# Software Engineering Project Presentation (*Group1*)

Karim TEKKAL
Gustavo PETRI

Realised by:

- BRACCHI Pierre
- KABA Saran
- KANDE Desse
- PULLICINO Perrine
- SIROUKANE Slimane
- YUGANSAN Yogaratnam

Github :
https://github.com/kabasaran/Favorite_Places_Management

# An App to share your favourite places !

This app will allow you to keep track of your favorite places and your upcoming events while sharing all of that with your friends !

# System Definition

- Mark a place down as one of your favourites in the world.
- Plan your upcoming events and share them with your friends.
- Share your maps with your friends or make it available publicly.
- Manage multiple maps, and combine them.
- Add your own photos of the events and places you go to.
- Find the shortest way to go where you want to go !

# Users

There is only one type of user in this app : The regular one.

He is able to **create**, **edit**, **manage** and **share maps.**
On them, he can organise events, place points of interest and check what others have decided to share.
He has the possibility to **publish photos** and **messages**, and to **befriend** others.

# Business Objects : recap

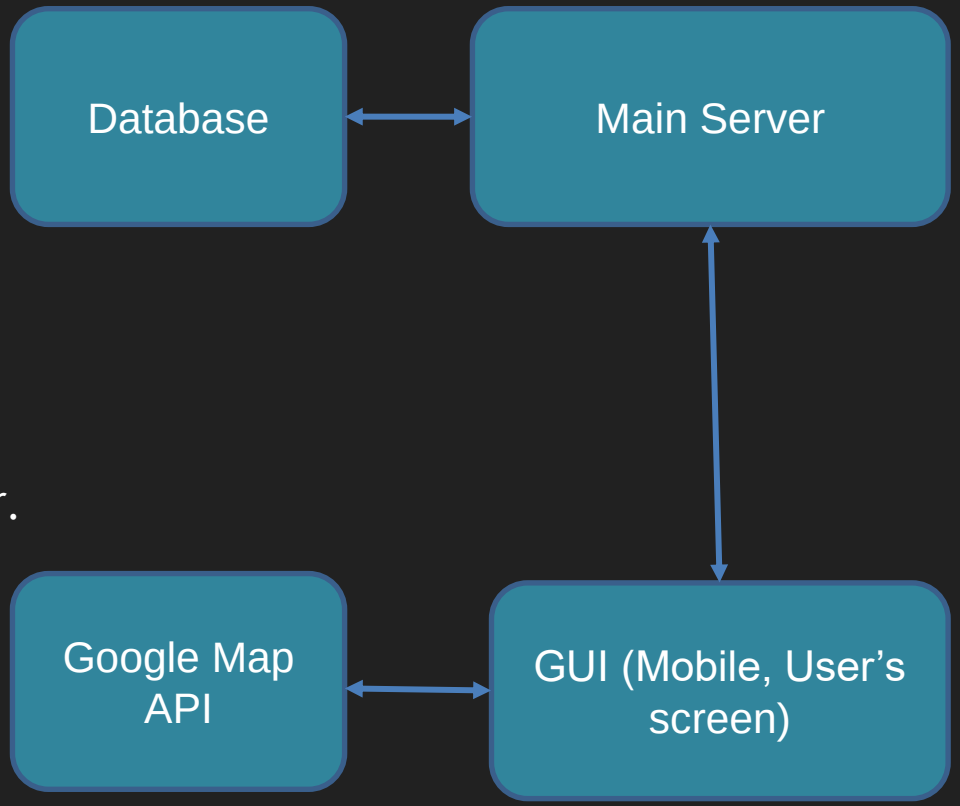
The business objects would be :

- The Users themselves.
- The Places of interest a user can pinpoint.
- The Events the users can schedule.
- The Maps a user will create and manage.
- The messages and pictures a user can send.

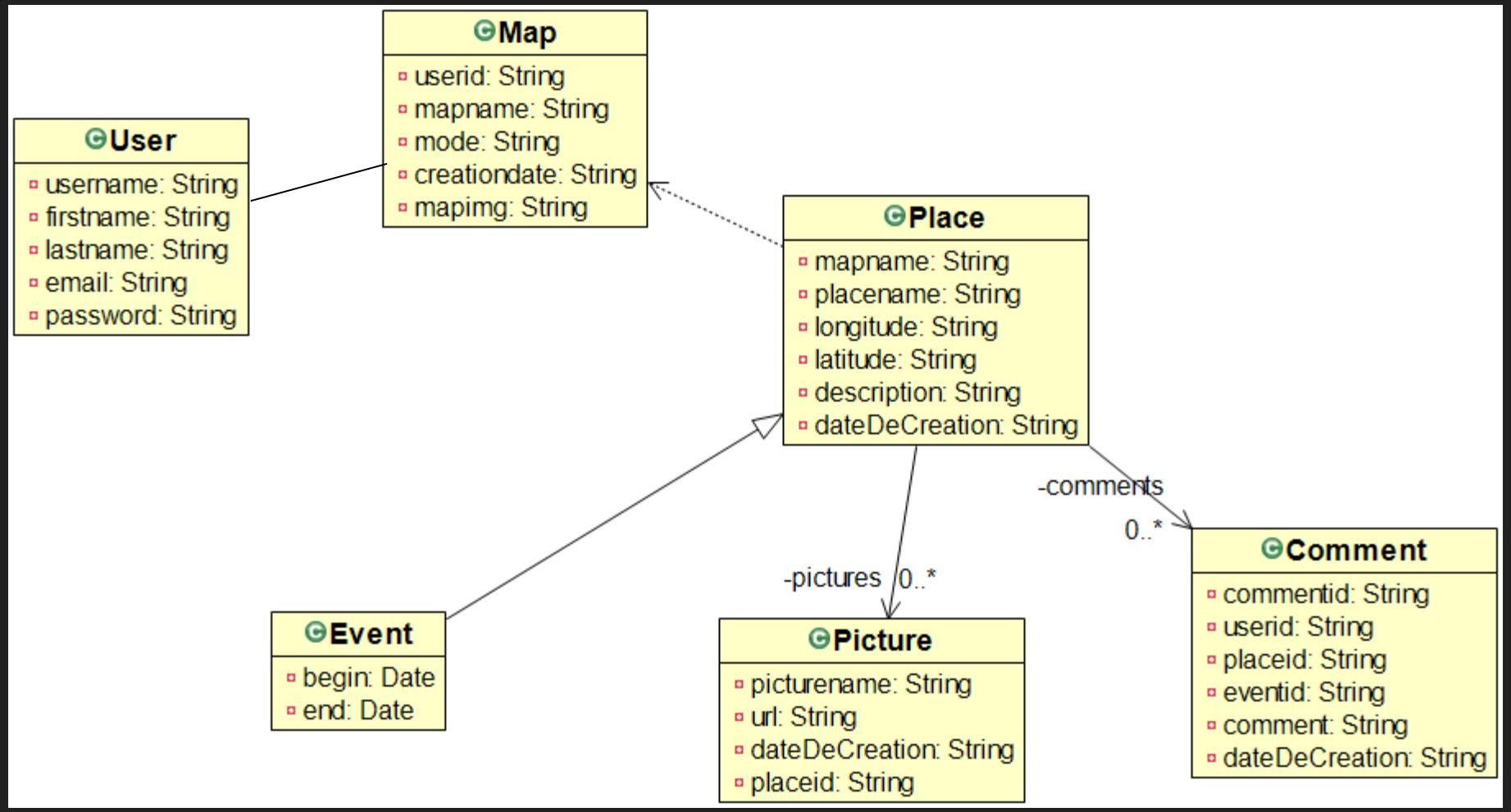
# Technical Proposition

We have four main subsystems coexisting in our app :

- Database stocking data of users and places/events.
- The main server running the code
- The Google Map API, called to display the map to the user.
- The user's own screen, on which we will render the map.



