

Haochen Zhou (hz2204)
Jia Shi (js11182)

Final Project Report – Gululu

Introduction

Gululu is an online social networking system for residents living in different places in the United States. People are allowed to register an account and then login or logout. Besides, users also can join in blocks, add friends, add neighbors to build relationships with others who near them. What's more, users are allowed to send or reply messages with their friends, neighbors, members in the same block or hood. And they can receive updated notifications if they have unread messages or other people want to add them as friends or neighbors or want to join in their block. The following content will tell more about the design and implementation of the system and how a user should use the system.

Environment and Language Used

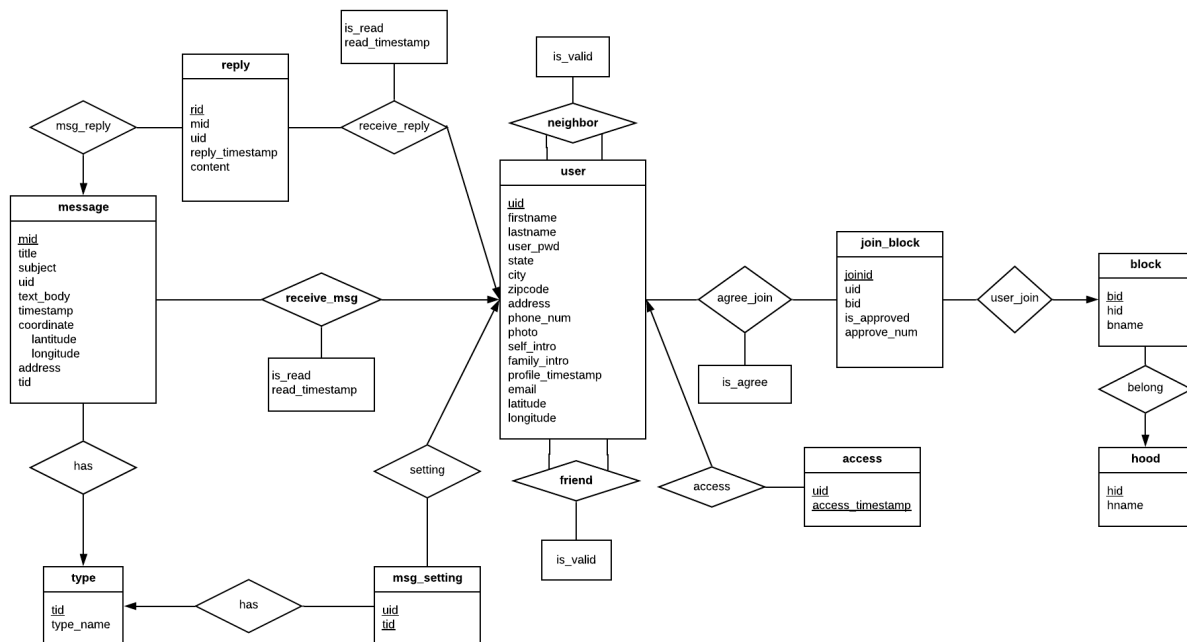
Environment: LAMP (Linux + Apache + MySQL + PHP)

Languages: PHP, JavaScript (jQuery Structure), HTML, CSS

Design

Database Design

ER Diagram:



Improve the design of Project 1:

- Make the email in the user table unique.

Because users need to use their emails to log in to the system rather than their names. We must make sure that every record in the user table has a unique email.

