

Hy925 Huiyu Yang
XI3074 Xinqi Liu

Project 2

DataBase Design:

Our DataBase Include 9 Tables

Hood table is used to store all the predefined neighborhood information, each neighbor has a unique hood id works as primary key to identify different neighborhood. Also we store the neighborhood's name in our database

```
hood (  
  hid int(11) NOT NULL,  
  hname VARCHAR(20) NOT NULL,  
  PRIMARY KEY (`hid`)  
)
```

Block table is used to store all the predefined block information. Each block has a unique block id works as primary key to identify different block. We store the neighborhood id in the block table to suggest the neighborhood that the block belongs to. Also, we store the geo information for block, latitude1 and 2 are used to define the hood's latitude range, longitude1 and 2 are used to define the hood's longitude range

```
block(  
  bid int auto_increment,  
  hid int,  
  bname varchar(20) NOT NULL,  
  latitude1 float NOT NULL,  
  latitude2 float NOT NULL,  
  longitude1 float NOT NULL,
```

```

longitude2 float NOT NULL,
PRIMARY KEY (`bid`),
FOREIGN KEY (`hid`) REFERENCES `hood`(`hid`)
)

```

UserInfo table is used to store all the user's information. Each user has a unique uid works as primary key to identity different user. Also we store each user's password, email, and address these credential information. We store the hood and block information for user, if user does not belong to any hood or block, these value are default to be null.

```

userInfo (
  uid VARCHAR(30) NOT NULL,
  upassword VARCHAR(225) NOT NULL,
  uemail VARCHAR(225) NOT NULL,
  `uaddress` VARCHAR(2250) NOT NULL,
  `upostcode` int(11) NOT NULL,
  `ublock` int(11) default NULL,
  `uhood` int(11) default NULL,
  `uintro` VARCHAR(30) default NULL,
  `uimg` LONGBLOB default null,
  PRIMARY KEY (`uid`),
  FOREIGN KEY (`ublock`) REFERENCES `block` (`bid`),
  FOREIGN KEY (`uhood`) REFERENCES `hood` (`hid`)
)

```

The block application table stores all users' block application records, we store the applicant's uid and which block he has applied to. The 'approver' column is used to store how many block members has approved the applicant's request

```

blockApplication (
  `uid` VARCHAR(30),
  `bid` int(11) NOT NULL,
  `approver` int(11) default 0,
  primary key(`uid`,`bid`),
  FOREIGN KEY (`uid`) REFERENCES `userInfo` (`uid`),
  FOREIGN KEY (`bid`) REFERENCES `block` (`bid`)
)

```

The message table is used to store all the messages that users send out, each message has a unique message id to identify itself. We store the message's title, author, sent out time, content, and type(to all the block members, friends, or neighbors)

```

messages (
  `mid` INT(11) AUTO_INCREMENT,
  `mtitle` VARCHAR(20) NOT NULL,
  `mauthor` VARCHAR(30) NOT NULL,
  `mtimestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `mtext` char(100) NOT NULL,
  `type` INT(11) NOT NULL,/*friends=1,neighbor=2,block=3*/
  primary key(`mid`, `type`),
  FOREIGN KEY (`mauthor`) REFERENCES `userInfo` (`uid`)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

The messages permission store's the message and user relation. It store's each message can be read by which users and store that mid and uid into the table, because mtimestamp and user can also be used to identify unique message. So here we use mauthor, mtimestamp and user's id (user who can read this message) as primary key.

```

`messagePermission` (
  `mauthor` VARCHAR(30) NOT NULL,
  `mtimestamp` timestamp NOT NULL,
  `rid` VARCHAR(30),
  `type` INT(11) NOT NULL,/*friend=2,neighbor=1,block=0*/
  `readStatus` INT(11) default 0,/*read=1*/
  KEY `mtimestamp` (`mtimestamp`),
  KEY `rid` (`rid`),
  KEY `type` (`type`),
  KEY `readStatus` (`readStatus`),
  primary key (`mauthor`, `mtimestamp`, `rid`),
  FOREIGN KEY (`mauthor`) REFERENCES `userInfo` (`uid`)
)

```

The friend table used to store all the friend request and friends information. We store two user's uid in this table and use status to show whether they are friends or not

