Expertise
and insight
for the future

Ahmed Fahady

# Development of an IoT-based Student Attendance System

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

27 October 2020

## PREFACE

Being in Finland as a new comer from a third world country where everything there is delayed, made me amazed about Finland when I saw almost everything is computerized and automated. However, this vision didn't stay a long until I encountered with the way how students' attendance is taken. I got the passion to do something for this issue certainly, but the challenge was to know where to start from. With days and when I started to for my master in the program of engineering information technology. There, I met the head of the master program (Ville Jääskeläinen) who supported my idea and opened all doors since the first day I met him to accomplish this project.

Some challenges were also faced during this study. Perhaps the biggest challenge was the study and work at the same time and COVID-19 period when I had to postpone the testing of the project. However, everything was settled at the end. Now I'm in a phase of submitting this thesis and go forward in my life and have more challenges.

In truth, I could not have achieved my current level of success without a strong support group. First of all, my parents, who supported me with love and understanding. And secondly, my peer group who has provided patient advice and guidance throughout the research process. And thirdly, my better half who supported me with everything possible emotionally and mentally. Thank you all for your unwavering support.

Helsinki 31st August 2020
Ahmed Fahady

| | |
|---|---|
| Author<br>Title<br>Number of Pages<br>Date | Ahmed Fahady<br>Development of IoT-based Student Attendance System<br>28 pages<br>15 Sep 2020 |
| Degree | Master of Engineering |
| Degree Programme | Information Technology |
| Instructor(s) | Ville Jääskeläinen, The head of Master of Eng. IT program in Metropolia University of Applied Science. |

This thesis was carried out to study the ability to create students' attendance system by using the available technologies in the frontend, backend, and IoT technologies to substitute the current way of manually taking the attendance in Metropolia University of Applied Sciences.

A comprehensive analysis of the topic was conducted. Such technologies, as React JS, Firebase, Linux and NoSQL databases were examined in detail. Their pros and cons were identified and compared to other technologies that are used in cloud-based development. After a long research, a working IoT-based application prototype for student's attendance was developed. The prototype consists of the IoT kit component which is put in the classroom to get the attendance using a raspberry pi, RFID reader, and 7" touchable screen. user-friendly interface, a possibility to easily check in and out without contact with the IoT kit, monitor the attendance by the lecturer and access real-time information anywhere via the Internet. The prototype also includes an interface for the administrator of the system to manage databases, storage, hosting, and authentication of the system.

The application was built based on the studied technologies and uses the IoT, cloud computing, big data as well single page application frame work (React JS) as an infrastructure to have a better control and flexibility. The project also suggested a future development after it was used and started to collect data. Machine learning analysis could be used to study the behavior of students' attendance individually.

The thesis concludes with the thoughts about the technologies used in the development and the future plan to integrate the system with the organization system to document the attendance automatically in students page.

| | |
|---|---|
| Keywords | student attendance system, Internet of Things (IoT), cloud computing, machine learning, JavaScript, Firebase, Raspberry Pi, single page application, Near Field Communication (NFC), Software engineering. |

**Contents**

## List of Figures

## List of Abbreviations

| | |
|---|---|
| NFC | Near-field communication |
| RFID | Radio-frequency identification |
| GDPR | General Data Protection Regulation |
| IoT | Internet of Things |
| Metropolia | Metropolia University of Applied Science |
| HTML | Hyper Text Markup Language |
| CSS | Cascading Style Sheets |
| SaaS | Software-as-a-Service |
| PaaS | Platform-as-a-Service |
| FaaS | Functions-as-a-Service |
| IaaS | Infrastructure-as-a-Service |
| UI | User Interface |
| UX | User Experience |
| iOS | iPhone Operating System |
| AWS | Amazon Web Services |
| GPU | Graphics Processing Unit |
| RFID | Radio Frequency Identification |
| NPM | Node Package Manager |
| CLI | Command-line interface |
| API | Application Programming Interface |
| SDK | Software Development Kit |

# 1 Introduction

While we are in 2020 and almost everything has become automated, still the student attendance systems are as they were 100 years ago. Currently, Metropolia marks the attendance on paper and later it's moved to the system. Through the last few years, the cloud computing technology have developed a lot, which makes it possible to create an electrical system to track and mark all students' attendance easily and with low cost.