- Add latitude and longitude in the user table

We add latitude and longitude in the user table to match a user's address. Because we need the user's latitude and longitude to draw the markers of user address on the map. If we translate the address to latitude and longitude using API every time, it will be very time-consuming. So we save it in the user table. In this way, every time we draw the marker, we just need to query the location information from the user table, which will save a lot of page rendering time. It will improve the user experience.

```
mysql> desc user;
```

Field	Type	Null	Key	Default	Extra
uid	int(11)	NO	PRI	NULL	auto_increment
firstname	varchar(45)	NO		NULL	
lastname	varchar(45)	NO		NULL	
user_pwd	char(32)	NO		NULL	
state	varchar(15)	NO		NULL	
city	varchar(45)	NO		NULL	
zipcode	int(5)	NO		NULL	
address	text	NO		NULL	
phone_num	bigint(20)	NO		NULL	
photo	text	YES		NULL	
self_intro	text	YES		NULL	
family_intro	text	YES		NULL	
profile_timestamp	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP
email	varchar(50)	NO	UNI	NULL	
latitude	decimal(10,8)	YES		NULL	
longitude	decimal(10,8)	YES		NULL	

16 rows in set (0.01 sec)

- Add address in message table.

Because we need to search the message by their location address. It is time consuming to map the latitude and longitude with address using map api. So we save the mapped address data in the message table. Every time we search the location, we just query the data in the message, which will save the page rendering time and improve the user experience.

```
mysql> desc message;
```

Field	Type	Null	Key	Default	Extra
mid	int(11)	NO	PRI	NULL	auto_increment
title	varchar(200)	NO		NULL	
subject	varchar(200)	NO		NULL	
uid	int(11)	NO	MUL	NULL	
text_body	text	NO		NULL	
timestamp	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP
latitude	decimal(10,8)	YES		NULL	
longitude	decimal(11,8)	YES		NULL	
tid	int(11)	NO	MUL	NULL	
address	text	YES		NULL	

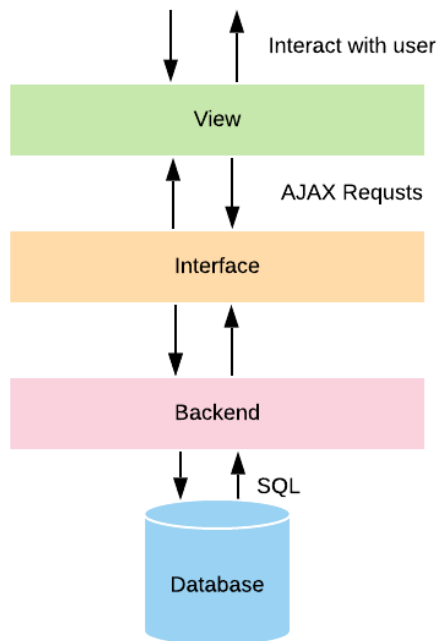
10 rows in set (0.00 sec)

- Delete the access table.

We have the is_read attribute in the receive_msg table, so we do not need the access table to record the access time of each user.

Project Structure Design

We separated the backend and front end. They use HTTP requests to interact with each other. The front end is responsible for displaying stylings and user interaction. The backend is responsible for providing data and updating data. It query data from database or save data to database. Their relation diagram is as follows.



Features Design

- Register
 - Add one record in user.
 - Use md5 to hash the password in case of clear text password.
 - Check if the email already exists using php to fetch records from database.
 - Use JavaScript and PHP to check if the required input items is null.
- Log in
 - check if the email and the password is matched.
 - Create a session for this user. Add cookies to remember the login status of this user.
- Log out
 - Destroy session and cookie.
- Feed
 - Query the messages and replies by friend, neighbor, block and hood.
- Send Message

