1. Assumptions:

- When a new user signs up, he has to provide name, email, state, city and zip. These fields are mandatory. The other fields like family, photo etc. can be updated later through an *Edit Profile* option.
- For simplicity, hoods have a unique name and there is no address related to a hood. Hence, irrespective of the user's address, he can select any one of the listed hoods and then a block within that hood.
- At any given point of time, a user can be a member of only 1 block.
- User can become friends or neighbors with anyone in his hood.
- If user moves out of a hood, he will lose all his friends and neighbors and will not be able to view any of the old messages.
- Messages are placed in the order of recent ones first.

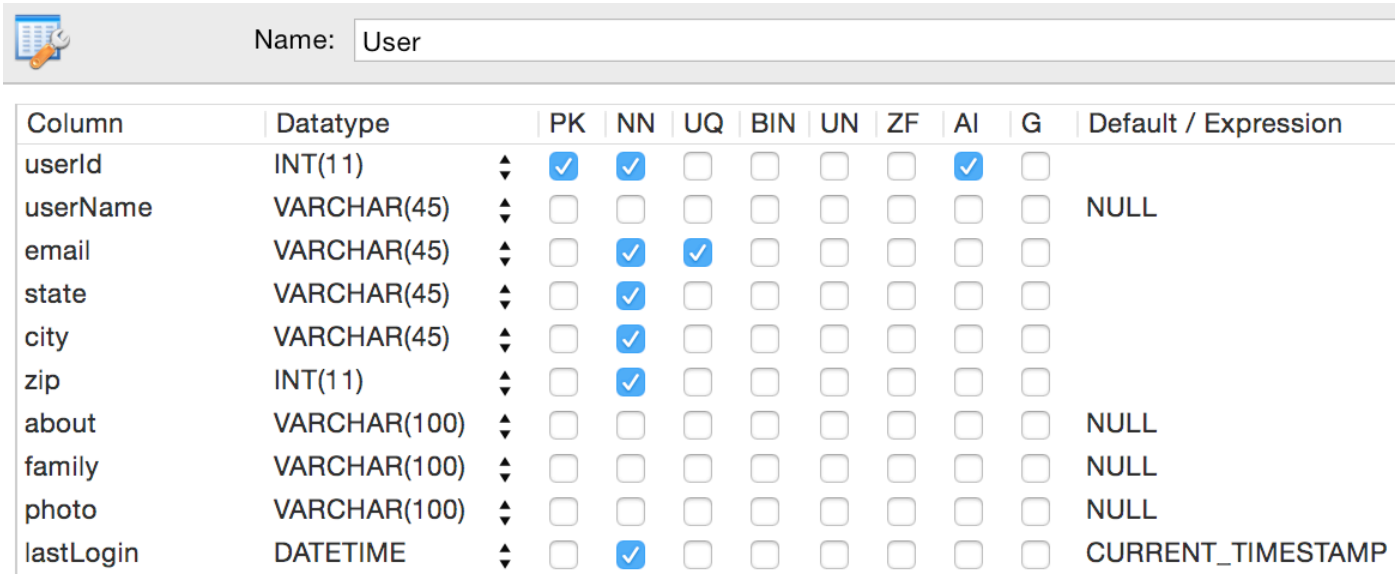
2. Schema:

ser	UserPreference	Hood	HoodMap	Block	BlockMembe	BlockApproval	FriendReques	Relation	Topic	Message
serId	*userId*	**hoodId**	*hoodId*	**blockId**	*blockId*	*fromUserId*	*fromUserId*	*firstUserId*	**topicId**	**messageId**
serName	prefName	hoodName	lat	*hoodId*	*userId*	*toUserId*	*toUserId*	*secondUserID*	subject	*topicId*
mail	prefValue		long	blockNam	count			isFriend	message	message
ate								isNeighbor	*authorUserId*	*authorUserI*
ty									location	created
p									created	
out									modified	
mily									tagType	
hoto									recipient	
stLogin									*parentHoodId*	

Primary keys are indicated in bold and underline.
Foreign keys are indicated in orange and italics.

3. **Schema structure and explanation:**

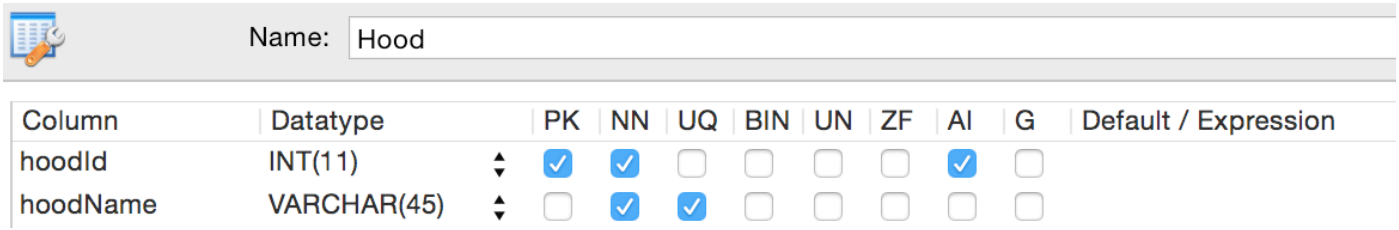
- User: This table holds the information of all users who have signed up on the system. The *email* field is unique. The *lastLogin* is the timestamp when the user last logged in. This will be used to determine which are the new/unread messages to be shown to the user.



Name: User

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
userId	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
userName	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
email	VARCHAR(45)	☐	☑	☑	☐	☐	☐	☐	☐	
state	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
city	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
zip	INT(11)	☐	☑	☐	☐	☐	☐	☐	☐	
about	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
family	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
photo	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
lastLogin	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP

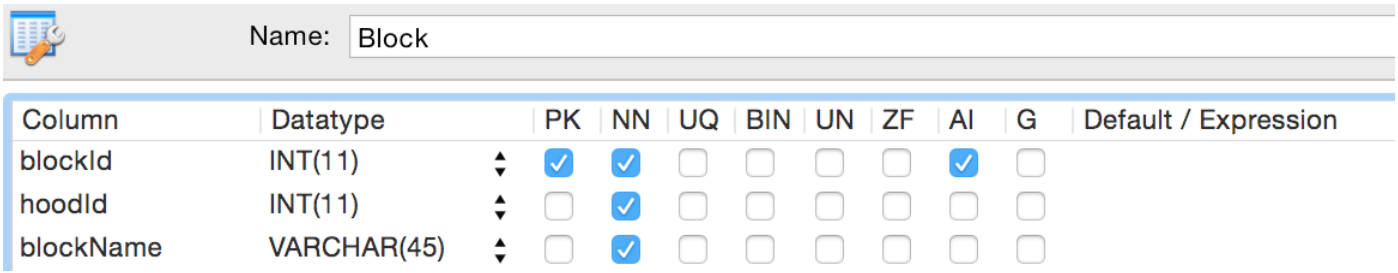
- Hood: This table hold the information of all hoods. For simplicity, we are pre defining a set of hoods that have unique names. These hoods do not have any address fields such as state, city or zip associated with them. Hence a user can select any hood irrespective of his address.



Name: Hood

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
hoodId	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
hoodName	VARCHAR(45)	☐	☑	☑	☐	☐	☐	☐	☐	

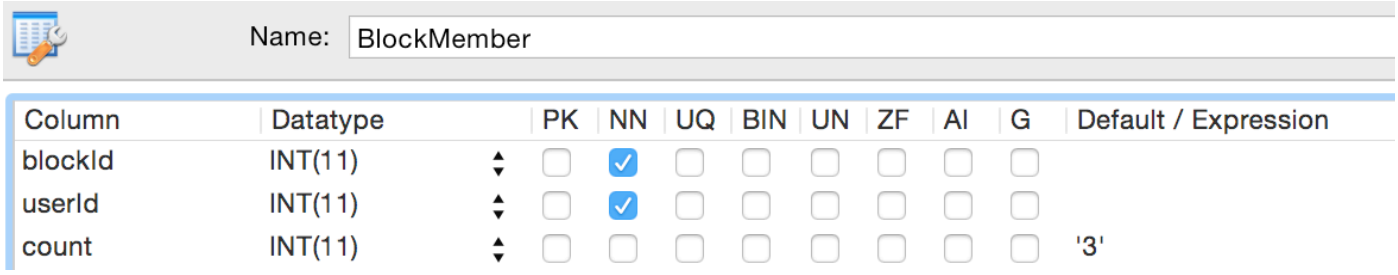
- Block: This table holds the information of all blocks. Each block belongs to one of the hoods defined above. Again, these are pre defined blocks.



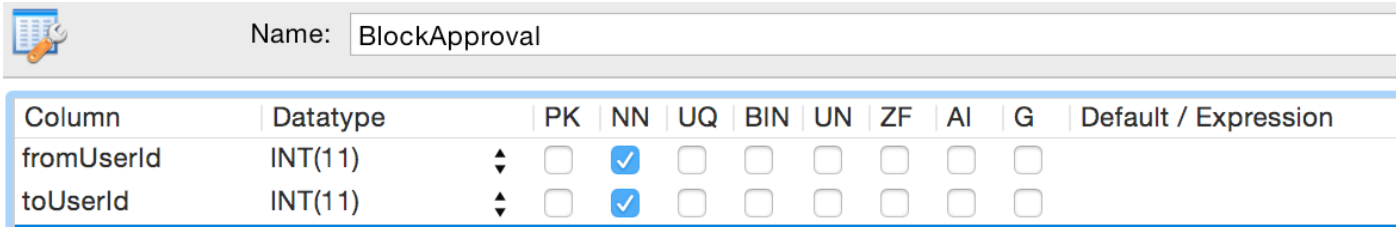
- BlockMember: This table holds the information of both- the approved members of a block and those users who are waiting for approval.
  Whenever a user requests to join a block, he will be added into this table with a *count* value. The *count*value indicates if the user is approved or awaiting approval.
  If *count* is 0: this implies that the user is waiting for 0 approvals, hence he is an approved member.
  If *count* is 1: the user is waiting for 1 other approval, hence he is still not an approved member.
  If *count* is 2: the user is waiting for 2 other approvals, again he is still not an approved member.
  If *count* is 3: the user is waiting for 3 other approvals.

