



TECHNOLOGICAL EDUCATIONAL  
INSTITUTE OF CRETE

School of Applied Technology  
Department of Informatics Engineering

## **Thesis**

**Title: Application-Website Portal for parents of kids  
with epilepsy**

Marinakis Ioannis (4244)

Advisor: Papadourakis Georgios

## ACKNOWLEDGEMENTS

I would first like to thank my professor and thesis advisor, Dr. Georgios Papadourakis for giving me the opportunity to participate in Blended Academic International Mobility project and for his guidance during it. It was a great experience which provided me with many skills that will be useful in the future.

Also, I would like to thank the students and the professors of the Blended AIM 2018 for the great experience of working as a team on a big project for a company.

A special thanks to Tim Buckinx founder and CEO of Epihunter, who gave us the opportunity to work on a special project that, it's going to make a great impact on many people's lives.

Finally, I would like to thank the thesis committee members Gareth Owens and Athanasios Malamos for their valuable input on my thesis.

## DEDICATION

This thesis is dedicated to parents and family for their endless love, support and continuous encouragement throughout my years of study and through the development and writing of this thesis. This would not have been possible without them.

## ABSTRACT

The purpose of this thesis is the designing and development of an application for people that suffer from epilepsy. People with epilepsy are threatened by social isolation as seizures come unexpected. In this project, we envision a concept that allows people with epilepsy to go out with their friends and family safely and without stigmatization. In this Thesis we are going to examine the case of a kid suffering from epilepsy and our solution is helping both kids and parents to ensure that their kids can lead to a normal life.

The project consists of 2 hybrid mobile applications, one for the children and another for the parents. The child's application, using an EEG headset and a smartphone can detect seizures. Once a seizure is detected, the child's application pushes a notification and transmits live video of the child to the parent's smartphone. Records of the seizures and the medication are stored on a cloud database to help doctors with the treatment. Those data are visualized to the associated doctor via a website, in order to help with the diagnosis and the medication adjustment.

## ΣΥΝΟΨΗ

Σκοπός αυτής της εργασίας είναι ο σχεδιασμός και η ανάπτυξη μιας εφαρμογής για άτομα που πάσχουν από επιληψία. Τα άτομα με επιληψία βιώνουν κοινωνική απομόνωση καθώς οι επιληπτικές κρίσεις συμβαίνουν απροσδόκητα. Σε αυτό το project, οραματιζόμαστε μια εφαρμογή η οποία θα επιτρέπει στα άτομα με επιληψία να βγουν έξω με τους φίλους και την οικογένειά τους με ασφάλεια και χωρίς στιγματισμό. Στην παρούσα πτυχιακή θα εξετάσουμε την περίπτωση ενός παιδιού που πάσχει από επιληψία, με τη λύση μας να βοηθά τόσο το παιδί, όσο και τους γονείς. Διασφαλίζοντας πως το παιδί μπορεί να έχει μια φυσιολογική ζωή.

Το project αποτελείται από 2 υβριδικές εφαρμογές για κινητές συσκευές, μία για τα παιδιά και μία για τους γονείς. Η εφαρμογή των παιδιών χρησιμοποιώντας ένα EEG headset το οποίο καταγράφει τη δραστηριότητα του εγκεφάλου και ένα smartphone, μπορεί να εντοπίσει επιληπτικές κρίσεις. Μόλις εντοπιστεί μία επιληπτική κρίση, η εφαρμογή του παιδιού στέλνει μια ειδοποίηση αλλά και ζωντανό βίντεο στην εφαρμογή των γονέων. Τα δεδομένα των επιληπτικών κρίσεων και των φαρμακευτικών αγωγών, αποθηκεύονται σε μία βάση δεδομένων cloud με σκοπό να βοηθήσουν τους ειδικούς. Επίσης, μια ιστοσελίδα δίνει τη δυνατότητα σε γονείς και ιατρούς, να οπτικοποιήσουν τα δεδομένα των κρίσεων, προκειμένου να βοηθήσουν τους ειδικούς με τη διάγνωση αλλά και τη φαρμακευτική αγωγή.



# Table of Contents

<b>ACKNOWLEDGEMENTS</b> .....	- 2 -
<b>DEDICATION</b> .....	- 3 -
<b>ABSTRACT</b> .....	- 4 -
<b>ΣΥΝΟΨΗ</b> .....	- 5 -
<b>Table of figures</b> .....	- 9 -
<b>Table of pictures</b> .....	- 10 -
<b>CHAPTER 1: INTRODUCTION</b> .....	- 11 -
1.1 Summary.....	- 11 -
1.2 Motive.....	- 11 -
1.3 Blended AIM (Academic International Mobility).....	- 12 -
1.3.1 The history of BlendEd mobility.....	- 12 -
<b>CHAPTER 2: FUNDAMENTALS</b> .....	- 13 -
2.1 Analysis and Development Methods.....	- 13 -
2.1.1 Nexus Scrum.....	- 13 -
2.1.2 Trello.....	- 13 -
2.1.3 Bitbucket.....	- 14 -
2.1.4 Slack.....	- 14 -
<b>CHAPTER 3: WORK PLAN</b> .....	- 16 -
3.1 State of the Art.....	- 16 -
3.2 Technologies.....	- 17 -
3.2.1 Node.js.....	- 17 -
3.2.2 Angular JS.....	- 18 -
3.2.3 Android.....	- 18 -
3.2.4 iOS.....	- 19 -
3.2.5 Ionic Framework.....	- 19 -
3.2.6 npm.....	- 20 -
3.2.7 Apache Cordova.....	- 20 -
3.2.8 Firebase.....	- 21 -
3.2.9 Spring Boot RESTful Web Service (API).....	- 22 -

3.2.10	<i>Apache Maven</i> .....	- 23 -
3.2.11	<i>Google Maps API</i> .....	- 24 -
3.2.12	<i>Google Cloud functions</i> .....	- 24 -
3.2.13	<i>Material design</i> .....	- 25 -
3.2.14	<i>Design Patterns</i> .....	- 25 -
3.2.15	<i>Java</i> .....	- 25 -
<b>CHAPTER 4: MAIN PART</b> .....		- 27 -
4.1	<i>Problem Analysis</i> .....	- 27 -
4.1.1	<i>Problem Description</i> .....	- 27 -
4.1.2	<i>System Requirements</i> .....	- 28 -
4.2	<i>Implementation</i> .....	- 30 -
4.2.1	<i>Project Implementation</i> .....	- 30 -
4.2.2	<i>Detailed Project Implementation</i> .....	- 31 -
4.2.2.1	<i>User Interface Design</i> .....	- 38 -
4.2.2.2	<i>Firebase</i> .....	- 46 -
4.2.2.3	<i>Google Cloud Functions</i> .....	- 46 -
4.2.2.4	<i>RestAPI (Spring boot)</i> .....	- 48 -
4.2.2.5	<i>ApiRTC</i> .....	- 51 -
4.2.2.6	<i>Google Maps API</i> .....	- 54 -
4.2.2.7	<i>Activities</i> .....	- 55 -
4.2.2.8	<i>Fragments</i> .....	- 58 -
4.2.2.9	<i>Modals</i> .....	- 60 -
4.2.2.10	<i>App Diagrams</i> .....	- 61 -
4.3	<i>Manual</i> .....	- 63 -
<b>CHAPTER 5: RESULTS</b> .....		- 78 -
5.1	<i>Conclusion</i> .....	- 78 -
5.2	<i>Future work and extensions</i> .....	- 79 -



## Table of figures

Sprints Table 1 .....	- 28 -
Sprints Table 2 .....	- 28 -
Sprints Table 3 .....	- 29 -
Sprints Table 4 .....	- 29 -
Sprints Table 5 .....	- 29 -
Sprints Table 6 .....	- 29 -
Sprints Table 7 .....	- 29 -
Sprints Table 8 .....	- 29 -
Doctor Sprint Table 1 .....	- 30 -
Parent Story 1 .....	- 31 -
Parent Story 2 .....	- 32 -
Parent Story 3 .....	- 32 -
Parent Story 4 .....	- 32 -
Parent Story 5 .....	- 33 -
Parent Story 6 .....	- 34 -
Parent Story 7 .....	- 34 -
Child Story 1 .....	- 35 -
Child Story 2 .....	- 35 -
Child Story 3 .....	- 36 -
Child Story 4 .....	- 36 -
Child Story 5 .....	- 36 -
API Story 1 .....	- 37 -
API Story 2 .....	- 37 -
API Story 3 .....	- 37 -

## Table of pictures

Picture 1, Logged Out Side-Menu.....	- 41 -
Picture 2, Logged In Side-Menu .....	- 41 -
Picture 3, Bottom navigation bar.....	- 42 -
Picture 4, Top navigation bar 1 .....	- 42 -
Picture 5, Top navigation bar 2 .....	- 43 -
Picture 6, Seizure Category Color Codes .....	- 44 -
Picture 7, Notification pop-over .....	- 45 -
Picture 8, Cloud Messaging.....	- 47 -
Picture 9, Cloud Function Execution Time .....	- 48 -
Picture 10, Project General Diagram.....	- 61 -
Picture 11, Extended Use Case Diagram.....	- 62 -
Picture 12, Manual, Tutorial 1.....	- 63 -
Picture 13, Manual, Tutorial 2.....	- 63 -
Picture 14, Manual, Tutorial 3.....	- 64 -
Picture 15, Manual, Terms of Service .....	- 64 -
Picture 16, Manual, Login.....	- 65 -
Picture 17, Manual, Register .....	- 65 -
Picture 18, Manual, Logged In Menu.....	- 66 -
Picture 19, Manual, Logged Out Menu .....	- 66 -
Picture 21, Manual, Top nav-bar .....	- 66 -
Picture 20, Manual, Bottom nav-bar .....	- 66 -
Picture 22, Manual, Calendar Date-picker .....	- 67 -
Picture 23, Manual, Calendar Daily Timeline.....	- 68 -
Picture 24, Manual, Seizure Record Details.....	- 68 -
Picture 25, Manual, Insert Record Manually.....	- 69 -
Picture 26, Manual, Record Inserted.....	- 69 -
Picture 27, Manual, Medication Reminders .....	- 70 -
Picture 28, Manual, Medication Notification .....	- 70 -
Picture 29, Manual, Recorded streams .....	- 71 -
Picture 30, Manual, Video Player-Details.....	- 71 -
Picture 31, Manual, Seizure Notifications.....	- 72 -
Picture 32, Manual, Live video-stream .....	- 73 -
Picture 33, Manual, Map Location .....	- 73 -
Picture 34, Manual, Application Settings.....	- 74 -
Picture 35, Manual, Account Settings .....	- 74 -
Picture 36, Manual, Doctor Patients List.....	- 75 -
Picture 37, Manual, Doctor Patient Details .....	- 76 -
Picture 38, Manual, Doctor EEG and Video .....	- 77 -

# CHAPTER 1: INTRODUCTION

## *1.1 Summary*

This thesis is part of an international Erasmus+ project named Blended-AIM (Academic International Mobility), where students from Educational Institutes and Universities around Europe formed two teams to develop two different applications requested by startup companies.

The goal of my team was to design and develop an application for the parents of children that suffer from epilepsy. This application is going to notify the parents once a seizure event is detected. Also, it's going to help the doctors to make the right adjustments on medication, by keeping track of the seizure records. To achieve that our team was split further into a development team, a design team and a business team.

For the back-end development of the app, technologies like Node.js, npm, Angular.JS, Cordova Plugins and Android Studio were used. As for the front-end we used Ionic Framework, HTML and CSS. The data from the app were stored to Firebase, Google's cloud database and the connection between the app and the database is done by developing a Spring Boot RESTful Web Service (API) using Java.

The project was hosted in Bitbucket from where all the developers had access to the source code.

## *1.2 Motive*

More than 65 million people around the world have epilepsy and 1 out of 7 people with epilepsy suffer from absence seizures. "Daddy, you work in digital, can you make a light that turns on when my brain switches off?" This question from a son to his father led to this new start-up. Epihunter Classroom using an EEG headset and a mobile device can detect an absence seizure and turn on the flashlight of the child's phone. This is very helpful for teachers and parents because absence seizures are really difficult to notice. With Epihunter Classroom teachers and parents are now aware of the absence seizures. But is there any room for improvement for the Epihunter Classroom? Can parents be notified in another way except the flashlight?

That's where Epigo, our mobile app for the parents, comes in to notify them when an absence seizure occurs. Parents are now aware of the absence seizures. With this app they can check their child's location from their phone and go to help if that's needed. Also, live video of their child is streamed to their phone during a seizure. Epigo, sends also reminders to the parents about their child's medications. All the seizure events and medications are visualized on a calendar view. Finally, the recorded data and the video are accessible from the website in order to help the associated doctors to improve the child's treatment.

### *1.3 Blended AIM (Academic International Mobility)*

Blended AIM (Academic International Mobility) is an Erasmus+ funded project made to promote students' employability and support companies hosting internships. Every year 10 educational institutes from European countries like Portugal, Greece, Belgium, United Kingdom, Germany, Iraq and Austria end up to 2 students each to form a team. The purpose of that is to support the students develop soft skills in an international environment by means of blended mobility.

At this moment, a student's professional career depends on mobility and demands certain intercultural skills. However, most institutes don't have courses that provide international exposure. Blended mobility helps the students adapt and learn but it's hardly considered, let alone used, a solution to international mobility's problems. Blended Aim sets the foundation to promote and test blended mobility by providing the resources, training, supporting tools and information to the students and the companies that host internships. For that reason Praxis was created. Praxis is a consortium of higher education institutions, companies, associations, research labs and chambers of commerce, all committed to enhance a student's or a company's Project/Internship experience and to promote innovation in the field.

#### *1.3.1 The history of BlendEd mobility*

BlendEd Mobility is the sequel of former collaborations between several European universities. One of the most valuable predecessors is the MUTW project (Multinational Undergraduate Team Work - funded by ERASMUS KA2 - 2009). This Erasmus project's main goal was to strengthen the students' communication skills by organizing them in an international, multicultural team. In the very first phase we simply had an international team of IT students working together on a common project, collaborating mainly online. As a second step, we extended the student team with members from other study areas. Next to IT-students, designers got involved, as well as students from Business Development and Management. The technical outcome was enriched with design and the resulting product was accompanied by a real business plan. Students learned to communicate throughout the borders of their own discipline. In between the former MUTW project (2009-2011) and the new BlendEd Mobility project (2016-2018) we ran several editions on our own. During this period we extended our goals by involving companies as provider for real project proposals. With this professional involvement the students get a context which is international, multicultural, multidisciplinary and professional!

Today some of the partner universities integrated BlendEd mobility as a course unit in their curricula.

## CHAPTER 2: FUNDAMENTALS

### *2.1 Analysis and Development Methods*

#### *2.1.1 Nexus Scrum*

Nexus is a framework, based on the principles of Scrum and the Agile Manifesto that helps the developers by minimizing cross-team dependencies and integration issues. It was developed by Ken Schwaber, the co-creator of Scrum and founder of Scrum.org, in 2015. Nexus framework is an extension to the Scrum framework and it uses an iterative and incremental approach to scaling software and product development. It allows multiple Scrum teams that work on a product, to unify as a single larger team.

Nexus framework consists of 3-9 scrum teams that work on the same product backlog for the successful development of a product. The main goal is the identification of cross-team issues and making sure that integration tools are understood and used. Integration team is a new feature in the Nexus framework and it is the team that is accountable for the successful integration of all work created by all the Scrum Teams in a Nexus. It consists of a Product Owner, a Scrum master and few team members of each scrum team in a nexus.

Each scrum team needs to be represented by a team member. Then the representatives meet to identify, resolve in-Sprint dependencies and set the goals that they want to achieve in each Sprint. These steps explain the “Nexus Sprint Planning” term which is crucial in a Nexus framework.

Furthermore, Nexus Daily Scrums help the teams to plan correctly their next steps by inspecting the integrated work. Nexus Daily Scrum is a brief meeting between the Scrum teams’ members. Not everyone has to attend but each team’s representative must.

#### *2.1.2 Trello*

Trello is a web-based project management application originally made by Fog Creek Software in 2011 that was spun out to form the basis of a separate company in 2014 and later sold to Atlassian in January 2017.

According to a Fog Creek blog post in January 2012, the client was an extremely thin web layer which downloads the main app, written in CoffeeScript and compiled to minified JavaScript, using Backbone.js HTML5 .pushState() and the Mustache templating language. The server side was built on top of MongoDB, Node.js and a modified version of Socket.io

### *2.1.3 Bitbucket*

Bitbucket is a web-based version control repository hosting service owned by Atlassian, for source code and development projects that use either Mercurial (since launch) or Git (since October 2011) revision control systems. Bitbucket offers both commercial plans and free accounts. It offers free accounts with an unlimited number of private repositories (which can have up to five users in the case of free accounts) as of September 2010. Bitbucket integrates with other Atlassian software like Jira, HipChat, Confluence and Bamboo.

It is similar to GitHub, which primarily uses Git. Bitbucket has traditionally marketed its services to professional developers with private proprietary software code, especially since being acquired by Atlassian in 2010. In September 2016, Bitbucket announced it had reached 5 million developers and 900,000 teams on its platform. Bitbucket has 3 deployment models: Cloud, Bitbucket Server and Data Center.

### *2.1.4 Slack*

Slack is a cloud-based set of proprietary team collaboration tools and services, founded by Stewart Butterfield. Slack began as an internal tool used by his company, Tiny Speck, in the development of Glitch, a now defunct online game. The name is an acronym for "Searchable Log of All Conversation and Knowledge".

#### **Features**

While no longer using an IRC backend, Slack offers a lot of IRC-like features, including persistent chat rooms (channels) organized by topic, private groups and direct messaging. All content inside Slack is searchable, including files, conversations, and people. On the free plan, only the 10,000 most recent messages can be viewed and searched. Users can add emoji buttons to their messages, which other users can then click on to express their reactions to messages.

#### **Teams**

Slack teams allow communities, groups, or teams to join through a specific URL or invitation sent by a team admin or owner. Although Slack was meant for organizational communication, it has been slowly turning into a community platform, a function for which users had previously used message boards or social media such as Facebook or LinkedIn groups. Many of these communities are categorized by topics which a group of people may be interested in discussing.

#### **Messaging**

Public channels allow team members to communicate without the use of email or group SMS (texting). They are open to everyone in the chat provided they have first been invited to join the client. Private channels allow for private conversation between smaller sects of the overall group. These can be used to break up large teams into their own respective projects. Direct messages allow users to send private messages to a specific user rather than a group of people.[24] Direct

messages can include up to nine people (the originator plus eight people). Once started this direct message group can be converted to a private channel.

### **Integrations**

Slack integrates with a large number of third-party services and supports community-built integrations. Major integrations include services such as Google Drive, Trello, Dropbox, Box, Heroku, IBM Bluemix, Crashlytics, GitHub, Runscope, Zendesk and Zapier. In December 2015, Slack announced their app directory, consisting of over 150 integrations that users can install. In March 2018, Slack announced its partnership with the financial and human capital management firm Workday; Adding to Slack's list of plugins. This integration allows Workday customers to access features from directly within the Slack interface.

### **Platforms**

Slack provides mobile apps for iOS, Android, and Windows Phone (beta), in addition to their web browser client and electron desktop clients for macOS, Windows, and Linux (beta). Slack is also available for the Apple Watch, allowing users to send direct messages, see mentions, and make simple replies. It was featured on the home screen of the Apple Watch in a 2015 promotional video.

### **Business model**

Slack is a freemium product, whose main paid feature is the ability to search more than 10,000 archived messages. They claim support for an unlimited number of users. However, when freeCodeCamp attempted to switch its community of over 8,000 users to Slack in 2015, they experienced many technical issues and were advised by Slack support to limit their channels to "no more than 1,000 users (ideally more like 500)".

## CHAPTER 3: WORK PLAN

### *3.1 State of the Art*

The main goal of this application is to notify the parents when an absence seizure happens. The EEG headset records continuously the child's brain activity in real-time and by processing those data simultaneously on the child's smartphone, an absence seizure can be detected on the moment that happens. The absence seizures can be detected, but how can the parents be aware of them? Light alerts and sound alerts on the child's smartphone is way to notify the parents, but what happens when the parents are away from their child? That's why there is the need of an application on the parent's smartphones. By using this application, parents can receive notifications from their child's smartphone when a seizure happens.

With Firebase and Google Cloud functions, a notification reaches on parents smartphones almost in real-time. If the parent opens this notification, live-video of the child is streamed to the parent smartphone by using Api-RTC, a real-time web communication library. Also, through this notification, the child's exact location is pushed on the parent's smartphone. This location is displayed on the application using the Google Maps API. Now parents are aware of their child's absence seizures and they can go to help their child if that's needed. This application can also help doctors with the treatment. The most common question that is asked by the doctors to the parents is when those absence seizures happened and how much time they lasted. By storing the absence seizures data, doctors now have an overview of all the absence seizures and make adjustments on the medication.