- Add message record into message table, and add the tuples recording who can read and reply this message into receive_msg table.
- **Add Message Location [Extra feature]:** Use HTML5 getCurrentPosition() function of navigator to get the latitude and longitude of the current position. Then use Google Map API to translate the location information into address.
- Reply Message
 - Add a reply to the reply table and add records in receive_reply to save the users who can get access to this reply.
- Add Friend
 - Add two records to the friend table, which contains uid and friend_uid. Only when the is_valid in these two records are both 1, they will become friends.
- Add Neighbor
 - Add one record into the neighbor table. Only one is_valid means that the neighbor relation is valid.
- Add Block
 - Add record into join_block, only when the is_approved is 1, the user is added to this block, which means there are more than 3 people agree this user to join the block.
- Show Map
 - **Current Location [Extra feature]:** Use HTML5 getCurrentPosition() function of navigator to get the latitude and longitude of the current position. Then use Google Map API to add a marker to this location.
 - Message marker and neighbor marker: Use the location data saved in the database of the message and the user to draw marker on the map using Google Map API.
 - **Click message marker to show the message, click neighbor marker to show his or her first name [Extra feature]:** Use the Google Map API to add listener of click event then show the title or first name queried from database.
- Show profile
 - Show Profile: Query self_intro and family_intro data from the user table.
 - Edit Profile: Update self_intro and family_intro data in the user table.
 - **Upload Avatar [Extra feature]:** Use formdata to send a post request to send the avatar image to the backend. When backend gets the image, it check if the image is valid, then save it to a path of the server.
- Apply Friends or block Notifications
 - Query data from database to show the pending applications for friend and joining block.
 - **Neighbor recommendation [Extra feature]:** Query those user who was add to neighbor list by other people but he or her did not add the user who add the to his or her neighbor list.
- Unread Messages Notifications
 - Query the message that the user did not read until the last time they get access to the feed. We don not use the access time. We use the is_read attribute in receive_msg table to check the unread messages.

API Design

All the APIs extend the api.php, which executes functions like database connection, check parameters, check if the user has logged in. The design of some major APIs are listed below

Friend Feed:

- **method:** GET
- **parameter**

- isUnRead: int
- uid: int

- **Sample Data**

```
{
  "status": 0,
  "data": [
    {
      "firstname": "Jia",
      "lastname": "Shi",
      "photo": "jia.jpg",
      "mid": "13",
      "title": "hello Choing",
      "subject": "hello ",
      "uid": "1",
      "text_body": "Are you free tomorrow?",
      "timestamp": "2019-12-18 10:47:59",
      "lantitude": "0.00000000",
      "longitude": "0.00000000",
      "tid": "1",
      "address": "undefined",
      "reply": [

    ]
  },
  ...
],
  "message": "Success"
}
```

Get Notification:

- **method:** GET
- **parameter**
 - uid: int, COOKIE

- **Sample Data**

```
{
  "status": 0,
  "data": {
    "totalNum": 1,
    "notifications": {
      "friendNoti": [
        {
          "uid": "5",
          "firstname": "Lisa",
          "lastname": "Zhou",
          "photo": null
        }
      ],
      "neighborNoti": [
```

```
    ],  
    "joinBlockNoti": [  
    ]  
  },  
  "message": "Success"  
}
```

Sign In:

- **method:** POST
- **parameter**
 - email: string
 - userPwd: string
- **Sample Data**

```
{  
  "status": 0,  
  "data": [  
  ],  
  "message": "Success"  
}
```

All other APIs:

```

getTypes: 'api/getTypes',
sendMsgToOne: 'api/sendMsgToOnePerson',
sendMsgToAll: 'api/sendMsgToAll',
getFriendFeed: 'api/getFriendFeed',
getBlockFeed: 'api/getBlockFeed',
getHoodFeed: 'api/getHoodFeed',
getNeighborFeed: 'api/getNeighborFeed',
sendReply: 'api/sendReply',
signOut: 'api/signOut',
listFriends: 'api/getFriendInfo',
addFriend: 'api/addFriend',
availFriend: 'api/availableFriend',
listNeighbor: 'api/getNeighborInfo',
addOrCancelNeighbor: 'api/addNeighbor',
acceptOrCancelFriend: 'api/acceptFriend',
availNeighbor: 'api/avaNeighbor',
listblock: 'api/getBlockInfo',
showblock: 'api/checkInBlock',
joinBlock: 'api/joinBlock',
leaveBlock: 'api/leaveBlock',
getProfile: 'api/getProfile',
parseLoc: 'api/parseLocToAddr',
getUserInBlock: 'api/getUserInBlock',
search: 'api/searchKeywords',
getNoti: 'api/getNotification',
accOrDenyJoinBlock: 'api/updateBlock',
updateProfile: 'api/updateProfile',
unreadNum: 'api/numMessage',
updateMsgToRead: 'api/msgToRead',
uploadAvatar: 'api/uploadAvatar',
getUserInfo: 'api/getUserInfo',