```

`friends` (
  `friend1` VARCHAR(30) NOT NULL,
  `friend2` VARCHAR(30) NOT NULL,
  `status` INT(11) default 0,/*pending=0,accept=1,*/
  primary key(`friend1`, `friend2`),
  FOREIGN KEY (`friend1`) REFERENCES `userInfo` (`uid`),
  FOREIGN KEY (`friend2`) REFERENCES `userInfo` (`uid`)
)

```

```
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

The friend table used to store all the neighbor information. Since neighbor relation is not mutual, we use neighbor1 as primary key and neighbor2 are his neighbors

```
`neighbors` (  
  `neighbor1` VARCHAR(30) NOT NULL,  
  `neighbor2` VARCHAR(30) NOT NULL,  
  primary key(`neighbor1`,`neighbor2`),  
  FOREIGN KEY (`neighbor1`) REFERENCES `userInfo` (`uid`),  
  FOREIGN KEY (`neighbor2`) REFERENCES `userInfo` (`uid`)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

We use userLogStatus to store all the user's login and logout activities

```
`userLogStatus`(  
  `uid` VARCHAR(30) NOT NULL,  
  `loginTime` timestamp DEFAULT CURRENT_TIMESTAMP ON UPDATE  
  CURRENT_TIMESTAMP,  
  `logoutTime` timestamp DEFAULT CURRENT_TIMESTAMP ON UPDATE  
  CURRENT_TIMESTAMP,  
  primary key(`uid`),  
  FOREIGN KEY (`uid`) REFERENCES `userInfo` (`uid`)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Database insertion and update and selection:

Basic Information:

When the user want to register his id and insert into the userInfo table with information like password,uemail,uaddress,upostcode from the several bars:

```
INSERT into userInfo(uid,upassword,uemail,uaddress,upostcode) VALUES(?, ?, ?, ?, ?)
```

When the user needs to login in the myNextDoor:

```
SELECT uid, upassword FROM userInfo WHERE uid = ?
```

When the pages wants to get the information about himself:

```
SELECT * FROM userInfo WHERE uid = ?
```

When user wants to update his profile with user's new data input:

UPDATE userinfo SET uaddress=?, upostcode=?, uintro=?, uimg=? WHERE uid=?;

Message Information:

When user needs to store message info into message table:INSERT into messages(mtitle,mauthor,mtimestamp,mtext,type) VALUES(?, ?, ?, ?, ?)

When user needs to store permission list for friend msg into messagePermission table:
INSERT into messagePermission(mauthor, mtimestamp, rid, type) select ?, ?, uid, ?
from userInfo WHERE uid in (select friend1 from friends where status = 1 and friend2 =
?) or uid in (select friend2 from friends where status = 1 and friend1 = ?) or uid = ?;

When user needs to store permission list for neighbor message into messagePermission table:

INSERT into messagePermission(mauthor, mtimestamp, rid, type) select ?, ?, uid, ?
from userInfo WHERE uid in (select neighbor2 from neighbors where neighbor1 = ?) or
uid = ?;

When user needs to store permission list for block message into messagePermission table:

INSERT into messagePermission(mauthor, mtimestamp, rid, type) select ?, ?, uid, ?
from userInfo a WHERE a.ublock = (select ublock from userInfo where uid = ?)

ApplyInformation:

When user wants to get user's block member list:

select a.uid from userInfo a, userInfo b where b.uid=? and a.uid<>b.uid and
a.uaddress=b.uaddress;

When user wants to get user's neighbor list:

Select neighbor2 from neighbors where neighbor1=?;

User log information:

When user needs to update his log related information:

insert into userLogStatus(uid, loginTime, logoutTime) values (?, ?, ?);

When user needs to logout his mainpage:

update userLogStatus set logoutTime = ? where uid = ?;

Display requested messages:

Display All messages's brief information without any limitation:

select mid, mtitle, mauthor, mtimestamp, mtext from messages where (mauthor, mtimestamp) in (SELECT mauthor, mtimestamp from messagePermission where rid = ?)