# Business Objects : Diagram



# The Database : How it works

**All our DAOs represent a Database Entity : User, Event/Place, Map,..**

They share common functions extended from the DAO abstract class :

- The **add** function is the same for all the DAOs and allow us to push
- The **delete** function is not shared and takes care, for each DAO, to delete « en cascade » the appropriate data in linked tables.
- The **find** function is explicit and returns the row of the found row.
- The **exist** function allows for the real checks to take place

(For example, we check in it that the end date of an event isn't already over)

# GOALS IN MIND

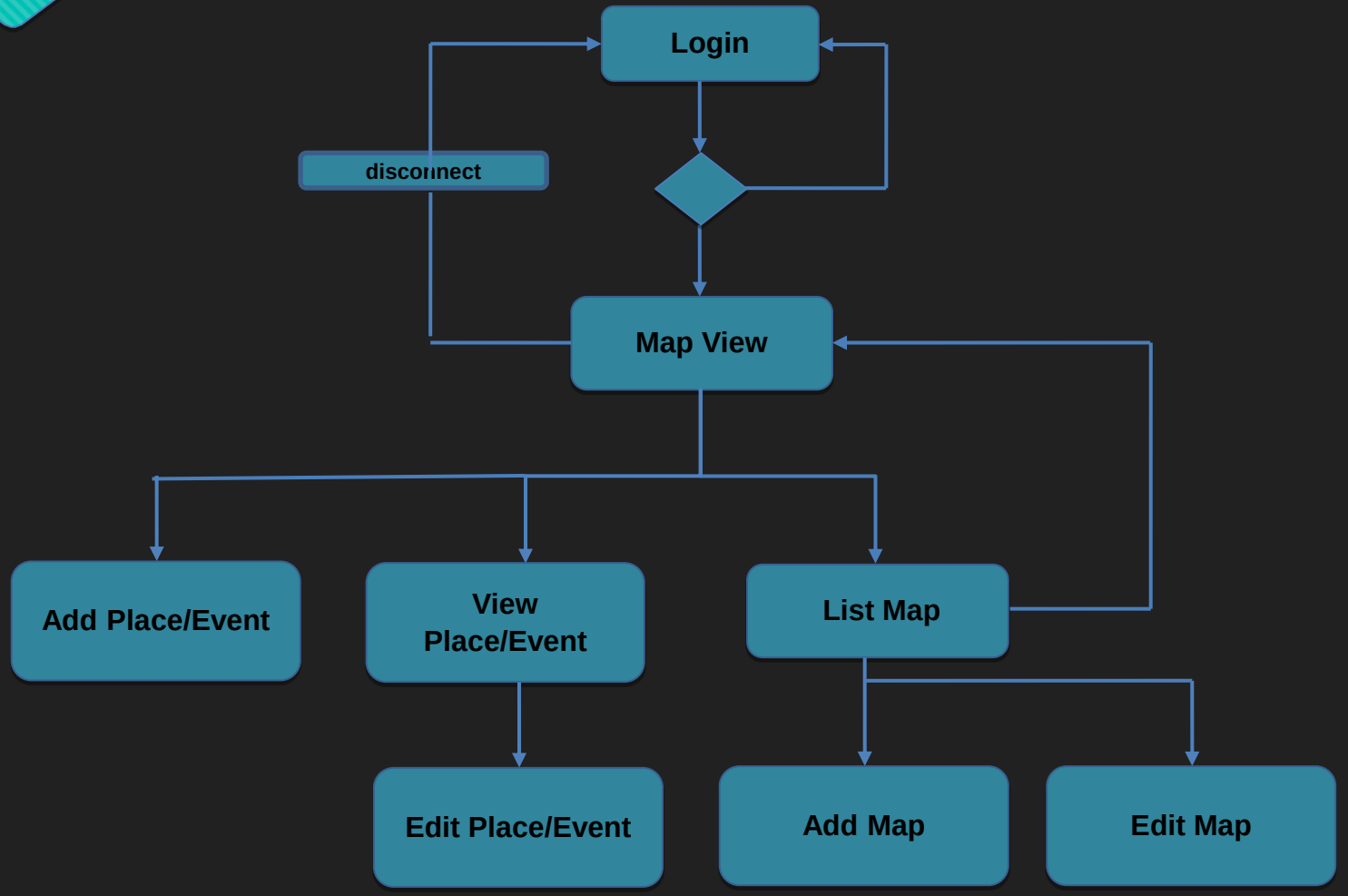
Design

Efficiency

