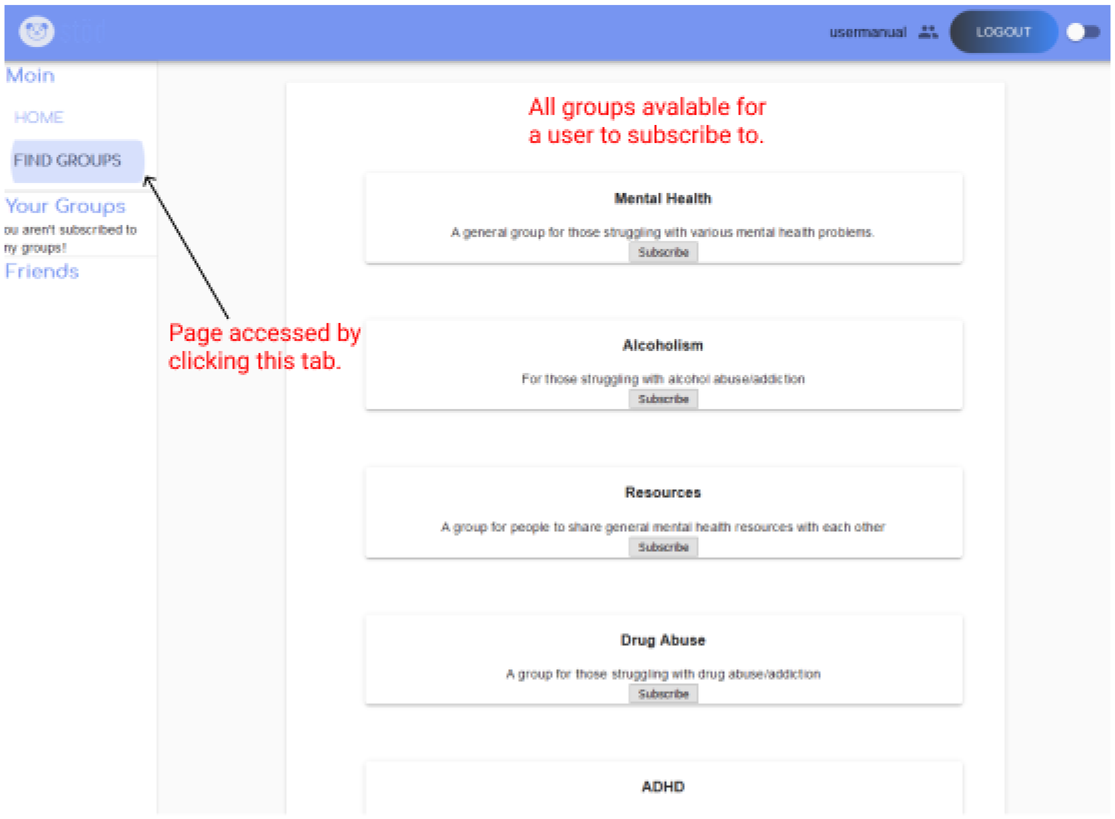
Poster: quinn
Created: 2021-04-13
Posted in: Alcoholism
Tags:

Anger Issues

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure

Fig 2.2.2 home page

The user can then navigate to “Find Groups” from the sidebar to choose groups to subscribe to.



Mental Health
A general group for those struggling with various mental health problems.
Subscribe