Update message readStatus when the user has read this message:

update messagePermission set readStatus = 1 where (mauthor, mtimestamp) = (select mauthor, mtimestamp from messages where mid = ?)

Display All messages without any limitation:

select mid, mtitle, mauthor, mtimestamp, mtext, type from messages where mid = ?

Display messages without topic selection:

select mid, mtitle, mauthor, mtimestamp, mtext from messages where (mauthor, mtimestamp) in (SELECT mauthor, mtimestamp from messagePermission where rid = ? and type = ? and readStatus <= ?)

Display message with topic selection:

select mid, mtitle, mauthor, mtimestamp, mtext from messages where (mauthor, mtimestamp) in (SELECT mauthor, mtimestamp from messagePermission where rid = ? and type = ? and readStatus <= ?) and (find_in_set(?, mtext) or find_in_set(?, mtitle));

Block application:

Check application Status:

select bid from block b where exists (select * from userInfo where uid = ? and ublock = 4 and ublock = b.bid) or exists (select * from blockApplication where uid = ? and bid = 4 and bid = b.bid);

Update application Statue:

insert into blockApplication(uid, bid) values (?,4);'

Get and Response to requests:

Select who is applying:

select a.uid from blockApplication a where not exists (select * from userInfo where uid = a.uid and ublock = a.bid) and a.bid = (select ublock from userInfo where uid = ?);

Increase approver number to blockApplication:

update blockApplication set approver = approver+1 where uid = ?;'

Check number of current approvers:

select approver from blockApplication where uid = ?;

Update ublock in userInfo:

update userInfo set ublock = 4, uhood = 3 where uid = ?;

Neighbor Application:

Recommend Neighbor Application:

select a.uid from userInfo a, userInfo b where b.uid = ? and b.ublock = a.ublock and a.uid <> b.uid and (b.uid, a.uid) not in (select neighbor1, neighbor2 from neighbors);'

Add neighbor record after apply, no need to further accept:

insert into neighbors(neighbor1, neighbor2) values (?,?);'

Friends Application:

Recommend Friends Application:

select a.uid from userInfo a, userInfo b where b.uid = ? and b.ublock = a.ublock and a.uid <> b.uid and ((b.uid, a.uid) not in (select friend1, friend2 from friends)) and ((a.uid, b.uid) not in (select friend1, friend2 from friends))

Add friend record after apply, status = 0, still not friend

insert into friends(friend1, friend2) values (?,?)

Show list of who applied to be friend and is pending

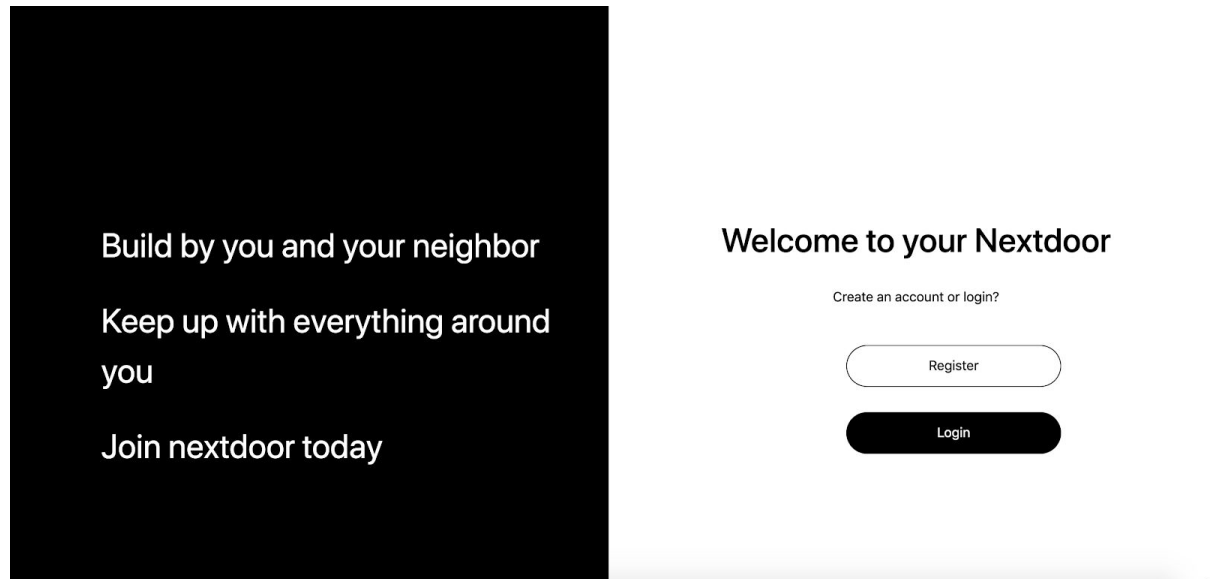
select friend1 from friends where friend2 = ? and status = 0;