```

Security and Robustness Design

Security:

- SQL Injection
 - When the parameter is String type, we use `mysqli_real_escape_string($this->conn, $param)` in case of SQL Injection.
 - When the parameter is Integer type, we use `intval()` function to parse params into Integer.
- XSS
 - We do not add script from input or parameters into our HTML. So our system can prevent XSS attack.
 - When add element into HTML, we use template engine to parse HTML, which has the mechanism to prevent XSS attack.
- CSRF
 - We use check if the session and cookie match to prevent CSRF attack, which means we never believe the user, we only believe the session in our server.

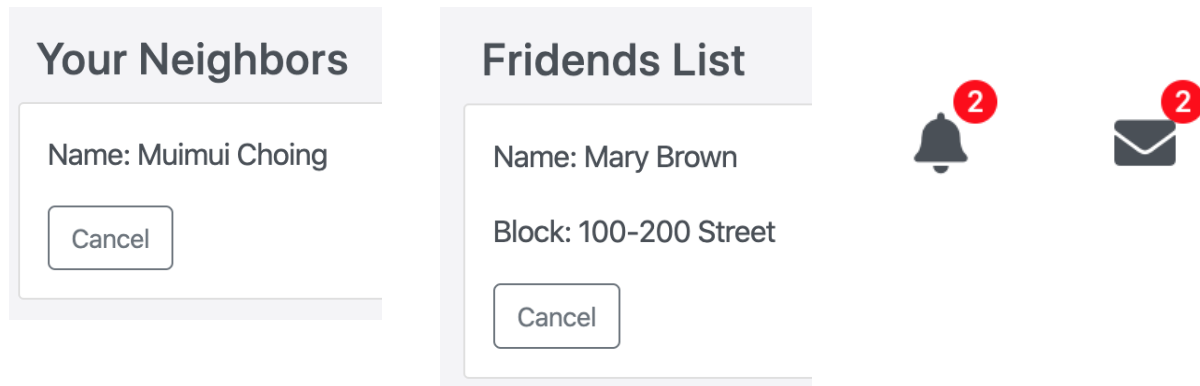
Robustness:

- Transaction
 - We use transaction when there are atomicity updates or inserts. When there is one update or insert statement fails in this transaction, all the sql statements in this transaction will be roll back.
- Customized check for every API
 - We add not null check and customized check for each api in case of dirty data.

UI Design

Consistency

- If users are able to execute different actions in the same way, they will not have to spend much time in studying all different procedures when using the system.
- For example, users can cancel either a neighbor relationship or friendship by clicking “Cancel” button. Besides, the number in the red circle means that there are some unread notification or message. After saw it, users can know they are able to click it to check update information. It is convenient for users to find all tools and users do not need to learn many new things when using the system.



Offer Informative Feedback

- Telling users their actions are succeed or not and presenting the current state on the interface to users.
- When a user adds Lisa as his friend, he will receive a respond “Waiting” for Lisa’s agreement. Besides, when a user joined in a block, the interface will show a message telling which block the user is in now. After receiving a response from the interface, users are able to decide and have confident about what they should do next.

Name: Lisa Zhou
Block: 10-45 Avenue

Waiting...

You are now in 100-200 Street Block

Leave Block

Reduce short-term memory load

- Avoiding users remembering many unnecessary things or information. Making options and buttons are clearly visible for users and allowing them to choose from a list.
- When a user wants to add a friend, he does not need remember all members in his hood. He can just check all users in the recommendation list to find a person he wants to add. Join block and add neighbors are similar. The interface reduces users memory load and enables them to take actions more easily and conveniently.