## *3.2 Technologies*

During the implementation of this project we used a variety of technologies. For the back-end development of the app, technologies like Node.js, npm, Angular.JS, Apache Cordova and Android Studio were used. As for the front-end we used Ionic Framework. Ionic and Angular.JS are really powerful tools to develop cross platform mobile applications cause from the same source code you can built applications or Android, iOS and Windows phone. Also, several Apache Cordova plug-in's gave more functionalities to the application. The database we used is Firebase, Google's latest cloud database and the connection between the app and the database is done by developing a Spring Boot RESTful Web Service (API) using Java and Maven. Also, Google's cloud functions were used and Google Maps API. Several Design Patterns were used, as for the styling of the app we used mostly Material Design. Bellow, information on the above-mentioned technologies are given.

### *3.2.1 Node.js*

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside the browser. Historically, JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript to write Command Line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.

Though .js is the conventional filename extension for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

### *3.2.2 Angular JS*

AngularJS (commonly referred to as "Angular.js" or "AngularJS") is a JavaScript-based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. The JavaScript components complement Apache Cordova, a framework used for developing cross-platform mobile apps. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–viewmodel (MVVM) architectures, along with components commonly used in rich Internet applications. In 2014, the original AngularJS team began working on the Angular application platform.

The AngularJS framework works by first reading the HTML page, which has additional custom tag attributes embedded into it. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.

According to JavaScript analytics service Libscore, AngularJS is used on the websites of Wolfram Alpha, NBC, Walgreens, Intel, Sprint, ABC News, and about 12,000 other sites out of 1 million tested in October 2016. AngularJS is currently in the top 100 of the most starred projects on GitHub

### *3.2.3 Android*

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

Its open source is what attracted many developers but more importantly technology companies that require a low-cost and customizable operating system for their high-tech devices. By having such a huge development community, the variety of applications, tutorials on development, development tools and supported programming languages is increasing.

Applications are developed using the Android Software Development Kit (SDK) and usually the Java programming language through the Android Studio or Eclipse Integrated Development Environments (IDEs).

### *3.2.4 iOS*

iOS (formerly iPhone OS) is a mobile operating system created and developed by Apple Inc. exclusively for its hardware. It is the operating system that presently powers many of the company's mobile devices, including the iPhone, iPad, and iPod Touch. It is the second most popular mobile operating system globally after Android.

Originally unveiled in 2007 for the iPhone, iOS has been extended to support other Apple devices such as the iPod Touch (September 2007) and the iPad (January 2010). As of January 2017, Apple's App Store contains more than 2.2 million iOS applications, 1 million of which are native for iPads. These mobile apps have collectively been downloaded more than 130 billion times.

The iOS user interface is based upon direct manipulation, using multi-touch gestures. Interface control elements consist of sliders, switches, and buttons. Interaction with the OS includes gestures such as swipe, tap, pinch, and reverse pinch, all of which have specific definitions within the context of the iOS operating system and its multi-touch interface. Internal accelerometers are used by some applications to respond to shaking the device (one common result is the undo command) or rotating it in three dimensions (one common result is switching between portrait and landscape mode). Apple has been significantly praised for incorporating thorough accessibility functions into iOS, enabling users with vision and hearing disabilities to properly use its products.

### *3.2.5 Ionic Framework*

Ionic is a complete open-source SDK for hybrid mobile app development. The original version was released in 2013 and built on top of AngularJS and Apache Cordova. The more recent releases, known as Ionic 3 or simply "Ionic", are built on Angular. Ionic provides tools and services for developing hybrid mobile apps using Web technologies like CSS, HTML5, and Sass. Apps can be built with these Web technologies and then distributed through native app stores to be installed on devices by leveraging Cordova. Ionic was created by Max Lynch, Ben Sperry, and Adam Bradley of Drifty Co. in 2013. Ionic uses Cordova plugins to gain access to host operating systems features such as Camera, GPS, Flashlight, etc. Users can build their apps, and they can then be customized for Android, iOS, Windows, or modern browsers. Ionic allows you to build and deploy your apps by wrapping around the build tool Cordova with a simplified 'ionic' command line tool. Ionic includes mobile components, typography, interactive paradigms, and an extensible base theme. Using Angular, Ionic provides custom components and methods for interacting with them. One such component, collection repeat, allows users to scroll through a list of thousands of items without any performance hits. Another component, scroll-view, creates a scrollable container with which users can interact using a native-influenced delegate system.

### 3.2.6 *npm*

npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. The package manager and the registry are managed by npm, Inc.

npm is included as a recommended feature in Node.js installer. npm consists of a command line client that interacts with a remote registry. It allows users to consume and distribute JavaScript modules that are available on the registry. Packages on the registry are in CommonJS format and include a metadata file in JSON format. Over 477,000 packages are available on the main npm registry. The registry has no vetting process for submission, which means that packages found there can be low quality, insecure, or malicious. Instead, npm relies on user reports to take down packages if they violate policies by being low quality, insecure or malicious. npm exposes statistics including number of downloads and number of depending packages to assist developers in judging the quality of packages.

### 3.2.7 *Apache Cordova*

Apache Cordova (formerly PhoneGap) is a mobile application development framework originally created by Nitobi. Adobe Systems purchased Nitobi in 2011, rebranded it as PhoneGap, and later released an open source version of the software called Apache Cordova. Apache Cordova enables software programmers to build applications for mobile devices using CSS3, HTML5, and JavaScript instead of relying on platform-specific APIs like those in Android, iOS, or Windows Phone. It enables wrapping up of CSS, HTML, and JavaScript code depending upon the platform of the device. It extends the features of HTML and JavaScript to work with the device. The resulting applications are hybrid, meaning that they are neither truly native mobile application (because all layout rendering is done via Web views instead of the platform's native UI framework) nor purely Web-based (because they are not just Web apps, but are packaged as apps for distribution and have access to native device APIs). Mixing native and hybrid code snippets has been possible since version 1.9. The software was previously called just "PhoneGap", then "Apache Callback". As open-source software, Apache Cordova allows wrappers around it, such as Appery.io or Intel XDK. PhoneGap is Adobe's commercial version of Cordova along with its associated ecosystem. Many other tools and frameworks are also built on top of Cordova, including Ionic, Monaca, TACO, Onsen UI, Visual Studio, GapDebug, App Builder, Cocoon, Framework7, Quasar Framework, Evothings Studio, NSB/AppStudio, Mobiscroll, the Intel XDK, and the Telerik Platform. These tools use Cordova, and not PhoneGap for their core tools.

### *3.2.8 Firebase*

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014.

#### Firestore

Formerly known as Google Cloud Messaging (GCM), Firestore is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which currently can be used at no cost.

#### Auth

Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firestore.

#### Cloud Messaging

Cloud Messaging is a service that enables targeted user notifications for mobile app developers at no cost.

#### Real-time Database

Real-time Database is a flexible, scalable database for mobile, web, and server development from Firestore and Google Cloud Platform. Like Firestore, it keeps your data in sync across client apps through real-time listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Real-time Database also offers seamless integration with other Firestore and Google Cloud Platform products, including Cloud Functions.

### 3.2.9 Spring Boot RESTful Web Service (API)

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an addition to, or even replacement for the Enterprise JavaBeans (EJB) model. The Spring Framework is open source.

#### Remote access framework

Spring's Remote Access framework is an abstraction for working with various RPC (remote procedure call)-based technologies available on the Java platform both for client connectivity and marshalling objects on servers. The most important feature offered by this framework is to ease configuration and usage of these technologies as much as possible by combining inversion of control and AOP. The framework also provides fault-recovery (automatic reconnection after connection failure) and some optimizations for client-side use of EJB remote stateless session beans. Spring provides support for these protocols and products out of the box

- HTTP-based protocols
  - Hessian: binary serialization protocol, open-sourced and maintained by CORBA-based protocols
  - RMI (1): method invocations using RMI infrastructure yet specific to Spring
  - RMI (2): method invocations using RMI interfaces complying with regular RMI usage
  - RMI-IIOP (CORBA): method invocations using RMI-IIOP/CORBA
- Enterprise JavaBean client integration
  - Local EJB stateless session bean connectivity: connecting to local stateless session beans
  - Remote EJB stateless session bean connectivity: connecting to remote stateless session beans
- SOAP
  - Integration with the Apache Axis Web services framework

Apache CXF provides integration with the Spring Framework for RPC-style exporting of objects on the server side. Both client and server setup for all RPC-style protocols and products supported by the Spring Remote access framework (except for the Apache Axis support) is configured in the Spring Core container. There is alternative open-source implementation (Cluster4Spring) of a remoting subsystem included into Spring Framework that is intended to support various schemes of remoting (1-1, 1-many, dynamic services discovering)

#### Spring Boot

Spring Boot is Spring's convention-over-configuration solution for creating stand-alone, production-grade Spring-based Applications that you can "just run". It is preconfigured with the Spring's "opinionated view" of the best configuration and use of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration. Features:

- Create stand-alone Spring applications
- Embed Tomcat or Jetty directly (no need to deploy WAR files)
- Provide opinionated 'starter' Project Object Models (POMs) to simplify your Maven configuration
- Automatically configure Spring whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

### 3.2.10 *Apache Maven*

Maven is a build automation tool used primarily for Java projects. Maven addresses two aspects of building software: first, it describes how software is built,[clarification needed] and second, it describes its dependencies. Unlike earlier tools like Apache Ant, it uses conventions for the build procedure, and only exceptions need to be written down. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging.

Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository, and stores them in a local cache. This local cache of downloaded artifacts can also be updated with artifacts created by local projects. Public repositories can also be updated.

Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

Maven is built using a plugin-based architecture that allows it to make use of any application controllable through standard input. Theoretically, this would allow anyone to write plugins to interface with build tools (compilers, unit test tools, etc.) for any other language. In reality, support and use for languages other than Java has been minimal. Currently a plugin for the .NET framework exists and is maintained, and a C/C++ native plugin is maintained for Maven 2.

Alternative technologies like Gradle and sbt as build tools do not rely on XML, but keep the key concepts Maven introduced. With Apache Ivy, a dedicated dependency manager was developed as well that also supports Maven repositories.

### 3.2.11 *Google Maps API*

Google launched the Google Maps API in June 2005 in order to allow developers to integrate Google Maps into their websites. It was a free service that didn't require an API key until June 2018 (changes went into effect on July 16th), when it was announced that an API key linked to a Google Cloud account with billing enabled would be required to access the API. The API currently does not contain ads, but Google states in their terms of use that they reserve the right to display ads in the future

By using the Google Maps API, it is possible to embed Google Maps into an external website, on to which site-specific data can be overlaid. Although initially only a JavaScript API, the Maps API was expanded to include an API for Adobe Flash applications (but this has been deprecated), a service for retrieving static map images, and web services for performing geocoding, generating driving directions, and obtaining elevation profiles. Over 1,000,000 web sites use the Google Maps API, making it the most heavily used web application development API.

The Google Maps API is free for commercial use, provided that the site on which it is being used is publicly accessible and does not charge for access, and is not generating more than 25,000 map accesses a day. Sites that do not meet these requirements can purchase the Google Maps API for Business.

The success of the Google Maps API has spawned a number of competing alternatives, including the HERE Maps API, Bing Maps Platform, Leaflet and OpenLayers via self-hosting.[citation needed]. The Yahoo! Maps API is in the process of being shut down.

### 3.2.12 *Google Cloud functions*

Cloud Functions lets application developers spin up code on demand in response to events originating from anywhere. Treat all Google and third-party cloud services as building blocks, connect and extend them with code, and build applications that scale from zero to planet-scale—without provisioning or managing a single server.



### 3.2.13 *Material design*

Material Design (codenamed Quantum Paper) is a design language developed in 2014 by Google. Expanding upon the "card" motifs that debuted in Google Now, Material Design makes more liberal use of grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows. Designer Matías Duarte explained that, "unlike real paper, our digital material can expand and reform intelligently. Material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch." Google states that their new design language is based on paper and ink but implementation will take place in an advanced manner.

Material Design can be used in all supported versions of Android, or in API Level 21 (Android 5.0) and newer (or for older via the v7 appcompat library), which is used on virtually all Android devices manufactured after 2009. Material Design will gradually be extended throughout Google's array of web and mobile products, providing a consistent experience across all platforms and applications. Google has also released application programming interfaces (APIs) for third-party developers to incorporate the design language into their applications. The main purpose of material design is creation of new visual language that combines principles of good design with technical and scientific innovation.

### 3.2.14 *Design Patterns*

By definition, Design Patterns are reusable solutions to commonly occurring problems (in the context of software design). Design patterns were started as best practices that were applied again and again to similar problems encountered in different contexts. They become popular after they were collected, in a formalized form, in the Gang of Four book in 1994. Originally published with c++ and smaltalk code samples, design patterns are very popular in Java and C# can be applied in all object oriented languages. In functional languages like Scala, certain patterns are not necessary anymore.

### 3.2.15 *Java*

Java is a powerful class-based and object-oriented computer programming language that was developed by James Gosling for Sun Microsystems which was acquired by Oracle Corporation. Java is one of the most popular programming languages in use especially for client-server web applications, with a reported 9 million users. It was based on C/C++ syntax design which application developers found familiar. Originally it was designed for interactive television but it was too advanced for the digital cable television at the time (1991).

Java's major goal is portability which is accomplished by compiling the Java code to Java bytecode instead of machine code. Java bytecode instructions are analogous to machine code, but

they are intended to be executed by a virtual machine written specifically for the host hardware. Because Java bytecode runs on any platform that supports a Java Virtual Machine the term “write once, run anywhere” was introduced to application developers.

After the acquisition of Sun Microsystems by Oracle Corporation on January 27, 2010 the Java platform implementation was split into two different distributions. The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs, and the Java Development Kit (JDK) which is intended for software developers and includes development tools such as the Java compiler, Javadoc and a debugger.

As mentioned Java is an object-oriented programming language. That means that the memory management gets more difficult for the programmers due to pointer references. For example, if the programmer holds the reference to an object that no longer exists or is needed then a “null pointer exception” error is thrown. To solve these problems Java is accompanied by Automatic Garbage Collectors that keep the memory up to date by managing the pointers of the created objects. That is the reason why Java does not support C/C++ style pointer usage.

Java is used for the development of Android applications through Android Studio or Eclipse IDEs but the Java bytecode runs on its own virtual machine, optimized for low memory devices.

## CHAPTER 4: MAIN PART

### *4.1 Problem Analysis*

Epihunter is a Belgian start-up founded by Tim Buckinx. "Daddy, you work in digital, can you make a light that turns on when my brain switches off?" This question from Tim's son, led his father to create this new start-up. Epihunter builds digital solutions to help normalize the daily life of people with epilepsy.

Epihunter's first solution, Epihunter Classroom, focuses on children with absence seizures. These make it especially tough in the classroom. We make seizures visible to the teacher and provide parents with an objective overview.

One of Epihunter's most recent projects is the development of an application that notifies the parents when an absence seizure is detected on the Epihunter Classroom app. In this Thesis Epihunter's parent application, epigo is going to be analyzed. The goal of epigo is to extend the possibilities of Epihunter Classroom app by helping the lives of both parents and children with epilepsy. Epigo notifies parents when an absence seizure is detected on real-time. Parents can now check their child's location and receive live video-stream of their child at the moment of an absence seizure.

#### *4.1.1 Problem Description*

During our first meeting in Ghent the CEO of the company addressed the difficulties that a child with epilepsy is facing in its daily life at school. The story's main subject was that the absence seizures are really hard to detect, a seizure can happen at any moment and the child may seem normal. As a result, the teacher can't be aware that this child is having an absence seizure in the classroom. That's why the Epihunter Classroom is created, in order to notify the teachers with a light alert from the child's smartphone. The child is wearing an EEG headset that records real-time brain activity. The EEG data from the headset are processed on child's smartphone in real-time which is placed on the desk. Once an absence seizure is detected the smartphone flashlight is turned on, reminding us the strong question Tim's son made to his father. Now if the teacher notice a turned on flashlight, knows that this child is having an absence seizure and special attention should be given. This is very important cause after a seizure ends, the child can't remember anything at all from the time that it was under an absence seizure. There are many types of seizures, but in this project we are focusing on the absence once. Absence seizures have 3 types depending the duration of them. They can be sort, long and multiple sort seizures.

So, the Blended AIM team was called to give some ideas on who can we expand the possibilities of Epihunter Classroom. After discussing many ideas with the CEO of Epihunter, we decided to develop an application for the parents of kids with epilepsy that works with Epihunter Classroom. The main function of this application is to notify parents when an absence seizure event happens.

## 4.1.2 System Requirements

To complete the project that was given the students had to decompose the main task into simpler and shorter ones that could be handled more effectively. The basic needs of this project were 2 mobile applications, an API for the database connections and a website special for doctors. The tools we used for the development of the applications can export cross-platform software, so the applications are available for Android, iOS and Windows Phone. The first application is the parent app which is called epigo. In this app, parents can receive notifications when an absence seizure occurs. This notification is sent from the Epihunter Classroom app. For the needs of this project we created a dummy app that simulates the events that can happen on Epihunter Classroom app (ex. Seizure event, headset connection). The functionalities of this dummy app can be used for future development on the Epihunter Classroom app. On the doctor's website, the associated doctor will be able to check the seizure data that are saved to the database. The backend services (API) would take care of data storage and manipulation thus allowing the correct connection between the applications.

For better workflow management and easier achievable tasks, the students split the components of the 2 applications into smaller app pages to scale down the work load of each individual task. So, the final decomposition consisted of 2 mobile applications (one for the parents and one for children to simulate the Classroom app event) and the backend services (API). The website development was too much for the given deadline of the project, so only the design mock-ups have been made to help the future development.

Those 2 apps are working together but there are also functionalities on the parent app that can work without the dummy app. We used the stories below, referring to the parent app as Parent and to the dummy app as Child, to help us creating and managing the tasks. For these users, user stories were created in each sprint as upcoming goals. Also, API tasks are referred.

<b><u>Sprint 1</u></b>
As Parent, I want to Sign-in
As Child, I want to Sign-in
As Parent, I want to Log-in
As Child, I want to Sign-in

**Sprints Table 1**

<b><u>Sprint 2</u></b>
As Child, I want to track and display the time that I wear the EEG headset
API has to be created to make the connection between Parent and Child
As Child, I want to simulate a seizure event
As Child, I want to know if the headset doesn't record EEG data

**Sprints Table 2**

<b><u>Sprint 3</u></b>
As Child, I want to send a notification
As Parent, I want to receive a notification
As Child, I want to record a video during a seizure
As Child, I want to know my GPS location

**Sprints Table 3**

<b><u>Sprint 4</u></b>
As Child, I want to send live-stream video to the Parent
As Parent, I want to receive live-stream video from the Child
API has to store the seizure data on the database

**Sprints Table 4**

<b><u>Sprint 5</u></b>
As Parent, I want to see the headset connection status from my app
As Parent, I want to watch the recorded streams available on the database

**Sprints Table 5**

<b><u>Sprint 6</u></b>
As Parent, I want to visualize the daily seizure data
As Parent, I want to visualize the monthly seizure data
As Parent, I want to have settings on my app

**Sprints Table 6**

<b><u>Sprint 7</u></b>
As Child, I want to send my GPS Location
As Parent, I want to view the GPS Location of the Child
As Parent, I want to set reminders for Medication

**Sprints Table 7**

<b><u>Sprint 8</u></b>
As Parent, I want to add a seizure event manually
As Parent, I want to see a recoded seizure details(duration, type)
As Parent, I want to comment on a seizure
API has to retrieve the requested data from the database

**Sprints Table 8**

<b>Doctor's website Sprint</b>
As Doctor, I want to have a list with all my patients
As Doctor, I want to add patients
As Doctor, I want to remove patients
As Doctor, I want to have a monthly overview of the seizures
As Doctor, I want to have a daily view of the seizures
As Doctor, I want to filter the recorded data
As Doctor, I want to have a view with details of the seizure, the EEG data and the video
As Doctor, I want to zoom in out to EEG graph
As Doctor, I want by moving the EEG cursor redirect to the exact timestamp of the video
As Doctor, I want to comment on a seizure

**Doctor Sprint Table 1**

## *4.2 Implementation*

### *4.2.1 Project Implementation*

In Blended AIM 2018, 19 students participated from universities in Germany, Portugal, Greece, Austria, Belgium, Scotland and Iraq. Students were divided in 2 teams and worked on different projects. At first, the 2 teams had to get acquainted with each other and also with the Scrum framework that they would use for the agile software development. Following the Nexus Scrum, they rearranged the initial team into separate smaller ones consisting of the design team, the application development team, the business team and the marketing team. There were even smaller sub-teams in the development team to make the software development faster. The rearrangement was done according to the student's educational field. The first Sprint took place in Ghent, Belgium (IMEC company headquarters). During this week we made the project goals more clear and we discussed the technologies that would be used and on the final product with Epihunter's CEO. After that meeting each student returned to his country and that made the project management a bit harder, mostly on communication. Every 2 weeks, on Monday the students had a general meeting where all the students from all the teams participated and there was an update from every member about their accomplishments and their future goals. There were also internal of the app team meetings about the app development every Thursday. Those meetings were very important in order to discuss the problems we had on the development and find solutions. Also, every 2 Thursdays, a sprint review meeting was held where every student had to present the work that was done and discuss with all the team members about the next tasks. A meeting with Epihunter's CEO was held after the end of each sprint. The leaders of each team participated to update the customer about the progress and answer to any questions from him.