  The *count* value is initially set depending on the number of existing approved members in that block.
  If there are 0 existing approved members, then the new user is the very first member of the block, hence the *count* will be set to 0 for him.
  If there is 1 existing approved member, then the new user needs only 1 approval, hence the *count* will be set to 1.
  If there are 2 existing approved members, then the new user needs only 2 approvals, hence the *count* will be set to 2.
  If there are 3 or more existing approved members, then the new user needs any 3 approvals, hence the *count* will be set to 3.

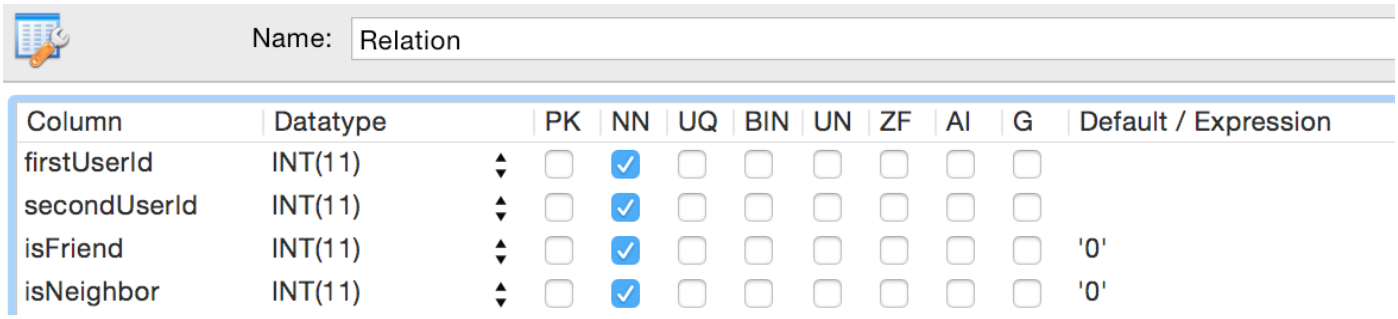
  Each time an existing member approves a new user, the *count* for the new user will be decremented by 1.



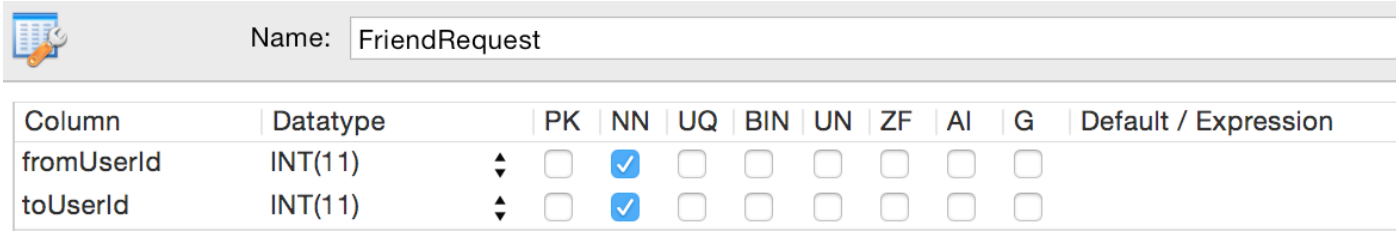
- BlockApproval: This table is used to temporarily track who has approved whom. This is required because, say for example: User B requests to join a block and requires 3 approvals. User A who is an existing member receives the request and he approves it. Now user B needs 2 more approvals. User A has already approved, so he should not receive the request again. Hence this table will help track this scenario. Once User B receives all 3 approvals, then all his entries will be deleted from this table.



- Relation: This table hold the relationship information between different users. The *isFriend* and *isNeighbor* fields indicate the relation. The value 1 indicates true and the value 0 indicates false. When User A adds User B as neighbor, then an insert/update is made on this table. In the case where User A wants to add User B as a friend, then a friend request is sent and the *FriendRequest* table is used to track the relation as explained below.



- FriendRequest: This table is used to temporarily track the friendship status between 2 users. When User A sends a friend request to User B, an entry will be made in this table. If User B accepts the request or sends a friend request to User A, then we use this table to determine that both want to be friends with each other, hence insert/update the *Relation* table. We then delete the entry from this table.



- Topic: This table holds the information about the initial message that was created by a user. A user can make the message accessible to the entire hood or to the entire block or to all his friends or to all his neighbors or to certain selected individuals. The *tagType* indicates the category. The possible values are HOOD, BLOCK, FRIENDS, NEIGHBORS, PERSONAL.

  When the *tagType* is HOOD, the recipient column will store the user's *hoodId*.
  When the *tagType* is BLOCK, the recipient column will hold the user's *blockId*.
  When the *tagType* is FRIENDS, the recipient column will hold comma delimited *userId* of all users who are friends.
  When the *tagType* is NEIGHBORS, the recipient column will hold comma delimited *userId* of all users who are neighbors.
  When the *tagType* is PERSONAL, the recipient column will hold comma delimited *userId* of selected members that the user selected.

  The *created* column stores the timestamp when the topic was created and the *modified* column stores the timestamp when the most recent reply to the topic was posted.

  The *parentHoodId* column stores the user's *hoodId* and it indicates the maximum scope of the topic. This topic cannot be accessed beyond this hood. This is required because, say the user moves into a different hood. Then while retrieving topics for the user, a check is made to see if his current *hoodId* matches the *parentHoodId*. If it does not, then we do not show those topics and its child messages.

*

Name: Topic

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
topicId	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
subject	VARCHAR(100)	☐	☑	☐	☐	☐	☐	☐	☐	
authorUserId	INT(11)	☐	☑	☐	☐	☐	☐	☐	☐	
location	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
created	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
tagType	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
recipient	VARCHAR(200)	☐	☑	☐	☐	☐	☐	☐	☐	
parentHoodId	INT(11)	☐	☑	☐	☐	☐	☐	☐	☐	
modified	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP

- Message: This table holds the information of all messages that were replied to a topic.

Name: Message

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
messageId	INT(11)	☑	☑	☐	☐	☐	☐	☑	☐	
topicId	INT(11)	☐	☑	☐	☐	☐	☐	☐	☐	
authorUserId	INT(11)	☐	☑	☐	☐	☐	☐	☐	☐	
created	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
message	VARCHAR(200)	☐	☑	☐	☐	☐	☐	☐	☐	

- HoodMap: This table holds the latitude and longitude information of all the hoods. This information shall later be used to draw polygons on maps which specify the neighborhoods (which shall be predefined)

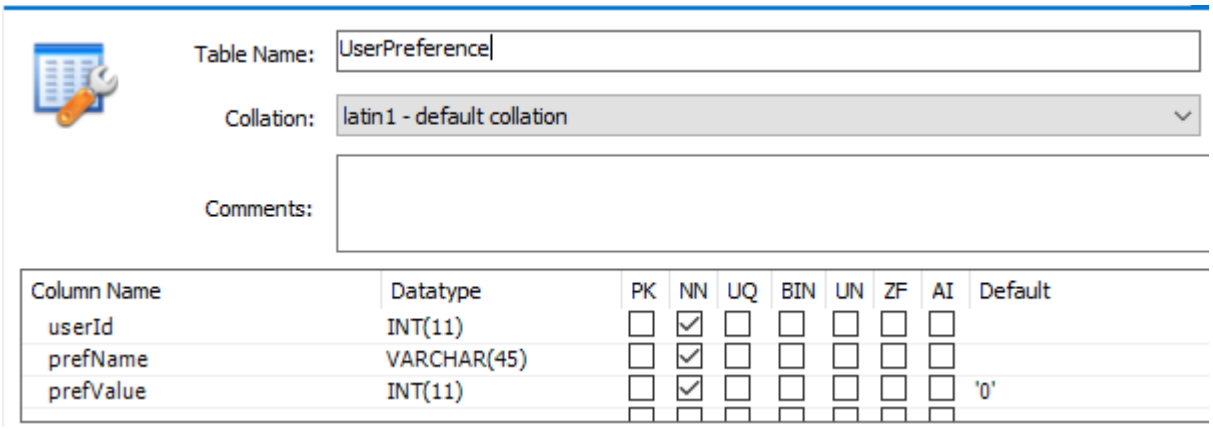
Table Name: HoodMap

Collation: latin1 - default collation

Comments:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
hoodId	INT(11)	☐	☑	☐	☐	☐	☐	☐	
lat	DECIMAL(13,10)	☐	☑	☐	☐	☐	☐	☐	
lng	DECIMAL(13,10)	☐	☑	☐	☐	☐	☐	☐	

- UserPreference: This table holds the details of all the users who have subscribed for email alerts.