Recommendations

Name: Jack Lee

Block: 10-45 Avenue

Add

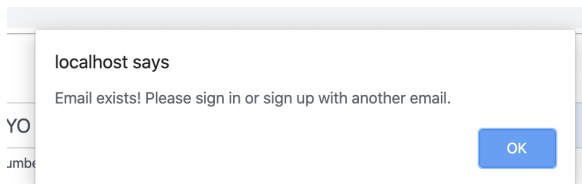
Name: Demi Hutchings

Block: 10-45 Avenue

Add

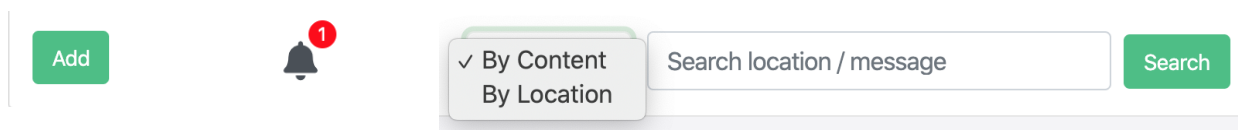
Prevent errors & documentation and help

- Avoiding users making errors and providing tips to users and telling them what is going on now.
- Reject empty input and all illegal actions. If a user inputs an invalid value, an error window will pop-up giving the user an error message and helping the user find out what cause the error and may also learn how to avoid it later.



Visibility

- Using different buttons allows users to see options clearly and learn how to use the system quickly.
- For example, list options (“by content” or “by location”) can help users understand what these options are for what functions. The “Add” button helps users quickly know they can click it to add friends or neighbors.



Features

Register

Users are allowed to fill out their personal information to create an account.

1. Input personal information
2. Click “Register Now” to create a new account
3. Click “Sign in” to get to the login page

Gululu – Sign Up

[Sign in](#)

Email

First name

Last name

State

City

e.g. New York

Address

ZIP code

e.g. Street or Apt. Number

Phone

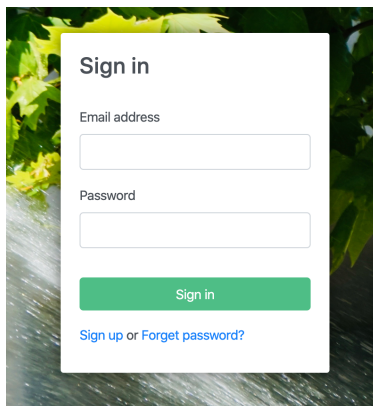
Password

Register Now

Sign In

Users are allowed to sign in with their account.

1. Input their email address and password
2. Click “Sign In”
3. If you are a new user, can click “Sign Up” to create a new account



Sign in

Email address

Password

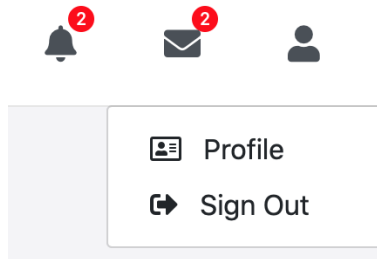
Sign in

[Sign up](#) or [Forget password?](#)

Sign Out

Sign out of the user's account

1. Click the icon (person) on the top right of the page
2. Click “Sign Out”



Edit Profile

Users are allowed to update self introduction and family introduction on the profile page

1. Click the icon (person) on the top right of the page and choose “Profile” option
2. Click “Edit” and then can input information you want to edit
3. Submit

About me [\(Edit\)](#)

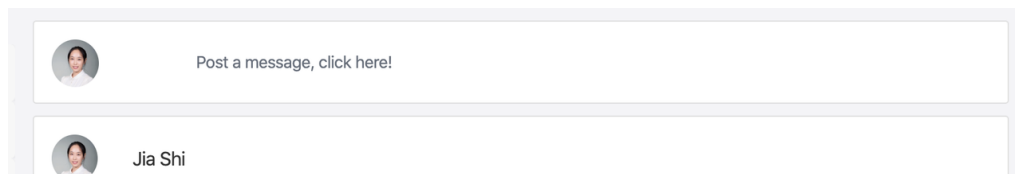
Hello, my name is Mary Brown

Submit

Send Message

Users are allowed to send messages to their friends/ neighbors/ entire block/ entire hood

