**VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY**  
**INTERNATIONAL UNIVERSITY**  
**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**



THESIS REPORT

**Nguyễn Thành Thiện**

**ITITIU14089**

**SPORT EVENTS MANAGEMENT MOBILE APP**

by

**NGUYEN THANH THIEN**

***Submitted to:*** School of Computer Science and Engineering - International University, VNU-HCM

APPROVED BY SUPERVISOR APPROVED BY: COMMITTEE

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tran Thanh Tung, Ph.D. Assoc. Prof. Chair Tran Manh Ha,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Thesis committee**

# **ACKNOWLEDGMENTS**

First of all, I want to express my most profound appreciation for Mr.Trần Thanh Tùng, who has carefully guided me and also pushed me for the best in the project. He is my advisor at the university. I was impacted by his advice on my career path, on the industry and my next step after graduating. His knowledge and experiences helped me a lot on the way of completing my thesis.

Subsequently, I want to thank my family, my girlfriend, who always support me in the life. They still at my back and encourage me every time I was down mood.

Furthermore, I would like to express my gratefulness for my friends. They helped me very much to make me become more perfect.

I want to thank all of my colleagues, they helped me very much in the job and also give me many practical experiments for my ultimate way.

**TABLE OF CONTENTS**

**Chapter I. INTRODUCTION**

**1. Objective**

**2. Problems and Solutions**

**3. System requirements**

3.1/ React (React Native and ReactJs)

3.2/ Why React Native

*3.2.1/ Saving time and reducing development costs.*

*3.2.2/ Performance*

*3.2.3/ Reusability*

*3.2.4/ Maintainability*

**4. Goal**

**Chapter II. BACKGROUND**

**1. Flux and Redux**

1.1/ Flux

1.2/ Different of Flux with MVC

1.3/ Redux

**2. Firebase**

2.1/ Cloud Function

2.2/ Real time database

2.3/ Authentication

**3. Expo**

3.1/ Expo CLI

3.2/ Expo components

3.3/ Expo Push Notification

**4. Open source**

4.1/ react-native-maps

4.2/ react-native-elements

4.3/ geolib

4.4/ react-native-scrollable-tab-view

4.5/ material-ui

**Chapter III. SYSTEM ARCHITECTURE**

**1. Use case by user-stories**

1.1/ User (visitor, member).

1.2/ Owner, Administrator.

**2. Diagrams**

**Chapter IV. IMPLEMENTATION**

**1. Code Functions**

1.1/ Mobile application

1.2/ Web UI application

**2. Result**

1.1/ Mobile application

1.2/ Web UI application

**Chapter V. CONCLUSION**

**1. Experiences**

**2. Conclusion**

**REFERENCES**

**LIST OF FIGURES**

**I. INTRODUCTION**

**1. Objective**

With the development of society, the application of modern technology is more and more popular in our world, especially of using phone in daily life. Today's phones can meet most user requirements, from keeping contacts, taking photos/videos to serve entertainment purpose such as music, games, etc.

One of the benefits of a modern phone (smartphone) is that users can select their favorite applications and use them independently as regular web applications.

Applications that are the most popular have basic requirements such as tracking the status of an in-app object, finding an object, controlling personal information from the application and the interaction between users and apps, etc.

With my product (called "Sport News"), you can use the functions needed for specific sport community events in your area.

**2. Problems and Solutions:**

*\* Problems:*

- With customer:

+ There are too less applications that used to track community sports events in smartphone.

+ It is difficult to keep tracking favorite kind of sports.

+ The event positioning system has not been applied.

+ Review tournaments with pictures is hard.

- With owner:

+ Upload event into native mobile application is limited.

+ Hard to manage the events in smartphone.

*\* Solution:*

- Realtime Push Notifications about the event into customer’s phone.

- Capture their moment using to review the event.

- Tracking events by real time map and GPS.

- A web application for event’s hosts.

**3. System requirements**

**3.1/ React (React Native and ReactJs)**

React was introduced and developed by Facebook, the biggest community network in the world. React was created to optimize the DOM handling (by saving the state of an application and just change the DOM when state changed). By this way, it is really helpful with constant data.

Beside that, React Native is a framework that lets you build native iOS and Android apps with one unique language of Javascript. React Native provides a virtual DOM (Document Object Model) which is a generic representation of what your UI will look like independent of the two platforms. React Native then renders the native control based on your UI markup representation which then renders native controls. It is able to take native platform components (sliders, switches, labels, tab bars) and wrap them in React component counterparts.

**3.2/ Why React Native**

***3.2.1/ Saving time and reducing development costs.***

*\* Easy to learn*

- Getting started with React Native is easy - especially for JavaScript professionals. Just download the open-source code from Github and make sure you’re familiar with a few tools and constructs in the React Native library including NodeJS, the CSS Flexbox system, ECMAScript 6, and JSX.

### *\* Cross-platform*

- React Native was originally only developed for iOS support, but due to its success and popularity, Facebook decided to develop support for Android as well, so which translates to lower development costs.

*\* The ability to combine good features and build apps*

- Whereas native app development is usually associated with inefficiency, less developer productivity, and slower time to deployment, React Native brings the speed and agility of web app development to the hybrid space with native result (the combination between pros of Hybrid and Native apps).

- React Native provided "Hot Reloading" (the previous version is “Live Reloading”) that keeps the app running and to inject new versions of the files that you edited at runtime. By this, you don't lose any of your state which is especially useful if you are tweaking the UI. The result is times between you save a file and be able to get this feedback loop can be under 1 second, even as your app grows.

***3.2.2/ Performance***

- Traditionally, “Hybrid” or “Mobile web” apps that work on both iOS and Android tends to go down for more complicated apps. Apps built through React Native are compiled into natively written code, so they not only work on both operating systems, but also function the same as a natively written app (**“learn once, write anywhere”**).

- React Native is focused solely on building a mobile UI, making it more like a JavaScript library than a framework. It is developed based on Facebook’s popular ReactJS UI library for web applications, so it brings all of ReactJS’s better app performance: DOM abstraction, and simplified programming methods to hybrid mobile development. The resulting UI is highly responsive and feels fluid (the app will have quicker load times than a typical hybrid app, and a smoother feel) thanks to asynchronous between Javascript and Native.

***3.2.3/ Reusability***

- Reusability is key in React Native. That mean the building blocks are treated as native components that can be compiled directly into the native languages (Objective-C for iOS and Java for Android). This is a huge bonus for businesses that want to augment an existing app but don’t want to overhaul it by Incorporate React Native components into the app’s code.

- This is possible to exclude the WebView components of other Hybrid mobile apps because React Native’s building blocks are reusable “native components” that compile directly to native. For example, if you’re adding Google Maps functionality to your app, React Native lets you link the plugin with a native module without have to rely on a WebView for certain functions. Components you would use in platforms have counterparts right in React, so you’ll get a consistent look and feel.

***3.2.4/ Maintainability***

- React Native uses Javascript and this, coupled with the intuitive architecture of the framework itself, allows engineers to jump to and from each other’s projects on Native apps with relative ease.

- The typical workflow would consist of writing most of the code on React Native and then using the native languages to optimize certain elements of the app. Parts of a specific app written in React Native will have no trouble combining with parts written in the native languages for iOS or Android. It is completely up to the developer to build the app, it not only can the native languages of an OS be used to optimize an app, they can also be used to write parts of the app.

**4. Goal**

After developed the project, I want to gain the needs that:

- Having an application for user to track sports community events (in both client-server sides).

- Using React to build application (on both smartphone and web environment) with smoothly.

- Applying Firebase into future projects as a steady server to store and analyze data.

**II. BACKGROUND**

**1. Flux and Redux**

**1.1/ Flux**

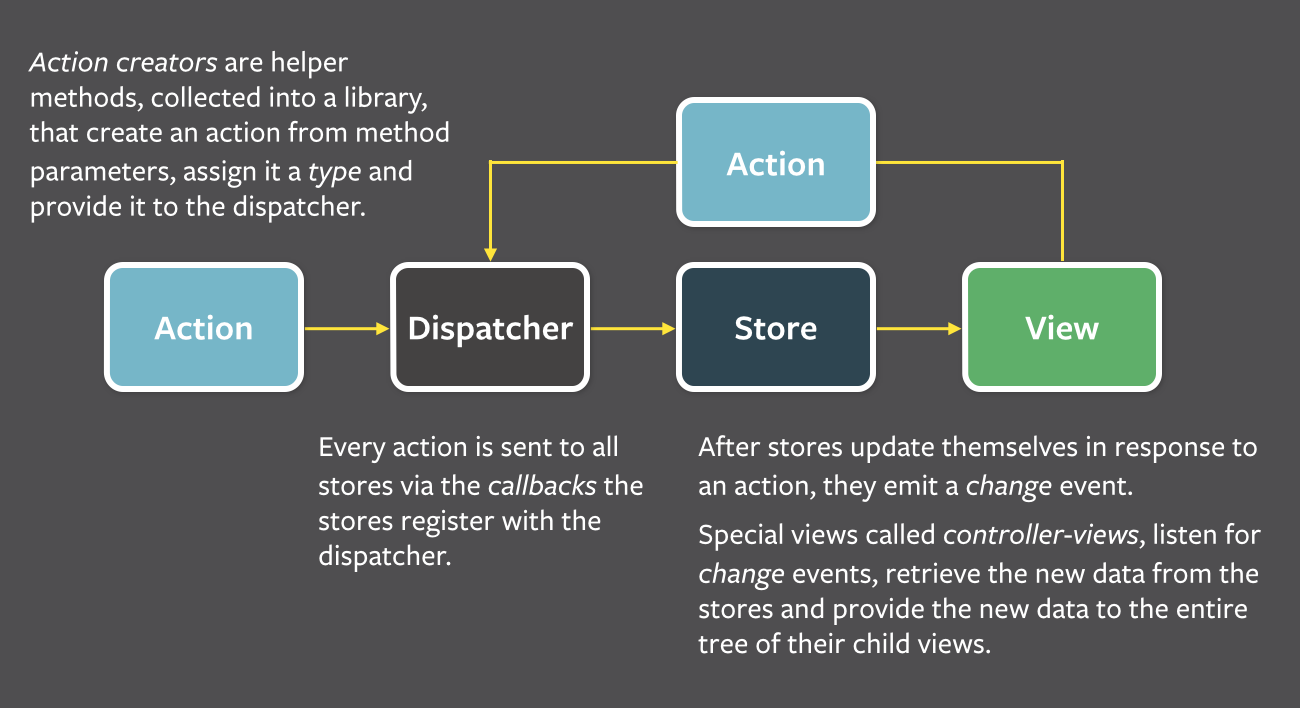
- Flux is an architecture for creating data layers in JavaScript applications. It was designed along with the React view library. It places a focus on creating explicit and understandable update paths for your application's data, which makes tracing changes during development simpler and makes bugs easier to track down and fix.