4. Table data and user flow:

Block

blockId	hoodId	blockName	lat	lng
40	5	330 W Broadway	40.7221810000	-74.0039780000
42	5	96 Greene St	40.7240680000	-73.9998000000
43	5	157 Prince St	40.7259700000	-74.0012590000
44	5	558 Broadway	40.7236610000	-73.9974820000
45	7	183 Centre St	40.7183430000	-73.9997090000
46	7	126 Baxter St	40.7181080000	-73.9989800000
47	7	152A Mott St	40.7190830000	-73.9964690000
48	7	140 Bowery	40.7192330000	-73.9949730000
49	7	389 Broome St	40.7203770000	-73.9970630000
50	7	179 Grand St	40.7194210000	-73.9979280000
51	8	70 Lafayette St	40.7171390000	-74.0021270000
52	8	65 Bayard St	40.7152190000	-73.9982010000
53	8	46 Eldridge St	40.7158050000	-73.9927930000
54	5	330 W Broadway	40.7221810000	-74.0039780000
56	5	96 Greene St	40.7240680000	-73.9998000000
57	5	157 Prince St	40.7259700000	-74.0012590000
58	5	558 Broadway	40.7236610000	-73.9974820000
59	7	183 Centre St	40.7183430000	-73.9997090000
60	7	126 Baxter St	40.7181080000	-73.9989800000
84	6	48 W 12th St	40.7282660000	-74.0030270000
85	4	90 Beekman St	40.7130520000	-74.0042700000
89	4	66 Frankfort St	40.7130520000	-74.0042700000
90	6	6 Grove Ct	40.7282660000	-74.0030270000
91	4	10 Hanover Squ...	40.7130520000	-74.0042700000

Hood

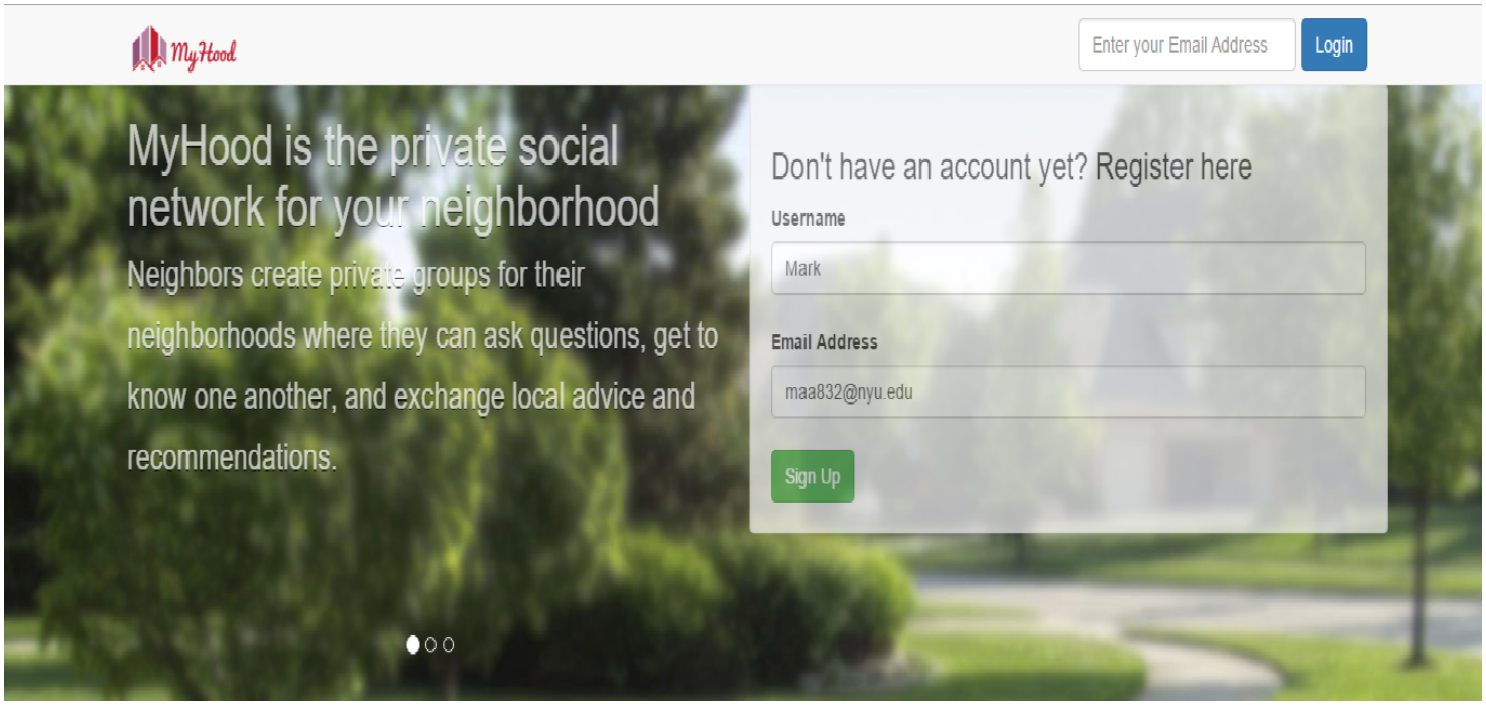
hoodId	hoodName
8	Chinatown
4	Financial District
6	Greenwich Village
7	Little Italy
5	Soho
NULL	NULL

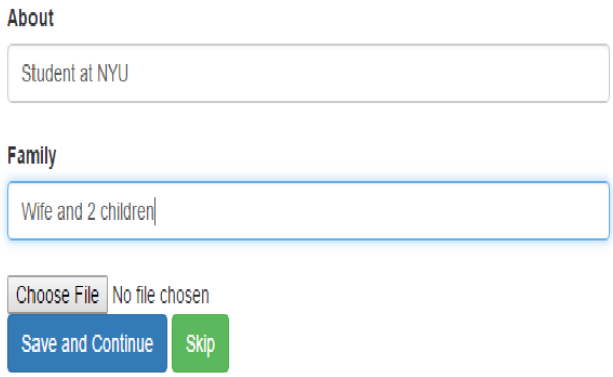
User

	userId	userName	email	state	city	zip	about	family	photo	lastLogin
▶	21	Adam Gabriel	adamgabriel@gmail.com	NY	Manhattan	10004	Working at Chase bank	Unmarried		2015-12-17 04:24:21
	22	Julie S Wee	julieswee@gmail.com	NY	Manhattan	10004	Student at CUNY	Staying with mom		2015-12-17 04:24:21
	23	Lori Greene	lorigreene@gmail.com	NY	Manhattan	10004	Studying at Stern Business school	3 roommates		2015-12-17 04:24:21
	24	Renata Jacobs	renetajacobs@gmail.com	NY	Manhattan	10004	Teacher at St. Agnes			2015-12-17 04:24:21
	25	Ophir Marom	ophirmarom@gmail.com	NY	Manhattan	10004				2015-12-17 04:24:22
	26	Dedrick Bullard	dedrickbullard@gmail.com	NY	Manhattan	10004		Family of 4.		2015-12-17 04:24:22
	27	Carla Davis	carladavis@gmail.com	NY	Manhattan	10004		Eldery couple		2015-12-17 04:24:22
	28	Tristan A Daley	tristanadadley@gmail.com	NY	Manhattan	10004	Business analyst			2015-12-17 04:24:22
	29	Kun Jiang	kunjiang@gmail.com	NY	Manhattan	10004	Soccer coach, train kids under 12.			2015-12-17 04:24:22
	30	Neha Sharma	nehasharma@gmail.com	NY	Manhattan	10004	International student at Stony Brooke			2015-12-17 04:24:23
	31	Waqid Munawar	waqid7@gmail.com	NY	Manhattan	10004	Student at NYU Engineering	Live with uncle		2015-12-17 04:24:23
	32	Ugur Iniac	uguriniac@gmail.com	NY	Manhattan	10004				2015-12-17 04:24:23
	33	Mohammed Ai...	ali.aizu94@gmail.com	NY	Manhattan	10004	Student			2015-12-17 04:24:23
	70	newuser123	waqidvolli@gmail.com			0	newuser	newuser	NULL	2015-12-18 21:47:41
	72	seconduser	seconduser@gmail.com			0	su	su	NULL	2015-12-18 21:49:22
	74	thirduser	wmv214@nyu.edu			0	third	third	NULL	2015-12-18 22:16:44

Step1: User Signup

When the user clicks on Save and Continue on the second page then the user gets registered and all his details are stored in the User table of the database. For example, consider username: Mark, email: maa832@nyu.edu, about: Student at NYU, family: wife and 2 children





userId	userName	email	state	city	zip	about	family	photo	lastLogin
21	Adam Gabriel	adamgabriel@gmail.com	NY	Manhattan	10004	Working at Chase bank	Unmarried		2015-12-17 04:24:21
22	Julie S Wee	julieswee@gmail.com	NY	Manhattan	10004	Student at CUNY	Staying with mom		2015-12-17 04:24:21
23	Lori Greene	lorigreene@gmail.com	NY	Manhattan	10004	Studying at Stern Business school	3 roommates		2015-12-17 04:24:21
24	Renata Jacobs	renetajacobs@gmail.com	NY	Manhattan	10004	Teacher at St. Agnes			2015-12-17 04:24:21
25	Ophir Marom	ophirmarom@gmail.com	NY	Manhattan	10004				2015-12-17 04:24:22
26	Dedrick Bullard	dedrickbullard@gmail.com	NY	Manhattan	10004		Family of 4.		2015-12-17 04:24:22
27	Carla Davis	carladavis@gmail.com	NY	Manhattan	10004		Eldery couple		2015-12-17 04:24:22
28	Tristan A Daley	tristanadadley@gmail.com	NY	Manhattan	10004	Business analyst			2015-12-17 04:24:22
29	Kun Jiang	kunjiang@gmail.com	NY	Manhattan	10004	Soccer coach, train kids under 12.			2015-12-17 04:24:22
30	Neha Sharma	nehasharma@gmail.com	NY	Manhattan	10004	International student at Stony Brooke			2015-12-17 04:24:23
31	Waqid Munawar	waqid7@gmail.com	NY	Manhattan	10004	Student at NYU Engineering	Live with uncle		2015-12-17 04:24:23
32	Ugur Iniac	uguriniac@gmail.com	NY	Manhattan	10004				2015-12-17 04:24:23
33	Mohammed Ai...	ali.aizu94@gmail.com	NY	Manhattan	10004	Student			2015-12-17 04:24:23
70	newuser123	waqidvolli@gmail.com			0	newuser	newuser	NULL	2015-12-18 21:47:41
72	seconduser	seconduser@gmail.com			0	su	su	NULL	2015-12-18 21:49:22
74	thirduser	wmv214@nyu.edu			0	third	third	NULL	2015-12-18 22:16:44
76	Mark	maa832@nyu.edu			0	Student at NYU	Wife and 2 children	NULL	2015-12-24 05:45:43
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Step2: Join Block

We have a predefined set of hoods and a user can choose the hood in which he wants to join. Once a user selects a hood on the map he can join any one block belonging to that hood. For example, Mark lives at 558 Broadway New York, NY. Corresponding blockId is 58 and hoodId is 5. An entry is made into the BlockMember table with count as 3, indicating the numberof approvals it needsto become a member of that block.