1. Click the top rectangle box to post a message
2. Input Title, Subject and message and Choosing who you want to send for
3. Click “Add your location” to get your real-time location
4. Click “POST” to post the message



Edit Message ✕

Title

Subject

Type

✓ Choose to send to friends, neighbors or block
friend
neighbor
all friends
entire block
entire hood

Message

📍 Add your location

📷 Add a photo

Reply Message

Users are allowed to reply other user's message

1. Click the “message body” to input any reply message
2. Click “Reply” to post the message

Haochen Zhou

Buy a dog

Hello, I want to buy a dog. Do you know someone who wanna sell one?

2019-12-17 20:55:42 📍 55 Allen St, New York, NY 10002, USA

🗨️ reply comment 1

Haochen Zhou , 436 Albee Square

hahahah

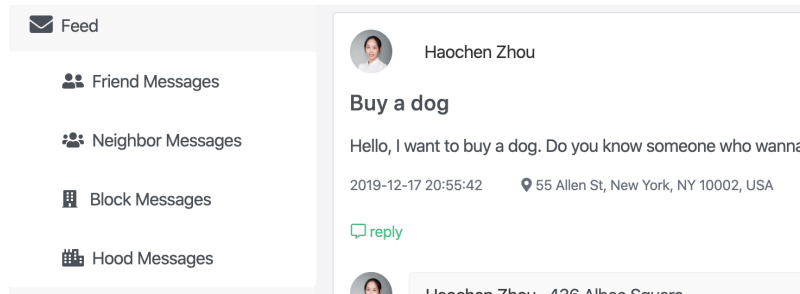
2019-12-18 03:07:26

message body

Feed

Users are allowed to choose to see whose messages

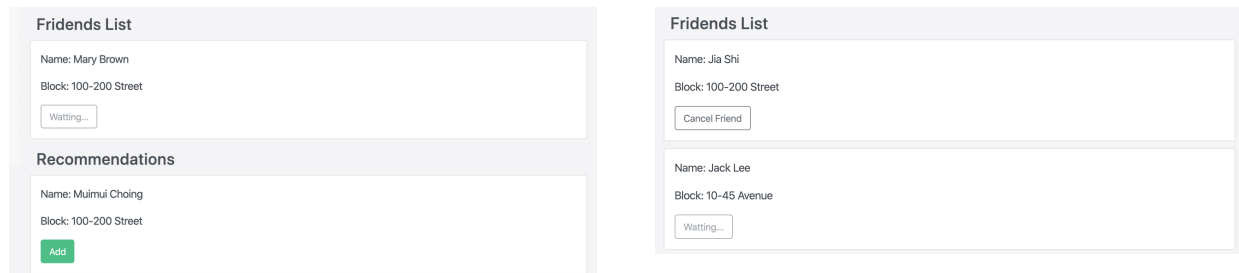
1. Click “Feed” on the left of the page and then choose whose messages you want to see
2. All matching messages will be listed on the right



Add Friend

Allowed users to add friends in the same block

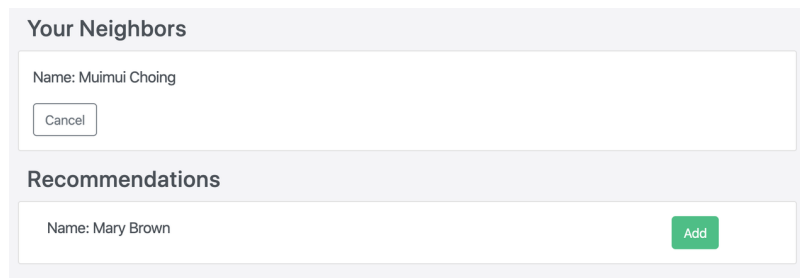
1. Click the “Add Friend” on the right of the page
2. Click “Add” to add friend/ Click “Cancel Friend” to remove the friend from friend list



