

# Software Engineering Project Presentation (*Group 1*)

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](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

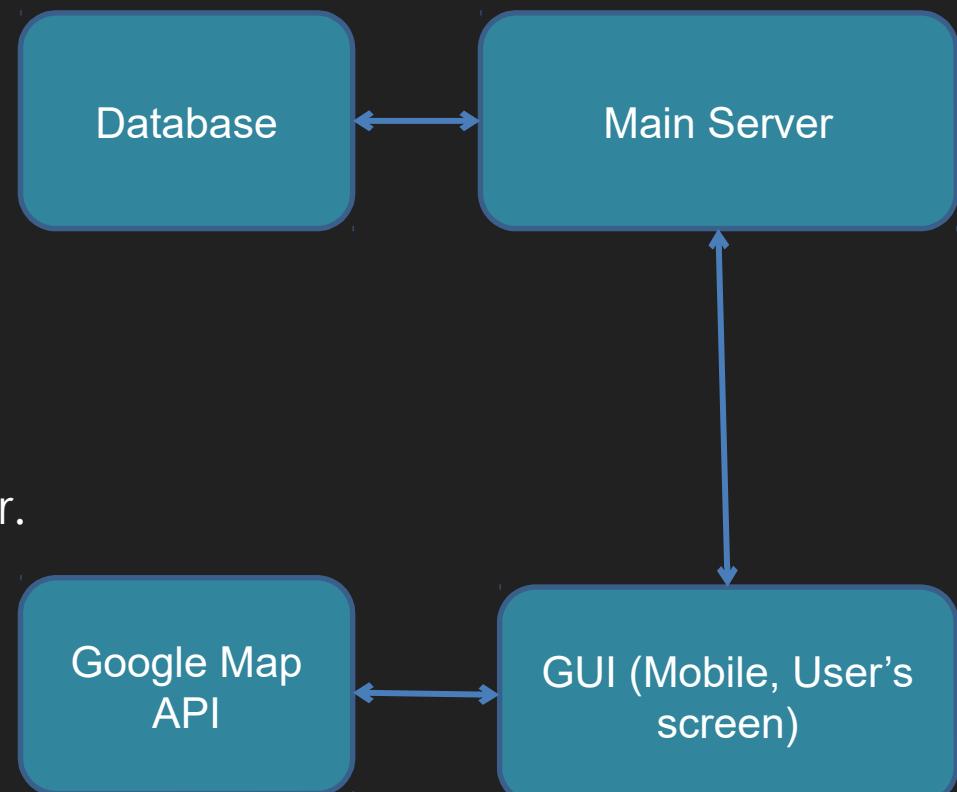
The business objects would be :

- 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.