The whole project was divided in 8 sprints, one every two weeks, until its completion. There was daily communication between the students through the Slack application. Meetings were hosted on Skype. That type of communication needed some time to be adapted by all the members but

after some sprints everyone was used to it and the communication was really direct. Although, it was really difficult to find a convenient time to schedule the meetings cause all the students had different time zones and daily routines.

All the tasks were held on Trello and every scrum team had its own tasks. In the end of each Sprint, the product owner had to assign new tasks for the upcoming Sprint. When a task wasn't completed till the end of a Sprint, it was transferred to the next Sprint and marked as priority. That meant it has to be done as soon as possible before moving to the new tasks.

The last Sprint took place in Heraklion, Greece (in Technological Institute of Crete) where the final presentation of the product also took place. During the last week of the last Sprint students worked together to finalize their work, fix existing bugs and problems. Demo of the application was presented and preserve the certainty of a smooth run, demonstrating all the developed features of the application. At the end of the product presentation the students work was evaluated by the company representatives and the professors, then questions and open discussion were made. The students also were rated for their participation by each other.

### 4.2.2 Detailed Project Implementation

At the beginning of the project we were divided in sub-development teams that were targeting a specific product. The end-products were 2 mobile applications and an API, the website will be developed in the future. My role on the development was really agile and flexible, so during this project I have worked on all the products but mostly on the mobile applications. Also, I was helping other members by finalizing tasks that weren't fully implemented. Bellow there is a further explanation of the exported tasks from the user stories. Furthermore, some of the code as well as certain techniques that were used for the completion of the application will be explained. In the next tables the user stories and the exported tasks of each product are presented.

<b>Parent App</b>	
<b>User story: As Parent, I want to Sign-in</b>	<b>Tasks</b>
1. Register to firebase 2. UI of Registration page 3. Backend of Registration Page	A new Ionic page has to be created. Registration form. Create a provider to handle the registration requests to firebase. UI design implementation.
<b>User story: As Parent, I want to Log-in</b>	<b>Tasks</b>
1. Login to firebase 2. UI of Login page 3. Backend of Login Page	A new Ionic page has to be created. Login form. Create a provider to handle the authentication requests to firebase. UI design implementation.

#### Parent Story 1

<b>Parent App</b>	
<b>User story:</b> As Parent, I want to receive a notification	<b>Tasks</b>
1. Receive cloud notifications from Child App 2. Display notifications on the app	Create a provider to handle the notifications requests from the cloud functions. Create pop-over window on the app when a notification is received

### Parent Story 2

<b>Parent App</b>	
<b>User story:</b> As Parent, I want to receive live-stream video from the Child	<b>Tasks</b>
1. Receive live-stream video from Child App 2. Open video when a notification is tapped	Create a provider to handle the live video requests from the cloud functions. Create new div on the app template to stream the video elements Auto-answer to calls Add a button to cancel the call

### Parent Story 3

<b>Parent App</b>	
<b>User story:</b> As Parent, I want to see the headset connection status from my app	<b>Tasks</b>
1. UI on the top nav-bar 2. Get connection status	Display headset connection with an icon on the top nav-bar(active, non-active) Change icon once the headset is connected or disconnected Get a notification if the connection is lost
<b>User story:</b> As Parent, I want to watch the recorded streams available on the database	<b>Tasks</b>
1. UI of the list with the recorded videos 2. Get videos list from cloud 3. UI of the sub-video page(Selected video) 4. Comment on Video	Create a new Ionic page for the available recorded videos. Fetch available videos from cloud storage. When a video is selected from the list new page is now open with a video player and a comment box. The parent can comment on a video and the comments are stored on the database.

### Parent Story 4



<b>Parent App</b>	
<b>User story:</b> As Parent, I want to visualize the monthly seizure data	<b>Tasks</b>
1. UI of the monthly Calendar view 2. Select a date	Create a new Ionic page for the monthly calendar view. Selected a date from the calendar view Redirect to daily calendar view and fetch the data of the selected data via API.
<b>User story:</b> As Parent, I want to visualize the daily seizure data	<b>Tasks</b>
1. UI of the daily Calendar view 2. Get stored seizure data from via API	Create a new Ionic page for the daily calendar view. Fetch the data of the selected date from the database via API.
<b>User story:</b> As Parent, I want to have settings on my app	<b>Tasks</b>
1. UI of Settings Page 2. UI of Account Page 3. Insert settings icon on to nav-bar 3. Insert notification icon on to nav-bar	Create a new Ionic page for the settings page. Enable light alerts. Enable vibration alerts. Enable sound alerts. Change Language. View Terms of Service Create a new Ionic page for the account page. Change username Change password Logout Insert a gear icon on the top nav-bar so the settings are easily assessable. Insert a bell icon on the top nav-bar to turn on/off the notifications.

### Parent Story 5

<b>Parent App</b>	
<b>User story:</b> As Parent, I want to set reminders for Medication	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. UI of Medication Page</li> <li>2. Set medication reminders</li> <li>3. Insert medication records on the database</li> </ol>	Create a new Ionic page for the monthly calendar view. Selected the name of the medication. Select a time to get a reminder. Select an additional time to get a reminder. Select the days you what get reminders for this medicine. Select the recurrence of the medicine. Add comments for the medication. Set the reminder.
<b>User story:</b> As Parent, I want a map view of the GPS Location of the Child	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. Receive GPS Location from Child App</li> <li>2. Display location on the map.</li> <li>3. Extend UI in the live-video page</li> </ol>	Initialize the map with Google Maps API Get the GPS coordinates from the database. Set the coordinates on the map and display them on the live-video page.

### Parent Story 6

<b>Parent App</b>	
<b>User story:</b> As Parent, I want to add a seizure event manually	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. UI of the add Seizure view</li> <li>2. Insert a new Record on the calendar</li> <li>3. Store new data via API</li> </ol>	Create a new pop-over page to add seizure event manually. Select seizure type. Set duration (time seizure started/ended). Insert data on the selected date of the calendar view and store them via API. Redirect to daily calendar view and fetch the data of the selected data via API.
<b>User story:</b> As Parent, I want to see a recoded seizure details(duration, type)	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. UI of the seizure details view</li> <li>2. Get stored seizure data from via API</li> </ol>	Create a new Ionic page for the seizure details view. Fetch the data of the selected date from the database via API and display them in detailed view. Delete a record form the database with API.
<b>User story:</b> As Parent, I want to comment on a seizure	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. Extend UI of seizure details page</li> <li>2. Get seizure details from database</li> <li>3. Comment on a seizure</li> </ol>	Details of the selected seizure are displayed. Parent can make comments on the seizure. Comments are stored on the database via API.

### Parent Story 7

Child App	
<b>User story:</b> As Child, I want to Sign-in	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. Register to firebase</li> <li>2. UI of Registration page</li> <li>3. Backend of Registration Page</li> </ol>	A new Ionic page has to be created. Registration form. Create a provider to handle the registration requests to firebase. UI design implementation.
<b>User story:</b> As Child, I want to Log-in	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. Login to firebase</li> <li>2. UI of Login page</li> <li>3. Backend of Login Page</li> </ol>	A new Ionic page has to be created. Login form. Create a provider to handle the authentication requests to firebase. UI design implementation.

### Child Story 1

Child App	
<b>User story:</b> As Child, I want to track and display the time that I wear the EEG headset	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. Activate a timer(Trigger event)</li> <li>2. Deactivate the timer(Trigger event)</li> <li>3. Get timer duration, timestamp (Note: This will be used in later tasks to push cloud notifications)</li> </ol>	Create a toggle button in frontend to activate and deactivate the timer. Create a timer on backend to count the time duration. Get a timestamp and the time duration once the timer is deactivated.
<b>User story:</b> As Child, I want to simulate a seizure event	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. Activate a timer(Trigger event)</li> <li>2. Deactivate the timer(Trigger event)</li> <li>3. Get timer duration, timestamp (Note: This will be used in later tasks to push cloud notifications)</li> </ol>	Create 2 buttons in frontend to activate and deactivate the timer. Create a timer on backend to count the time duration. Get a timestamp and the time duration once the timer is deactivated.
<b>User story:</b> As Child, I want to know if the headset doesn't record EEG data	<b>Tasks</b>
<ol style="list-style-type: none"> <li>1. Trigger event (Note: This will be used in later tasks to push cloud notifications)</li> </ol>	A new button has to be created in frontend. A pop-up notification should be triggered.

### Child Story 2

<b>Child App</b>	
<b>User story:</b> As Child, I want to send a notification	<b>Tasks</b>
1. Sent a cloud notification when an event is triggered	Built a Google cloud function to push notifications. Get userID and deviceID to send a notification. Create a provider on the app to trigger notifications on the cloud functions. Sent a push notification to the parent app when an event is triggered (Seizure, Headset Connection).
<b>User story:</b> As Child, I want to record a video during a seizure	<b>Tasks</b>
1. Activate the camera 2. Record video 3. Stop recording when seizure ends (Note: This will be used in later tasks to make the live-stream video calls)	Trigger recording on a seizure event. Stop recording when a seizure ends Store the video locally and play it.
<b>User story:</b> As Child, I want to know my GPS location	<b>Tasks</b>
1. Get GPS Location	Trigger a geolocation request on the start of a seizure.

### Child Story 3

<b>Child App</b>	
<b>User story:</b> As Child, I want to send live-stream video to the Parent	<b>Tasks</b>
1. Sent live-stream video to the Parent App	Create a provider to handle the live video function. Give camera permissions to the app. Initialize web client for the call. Make a call. Record video call. Store it on cloud.

### Child Story 4

<b>Child App</b>	
<b>User story:</b> As Child, I want to send my GPS Location	<b>Tasks</b>
1. Push GPS Location through a notification	Pass GPS coordinates through the notification when a seizure event is triggered.

### Child Story 5

<b>API</b>	
<b>User story:</b> API has to be created to make the connection between Parent and Child	<b>Tasks</b>
1. Connection to firebase 2. Pull requests 3. Push requests 4. Implementation as service to the apps	Built the Java API (Spring Boot). Connection between Firebase API & our own API. Make test request. Implement API to the apps. Test connection

**API Story 1**

<b>API</b>	
<b>User story:</b> API has to store the seizure data on the database	<b>Tasks</b>
1. Store data to firebase	Write tests for the API. Set up API methods to store data into the database. Connect with Child App with API to feed through simulation data.

**API Story 2**

<b>API</b>	
<b>User story:</b> API has to retrieve the requested data from the database	<b>Tasks</b>
1. Add API to the parent app 2. Handle requests 3. Fetch data from firebase	Add APIHandler service to parent app. Get the data from API into Parent-app. Convert parent app to accept new JSON data format. Add offline support.

**API Story 3**

### 4.2.2.1 *User Interface Design*

The user interface was designed by our designer's team. During our first Sprint, back in Ghent, the designers had to select the colors, the fonts and the app icons were going to use. They had also to present some sample mock-ups of the app. There were 2 approaches on the app colors. The first one was to use the same colors as the company's website. That meant the main colors were green, light green, cyan and white. The second approach had totally different colors with the main colors being purple, gray, cyan and white. The icon of the app stayed the same as company's logo. The final selection from the company was to keep the colors from their website and use the Dosis font that we suggested for our texts.

After the colors and font selection the app mock ups for each page had to be made using Material Design. We had 8 different versions of the design mock-ups. That caused some problems on the development cause even till the last sprints changes were made. That affected the UI implementation on the front-end and with the new designs some extra back-end functions had to be developed. In every Sprint the mock-ups were reviewed by the company and useful feedback about them was given on what has to be changed. Having so many iterations on the designing definitely affected the development speed. Although, the final design was exactly what the company wanted and a big part of it also implemented in the app.

Each page had its own mock up. The developers had to implement the design on the app as it was on the mock-up. All the icons and design elements were provided from the designers to the developer team to help the design implementation on the app. It was a really efficient way for the design team to work with the developers. By reviewing a mock-up as a developer you knew what you have to develop both in front-end and back-end. Also a site-map with all the app pages was provided from the designer's team to help developer with the navigation between the pages.

The design implementation with the Ionic Framework actually wasn't very difficult. An Ionic Page has the same structure as a webpage. That means each individual page consists of these 3 files:

- Page.html (HTML file)
- Style.scss (Style CSS file)
- Backend.ts (AngularJS typescript file)

It really reminds us a webpage and the development of it it's pretty much the same, but it's also enhanced with Ionic and Cordova Plug-ins that helps the design and the functions implementation. AngularJS is handling the back-end of the page and it has the same role as a JavaScript file. So modifying those 3 files we can implement any style we want in each page as we would on a webpage. There are some small differences on the Ionic and AngularJS from the classic web-development but for a developer that is experienced with web-development it will be really easy to adapt and use the Ionic framework to develop cross-platform mobile applications.

Every design element was placed under the assets folder which is contained inside the source folder of our project. In this folder there were 4 subfolders that contained images, icons, fonts and data.

- img  
All the images of the app are contained in this folder. Those are design elements files (.png .svg) that are used on the app pages and menus.
- icon  
Application icon is contained (.ico).

- fonts  
All the files that need for the Dosis font are contained in this folder (.ttf)
- data  
A JSON tree is contained with calendar and medication data that are used in the app (.json)

There is also a theme folder inside the source folder, where the font and the selected colors are implemented. In this folder there is a file called variables.scss and by modifying it we changed the app default colors and font.

- Colors  
There are many colors in the app, so there are many different color codes. We saved all the color codes we used in a list as variables. Doing that helped a lot during the styling of the app because the developers were able to set a color by its given name without typing the color code. Here is an example:

```
$colors: (
  primary:    #289784, // #3b5998, // #36008A
  secondary:  #8EC853,
  danger:     #f53d3d,
  light:      #f4f4f4,
  dark:       #222,
  darkgreen:  #248875,
);

// Use the primary color as the background for the toolbar-md-background
$toolbar-md-background: color($colors, primary);
```

- Font  
The Dosis font we stored in the assets folder has to be imported to this file.

```
//for words
@font-face {
  font-family: "Dosis";
  src: url("../assets/fonts/Dosis-Light.ttf") format("truetype");
}
```

As we told above, each individual page consists of an html, a scss and a typescript file. But on top of those pages there is the app page. This is the root page of the app and it's the most important of them all. It handles boot-up of the app by initializing the pages that have to be displayed. It also handles other important functions like the initialization of the APIs and the database service. The root page is always running while the application is active or running in the background. That's why the root pages in Ionic development are mostly used to implement menus that would handle the navigation and will also be accessible from every page on the app. The most important part of the root page is its back-end code. For example, there we can check if a user is logged in or not. If the user is logged in we give access to all the pages and the logged in menu. Otherwise, the user has access only to the logged out pages and the logged out menu. This

happens by giving dynamically the right permission to the app page by the user connection status. Another important function that is also implemented in the app page is the notification part. As long as the app page is running always when the app is active or in background is important to receive notifications in every page of the app but also even you are not using the app but its running on the background. The structure of the app page is the following:

### App source folder

- app.component.ts

In this typescript file all the back-end code exists. Using some functions right permissions are given to the users. Also, the notifications are received and also important APIs (like the API for the live-stream video) are initialized. Some plug-ins are imported in this file to extend the functionalities. With these plug-ins the app can operate on background mode.

This script is handling which menu is going to be displayed to the user:

```
loggedInPages: PageInterface[] = [
  { title: 'Account', name: 'AccountPage', component: AccountPage, icon: 'person' },
  { title: 'Support', name: 'SupportPage', component: SupportPage, icon: 'help' },
  { title: 'Logout', name: 'TabsPage', component: LoginPage, icon: 'log-out', logsOut: true }
];
loggedOutPages: PageInterface[] = [
  { title: 'Login', name: 'LoginPage', component: LoginPage, icon: 'log-in' },
  { title: 'Support', name: 'SupportPage', component: SupportPage, icon: 'help' },
  { title: 'Signup', name: 'SignupPage', component: SignupPage, icon: 'person-add' }
```

```
enableMenu(loggedIn: boolean) {
  this.menu.enable(loggedIn, 'loggedInMenu');
  this.menu.enable(!loggedIn, 'loggedOutMenu');
}
```

- app.html

This HTML file is used for the navigation of the other pages.

- app.modules.ts

In this file all the declarations of the ionic pages, services (API) and providers (Firebase Auth, Data) is made. It operates like a local library of all components of the app so connections between the pages ,services and providers can be made.

- app.template.html

This HTML file is used to visualize the side menus. There are 2 menus with 2 different ids (logged in, logged out). The displayed menu to the user depends from the user connection status.

```
<!-- Logged out menu -->
<ion-menu id="loggedOutMenu" [content]="content">

<!-- Logged in menu -->
<ion-menu id="loggedInMenu" [content]="content">
```

- app.scss

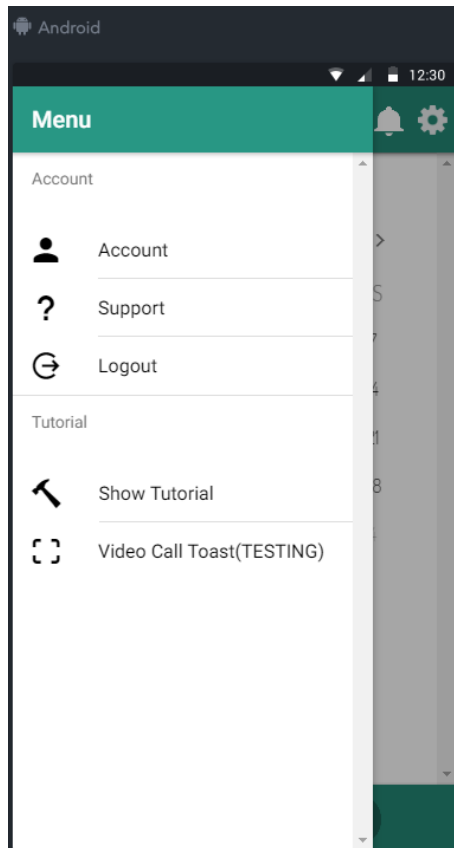
The styling variables of the app are in this file.

- main.ts

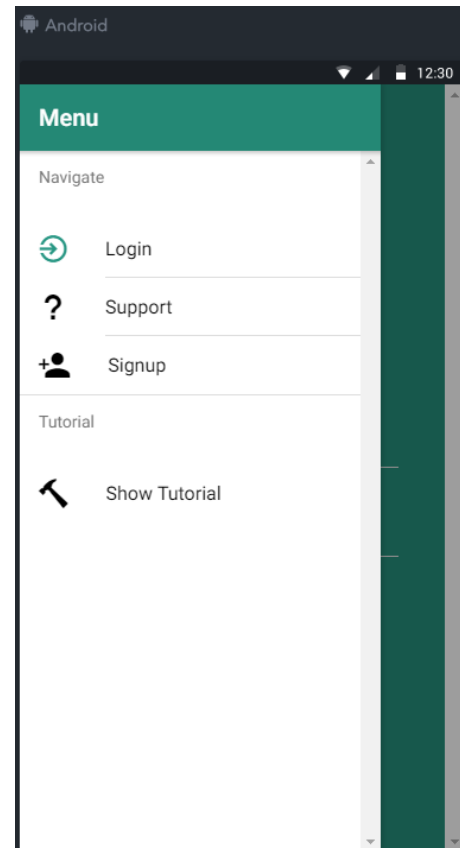
This typescript file is used to bootstrap AppModule using the browser platform.



Bellow there are screenshots of the 2 side menus from the app.template.html file.



Picture 2, Logged In Side-Menu



Picture 1, Logged Out Side-Menu

## Tabs Page

In this page 2 navigation bars are contained. The bottom nav-bar is used to navigate the user between the 3 main pages of the app. There are 3 tabs that can navigate to these pages:

- Calendar Page
- Medication Page
- Videos Page

This is the code for the tabs from the tabs-page.html file:

```
<ion-tabs [selectedIndex]="mySelectedIndex" name="conference" >
  <ion-tab [root]="tab1Root" tabIcon="calendars" tabUrlPath="conference-schedule"></ion-tab>
  <ion-tab [root]="tab2Root" tabIcon="meds"></ion-tab>
  <ion-tab [root]="tab3Root" tabIcon="videos"></ion-tab>
</ion-tabs>
```

This is a screenshot of the tabs-page bottom nav-bar



Picture 3, Bottom navigation bar

In the top nav-bar, there are 3 clickable icons and a non-clickable icon that is displaying the EEG headset connection status from the Child app. This icon has 2 statuses, active and non-active. The 3 clickable icons are:

- Menu Icon
- Notification Bell (active and non-active)
- Settings Page (Gear Icon)

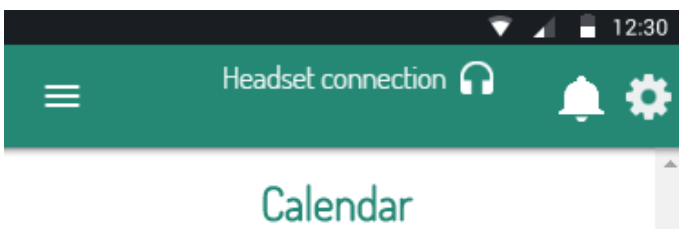
If the Menu Icon is tapped the side menu shows up. By tapping the notification bell you can turn on or off the notifications and by tapping on the Gear Icon user is navigated to the settings page.

```
<ion-header>
  <ion-navbar no-border-bottom color="darkgreen">
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-buttons right>
      <button ion-button icon-only (click)="gotosetting()">
        
      </button>
    </ion-buttons>

    <div class="txtheadset" >Headset connection
      
    </div>

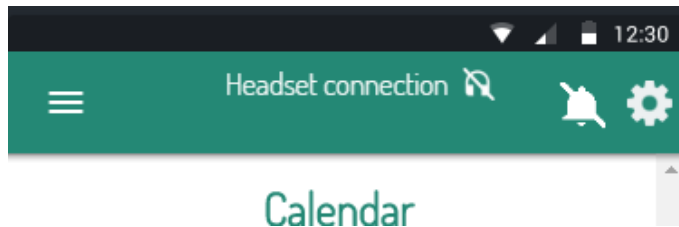
    <div style="position:fixed;right:12%;bottom:17%;">
      
    </div>
  </ion-navbar>
</ion-header>
```

This is a screenshot of the tabs-page top nav-bar with active notifications and headset connection.



