



Pinnect

Design document

Feb. 19, 2021

Team 20

- Rafael Zhu
- Tianyi wang
- Benyu Jia
- Yuquan Xu

Purpose:	2
Design Outline:	2
High-level structure of the project system:	2
The purpose of each components:	3
The interactions between individual system components:	3
Design Issues:	4
Functional Issues:	4
Non Functional Issues:	6
Design Details:	8
Class level design of the system:	8
Sequence diagrams for different activities in the system:	10
UI mockup:	16

Purpose:

Sometimes, it is hard to know what is happening around you. Is there a newly opened shop? Has something special happened among your neighborhood last night? Is there something happened in your street that you just missed, so you have no ideas what your friends are talking about?

The purpose of this project is to develop a social media where users can pin a text or a series of images at their locations and interact with elements pinned by others, which allow people to connect to each other in a more fun and open way.

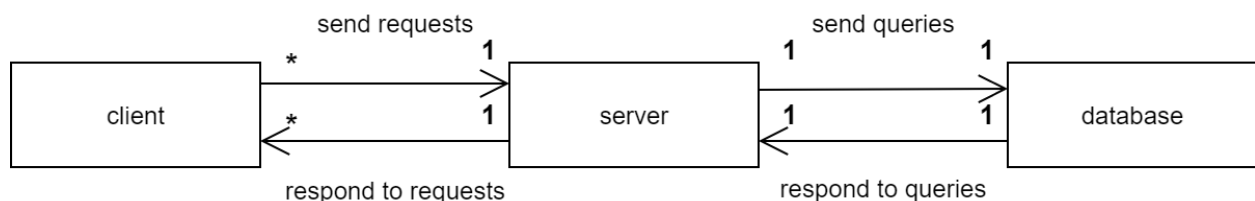
Popular social media like Zenly and Snapchat provide similar functions, but they tend to focus more on messages and images while location sharing is optional: For example, Zenly does not support leaving messages on the map, and Snapchat only supports video sharing with coarse location. Pinnect, however, tackles the problems above by enabling users to share versatile contents more precisely in terms of locations and interact with all users around the world. On top of that, the project also keeps users updated with the top trending posts and recommends the pins that are likely to match with the users' personal interests.

During the pandemic of Covid-19, people have been encouraged to isolate themselves from almost all social activities, but it is never a good solution in terms of psychological well-being. We initiated this project with the hope that Pinnect could make people socialize better in a more open and convenient manner and have everyone realize that there are always people around us who are willing to share and listen.

Design Outline:**High-level structure of the project system:**

Our project will be a location-based social media that allows users dropping a pin with texts, images, or #tags at their locations, browsing dropped pins around the world, and interacting with other pins. This application will use the client-server model due to its nature of allowing the server to connect a large number of distributed clients at the same time.

As shown in the figure, the clients will send requests to the server according to the users' operation, and the server will access the database depending on the request and respond to the clients' request.



The purpose of each components:Client:

1. The client initializes the graphic user interface.
2. The client sends http requests to the server.
3. The client establishes a websocket connection to the server.
4. The client receives and parses both the http and websocket response from the server and updates the user interface accordingly.

Server:

1. Server receives and handles http requests from clients.
2. Server handles websocket connections from clients.
3. Server fetches data from the database according to the request being handled.

Database:

1. Database stores all related data of the project.
2. Database responses to queries sent from the server by adding, finding, deleting, or updating data.

The interactions between individual system components:

Pinnect can be divided into three components: the client, the server, and the database.

The client and the server will be connected through the Internet. We will use two connecting technologies: HTTP and WebSocket. HTTP will be used for general-purpose requests like login, viewing pins. WebSocket will be used for user online chatting and receiving push messages.

The client will provide an interface for users to interact and handle their operations. However, most of the operations require the help from the server. Therefore, the client will send a request with necessary data to the server and wait for the response.

The server and the database are deployed on different machines due to security and scalability reasons and connect to each other via the Internet. The database will store all data for the Pinnect and provide methods for adding, deleting, editing, and querying them. If the server receives a request that requires an access to the data stored inside the database, the server sends queries to the database and handles the response accordingly.

Design Issues:**Functional Issues:****1. Do users need to login to use our service?**

- Option 1: No
- Option 2: Optional: pins can be viewed without logged in.
- Option 3: Yes

Choice: Option 1.

Justification: We choose to make the login optional because we feel that there is no apparent need for users to login if they simply want to browse the pins to see what interesting things are happening. Therefore, we decided that users can view and share pins without logged in. All other features are unlocked once they are signed in.

2. What information do we need for signing up an account?

- Option 1: Username and password
- Option 2: Username and password, and email address
- Option 3: Phone number and password

Choice: Option 2

Justification: A user has to set up an account to be identified as an interactable individual in Pinnect and unlock the full functionality.

Option 1 is the most basic combination of a credential that is also easy to implement.

Option 2 extends option 1 and furtherly gives the user a chance to recover its account when he loses access to it.

Option 3 makes it easier for users to sign up. However, we need to send verification code through SMS, which is a paid service and does not add much security to our app eventually.

Since we do not have the intention to add that cost to our project in the current phase and after considering the merits above, we eventually opted for option 2.

3. What protocol do we use to communicate with the server?

- Option 1: HTTP
- Option 2: WebSocket
- Option 3: HTTP + WebSocket

Choice: Option 3

Justification: We decided to use HTTP for user authentication and registration because they do not need continuous connection to work. In contrast, we use WebSocket for instant messaging because they rely on continuous connections for receiving pushed messages.

4. When do we acquire users' location information?

- Option 1: Always continuously acquire users' locations
- Option 2: Continuously acquire users' locations only when users are online
- Option 3: Only acquire users' locations when users login and create pins
- Option 4: Has a location-update button for users to voluntarily acquire current location

Choice: Option 2

Justification: We finally decided to use option 2 because of the feature: users are able to see friends' location. This requires the latest location coordinate from users. However, the location of the users is sensitive. Therefore, we decided only to collect locations of the users when they are online and provide an option for users to stop sharing locations.

5. How should we manage the tags of a pin?

- Option 1: Predefined set of pin tags
- Option 2: Customizable pin tags activated by hashtags
- Option 3: Customizable pin tags separated from contents

Choice: Option 2

Justification: Creating new tags using hashtags gives high flexibility for users to use any tags they like. It helps keeping up with the social trends since the topics vary from time to time and users can invent a new tag whenever a new trend emerges.

In terms of having a finite set of tags, it saves a huge effort for developers to manage the tags as they remain unchanged, but it limits users' creativity and blocks users from joining in trending topics. Having customizable tags outside of the text section gives the users freedom to create contents and help to organize the text, but it creates more problems and difficulties in developments. Hence, we decided to go with option 2.

6. What actions should we take on users who violate the rules?

- Option 1: Ban the account.
- Option 2: Restrict the accessible functionality.
- Option 3: Tag the user as inappropriate
- Option 4: Track these users' future activities

Choice: Option 4

Justification: We decided to punish the users who violate the rules depending on their past and future performance. Some users may violate rules unintentionally and it is certainly understandable, and we do not want to have them punished. However, some of

the users may violate the rules repeatedly on purpose, and in which case the punishment should be strong to keep the platform safe.