- Flux consists of the following basic components:

**+ Action:** This is the place to register the functions that will be called when View needed.

**+ Dispatcher:** Has the role of transmitting calls from Action to the Store. When an Action is called, the Dispatcher will broadcast an event to all Stores, with Action Type, or any other required data.

**+ Store:** The place to store data, is also the only place where you can add, edit or delete data. The store will listen to the events that come from the Action through the Dispatcher, check that the event is under its processing, and make the necessary data changes, corresponding to each event. After changing the data, Store will give another event to report its change.

**+ View:** Retrieves data from functions.

*Figure – Flux architecture*

- React using **one-way data bindings**, because they think that two-way data bindings will become "When one object update, it many other objects will update, and then makes more updates". With one-way, when all the changes go to the Dispatcher, they can easily find where the change comes from, so the system will become "predictable".

- Flux follows the concept of **Unidirectional Data Flow** (UDF is to keep the data flow in the application moving in a single direction. When the data changes, this stream reboot from scratch) making it much easier to zero in of where the error lies. The data goes through a strict pipeline through your application:

+ Any changes, or Actions, **must go** through the Dispatcher!

+ Store is just only public getter, not public setter. Data change handlers can only be called inside the Store itself. This means:

+ You cannot change the data in the Store directly from the View.

+ You also can not change the data in one Store from another. As mentioned above, Store is not public setter. To do that, you must go through Action and Dispatcher. This makes the data inside your Store more manageable, you can easily debug where the change comes from when data changes.

**1.2/ Different of Flux with MVC**

*\* Easier to understand the data flow.*

- In the bidirectional data flow, you have the typical data flow   MVC. But when applications became more complex, the Controller takes the huge responsibility of maintaining both the application state and the data. Also, the cascading updates makes the app really difficult to understand and debug.

- With UDF, changes in the application view layer will trigger an action in the data layer. These changes will then be reflected in the View. The View does not directly affect application data.

*\* Reducing the role of Controller.*

- Dispatcher is not a controller, it does not contain business logic. It simply is a "coordinating center", which send an Action to every Store. In the MVC model, you can design any Controller as you can. But in Flux, there is only one Dispatcher and all Actions must go through it. The appearance of the Dispatcher is important because it ensures the design of UDF.

*\* Store can handle many objects.*

- In the MVC model, the Model usually manage a particular object. In opposition to the Model, the Store may not handle any data or handle multiple states of the application, or manage multiple objects at once.

**1.3/ Redux**

- Redux is an implementation library of Flux with minimal API but completely predictable behavior, so it is possible to implement logging, hot reloading, time travel, universal apps, record and replay, without any buy-in from the developer.

- Redux can be used together with React, or with any other view library, and following 3 principles:

+ All application state is contained within a ***single store***, which is most often a JavaScript object.

+ The application's state is ***immutable***. This means that at no point should the object representing the state be modified in any way by any component.

+ All functions that compute the new state (called reducer functions) must be **pure functions**. Pure functions are functions that produce no side-effects and are deterministic - for a given set of inputs, the output will always be the same.

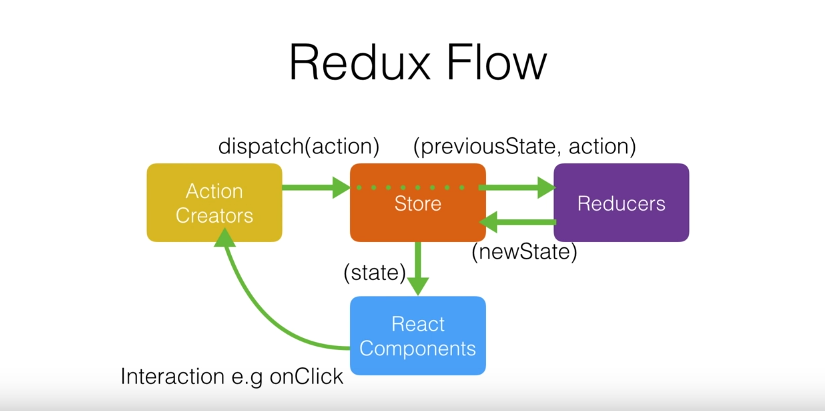
- General structure of Redux design pattern:

+ ***actions*:** Are payloads of information that send data from your application to your store. They are the only source of information for the store

+ ***reducers*:** Reducers specify how the application's state changes in response.

+ ***stores*:** The actions that represent the facts about “what happened” and the reducers that update the state according to those actions. The Store is the object that brings them together.

+ ***components:*** Folder contains the components that is only responsible for view and receive user interaction (How things look: markup, styles). These files read and change data from *props* => Dump components.

+ ***container:*** The components that are aware of Redux, Router, etc. They are more coupled to the app. They transmit the data to dump components with *props* (How things work: data fetching, state update) => Smart components.

*Figure – Redux life cycle*

**2/ NodeJs**

**2.1/ Definition**

Node.js is a great library environment for executing JavaScript functions outside the browser. It is also a helpful multi-platform to create a web applications and server-side. Node.js provides a great model which makes it become very lightweight, and can do smoothly with information serious. It also has a rich JavaScript's library modules to decrease the complication of applications.

**2/ Firebase**

**2.1/ Cloud Function**

- Cloud Function will trigger a function (backend code) to run HTTPS requests, and Firebase features automatically. It likes a tool help to manage your server, and your code is stored in Google's cloud.

***2.1.1/ Definition***

- Cloud Function is an event-driven computing service and does not need a server. Instead of having basic infrastructures such as servers, storage and other resources, developers can use Cloud Function to implement and initialize functions in Google's public cloud.

***2.1.2/ Using Cloud Function***

- A Life cycle of function includes:

+ Define a new function (two support languages for writing are Javascript and Typescript) -> Choose an event provider -> how to execute the function.

+ Deploy function, Firebase will automatically connects to provided event.

+ When function's conditions is called -> functions is deployed -> The code is invoked.

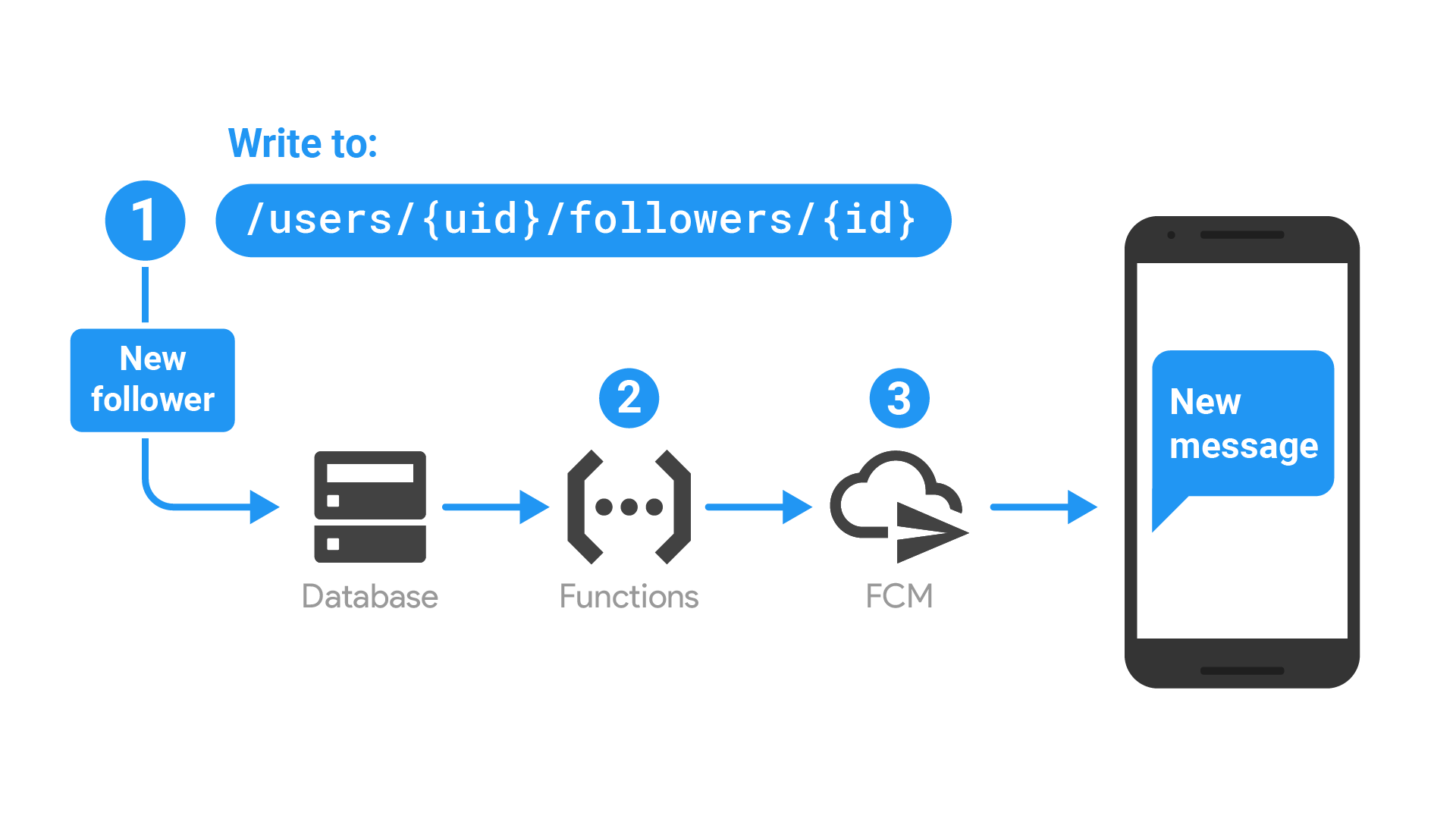
+ Google will handle the sessions of function, if there are many events -> create more instances and vice versa.

