Fall 2015

CS 307

Team 6 Design Document

SuLife

Wentai Cui

Shayin Feng

Bangqin Lin

Fangzhou Lin

Chi Luo

Qi Zhang

Table Of Contents

Purpose

Currently, many calendar applications have been developed and used by a large amount of people. However, the functionality for those applications are quite restricted and too personal. Therefore, we designed a socialized calendar(SuLife) application which allows users to share events and make group activities easier. SuLife is a calendar with three main components. They are database, server and client.

The design document thoroughly describes our functional requirements.

1. Create a log-in page
    a. a user can created an account in this app.
    b. username and password match
        i. only if the username and password can match together, a user can log in.
        ii. when click the the login button, if the username doesn't exist, display a message: username not exist.
        iii. if password and username do not match, display a message: password or username not correct!

2. A user can create an event, and modify or view an existing event
    a. a user is able to delete an event while viewing it
    b. a user can invite friends to join the viewing event or meeting
    c. a user can view the location on map where an event takes place

3. Date of the server and front-end are synchronized
    a. the server and client can automatically synchronize list of events and calendars in a certain account
    b. the server is able to synchronize data on all devices for the same account

4. The system is able to send push notifications to users
    a. to remind users that a certain event or meeting is approaching
    b. to notify users that a new event in a group is created
    c. to notify users when receving a new invitation

5. Account creation page

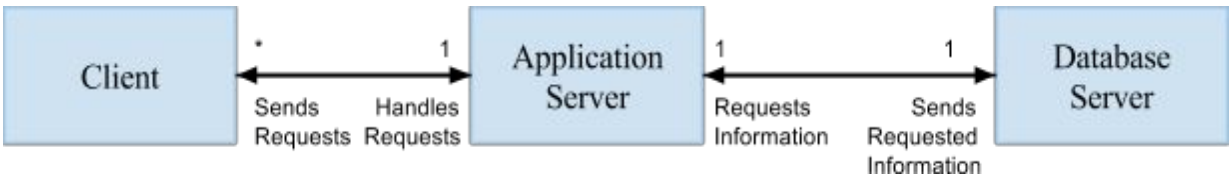
a. For the create account button, a new user must provide a unique username, password, and E-mail. If a username already exists, display a message: username has been used!

6. Profile page
    a. The profile page contains a head portray which users can upload photos if they want.
    b. The profile page contains personal information which can be seen by contacts.
    c. User can edit profile and save changes.

7. Personal calendar and socialized calendar
    a. For the personal calendar interface, it will show activities for a certain period of time.
    b. Once a user add an event, his/her personal calendar should be update.
    c. Once a user create a group, system will display a group page which contains the group infromation, members' head portraies, and option_buttons(create event, task, view event/task list, leave group, etc.)
    d. Once a user clicks "create event", all schedules of group members will be overlapped by the system, and the new schedule which contains the common free-time of all members will be displayed on the user's interface.
    e. After the free-time sechdule is built, all members in the group are available to view the schedule and choose a proporite time period for the event.
    f. Once the time period has been confirmed, the event will be created and can be viewed in the event list as well as the Calendar_View.
    g. Once a user clicks "create task" and add the proper information, the new task will be created and can be viewed in the task list as well as the Task_View.

Design Outline

Our project is an iOS application which allows users to create sharable calendars and interact with their friends in a group. It will use a client-server model and contain three main components: client(iOS application), an application server and a database server. Client and application server will interact with restful API. That is, client will send HTTP requests(post, get,etc.) to application server, while server will response in JSON format. We will use MongoDB for database and database communicates with server by query language.



Purpose of each component:

1. Client
Users will interact with the application and communicate with application server. The client will send requests to the server and displays the calendars, groups and maps clearly. The client will be implemented in Swift, Objective-C, C++.

2. Application Server
The server will be the connection between client and database. It will accept HTTP requests from the client and send HTTP response to the client. Calculation and data process will happen in the application server. The application server also interacts with database server by query language. It send information to and requested information from the database. The application server will be written in node.js.

3. Database Server
The database will store major data in our application including user information, event information, group information, calendar data, location data, etc. It will receive data and requests from application server and send requested information back to the application server. We will use MongoDB for database.

Design Issues

Issue 1: Which language should we use to implement frontend client

Option 1: Swift

Option 2: Objective-C

Decision: After comparing Swift and Object-C, we dicided to use Swift. There are two resons:

(1) Swift code more closely resembles natural English, in contrast, Objective-C is ugly duckling and hard to read.

(2) Swift provides a good balance of programmer productivity and performance. It has deterministic memory management and RAII support while at the same time supporting concise and clear logic more like dynamic languages like Javascript or Python. In most cases application code is not so performance sensitive as system code and Swift provides efficiency and leanness in a prettier and more maintainable language than other choices.

Issue 2: Which database should we used

Option 1 : MySQL

Option 2 : MongoDB

Decision: After comparing the MySQL and MongoDB, we dicided to use MongoDB. There are two resons:

(1) It enables us to build applications faster, handle highly diverse data types, and manage applications more efficiently at scale.

(2) Using MongoDB removes the complex object-relational mapping (ORM) layer that translates objects in code to relational tables.

Issue 3: Map API decision

Option 1: Apple map

Option 2: Google

Option 3: Yahoo

Option 4: Bing

Decision:

We decided to use Apple map as the API decision. Since:

(1) Apple Map is easier to implement.

(2) Apple Map offers better integration into iOS.

(3) Apple Map is preinstalled in every iOS device.

Issue 4: Which language should we use to implement backend server

Option 1: Java

Option 2: Ruby

Option 3: Go

Option 4: Node.js

Decision: We decided to use Node.js. The main reason why we chose Node.js is that Node.js efficently handles asynchronous input and output. Node.js is a kind of functional programming, it provides Event Loop. Even though we have a large volumes of clients, all of the I/O intensive operations can perform asynchronously.

Issue 5: How should calendar page be displayed?

Option 1: Display the month calendar and mark events on certain dates.

Option 2: Display the timeline of the a day and mark events on a period of time.

Decision: Option 2 is the way to go. Our application will be on iOS platform and is mainly designed for iPhone. Therefore, the size of screen will be a limitation. Displaying the time of the current day will allow users to see the events more clearly and allow more details to be showed. If we chose Option 1, the calendar will be too small and events information will be blurred.

Issue 6: How should the contact list be sorted?

Option 1: Sort the contact list in alphabetic order.

Option 2: Sort the contact list by the level of friendliness.

Desicion:

We decided to choose Option 2. Since:

(1) It's the fastest way to find the closest friend.

(2) It's easier to set up a small group for best friends.

Issue 7: How should users access the location information?

Option 1: Click an event, then click "location" button on the event page to see the location on map.

Option 2: Add a map tab on the bottom. When the user hit this tab, the Calendar_View will switch to the Map_View and all locations of the events in a day will show up as highligted spots on the map and are marked in order.

Decision: Option 2 is chosen for now. First, we would like to implement some different features from the existing applications which do not have such map overview features. Second, it would be easier for users to schedule and plan for a day if they can

have an overview of all places and identify the first location they would go. Option 1 is also useful but Option 2 is better for a scheduling purpose.

Issue 8: How should user associates with friends through Calendar?

Option 1: Click an event page and find the person who invites you to the event and send message and email to the person.

Option 2: Add contacts to the contact list and send request to invite one or more people to create a group. Then user can create and share group events whenever they want by sending request and accepting request.

Decision: Option 2 is chosen for now. First, the feature of group_making is differennt from the existing calendar applications which only share events to one person at a time. Furthermore, it is easier for people to update group activities. All members in the group can catch up with the changes of events whenever they want.

Issue 9: How should a user acknowledge the group members current status towards a certain event?(eg. already arrived, 10 minutes till arriving, get stuck on the University Street)

Option 1: Send message in a group chat.