Therefore, we decided to develop a small system in Pinnect that keeps track of users' behaviors and rates them with scores. If a user has a low score, it will receive a punishment from option 1 to 3 accordingly. We hope this feature can decrease the harmful content on the media.

7. How should we recommend pins users might be interested in to them?

- Option 1: recommend pins when users login
- Option 2: recommend pins when users are online and do not open new pins for 10 minutes
- Option 3: recommend pins when users do not login for a long time

Choice: Option1

Justification: We came to the conclusion that option 1 is optimal because it makes use of the recommendation system at a preferable frequency and does not interrupt the users during the app as it pops up before any user activities. It also has enough space to have the user informed about the pin contents at once.

Option 2 is not practical considering users' usual activities. Based on our experience, most people would not be hesitating on an app for long and so the recommendation might rarely appear if it waits for 10 minutes of no response.

Option 3 is a practical solution and it does not cause disturbance but it creates additional problems as to how to show the content of the recommended pin inside a small pop-up window.

Non Functional Issues:

1. What backend language/framework should we use?

- Option 1: Java
- Option 2: Python
- Option 3: Go

Choice: Option 2.

Justification: Java has straightforward syntax that makes the project easy to maintain, and Go's goroutine feature allows developers to develop a high performance server with few lines of code. However, we finally chose Python as the primary language for backend development. Although it might have a slight impact on performance, its simple syntax allows fast prototyping, and the numerous libraries available greatly facilitates a rapid development.

2. What frontend language/framework should we use?

- Option 1: React Native
- Option 2: Flutter
- Option 3: Swift and Java

Choice: Option 1.

Justification: Different frontend frameworks have different development styles, libraries, and community support which will subsequently have an impact on our development. After consideration we decided to use React Native as the framework for frontend development because it allows us to write cross-platform mobile apps in JavaScript and also easy to learn. Its component architecture also helps us produce reusable and maintainable code.

Flutter is similar to React Native and is claimed to have a slightly better performance, but since it is newer, it has fewer libraries and smaller community, so we did not choose it.

Swift and Java can be used separately to write native mobile apps for iOS and Android and their builds usually have better results in terms of performance. But, in that case we have to maintain two different versions of code, which substantially slows down our development, and we consider the boost of performance is not needed in our project.

3. What database should we use?

- Option 1: SQL Relative Database like MySQL
- Option 2: NoSQL Database like MongoDB

Choice: Option 2.

Justification: We decided to use MongoDB over MySQL because it is schemaless, fast, and easy to scale; MongoDB documents can be converted to JSON objects naturally, which makes our frontend development easier; Also, because it does not require the structure of a document to be fixed, we can easily tailor our data structure to meet our needs without needing a lot of work.

4. What cloud service should we choose?

- Option 1: AWS
- Option 2: Azure
- Option 3: Google Cloud
- Option 4: Heroku

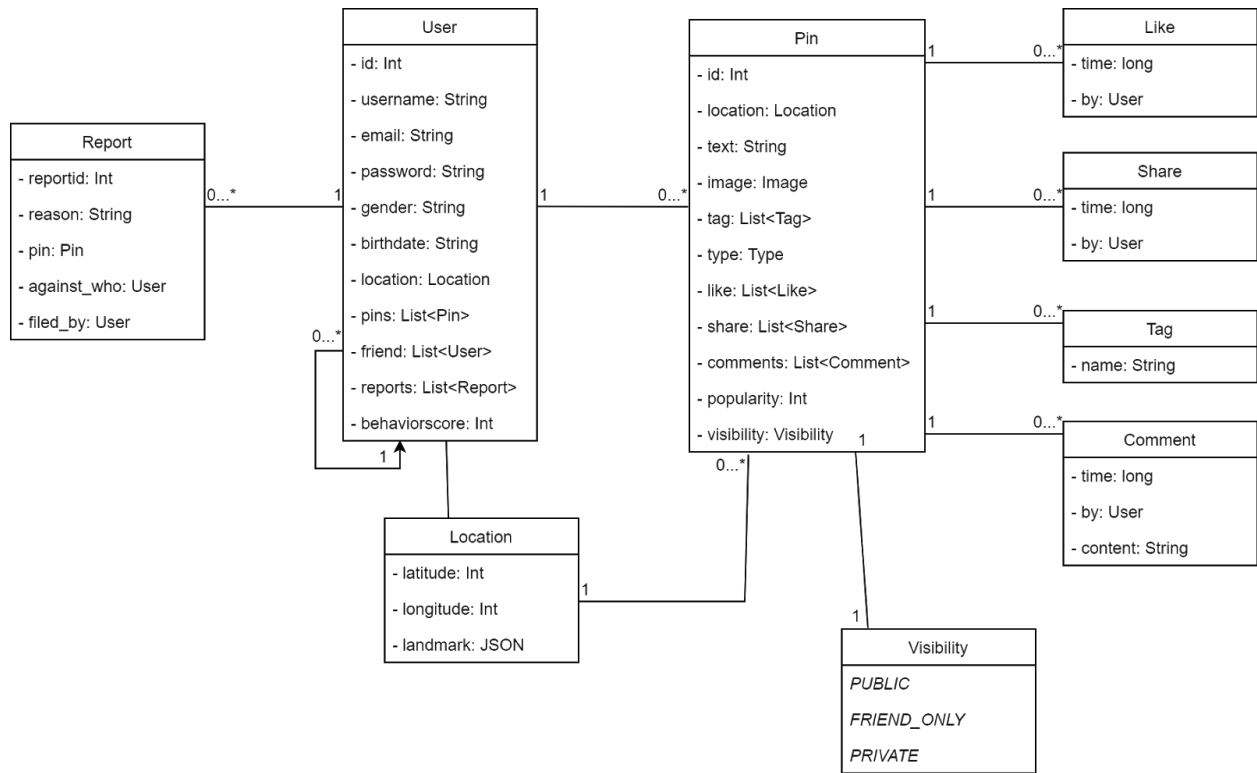
Choice: Option 4

Justification: Option 1 to 3 are IaaS cloud-computing platforms, meaning we need to maintain the system in a lower-level, including system configuration and monitoring, etc.

However, Heroku is a PaaS that does not require users to interact with low-level details; it also provides a better way to deploy applications. Therefore, we finally chose Heroku so we can focus on the development and deployment more.

Design Details:

Class level design of the system:



Descriptions of Classes:

The classes are designed based on the fundamental needs of the Pinnect system. Each class has the basic attributes listed along with their data type. The numeric relationships between classes are also shown. Below are detailed explanations of the design.

User:

- Each user has a unique user id, assigned by the system using a random number of a specific range.
- Basic user information, including username, email, gender, and birthdate, are set by users.
- Pin is a pin object created by a user. A user can have zero to multiple pins.
- Friend is another user the current user has established a friend relationship with. A user can have zero or multiple friends.

- Location is the location that the user is at. It will be updated accordingly to ensure the user location is accurate.
- Behavior score is a number initialized by the system, calculated based on the user behavior. If a user has a score below a certain threshold, the system will consider the user's behavior, and the user may receive punishments including limited access to functions or a ban.