Picture 4, Top navigation bar 1

This is a screenshot of the tabs-page top nav-bar with non-active notifications and headset connection.



Picture 5, Top navigation bar 2

Tabs-page may don't have any inner content but it saved us a many lines of code in this project. By displaying the tabs-page over the other main pages, we didn't have to write the same code both on front-end and back end, over and over again in every page, just to display the 2 nav-bars. That was really inefficient and developing a page just for the nav-bars was a really nice solution.

## Pages

Having built the main components for the navigation and the menus, rest pages had to be implemented. The application consists of 15 Ionic pages in total, which 7 of them are activities, 5 are fragments and 3 of them are modal pages.

- Activity\_login  
In this layout, there is an image with company's logo, 2 text fields for the username and password and a button for the login. There is also a clickable text with a link that redirects the user to the register page.
- Activity\_register  
In this layout, there is an image with company's logo, 4 text fields for the email, username, password and confirmation of the password and a button for the registration.
- Activity\_main  
In this layout, the layouts of the side-menu and the navigations bars are contained. On top of that the layout of the fragment that is in use is shown accordingly.

These fragments are:

- Fragment\_calendar

In this fragment, a calendar is contained that operates as a date selector. The user can select any month and day he wants. Then the user is redirected to the daily layout of the selected day. In this layout all the recorded seizure events of this date are displayed in a timeline as colored dots. Each dot represents a seizure type. There are 4 types of seizures and each one of them has a different color so it's easily recognizable from the user. The seizure types are:

- Sort (Yellow dot)
- Multiple sort (3 Yellow dots)
- Long (Orange dot)
- Motor (Red dot)



Picture 6, Seizure Category Color Codes

There are also 2 modal layouts in the daily layout:

➤ Modal\_seizure\_details

In this modal layout, details of a seizure are displayed. This layout is presented to the user if a dot from the timeline is tapped. The data of the seizure event are the seizure type, start and end time of the seizure (duration) and comments on the seizure that can help the doctor with the diagnosis. Those data are displayed as text.

➤ Modal\_add\_event

In this modal layout, a seizure event can be added manually by the user. The user has to select the seizure type (by tapping on the seizure dots) and the start and end time of the seizure with 2 time selectors. Then by pressing the add button the data are stored on the database and the new event is also displayed on the timeline. There is also an exit button if the user doesn't want to make an entry and a button that redirects the user to the medication page.

○ Fragment\_medication

In this fragment, the user can set reminder for a medication. There is a text field to insert the medication name and 2 time selectors. Under those entries there is a (3x3) grid with the days of the week. In this grid, the user can select the days that the medication reminder is going to be sent. There is also an integer selector to set the recurrence of the medication for the following weeks. At the end, there is a text box to add details for the medication and a button to set the reminder.

○ Fragment\_videos

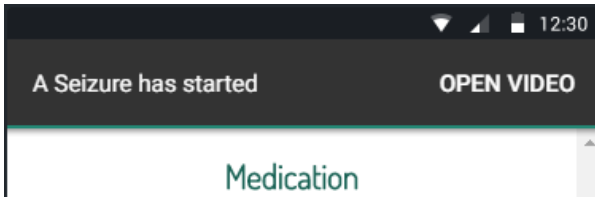
In this fragment, the user can see a list of the available recorded videos from the live-stream video calls. For each record of the list, the seizure type is displayed (as a colored dot), the start and end time and the duration of the seizure. There is also a star in each record so the user is able to mark the videos that are useful for the doctors. The date and the day are also displayed before the records. When a recorded is tapped from the user, a modal page is open to watch the video.

➤ Modal\_video\_player

In this modal layout, the video and the data of selected record from the video fragment are displayed. On top there is the date of the video and below it is a video player. Below the video player there are the seizure data displayed as a text and a comment box to add comments that are useful for the doctors. There is also a star icon to mark the video and a trash bin icon to delete it.

- Activity\_notification

This layout is triggered once a notification is received. A pop-over toast is presented at the top of the screen to notify the user about the new seizure event. This toast is presented on top of any layout of the app. If the user taps on the toast, the toast is dismissed and 2 fragment pages are created to show the location of the child and handle the live-stream video.



Picture 7, Notification pop-over

- Fragment\_maps

In this fragment, the user can see the location of the child. The coordinates of the location are received from the Child app via the API. Once the coordinates are received, a map is initialized using the Google maps API and the given coordinates. The map is now displayed and the location of the child at the center of the screen.

- Fragment\_live\_video

In this fragment, the user is receiving live video stream from the Child app during a seizure event. This fragment is actually displayed as a right-side menu of the maps fragment. Instead of the usual menu contents, in this side-menu the video-stream of the child is displayed and on the left bottom the parent video is streamed. There is also a button at the bottom to close the stream. That view allows the user to navigate from the live-stream to the map fragment and vice versa. This is important for the UX (user experience) of the app because the user has access to a full screen map inside the app and at the same time a view of the live video-stream.

- Activity\_settings

In this layout, there are 3 icons and by tapping them the user can select which types of alerts (vibrate, light, sound) are going to be set. Bellow there are 2 buttons and a drop-list menu. The first button redirects the user to the account settings and the second one redirects the user to the Terms of service layout. The Terms of service layout is displayed also on the first installation of the app and the user has to accept the term to start using the app. From the drop-list menu, the user can select a different language for the app.

- Activity\_account\_settings

In this layout, the user's username is displayed bellow an icon and there are 3 buttons to change the account settings. By tapping a button a pop-over window is created. On the first one there is a text field with the username and a button to change the username. On the second window, there are 2 password fields and a button to change the user's password. In

the last window there are 2 buttons the user can log out of the app or cancel the window. If the user logs out is redirected to the login page.

- Activity\_tutorial

In this layout, a sort tutorial of the app is displayed to the user.

#### 4.1.2.2 *Firestore*

To store our data we used Google's latest platform, Firestore. Firestore is a great platform for mobile apps development. Firestore gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users. Firestore has a majority of products that are really useful for mobile app development. Each feature works independently, and they work even better together. Those products can interact to each other and boost the app functionalities. For the needs of our project we used 3 of those products:

- Cloud Firestore (beta):

Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for mobile and web apps - at global scale. We used this database to store all our data. This database is still in beta version but it seems really promising because it can easily interact with other Firestore products like the Cloud functions were notifications can be pushed to the users.

- Cloud Functions:

Using Cloud Functions we were able to develop a backend without using any servers. Those functions are triggered by Firestore products, such as changes to data in the Firestore Database and new user sign-ups via Google's Authentication. There is more information about the Cloud Functions on the next section.

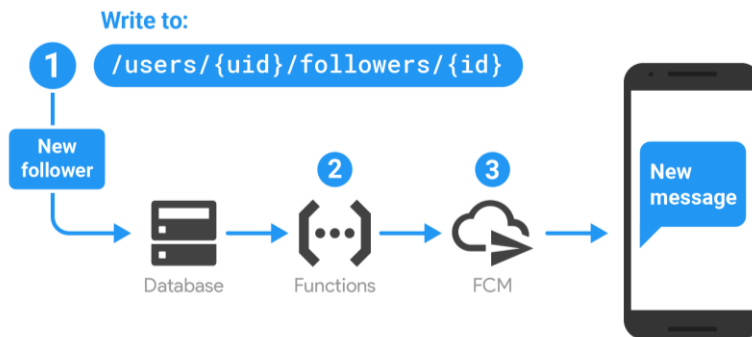
- Authentication:

Firestore Authentication aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end identity solution, supporting email and password accounts, phone auth, and Google, Twitter, Facebook, and GitHub login, and more. We used Firestore Authentication to authenticate our app users and we also used the unique generated id of each user to store the user's data on Firestore.

#### 4.2.2.3 *Google Cloud Functions*

Cloud Functions for Firestore can automatically run backend code in response to events triggered by Firestore features and HTTPS requests. The backend code is stored in Google's cloud and runs in a managed environment. A Cloud Function is developed to push notifications between the Child and the Parent app. When a seizure event occurs, the seizure event details are stored on the cloud database via the RestAPI. Once the data are written on the database, the function is starting its execution [1]. On the execution, the function composes a message to send via FCM (Firestore

Cloud Messaging) [2]. At the end of the execution, FCM sends the notification message to the user's device [3].



Picture 8, Cloud Messaging

On the example above, we can see the connections between the database, the cloud functions and the FCM (Firebase Cloud Messaging). The above function sent a notification to the user if user has a new follower. On [1] we can see there is a data path. Once new data are written to this path, the cloud function is triggered. With Cloud Functions, you can handle database events. Cloud Functions supports these event handlers:

- `onWrite()`, which triggers when data is created, updated, or deleted in the Database.
- `onCreate()`, which triggers when new data is created in the Database.
- `onUpdate()`, which triggers when data is updated in the Database.
- `onDelete()`, which triggers when data is deleted from the Database.

In the case we study, each time new data are written to the database. So by using the `onCreate()` method, we can handle the data creation event. Here is some code from the function:

```
5 //cloud function name
6 exports.seizure_notification = functions.firestore
7   .document('users/{userId}/seizure/{seizureId}')
8   .onCreate(async event => {
```

With this code [2], anytime a new record is inserted to the database, an asynchronous event is created to send the payload of the notification to the Parent's device. We use this async event, to return a promise once the notification is successfully send and it's also used to reduce the function execution time. Here we can see the notification payload:

```
6 exports.seizure_notification = functions.firestore
7   .document('users/{userId}/seizure/{seizureId}')
8   .onCreate(async event => {
9     // Notification content-payload
10    const payload = {
11      notification: {
12        title: 'Seizure',
13        body: `A seizure has just started!`
14      }
15    }
```

Once the notification payload is created, the notification is ready to send on the Parent's device. To do this, the device token from the parent's phone is needed. Each device token is unique and it's retrieved and stored to the database after the parents log in with their device. Using this async function on the parent app the device token is saved to the database:

```

18 // Get permission from the user
19 async getToken() {
20   let token;
21   if (this.platform.is('android')) {
22     token = await this.firebaseNative.getToken()
23   }
24   if (this.platform.is('ios')) {
25     token = await this.firebaseNative.getToken();
26     await this.firebaseNative.grantPermission();
27   }
28   return this.saveTokenToFirestore(token)
29 }

```



The device token is also linked with the user Id of each user. So now, by using the FCM we can send the payload to the device that is linked with the user Id. With command we can send an FCM notification:

```

21   return admin.messaging().sendToDevice(firebase.firestore.document('users/{userId}/devices/{deviceId}'), payload)
22
23 });

```

Once the message is successfully sent a successful promise is returned to the async event that was created before. Otherwise, an error is returned. The function is executed really fast, so the notification is arriving on the parent app almost in real-time. Here we can see the median execution time from the Google Cloud Platform.

Function	Event	Executions	Median run time
seizure_notification	 document.create /subscribers/{subscriptionId}	15 	116.68ms

Picture 9, Cloud Function Execution Time

#### 4.2.2.4 RestAPI (Spring boot)

A Rest API service was created using spring boot to make the connections between the apps and the database (Firestore). The API was split into several classes each class for a specific purpose. Some of the classes were used to write data on the database. For example, the data that are written to the database via API services are: seizure data (start/end time, duration), the headset connection status, the user's device ID (to send notifications), map coordinates and medication data. So, 5 methods for writing purposes were developed. Below are the snippets of 2 methods is shown:

The snippet of the device ID:



```

815 // Devices and Subscriber methods for notifications
816 public void addDevices(String token, String userId){
817     Map<String, Object> data = new HashMap<>();
818     data.put("token", token);
819     data.put("userId", userId);
820     ApiFuture<WriteResult> result = db.collection("devices").document().set(data);
821 }

```

Using the addDevices method of the RestAPI, the device token and the user ID are linked together. Then the result is written to the devices collection of Firestore.

The snippet of map coordinates:

```

157 public void addMap(MapCollection toAdd) throws Exception{
158     Map<String, Object> data = new HashMap<>();
159
160     data.put("name", toAdd.getName());
161     data.put("lat", toAdd.getLat());
162     data.put("lng", toAdd.getLng());
163     data.put("center", toAdd.getCenter());
164     ApiFuture<WriteResult> result = db.collection("map").document().set(data);
165 }

```

Using the addMap method of the RestAPI, the latitude and longitude (center of the map) of the child's location are written temporally on the database map collection of Firestore. On a seizure event, the parent can retrieve the map data on his device and using Google Maps the data are visualized.

Read methods were also developed to get data from the database into the applications. Those read methods were used to get seizure details data, medication data, and map data using the user's ID to fetch the right data. Below are the snippets that are used to get calendar data:

The snippets of the calendar data:

```

355 public ArrayList<Calendardata> getCalendarByUserId( String userId) throws Exception{
356     ArrayList<Calendardata> results = new ArrayList<Calendardata>();
357     String trueId = userId;
358     String splicedUserId = userId.replaceAll("\\\\", "");
359
360     ApiFuture<QuerySnapshot> query = db.collection("CalendarData").whereEqualTo("userId", splicedUserId).get();
361     QuerySnapshot querySnapshot = query.get();
362
363     // Loop over all retrieved documents and add them to arrayList
364     List<QueryDocumentSnapshot> documents = querySnapshot.getDocuments();
365     for (QueryDocumentSnapshot document : documents) {
366         Calendardata filled = new Calendardata();
367         filled.setUserId(document.getString("userId"));
368         String documentId = document.getId();
369         ArrayList<Days> foundDays = new ArrayList<Days>();
370
371         foundDays = getAllDaysFromCalendar(documentId);
372         filled.setDays(foundDays);
373         results.add(filled);
374     }
375     return results;
376 }

```

In this snippet, the API using a query, searches the CalendarData collection of Firestore to find the users ID in this collection. Once it's found, a for loop is used to go through all the user's documents. An Array List (foundDays) is created to store all the data of the different days that were found on CalendarData collection. Then, another method (getAllDaysFromCalendar) is called to get all the data of each day by the document ID. The results of this method are returned and stored on the foundDays list. This method is shown below:

```
379     public ArrayList<Days> getAllDaysFromCalendar(String documentId) throws Exception{
380         ArrayList<Days> results = new ArrayList<Days>();
381         ApiFuture<QuerySnapshot> query = db.collection("CalendarData").document(documentId).collection("Days").get();
382         QuerySnapshot querySnapshot = query.get();
383         // loop over all retrieved documents and add them to arraylist
384         List<QueryDocumentSnapshot> documents = querySnapshot.getDocuments();
385         for (QueryDocumentSnapshot document : documents) {
386             Days filled = new Days();
387             String dayId = document.getId();
388             filled.setDate(document.getString("date"));
389
390             ArrayList<Daydata> foundData= new ArrayList<Daydata>();
391             foundData = getAllDayDataFromCalendar(documentId, dayId);
392             filled.setDaydata(foundData);
393             results.add(filled);
394         }
395         return results;
396     }
```

In this snippet, the API using a query, searches the Days collection of the CalendarData collection, to find all the days with data by the document ID. The document ID is known method from the previous method (getCalendarByUserId) that it was called. An Array List (Daydata) is created to store all the data that were found on Days collection. Then, another method (getAllDayDataFromCalendar) is called to get all the daily data of each day using the document ID and the day ID. Using a for loop the API goes through all the documents of this user. This method is shown below:

```

415     public ArrayList<Daydata> getAllDayDataFromCalendar(String documentId, String dayId) throws Exception{
416         ArrayList<Daydata> results = new ArrayList<Daydata>();
417         ApiFuture<QuerySnapshot> query = db.collection("CalendarData").document(documentId)
418             .collection("Days").document(dayId).collection("Data").get();
419         QuerySnapshot querySnapshot = query.get();
420         // loop over all retrieved documents and add them to arraylist
421         List<QueryDocumentSnapshot> documents = querySnapshot.getDocuments();
422         for (QueryDocumentSnapshot document : documents) {
423             Daydata filled = new Daydata();
424             filled.set((int) document.getDouble("count0").doubleValue());
425             filled.setCount1((int) document.getDouble("count1").doubleValue());
426             filled.setCount2((int) document.getDouble("count2").doubleValue());
427             filled.setCount3((int) document.getDouble("count3").doubleValue());
428             filled.setCount4((int) document.getDouble("count4").doubleValue());
429             filled.setCount5((int) document.getDouble("count5").doubleValue());
430             filled.setCount6((int) document.getDouble("count6").doubleValue());
431             filled.setCount((int) document.getDouble("count7").doubleValue());
432             filled.setEndtime(document.getString("endtime"));
433             filled.setSeizurekind(document.getString("seizurekind"));
434             filled.setSeizurename(document.getString("seizurename"));
435             filled.setStarttime(document.getString("starttime"));
436             filled.setTimecount((int) document.getDouble("timecount").doubleValue());
437             filled.setTimegap(document.getString("timegap"));
438             filled.setTimeline0(document.getDouble("timeline0").doubleValue());
439             filled.setTimeline1(document.getDouble("timeline1").doubleValue());
440             filled.setTimeline2(document.getDouble("timeline2").doubleValue());
441             filled.setTimeline3(document.getDouble("timeline3").doubleValue());
442             filled.setTimeline4(document.getDouble("timeline4").doubleValue());
443             filled.setTimeline5(document.getDouble("timeline5").doubleValue());
444             filled.setTimeline6(document.getDouble("timeline6").doubleValue());
445             filled.setTimeline7(document.getDouble("timeline7").doubleValue());
446             results.add(filled);
447         }
448         return results;
449     }

```

In this final snippet, the API using a query, searches the Data collection of the Days collection, to find all the daily data. The document ID and the day ID is known from the previous method (getAllDaysFromCalendar). An Array List is created to store all the daily data of the day that were found on Data collection. Those data are the seizure details (start/end seizure time, seizure type and seizure name) and other useful timeline data (those data are used to place each record on the right timeline row of the daily record view on the Parent app. Then, the final results of all methods are returned to the first method and this method returns the final results to the user. At the end, all the data are visualized on the Calendar Page of the Parent app.

#### 4.2.2.5 *ApiRTC*

ApiRTC is a real time Web communication library. In this project, we used this library to stream video and audio from the Child's app to the Parent's app. To use this communication library, on a mobile application, several Cordova plug-ins are needed. Here is the list of the plug-ins:

- cordova plugin add cordova-plugin-console
- cordova plugin add cordova-custom-config

- cordova plugin add cordova-plugin-device
- cordova plugin add cordova-plugin-iosrtc
- cordova plugin add cordova-plugin-media
- cordova plugin add android-camera-permission
- cordova plugin add cordova-plugin-android-permissions@0.10.0\$
- cordova plugin add <https://github.com/alongubkin/audiotoggle.git>
- cordova plugin add cordova-plugin-audioinput
- cordova plugin add cordova-plugin-crosswalk-webview

Camera-permission plug-ins needed in order to get access of the smartphone's camera. Those are actually very important for both apps because we need to open the smartphone's camera and start recording without pressing any button once an absence seizure occurs. In native camera software of Android and iOS, the auto-recording is not supported. The camera can be activated automatically but the user will have to press the record button to start recording. That is not useful at all for our project because a person under an absence seizure can't make any actions. That's why is very important to give permissions to the camera in advance so the camera can be activated and start recording automatically. To use this web service, a connection with apiRTC communications servers has to be established. An Apikey has to be registered from the apiRTC website. The connection is made using this script:

```

39 //service init
40 InitializeApiRTC() {
41     //apiRTC initialization
42     apiRTC.init({
43         apiKey: "myDemoApiKey",
44         apiCCId : "3",
45         onReady: (e) => {
46             console.log(e);
47             this.sessionReadyHandler(e);
48         }
49     });
50 }

```

The apiCCId represent the users call ID and it must be different for each user. Using this ID calls can be made between different users. Once the apiRTC is initialized, the variable `e` contains all the information of the initialization. Using this information the user can join a session:

```

52 sessionReadyHandler(e) {
53     console.log(e);
54     this.myCallId = apiRTC.session.apiCCId;
55     this.InitializeControls();
56     this.InitializeWebRTCClient();
57     //autoanswer
58     this.webRTCClient.setUserAcceptOnIncomingCallBeforeGetUserMedia(true);
59 }
60 InitializeWebRTCClient() {
61     this.webRTCClient = apiRTC.session.createWebRTCClient({
62         status: "status" //Optional
63     });
64     this.webRTCClient.setAllowMultipleCalls(true);
65 }

```

On the session handler we initialize a WebRTCClient to make and accept calls. WebRTCClient has many methods to use. In the parent app we used a method called setUserAcceptOnIncomingCallBeforeGetUserMedia(value). This method is used to accept a call on the parent app even if the parent's camera isn't ready yet. Another method that is also very useful, is the setAllowMultipleCalls(value). With this method the parent can accept multiple calls from the Child app. To end a call the hangUp(callID) method is used and ends a session.