+ When function is updated, all old versions will be replaced by the new one.

+ When the function is removed, all things will be cleaned up, including the connection between function and provided event.

\* Usage of Cloud Function:

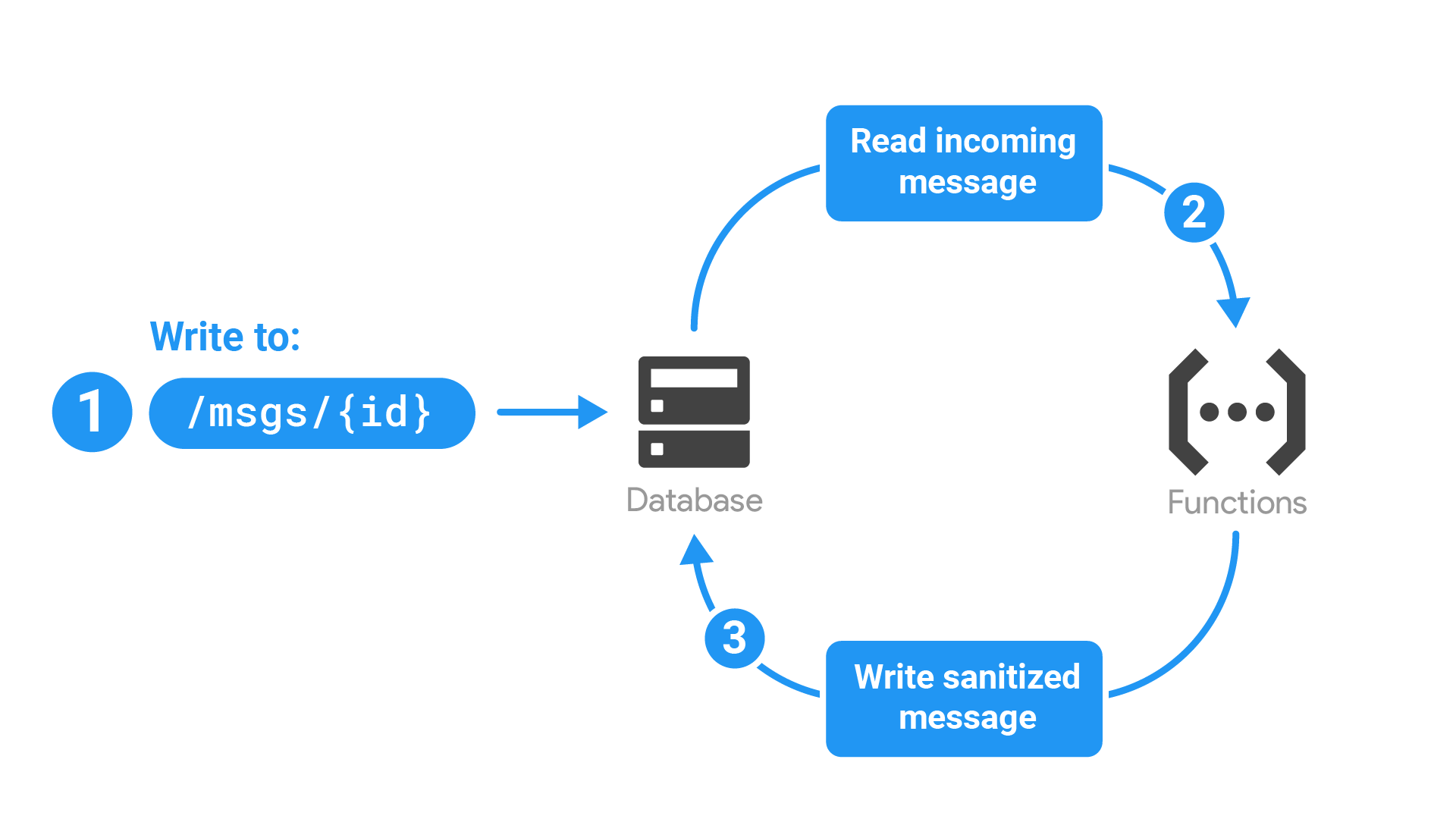
- Notify users when something interesting happens:



+ The function is triggered when database was updated.

+ Function execute to compose a message to send by using Firebase Cloud Messaging.

+ When Firebase Cloud Messaging is executed, a notification is sent to user’s device.

- Perform Realtime Database sanitization and maintenance

+ The function is triggered when database is written to store -> retrieving event data containing the text message.

+ Function execute to detect and handling text message.

+ Function send back an update to database.

- Execute intensive tasks in the cloud instead of in your app

- Integrate with third-party services and APIs

***2.1.3/ Why Cloud Function***

**2.2/ Real time database**

**2.3/ Authentication**

**3/ Expo**

- Expo is a set of tools, libraries and services help to develop mobile application using React Native.

- Expo uses React natives apps contain its SDK, which provides your accession to device's system functionally (camera, contact, local storage, GPS, etc.).

- It also offers simple UI components used for a mobile app such as Text, Touchable, View, etc.

- Expo SDK provides access to services and calls some general functions like Push Notification, Sign in with Google/Facebook, etc., and you can build native code within your React Native project using Expo efficiently.

**3.1/ Expo CLI**

- Expo CLI (command-line interface) is a developing tool associated with UI and Web Browser, which can show the project's logs and device information to manage and develop the application.

- To use Expo CLI, you first must have NodeJS (version 6 or newer) in your environment.

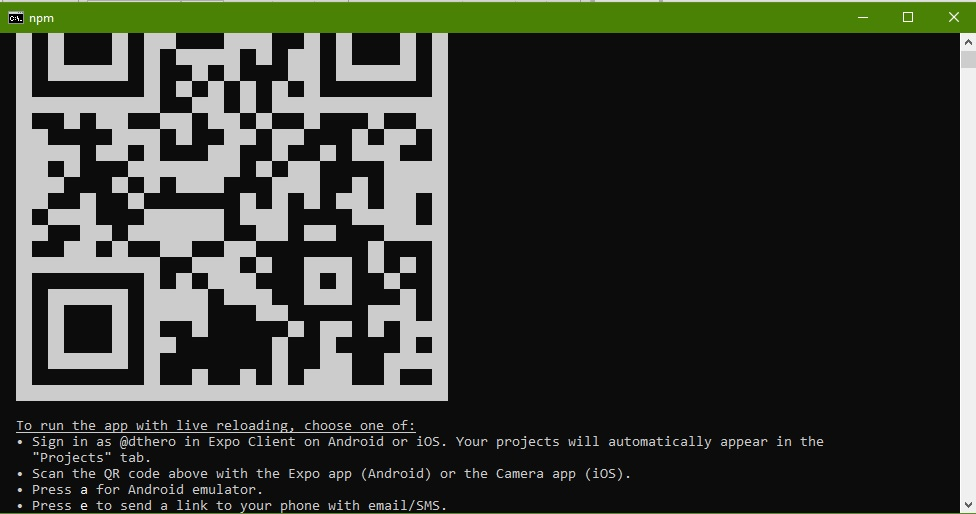
- First install Expo CLI, run command:

npm install -g expo-cli

- After installed Expo CLI, and start an Expo project by run following commands:

expo init *<project\_name>*

cd *<project\_name>*

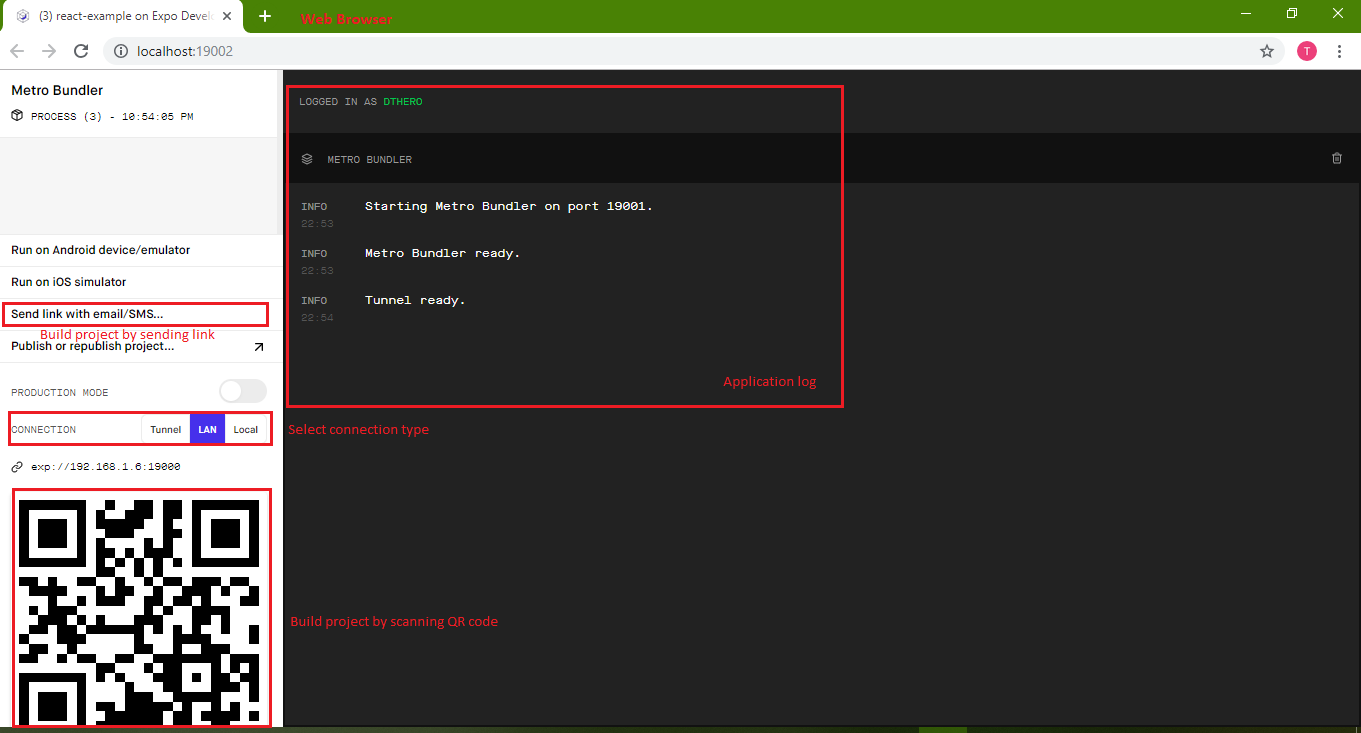
expo start

*Figure – Expo CLI after ran start command*

- A new Web Browser will show up for you with options to run project:

+ Run by the real device. In target device, install Expo client app from play store and open it, then we can build the project on the device via phone number (send build link into phone number as a message) or scan QR code on Web Browser.