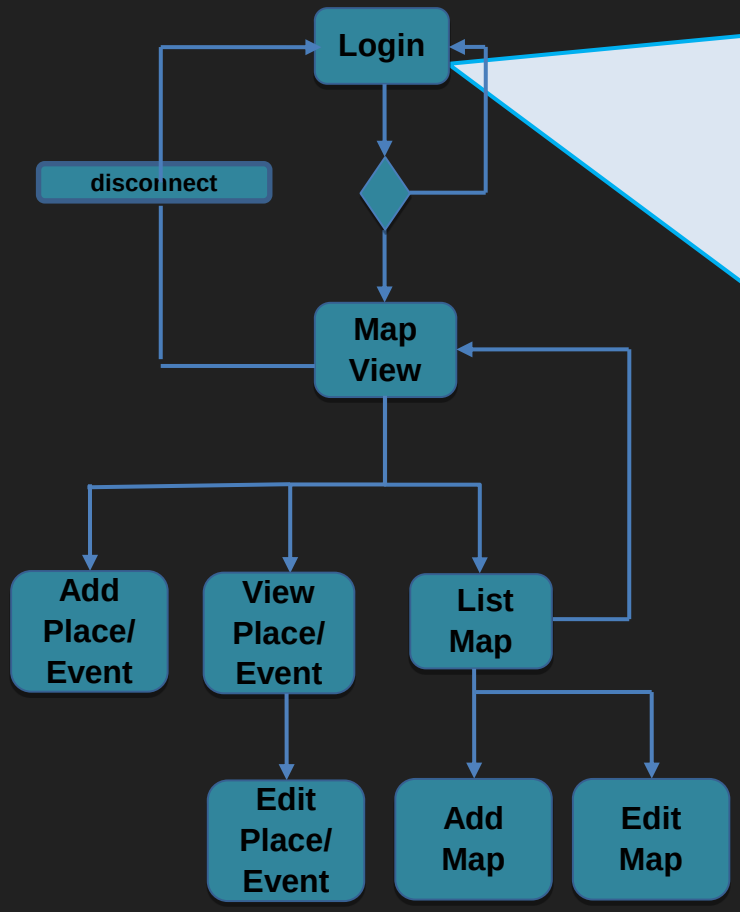
Simplicity

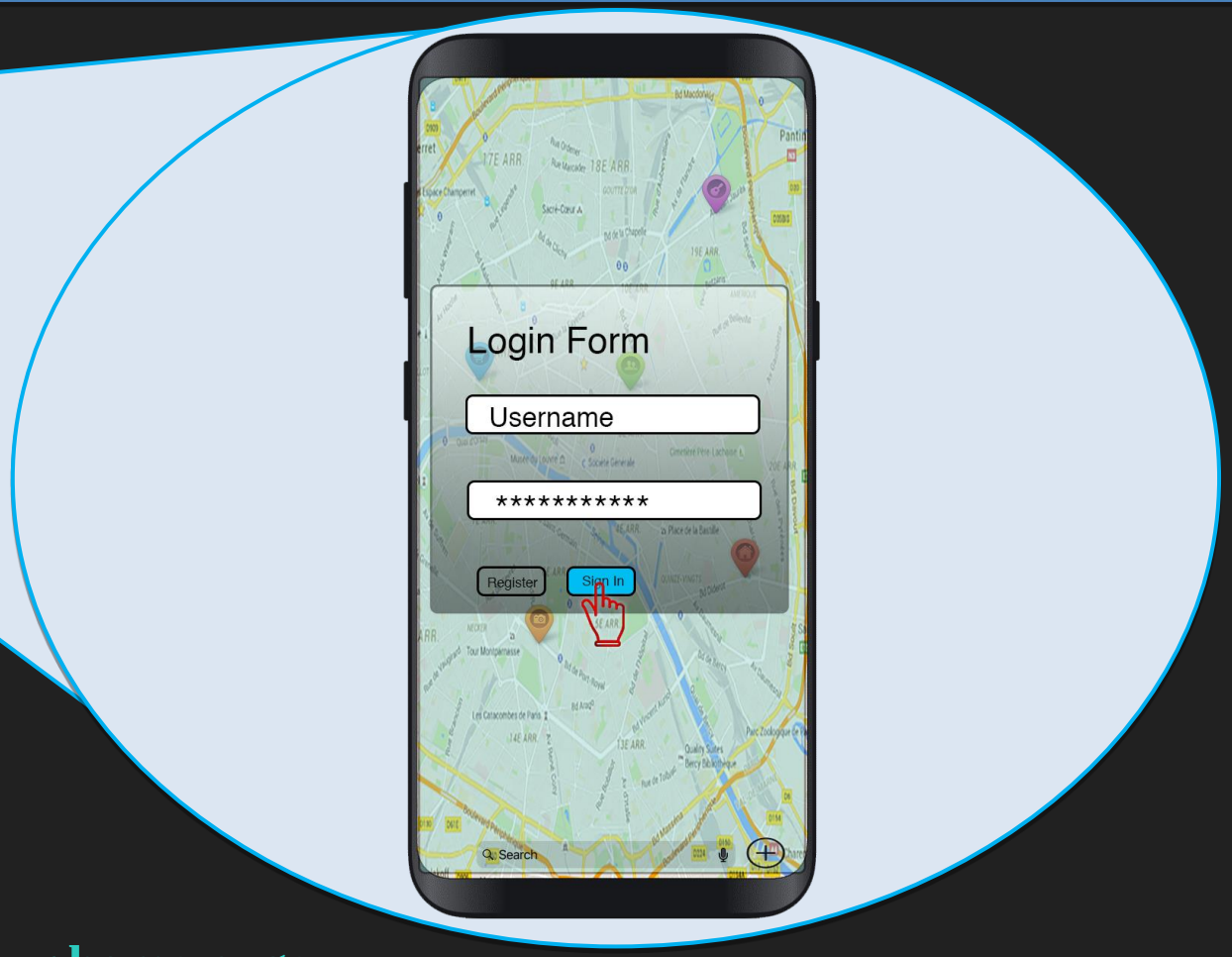
Easy handling

# Navigation Diagram



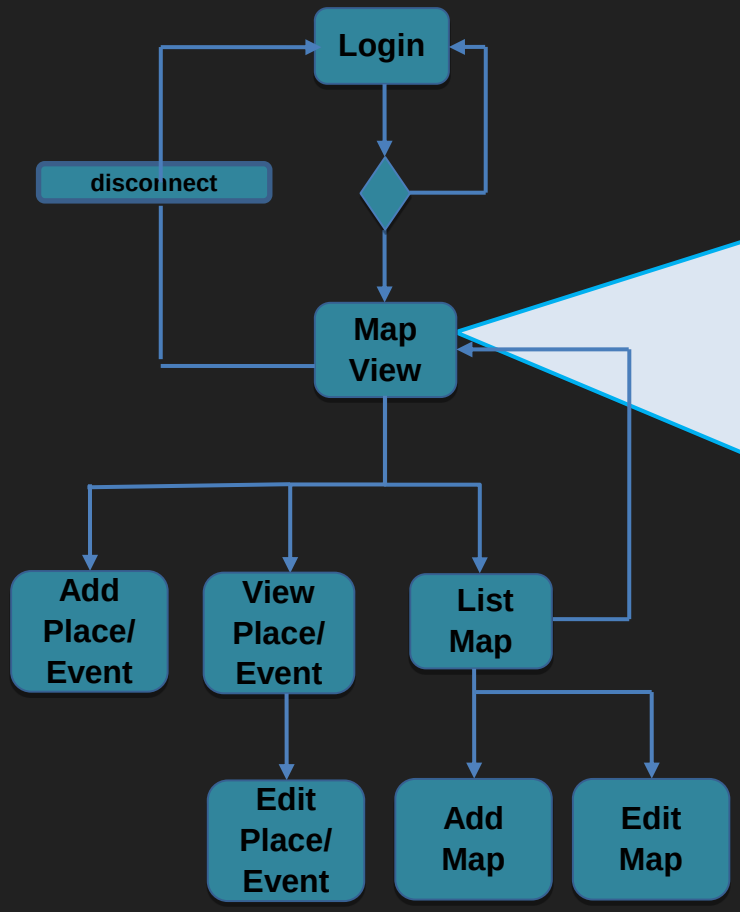
# Navigation Diagram




This would be your home page.

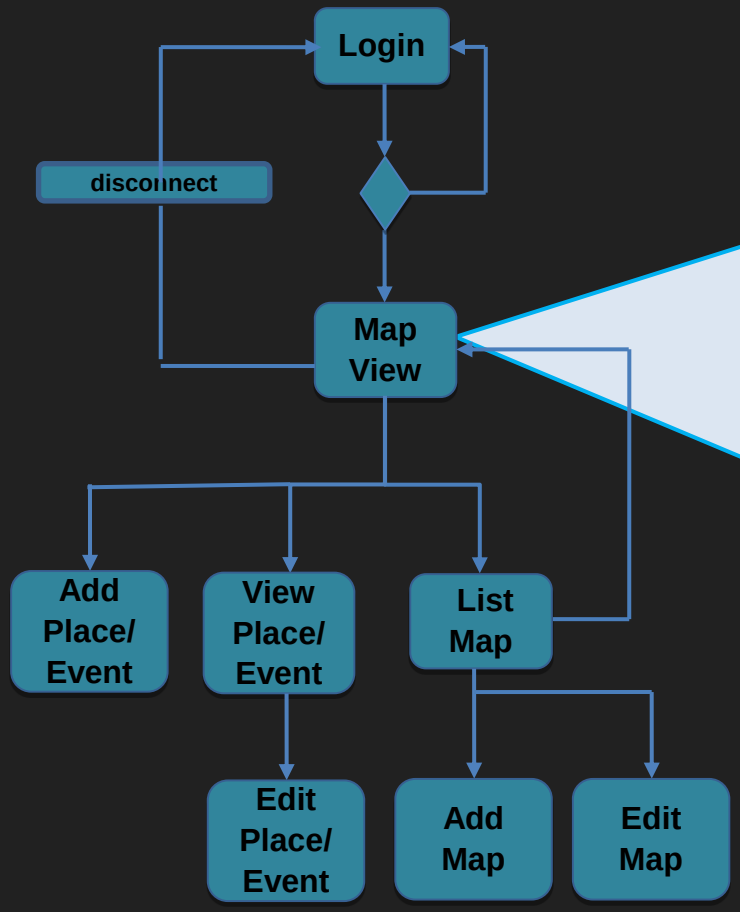
# Navigation Diagram





The idea is that as soon as you arrive on the homepage, you can see the relevant information you might need.

# Navigation Diagram

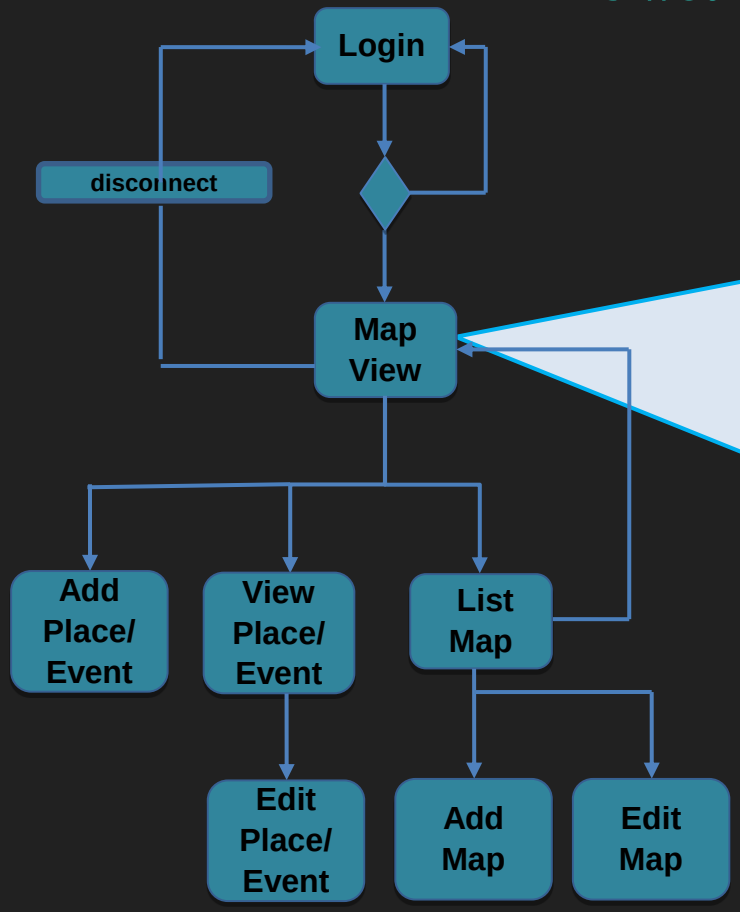




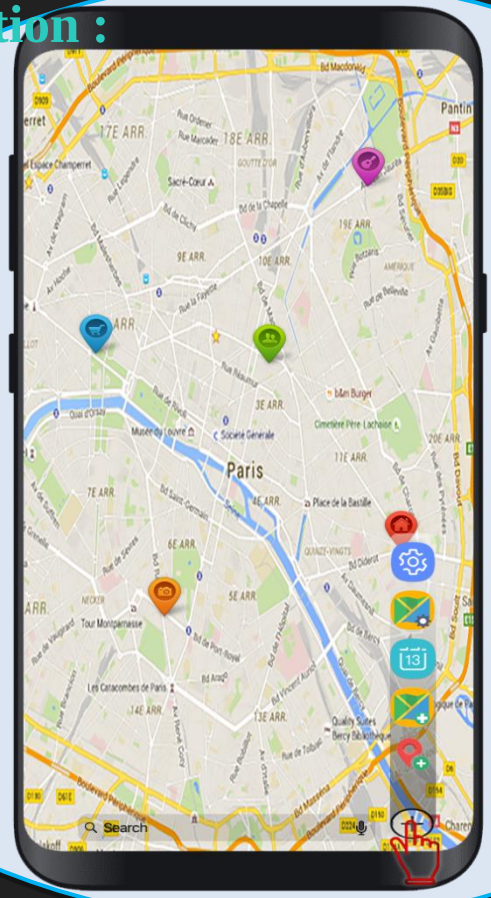
In this case, it would be streets around you and thus
the potential interest points you or other people may have, could be marked down

# Navigation Diagram

This would be the main way of using our application :