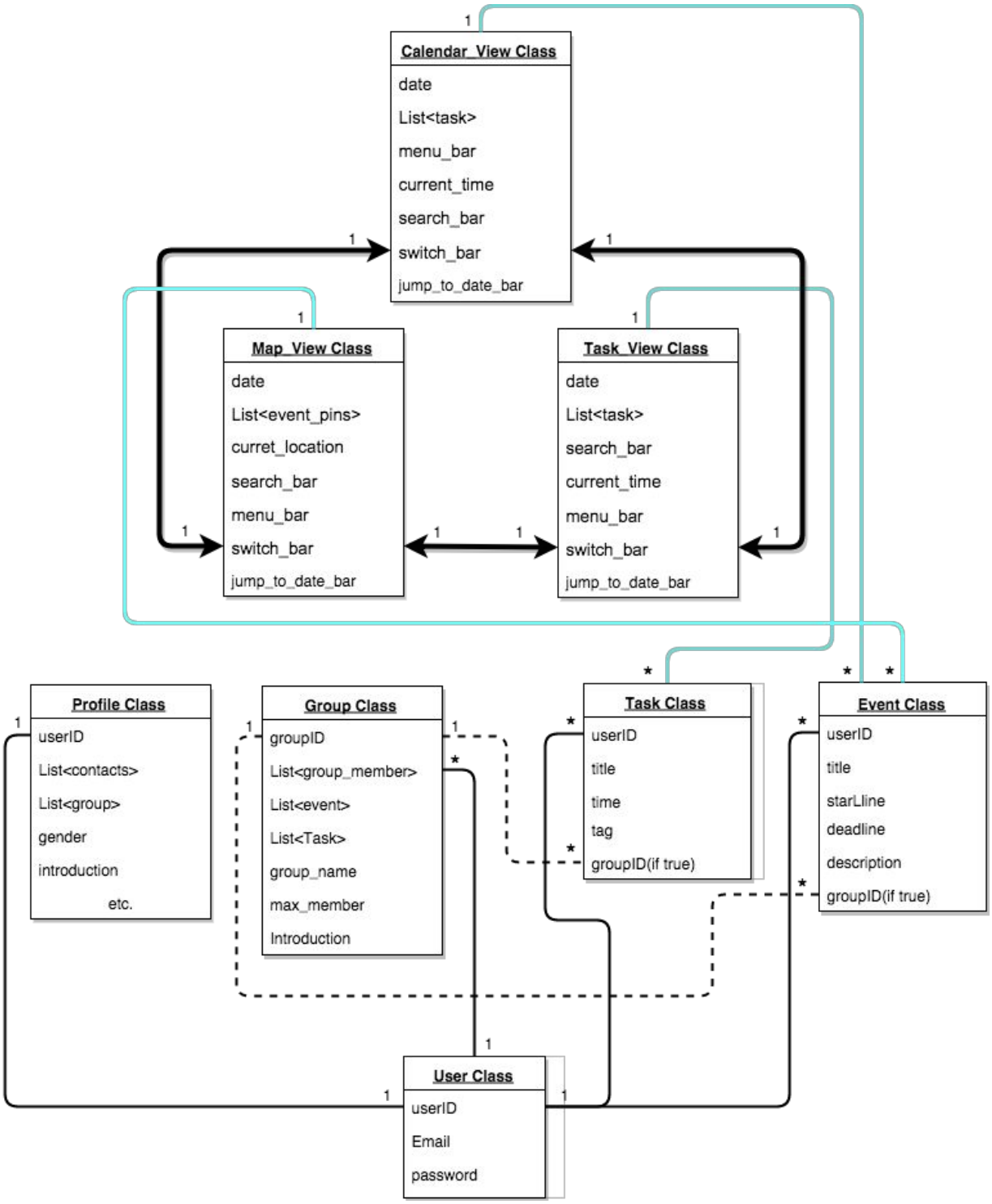
Option 2: Develop a functionality that allows users to set up their status and see all status of group members. A timer will count down automatically in terms of the status users set.

Decision: We choose Option 2 for this issue. Option 1 is commonly used in other applications. However, when we have many members in a group, the message will disappear on the screen very soon if people are chatting. Therefore, users need to take time to look for the relavant message to know the status of each member. To solve this problem, we choose the second option which enables user to see the status of everyone very conveniently.

Design Detail

System Interaction

a. Classes Design



b. Description of classes and interactions:

User Class

A class which records the personal identity information(email, password and userID).

Profile Class

A class which records personal information(name, group_list, contacts_list gender, etc) associated with userID.

Group Class

A class which records group information. It contains a user_list(for group member), a event_list and group information(name, max_member, brief introduction, etc.).

Event Class

A class which records event_information. It contains userID, title, time boundary, region and introduction. It will also associate with a groupID if it is a group event.

Task Class

A class which records task_information. It contains userID, title, time, and tags. It will also associate with a groupID if it is a group task.

Calendar_View

An entity which contains a view of timetable, event blocks and buttons (switch to other two main views and contacts, and for actions (add, jump, search, etc.))

Task_View

An entity which contains a view of timetable, tasklist at a certain time and buttons (switch to other two main views and contacts, and for actions (add, jump, search, etc.))

Map_View

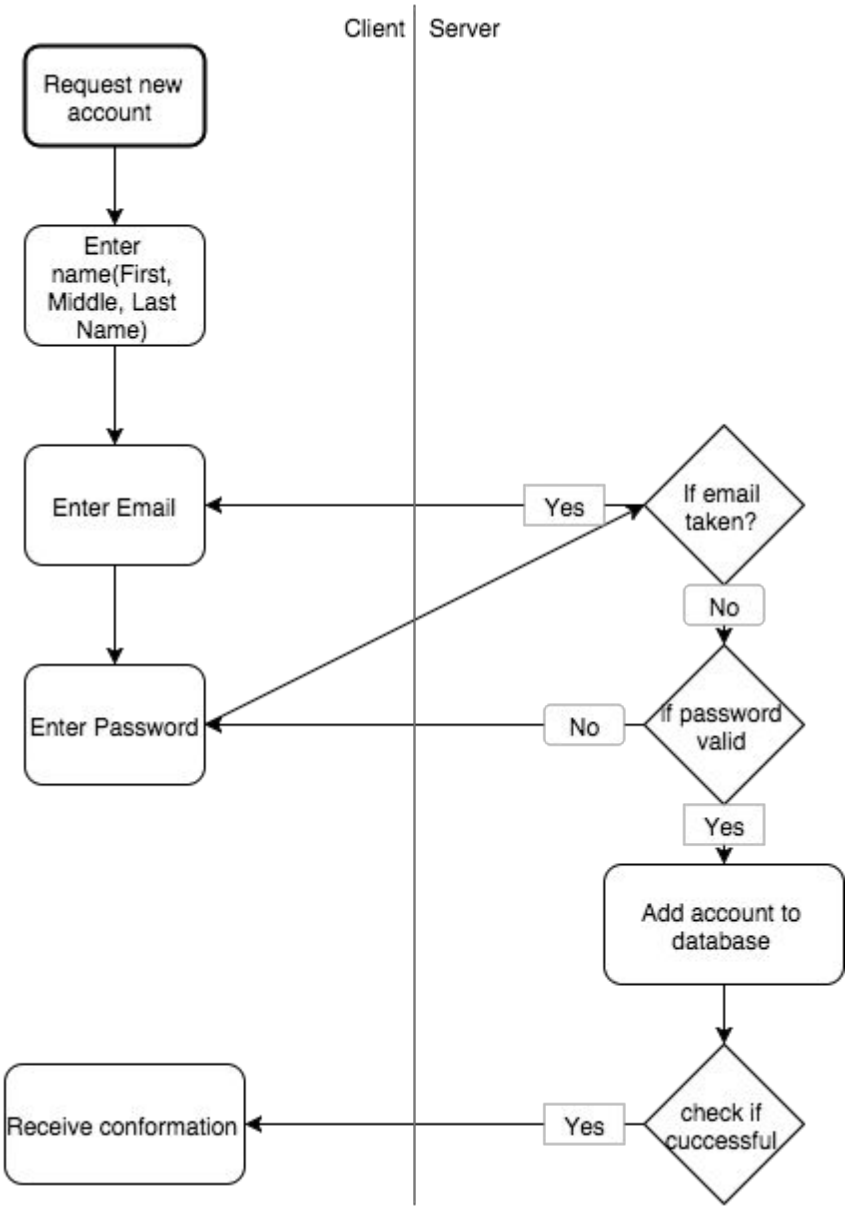
An entity which contains a view of map, location_pins, currrent location and buttons (switch to other two main views and contacts, and for actions (add, jump, search, etc.))

Classes interactions:

Create a new account

To create a new account, enter a correct email address and a password with the basic workflow below.