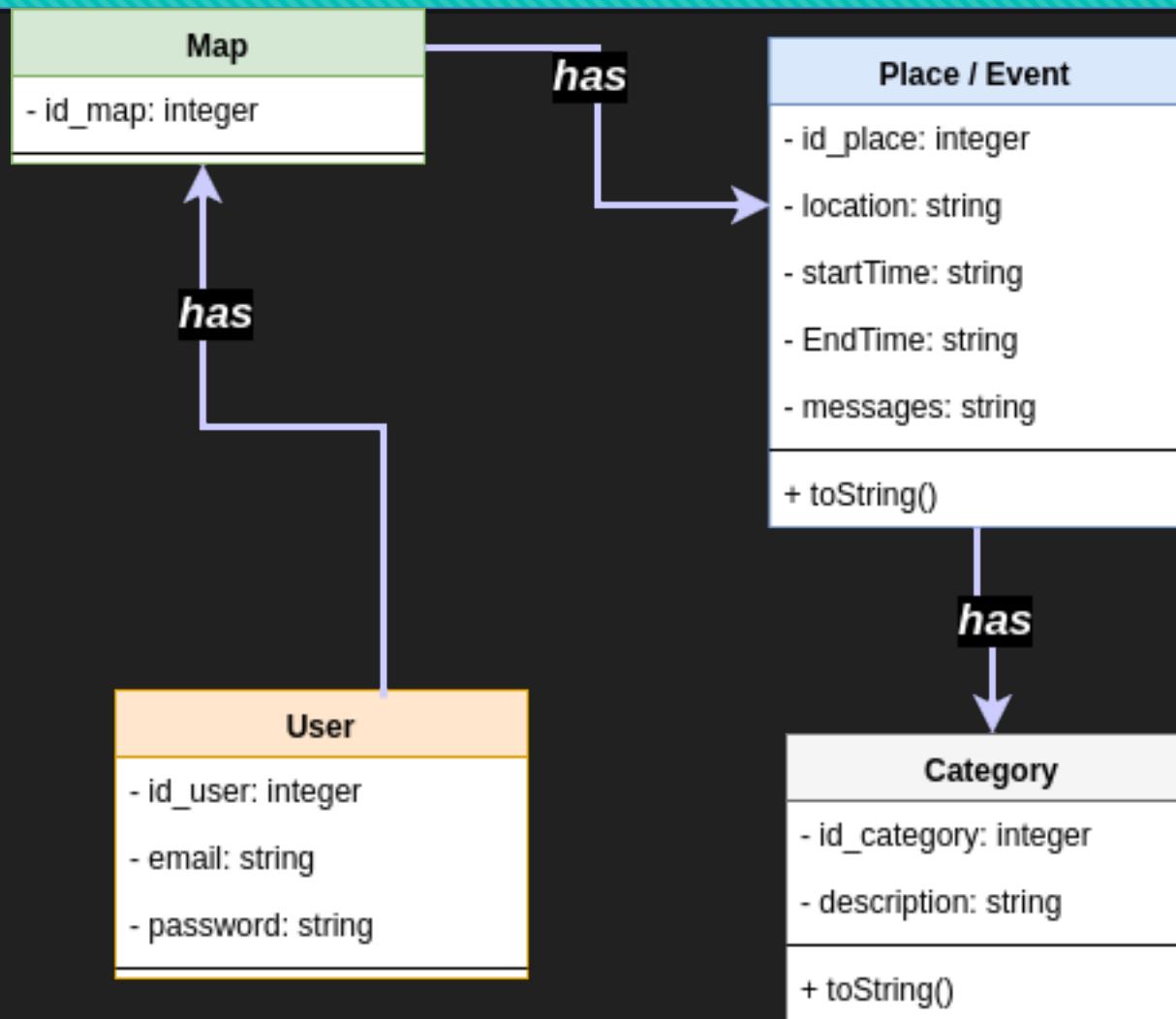
# Technical Proposition

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**