Alcoholism
For those struggling with alcohol abuse/addiction
Subscribe

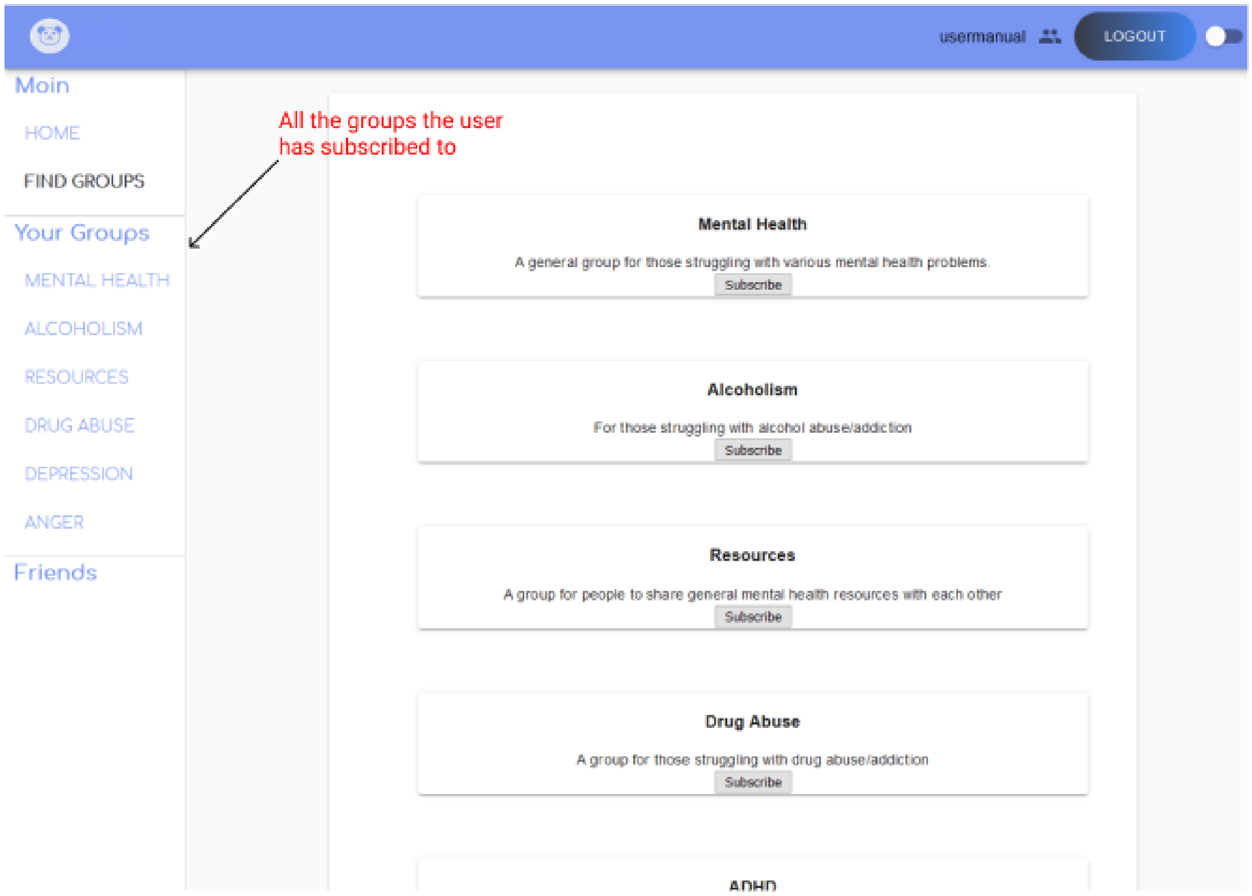
Resources
A group for people to share general mental health resources with each other
Subscribe

Drug Abuse
A group for those struggling with drug abuse/addiction
Subscribe

ADHD

Fig 2.2.3 find groups page

We can now view the subscribed groups in the sidebar.



Mental Health

A general group for those struggling with various mental health problems.

Alcoholism

For those struggling with alcohol abuse/addiction

Resources

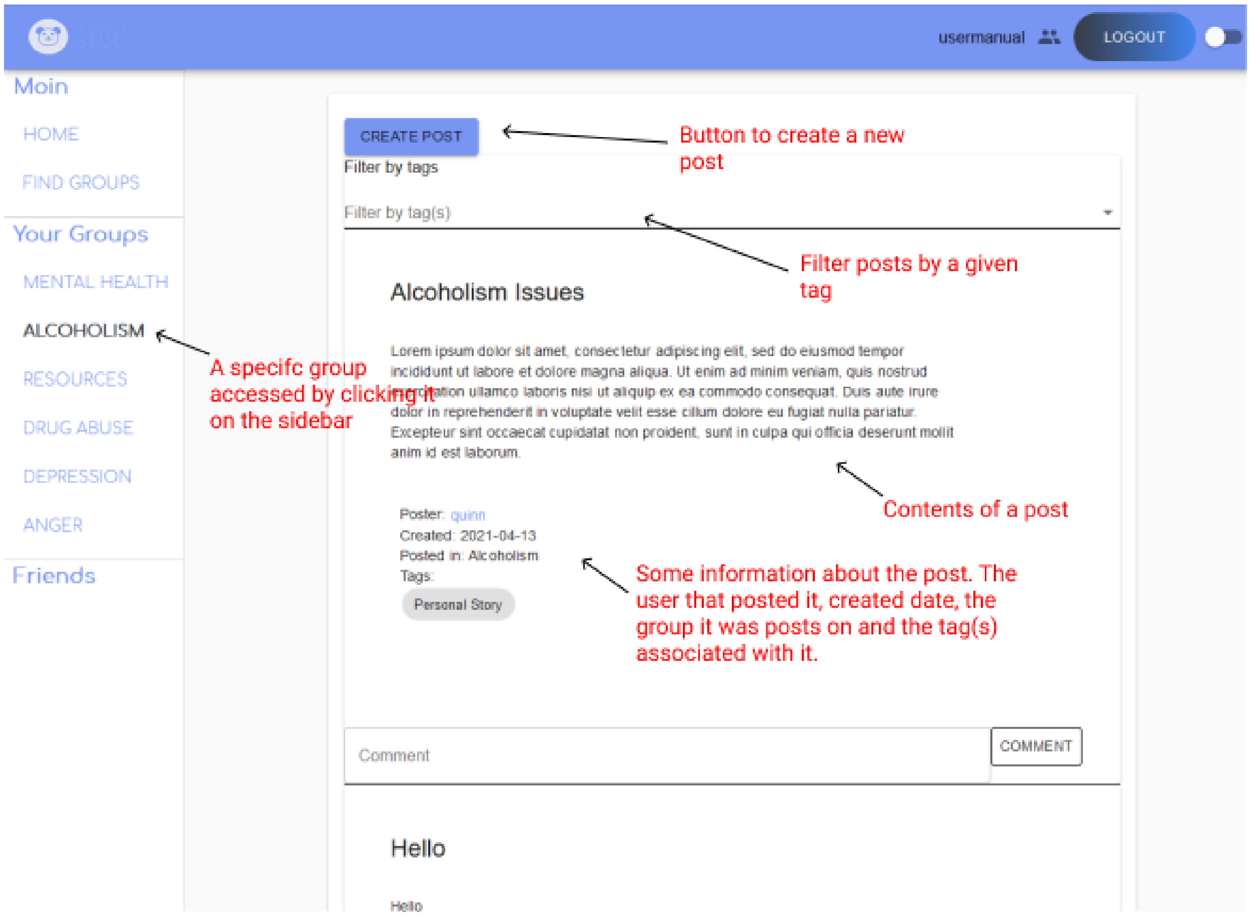
A group for people to share general mental health resources with each other