On the Child's App, the same configuration is made to initialize the ApiRTC but the child has a different call ID from the parent. As long as the parent and the child are on a session a call can be established. This call is made automatically when an absence seizure event is detected on the Child's app. Using this script the call can be made:

```

197 MakeCall(calleeId) {
198     var data = {};
199     var callConfiguration = {
200         mediaTypeForOutgoingCall : 'VIDEO',
201         record : true
202     };
203     var callId = this.webRTCClient.call(calleeId, data, callConfiguration);
204     if (callId != null) {
205         this.incomingCallId = callId;
206         this.showHangup = true;
207     }
208     var customIdToAddOnFilename = "myCustomIdToAddOnFilename";
209     this.webRTCClient.startRecording('VIDEO-ONLY', customIdToAddOnFilename);
210 }

```

The call is configured to record the video of the call so it can be stored on the cloud database. Then using the call(calledNumber, data, callConfiguration) method, a call is made to the parent app using the WebRTCClient. The call is made by calling the MakeCall function on the Child's app, using the parent ID as an argument. This call is made totally automatically as long as the app is active or it's running on the background, without any input from the child.

```

192     this.MakeCall(this.parentID);

```

The call is hanged up once the absence seizure ends calling the HangUp function of the Child's app.

```

212   HangUp() {
213       this.webRTCClient.hangUp(this.incomingCallId);
214       this.webRTCClient.stopRecording();
215   }

```

#### 4.2.2.6 *Google Maps API*

Google Maps were used to display the Child's Location. The map was initialized using Google Maps API and loaded with the child's location at the center of the map. Here is the div tag of the map from the HTML file:

```

11 <ion-content padding:left>
12 <!-- google maps location of child is displayed ath the center -->
13 <div style="height: 100%; width: 100%" #mapCanvas id="map_canvas"></div>

```

On the typescript file, the coordinates are taken from the mapData, to set the center of the map. Zoom in the center of the map is also made by default to have a better street view:

```

41 //map view
42 @ViewChild('mapCanvas') mapElement: ElementRef;
43 //Load mapCanvas
44 ionViewDidLoad() {
45     this.confData.getMap().subscribe((mapData: any) => {
46         let mapEle = this.mapElement.nativeElement;
47
48         let map = new google.maps.Map(mapEle, {
49             center: mapData.find((d: any) => d.center),
50             zoom: 16
51         });

```

The map actually works like a canvas, so we were able to add a marker and info window on the map, using google maps native elements like the Marker and the InfoWindow.

```

53     mapData.forEach((markerData: any) => {
54         let infoWindow = new google.maps.InfoWindow({
55             content: `<h5>${markerData.name}</h5>`
56         });

```

```

58         let marker = new google.maps.Marker({
59             position: markerData,
60             map: map,
61             title: markerData.name
62         });

```

## 4.2.2.7 Activities

### 1. Activity\_login

In the Login Activity the user has to type his credentials on the login form to log in. The users are authenticated by the Firebase Authentication Service, using email and password based authentication.

```
39  onLogin(form: NgForm) {
40    this.submitted = true;
41
42    if (form.valid) {
43      //Login using firebase AUTH provider
44      this.fire.auth.signInWithEmailAndPassword(this.login.username, this.login.password)
45      .then(() => //auth : any
46      {
47        this.userData.login(this.login.username);
48        this.navCtrl.setRoot(TabsPage);
49      })

```

First, a form validation check is made and then the Firebase Authentication checks the user's credentials on the database. If the user logs in to the database with success, the menus and the app permissions are updated. At the end, the user is navigated to the main page of the app.

### 2. Activity\_register

In the Register Activity the user has to type his credentials on the register form to sign in. The users are authenticated by the Firebase Authentication Service, using email and password based authentication as the Login page.

```
28  if (form.valid) {
29    //Login using firebase AUTH
30    this._AUTH.signInWithEmailAndPassword(this.signup.email, this.signup.password)
31    .then(() => //auth : any
32    {
33      this.userData.signup(this.signup.username);
34      this.navCtrl.setRoot(TabsPage);
35    })

```

The progress is the same as the login activity, but this time a new user is inserted to the Firebase. The user sign's in using the credentials from the registration form. After the registration, the user is logged in automatically. The menus and the app permissions are updated. At the end, the user is navigated to the main page of the app.

### 3. Activity\_main

In the Main Activity all the menus and the notifications bars are contained so the user can navigate to all the app pages. The main activity handles in which pages and menus the user can navigate by its current log status. Background processes like the video-stream service and listening to notifications from the Child app are always running in the main activity. For example, the logged in and the logged out menus are different. Using this function the menus

are updated dynamically if the user connection status is changed.

```
279 //enable the right menus
280 enableMenu(loggedIn: boolean) {
281   this.menu.enable(loggedIn, 'loggedInMenu');
282   this.menu.enable(loggedIn, 'call');
283   this.menu.enable(!loggedIn, 'loggedOutMenu');
284 }
```

The bellow commands are used to initialize the video stream service, so the parent can receive video calls from the Child app.

```
120 //Init API for live video calls
121 this.call.initializeApiRTC();
122 //add stream divs on the side menu
123 this.call.AddEventListeners();
```

#### 4. Activity\_notification

In the Notification Activity, the user can receive notifications from the Child app

```
129 // Listen to incoming messages
130 fcm.listenToNotifications().pipe(|
131   tap(msg => {
132     // show a toast
133     const toast = toastCtrl.create({
134       message: 'A Seizure has started',
135       showCloseButton: true,
136       closeButtonText: 'Open Video',
137       position: 'top'
138     });
139     toast.onDidDismiss(() => {
140       //when toast is dismissed user is nagigated to the call modal page
141       //In that page the map is initialized
142       this.nav.push(CallModalPage);
143       //the video is streamed inside a left side menu and it opens automatically
144       this.menu.open('call');
145       console.log('Dismissed toast');
146     });
147     toast.present();
148   }) //end tap event
149 ) //end pipe
150 .subscribe()
```

The above script is handles the notifications. A pop-over toast is created once a notification is received using the Toast Controller of Ionic. If the user taps the close button and dismisses the toast, navigation to the call page is made. The notification activity is actually part of the main activity and its always running in background no matter in which page the user is. That's why a notification can be received in any page.

#### 5. Activity\_settings

In Settings Activity, the user can change the settings of the app. By tapping the account settings the user is redirected to the account settings page. By tapping the terms of service the user is redirected to terms of service page and can check the terms that have already agreed. The user can also select which types of alerts received (sound, vibration, light).



## 6. Activity\_account\_settings

In Account Settings Activity, the user can change the account settings of his profile. Username and password can be changed. Using Ionic Alert Controller pop-over windows are created to make the settings.

```
138     alert.addButton({
139       text: 'Change',
140       handler: (data: any) => {
141         this.afAuth.auth.currentUser
142           .reauthenticateWithCredential(firebase.auth.EmailAuthProvider
143             .credential(this.afAuth.auth.currentUser.email,data.oldpassword ))
144           .then(() => console.log('reauth ok'));
145
146         this.afAuth.auth.currentUser.updatePassword(data.newpassword).then(function() {
147           // Update successful.
148         }).catch(function(error) {
149           // An error happened.
150         });
151       }
152     });
```

In In this script the user can change the current password to a new one. Re-authentication is made on the firebase to store the new password. Then the new password is updated on the database and the user can use it for the next login. Using an Alert Controller the user can log out of the app. On the logout event, first the user is logged out of the database, then the menu is updated and the user is navigated to the login page.

## 7. Activity\_tutorial

In the tutorial activity, the user goes through a sort tutorial (3 slides) of the app. The user can skip the tutorial by tapping the skip button on the top. At the end of the tutorial, navigation to the terms of service page is made. Then the user has to read and accept the terms to continue using the app. If the user accept the terms redirection to the login page is made. In both pages (Tutorial and Terms) a local variable is set if the user has seen the tutorial and has accepted the Terms.

```
31     onAccept(){
32       this.storage.get('hasSeenTutorial')
33         .then((hasSeenTutorial) => {
34           if (hasSeenTutorial) {
35             this.navCtrl.setRoot(TabsPage);
36           } else {
37             this.navCtrl.push(LoginPage).then(() => {
38               this.storage.set('hasSeenTutorial', 'true');
39             })
40           }
41         });
42     }
```

## 4.2.2.8 Fragments

### 1. Fragment\_calendar

In Calendar fragment, there is an Ionic calendar date selector based on monthly view.

```
10     <!-- dateselector-->
11     <ion-calendar class="textcalendar" [(ngModel)]="date"
12                 [options]="optionsRange"
13                 (onChange)="onSelect($event)"
14                 [type]="js-date"
15                 [format]='YYYY-MM-DD'>
16     </ion-calendar>
```

By selecting a date, a request is sent to the database via the RestAPI to fetch the seizure records of this date and display then on the daily timeline. A loader is created using Ionic loader controller to give some time to the RestAPI to fetch the data.

```
105     //When a date is selected give some time to the API to fetch the data
106     onSelect($event) {
107         let loading = this.loadingCtrl.create({
108             content: 'Loading...'
109         });
110         loading.present();
111         setTimeout(() => {
112             this.navCtrl.push(SubcalendarPage, { //push to the daily timeline of the selected date
113                 title : this.date,
114             });
115         }, 0);
116         setTimeout(() => {
117             loading.dismiss();
118         }, 2500); //Loading is dismissed ater 2,5 seconds
119     }
```

While the loader is active the RestAPI is fetching the data array of the selected day. Once the data are fetched the records are displayed on the daily timeline.

```
89         this._dataservice.getDaydataById(this.time, this.userId).subscribe(res=>{
90             console.log(res); // to show the data array with the Log record
91             this.notifications = res; // assign res to the notifications
92         })
```

There are also 2 modal pages in this fragment, one to see the details of a record and another to add a seizure event manually. More details for those, in the Modals section bellow.

### 2. Fragment\_medication

In the Medication fragment, medication reminders are set. Local Notifications are scheduled using the medicine name, details and the reminder time.

```

64     this.localNotifications.schedule({
65         id: 1,
66         title: this.med.medicationName,
67         text: this.med.details,
68         trigger: { at: dateObject } ,
69         launch: true,
70         silent: false,
71         led: 'FF0000',
72         vibrate: true,
73         actions: 'yes',
74     });

```

Settings about the vibration and sound alerts can be made. Also, the notification led color can be changed.

### 3. Fragment\_videos

In Videos fragment, there is a list with all the available recorded video-streams. Once a video is selected from the list, redirection is made to the Modal\_video\_player page to watch the video and the seizure details.

### 4. Fragment\_maps

In Maps fragment, Google maps are displayed with the Child's location at the center. The Maps are initialized using Google Maps API and the Child's Location is sent from the Child's smartphone using the RestAPI.

### 5. Fragment\_live\_video

In Live-Video fragment, live video is streamed to the parent's app from the child's device, during an absence seizure. As we told above, the service is initialized on the main activity and it's always active while the parent is using the app or the app runs in the background.

```

89     <ion-content>
90         <!-- Child stream -->
91         <div class="vid1" *ngIf="showRemoteVideo" id="remote" style="height: 100%;"> </div>
92         <!-- Parent stream -->
93         <div class="vid2" *ngIf="showMyVideo" id="mini"></div>
94         <!-- hang up button -->
95         <button class="vidbtn" ion-button block color="danger" (click)='HangUp()'>Hangup</button>
96     </ion-content>

```

The parent and the child video are streamed in different div's. The parent stream is overlay on the left bottom corner of the child stream. This is how the content of the parent stream is created (divId=mini):

```

147     //the parent div on the left side menu
148     this.webRTCClient.addStreamInDiv(e.detail.stream, e.detail.callType, "mini", 'miniElt-' + e.detail.callId, {
149         width: "30%",
150         height: "30%"
151     }, true);

```

This is how the content of the child stream is created (divId=remote):

```

181 //the child div on the left side menu
182 this.webRTCClient.addStreamInDiv(e.detail.stream, e.detail.callType, "remote", 'remoteElt-' + e.detail.callId, {
183   width: "100%",
184   height: "100%"
185 }, false);

```

#### 4.2.2.9 Modals

##### 1. Modal\_seizure\_details

This is a modal page of the calendar fragment. In this page all the seizure details like type, start/end time and duration are displayed. The data are already fetched on the previous page so we don't need to send a request to the RestAPI again. We just pass the data of the selected record through the NavParams of Ionic. After processing the date data to the desired format using the toString() function all the seizure details are displayed.

```

38 this.titleTitle = this.navParams.get('title1');
39
40 this.starttime = this.navParams.get('starttime');
41 this.endtime = this.navParams.get('endtime');
42 this.seizurename = this.navParams.get('seizurename');
43
44 this.time = this.titleTitle.toString().substring(0,15);
45 this.date = this.titleTitle.toString().substring(8,10);
46 this.month = this.titleTitle.toString().substring(4,7);
47 this.year = this.titleTitle.toString().substring(11,15); //get year

```

##### 2. Modal\_add\_event

This is a modal page of the calendar fragment. In this page the user can add manually a seizure event on the calendar. The user selects the seizure type and the start and end time of the seizure. Then the data are stored to the database using the addNewCalendar method of the RestAPI.

```

285 var day0: Days[] = [
286   {date:"" + this.time, daydata: daydata0}
287 ]
288
289 var calendar0: Calendar[] = [
290   {userId: "" + this.userId, days: day0}
291 ]
292 this._dataservice.addNewCalendar(calendar0);

```

##### 3. Modal\_video\_player

This is a modal page of the videos fragment. In this page there is a video player to play the selected recorded video from the recorded videos list.

```

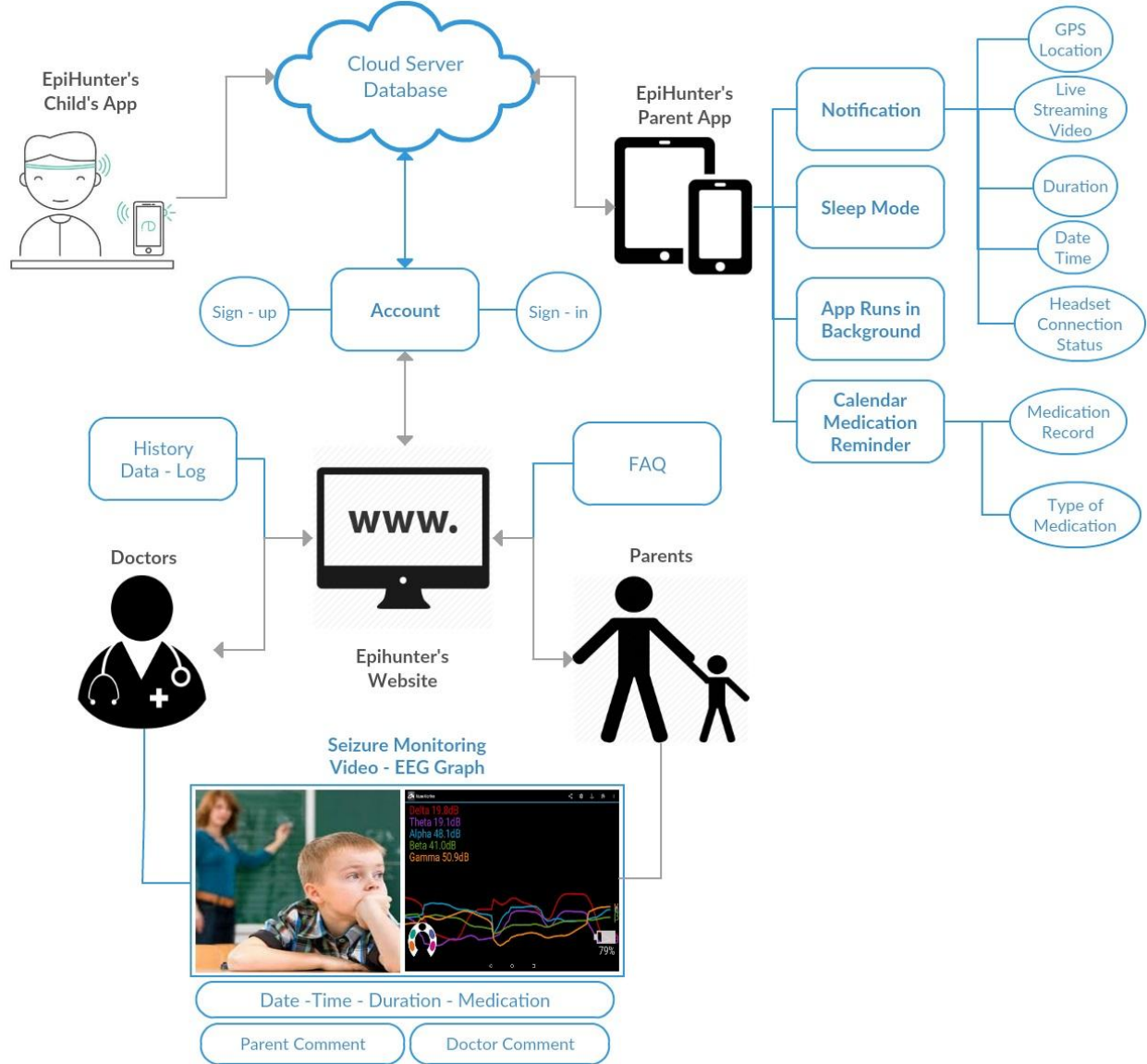
17 <video controls="controls" preload="metadata" webkit-playsinline="webkit-playsinline" class="videoPlayer">
18   <source src="http://www.w3schools.com/html/mov_bbb.mp4" type="video/mp4" />
19 </video>

```

Under the video play, there are the seizure details and comments from the doctors.