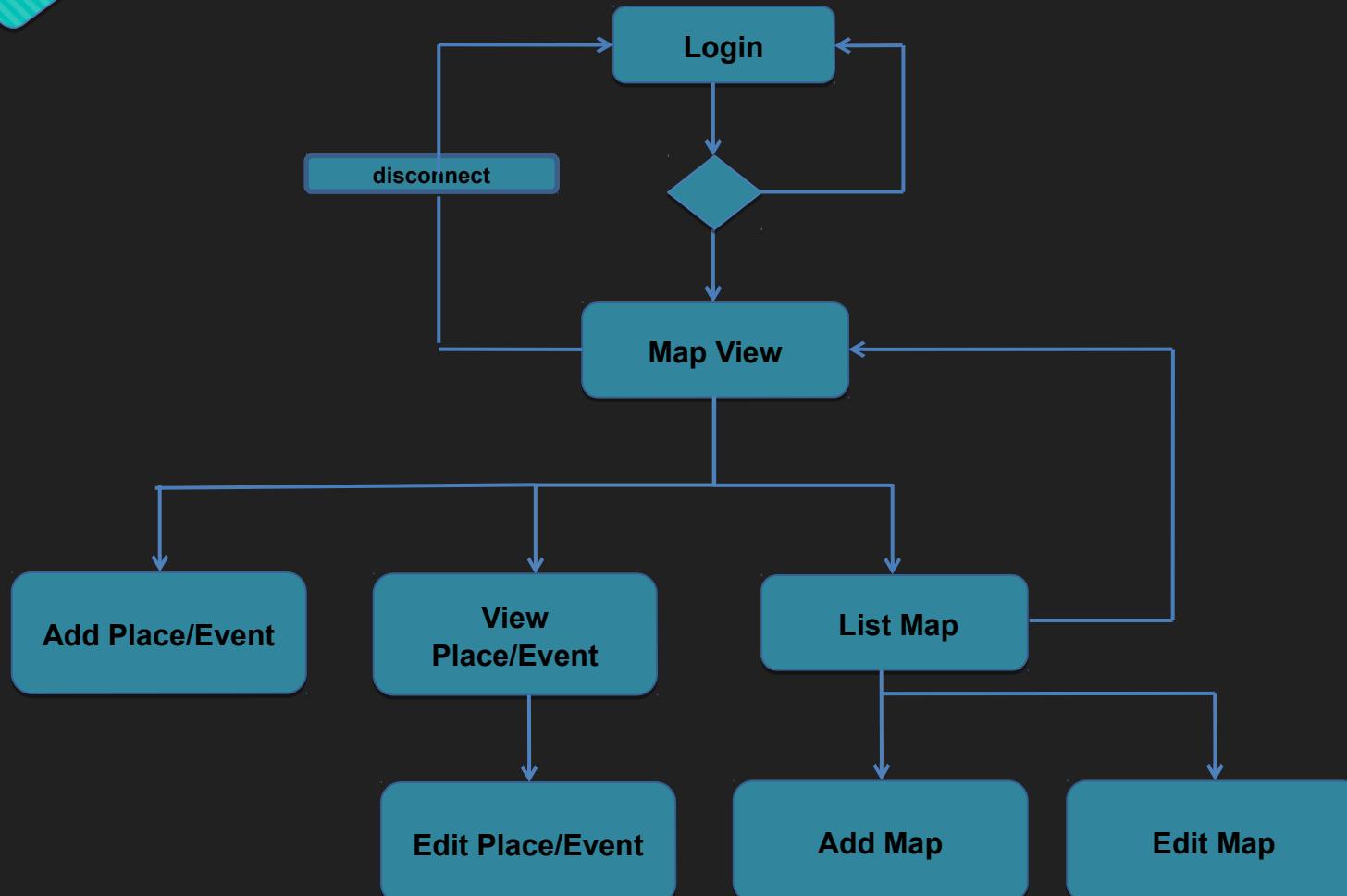
**Efficiency**

**Simplicity**

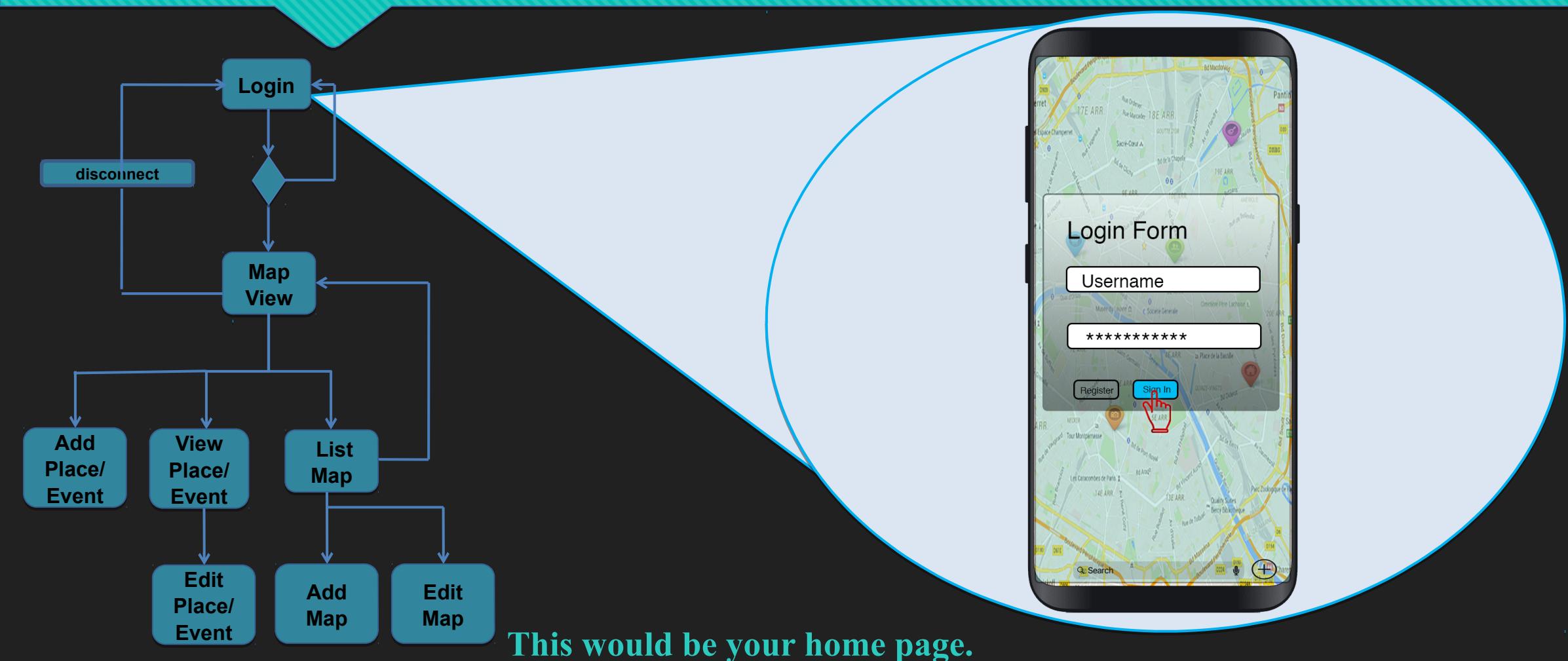
**Easy handling**



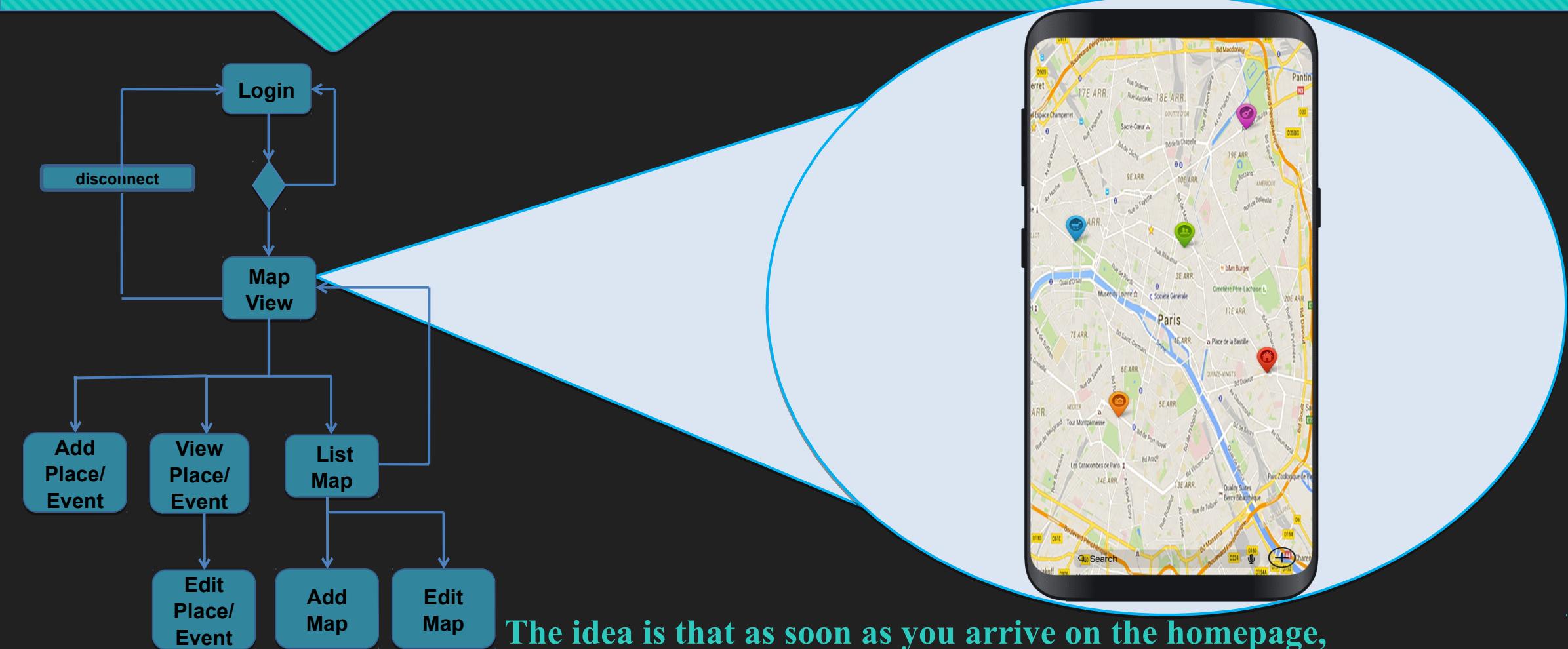