Add Neighbor

Users are allowed to add/cancel neighbors

1. Click “Add Neighbor”
2. Click “Cancel” to cancel your neighbors
3. Click “Add” to add a neighbor

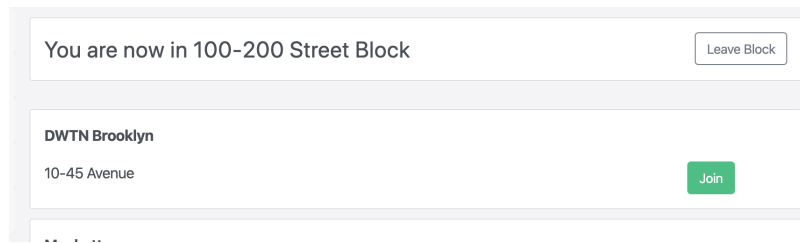


Join Block/Leave Block

Users are allowed to join in a block or leave a block

1. Click “Join Block”
2. Click “Leave Block”

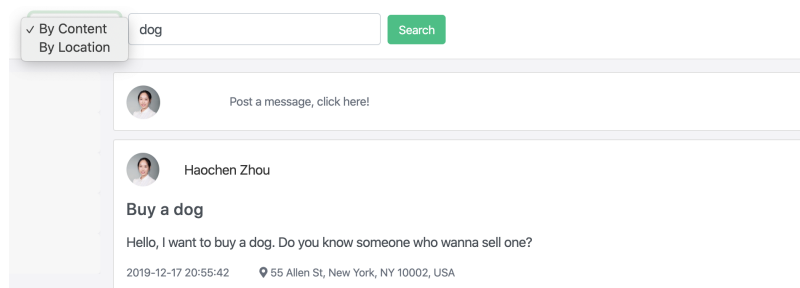
3. Click “Join”



Search keywords/ location

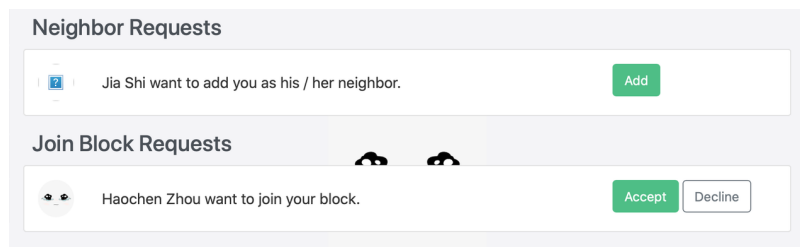
Users are allowed to search messages by content or by location

1. Choose search messages by content or by location
2. Input keywords in the search bar
3. Click “Search” to view all messages related to the keywords



Notification

1. Click the icon (notification) on the top right of the page
2. Users are able to see a list of notifications
3. Choose to add a neighbor/ accept or decline a user being your friend/ accept or decline a user joining in your block



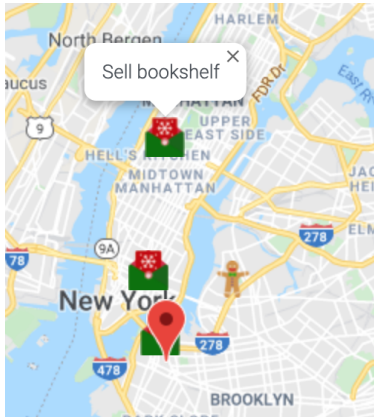
Unread Message

1. Click the icon (message) on the top right of the page if there are unread messages
2. Users are able to see all unread messages

Map

1. Users are able to see other users' message or location on the map

2. Click “Message” icon to check other’s message
3. Click “person” icon to check block member’s home address
4. The position icon represents the user’s location



Upload Photo

1. Click the icon (person) on the top right of the page and choose “Profile” option
2. Click “Edit” option
3. Click the photo and choose an image from user’s local computer