+ Run by simulator. You must have a virtual device on your build environment and connect to build tool.

*Figure – Expo Web Browser UI*

**3.2/ Expo log**

- Writing to the logs in an Expo app works just like in the browser: use console.log, console.warn and console.error. Note: Expo does not currently support console.table outside of remote debugging mode [1].

**-** If you use Expo CLI, bundler logs and app logs will both automatically stream as long as your project is running. To stop your project (and end the logs stream), terminate the process with ctrl+C [2].

**3.3/ Why Expo**

- Expo makes code project easier to deploy, maintain and execution. When you don't want to use Expo, eject your project from the Expo Client without effect to code running.

- Pros of Expo:

+ Over the air updates.

+ An easy and more reliable way of upgrading React Native versions.

+ Limitation of Android Studio/XCode configurations.

+ Certificates management with Apple or Keystores management with Google.

+ Easy to integrate with other services.

+ Sending Push Notifications.

+ Additional, easy to use modules such as Authentication Session or Blur View.

**4/ Open source**

**4.1/ react-native-maps**

- react-native-maps is an open source for using map view in React Native. It includes many useful component APIs to interact between user and map.

- You can integrate react-native-maps with google map by using Google API key and add it into your application.

**4.2/ react-native-elements**

- react-native-elements is a collection of pretty component UI using in React Native such as Button, Input, Checkbox, etc. it also includes a set of complex components and makes as only one component like Rating, Search Bar, Avatar, etc.

- react-native-elements helps to make an application using React Native look more beautiful and more natural to develop complex components.

**4.3/ geolib**

- geolib is a library developed entirely by Javascript and using calculation algorithms to estimate the result based on coordinates.

- geolib contains almost full of needed functions to get data with coordinates:

+ Calculate the distance between 2 points.

+ Get the centre point of the given list.

+ Verify that point satisfy conditions, etc.

**4.4/ react-native-scrollable-tab-view**

- react-native-scrollable-tab-view is a library written by Javascript-only for React Native. It uses navigate functions to define a navigation pattern on iOS and Android, and it helps navigation between components and pages become easier and makes UI look more clearly, suit with modern mobile application trend.

**4.5/ material-ui**

**III. IMPLEMENTATION**

**\* System requirements:**

- I will divide my group users into two groups:

+ User (visitor), member: Customers using my app on a smartphone.

+ Administrator, owner: The event’s hosts using web UI app to manage their events.

- My project will have two parts using React:

+ First is client side for user, and it is a mobile application built on multi-platform and include main features to the user.

+ Second is server side for event’s owner and administrator, it is a web UI application and contains some features help administrator can manage data and owner manage their event.

**1. Use case by user-stories**

**1.1/ user (visitor, member).**

1. Display Login page to register/login

|  |  |
| --- | --- |
| *As a* | visitor |
| *I want to* | See all information on Login page. |
| *so that I can* | Register new account or login to application. |

1. Register new account

|  |  |
| --- | --- |
| *As a* | visitor |
| *I want to* | Register new account |
| *so that I can* | Login into application. |

1. Login an account

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Login to application by my account. |
| *so that I can* | Use all features of application. |

1. Display homepage

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | see all information of Login page when open an app on smartphone. |
| *so that I can* | login by created account or register new account for using app. |

1. Navigate between tabs

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | see all information of Main page whenever I logged in successfully. |
| *so that I can* | navigate between tabs and see all information of each tab. |

1. Search events

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | use search tool. |
| *so that I can* | search keywords of the event on News tab and Tournaments tab. |

1. Display News page

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | see all information of events that are approved from administrator in form of list on News Tab. |
| *so that I can* | View new event created and refresh list event. |

1. Manage push notification

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Turn event’s notification on or off. |
| *so that I can* | Manage all notifications that I responded. |

1. Display event detail

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | see detail of event. |
| *so that I can* | enroll, comment or view detail about the event. |

1. Display map view

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | see map view on Map tab. |
| *so that I can* | Find all events near me |

1. Filter map view event by radius

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Filter map view by radius |
| *so that I can* | Find all events near me. |

1. Filter map view event by sport type

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Filter map view by sport type |
| *so that I can* | Find all events near me. |

1. Display personal information

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | see all information of user on Personal Info tab. |
| *so that I can* | Update and view all personal information. |

1. Take a photo

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Take a photo using my device’s camera. |
| *so that I can* | Update picture into application. |

1. Comment on event

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Comment on specific event. |
| *so that I can* | Update my comment into event detail successfully. |

1. Receive push notification

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Receive push notification from application. |
| *so that I can* | Know news about events. |

1. Re-render map when location change

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Filter map by changing my current location. |
| *so that I can* | Find all events near me. |

1. Change personal picture

|  |  |
| --- | --- |
| *As a* | user |
| *I want to* | Take or choose a picture. |
| *so that I can* | Update my avatar into new one. |

**1.2/ Owner, Administrator.**

1. Login to web UI

|  |  |
| --- | --- |
| *As an* | Owner/administrator |
| *I want to* | Login to web UI by my account |
| *so that I can* | Use all features of an event owner. |

1. Register new account

|  |  |
| --- | --- |
| *As an* | Owner/administrator |
| *I want to* | Register new account |
| *so that I can* | Login to web UI by created account. |

1. Create new event

|  |  |
| --- | --- |
| *As an* | owner |
| *I want to* | Fill all data to create new event. |
| *so that I can* | Create a new event successfully on database. |

1. Remove event

|  |  |
| --- | --- |
| *As an* | Owner/administrator |
| *I want to* | Remove a created event |
| *so that I can* | Delete event successfully from database. |

1. Update event

|  |  |
| --- | --- |
| *As an* | Owner/administrator |
| *I want to* | Update status of the event (Approved/Rejected/Pending) |
| *so that I can* | Update successfully to database. |

1. Filter event by alphabet

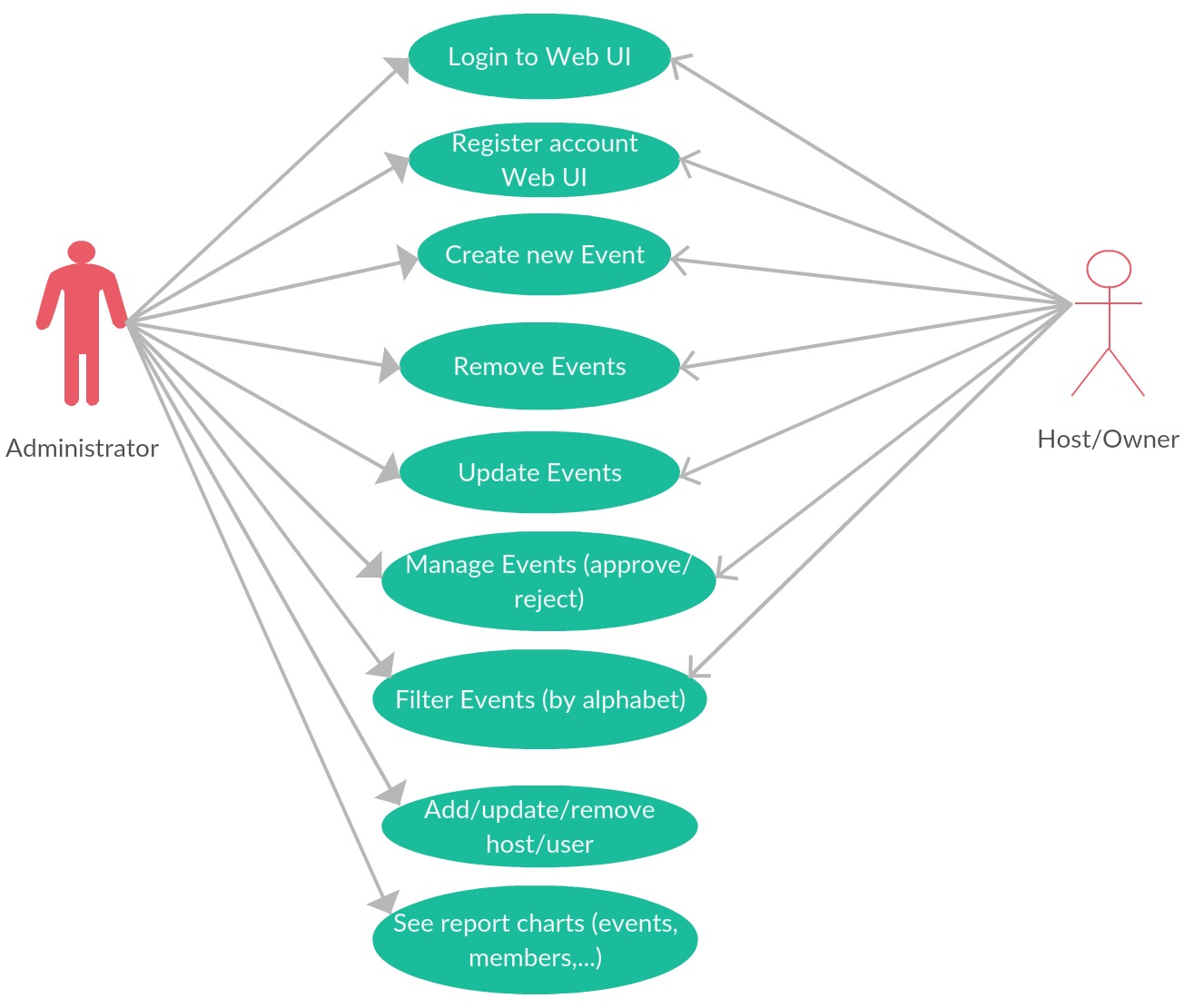
|  |  |
| --- | --- |
| *As an* | Owner/administrator |
| *I want to* | Filter list of events by alphabet |
| *so that I can* | See all events easily in Events tab. |