# Navigation Diagram



# Navigation Diagram



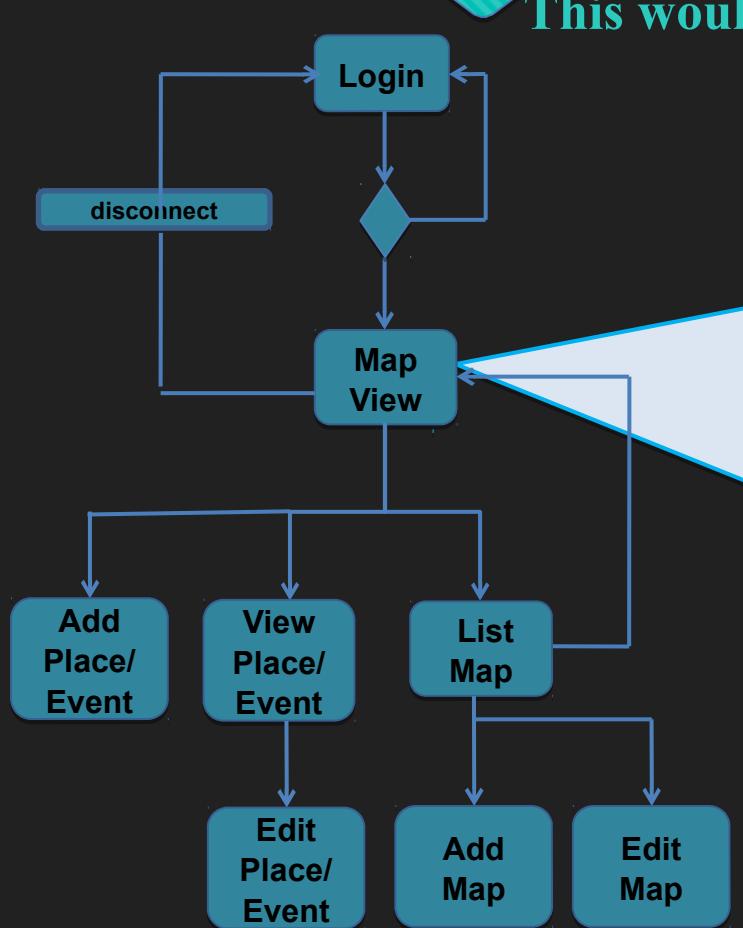
# Navigation Diagram