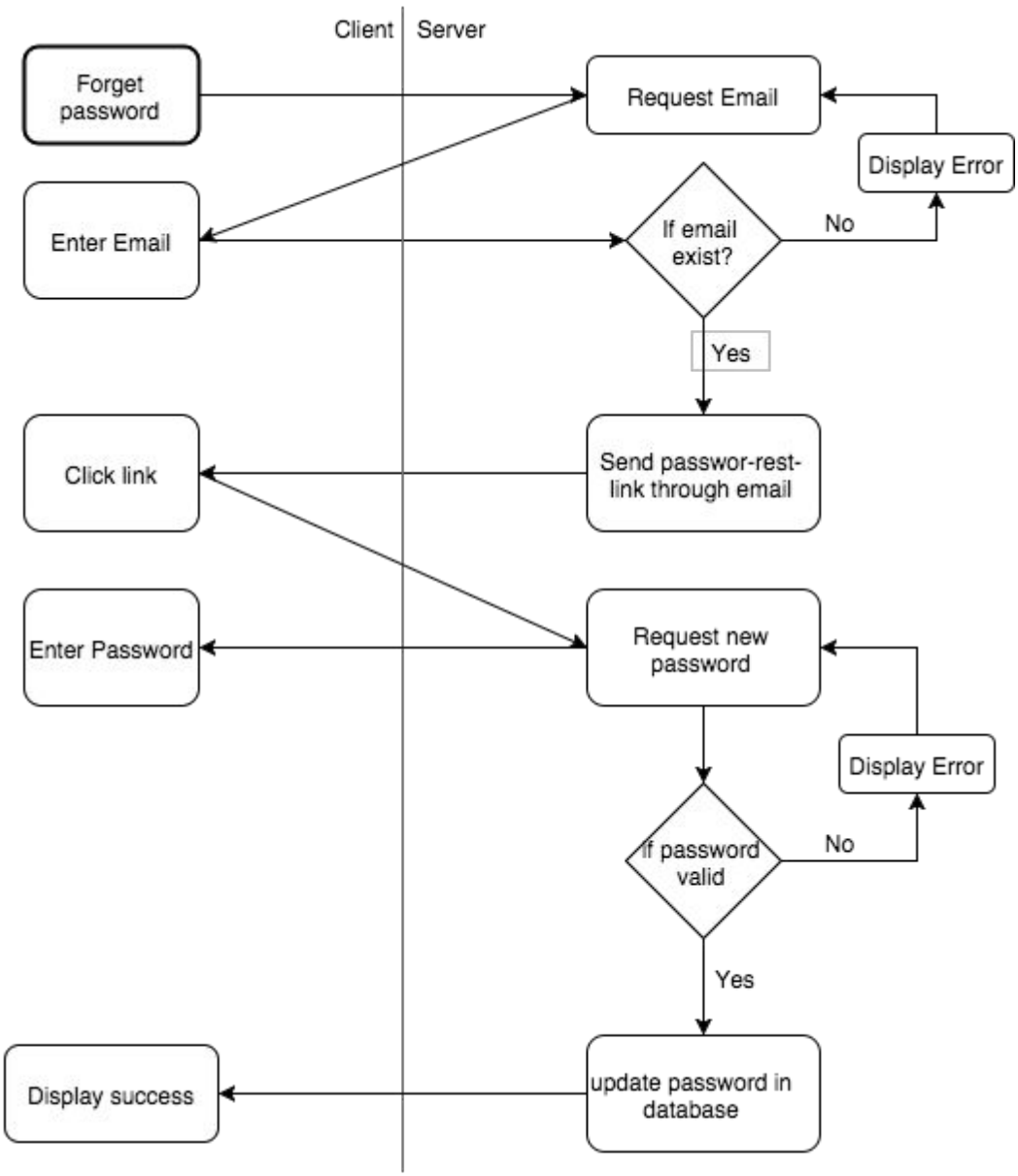
The new user will be add into user_list.



Reset password

If a user forgets his password, he must provide the email address in order to confirm his identity and use the link received by the email to change the password. The server will update the data in database.

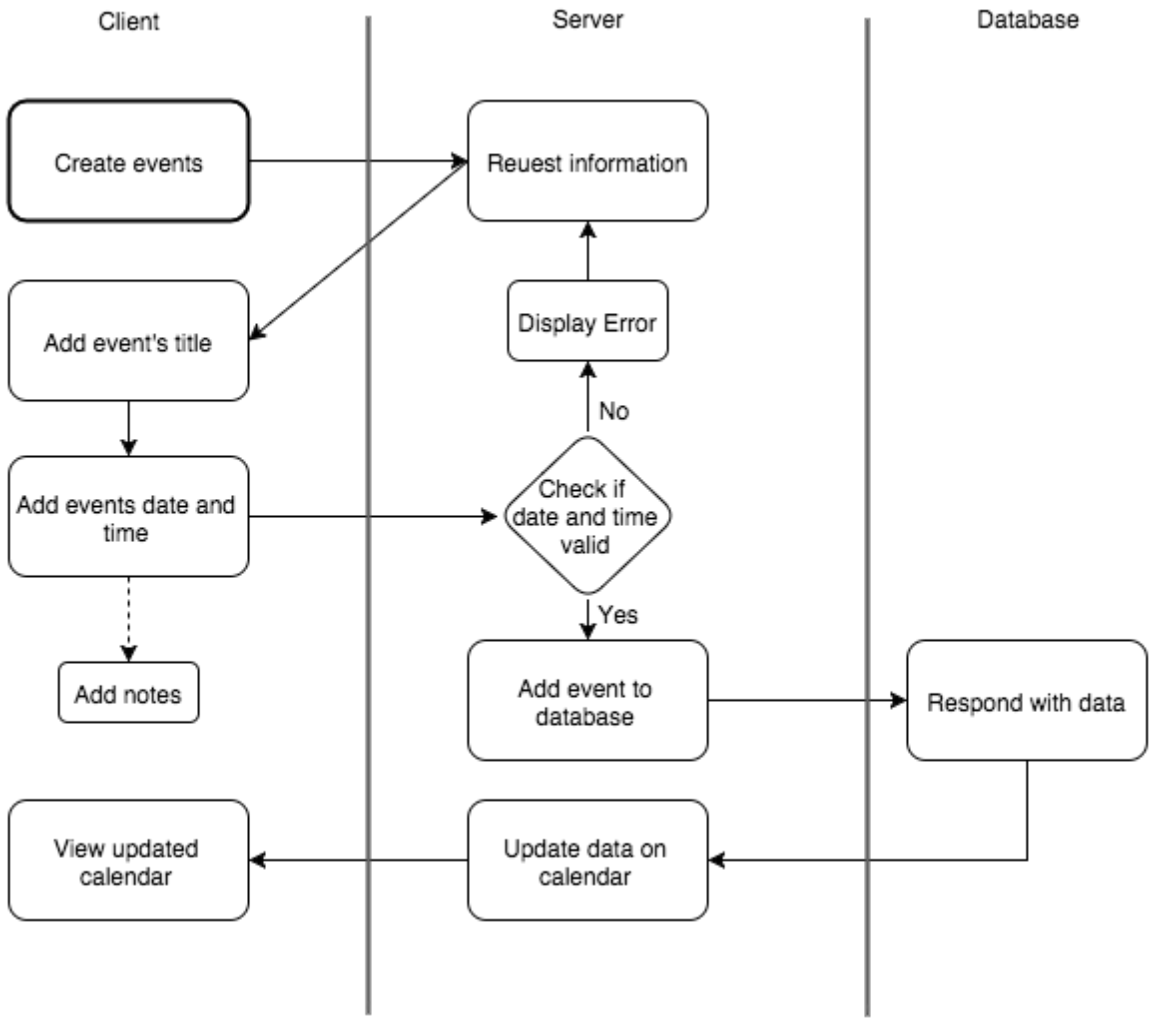
The workflow is shown below.



Create events/tasks

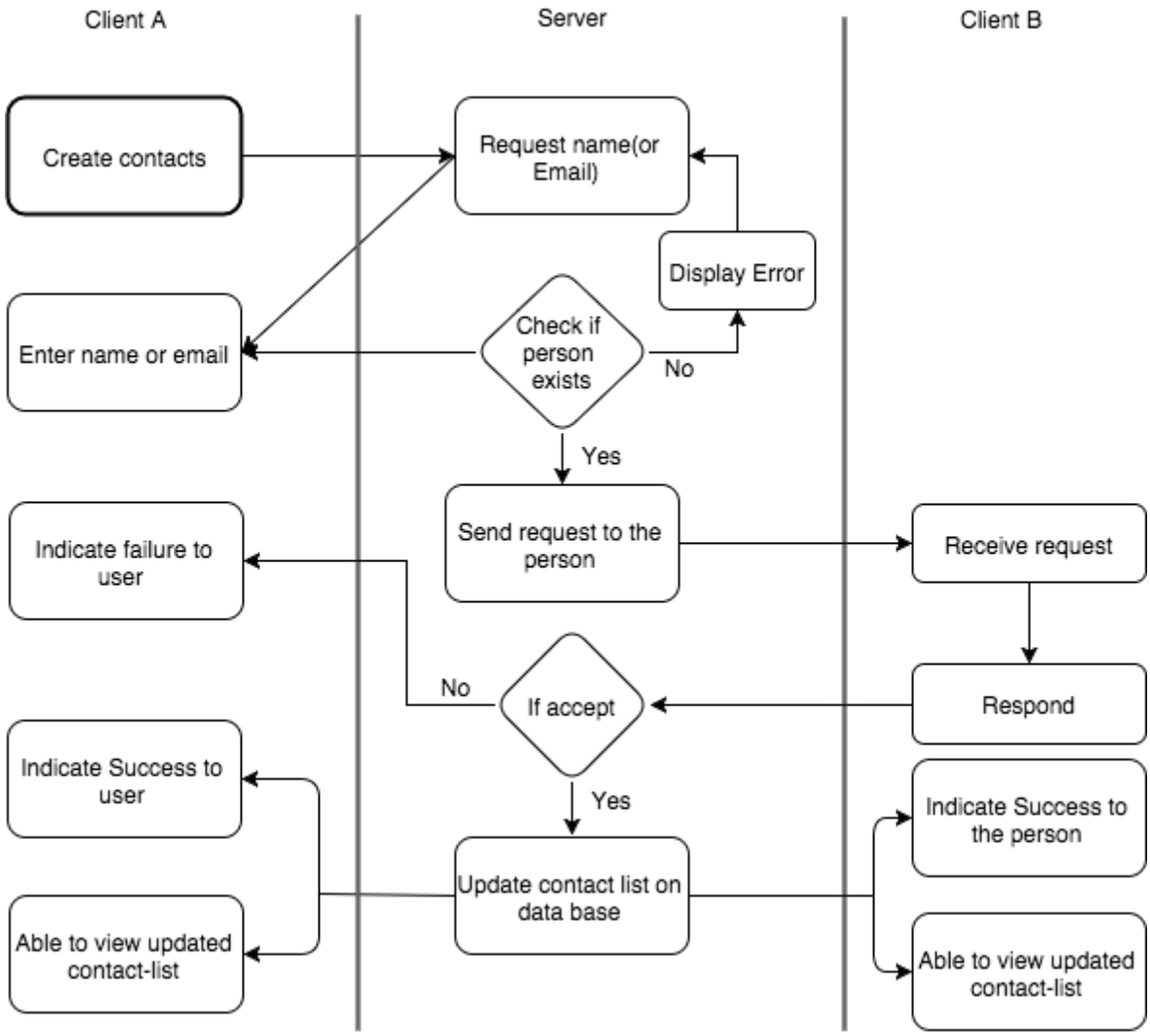
The events and tasks will be added to the event_list and task_list. The event and class are pointed by the user. And the result will be displayed on the calendar_view(if it is a task, it will be displayed on the task_view) as well as map_view.