Drug Abuse

A group for those struggling with drug abuse/addiction

Fig 2.2.4 users groups page

If we click on a group in the sidebar we can view only the posts pertaining to that group.


usermanual
LOGOUT
Main
HOME
FIND GROUPS
Your Groups
MENTAL HEALTH
ALCOHOLISM
RESOURCES
DRUG ABUSE
DEPRESSION
ANGER
Friends
CREATE POST
Filter by tags
Filter by tag(s)
Button to create a new post
Filter posts by a given tag
A specifc group accessed by clicking it on the sidebar
Contents of a post
Some information about the post. The user that posted it, created date, the group it was posts on and the tag(s) associated with it.
Personal Story
Comment
COMMENT

Alcoholism Issues

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Poster: quinn
Created: 2021-04-13
Posted in: Alcoholism
Tags:

Hello

Hello

Fig 2.2.5 specific group page

The user can also create a new post in the group.

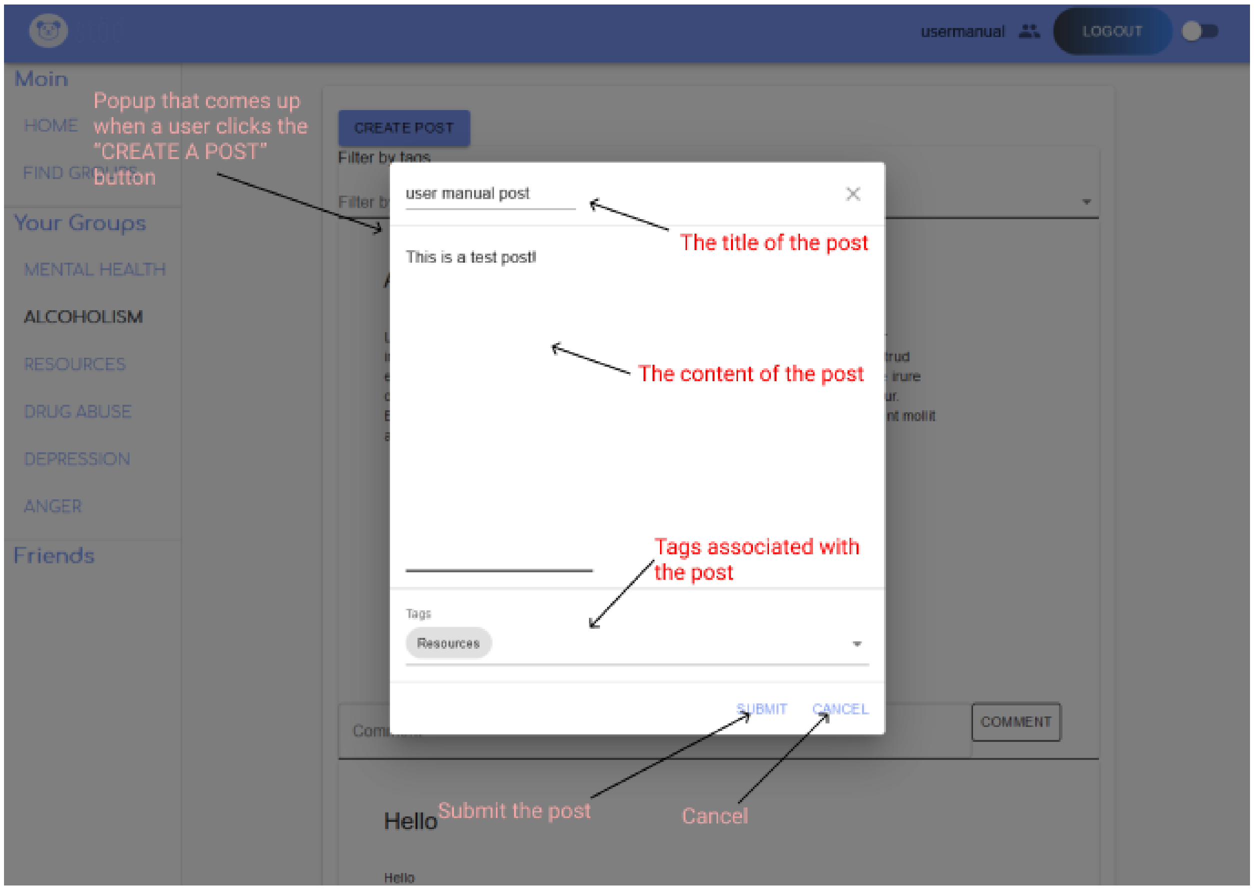
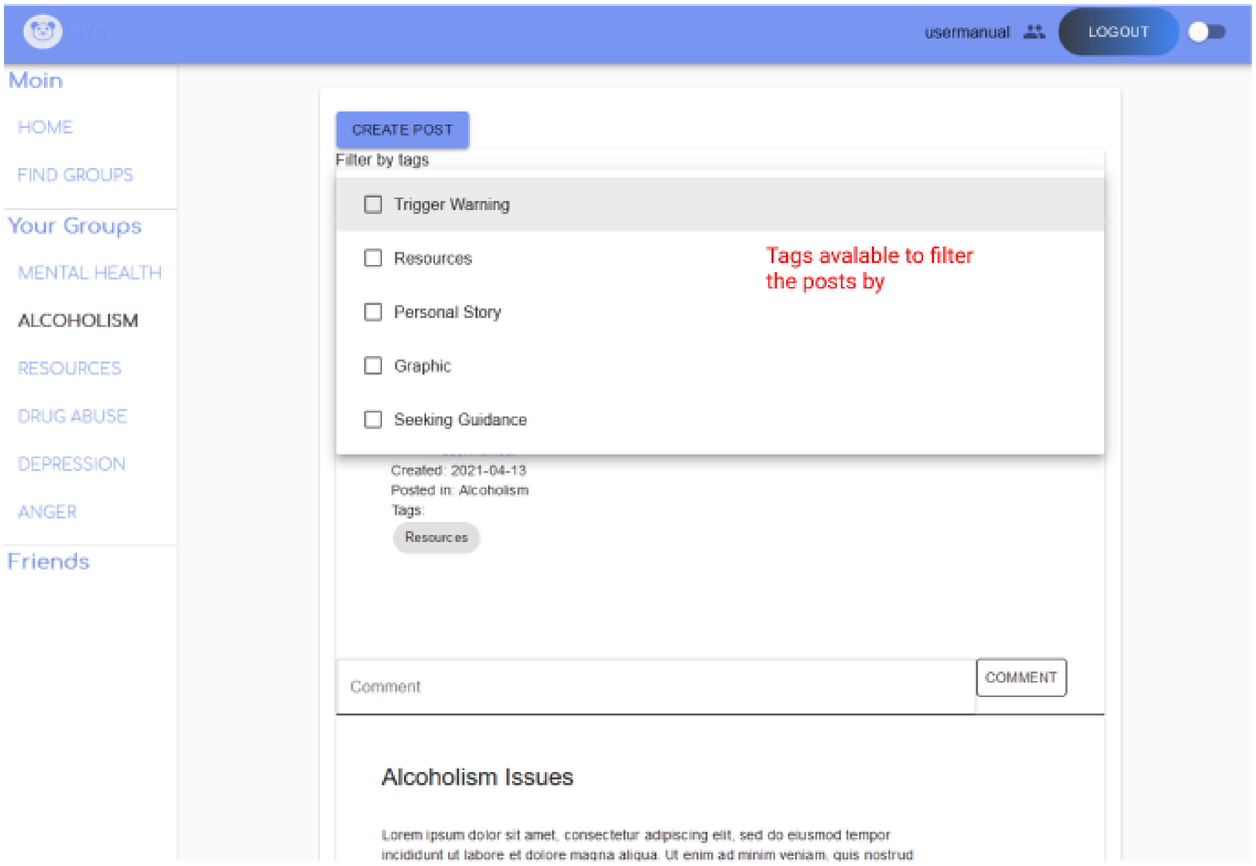