# Navigation Diagram

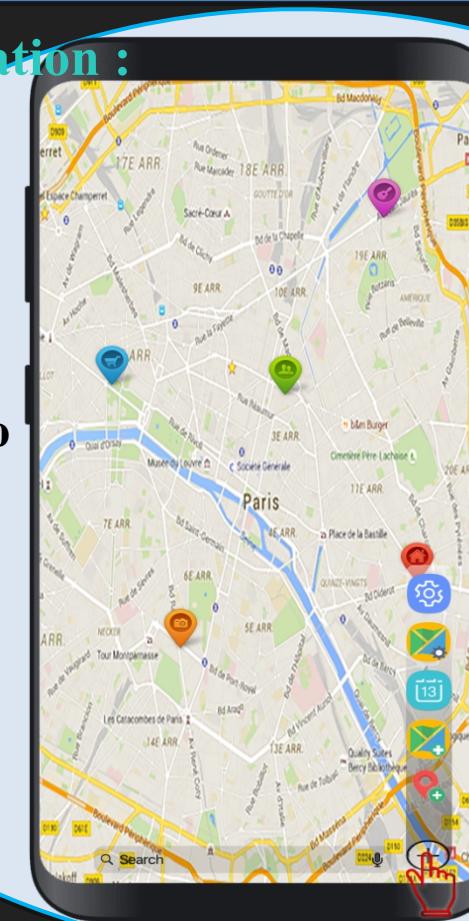


# Navigation Diagram



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.