The workflow is shown below.



Add contacts

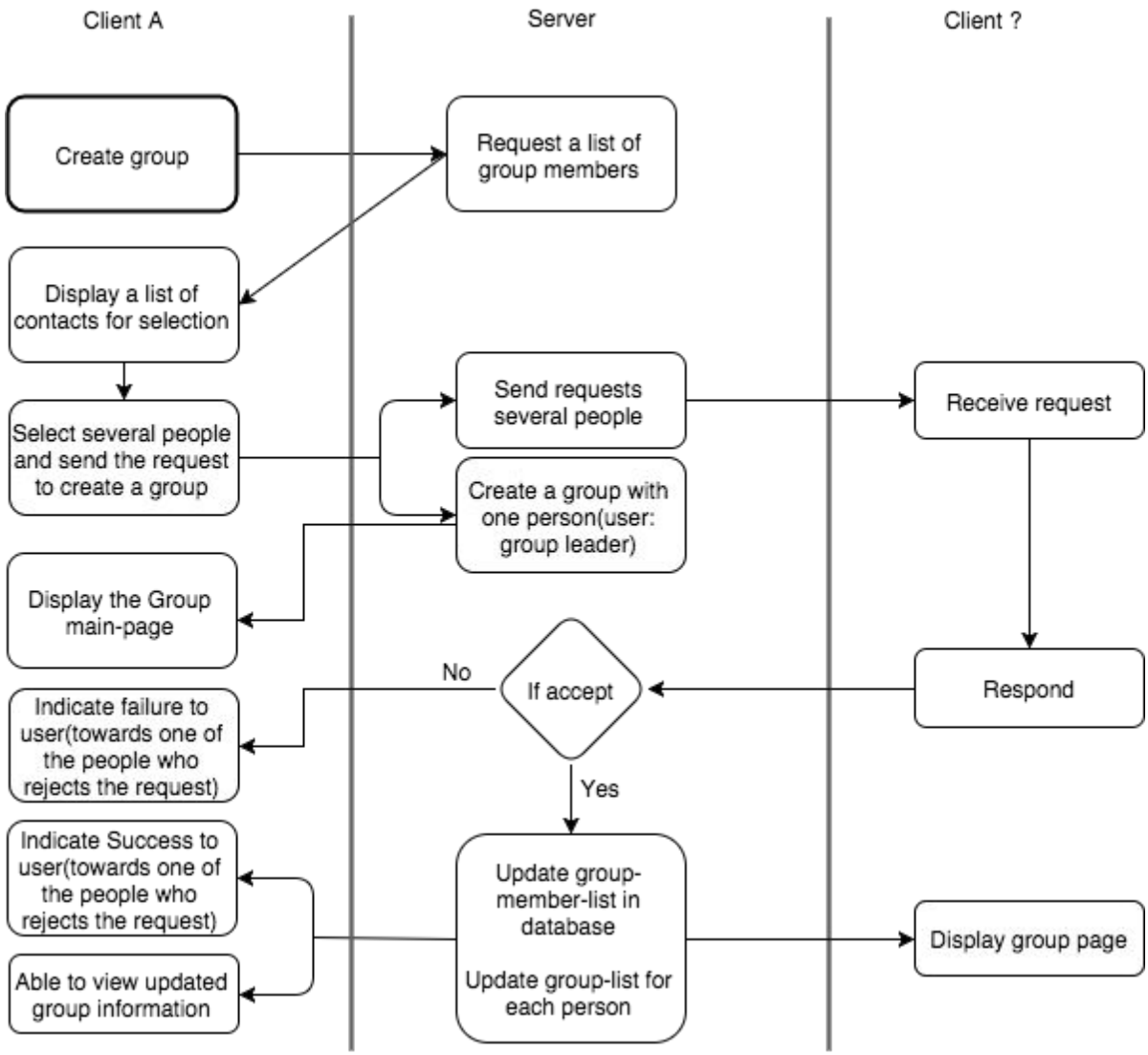
Adding a contact can be done by entering the email or name of a person. The system will check if the person exists or not. If true, the system will send the request to the person and wait for the reply. After the person accepts this request, the contact_list will be updated. And user can view the success_message and new contact_list right away. If the person rejects this request, the user will receive the fail_message.
The workflow is shown below.



Create Groups

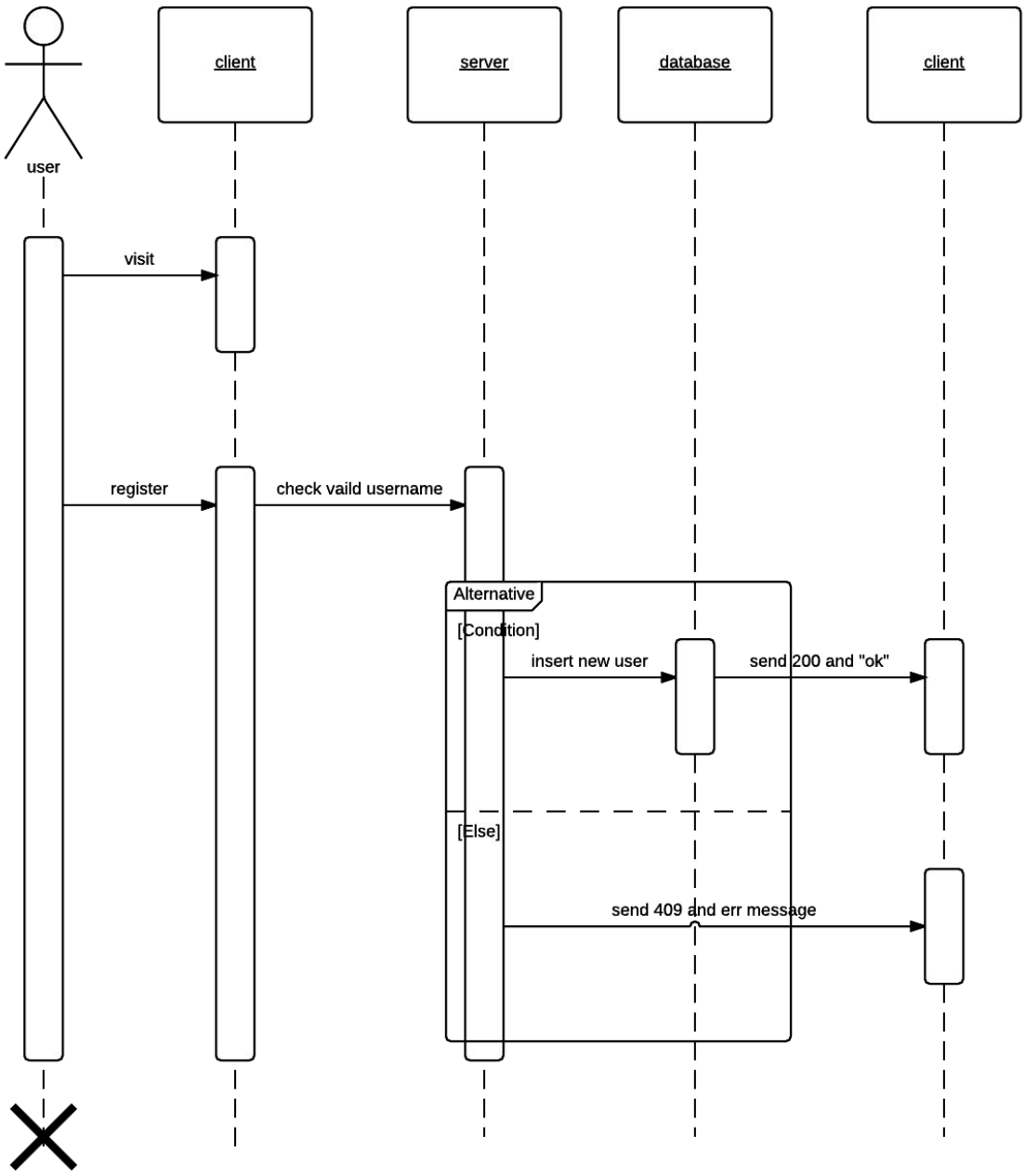
To create a group, system will get the contact_list. Every member selected by the user will receive a request. Group_member_list of a group will be updated whenever a person accept the request. And the group will be added to the group_list of each member in the group.