Accept friends request:

update messagePermission set readStatus = 1 where (mauthor, mtimestamp) = (select mauthor, mtimestamp from messages where mid = ?)

Website Design:

Welcome page:



In the welcome page user can hit register button to register if he does not have an account, or hit login button to login.

Register page:

User needs to provide his user name, email, password, address, and postcode to register. We check if password and password2 are agreed in the front end, we also use express passport to hash user's password and insert to database to ensure security

Login page

Build by you and your neighbor
Keep up with everything around
you
Join nextdoor today

Login

Username

Password

Build by you and your neighbor
Keep up with everything around
you
Join nextdoor today

Login

invalid username!

✕

Username

Password

Build by you and your neighbor
Keep up with everything around
you
Join nextdoor today

Login

invalid password!

✕

Username

Password

We use username to search in database, if no such user, we will ask the user to register an account. If the user exists in database use express passport the hash user's input password and compare with the hashed password in database, if they are the same user will enter main page, otherwise we will tell user it is a wrong password.

Main Page

Dashboard

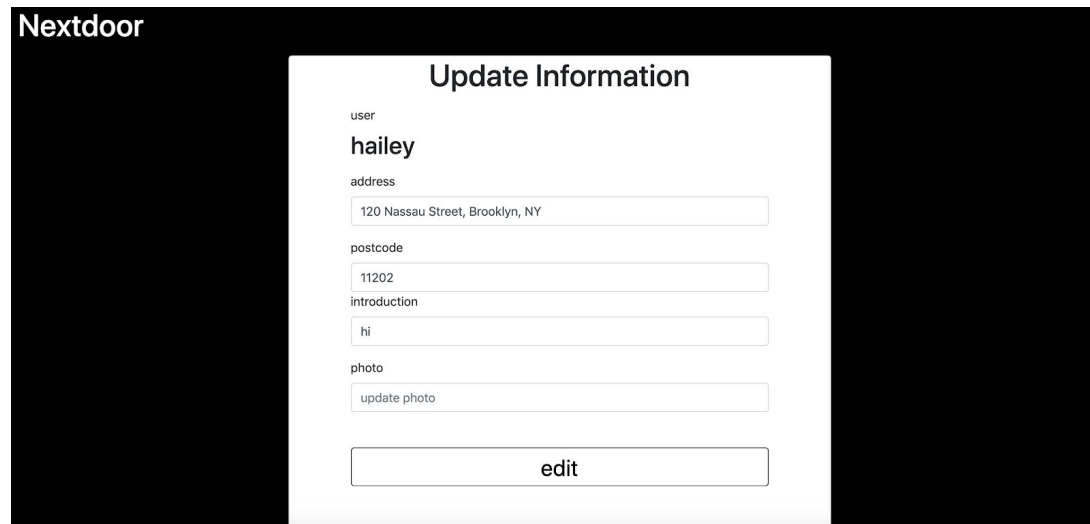
Welcome hailey

[logout](#) [profile](#) [add connection](#) [POST MESSAGE](#) [Join local block](#) [Friends/Block Requests](#) [Incoming Messages](#) [map](#) [t](#)

NY

Main page contains buttons that users can click on

Profile Page



The screenshot shows the 'Nextdoor' logo in the top left corner. The main content is a white box titled 'Update Information' on a black background. Inside the box, the user's name 'hailey' is displayed under the label 'user'. Below this, the 'address' field contains '120 Nassau Street, Brooklyn, NY'. The 'postcode' field contains '11202'. The 'introduction' field contains 'hi'. The 'photo' field has a button labeled 'update photo'. At the bottom of the form is a large 'edit' button.