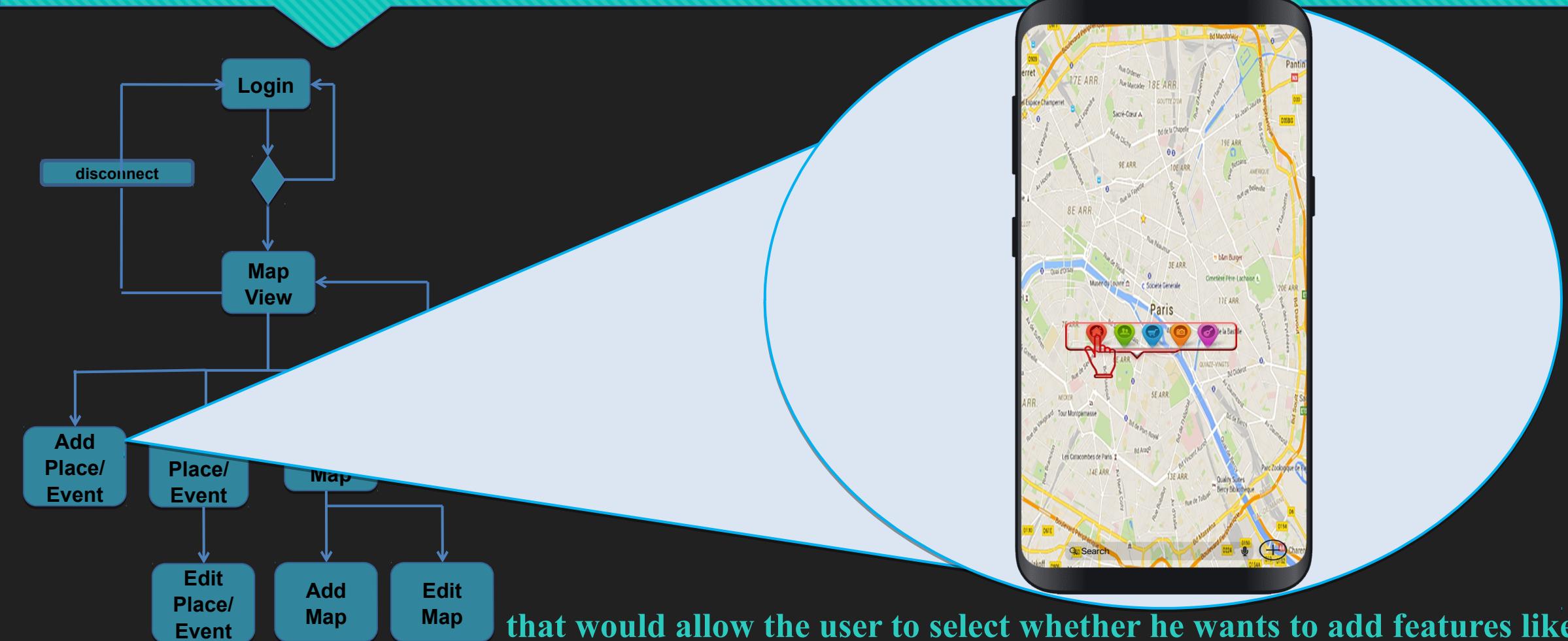
# Navigation Diagram



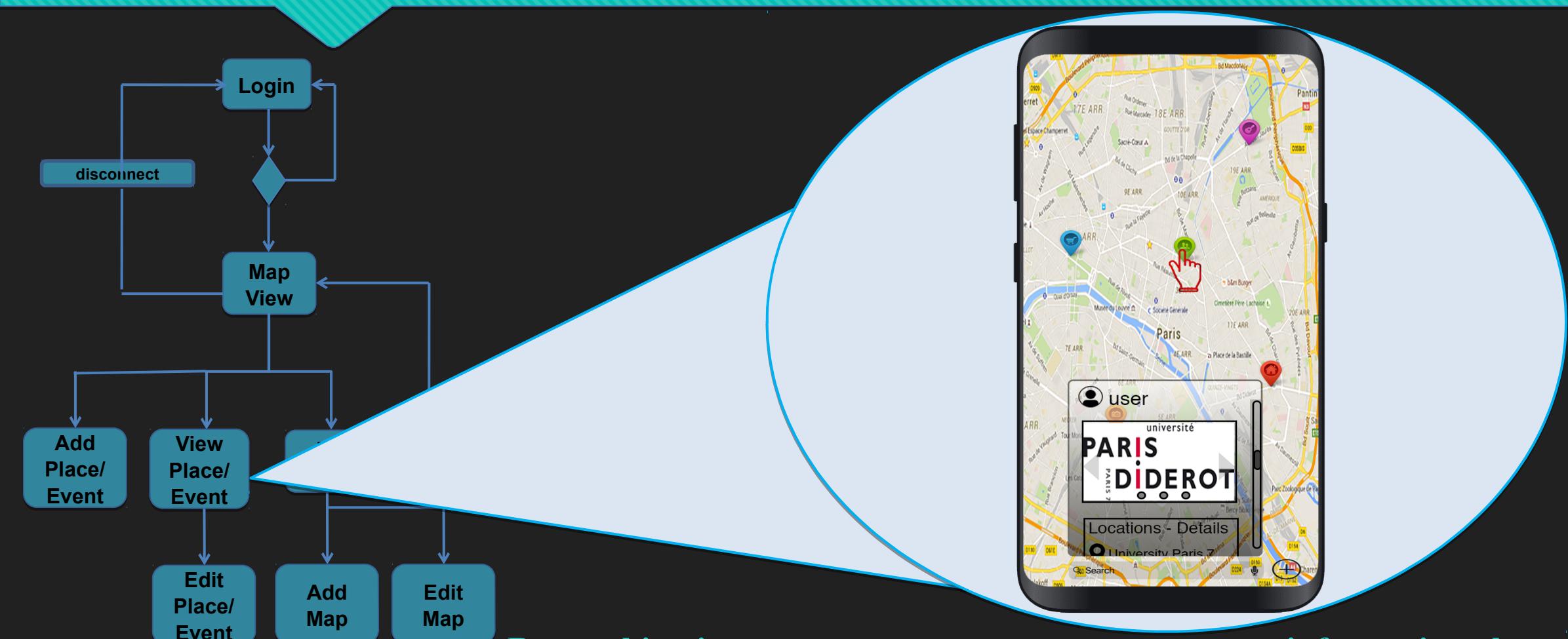
# Navigation Diagram



# Navigation