The workflow is shown as below.

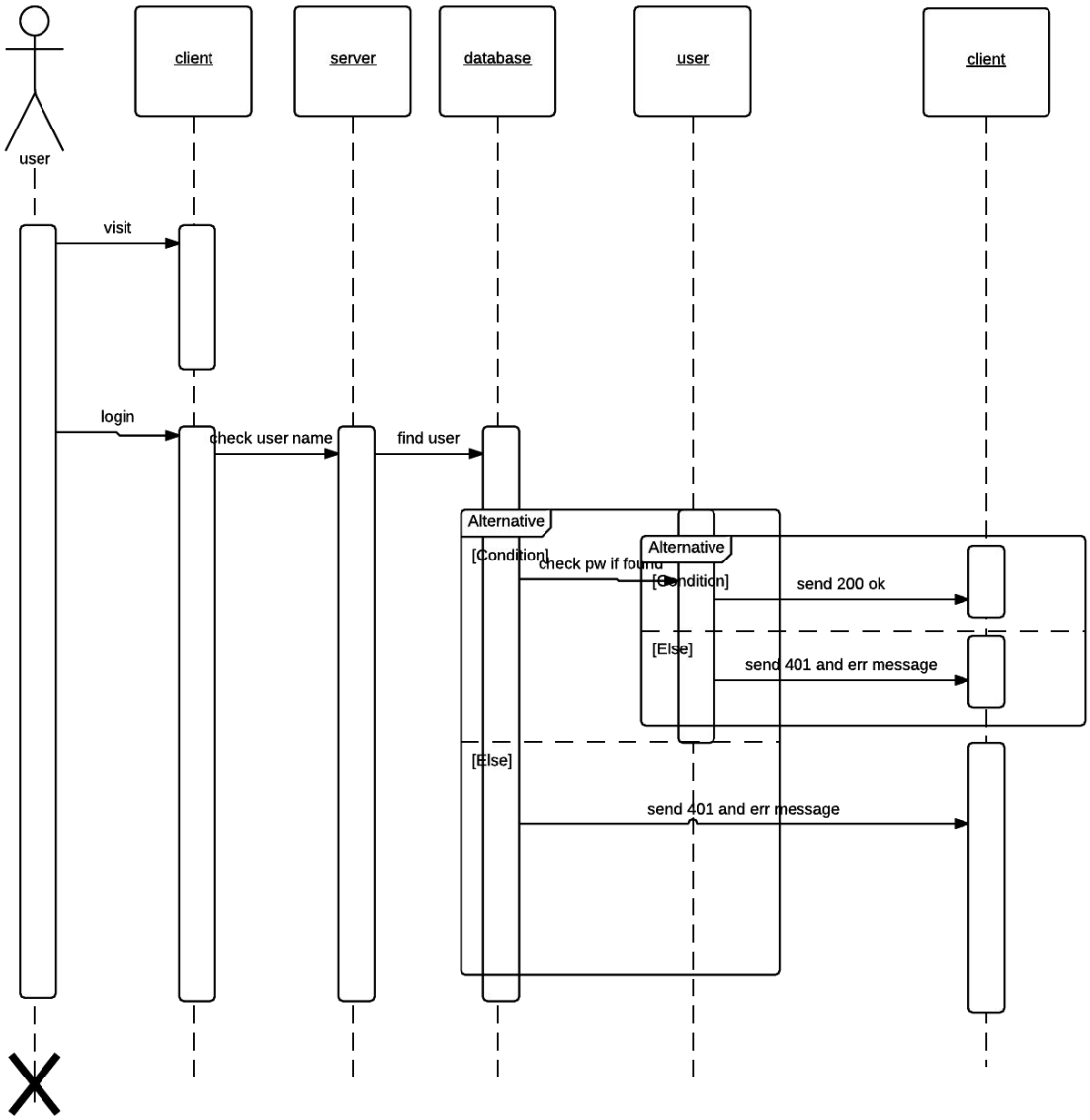


c. Sequence diagrams:

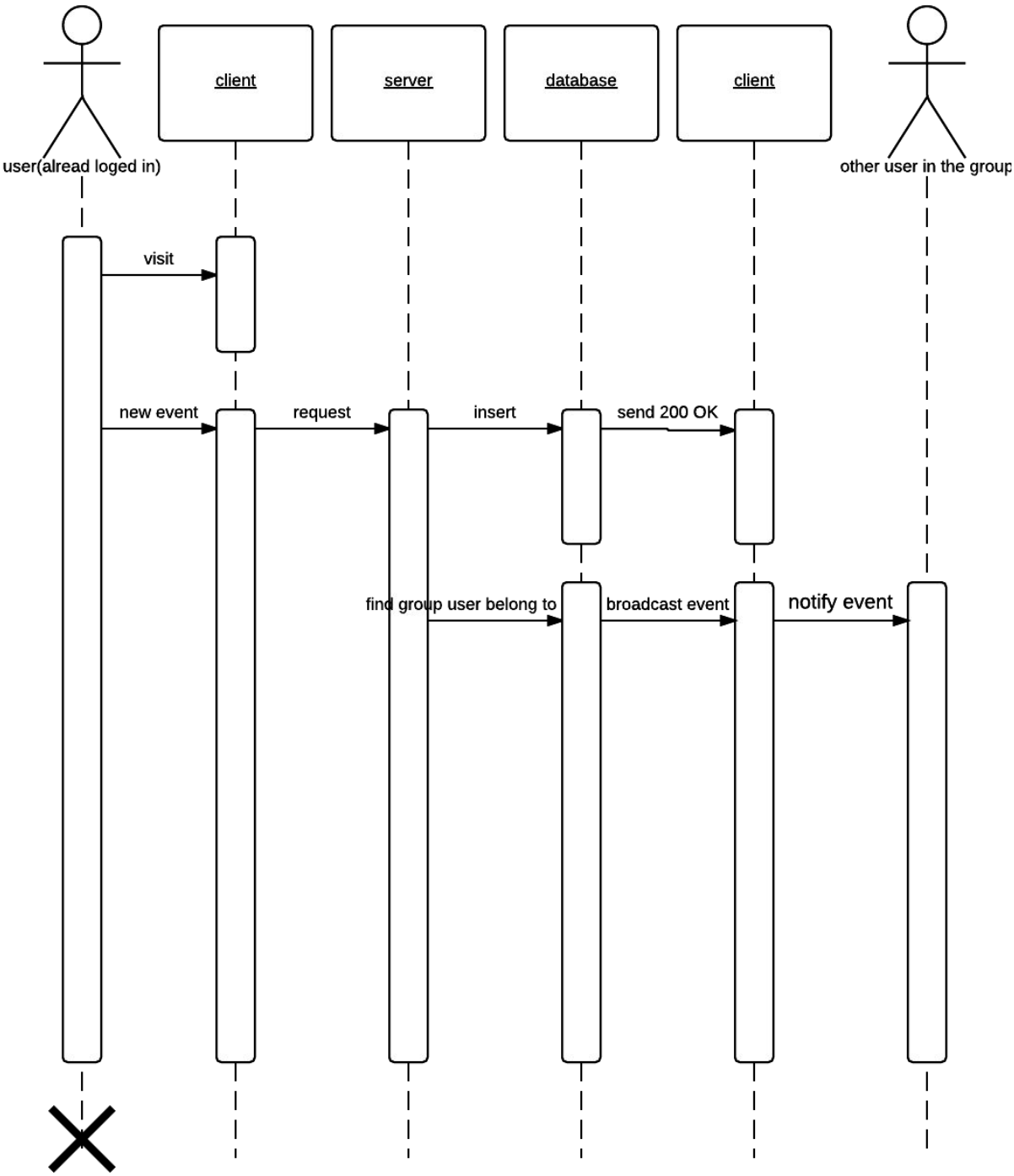
Squence of signing-up or registing.