1. Change user

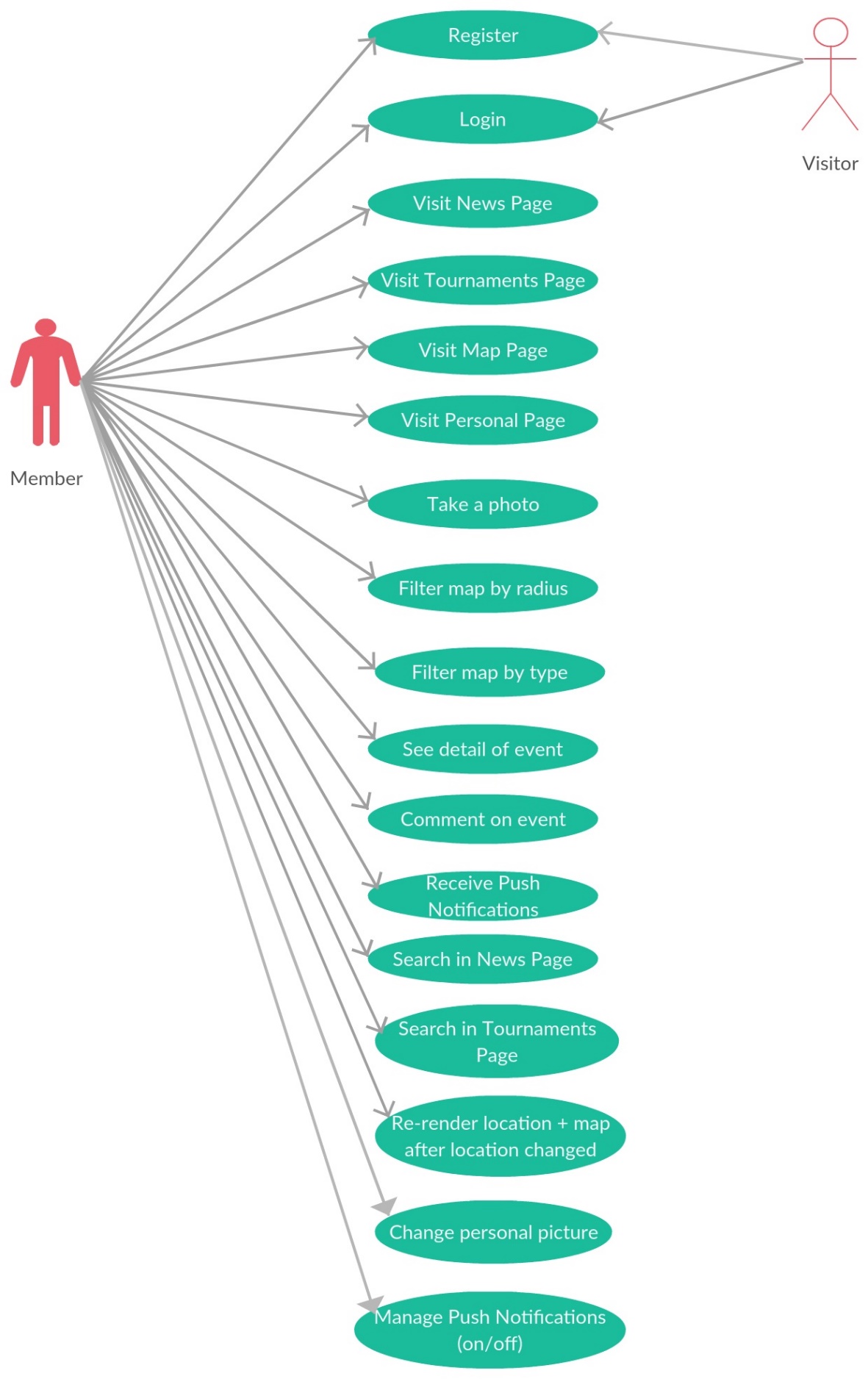
|  |  |
| --- | --- |
| *As an* | administrator |
| *I want to* | Add/update/remove user |
| *so that I can* | Update user successfully to database. |

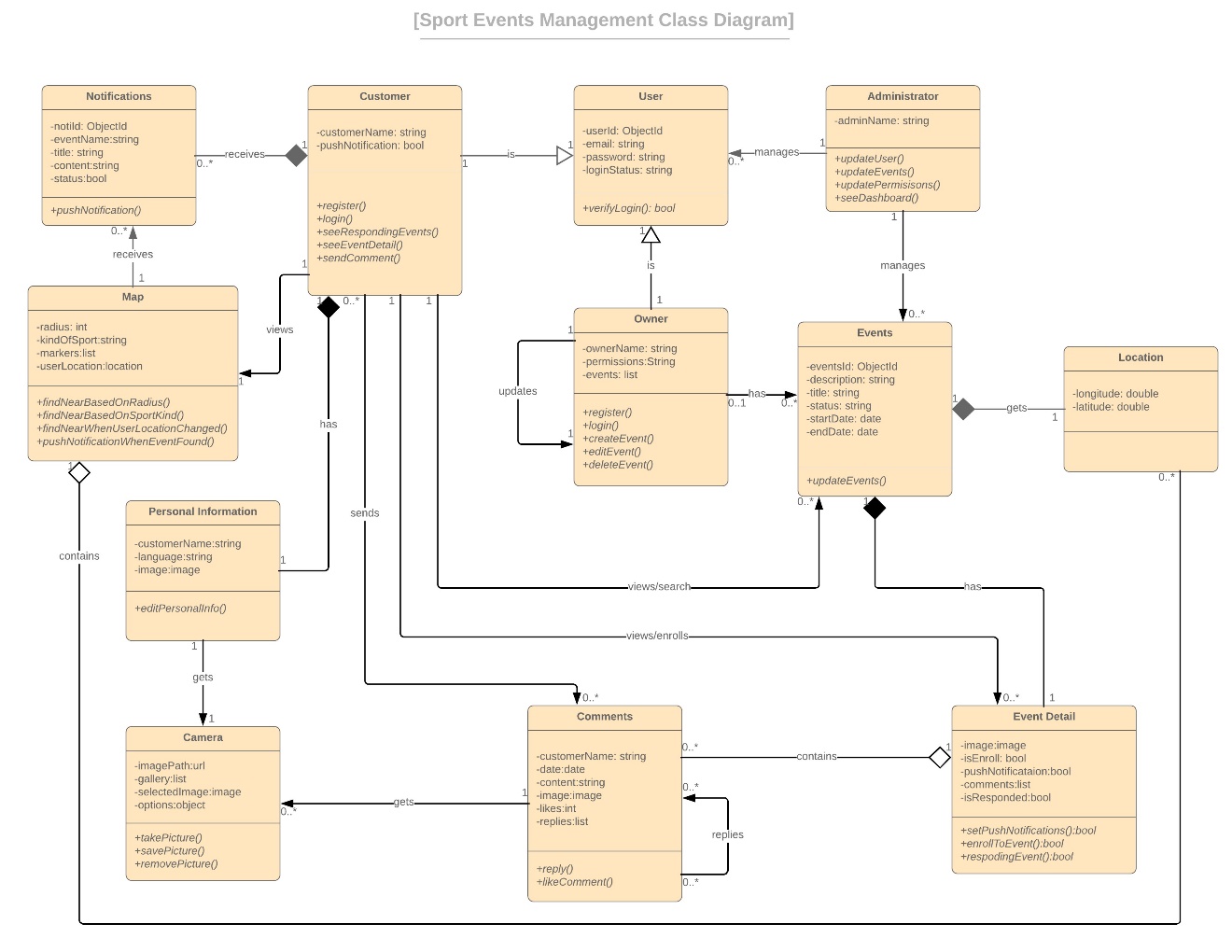
1. See Dashboard

|  |  |
| --- | --- |
| *As an* | administrator |
| *I want to* | See all dashboard chart of events management. |
| *so that I can* | Know the information about all events in database. |

**2. Diagrams**

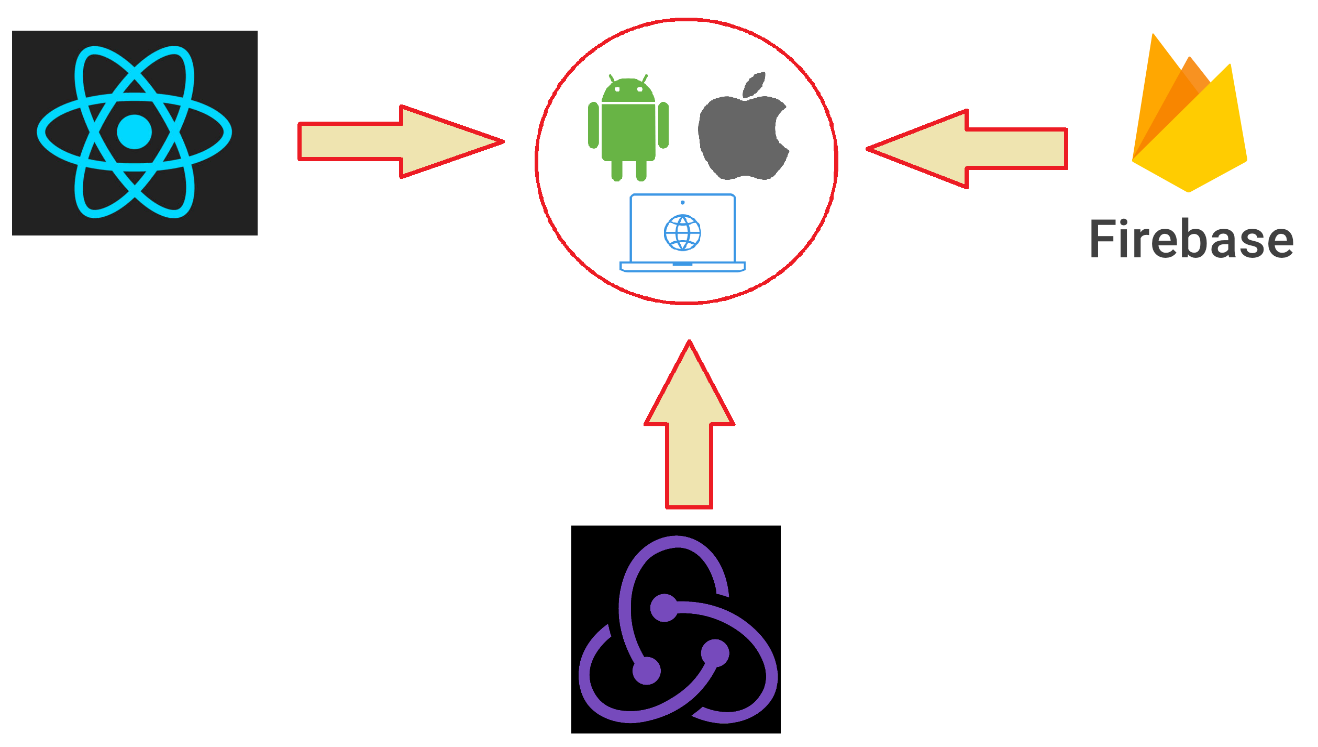
*Figure - Use case diagram for client*

*****Figure - Use case diagram for server*

*Figure - Sport Events Management Class diagram*

**III. IMPLEMENTATION**

\* In my project, I apply technologies following these layers:



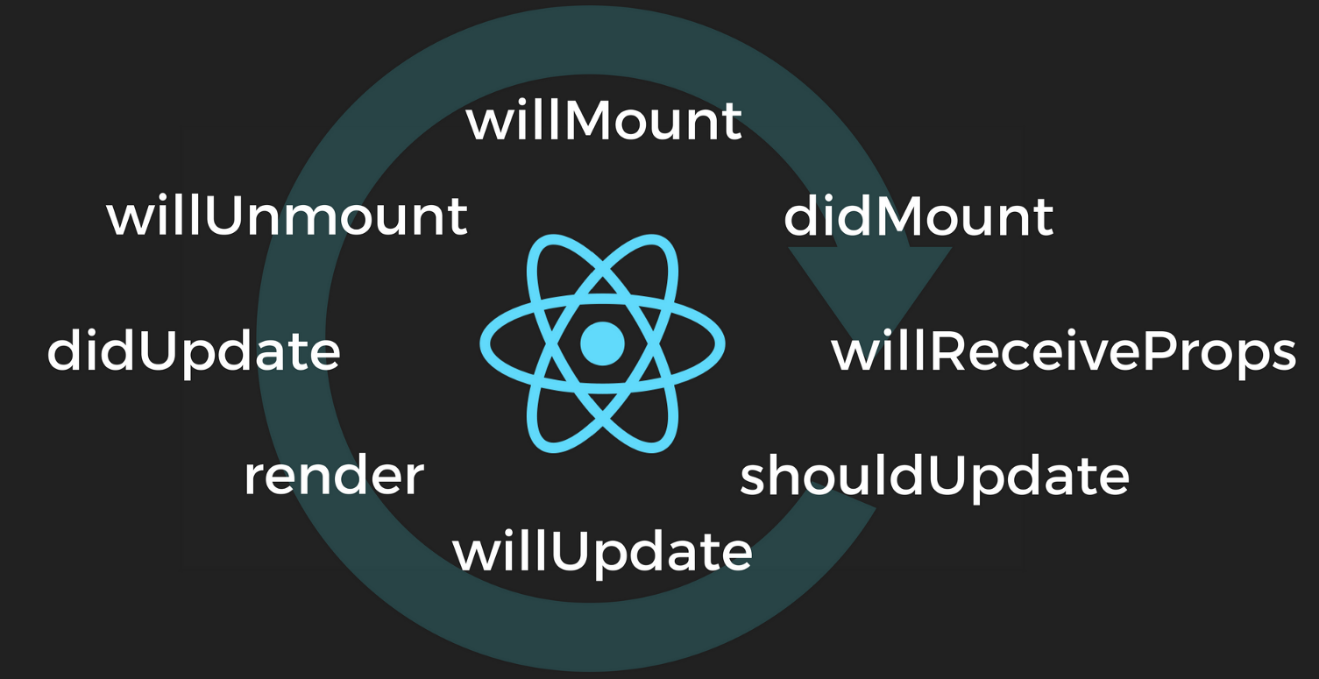
*Figure – Applying technologies into project*

\* Go to details, we have:

- React (React Native + React JS):

+ Use to build an UI for handling view layer of single page applications (React JS for Web and React Native for Mobile).

+ Create UI components for reuse.

Figure - React Life Cycle

- Redux:

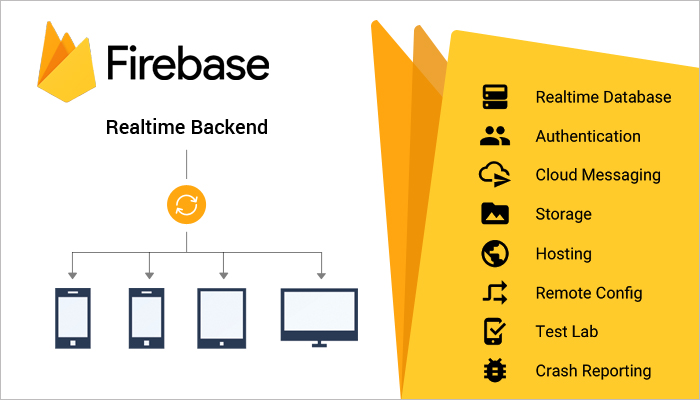
 + Apply Redux to control the state of application by providing "Unidirectional Data Flow" (UDF), then it helps to keep code in order and more flexible to maintain.

Figure - Redux Model

- Firebase:

+ To manage the backend database and trigger at the real time.

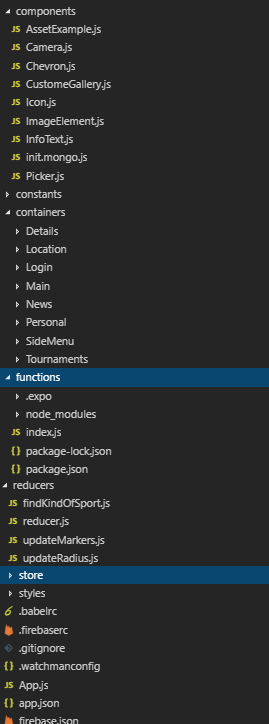
+ Integrate with various kinds of features to make application working smoothly.

Figure – Firebase features

**1. Layers, functions**

**1.1/ Mobile Application**

***1.1.1/ Overall structure***

- List of code structures:

+ component: Reuse UI components

+ constants: Storage of constants

+ containers: Contains all UI components for using in application’s features:

Details: Event detail feature

Location: Map feature

Login: Login feature

Main: Main page, contains all other UIs.

News: News feature

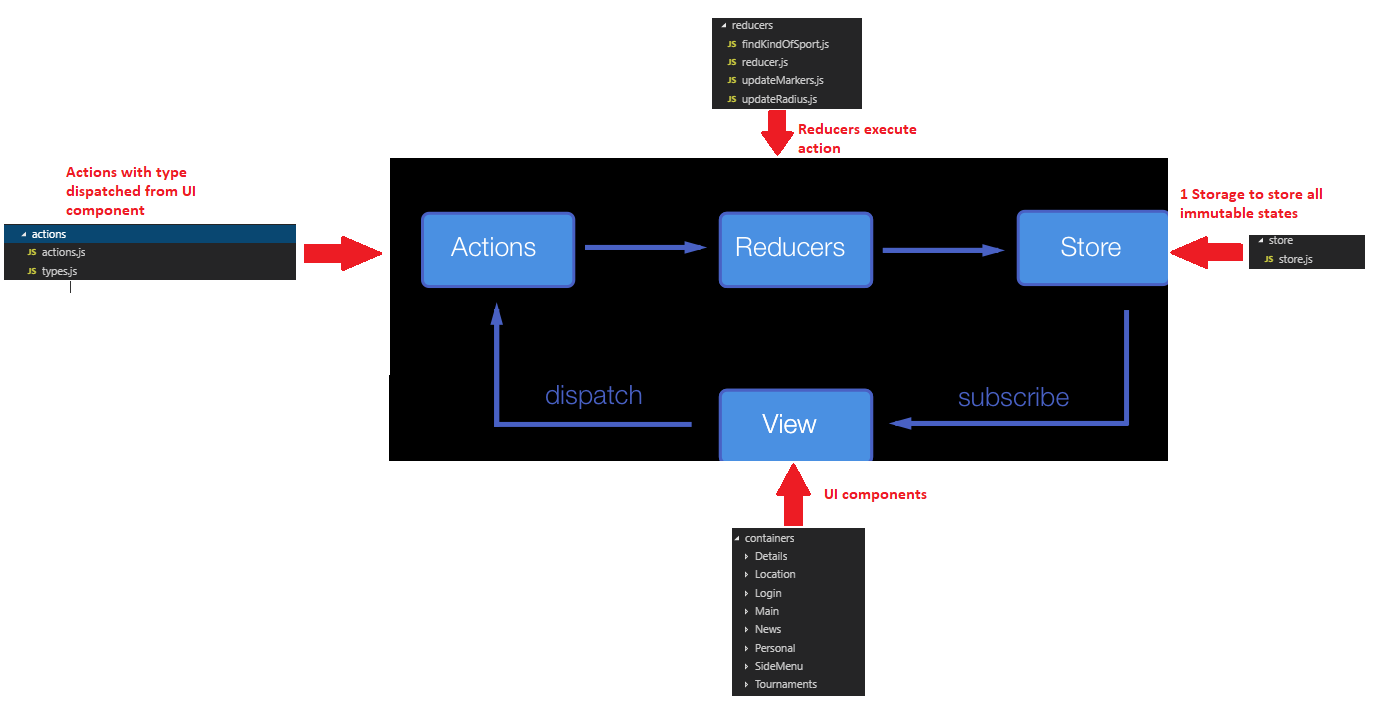
Personal: Personal Information feature

Tournaments: Event responding feature

+ functions: Container of Google Firebase - Cloud function

+ reducers, store: Applying of Redux into project

Figure - Code structure

- Mapping between packages and Redux:

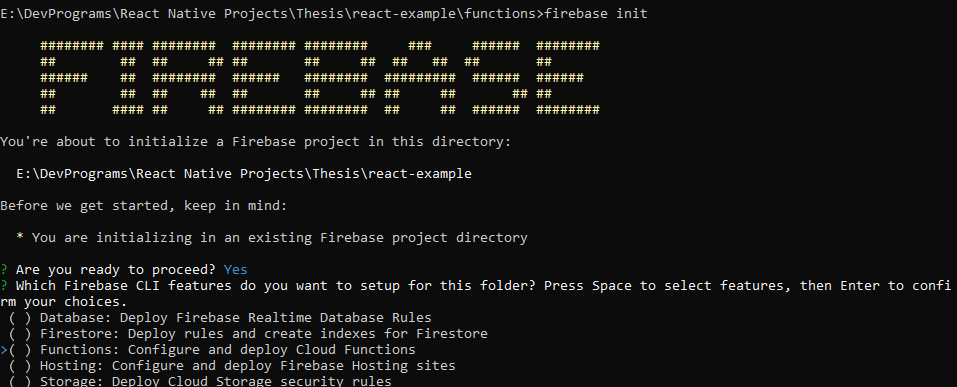
***1.1.2/ Push Notification***

- My application using Cloud Function integrate with Expo to send Push Notification at the real time to devices.

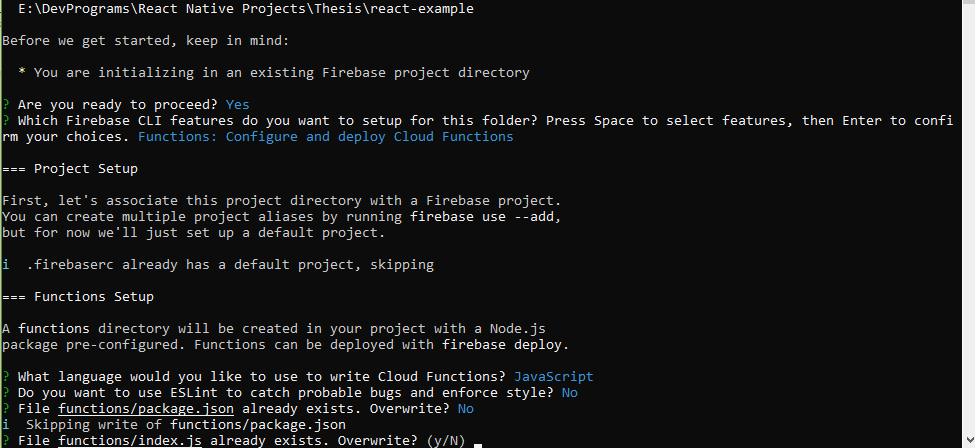
- To initialize Cloud Function in application, try following steps:

+ Initialize Firebase in product path:

firebase init

 + Type “Y” and select “Functions”

+ Select language to write function, then choose the other suitable options with you (in case of mine, I have already do the function before then I choose “N” all)



+ When select all options and execute, firebase will initialize successfully and show a file name Index.js inside created package “functions”

- Next, access to “functions” package and run command to install needed module:

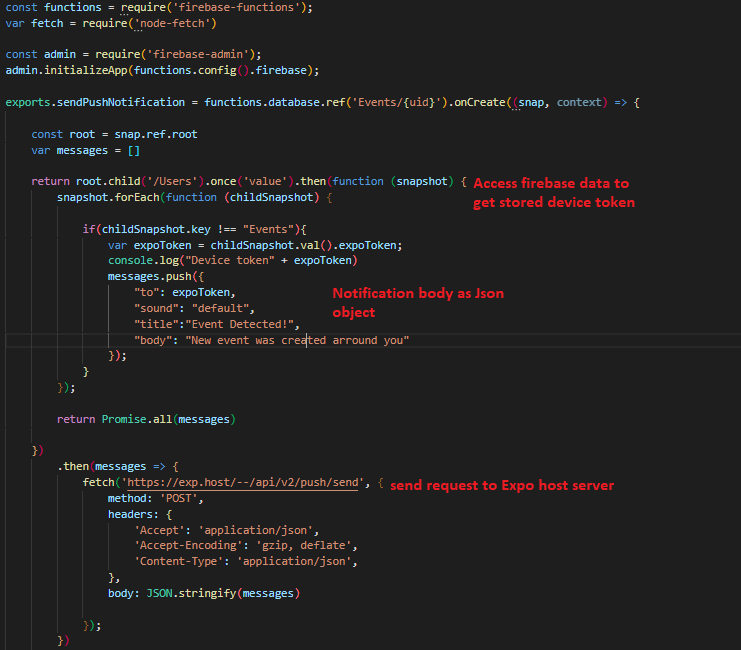
npm install node-fetch

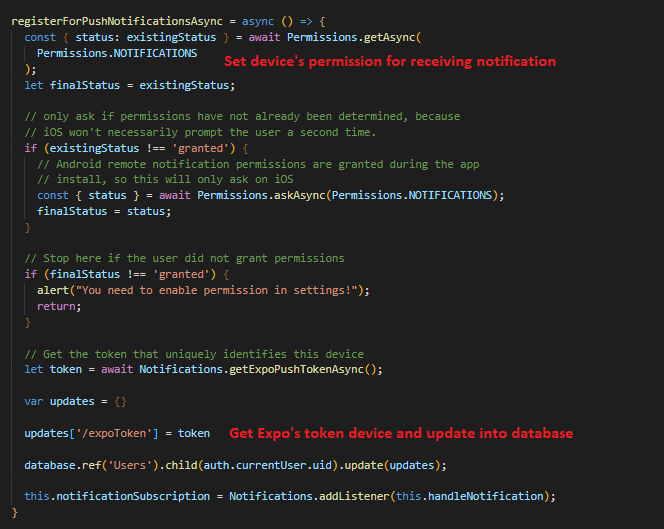
- Define your function in “Index.js” file and run command to deploy function on Firebase server:

firebase deploy

- Function deployed successfully will show with message “complete”.

- To receive notification, you must register your device’s token and there is how your device integrate with Firebase from UI component:

Figure - Write function to send Push Notification in “Index.js”

Figure - Register device’s token and update to database to receive notification in UI component

***1.1.3/ Map View***

- Map view in my application use “react-native-map” module integrated by Expo and Google Map API.

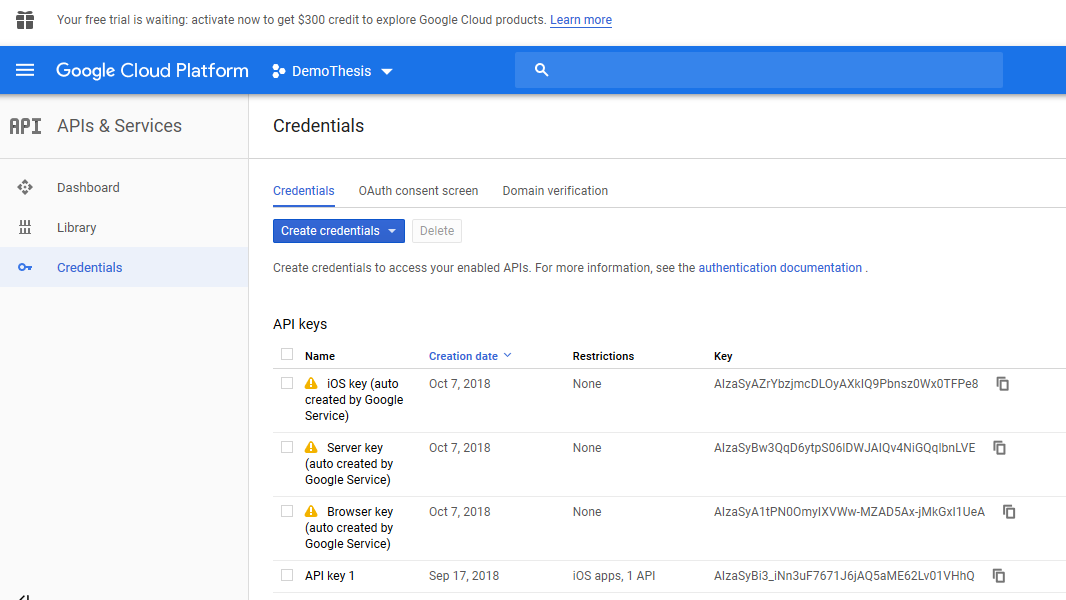
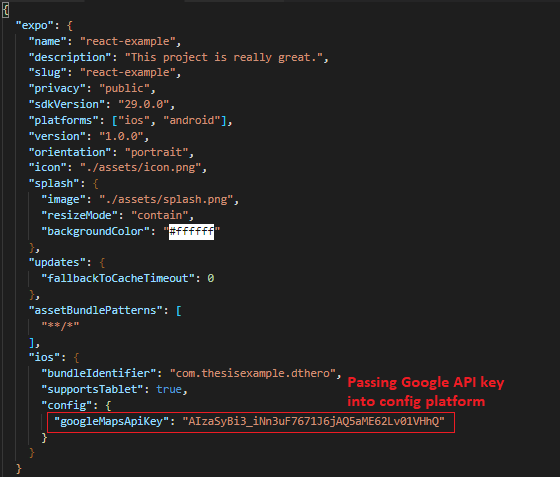
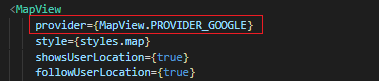
- First, get Google API by accessing <https://console.cloud.google.com> and register one, the API maybe provided for only mobile platform (iOS or Android) or for general purpose.