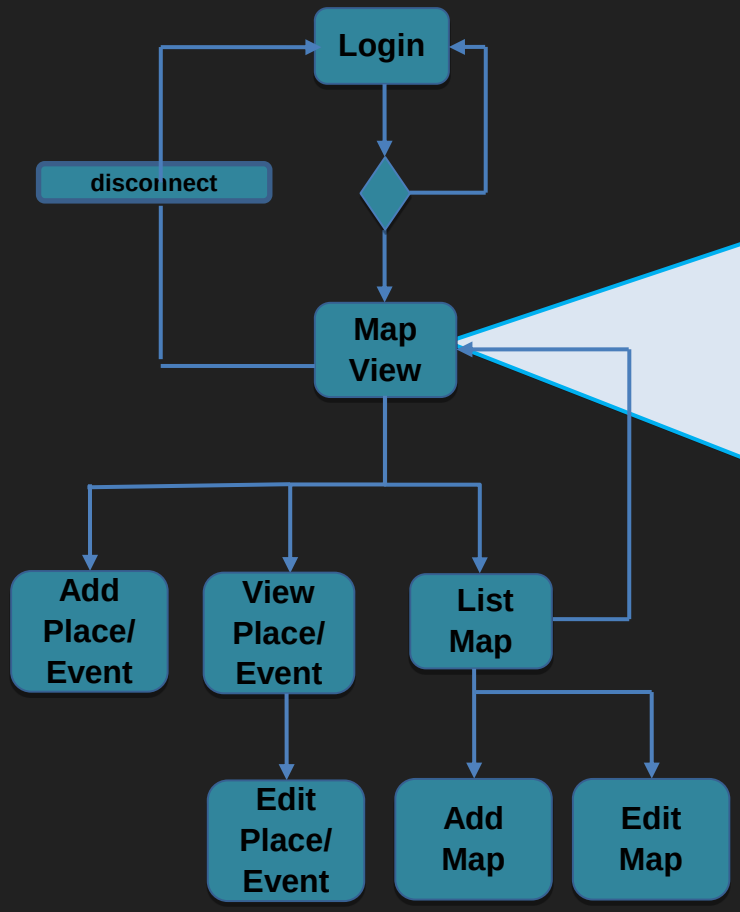


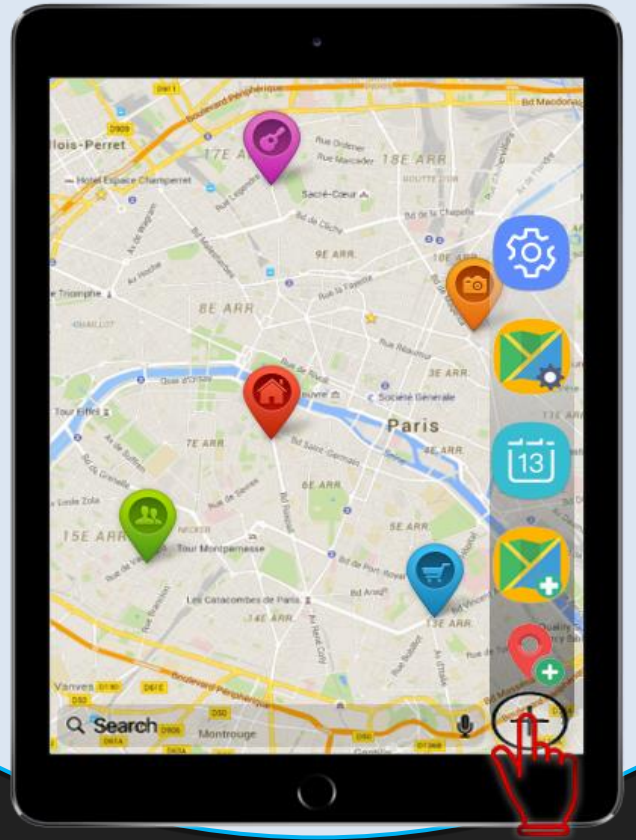

Clicking on the cross sign would open a pop-up menu that would allow the user to select what he wants to do with the app.




This seems to us to be the best way to provide maximum efficiency for the user.

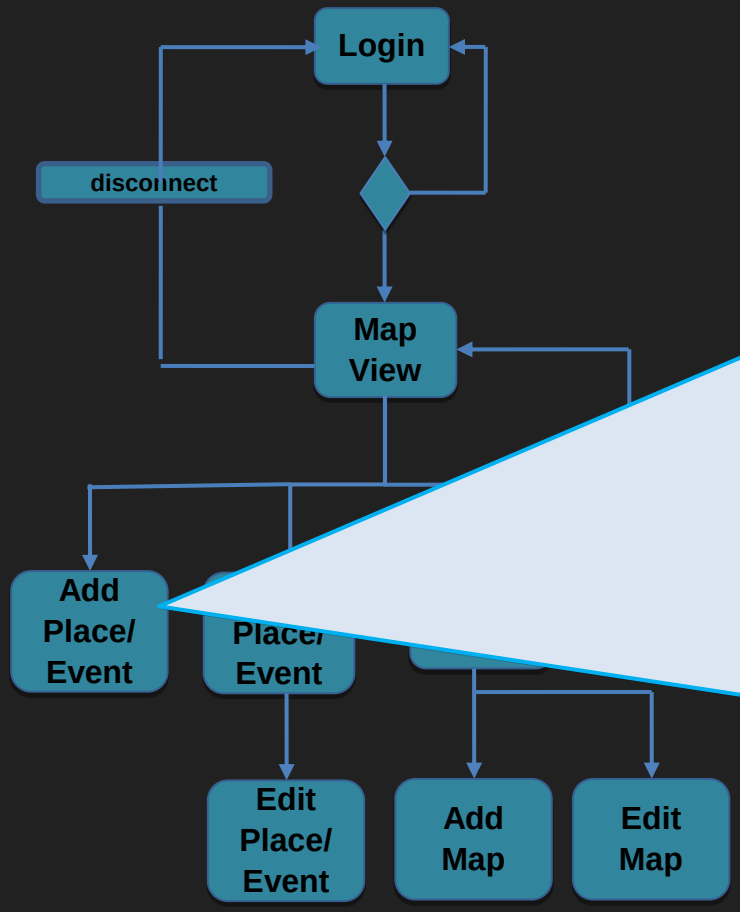
# Navigation Diagram

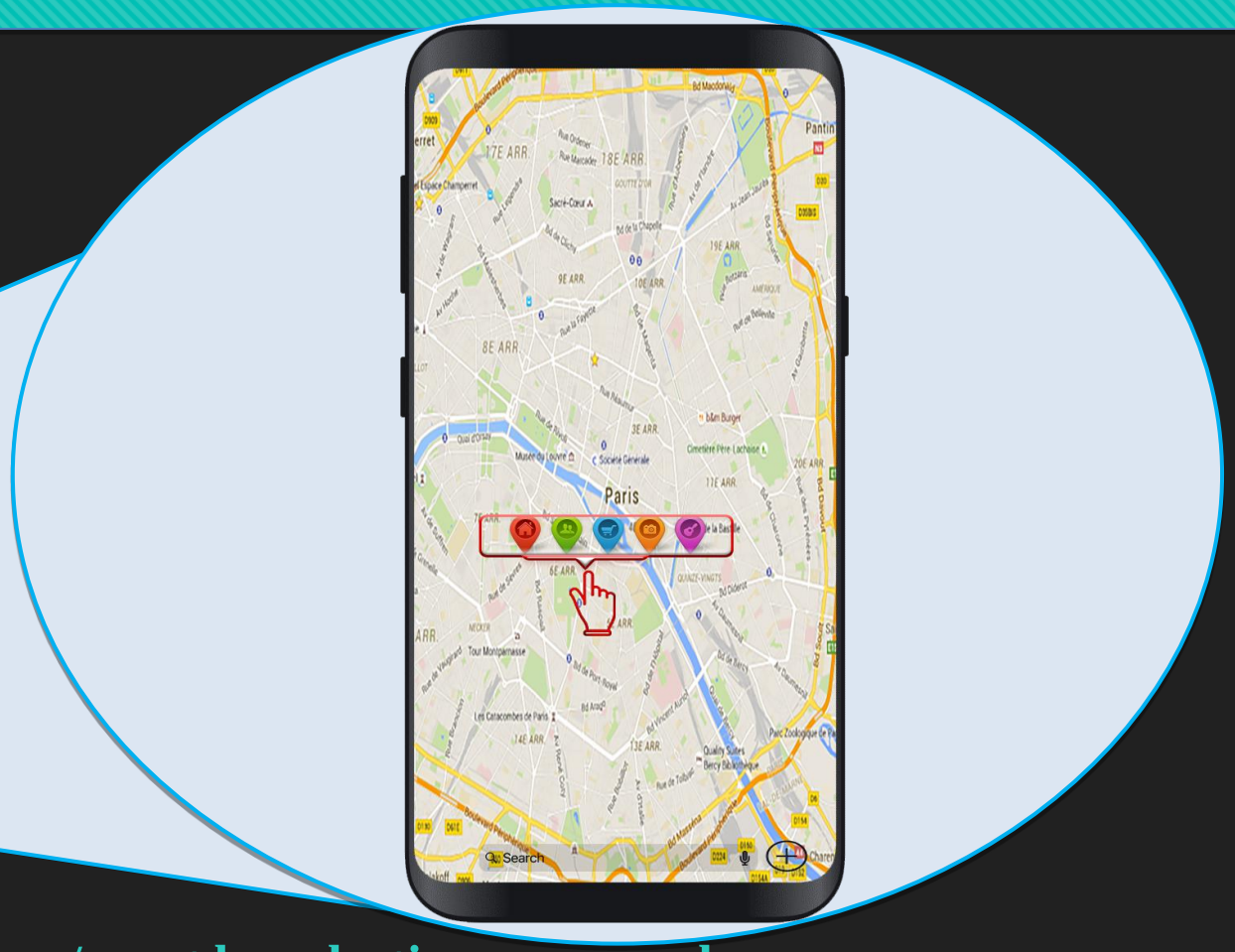



This is mainly the same idea as the previous mock-up, but the interface would have to be revamped to better fit a tablet.