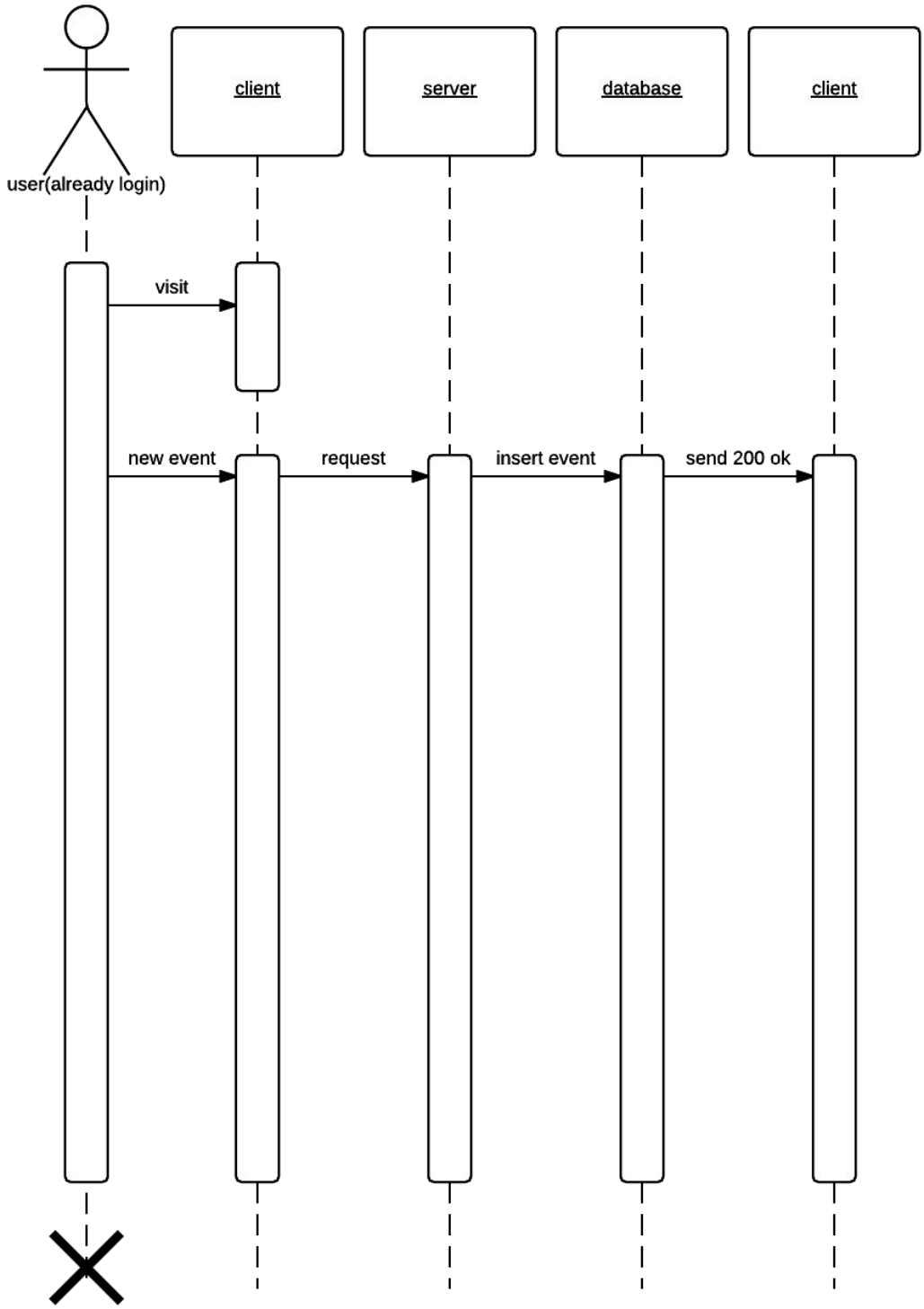
Squence of logging-in.



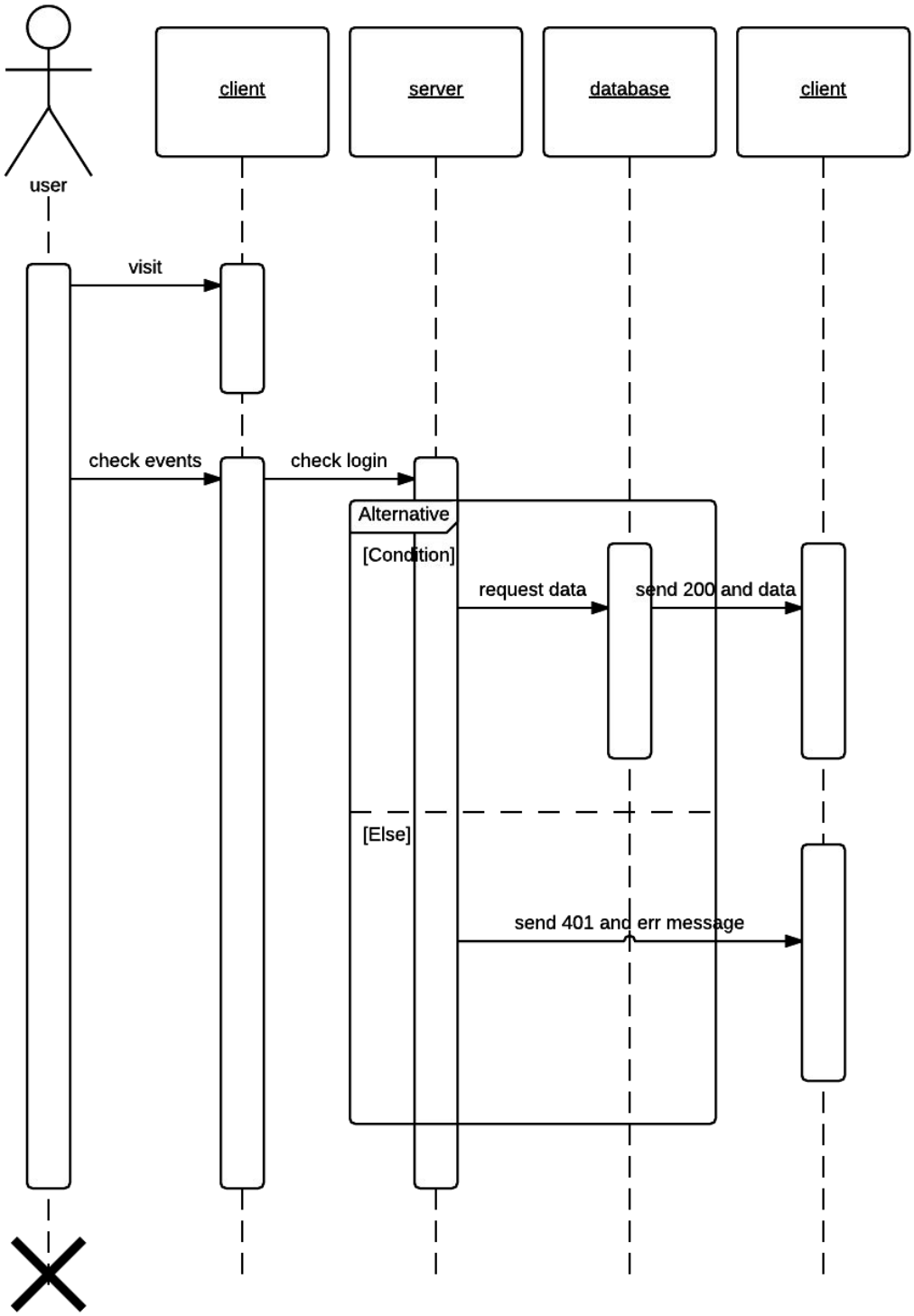
Squence of posting a shared event or task.



Squence of posting a private event or task.



Squence of checking an event or a task.