MyHood

Step 2: Join a Neighborhood !

Recently joined blocks

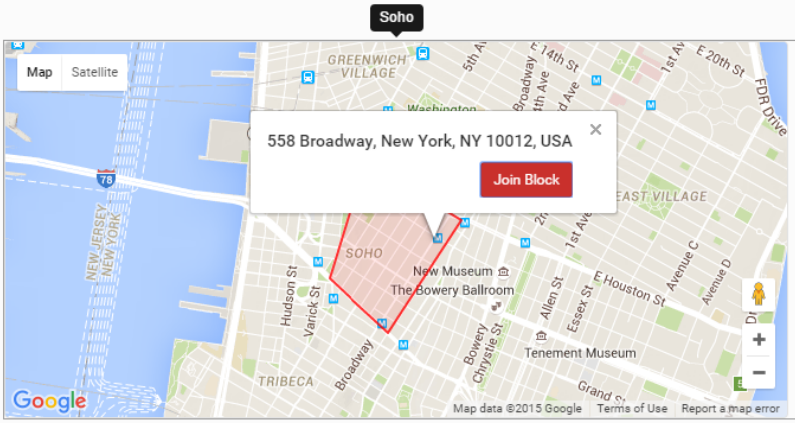
100 Gold St Financial District
71 W 12th St Greenwich Village
68 Jane St Greenwich Village
330 W Broadway Soho
157 Prince St Soho
96 Greene St Soho
558 Broadway Soho
96 Greene St Soho

Reset

MyHood

Sent request to join block !

Step 2: Join a Neighborhood !



Recently joined blocks

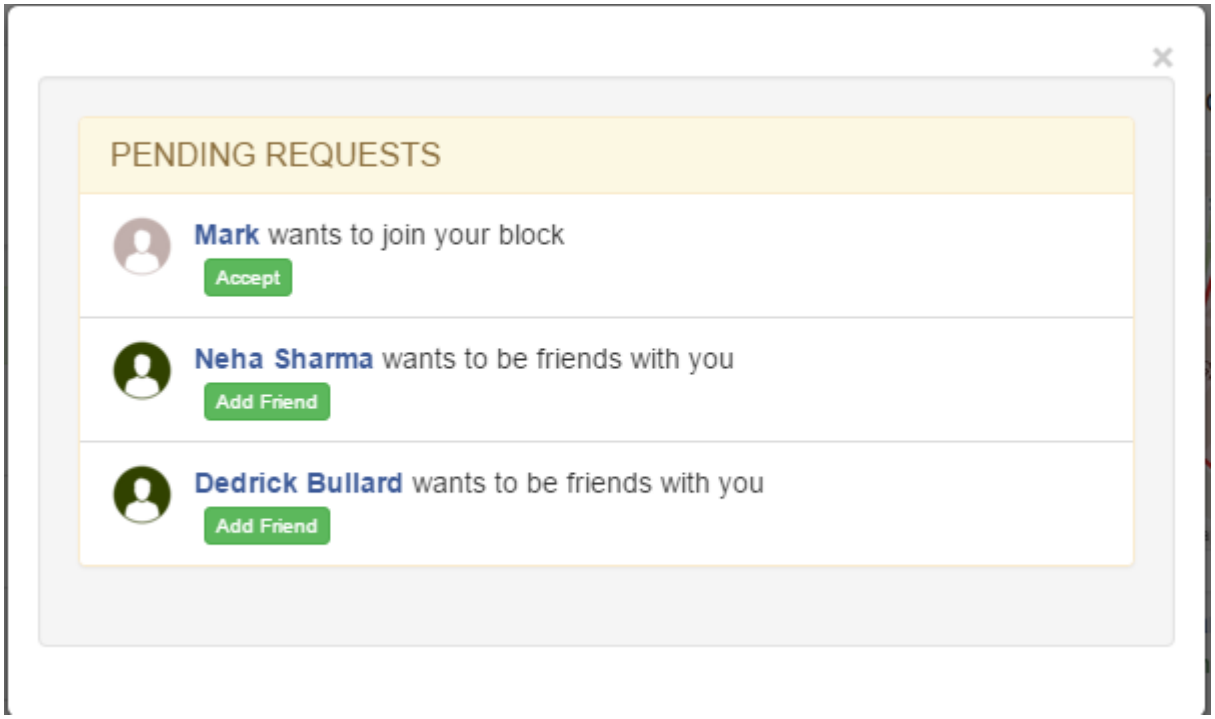
70 Lafayette St Chinatown
100 Gold St Financial District
71 W 12th St Greenwich Village
68 Jane St Greenwich Village
330 W Broadway Soho
558 Broadway Soho
96 Greene St Soho
96 Greene St Soho
157 Prince St Soho

Reset

blockId	userId	count	timestamp
40	21	0	2015-12-17 04:31:36
42	23	1	2015-12-17 04:31:36
56	24	0	2015-12-17 04:31:36
57	25	1	2015-12-17 04:31:36
58	22	1	2015-12-17 04:31:36
58	27	2	2015-12-17 04:31:36
58	28	0	2015-12-17 04:31:36
58	29	2	2015-12-17 04:31:36
58	30	0	2015-12-17 04:31:36
58	31	0	2015-12-18 19:54:39
58	32	2	2015-12-17 04:31:36
58	33	1	2015-12-17 04:31:36
58	76	3	2015-12-24 06:19:11
97	26	0	2015-12-18 19:18:08
106	70	0	2015-12-18 21:48:02
106	74	0	2015-12-18 22:17:49
108	72	0	2015-12-18 21:56:41

Step3: Notifications

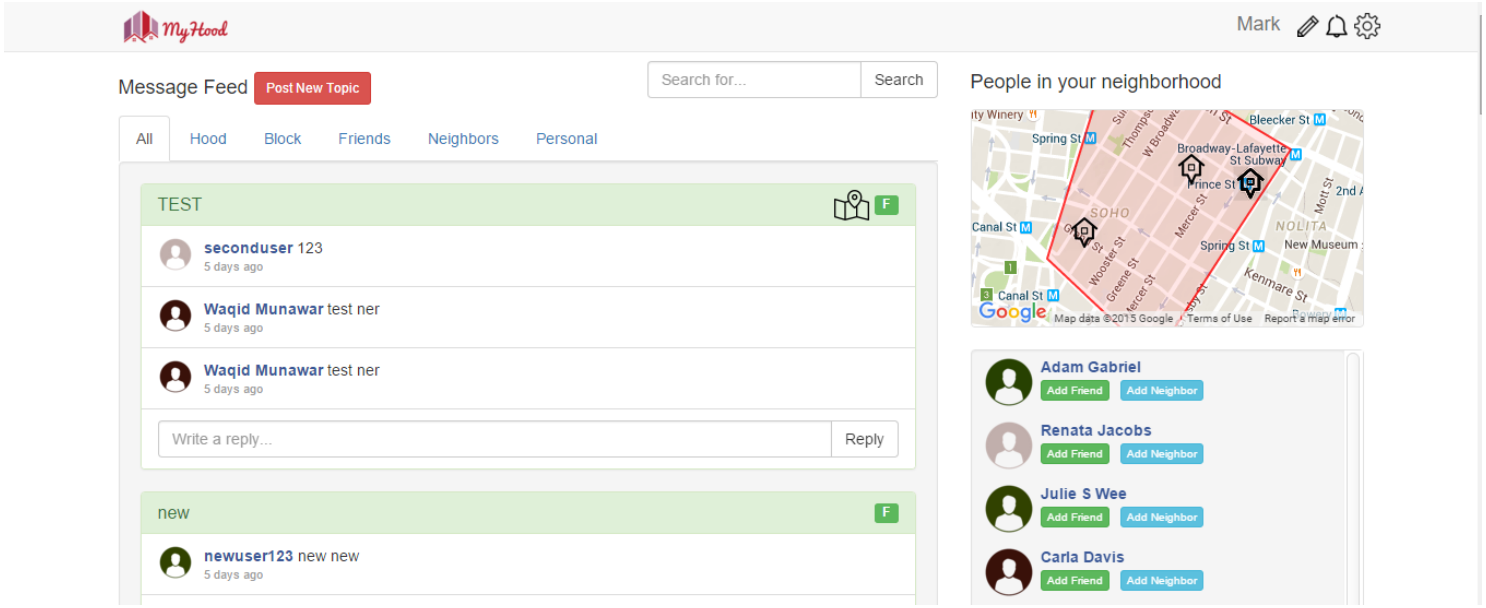
A user receives a notification where he can view the users awaiting block approvals and also the pending friend requests (if any). He then choses whether to accept those requests or not. Let us consider that the user Mark has been approved by 3 existing members of that block.