Diagram



# Navigation Diagram

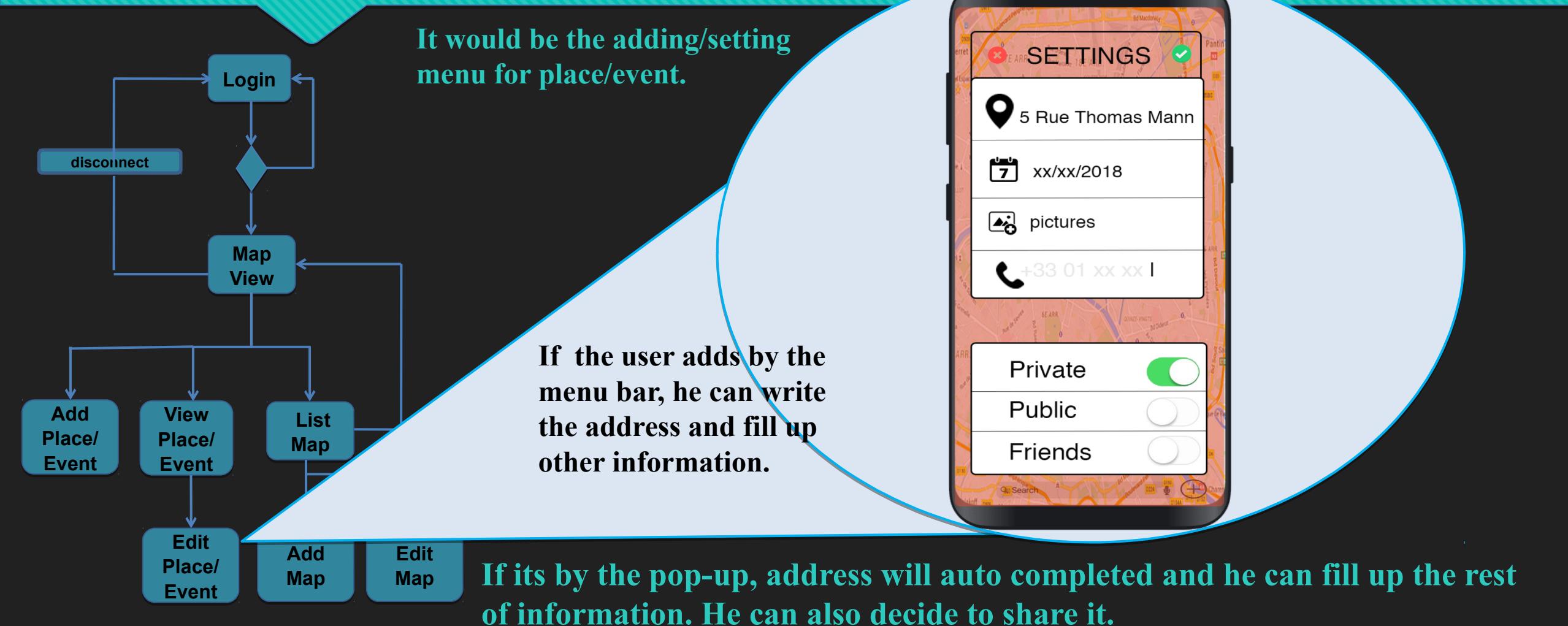


By touching icon on map, we can get access to more information about...