Fig 2.2.6 Add new post

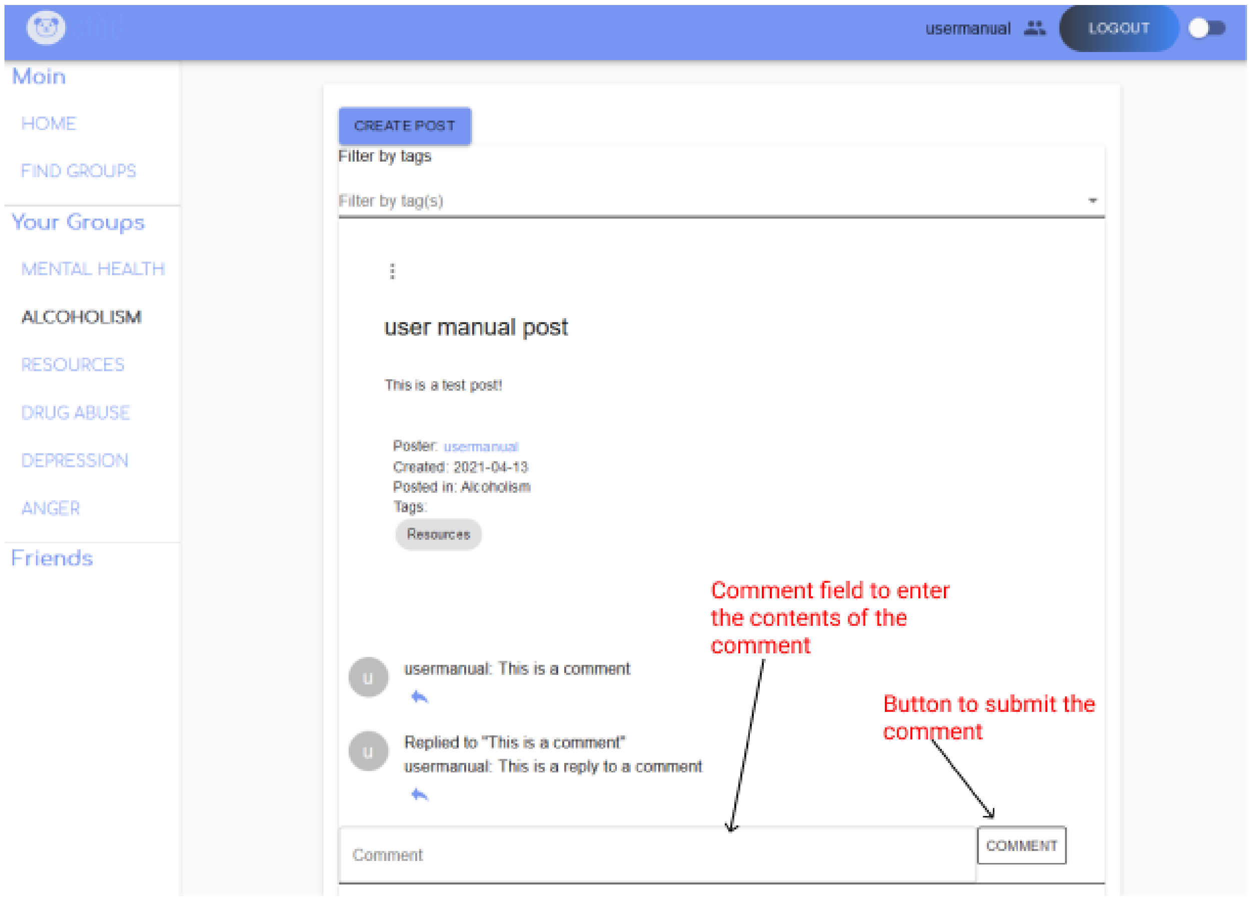
The user can also filter posts by their tag(s).



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

Fig 2.2.7 tags list

The user can also comment on posts and reply to comments



user manual post

This is a test post!

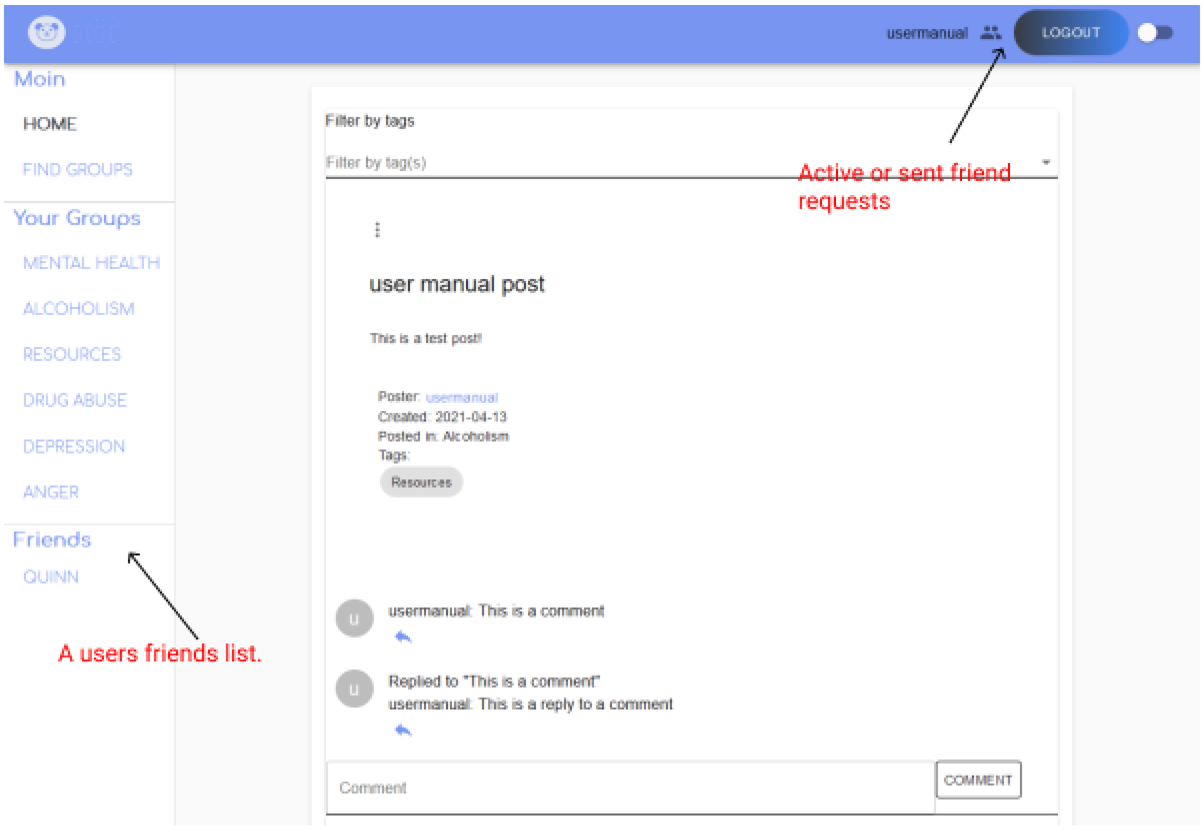
Poster: usermanual
Created: 2021-04-13
Posted in: Alcoholism
Tags:

usermanual: This is a comment

Replied to "This is a comment"
usermanual: This is a reply to a comment

Fig 2.2.8 Commenting page

To further connect with other users, the user can send a friend request.



user manual post

This is a test post!

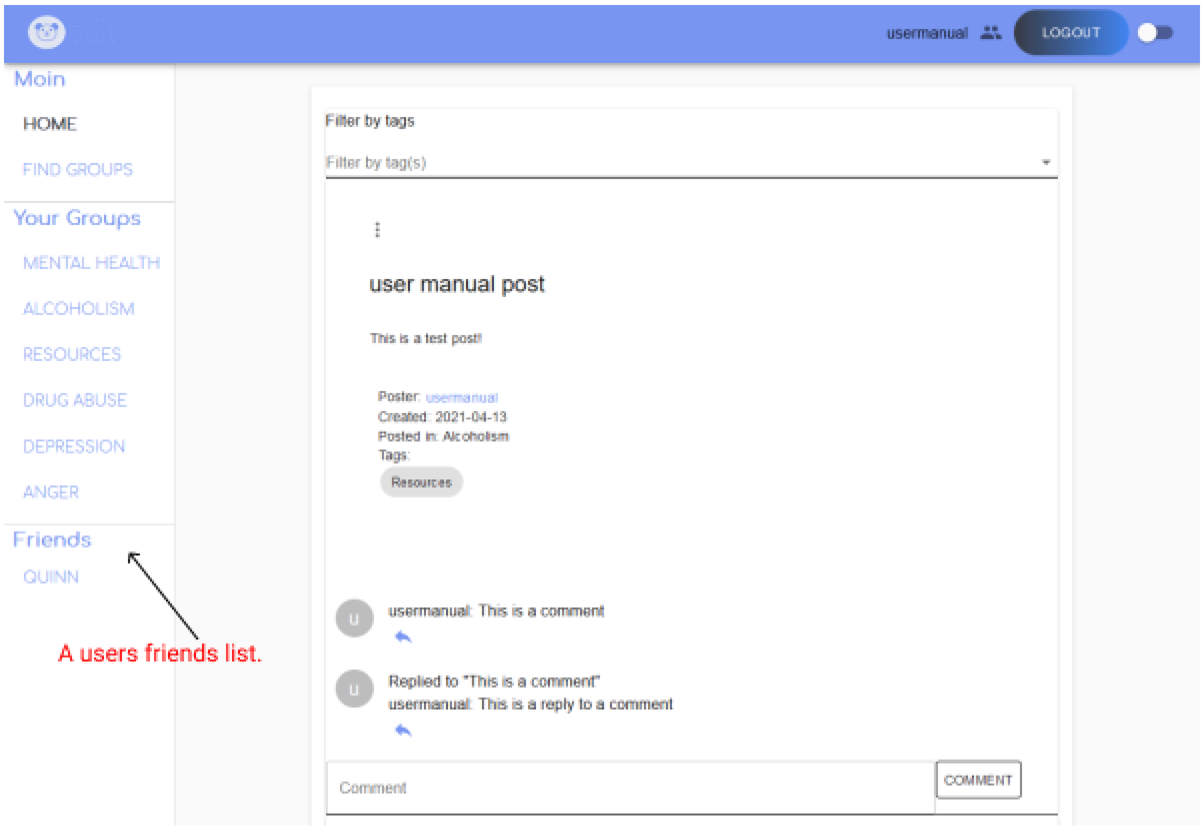
Poster: usermanual
Created: 2021-04-13
Posted in: Alcoholism
Tags:

usermanual: This is a comment

Replied to "This is a comment"
usermanual: This is a reply to a comment

Fig 2.2.9 Friend requests

When the other user accepts the friend request, they will appear as a friend in the sidebar.



user manual post

This is a test post!