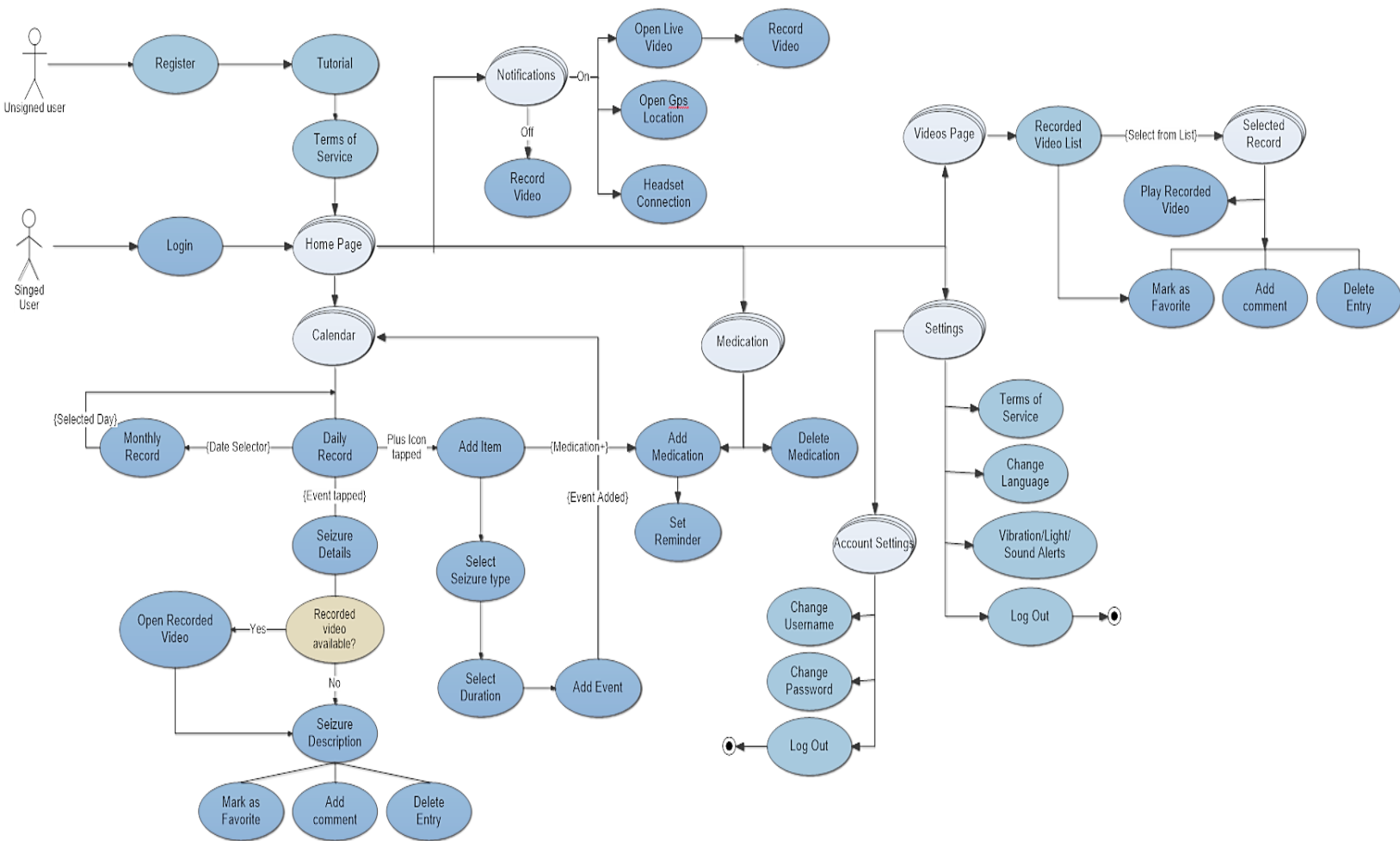
4.2.2.10 App Diagrams

### Project General Diagram



Picture 10, Project General Diagram

# Extended Use Case Diagram



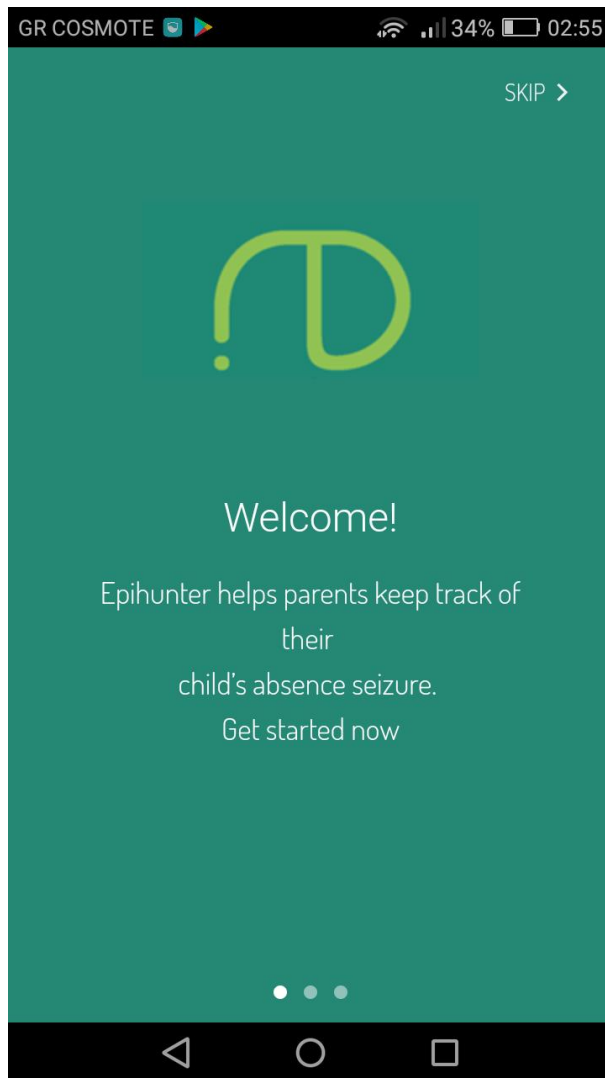
Picture 11, Extended Use Case Diagram

### 4.3 Manual

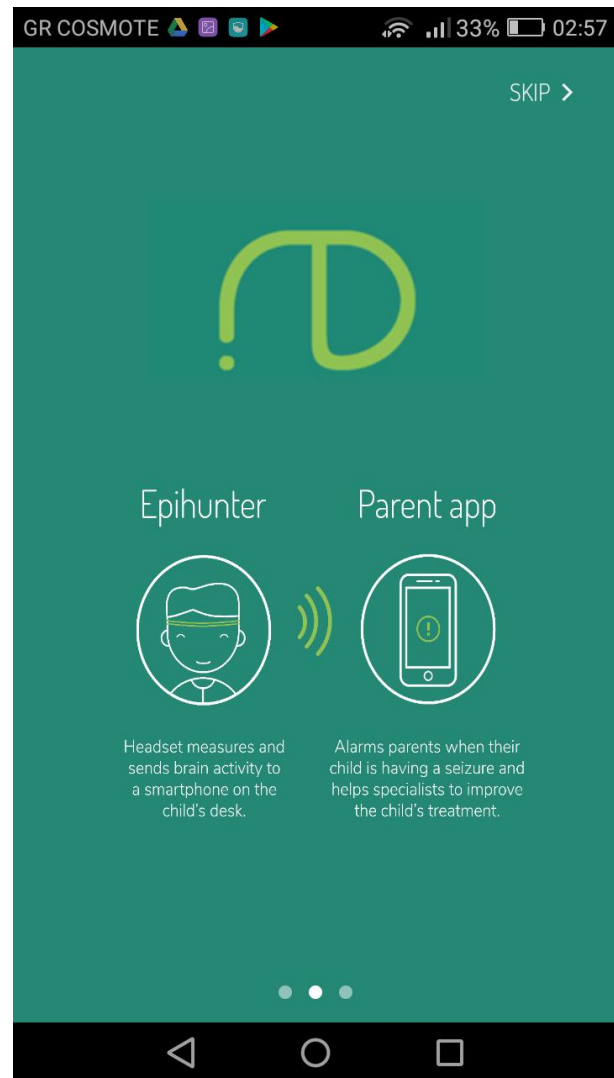
Below is a presentation of Parent app (Epigo) core features.

The first contact of the user with the mobile application in the first installation of the app is through the Tutorial screen. There are 3 slides in the Tutorial screen that inform the user about Epihunter and the app main features.

## Epigo

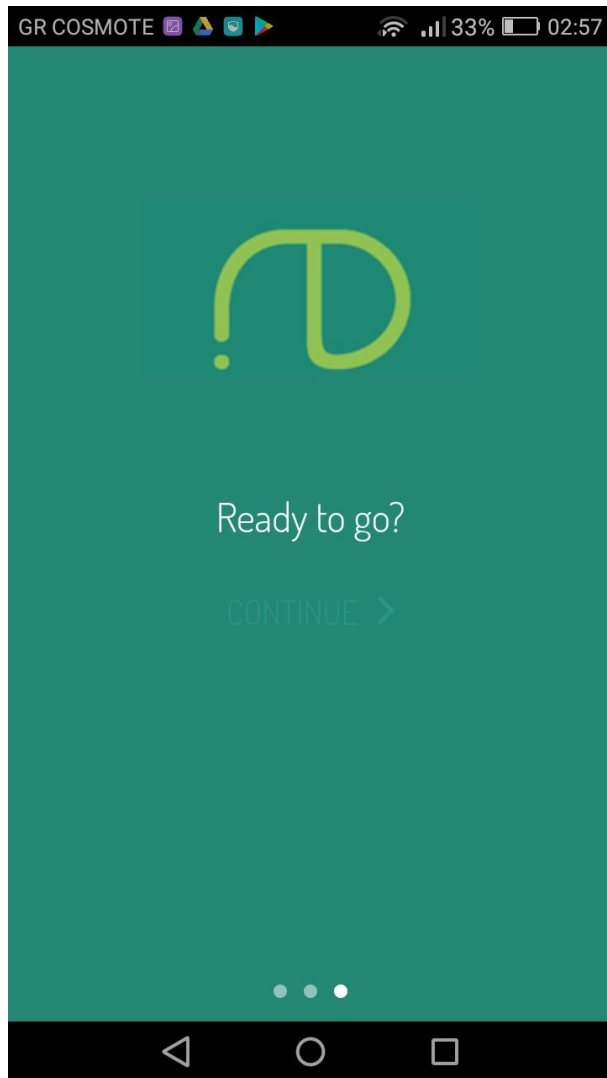


Picture 13, Manual, Tutorial 1



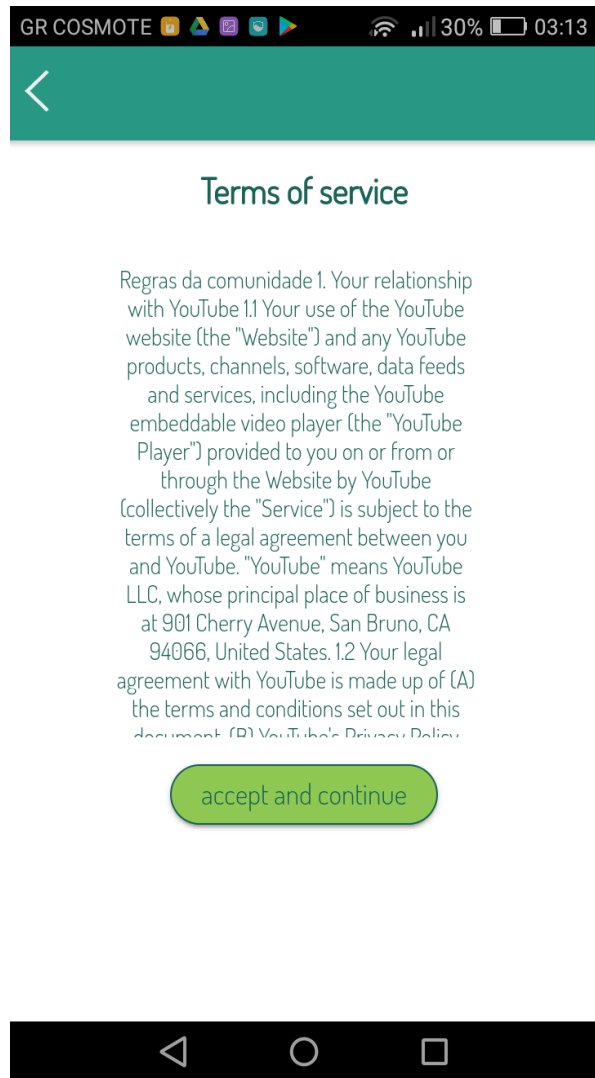
Picture 12, Manual, Tutorial 2

Those 2 slides give a brief description on who the app works and its main functions. By swiping left and right the user can be navigated between the tutorial slides.



Picture 15, Manual, Tutorial 3

This is the last page of the tutorial. By tapping continue the user is navigated to the Terms of Service Page. At this point, a variable is set on the phone local storage that the user has seen the tutorial, so on the next time this user opens the Parent app on his device, he hasn't to go through the tutorial slides again.

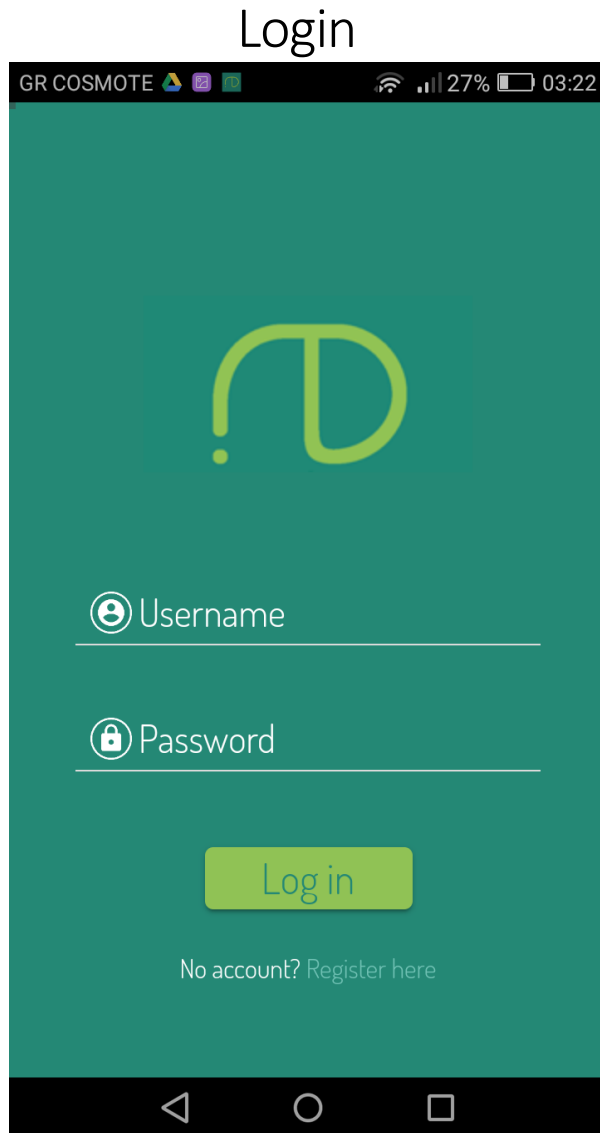


Picture 14, Manual, Terms of Service

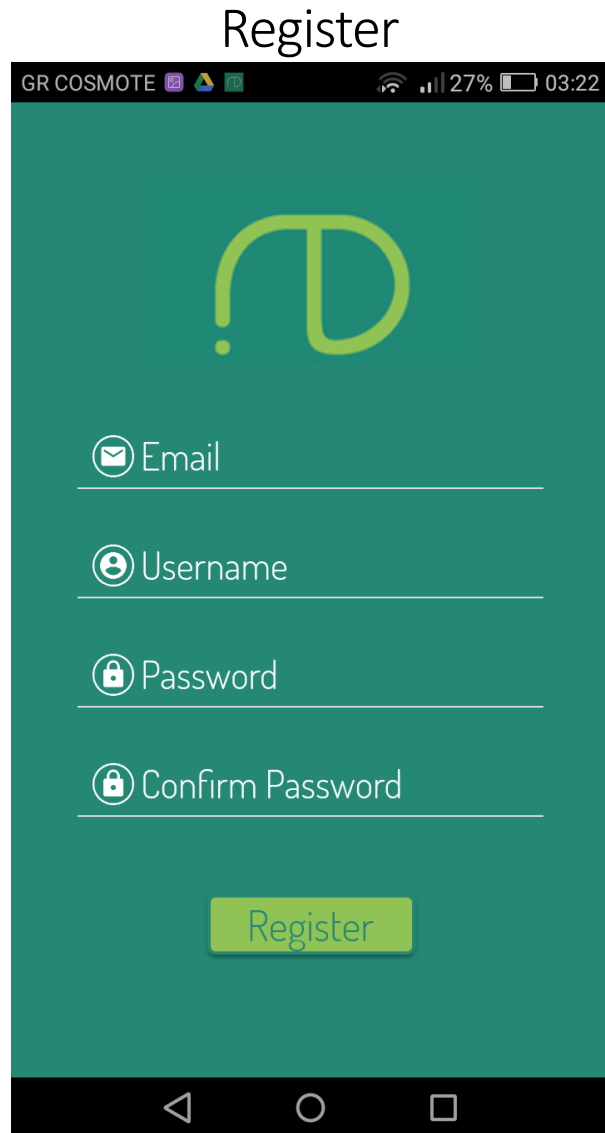
This is the Terms of service page. The user has to accept the terms to continue using the app. A variable is set on the phone local storage that the user accepts the terms, so on the next time this user opens the app hasn't to accept the terms again.

After the user has seen the tutorial and accepted the terms, a redirection is made to the login page.





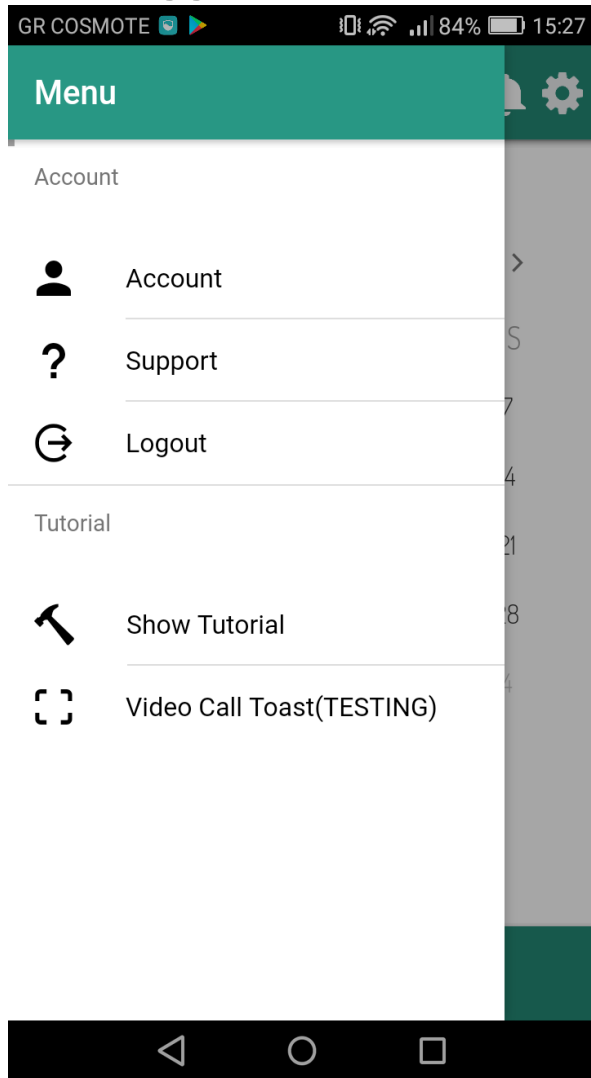
Picture 16, Manual, Login



Picture 17, Manual, Register

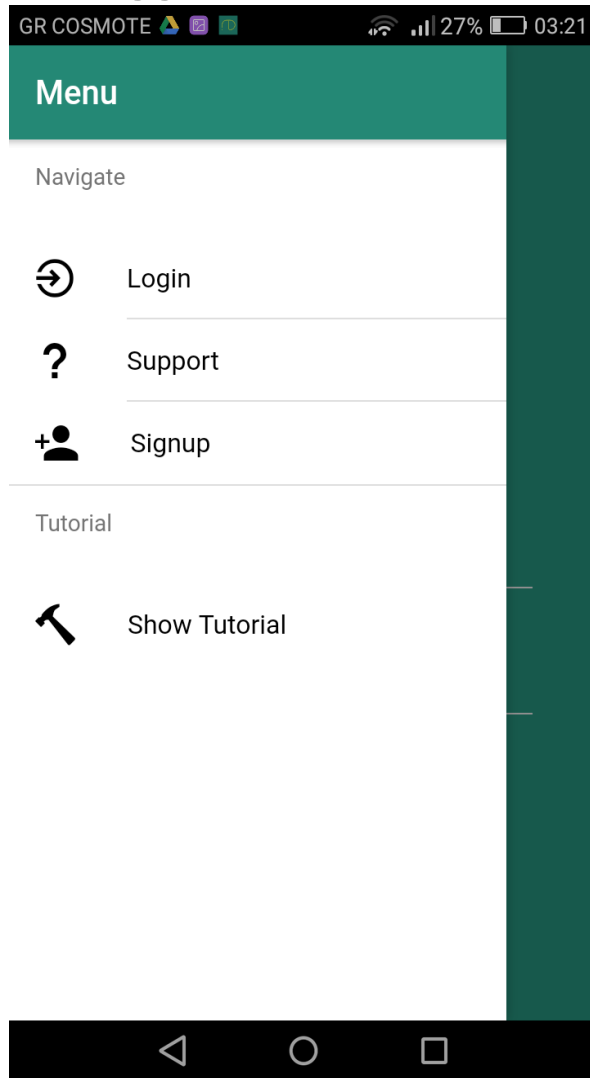
The user can login using his username and password. If the user is not registered, a registration can be made on the Registration page by filling the form. The Authentication to the database is made using Google's Firebase Authentication service. Once the user is logged in a lot of things happen. First of all, the logged in menu is enabled and the user now has access to all the pages of the app. Next, the ApiRTC web client is initialized in order to receive live video-stream from the Child app. Also, after the log in the user is able to receive notifications from the Google Cloud Functions. At the end, the user is navigated to the main page.

## Logged in Menu



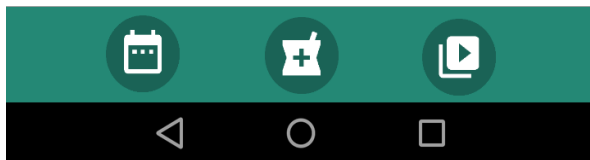
Picture 18, Manual, Logged In Menu

## Logged out Menu

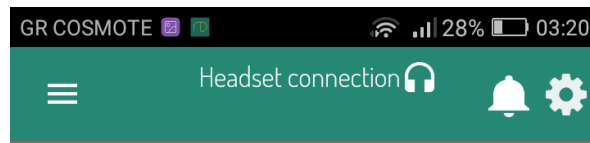


Picture 19, Manual, Logged Out Menu

We can see that the 2 menus give have different permissions to logged in/out users.