Pin:

- Each pin has a unique id assigned by the system using a random number of a specific range.
- Location is where the pin is located on the map, determined by the location of the user who creates the pin. It contains the coordinate of the pin, and sometimes an attached landmark. Once created, it remains unchanged.
- Text and image are the contents of a pin. The text section has a word limit. Images are optional.
- Tag is the tags attached on a pin that are set by users. It is identified by the word following a hashtag in the text section of a pin.
- A pin must be assigned with a level of visibility, such as public, friends-only, or private.
- Like and share are the interactions between pins and users. They contain timestamps indicating when the interactions happen, and are linked to the users associated with the interaction.
- Popularity is a numeric attribute that is calculated based on the number of likes and shares of a pin. It determines how long the pin lasts on the map.

Report:

- Each report has a unique report id, assigned by the system using a random number of a specific range.
- Pin is the pin associated with this report.
- Reason is the content of a report. It is written by the user who issued the report describing how a pin or user causes discomfort or violates the rules.
- It also contains the user being reported and the user sending the report.
- A user might receive multiple reports, while a report can only address to one user.

Tag:

- A tag is associated with a pin.
- Name is set by the word following a hashtag in the text section of a pin.

Visibility:

- Each visibility indicates who can see the pin.
- Public means the pin can be viewed by anyone.

- Friend-only means the pin can only be viewed by the friends of the pin's creator.
- Private means the pin is not visible to anyone else.

Location:

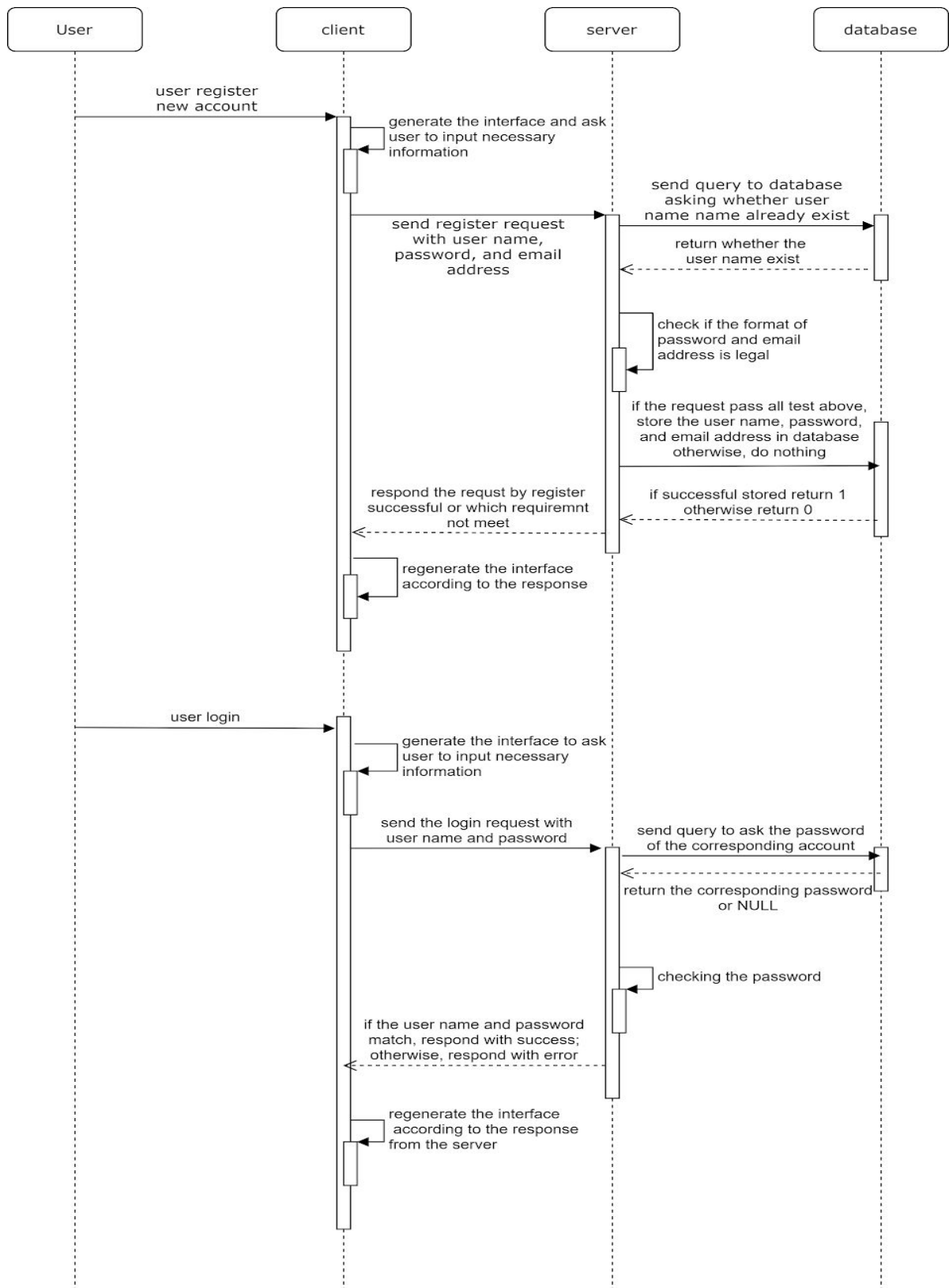
- Latitude and longitude are the coordinates of a user or a pin.
- Both attributes help locate precisely where the subject is on the map.
- Landmark is optional. It denotes an landmark associated with the location.

Like:

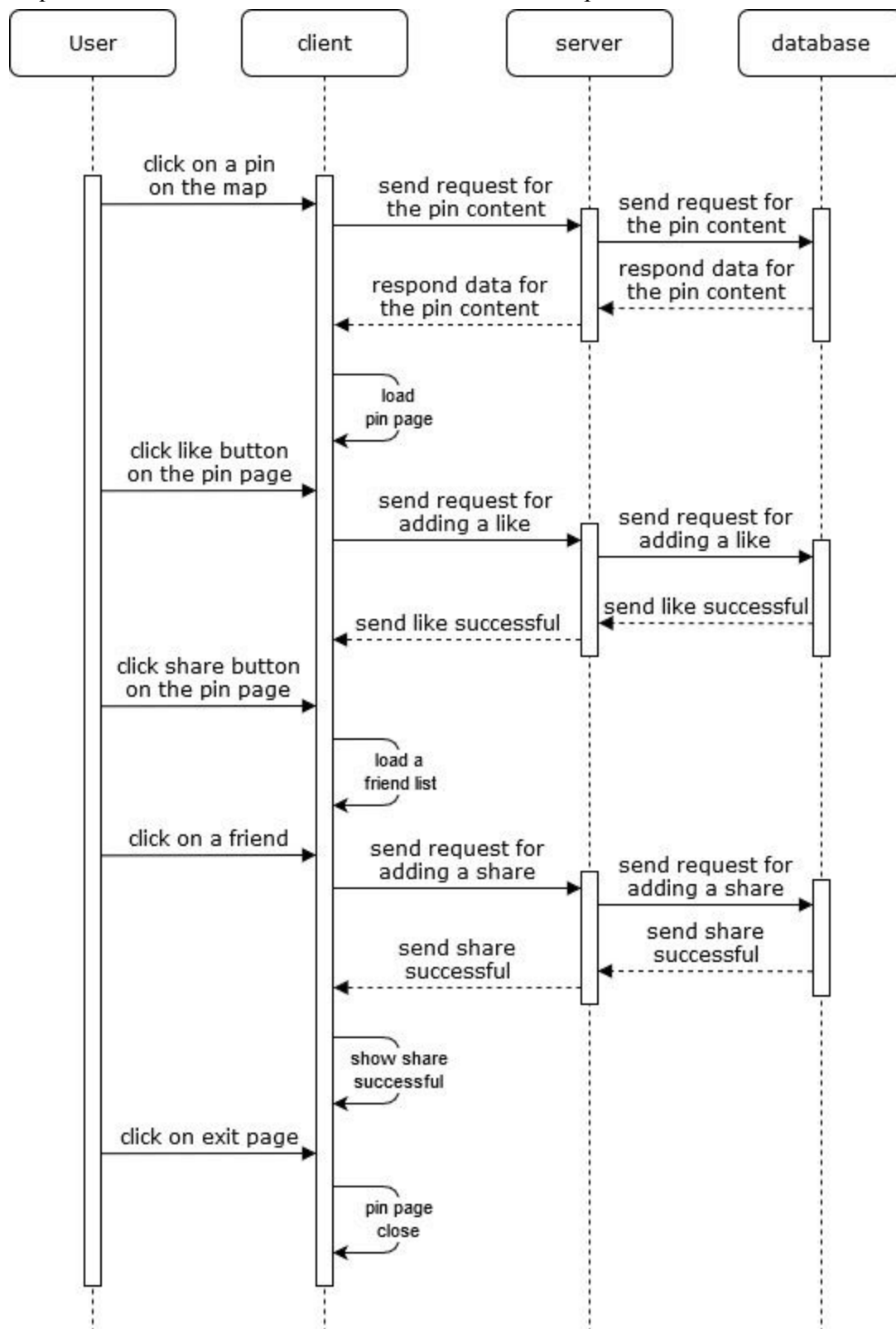
- A like is generated when the user hits the like button on a pin.
- User is the user who liked the pin.
- A user can either like or cancel previous like to a pin. A pin can have multiple likes.

Share:

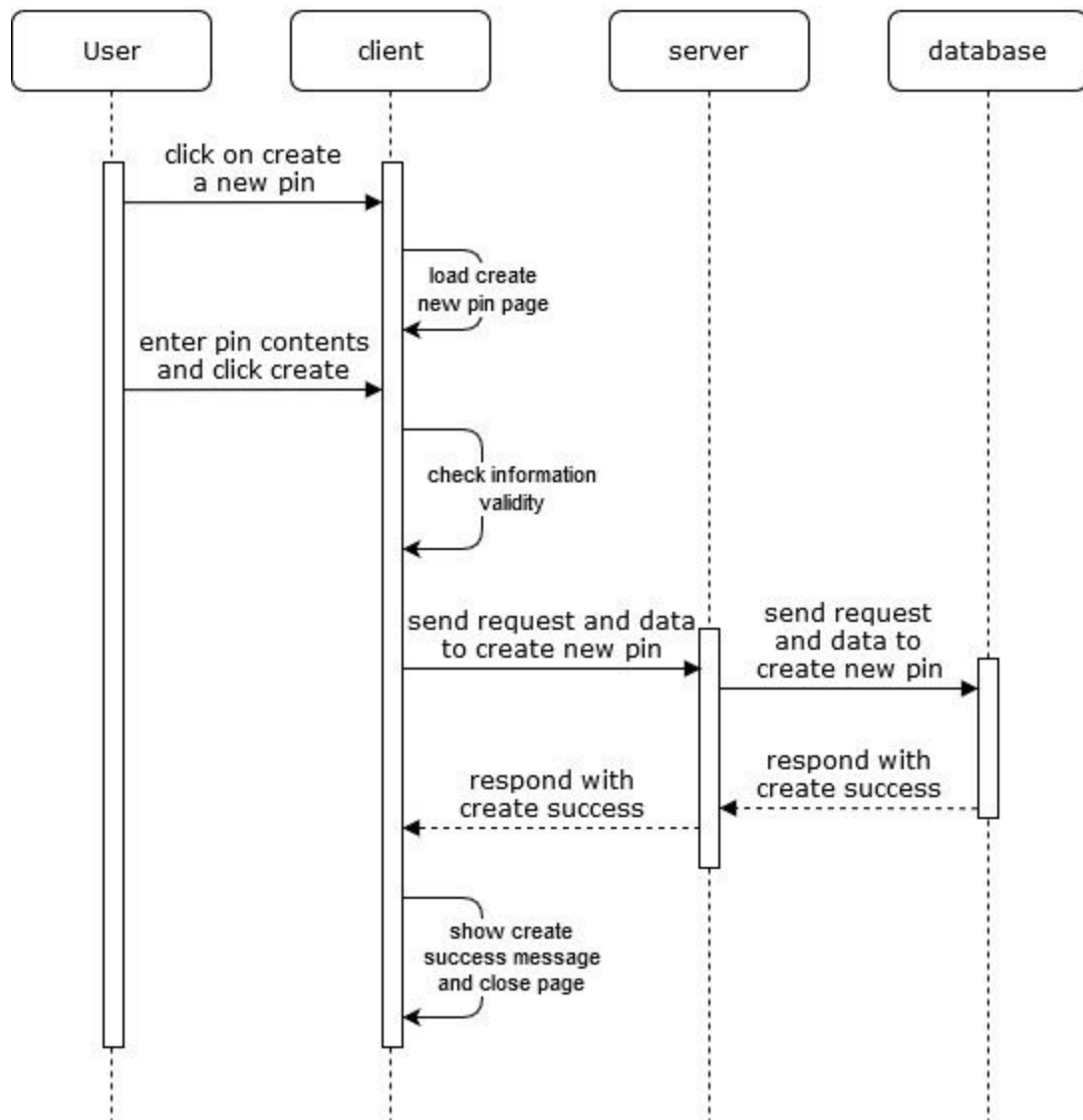
- A share is generated when the user hits the share button on a pin.
- User is the user who shared the pin.
- For a specific user, only one share is counted and it is not reversible. A pin can have multiple shares.