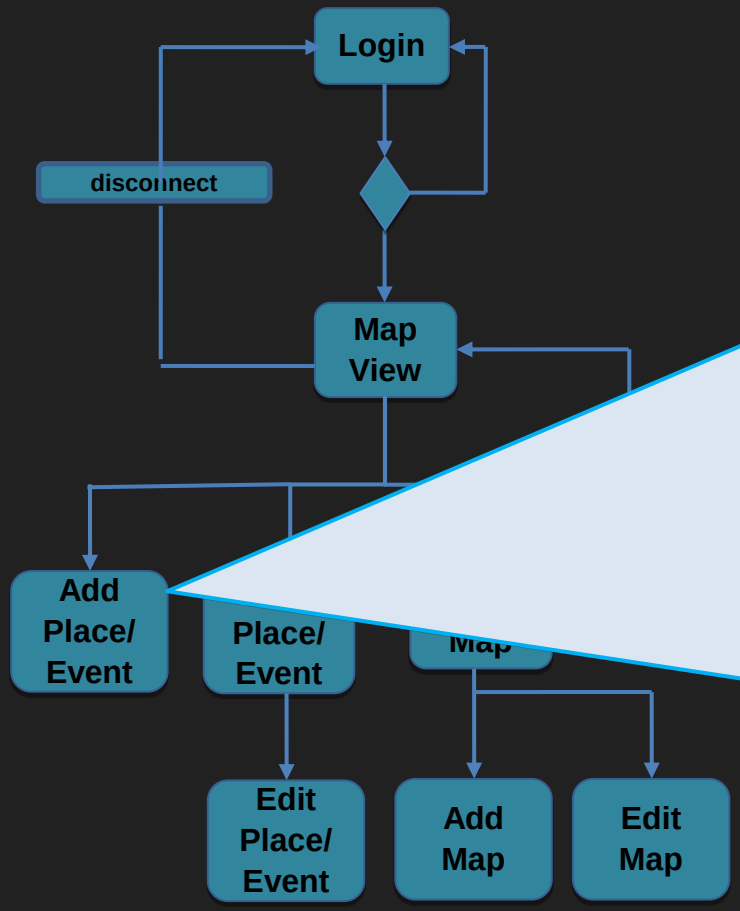
# Navigation Diagram

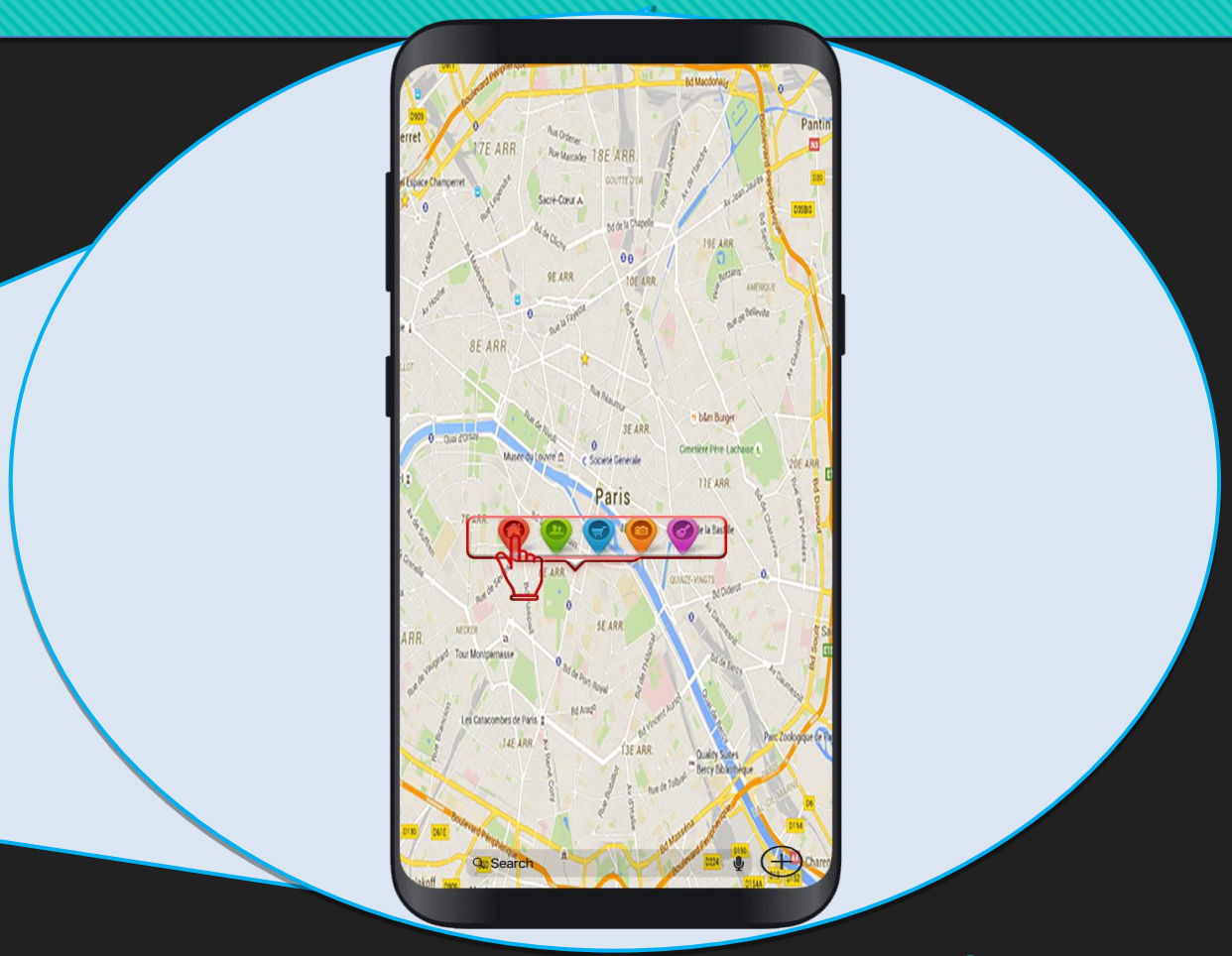



User can add places/event by selecting on menu bar.
Or by pressing on a specific place, a pop-up can be opened…