Picture 21, Manual, Bottom nav-bar

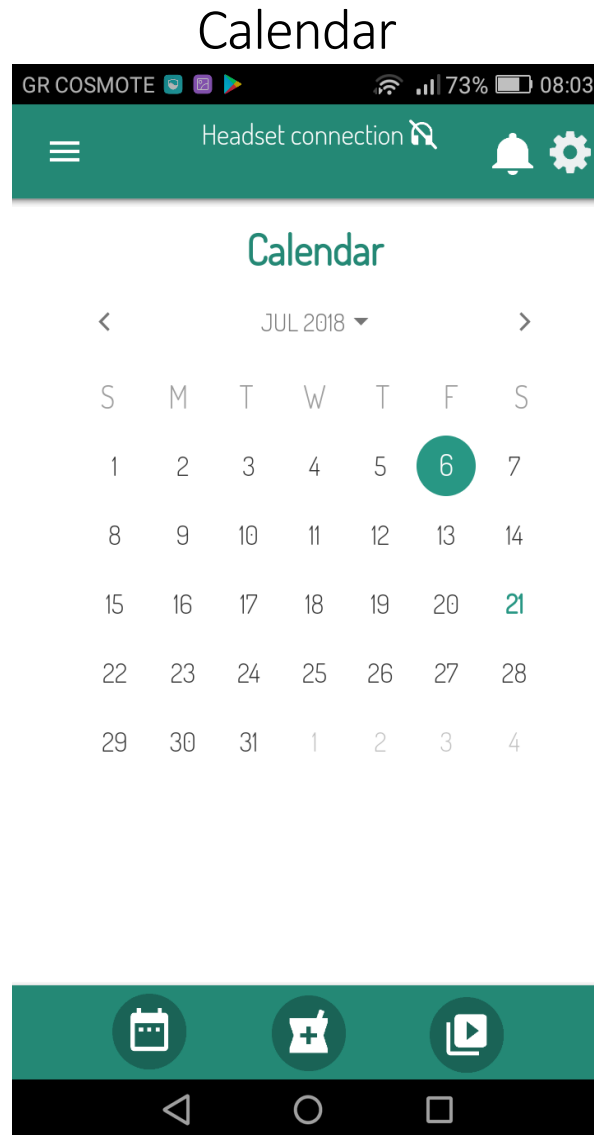


Picture 20, Manual, Top nav-bar

The top and bottom navigations bars are also on top of all the main pages (Calendar, Medication, and Video) to improve the UX.

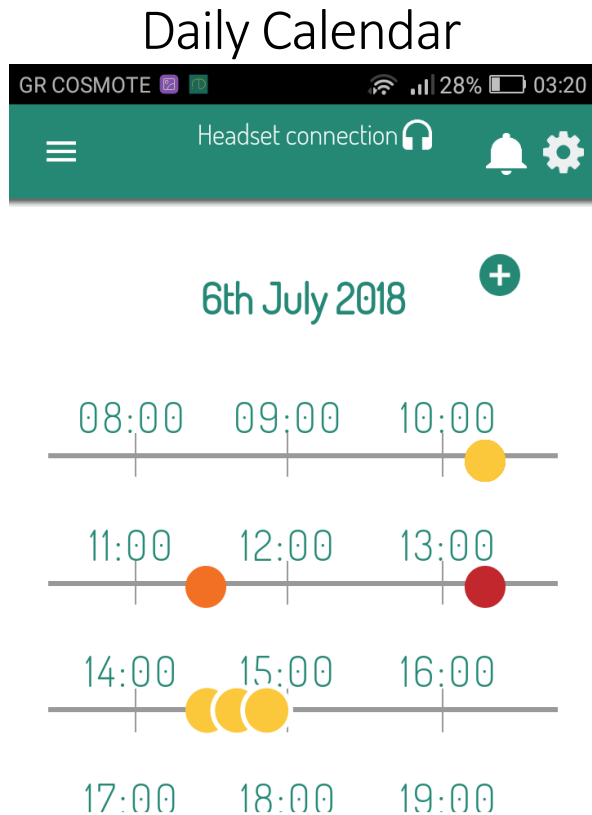
The bottom nav-bar has 3 icons and by tapping them the user is navigated to the main pages.

On the top nav-bar the headset connection is displayed and the user can also turn on or off the notifications by tapping the bell icon.

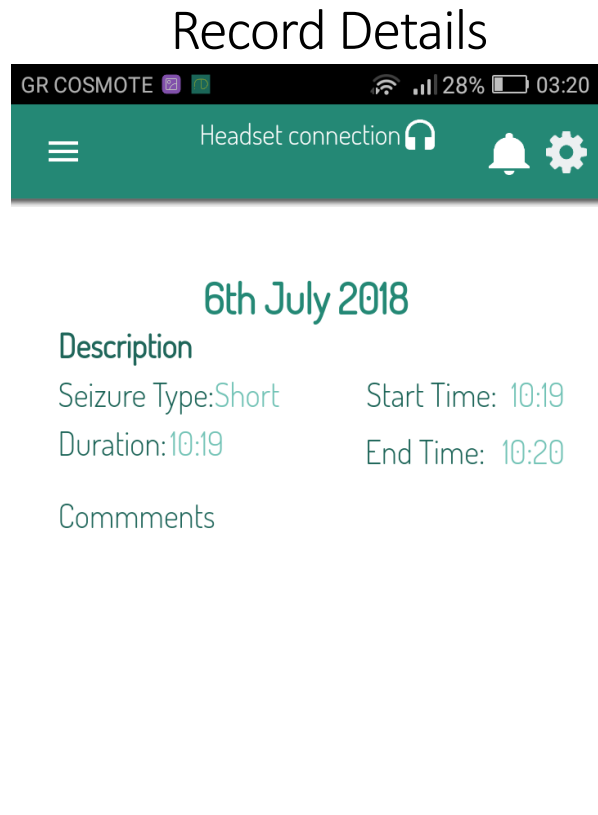


**Picture 22, Manual, Calendar Date-picker**

This is the calendar monthly view. There is a date picker, so the user can select a specific date to review the daily log. Once a date is selected, a request is sent via the RestAPI to the database to fetch the data of the selected date. Then a redirection to the daily calendar view is made and the data are visualized on the daily timeline.



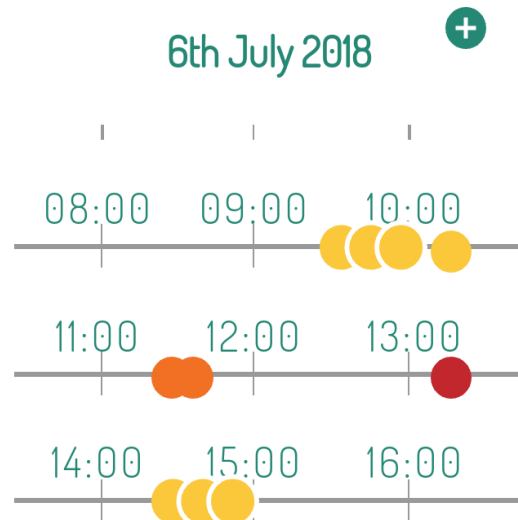
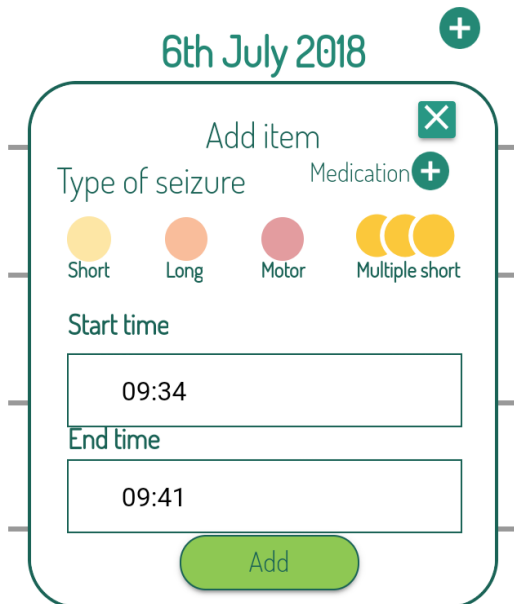
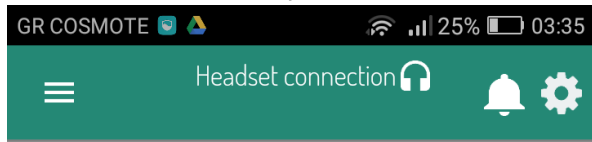
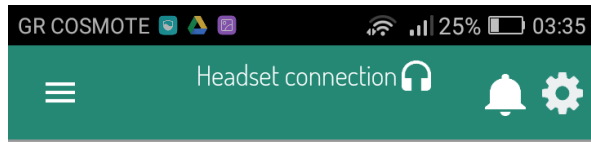
Picture 23, Manual, Calendar Daily Timeline



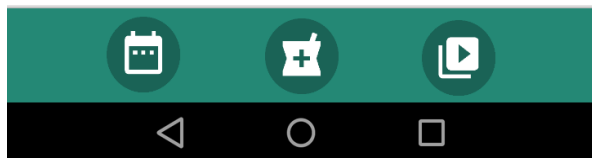
Picture 24, Manual, Seizure Record Details

In this view there are all the records that are fetched from the database. Those records are displayed on a daily timeline using colored dots that represents the different seizure types as we mentioned on the previous sections. By tapping on a record, a request is made via the RestAPI to fetch the details of this record. Once the data are fetched, a redirection is made to the detailed view. In the detailed view, the user can details of the seizure like start and end time, the type of the seizure and the duration. There is also a comment section where the user can add useful input about this seizure event that is helpful for the doctors.

## Add seizure event manually



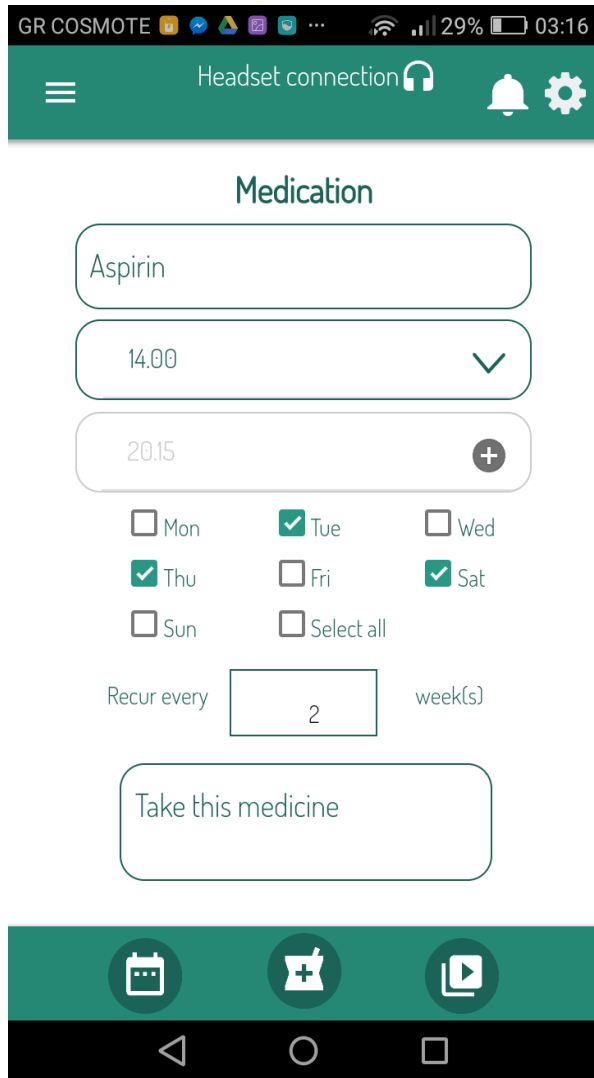
Picture 26, Manual, Insert Record Manually



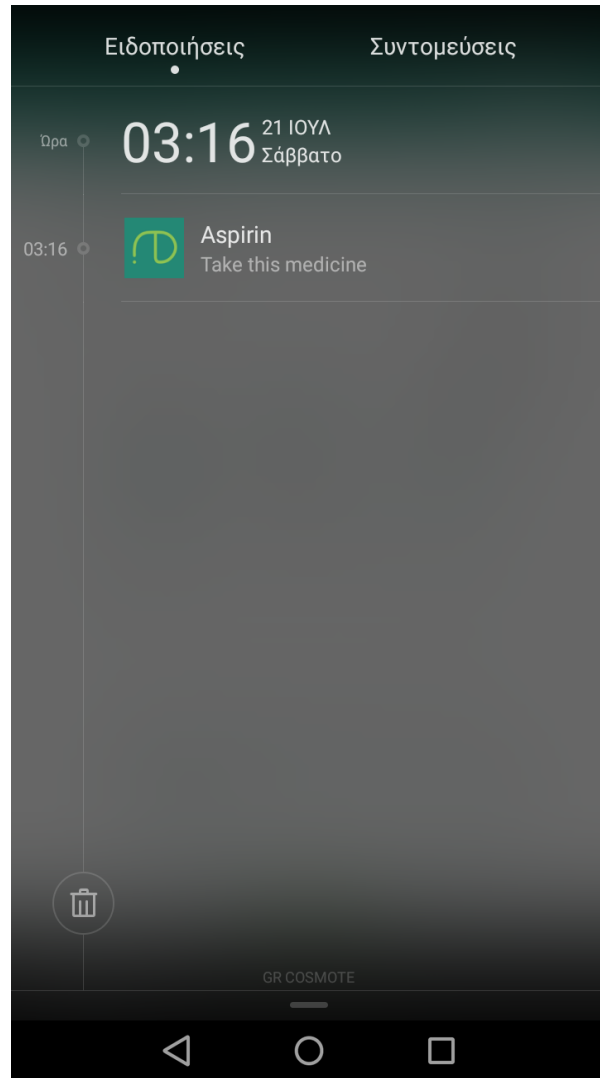
Picture 25, Manual, Record Inserted

On the daily timeline view there is a plus icon. By tapping that icon a pop-over box is presented to the user. This box is used to insert records manually to the database by selecting the seizure type and the start and end time of the seizure. Once the add button is pressed, a request via the RestAPI to the database to store the new record. The next time the user selects this date, the manually inserted record is going to be fetched with the all the other records of this date. If the user doesn't want to insert a record the pop-over box can be dismissed by pressing the close button.

# Medication Reminders



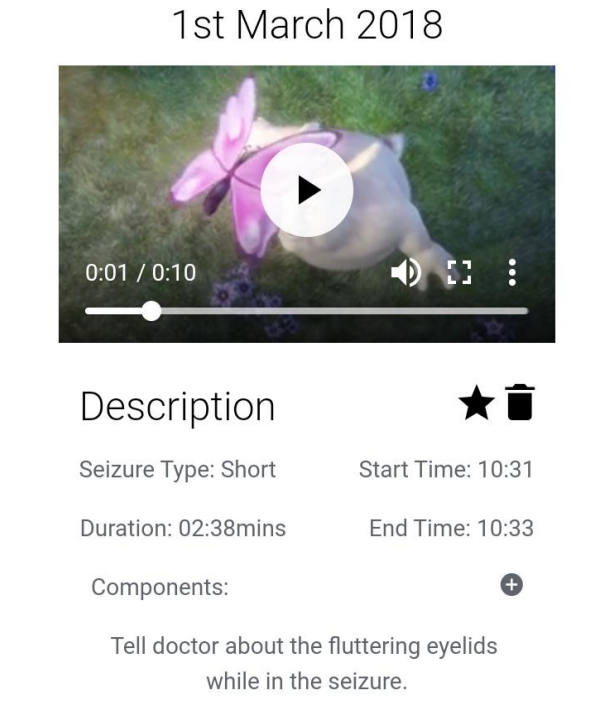
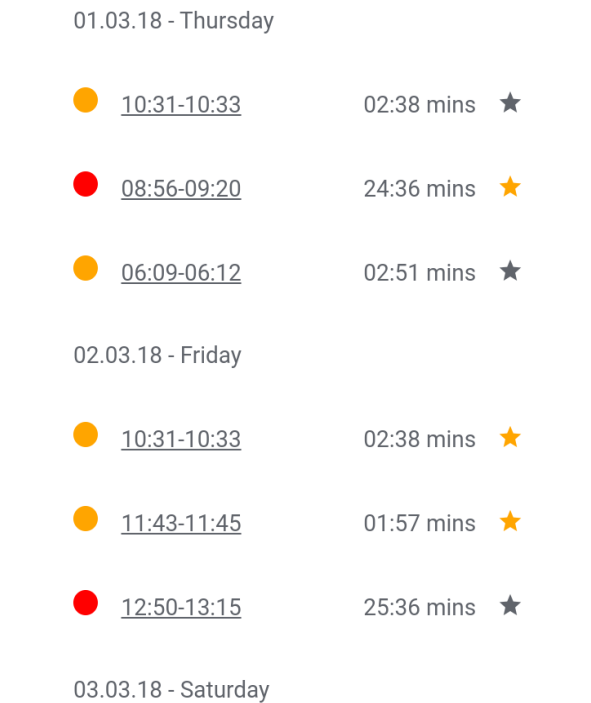
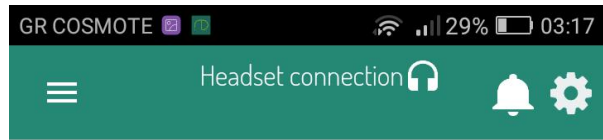
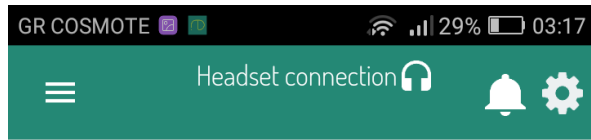
Picture 28, Manual, Medication Reminders



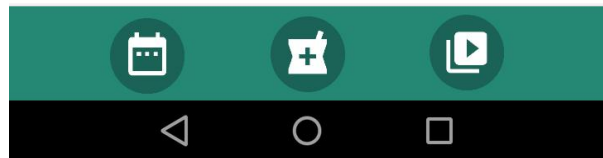
Picture 27, Manual, Medication Notification

This is the medication reminder view. Here medication reminders can be set. The user has to insert the medication name, the time that the medicine has to be taken and select days of the week that the notifications are going to be scheduled. Recurrence of these reminders can be made by selecting the amount of the weeks that this medication has to be taken. There is also a comment section so the user can add useful comments about the medication. Once the reminder is scheduled, a local notification is going to be pushed on the user's device.

# Recorded Videos



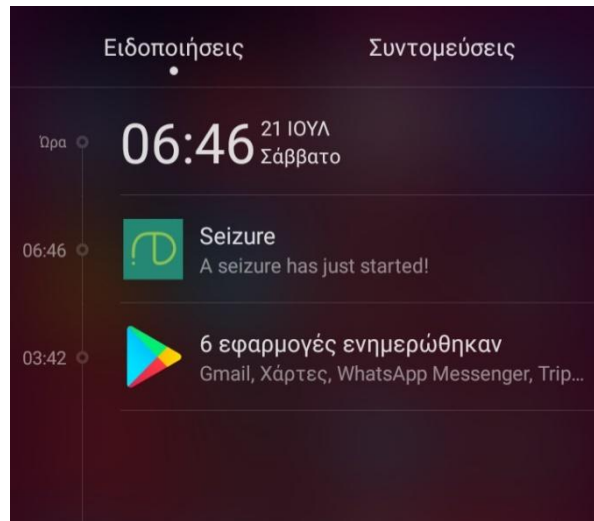
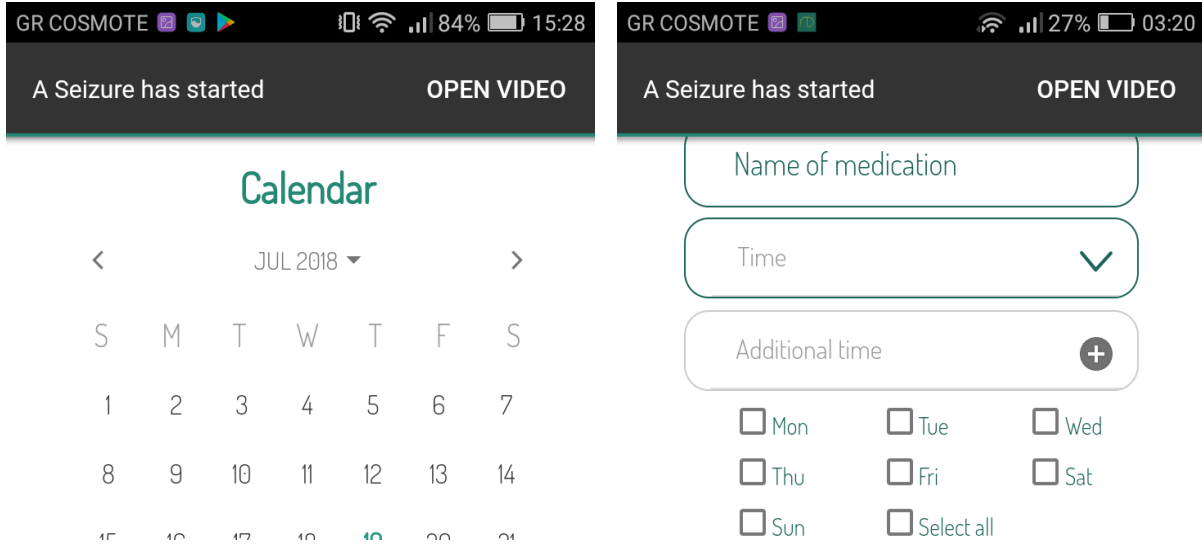
Picture 29, Manual, Recorded streams



Picture 30, Manual, Video Player-Details

This is the recorded videos view. In this view, there is a list with all the available recorded videos from the cloud database. For each record, the seizure type, start and end time, and the seizure duration are displayed. The users can also mark the important videos that are going to be useful to the doctors by tapping the star icon. By selecting a video record, the user is redirected to the video player where the video and the seizure details are displayed. The user can mark or delete the video or add a comment for the doctor.