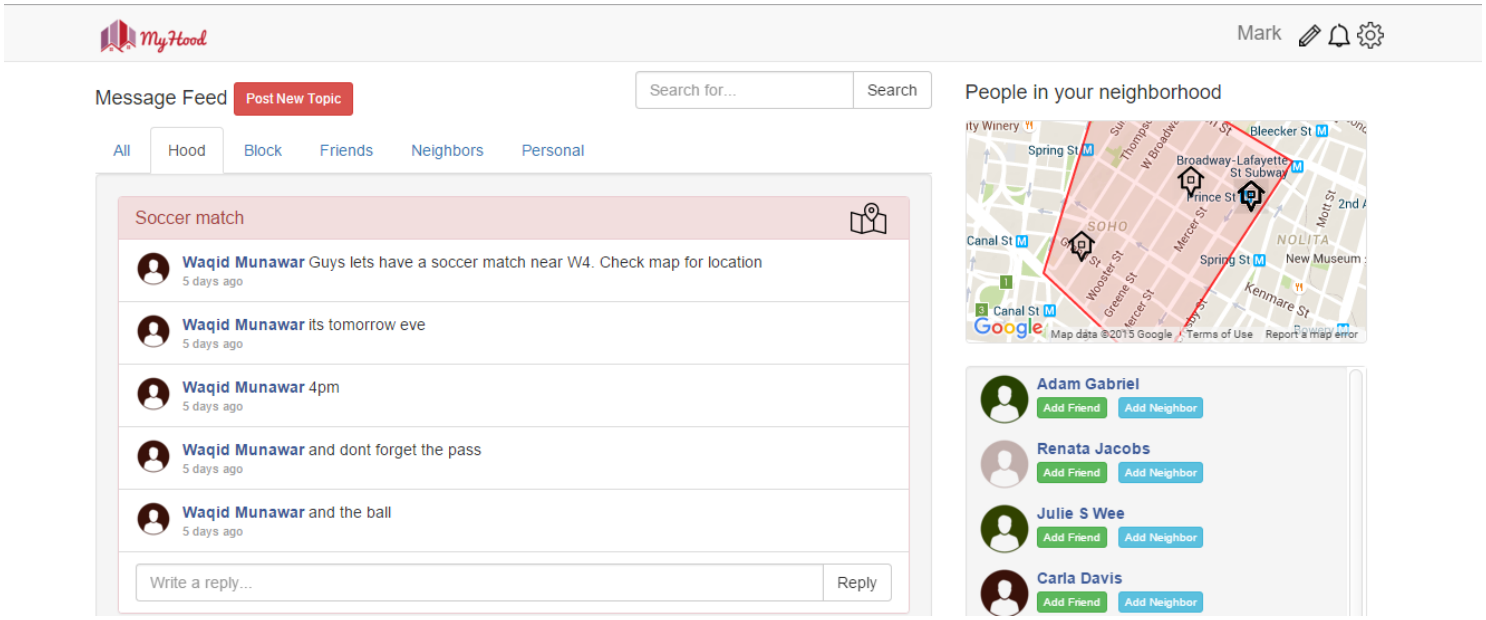
Step4: Message Feed

Once the user becomes a member of a block he can view message feed as a cluster or he can view them under various tabs as shown below.

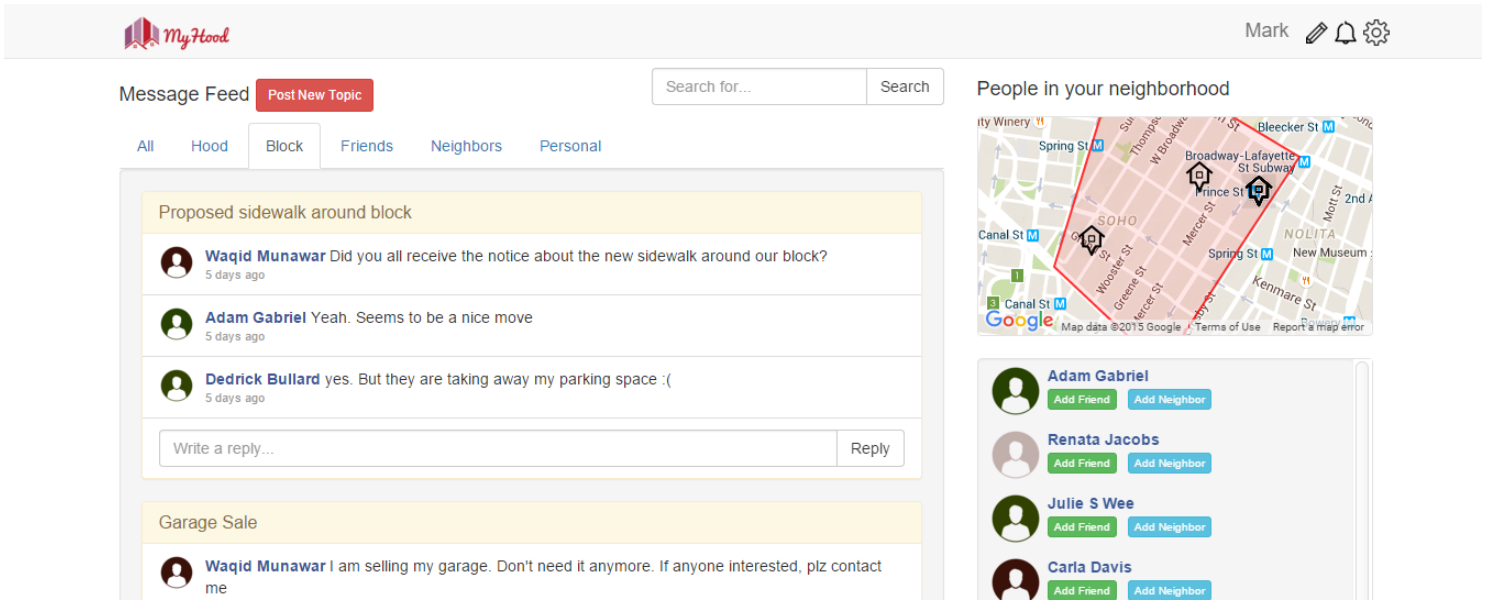
View all feed:



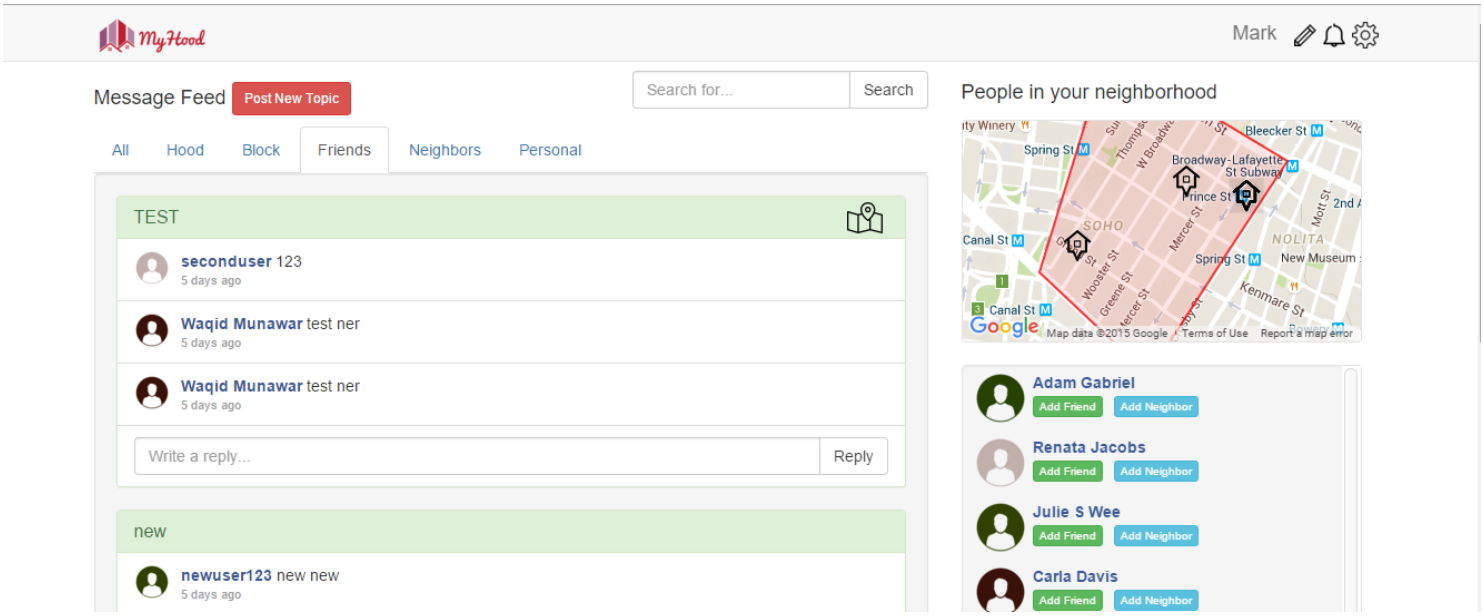
View Hood Feed:



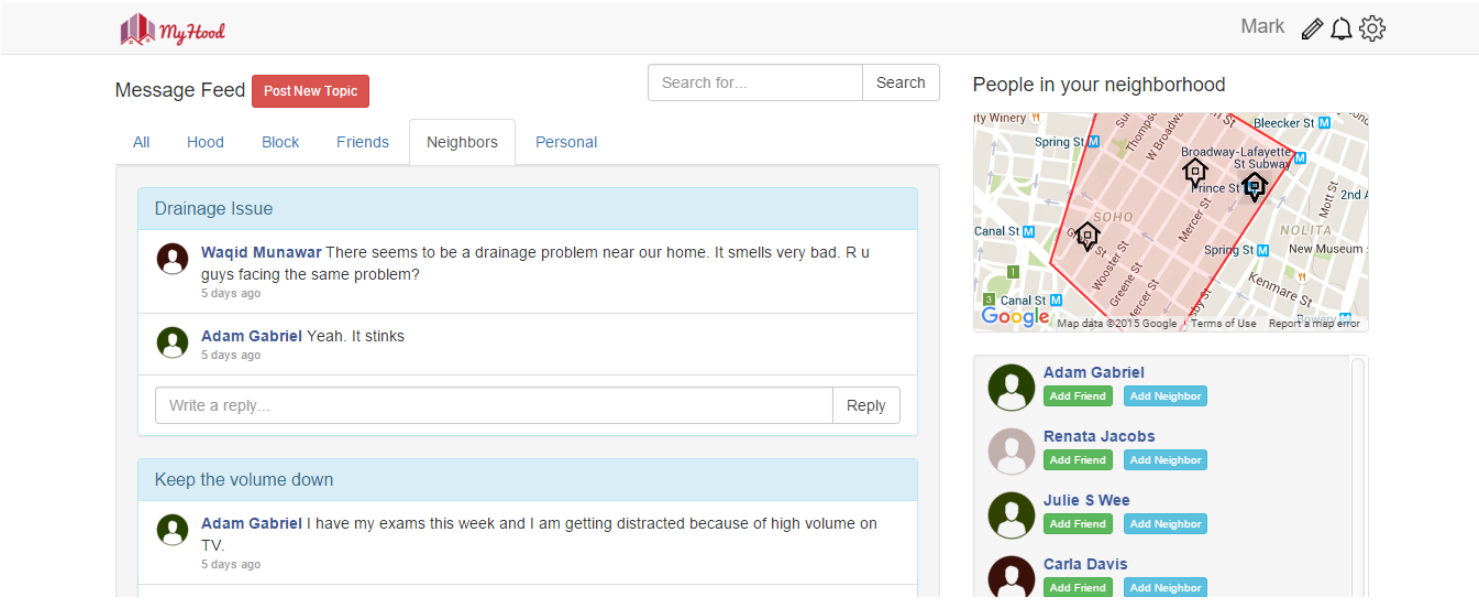
View Block Feed:



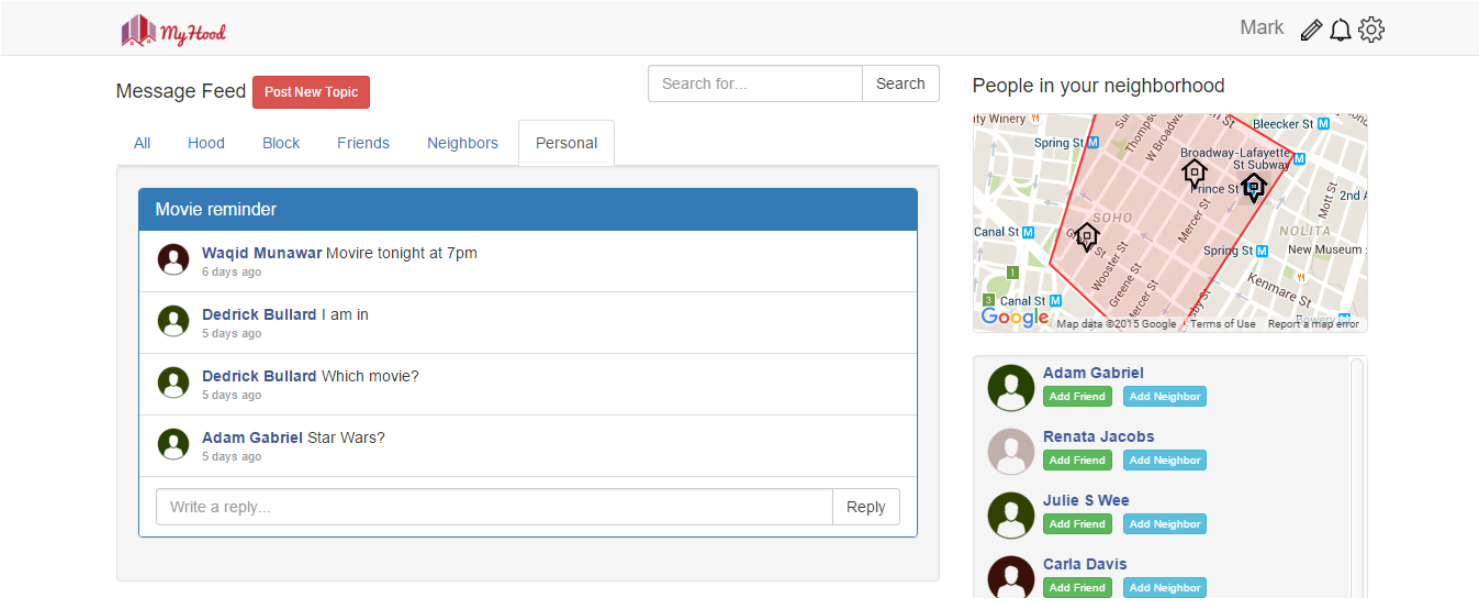
View Friends feed:



View Neighbor Feed:

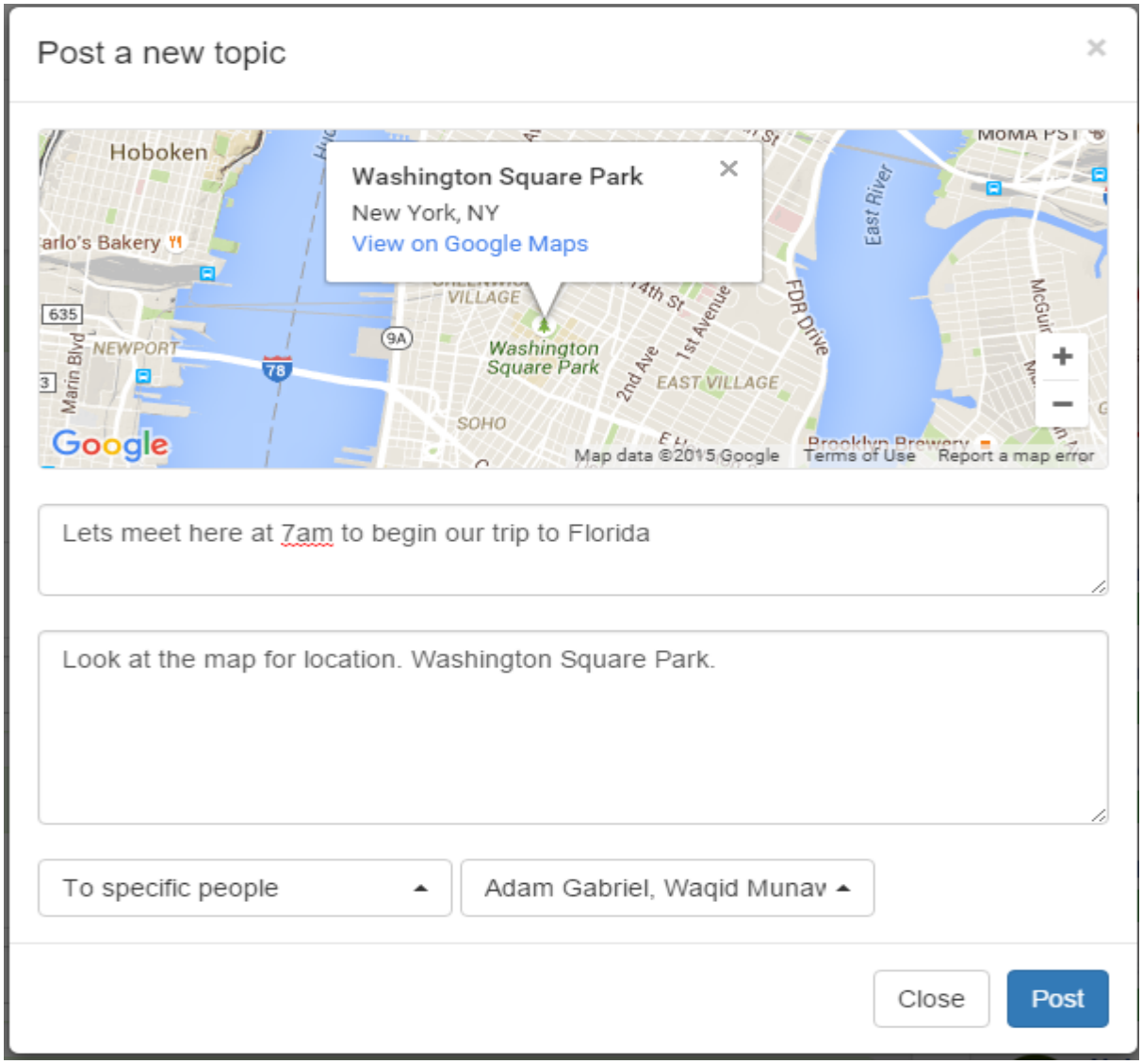


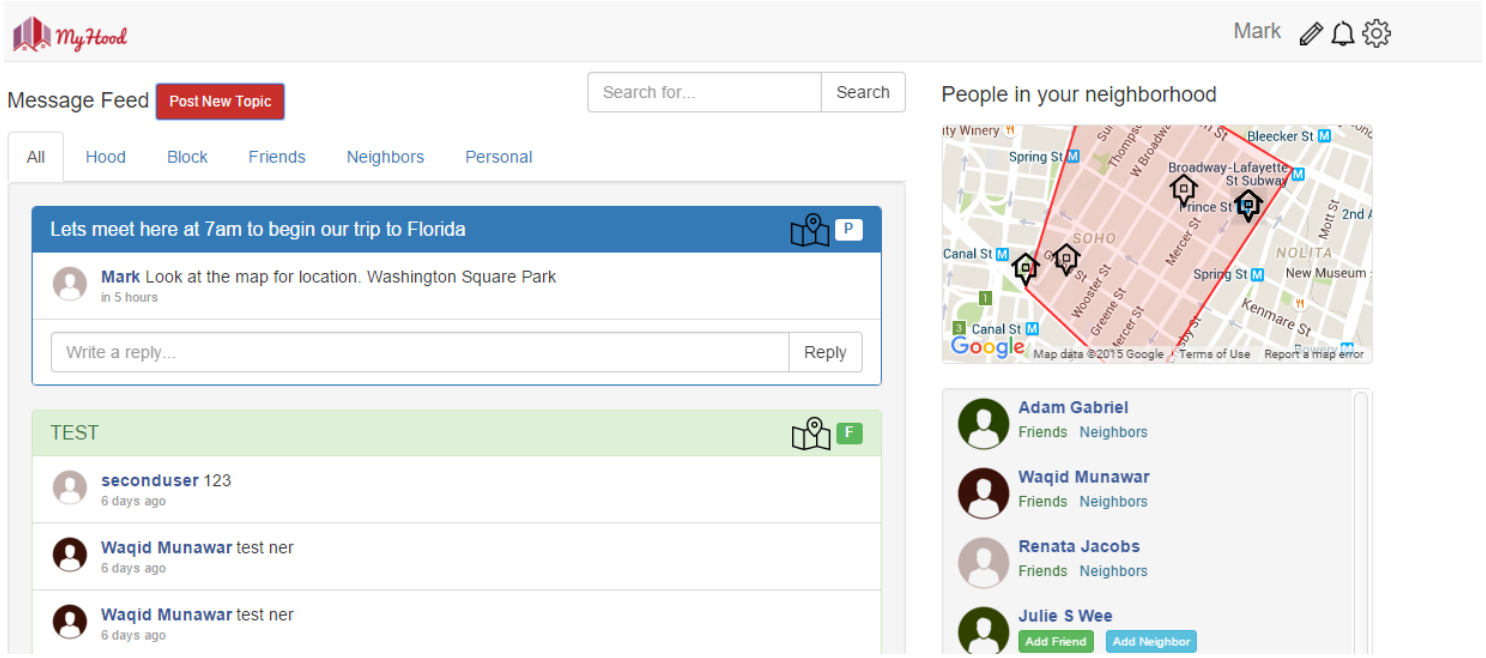
View Personal Feed:



Step5: Posting topics and replying to messages

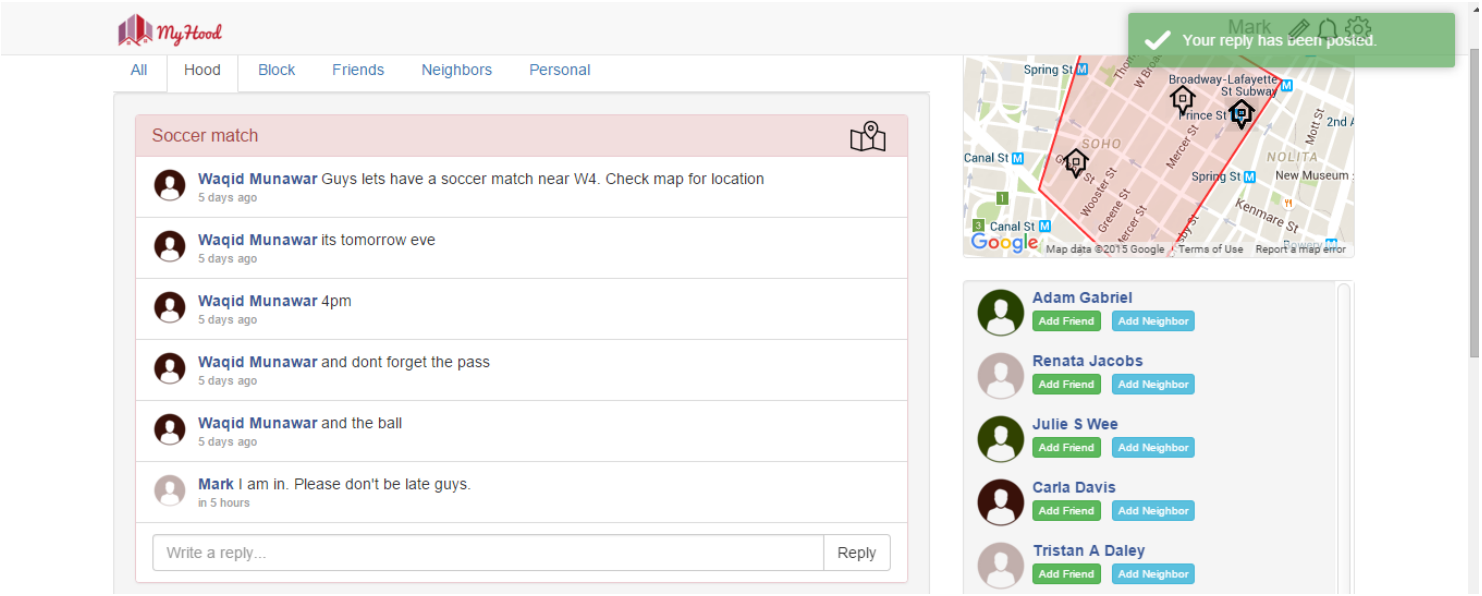
A user can also create new topics, tag locations for those topics, and also reply to messages for topics in which he is tagged in.


Post a new topic
Washington Square Park
New York, NY
View on Google Maps
Hoboken
NEWPORT
VILLAGE
Washington Square Park
EAST VILLAGE
SOHO
East River
FDR Drive
1st Avenue
2nd Ave
14th St
Google
Map data ©2015 Google
Terms of Use
Report a map error
Lets meet here at 7am to begin our trip to Florida
Look at the map for location. Washington Square Park.
To specific people
Adam Gabriel, Waqid Munav
Close
Post


MyHood
Mark
Message Feed
Post New Topic
Search for...
Search
All
Hood
Block
Friends
Neighbors
Personal
Lets meet here at 7am to begin our trip to Florida
P
Mark Look at the map for location. Washington Square Park
in 5 hours
Write a reply...
Reply
TEST
F
seconduser 123
6 days ago
Waqid Munawar test ner
6 days ago
Waqid Munawar test ner
6 days ago
People in your neighborhood
Spring St
Bleecker St
Broadway-Lafayette St
Prince St
SOHO
NOLITA
Canal St
Mercer St
Wooster St
Greene St
Spring St
New Museum
Kenmare St
Google
Map data ©2015 Google
Terms of Use
Report a map error
Adam Gabriel
Friends Neighbors
Waqid Munawar
Friends Neighbors
Renata Jacobs
Friends Neighbors
Julie S Wee
Add Friend
Add Neighbor

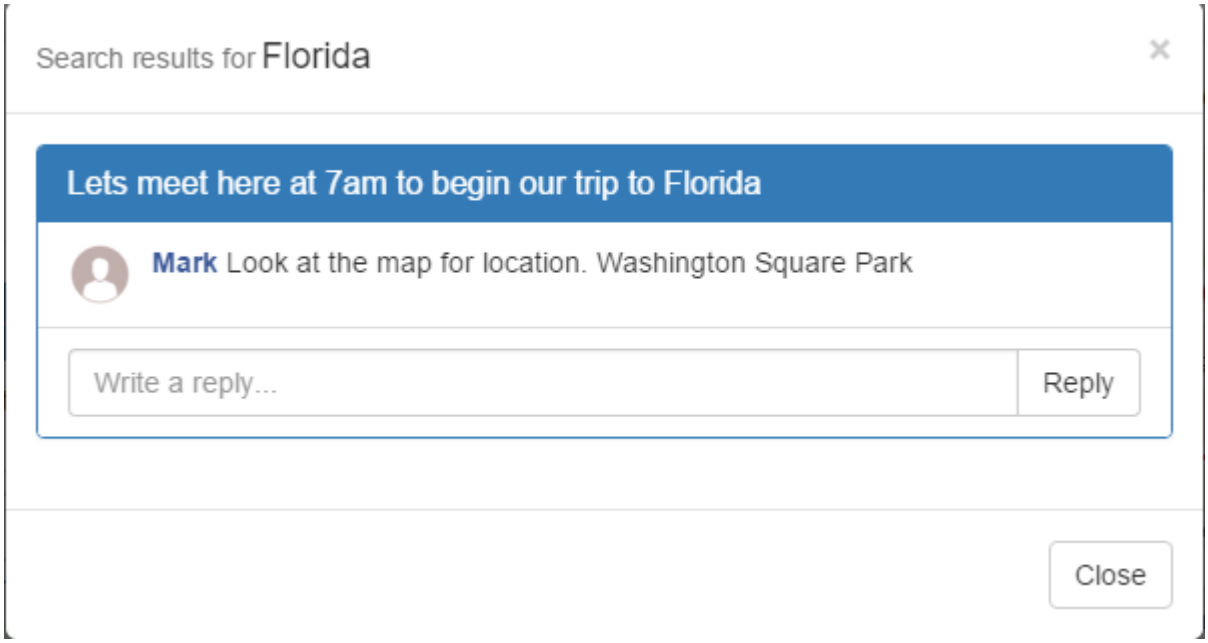
	topicId	subject	authorUserId	location	created	tagType	recipient	parentHoodId	modified
▶	35	Accident on main street!	21	0	2015-12-17 04:37:59	HOOD	5	5	2015-12-17 04:41:29
	36	Garage Sale	31	NULL	2015-12-17 06:56:34	BLOCK	58	5	2015-12-18 14:57:15
	37	Movie reminder	31		2015-12-18 10:30:08	PERSONAL	21,26	5	2015-12-18 15:00:52
	38	Christmas Party Next Week	31	NULL	2015-12-18 08:56:35	NEIGHBORS	21,24,28	5	2015-12-18 14:59:25
	39	Keep the volume down	21	NULL	2015-12-18 09:03:35	NEIGHBORS	24,26,31	5	2015-12-18 15:01:53
	40	Birthday Party Planning	24	NULL	2015-12-18 09:03:59	FRIENDS	21,26,28,31	5	2015-12-18 14:58:19
	41	Drainage Issue	31	NULL	2015-12-18 09:04:35	NEIGHBORS	21,24,28	5	2015-12-18 14:47:10
	42	Dinner at my place tonight	31	NULL	2015-12-18 09:09:35	FRIENDS	21,24,26,28,30	5	2015-12-18 14:57:53
	43	Proposed sidewalk around block	31	NULL	2015-12-18 09:18:35	BLOCK	58	5	2015-12-18 14:55:15
	44	Vote for our neighbourhood	21	NULL	2015-12-18 09:19:35	HOOD	5	5	2015-12-18 14:54:19
	45	Soccer match ...	31	40.730...	2015-12-18 18:59:14	HOOD	5	5	2015-12-24 07:30:49
	46	New post	70	40.724...	2015-12-18 21:57:53	FRIENDS	72	6	2015-12-18 21:57:53
	47	new	70		2015-12-18 22:00:01	FRIENDS	72	6	2015-12-18 22:00:01
	48	TEST	72	40.722...	2015-12-18 22:11:25	FRIENDS	70	4	2015-12-18 22:34:01
	51	Lets meet here at 7am to begi...	76	40.731...	2015-12-25 00:15:09	PERSONAL	21,31	5	2015-12-25 00:15:09