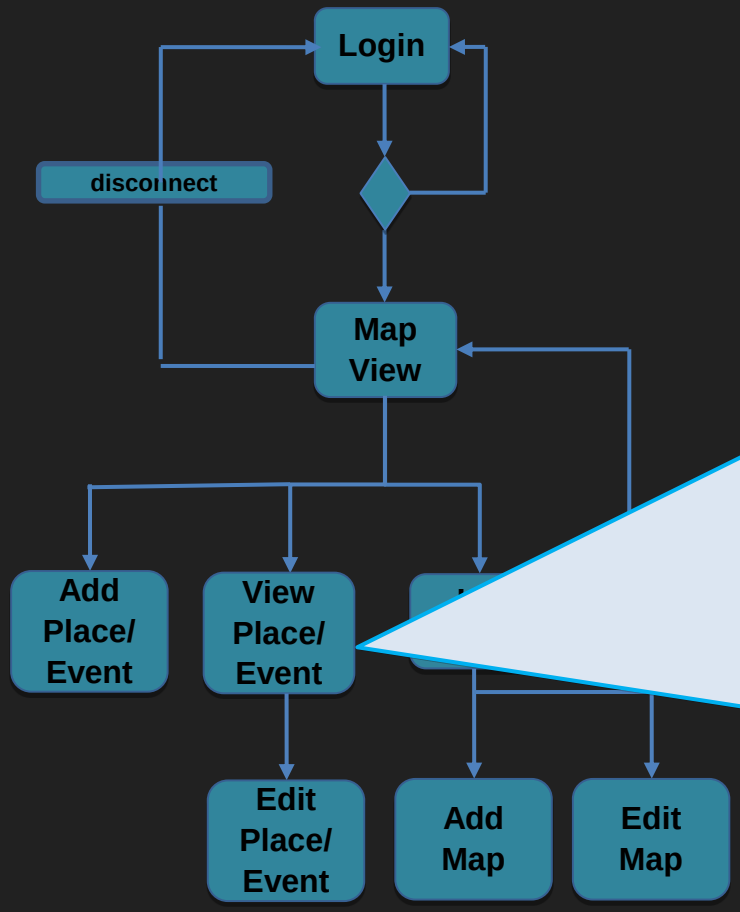
# Navigation Diagram

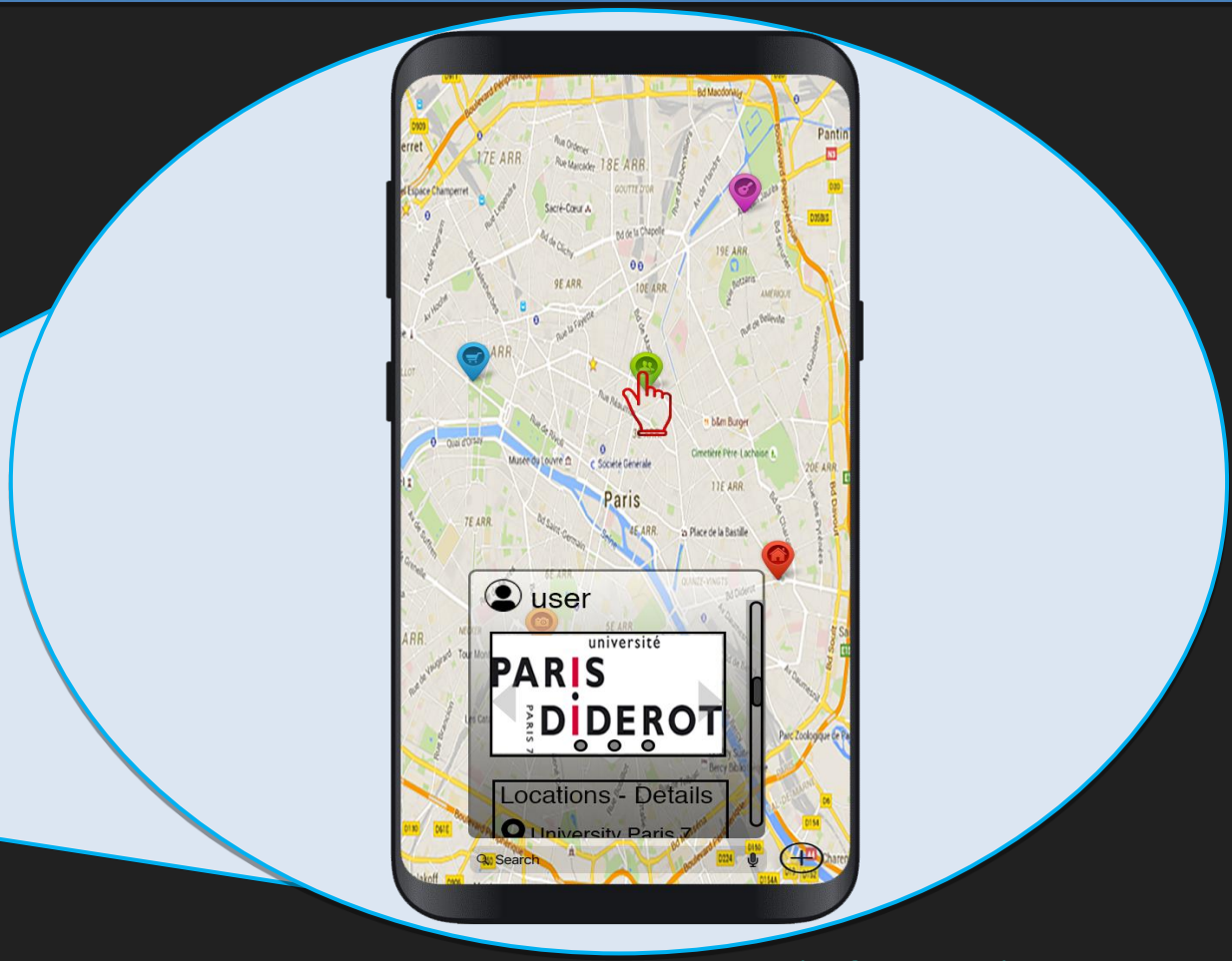



that would allow the user to select whether he wants to add features like places, events...