Sequence diagrams for different activities in the system:**1. Sequence of events when users register**

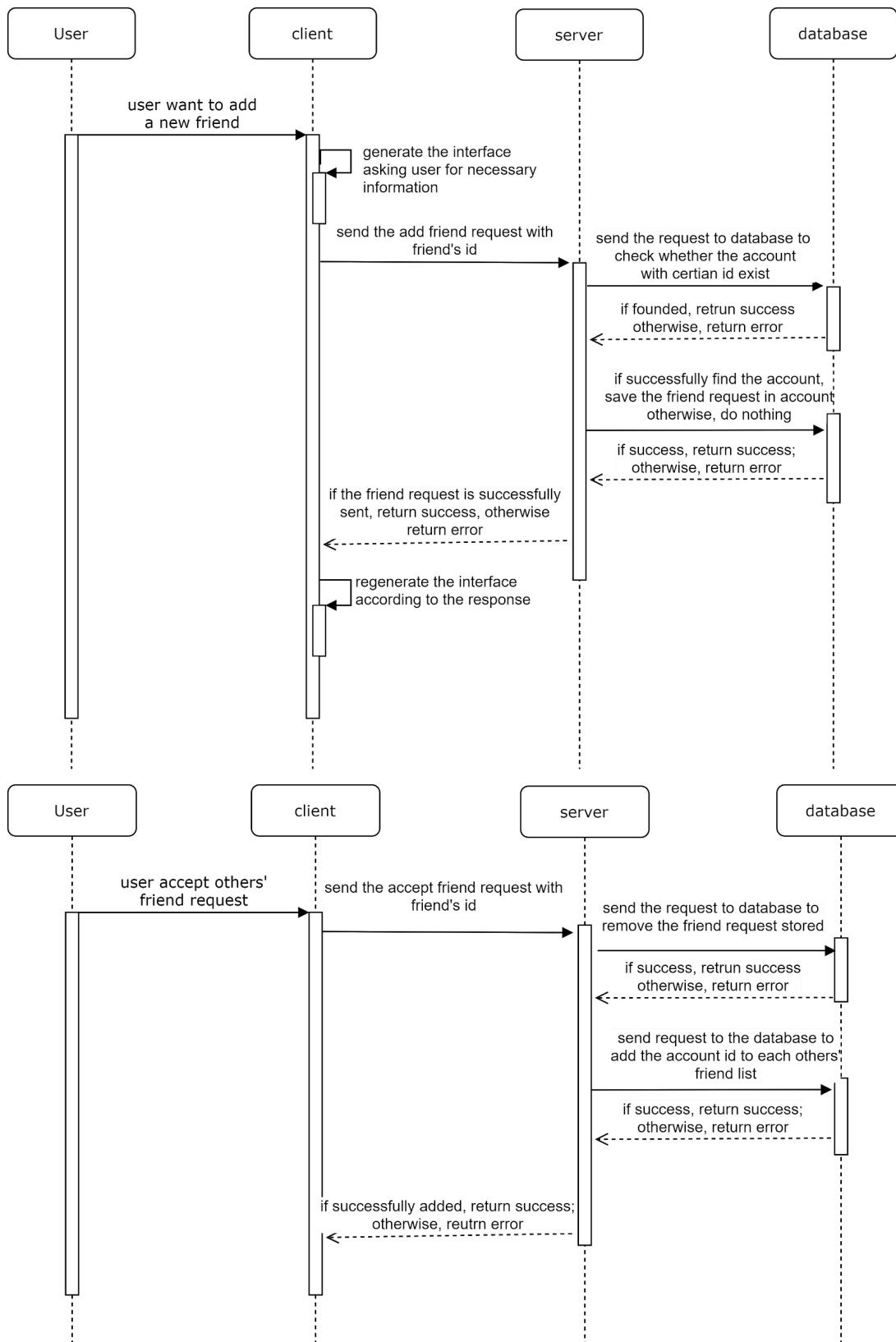
2. Sequence of events when users view and interact with pins:

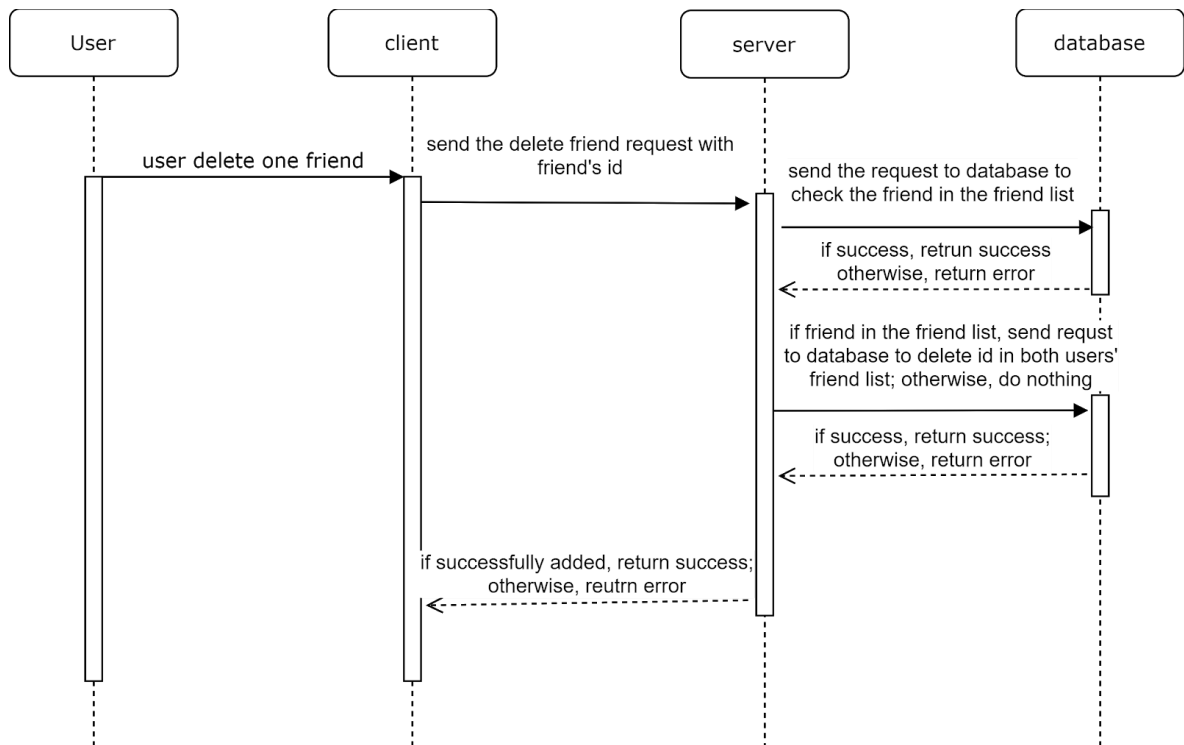


3. Sequence of events when users create a new pin:



4. Sequence of events when users add and delete friend:

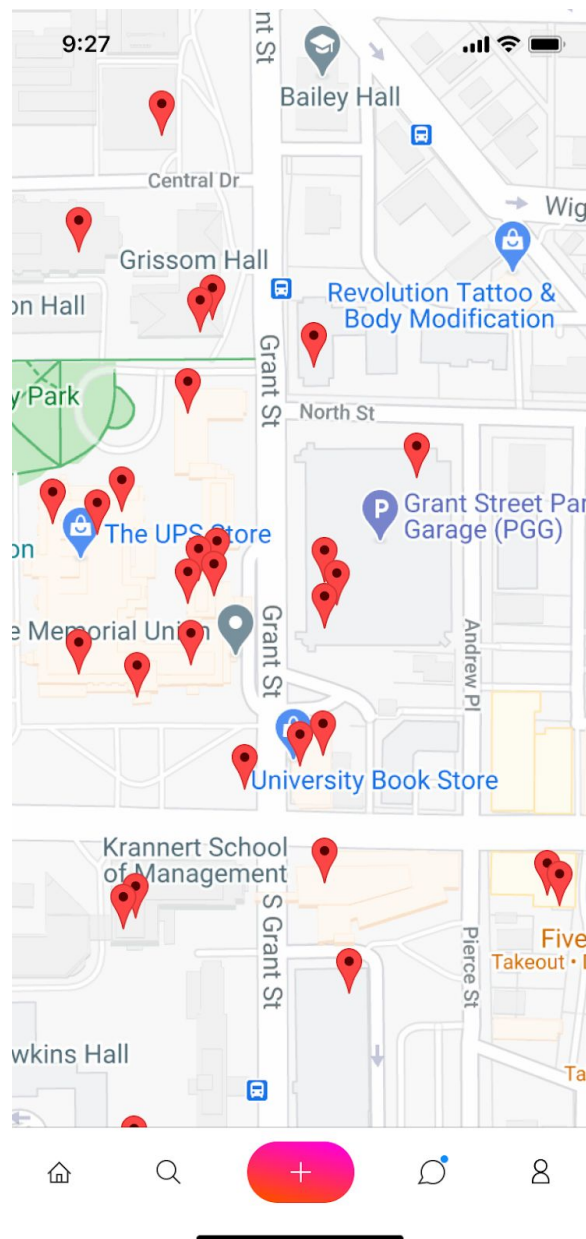




UI mockup:**1. Home view.**

This is the main page the user will be on. Here, you can see many pins on the map. Note these pins all in red, meaning they are public; friend-only and private pins are not shown here. Users can freely browse and search the public pins on the map without logged in.

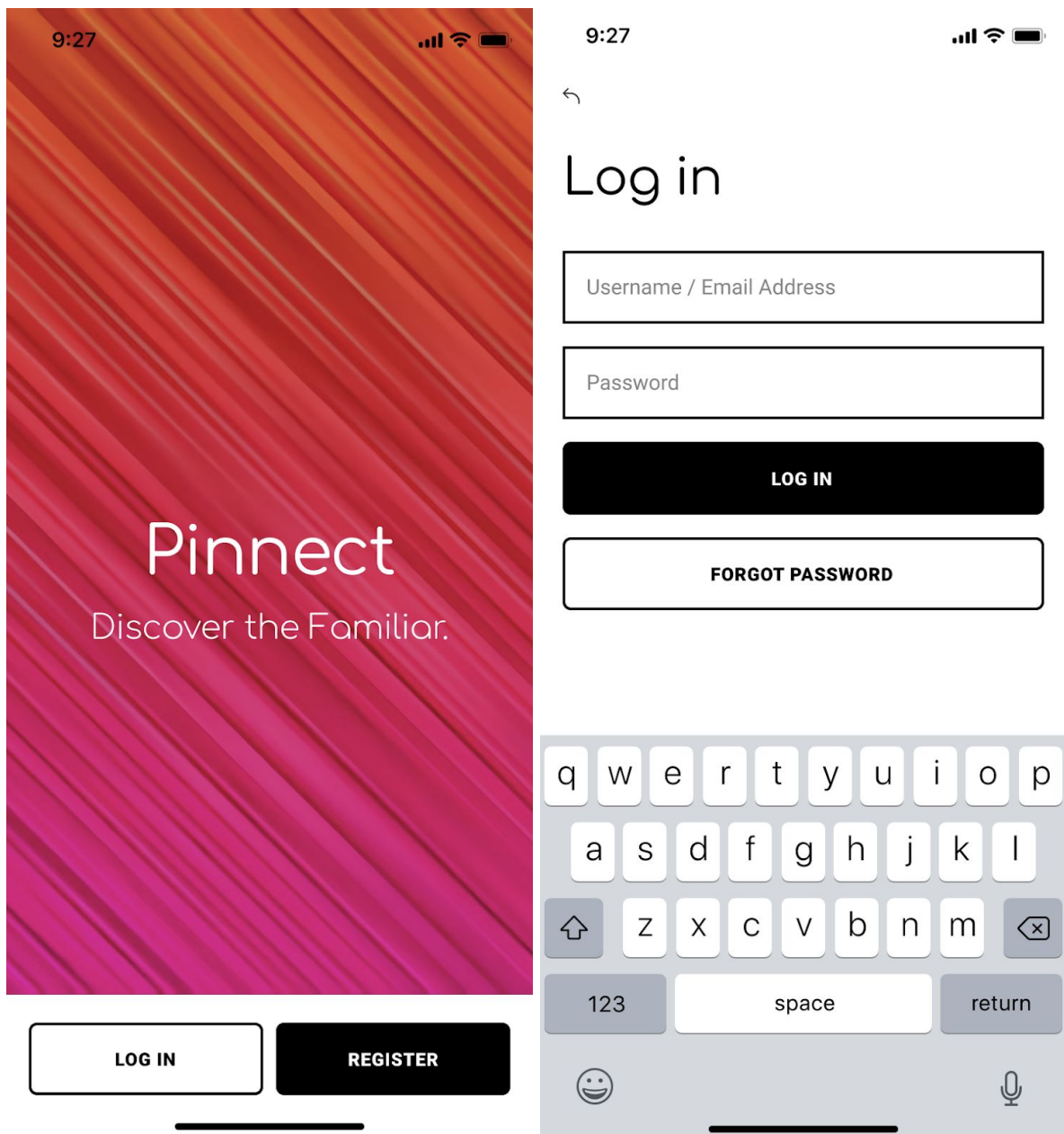
If users have interest in unlocking other functionalities like interacting with people, sending messages to friends, or interacting with private pins, users can click any corresponding buttons below and will be redirected to the next view.



2. Login/Register View.




A user can choose to log into an existing account or create a new account. The keyboard will be automatically triggered, and the user needs to enter either his/her username or the email address associated with his/her account to log in.

The *Log In* button will start an authentication and redirect the user to the next view. The *Forgot Password* button is also provided in case the user loses access to their account. The *back* button indicated by the back arrow redirects the user back to the main screen.



(cont.)