Message reply:



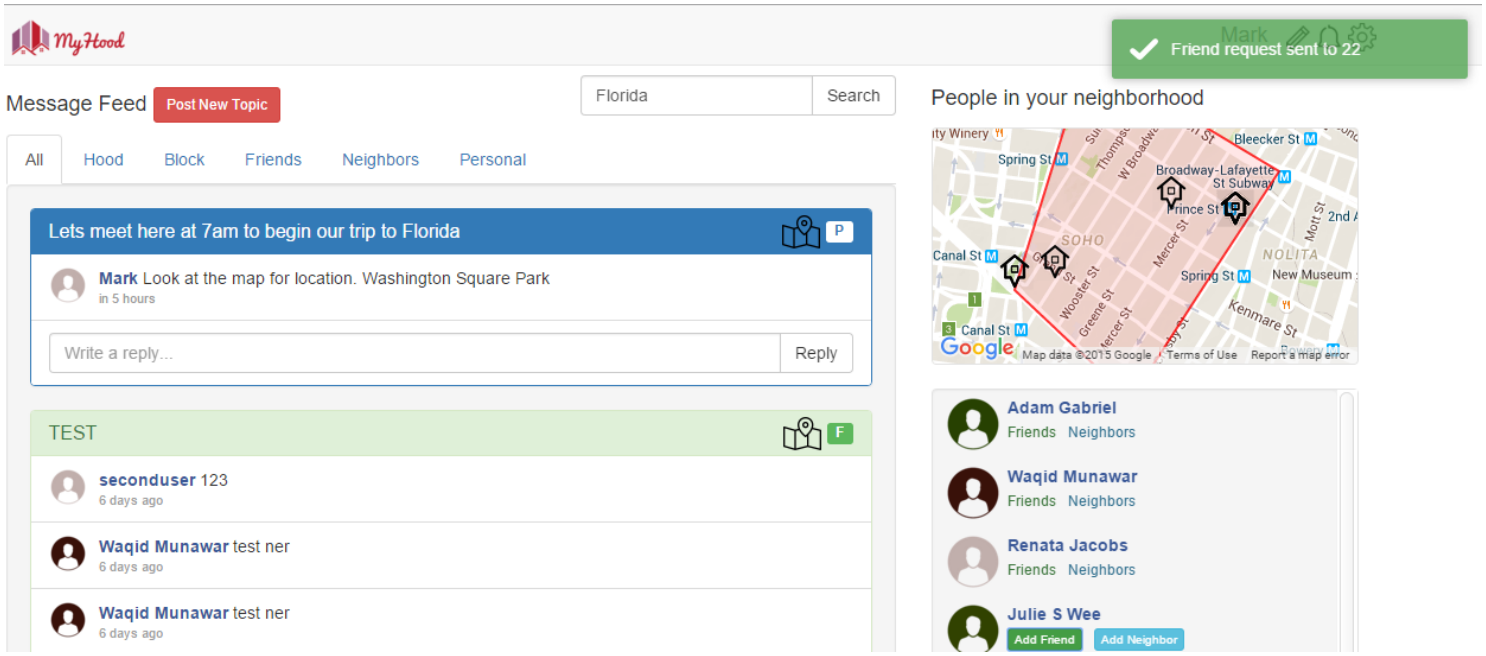
Step6: Search for messages

By entering a keyword in the search box, a user can search for all the messages which contain that particular keyword. For example searching for Florida, gives the output shown in the image.



Step7: Adding a Friend

A user can select "Add Friend" option and send a friend request to any person belonging to his hood. When the other user sees the request from the user in the notifications tab, he may decide whether to accept it or just ignore.



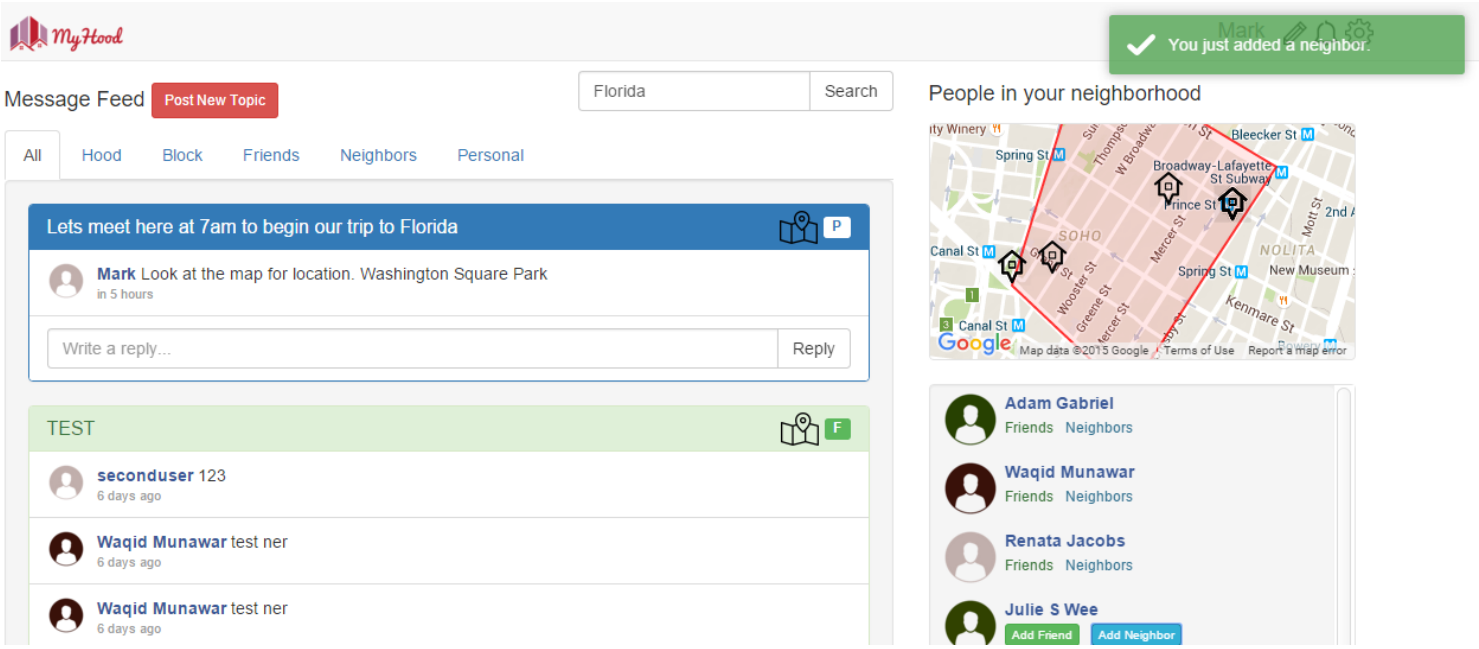
	fromUserId	toUserId
▶	30	31
	31	21
	26	31
	74	70
	76	22

When the other user (userId:22) accepts the friend request, it is updated in the Relation table.

	firstUserId	secondUserId	isFriend	isNeighbor
	24	31	1	1
	26	24	1	0
	24	28	1	1
	31	76	1	1
	24	76	1	1
▶	76	22	1	0

Step8: Adding neighbors

If a user sees anyone who is his neighbor, he can add him by clicking on the "Add Neighbor" button.



	firstUserId	secondUserId	isFriend	isNeighbor
	24	31	1	1
	26	24	1	0
	24	28	1	1
	31	76	1	1
	24	76	1	1
▶	76	22	1	1

Step9: Edit Profile

When a user wants to change his details, he can do so by clicking on the Edit Profile button.