When user hit the profile button, user can view his own information and edit it. By hitting edit button, the new user's information will be updated to database.

addConnection Page



In add connection page, user will be provided with neighbor suggestion list containing people in the same block with user and but yet in user's neighbor list. By clicking the name button user can add neighbor. Also, user will be provided with friend suggestion list containing people in the same block with user and but yet in user's friend list. By clicking the name button user can send friend request to other users who he wants to be friends with.

PostMessage Page

Share something interesting!

hailey

Title

Enter title

Content

Enter messeage

Visibility

block

post

Share something interesting!

Posted!!!

hailey

Title

Enter title

Content

Enter messeage

Visibility

block

post

In post

message page, user can send out messages by entering title, content, and the people he wants to see this post(block message,neighbors, friends).By hitting post, the message will be stored in database.

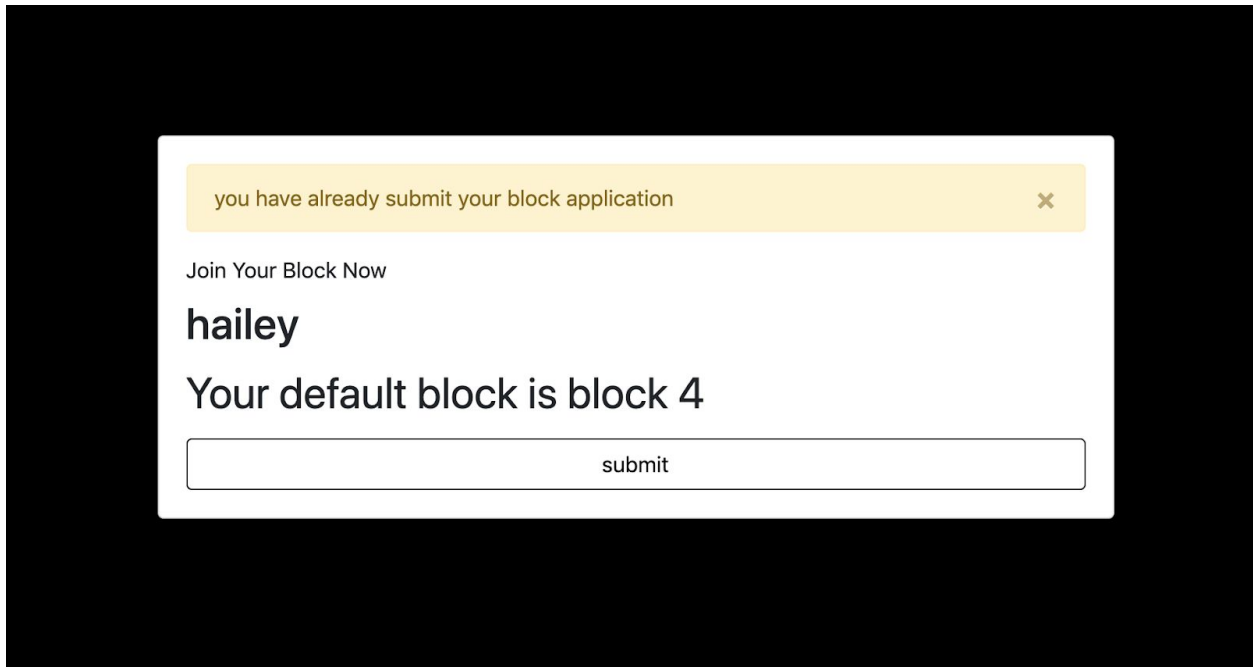
Join local block page

Join Your Block Now

hailey

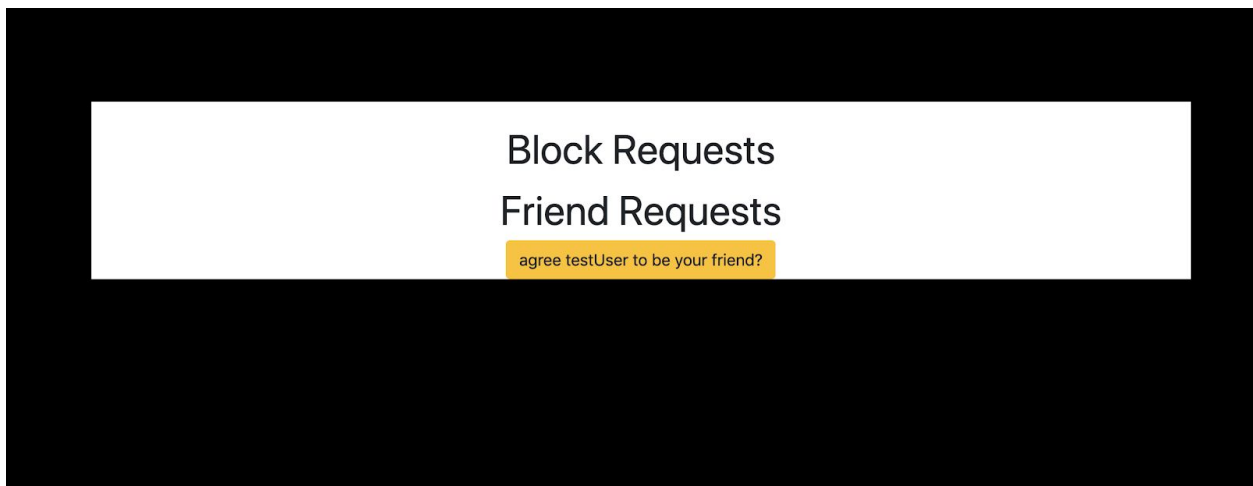
Your default block is block 4

submit



In join block page, user will see his default block, and by hitting submit, user will submit a join block request. If the user has already sent out the block requests or already in a block, we will tell user.

Friend/Block requests page