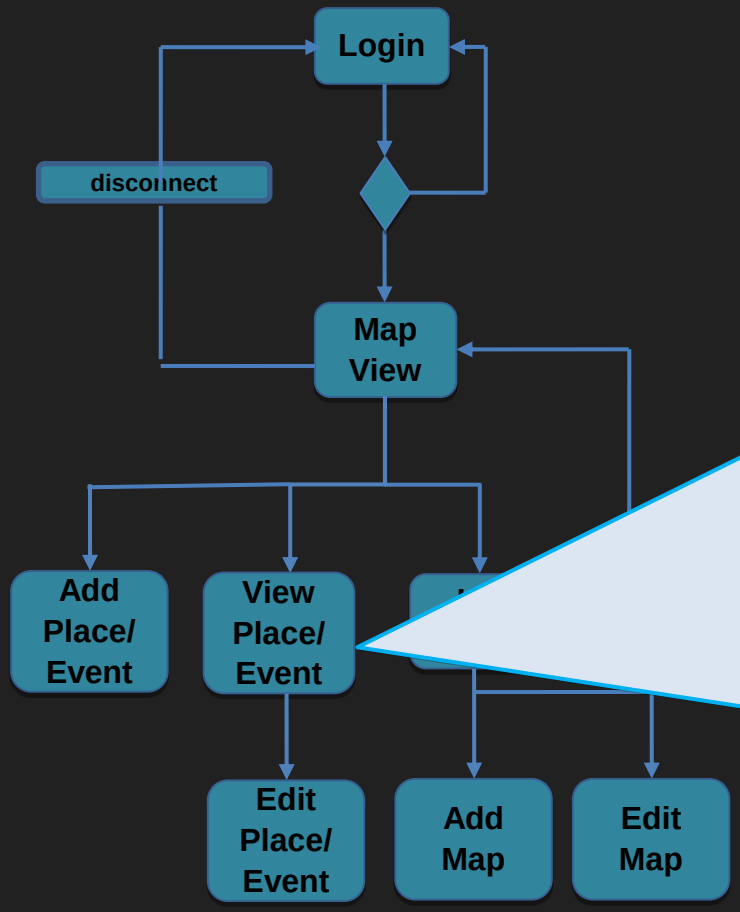
# Navigation Diagram

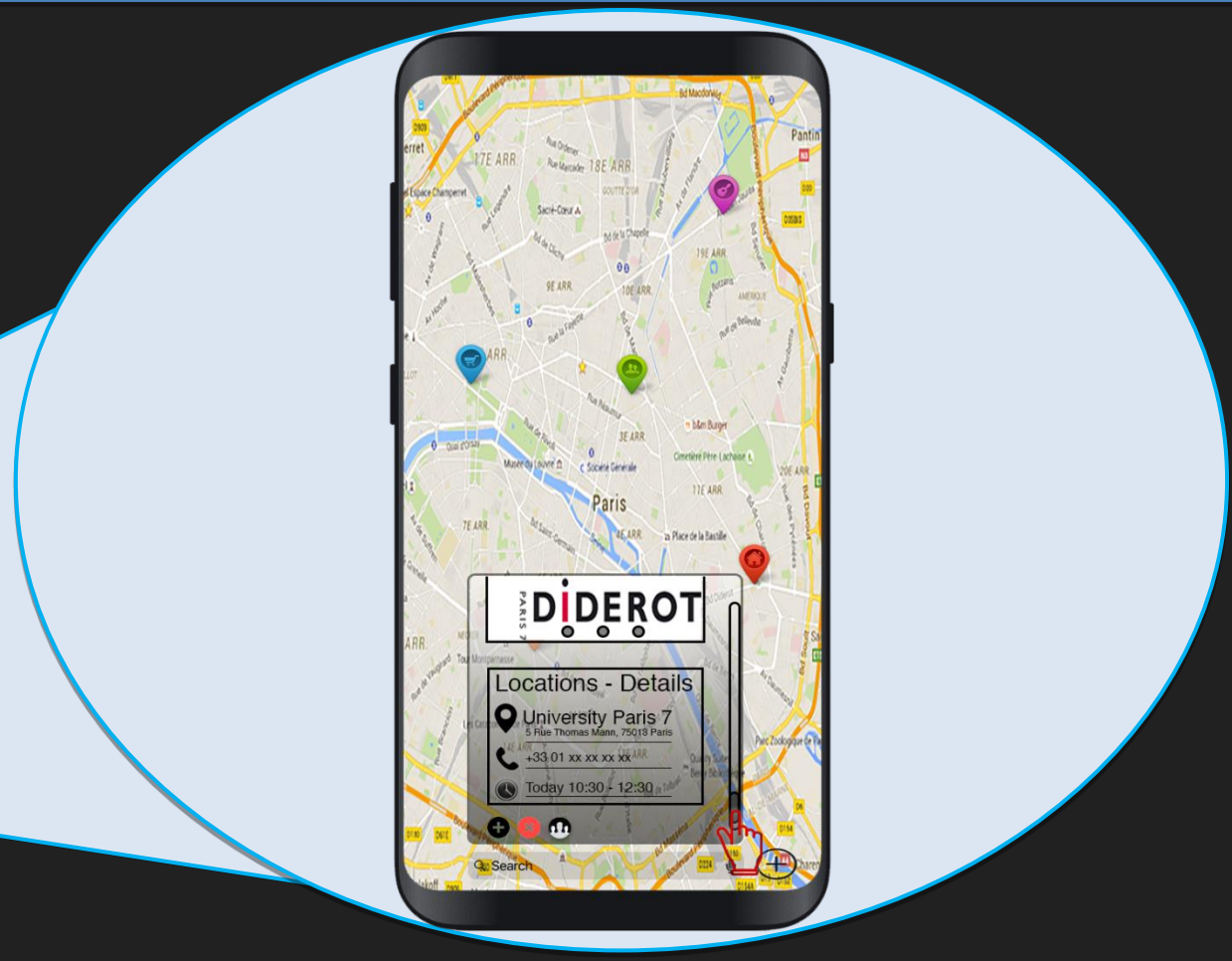



By touching icon on map, we can get access to more information about…

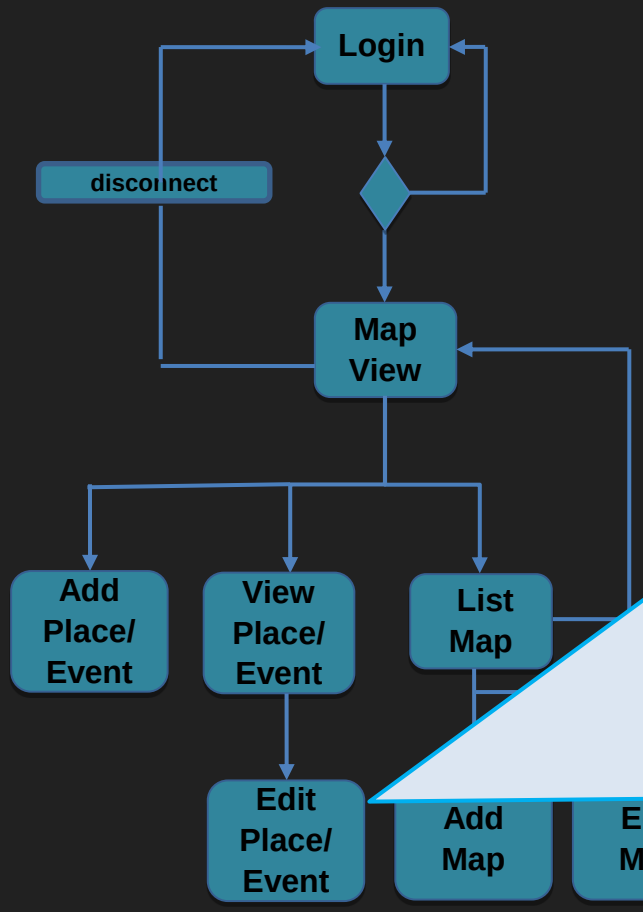
# Navigation Diagram



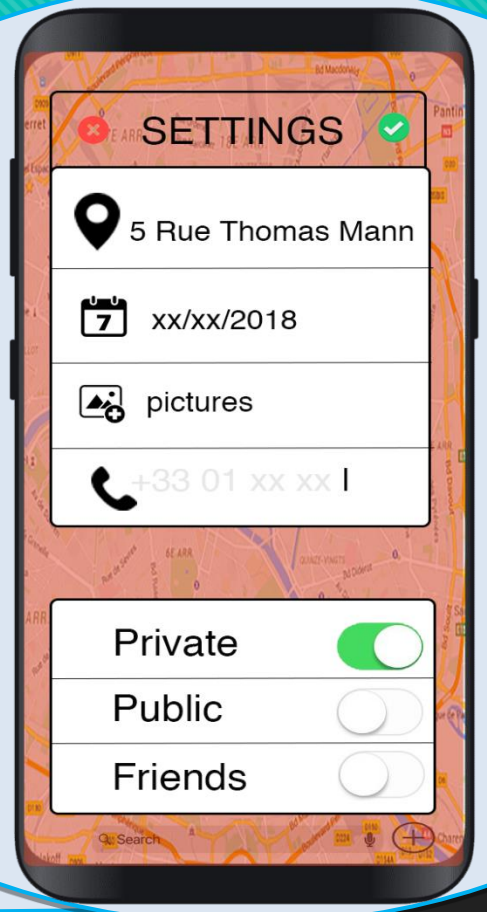

photos shared, information about events, description of place/event,…

# Navigation Diagram



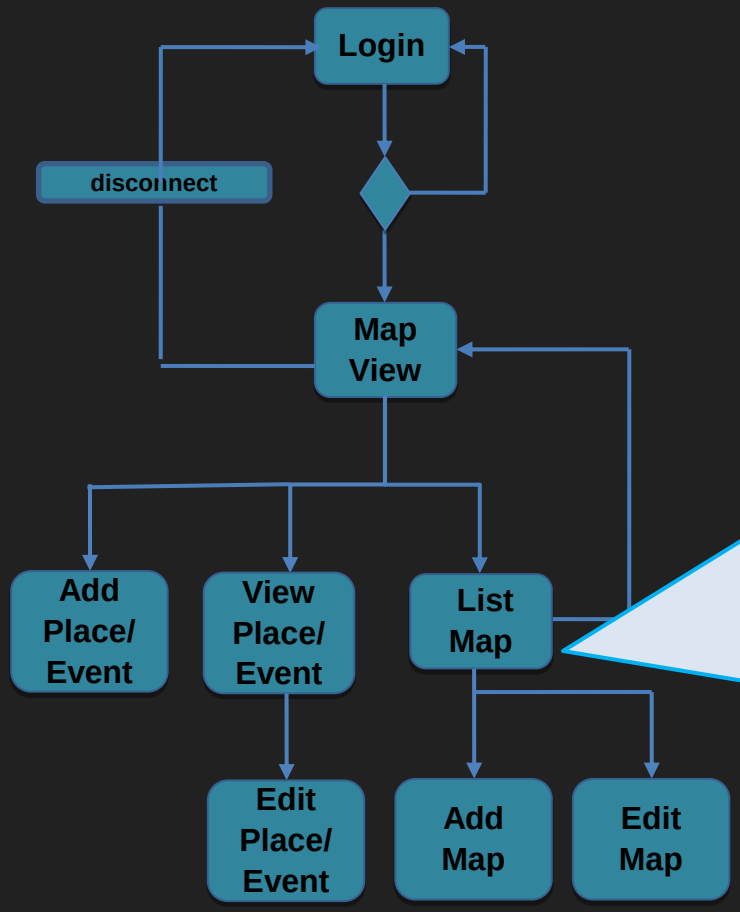

It would be the adding/setting menu for place/event.

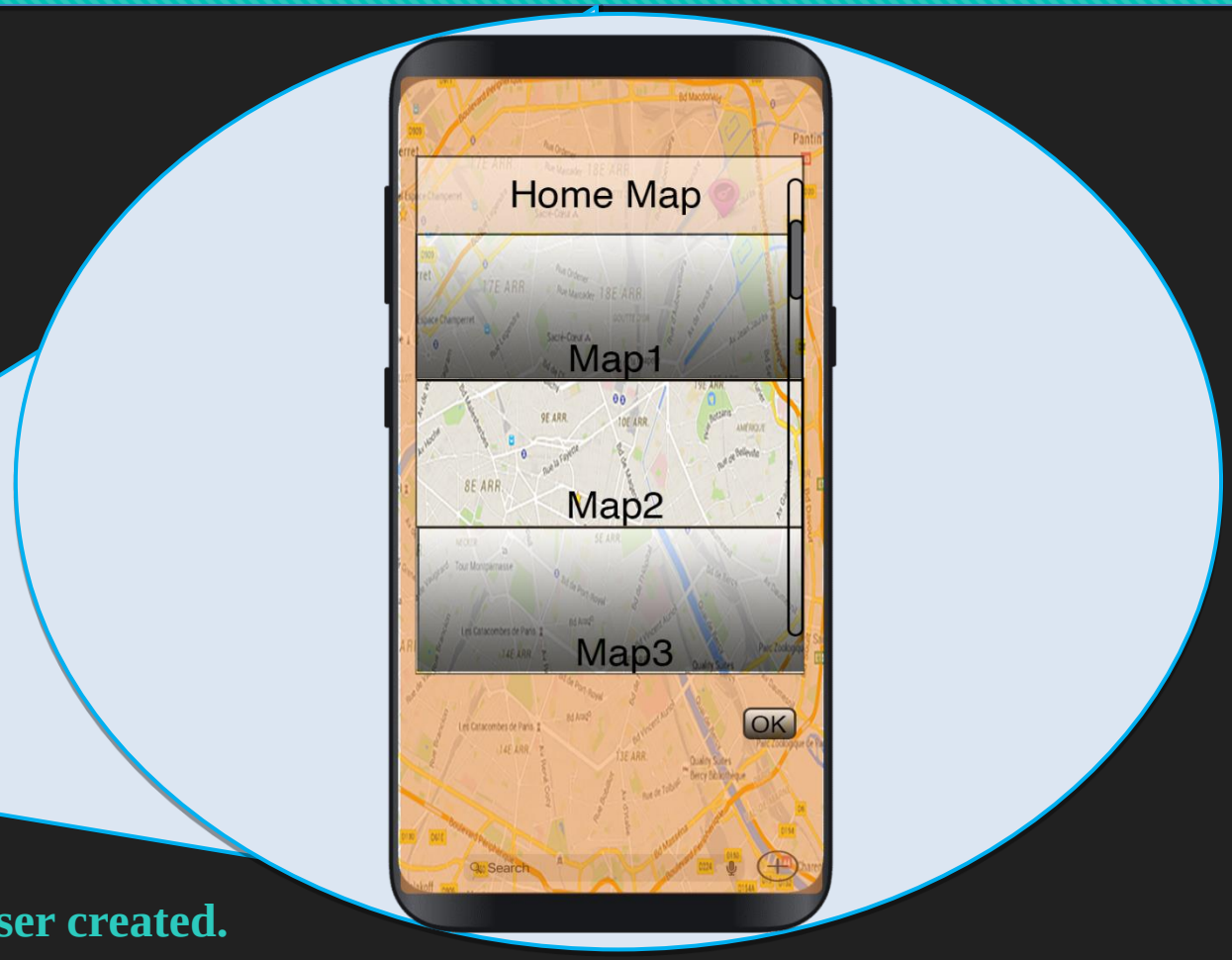
If the user adds by the menu bar, he can write the address and fill up other information.