# Seizure Notifications

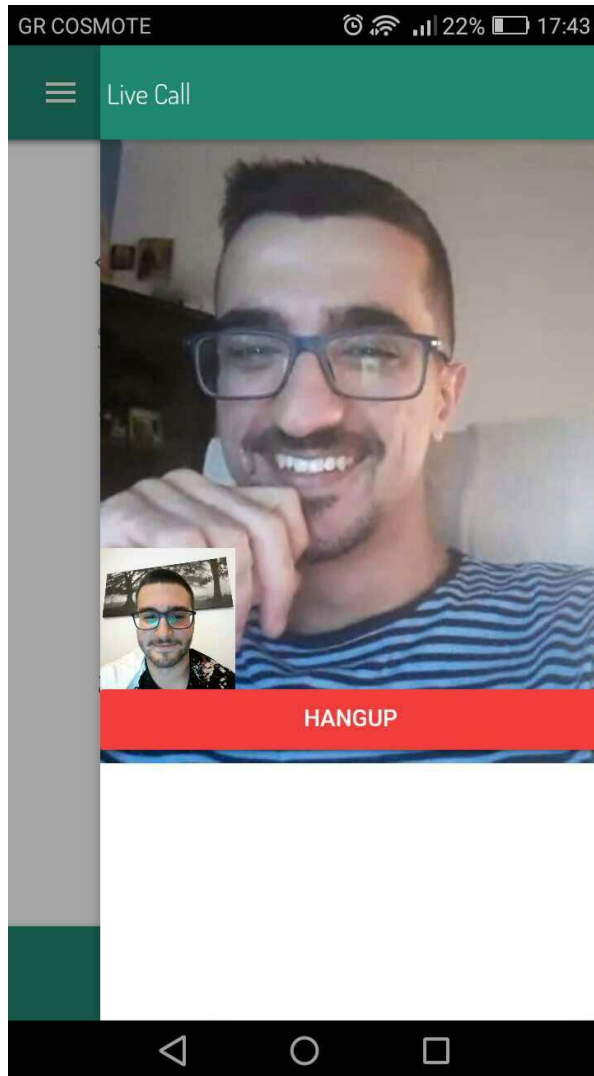


Picture 31, Manual, Seizure Notifications

Once a seizure is detected on the Child's app, a notification is pushed to the parent app. This notification is received in all views of the app and even when the user doesn't use the app the notification is still received on the user's device because the app is always running on the background in order to receive the notifications. If the user opens this notification, a redirection is made to the live video-stream and the map location view.



## Live video-stream



Picture 33, Manual, Live video-stream

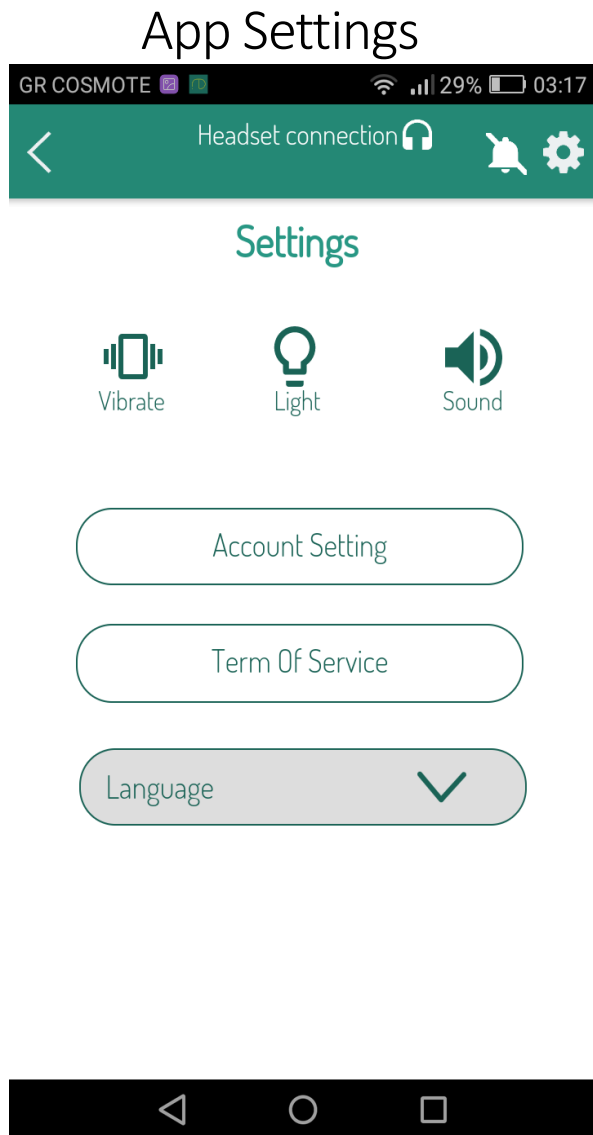
Once the notification is opened, live video-stream of the child is displayed to the parent's device. The real-time video is displayed as a left-side menu that is opened automatically once live video from the child app is streamed. By dismissing the left-side menu the video is hidden and the user has a large map view with the child's location.

## Map View



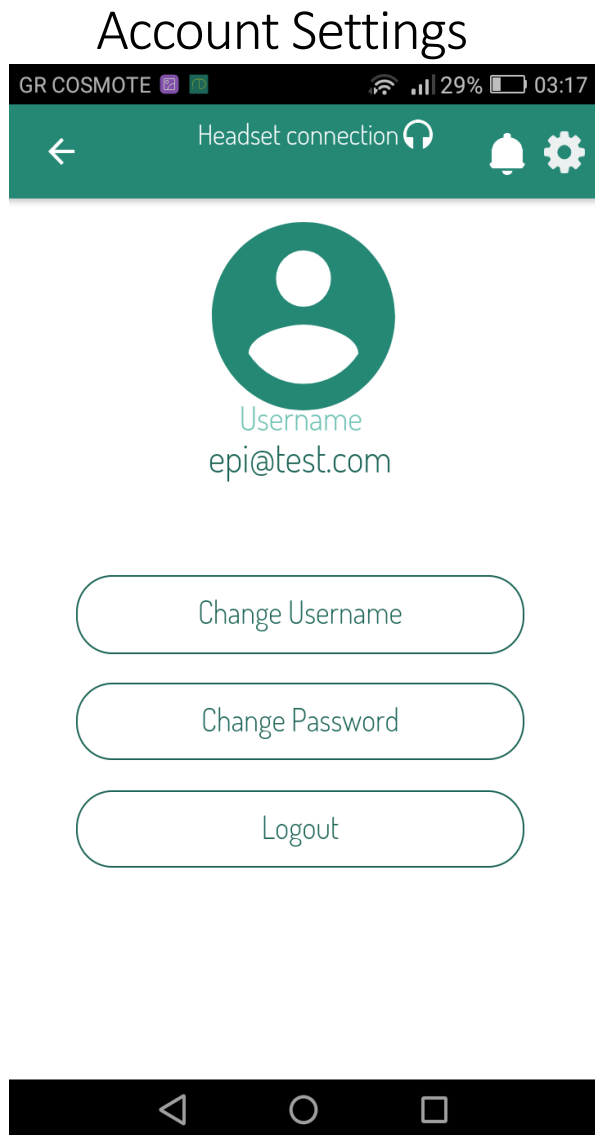
Picture 32, Manual, Map Location

The child's location is displayed on a Google Maps view. The user can zoom in and out on the map or even open the location on the Google Maps application. By tapping the camera icon or by swiping right, the user redirects to the live video-stream view. The user can also hang up the call by pressing the hang up button. A record of the stream is stored on the cloud database in order to be reviewed by the doctors and help with the treatment.



Picture 34, Manual, Application Settings

On the app settings, the user can change the application setting like the alerts (light, vibration, sound) and the language of the app. There is also a button that redirects the user to the account settings page and another that redirect the user to the terms of service page.



Picture 35, Manual, Account Settings

On the account settings, the user can change the account setting like username and password. Users can logout by pressing the logout button. Then a request is sent to the database to log-out the user of the database.

# Doctors Website

This website is dedicated to doctors in order to help them give the right treatment to their patients that suffer from epilepsy. With this website, the doctors can see all their patients' data. Data like seizure records, medications and the stored video-streams of the seizure sessions and help their patients with the treatment. Those data are fetched from the database via the RestAPI by making request.

## Patients List

FIRST NAME	LAST NAME	GENDER	DATE OF BIRTH	AMOUNT OF SESSIONS	LAST SESSION	ADD	SEARCH
Michel	Wouters	Female	10/10/2008	25	01/01/18		
Rosalie	DeMeer	Female	23/04/2012	16	30/12/17		

Picture 36, Manual, Doctor Patients List

On the main page of the website there is a list with all the patients that are associated with this doctor. There are some basic information for each patient like first and last name, gender, date of birth, the amount of session that this patient had in total and the date of the last session. The doctor can insert or delete patients from the list or search for a specific patient. By selecting a patient from the list a redirection is made to the patient's details page. There is also a top-navigation bar to help with the navigation of the website. There the user can access his profile or log out. There is also a contact and a FAQ section.

# Patient Details

**Calendar View:**

December 2017							January 2018						
S	M	T	W	T	F	S	S	M	T	W	T	F	S
			1	2	3				1	2	3		
4	5	6	7	8	9	10	4	5	6	7	8	9	10
11	12	13	14	15	16	17	11	12	13	14	15	16	17
18	19	20	21	22	23	24	18	19	20	21	22	23	24
25	26	27	28	29	30	31	25	26	27	28	29	30	31

**Patient Profile:**

- Full name: Michel Wouters
- Date of birth: 10/10/2008
- Gender: Female
- Parent(s) / Guardian(s): Marijke De Koninck, Peter Walschaerts
- City of birth: Leuven
- Amount of sessions: 25
- Average of seizures per session: 4

**Seizure Sessions Table:**

DATE	TIME	DURATION	AMOUNT OF SEIZURES	VIDEO & GRAPH	ADD	SEARCH
- 01/01/2018	4:45:23 AM	00:30:15	3			
01/01/2018	4:45:23 AM	00:02:38	☆			
01/01/2018	5:06:44 AM	00:24:56	★			
01/01/2018	5:56:31 AM	00:02:41	★			
- 31/12/2017	10:14:01 PM	00:38:14	4			
01/01/2018	10:14:01 PM	00:06:58	☆			
01/01/2018	10:28:44 PM	00:17:02	★			
01/01/2018	11:45:26 PM	00:10:40	☆			
01/01/2018	11:56:31 PM	00:03:34	☆			
+ 31/12/2017	09:30:56 PM	00:06:53	1			
+ 31/12/2017	12:56:43 PM	00:16:39	3			

**Navigation:** << < 1 2 3 4 > >>

Picture 37, Manual, Doctor Patient Details

In this detailed page, all seizure sessions of the patient are displayed on a monthly calendar view and also on a list view. On the list-view, the doctor can see all the dates that this patient had seizure sessions and select the date that he wants to examine. By selecting a date, the list is extended with the records of that date. Then by selecting a record a redirection is made to the Seizure Session Details. There is a star icon for each session and the important sessions are marked by the parents via the Parent App, in order to save time for the doctors so they don't have to go through all the videos. The doctors are able to delete and edit records of the patient. There is also a section that shows to the doctor if recorded video and EEG graph are available for this session.

# Session Details

D epihunter

**Jana Peeters**

**Datalog**

**Contact**

**FAQ**

Datalog
Michel W.
01/01/18

**SESSION**

DATE	TIME	DURATION	AMOUNT OF SEIZURES	SEARCH	
–	01/01/2018	4:45:23 AM	00:30:15	3	
●	01/01/2018	4:45:23 AM	00:02:38	☆	
●	01/01/2018	5:06:44 AM	00:24:56	★	
●	01/01/2018	5:56:31 AM	00:02:41	★	

**GRAPH & VIDEO**

**EEG**

**Video**

**OTHER SESSIONS**

See the first
10
25
50
100

DATE	TIME	DURATION	AMOUNT OF SEIZURES	VIDEO & GRAPH	ADD	SEARCH	
DATE	TIME	DURATION	AMOUNT OF SEIZURES		+		
+	01/01/2018	4:45:23 AM	00:30:07	3			
+	31/12/2017	10:14:01 PM	00:38:14	4			
+	31/12/2017	09:30:56 PM	00:06:53	1			
+	31/12/2017	12:56:43 PM	00:16:39	3			

<<
<
1
2
3
4
>
>>

**COMMENTS**

●
Marijke De Koninck
01/01/18 09:34 AM

We've noticed that she flutters her eyes quiet a lot right before an absent seizure.

●
01/01/18 11:56 AM
Dctr Haesen

Thats great of you to tell me, I'll defenitly look into it.

Sort by
>
DATE

**Picture 38, Manual, Doctor EEG and Video**

Once a session is selected, the EEG graph and the recorded video are displayed to the doctor. The EEG graph shows the brain activity of the patient during a seizure and next to it there is the recorded video of the patient that shows the reactions of the patient during this session. The video and the EEG graph are synchronized by using the same timestamp. The doctor can add comments on that session and read the comments that have been made by the parents. Doctors are also able to select a different session from this date or even select another date with sessions to examine.

## CHAPTER 5: RESULTS

### *5.1 Conclusion*

This thesis was attained within the Blended Aim project. Through Blended Aim, students across Europe were assigned with the development of 2 products for 2 different start-ups. My team's product was appointed by Epihunter, a Belgian start-up company specialized in absence seizure detection using EEG headsets. The outcome of the project was a product named Epigo which consists of two Android applications, a website and the backend service.

Epigo was designed to help the parents of children with epilepsy. Using this application parents can be aware once their child is having an absence seizure. Absence seizures are really hard to be noticed, that's why Epihunter developed the Epihunter classroom. Using an EEG headset and a smartphone, this app can detect an absence seizure event and notify the child's teacher, activating smartphone's flashlight. Our main goal in this project was to notify parents once an absence seizure is detected on the Epihunter Classroom app. That why we created Epigo, a mobile application for the parents. The main features of the app was to receive notifications from the Epihunter Classroom app once an absence seizure is detected, store the seizure records on a database and receive live video-stream of the child during the seizure. Epigo, is an application that can help parents and children on their daily live, but it can also help doctors with the child's treatment.

The opportunity to work and collaborate with students from different educational institutes and fields of study was a great experience and practice. The students shared their thoughts, worries, issues, solutions, knowledge and different points of view on a subject as one team. This served as a learning ground for the students who were not familiar with each other's field of study, thus providing a better and more complete understanding of their areas of expertise. It also improved the perception of how to approach a problem more efficiently. Though it might seem imaginary, basic intercommunication and efficient work flow is sometimes difficult for people of different countries because of contrasting habits and work procedures. Communication and better understanding was developed further during the project implementation.

By developing this project the students acquired more knowledge and experience. Through the usage of new technologies, which are considered state of the art in the field of informatics engineering, they became familiar and more comfortable using such technologies. Technologies like AngularJS, REST services and Ionic Framework keep rising by the day in the cross-platform mobile software development section. The combination of the above – mentioned with Android Application, iOS and Windows Phone development is a significant foundation for cross-platform development and a successful career.

## 5.2 Future work and extensions

Upcoming Features	
Doctors website	Develop the doctor's website to help with the treatment. Main functions: Patients list, absence seizure records, filter seizure records, display seizure data (video, EEG Graph) and other useful information, like comments from the parent about their child behavior during a seizure.
Store videos streams	Store the video streams on a secure cloud database so those video can be accessible from doctors. The parent of course will have the option to delete a stored video using the parent app.
Record video stream even when parent is offline	In the current implementation of the video stream function, both users (Parent and Child) must be online in order to establish a call between them. During the call the video is recorded. On future development, the video of the child has to be recorded even if the parent isn't online.
Improve medication reminders	Let the parents mark a checkbox, if the child has taken the medication and store the medication record on the database. This is going to be useful for the doctors to adjust the medication.
Translate Application in more languages	Translate the app and the website, in other languages.
Security	Insert security in the app and databases. This is very important as long as we speak for sensitive medical data from children.
Testing the app in real word	Test the app with parents and children. This is very important for the future development of the app. The company is going to get useful input from the parents about the app features. Then using this feedback extra functions can be developed on the app and the current function can also improve on the needs of the parents.
Improve UX/ UI	Improve UX and UI of the parent app (parents feedback)

Future work

# REFERENCES

- [1] Epihunter's Official Website  
<<https://epihunter.com/?lang=en>>
- [2] BAIM Official Website  
<<http://blendedmobility.com/en/about>>
- [3] Nexus Framework  
<<https://www.agilest.org/scaled-agile/nexus-framework/>>
- [4] ApiRTC Library  
<<https://apirtc.com/wp-content/themes/wikee/docs/apiRTC/index.html>>
- [5] Wikipedia-Node.js  
<<https://en.wikipedia.org/wiki/Node.js>>
- [6] Wikipedia-npm  
<[https://en.wikipedia.org/wiki/Npm\\_\(software\)](https://en.wikipedia.org/wiki/Npm_(software))>
- [7] Wikipedia-AngularJS  
<<https://en.wikipedia.org/wiki/AngularJS>>
- [8] Wikipedia-iOS  
<<https://en.wikipedia.org/wiki/IOS>>
- [9] Wikipedia-Android  
<<https://en.wikipedia.org/wiki/Android>>
- [10] Wikipedia-Ionic Framework  
<[https://en.wikipedia.org/wiki/Ionic\\_\(mobile\\_app\\_framework\)](https://en.wikipedia.org/wiki/Ionic_(mobile_app_framework))>
- [11] Wikipedia-Apache Cordova  
<[https://en.wikipedia.org/wiki/Apache\\_Cordova](https://en.wikipedia.org/wiki/Apache_Cordova)>
- [12] Wikipedia-Firebase  
<<https://en.wikipedia.org/wiki/Firebase>>
- [13] Firestore  
<<https://firebase.google.com/docs/firestore/>>
- [14] Google Cloud Functions  
<<https://cloud.google.com/functions/>>



- [15] Wikipedia-Google Maps  
<[https://en.wikipedia.org/wiki/Google\\_Maps](https://en.wikipedia.org/wiki/Google_Maps)>
- [16] Wikipedia-Use case diagram  
<[https://en.wikipedia.org/wiki/Use\\_case\\_diagram](https://en.wikipedia.org/wiki/Use_case_diagram)>
- [17] Wikipedia-Spring Framework  
<[https://en.wikipedia.org/wiki/Spring\\_Framework](https://en.wikipedia.org/wiki/Spring_Framework)>
- [18] Wikipedia-Apache Maven  
<[https://en.wikipedia.org/wiki/Apache\\_Maven](https://en.wikipedia.org/wiki/Apache_Maven)>
- [19] Wikipedia-Material Design  
<[https://en.wikipedia.org/wiki/Material\\_Design](https://en.wikipedia.org/wiki/Material_Design)>
- [20] Google's firebase platform  
<<https://firebase.google.com/>>
- [21] Google's firebase functions  
<<https://firebase.google.com/products/functions/>>
- [22] Google's firebase firestore  
<<https://firebase.google.com/products/firestore/>>
- [23] Google's firebase authentication  
<<https://firebase.google.com/products/auth/>>
- [24] Google's firebase notifications  
<<https://firebase.google.com/docs/functions/use-cases>>
- [25] Google's firebase database events  
<<https://firebase.google.com/docs/functions/database-events>>
- [26] Google's firebase authentication documentation  
<<https://firebase.google.com/docs/auth/>>
- [27] Google's firebase cloud firestore documentation  
<<https://firebase.google.com/docs/firestore/>>
- [28] Oracle Java  
<<https://www.oracle.com/java/index.html>>
- [29] Scrum  
<<https://www.scrum.org/>>
- [30] Praxis  
<<http://www.praxisnetwork.eu/>>

- [31] Wikipedia-Bitbucket  
<<https://en.wikipedia.org/wiki/Bitbucket>>
- [32] Wikipedia-Trello  
<<https://en.wikipedia.org/wiki/Trello>>
- [33] Wikipedia-Slack (software)  
<[https://en.wikipedia.org/wiki/Slack\\_\(software\)](https://en.wikipedia.org/wiki/Slack_(software))>
- [34] Java API for RESTful Web Services  
<[https://en.wikipedia.org/wiki/Java\\_API\\_for\\_RESTful\\_Web\\_Services](https://en.wikipedia.org/wiki/Java_API_for_RESTful_Web_Services)>
- [35] RESTful Web Service  
<<https://spring.io/guides/gs/rest-service/>>
- [36] Apache Maven  
<[https://en.wikipedia.org/wiki/Apache\\_Maven](https://en.wikipedia.org/wiki/Apache_Maven)>
- [37] Design Patterns  
<<https://www.oodeesign.com/>>
- [38] BAIM 2017 TEI of Crete student project.  
<<https://apothesis.lib.teicrete.gr/handle/11713/8377>>