Poster: usermanual
Created: 2021-04-13
Posted in: Alcoholism
Tags:
Resources

usermanual: This is a comment

Replied to "This is a comment"
usermanual: This is a reply to a comment

Fig 2.2.10 Friend requests

Upon clicking on their name the user will have access to a real-time chat service with the other user.

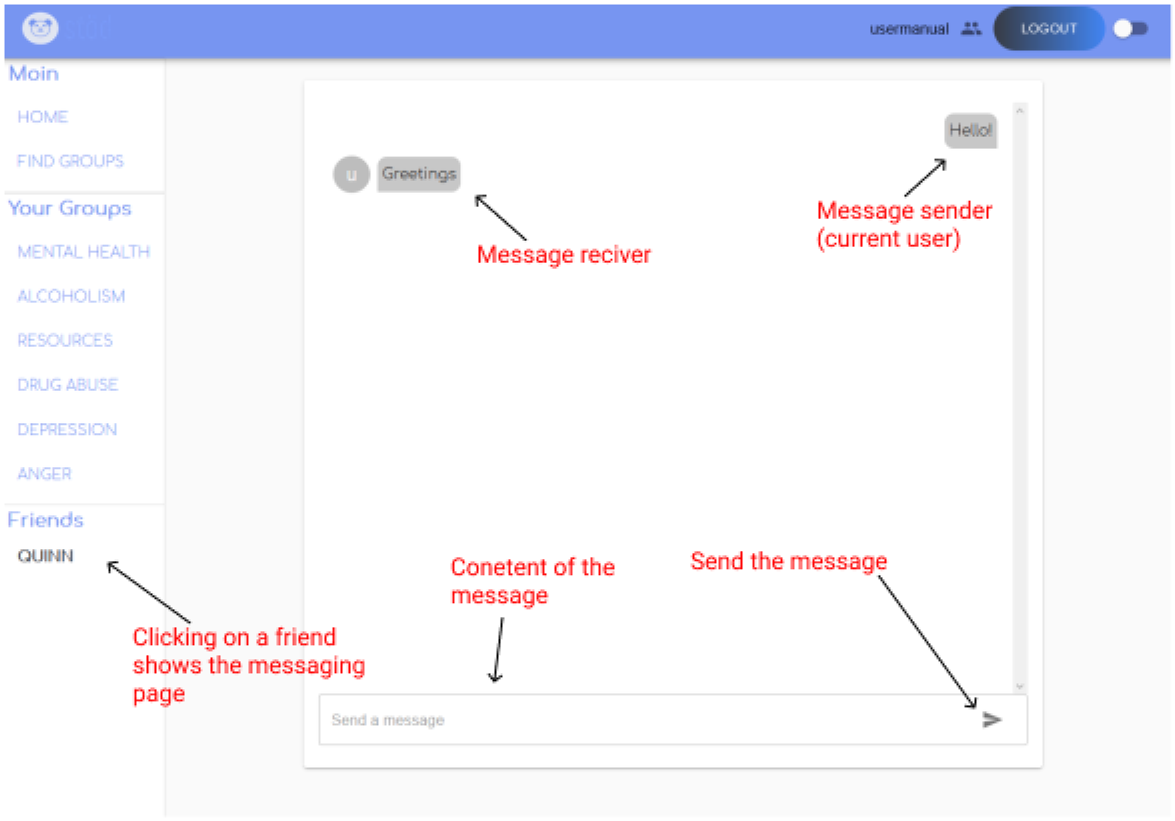


Fig 2.2.11 Messaging

2.3 *Installation, setup, and support*

All a user needs to do to use our software is navigate to www.stod.app in a browser of their choice. Once there, they can create a new account with a username and (secure) password and will be redirected to the homepage where they can view posts or subscribe to various groups in order to create their own posts.

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

Once the program has been set up and is running on a server, it is in the normal operating mode. At this point, users will be able to access the website with all the normal functionality. There is no additional setup or upkeep required for normal operation. No work needs to be done in the back end to keep up with normal operations and the user will not have to do anything should any updates occur.

3.2 Operating Mode 2: Abnormal Operations

If one or more functionalities fail to operate normally (groups, posts, commenting, tags, direct messaging, etc), then the program is considered to be in abnormal operation. To fix this, the back end containers may be taken down by entering the back end Ubuntu server, running *docker ps -q* then within each docker directory running *docker-compose down -v* in order to reset the databases for both the chat server and the normal database. This should only be run in very dire situations as all data will be lost permanently.

3.3 Safety Issues

The issue of maintaining anonymity and creating a safe space for users is inherent in our project. It is our main motive to provide a platform in which users are able to express themselves freely without criticism or hate. As a result, we have designed around this necessity and have therefore implemented measures to help provide this safe space. Because these measures are built into our platform, no additional action is required in regards to this safety issue of maintaining a safe space.