(cont'd from last page)

9:27   




←

Register

Username

Password

NEXT

9:27   

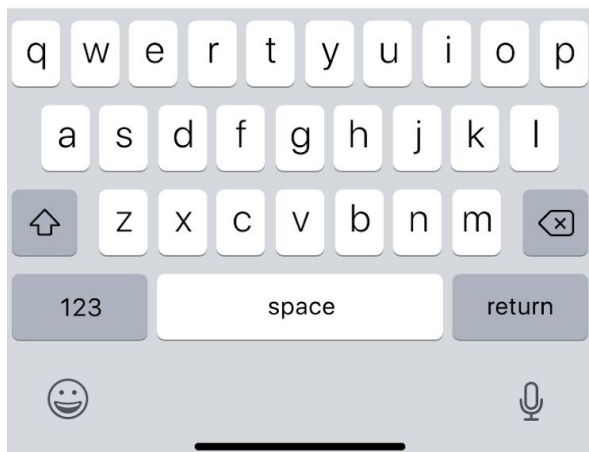
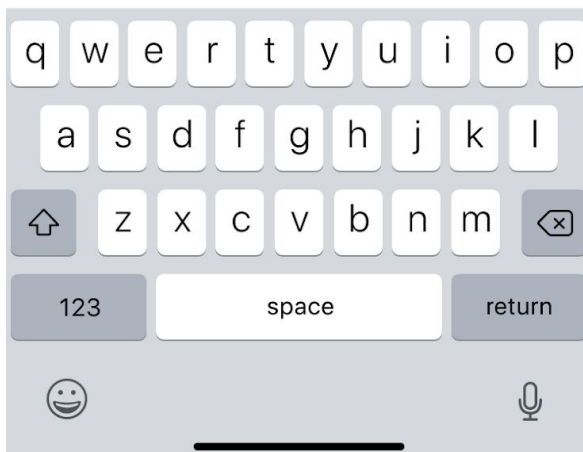
Register

example@email.com (Optional)

Connecting an email address to your account allows you to recover your account when you lose access to it.

SIGN UP

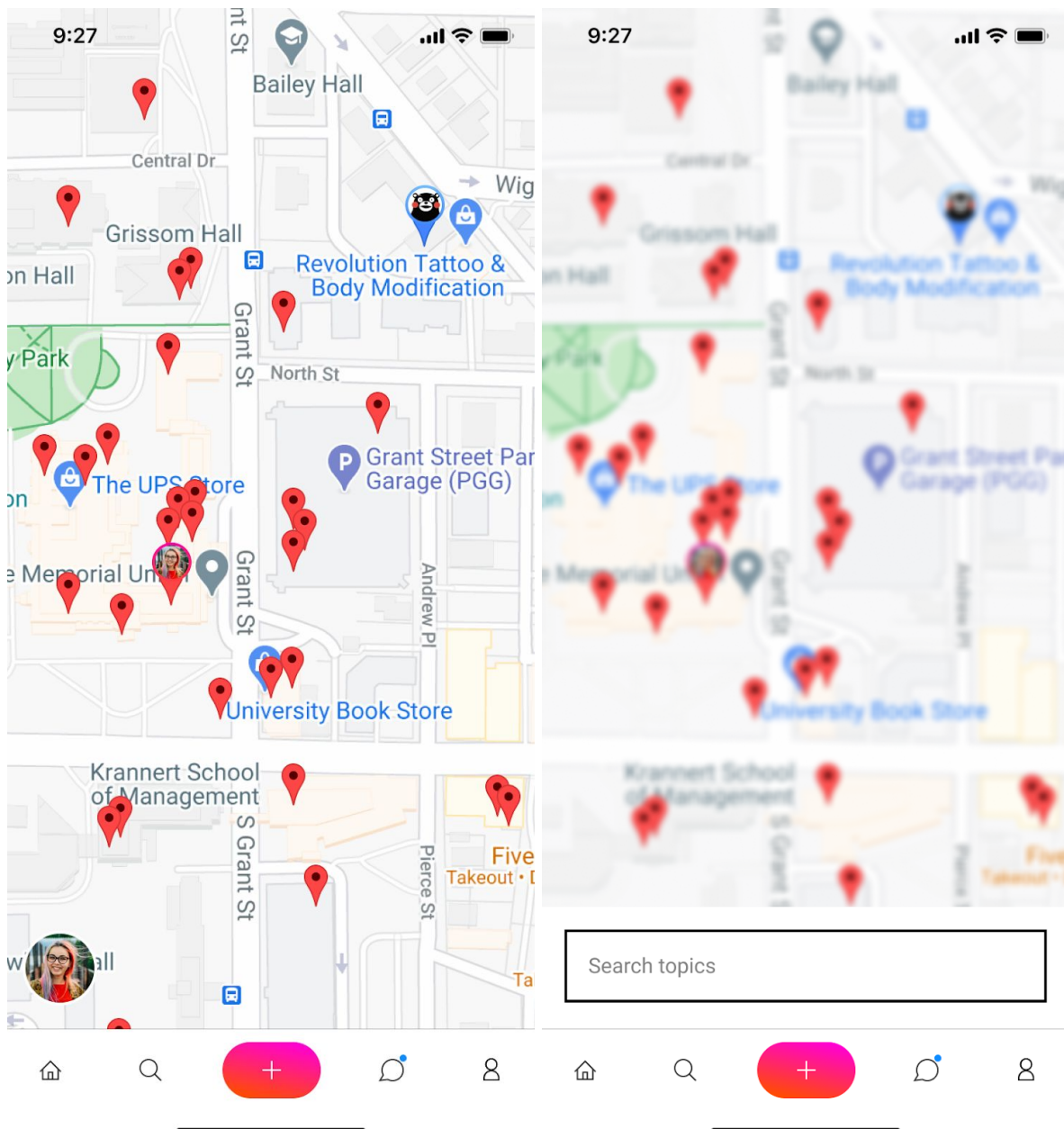
By signing up, you agree to Pinnect's [Terms of Service](#) and [Privacy Policy](#).



3. Home View (with all functionalities unlocked)

Once a user is authenticated, he/she can have access to all functionalities, as shown in the image. The mocked user is a girl named Lily whose avatar is displayed at the lower left corner of the screen. Here you can see two pins that were not in the logged out view:

- The first one is the blue pin at the upper right part of the screen. This is a friend-only pin - once Lily is logged in her account, she can see the pin.
- The second one is the red pin at the left center part of the screen. This is a pin published by Lily herself - and it is public, as the color suggests.