Figure – Google Cloud console of managing API keys

- To use Google Map in Map View, register it by passing API key into config file of project (in this case is Expo’s app.json file):



- Then pass prop of Map View component by Google as provider

***1.1.4/ Camera***

- Camera is a reuse component (using in both Event Detail and Personal Information features) so we must put it in “components” package

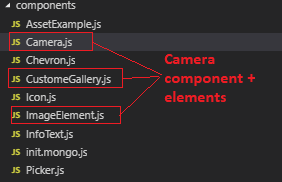


Figure - Camera component

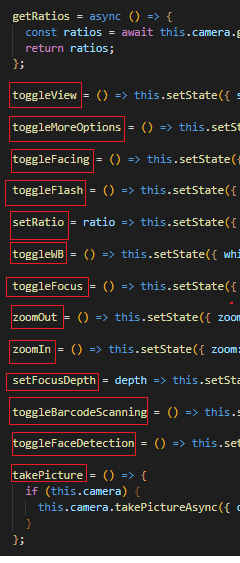
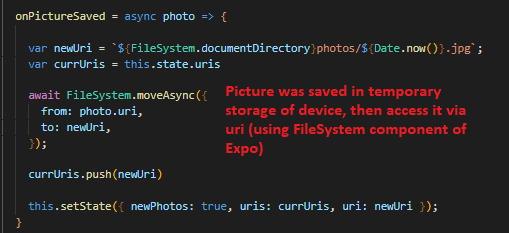
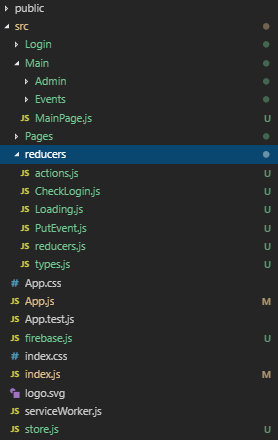


Figure - Actions inside Camera component



**1.2/ Web Application**

***1.2.1/ Overall Structure***



- List of features:

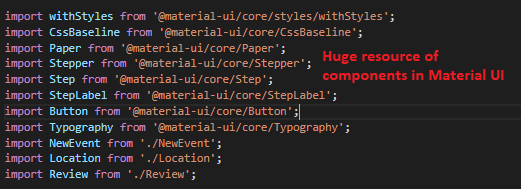
+ Login: Login feature

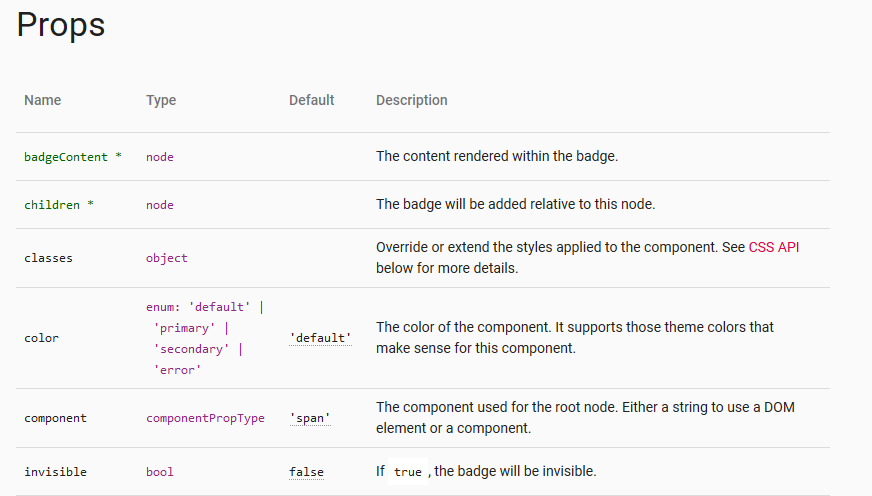
+ Main: Includes Main page and child component (Admin feature + Events feature)

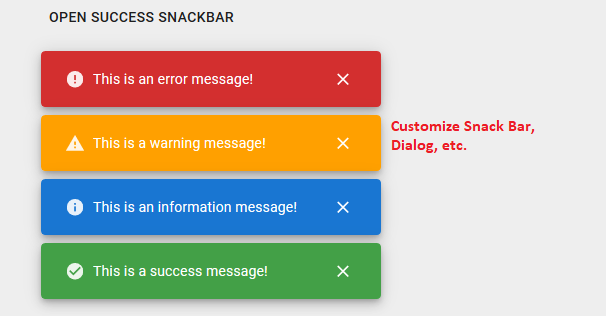
- reducers: Applying of Redux into project

Figure – Web application code structure

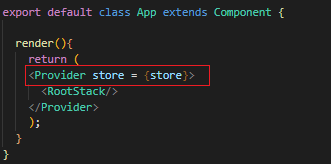
***1.2.2/ UI component - Material UI***

- Material UI is a great UI framework for developing React JS, with almost full of necessary components for a single page application.

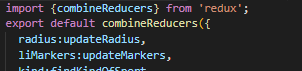
- It also provides many props relates and need for 1 component

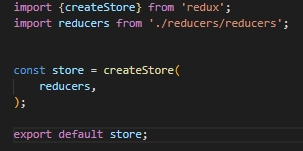
- Create a complex component to become more straightforward and customizable

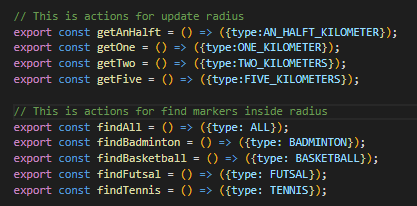
**1.3/ Redux implementation – react-redux**

- Define Redux at root component by providing Store class:

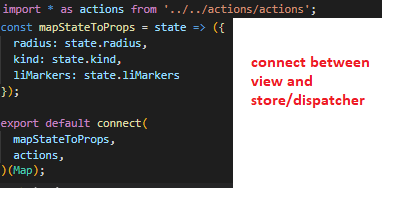
- Combine multi-reducers into only 1 and passing to Store:





- Initialize actions as callback functions for reducers:

- To trigger an action from component or update data from Store to View, we need to use connect() function in react-redux. Define all return states of Store as component’s props and import action, then pass it all to connect() function to initialize connection between component and states.



- After apply Redux into component, calling actions or getting latest state by using “this.props”

**1.4/ Firebase - Realtime database**

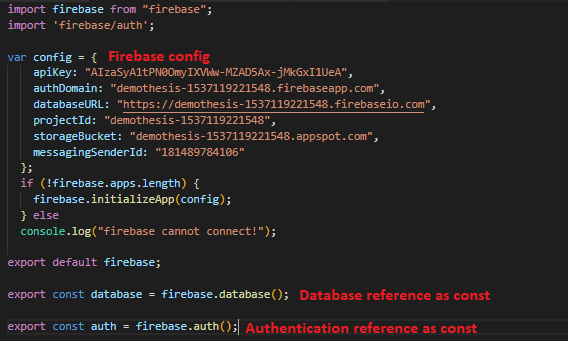
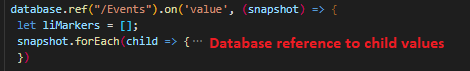
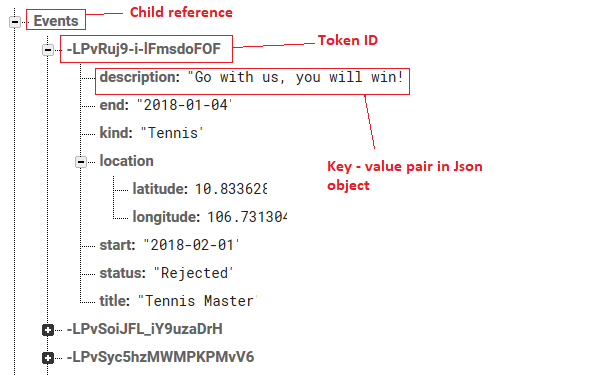
- To use Firebase database in the project, we must config our mapping project on Firebase console and call it when trigger data, since Firebase config is unchanged, so I saved “firebase.js” config file in “constants” package:

Figure - Firebase config and constant references

- To trigger data from server, using const above and access to desired child



- Firebase Realtime Database is save as Json object and include Token ID (the same as Object ID) in MongoDB. Each time we push new data into database, we can generate new Token ID data object:



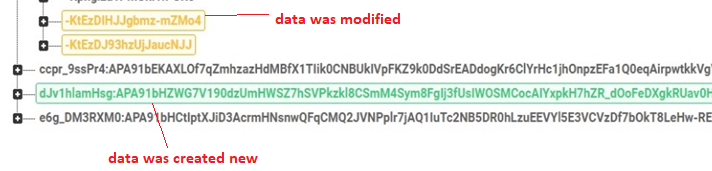
Figure - Firebase Realtime Database view

Figure - Synchronous database in real time

- Authentication in Firebase is used to verify that whether user has registered or not:

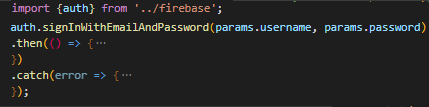


Figure – Using authentication in application

**2. Result**

**IV. CONCLUSION**

**1. Experiences**

\* I have gained many things after ran this thesis project:

- Firstly, I got a knowledge about React Native, a very similar to Javascript language. It is not only easy to learn and build a project by React but also use it to compile and run an application on multi platforms like a native application.

- Secondly, through the usage of Firebase, I made the connection to server side easier. In addition, using the combinations of functions in Firebase (Cloud Function, Real time database, Authentication) help to make the application becomes more interactive and works smoothly.

- Thirdly, I used NoSQL database for managing data and then it helps me to work with Json - a very popular data syntax - more and more efficiently.

- Finally, I have learnt many new technologies and got ability to work with new frameworks/environments. Using React Native helps me have experience to build a native application on mobile based on a single language, it will help me a lot in the future if becoming a mobile developer is the way for myself.

**2. Conclusion**

- React Native is gradually confirmed that it is one of the best language using for developing application on mobile. It is not for a specific platform, but it can also be embedded native language and build like a native application. The libraries of React Native is developed more and more, so it is easy to find one that satisfy product requirements.

- Beside that, using Flux architecture also help to manage product more efficiently. Applying Flux makes states is more clearly, so it makes application is easier to maintain and update.

- Firebase is also a good platform for developing mobile and web application. It contains many solutions help to manage and use data, especially notifications and messages system, etc. It also provides the storage to store/retrieve data as Json object at real time, which is very convenient for online applications on mobile.

**REFERENCES**

[1], [2] <https://docs.expo.io/versions/v31.0.0/workflow/logging>