4 Technical Background

Our overall system architecture consists of two main components, the frontend (client side) and the backend (server side). Fundamentally, our client side communicates with the server by making AJAX requests and requesting data, which is then received by our client side in the form of a JSON object. However, there are many libraries we use that abstract this process and allow for quicker development. Our React project uses TypeScript which is a superset of JavaScript and provides type safety for an otherwise weakly typed language. The main advantage of this is that it provides for much easier debugging for large projects. For facilitating the data flow on our client side, we are using Redux. Traditionally data from a React component must be transferred down the component tree to be accessed by another component. However, Redux provides a centralized "store" where all components can store and access data. Redux also provides "actions" which can be dispatched before storing data in the store. For our application, an action can be asynchronous, which means we are making a request to our backend for some data, and after it is received, it is stored in the store.

Additionally to this, we have adopted some ideas of a microservices architecture in that we have a separate service that runs the chat service for our application that handles other user data. In this way we are able to enforce separation of concerns between the chat service and the other services our application provides, and any bugs or problems stemming from one service will have no effect on the other. We deploy our services using Docker containers so that the environments are easily deployable and replicable.

Our back end server consist of an Ubuntu server running the two docker containers, one for the chat server utilizing Node.JS and MongoDB, communicating using sockets, and one for the web server utilizing Django API and PostgreSQL to handle all other user data outside the chat service. Both of these containers are within a network that communicate with outside requests using an Nginx docker image. To ensure secure communication from the backend to the user's front end, HTTPS is used throughout the pipeline of requests.

Our Django server acts as a standalone REST API which can be accessed by calling its endpoints. It consists of smaller mini "apps" where each is a main feature of our whole application. Each of these apps contains models which define our database design through the Django ORM. The Django ORM is an abstraction layer for SQL and allows developers to create database tables in the form of Python classes without having to write any SQL queries. The endpoints of our server are constructed by mapping a specific url to a "view" which handles the data logic. On the first entry into a view it checks which type of request it makes any database queries accordingly. Before storing or retrieving the data, it goes through a serializer or deserializer which converts the data from or into JSON.

The chat server runs on a single Nodejs process inside a Docker container. It communicates with websockets to receive and send message events. Additionally, it connects to a MongoDB database to store messages for users. MongoDB is a nosql database which allows us to store the data as JSON objects making it easy to extract and send the data as is. Compared to SQL databases, it has much faster access times, which is ideal for real time chat communication.

5 Relevant Engineering Standards

The following are a list of the most relevant engineering standards implemented in our project:

RESTful APIs: This is a software architectural standard which uses HTTP requests to access and manage data. The data can generally be created, read, updated, and deleted (CRUD). This is used within our project to allow for the front end to communicate with and access data from our backend.

Websockets: A connection-based communication protocol which allows for communication between two points. Websockets are able to work in the browser and have been implemented in our project for direct messaging. Two users are able to communicate with each other by using websockets which relays messages between them.

Containers: We specifically used Docker containers to spin up the separate sections of our project. Containers isolate sections of our project by packaging related code and its dependencies. These containers can then be weaved together seamlessly to work in tandem. Furthermore, additional features to the project can be added by spinning up a new container and connecting it to our current containers, allowing for ease of scalability.

Microservices: A software architecture which builds applications that are independently deployable, highly maintainable, and scalable. An application built using this architecture is distributed into separate components which will communicate and interact with each other. One way of implementing the microservice architecture is to use containers; we have used Docker containers for our project.

Cookies: Small pieces of data which are stored on the user's browser. Cookies are used within our project by keeping track of user's login preferences, login credentials, which account they are logged in with, ect.

6 Cost Breakdown

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
1	1	Frontend Domain (stod.app)	$14	$14
2	1	Backend Domain (stodbackend.app)	$10	$10
3	1	Digital Ocean Ubuntu Server	$96	$96
			Beta Version-Total Cost	$120

We have minimal costs for our project, with most of the cost going to the server hosting our backend servers and databases. The front end requires no charge, but domains were purchased for both the front end and back end, charged on a yearly basis.

7 Appendices.

7.1 Appendix A - Specifications

Specification	**Details**
Public domain to access front end	www.stod.app
Backend server	8gb/160gb disk/Ubuntu 18.04
Docker	React.js
Groups	Able to allow users to join groups
Posts	Able to allow users to create and view posts
Commenting	Able to allow users to comment on posts and reply to other comments
Messaging	Able to allow users to directly communicate with each other through direct messaging

7.2 Appendix B – Team Information

Camden Kronhaus
Camden Kronhaus is from Chapel Hill, North Carolina. He will be graduating with a degree in Computer Engineering. Once he graduates he will be attending BU Graduate School for a Master’s in Electrical and Computer Engineering.

Quinn Meurer
Quinn is a computer engineering senior from Newton, Massachusetts.

Sharith Godamanna
Sharith Godamanna is a Computer Engineering senior from Colombo, Sri Lanka. Once he graduates from BU’s college of Engineering, he will be working in the field of Web services and connected systems.

Alexander Trinh
Alex is a Computer Engineering senior from Marlborough, Massachusetts. After graduation, he will be working in the field of natural language processing at Amazon.