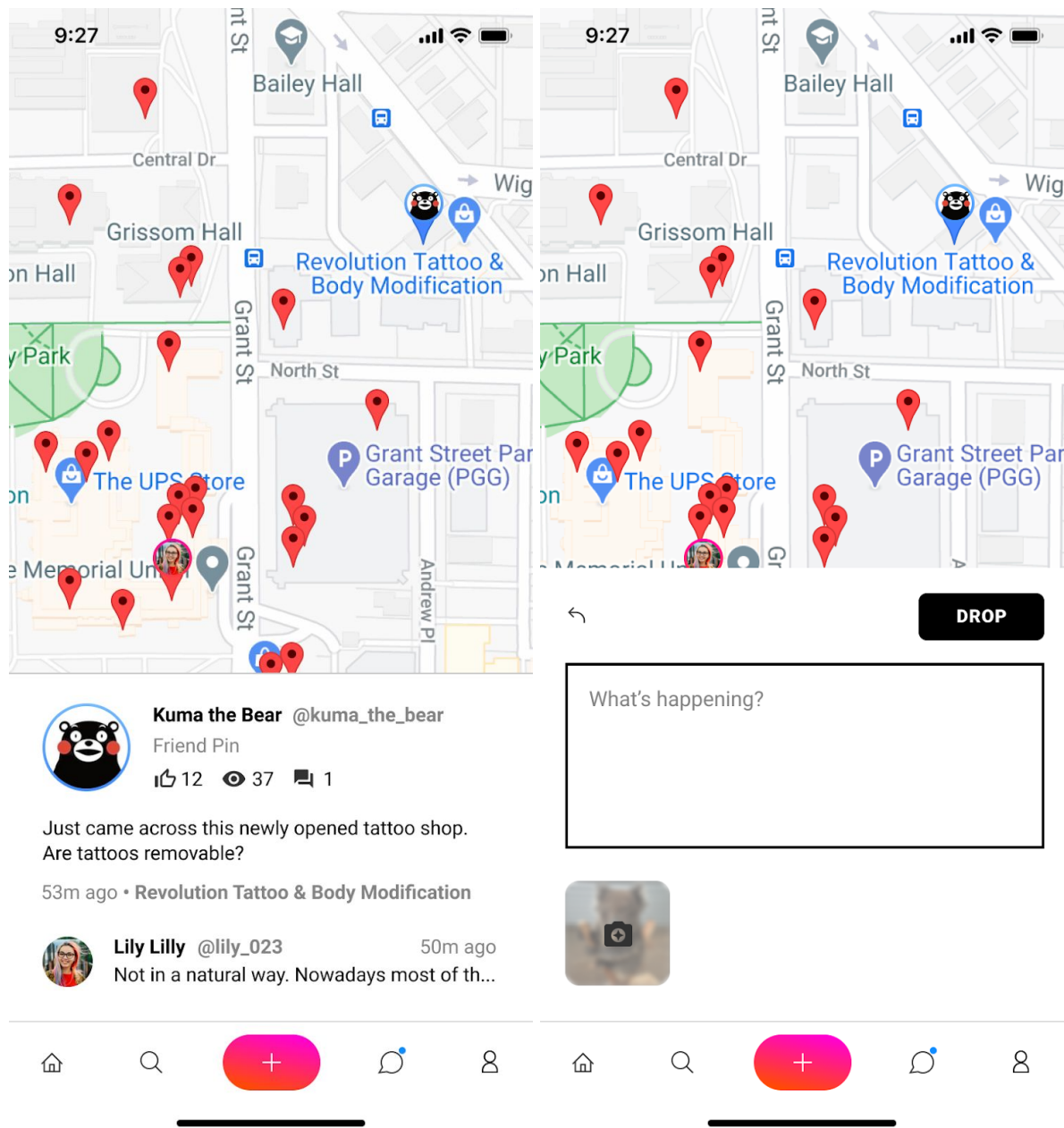


(cont.)

(cont'd from last page)

When Lily clicks on a pin, e.g., the blue pin, the detail content of the pin will be displayed at the lower part of the screen, and she can interact with the pin in various ways.

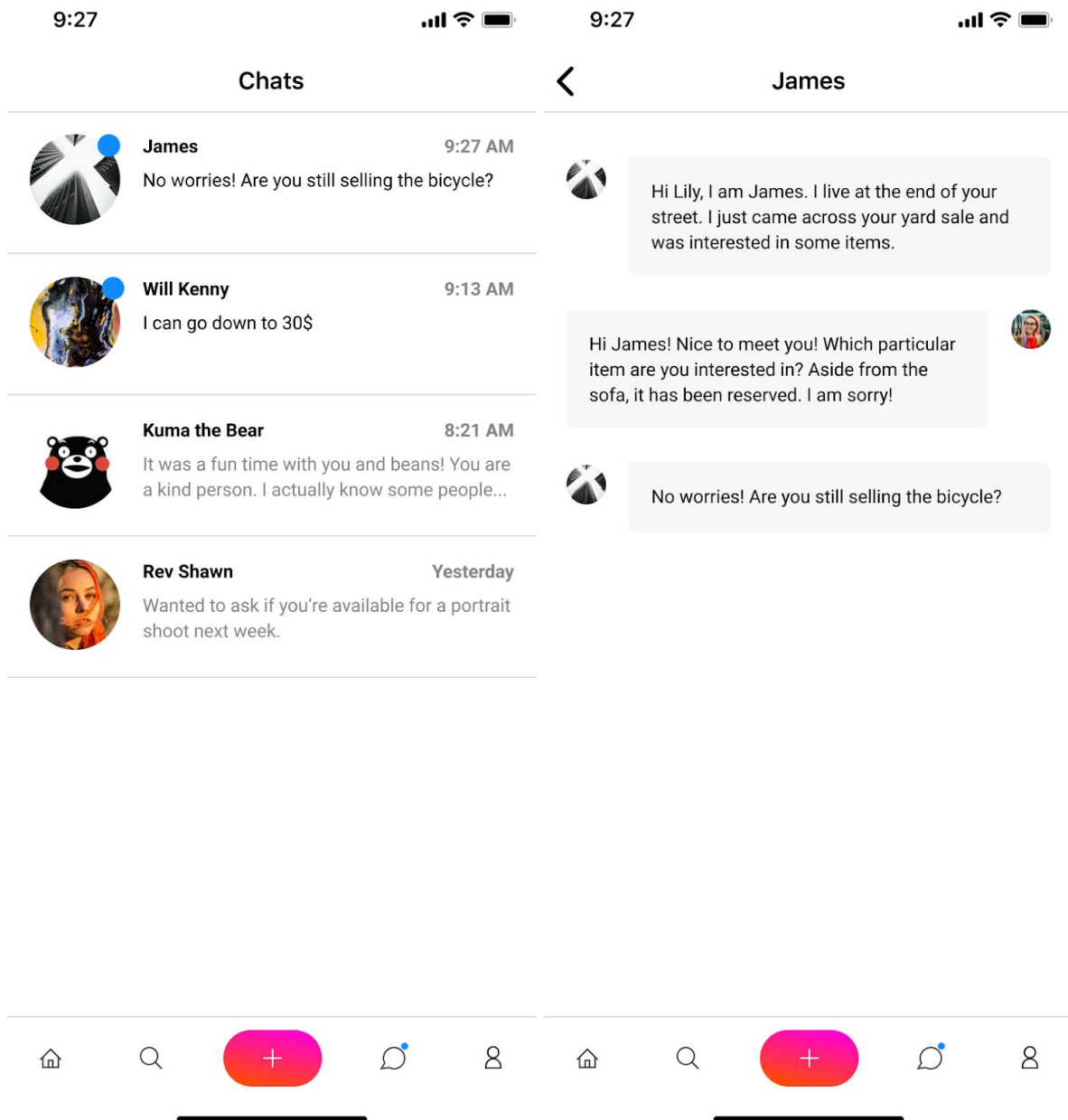
When lily clicks the “+” button at the bottom of the screen, an editor will be displayed where Lily can create a new pin at her current location, attached with the entered text and selected images.



4. Chat View.

When Lily clicks the chat bubble button at the bottom of the screen, she is redirected to the chat view where all her friends are listed. She can tap on a specific person and chat with the person individually.

The blue dots on avatars and the chat bubble button indicate there are unread messages.



5. Settings View

The last view is the settings, which can be entered by clicking the last button at the bottom tab. Lily will control the account information, manage generated contents, and adjust other settings on this page.

The user can also choose to log out of the current account by clicking the Log Out button below. When clicked, it will take the user to the first view introduced in the UI mockup sections.