In the block/friends, user will see all the requests he got.including block request and friend requests. By clicking the name button user can confirm the request

Messages page

All Messages

111	hailey	Mon Dec 23 2019 13:11:19 GMT-0500 (Eastern Standard Time)
222	hailey	Mon Dec 23 2019 13:11:23 GMT-0500 (Eastern Standard Time)
333	hailey	Mon Dec 23 2019 13:11:28 GMT-0500 (Eastern Standard Time)
hi	hailey	Mon Dec 23 2019 15:58:21 GMT-0500 (Eastern Standard Time)
hi for test	hailey	Mon Dec 23 2019 17:01:29 GMT-0500 (Eastern Standard Time)
hi	hailey	Mon Dec 23 2019 23:11:23 GMT-0500 (Eastern Standard Time)
hi	yishu	Mon Dec 23 2019 23:19:06 GMT-0500 (Eastern Standard Time)
what's up	niki	Mon Dec 23 2019 23:19:24 GMT-0500 (Eastern Standard Time)

block

all messages

Enter keywords

search

All Messages

[hi for test](#) hailey Mon Dec 23 2019 17:01:29 GMT-0500 (Eastern Standard Time)

[hi](#) yishu Mon Dec 23 2019 23:19:06 GMT-0500 (Eastern Standard Time)

[what's up](#) niki Mon Dec 23 2019 23:19:24 GMT-0500 (Eastern Standard Time)

block

unread messages

Enter keywords

search

All Messages

[111](#) hailey Mon Dec 23 2019 13:11:19 GMT-0500 (Eastern Standard Time)

block

all messages

111

search

In message page user can see all the message he received. By clicking the title, user can see the full content. Also, user can search on message including(block/neighbor/friends) (allmessages/unread messages) and keywords. The message will display the messages that user wanted

Map page

hailey

120 Nassau Street, Brooklyn, NY