If its by the pop-up, address will auto completed and he can fill up the rest of information. He can also decide to share it.

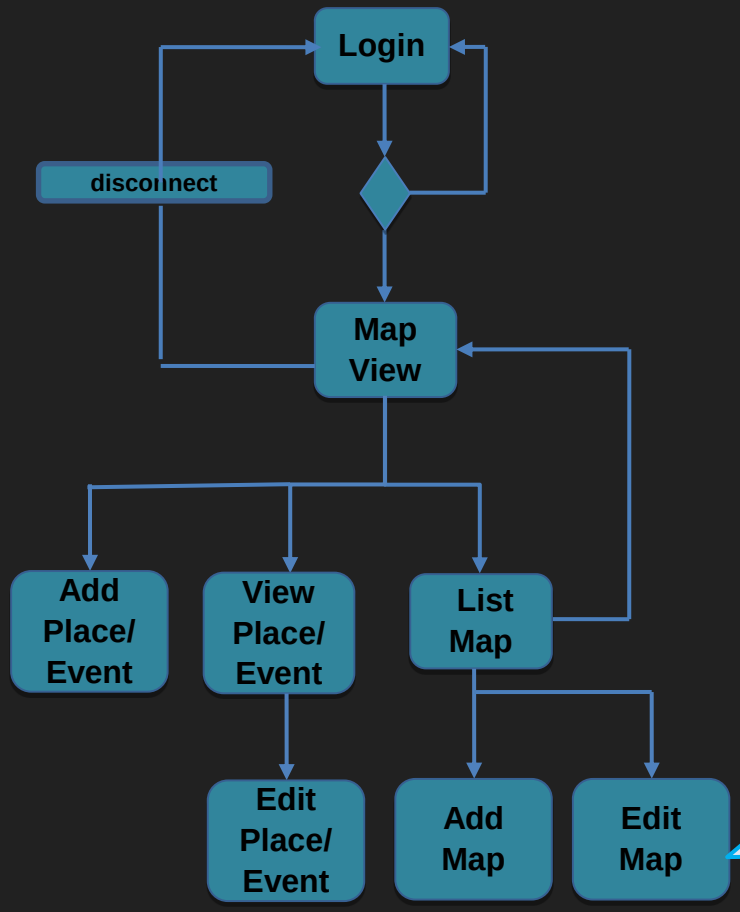
# Navigation Diagram

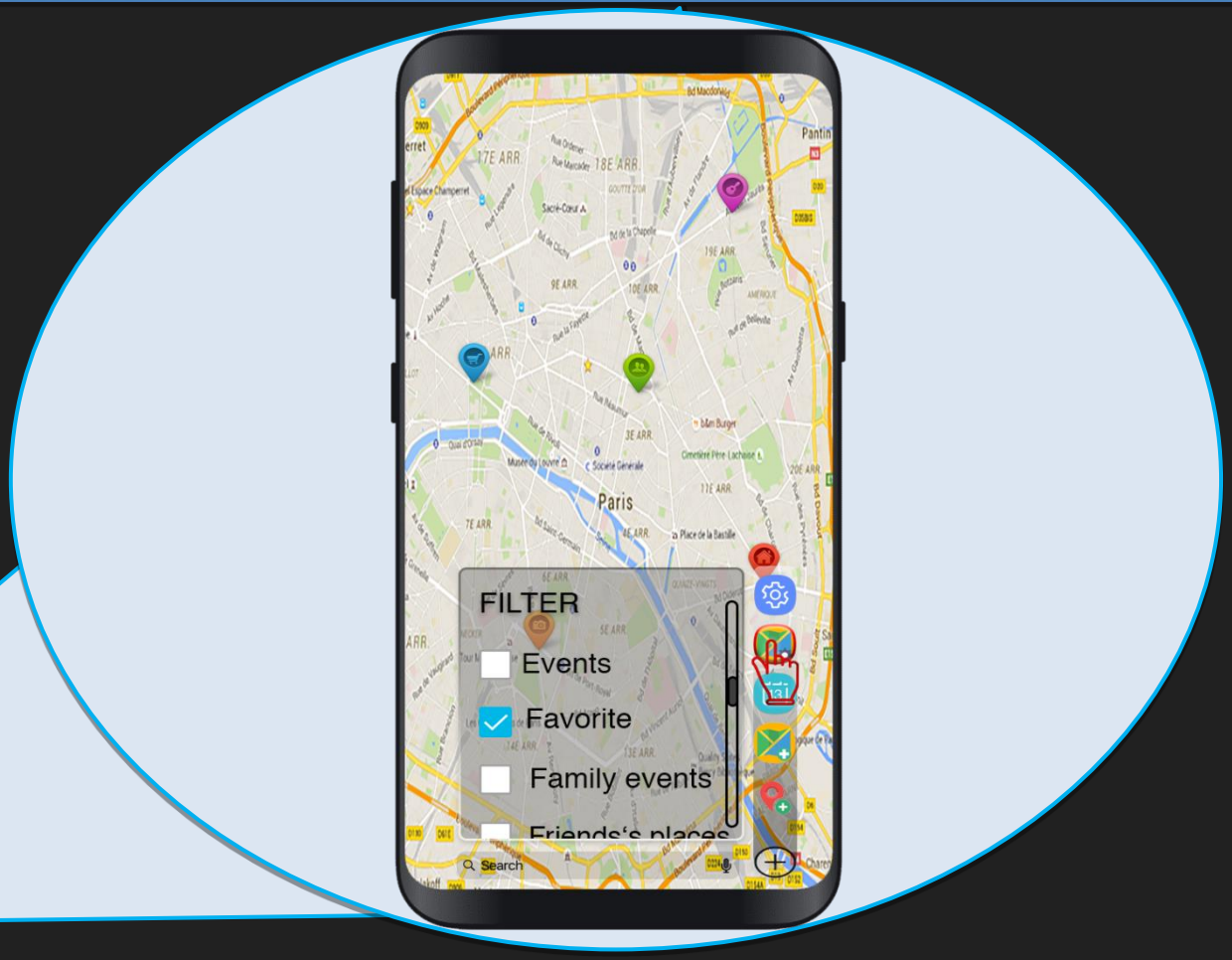



List of maps the user created.
He can select the maps he want to activate on the general plan, add or delete.

# Navigation Diagram




A quick-access to the filters of the current map.

# Backlog

| To do | Priority | Difficulty |
|---|---|---|
| Manage place, events : Add, delete or edit them. | 5 | 3 |
| Manage multiple maps : Add, delete, edit, share them. | 4 | 4-5 |
| Filter by category : You can select a category to display on your map. | 4 | 5 |
| Login/Logout protocol. | 3 | 2 |
| Friendlist :Add a friend, delete a friend, share with them. | 3 | 1 |
| Pictures : Add, delete, comment, share. | 3 | 3 |
| Messages/Comments : Add, delete, edit, share. | 3 | 3 |
| Pathfinding : Find the shortest path between two points. | 2 | 2 |
| Search for a map (from friend or a public one) | 2 | 2 |
| Search for a place (by name or by coordinates) | 2 | 2 |

# Web Services

| Type | URL | Behavior |
|---|---|---|
| GET | /list | Returns a list of maps |
| GET | /{id_map}/{id_place} | Returns detail of a place / event for the corresponding «id» |
| GET | /{id_map} | Returns detail of a map for the corresponding «id» |
| POST | /{id_map}/{id_place} | Modify a place / event for the corresponding «id» |
| POST | /{id_map} | Modify a map for the corresponding «id» |
| PUT | / | Add a new map |
| PUT | /{id_map}/ | Add a new place / event |
| DELETE | /{id_map} | Delete a map |
| DELETE | /{id_map}/{id_place} | Delete a place / event |
| PUT | /{id_map}/{id_place} | Add a picture to an event or place |
| DELETE | /{id_map}/{id_place}/{id_picture} | Delete a picture |