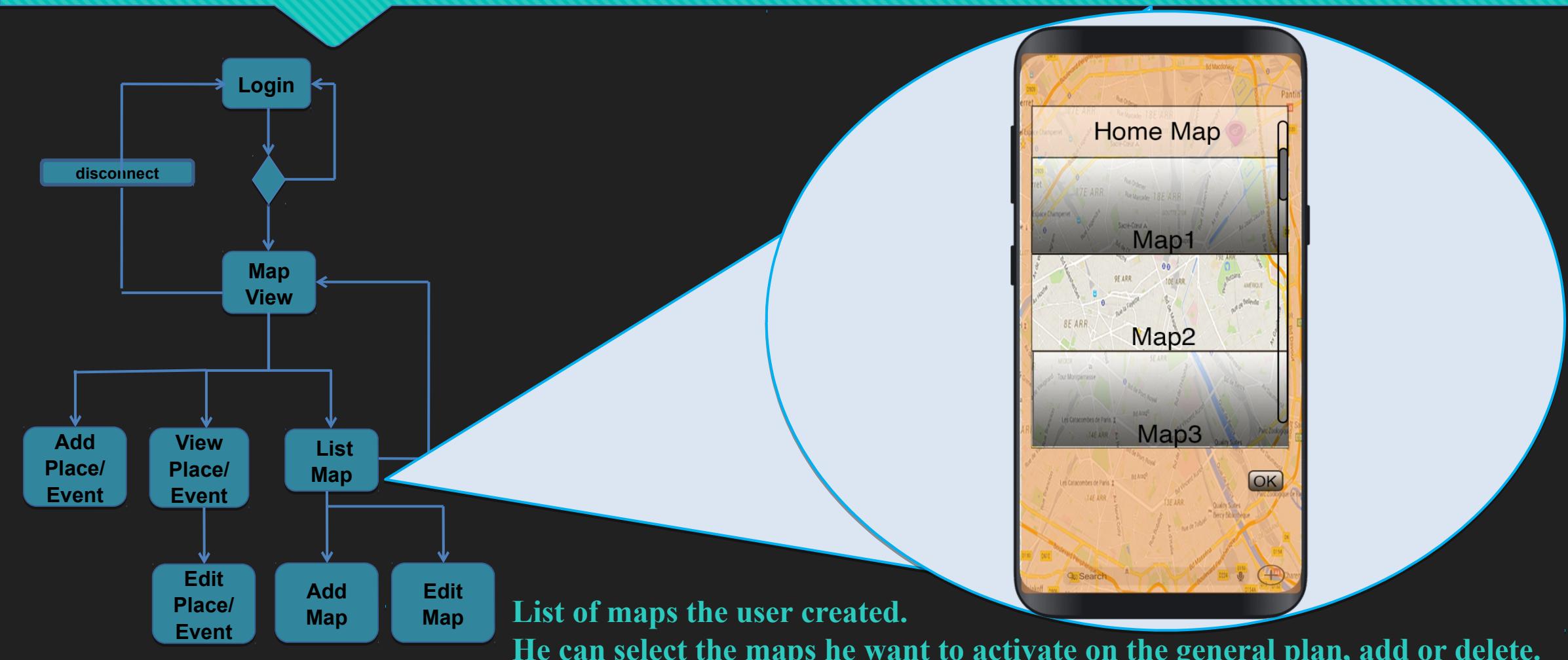
# Navigation Diagram



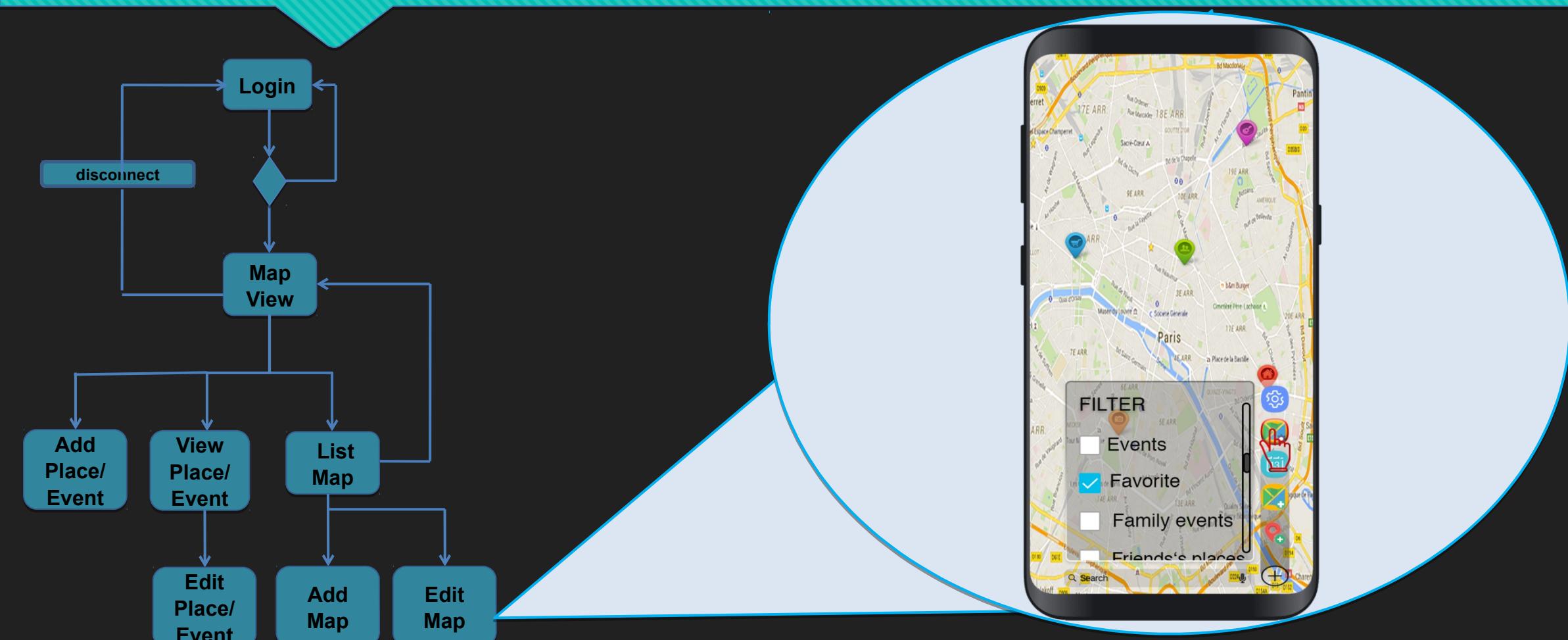
# Navigation Diagram



# Navigation Diagram



# 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