d. MockUp UI

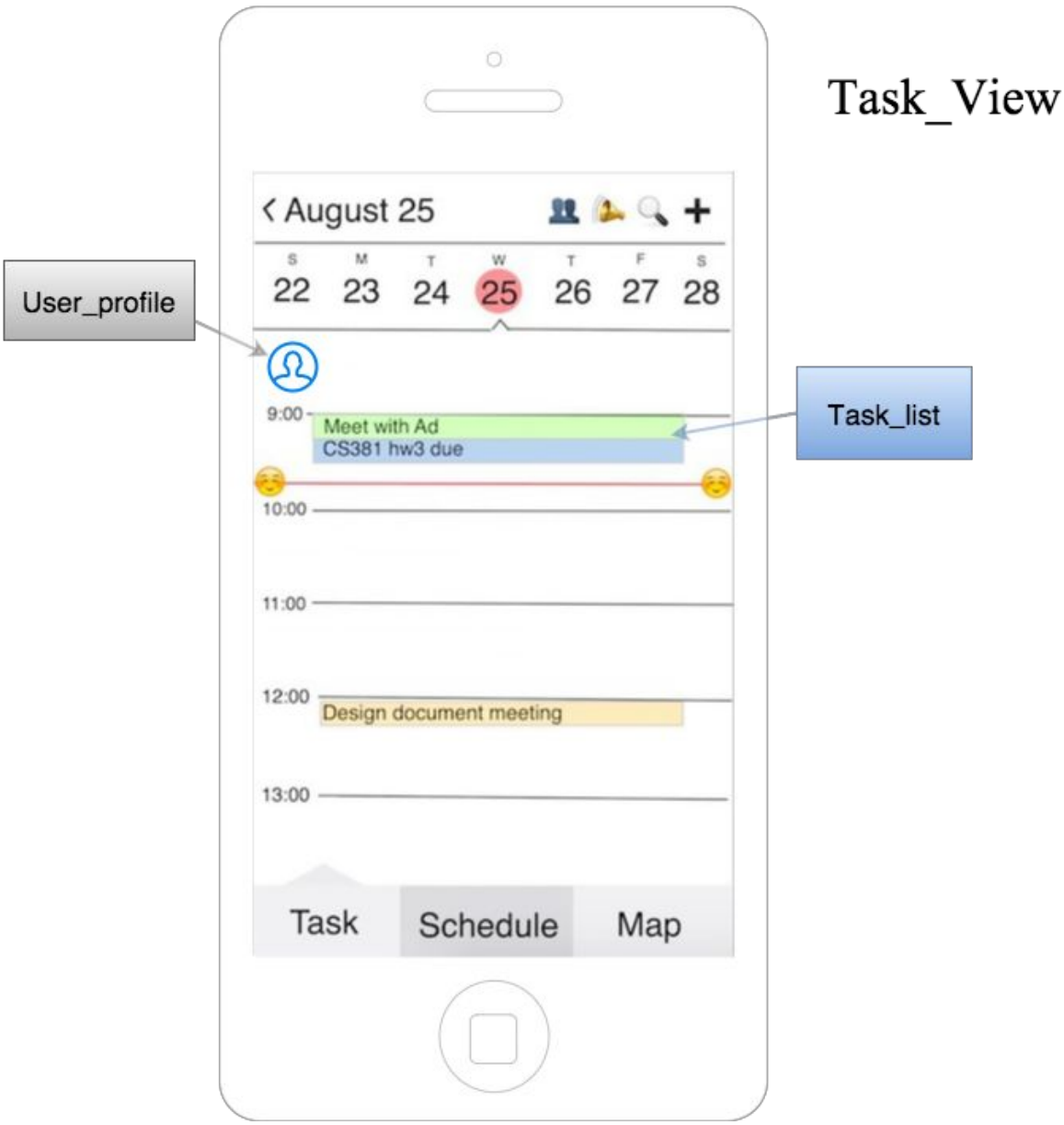
Login View

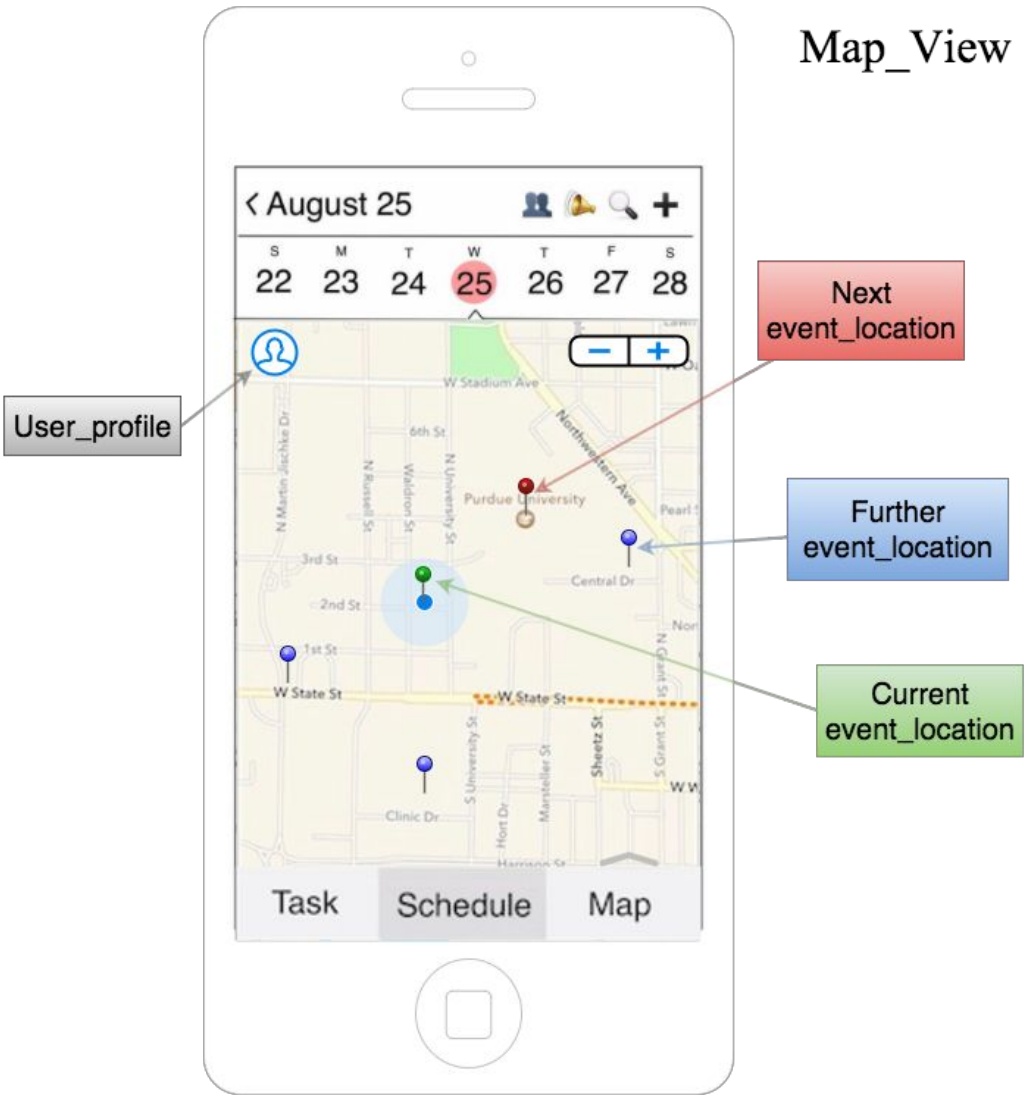
Register View

Profile

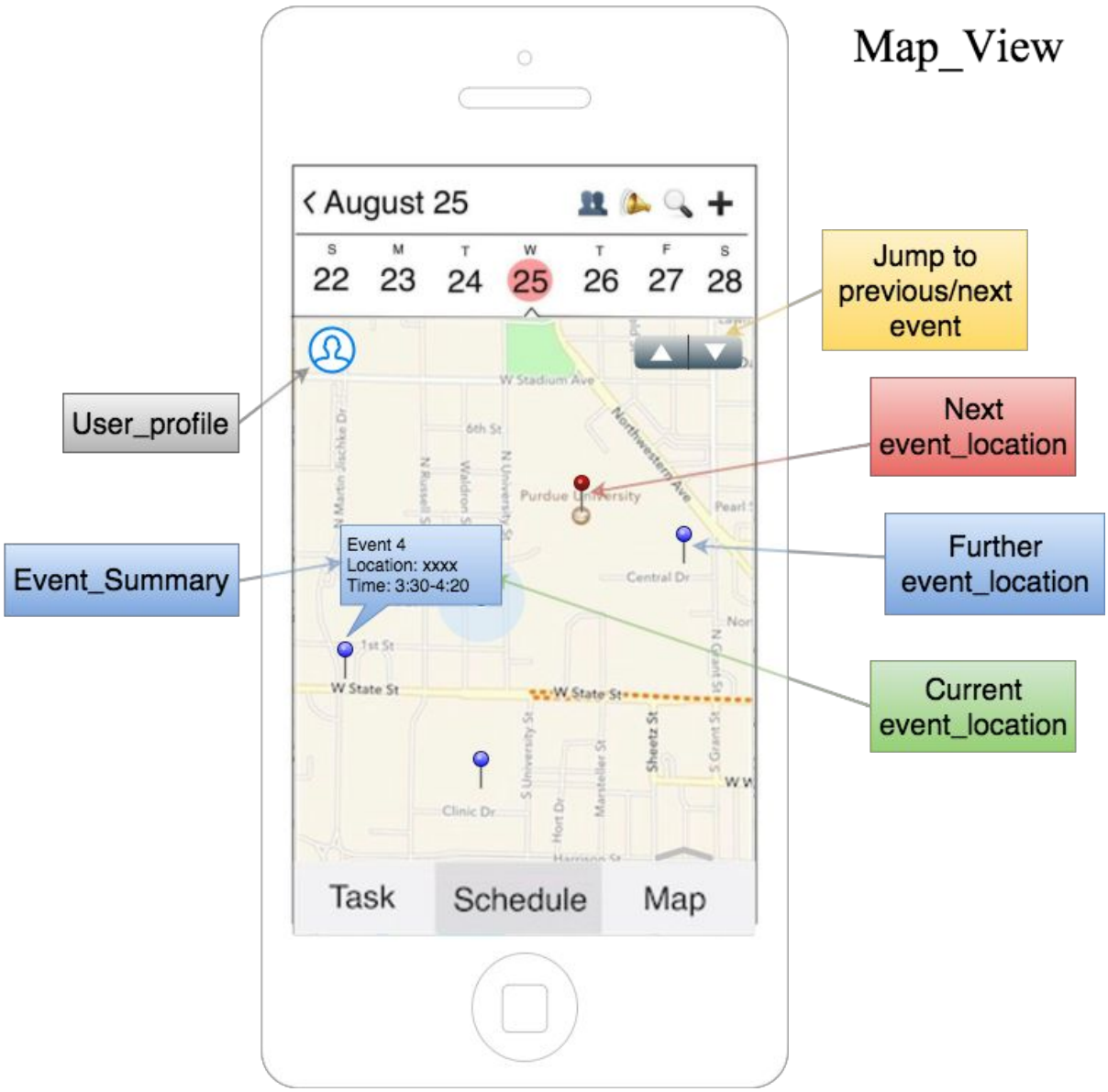
Group Infor

Three Main Views in ***SuLife***


Task_View
User_profile
Task_list
August 25
S M T W T F S
22 23 24 25 26 27 28
9:00
Meet with Ad
CS381 hw3 due
10:00
11:00
12:00
Design document meeting
13:00
Task
Schedule
Map

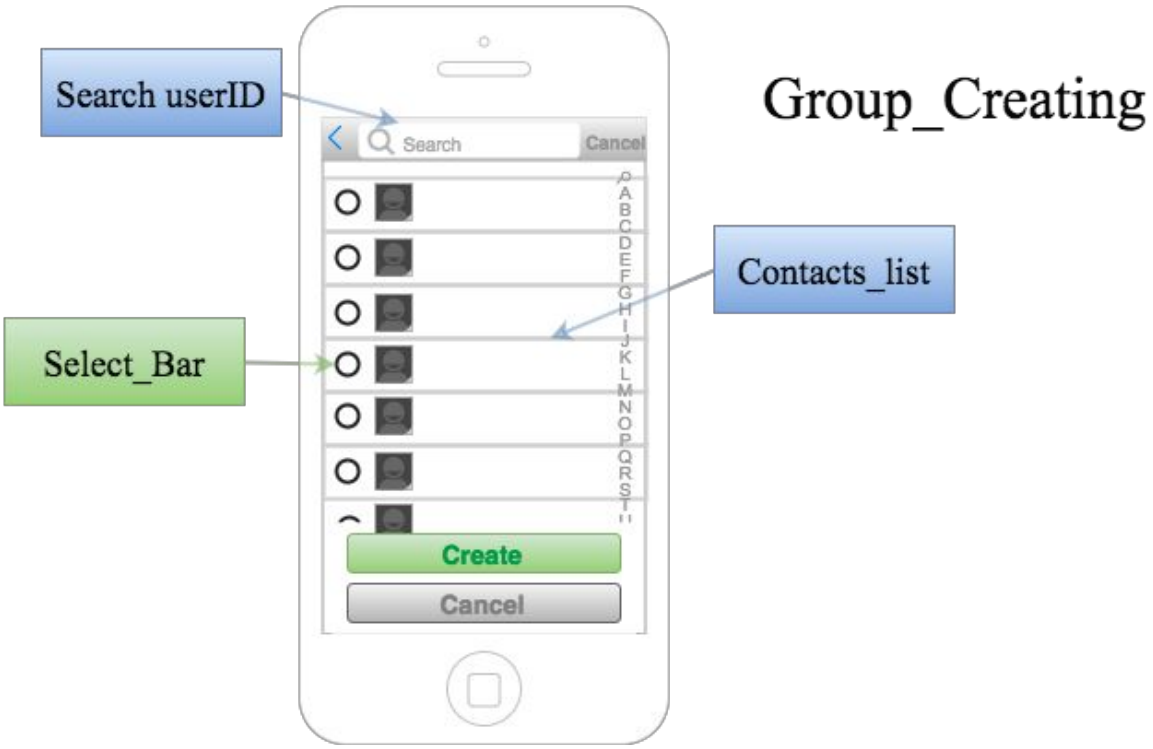


Map API for Map_View is Apple map.
The task is in a time ordered differentiated by color.

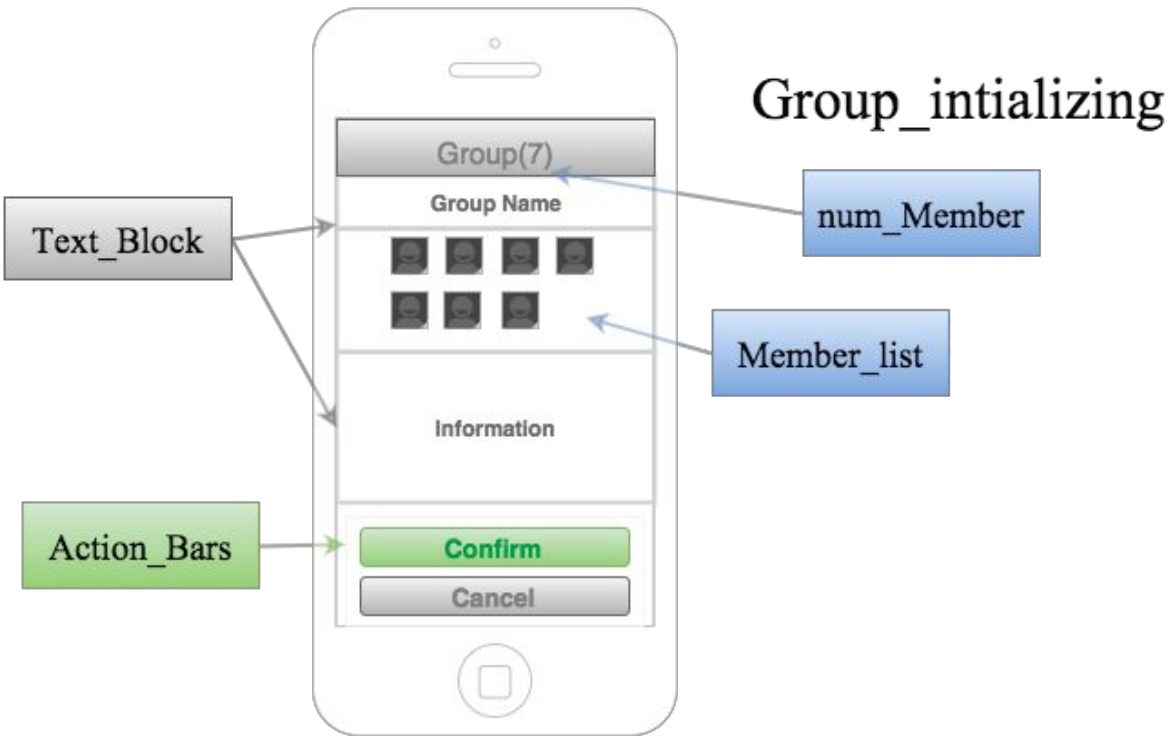


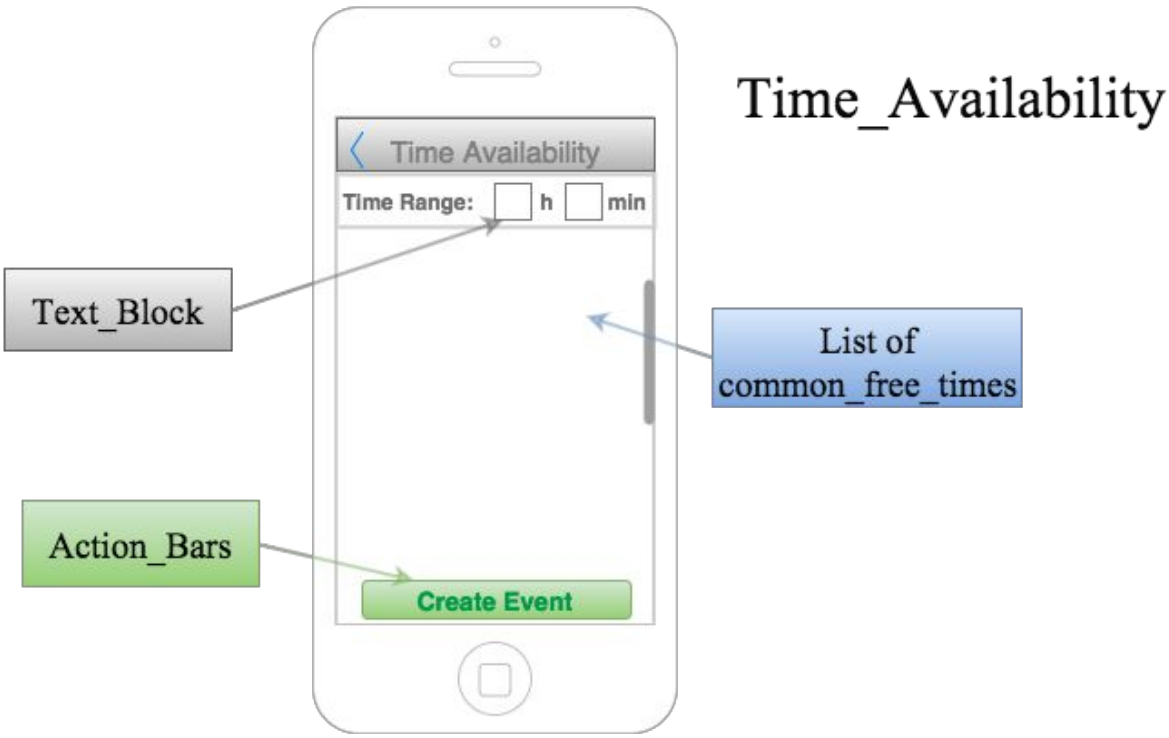
A block contains Event_Summary will show up when clicking on a pin.
User can jump to previous and next event by a single switch. The order is determined by timeline.

Other feature UI in ***SuLife***



To create a group, users could first select a group of people shown as above. Then initialize the group information, shown as below, to create a group.





The figure above shows the diagram when creating a group event. Users can select the time needed for the event, and choose a proper time in the list to create an event.