The current way of getting the attendance is time-consuming, as the attendance is part of the lecture time and students need to do it while the time is supposed to be for learning. On the other hand, it's time-consuming for teachers/lecturers to transfer the attendance from paper to their system. Papers have only one information that the student attended today, without more detailed information about whether the student came late or left early.

This thesis tried to find an efficient IoT based solution to automate the attendance taking process. Creating an independent system to follow the attendance with low cost will help a lot to eliminate many downsides which exist in the traditional way such as saving time and effort for both sides (students, teachers/lecturers), and to make useful analytics out of the data obtained from the system.

In addition, this system is supposed to be connected with an Android/IOS app to enable the students to view their attendance status, to use their phone to behave as a Near Field Communication (NFC) identification tag.

The system is able to check students' attendance (in and out) with no more than one click. This software keeps track of tardiness, letting teachers know if specific students are frequently late to class. Students can be tracked from the moment they enter the school, even before the class has officially begun. This is especially useful in case of emergency to know exactly who are in the school premises.

This thesis studied the present attendance in Metropolia and proposed an IoT-based system connected to the cloud that uses the NFC/RFID technology, using Firebase platform as a backend and storage service. The physical device which is put at the class room is a

Raspberry Pi 4 model B. The needed data of the students was encrypted by giving each student a User Identity (UID) and decrypted later when data was transferred to the school or the university official system. In this way, the privacy of the students will stay protected and the system will not be threatened with the possible General Data Protection Regulation (GDPR) in EU.

The project started by defining the research problem. The researcher's own experience in everyday interaction with contact attendance and the deliberate observations about the current way of doing the attendance contributed to defining the problem. The next step was to find the most suitable research design and empirical method for the thesis. The thesis continues with search of the available technologies which serve to create attendance system based on IoT and cloud computing solutions. The implementation started by coding such a cloud-based system which works with IoT solutions with taking into consideration the requirements of system and limitations. The testing and the feedback that came later from students and lecturers were both documented in the chapters of this thesis. Finally, the results and their discussion were written to make conclusions about the significance of the IOT-based attendance system and its implications.

The thesis has been divided into 6 sections. The first section introduces the problem of the thesis, including a short description of the project and the guidelines that were used to create the project. The second section continues with current state analysis and the project specifications. The main motivation for this is that the current technology has to be studied from an engineering perspective to allow for informed and well-grounded decision for what kind of system is needed and what are the limitations applied on the project such as the legal, security, and financial limitations. The third section dives into the available technologies in 2020 which can serve the goal of the thesis to be accomplished. The section studied the available programming languages and backend management methods as well as the cloud providers and IoT hardware. The fourth section describes the implementation of the project divided into three sub-sections which include frontend, backend, and the hardware (IoT) implementation. The fifth section talks about the results and analysis, discussing briefly what has been done in the project development. The sixth section includes the discussion and the conclusions of the thesis.

## 2 Current State Analysis / Project Specifications

Metropolia as a university has thousands of students divided between Bachelor, Masters, and Open University Students, and has a huge number of lectures everyday taking place in the 4 campuses (Arabia, Myyrmäki, Karamalmi, and Myllypuro). This has created a need to have a centralized system to get the attendance information of all these students in an efficient way. Currently, teachers and students consume time and effort from both sides when first doing the paper attendance and later moving it to the system. The need to get more data than just Attended/Absent students is growing many times especially for the analytics and studying the behavior of students' attendance. The new system should be able to provide a detailed information about each student.

Because of the current situation of COVID-19 and the need to have as less as possible of shared physical contact between students to avoid transferring viruses, this system should have the ability to be hygienic and work without a physical touching.

### 2.1 Security and GDPR

One of the first things that should be considered when creating a system which involves human data is the protection of the data and keeping people's privacy away from unauthorized people to view it. GDPR (General Data Protection Regulation) is the newest regulation in EU that became applicable May 2018. All systems and companies in the EU and non-EU companies who deal with people's data in the EU should follow this regulation.

The attendance system which was created needs to follow the GDPR requirements. The system must have high encryption that makes hacking impossible or at least very difficult. Only the most needed information of clients' privacy should be collected, for example the social security number of students should be changed with another code that only system workers know how this code is linked to the certain students (gdpr-info.eu, 2016).

## 2.2 Cost

It is known that the first thing all stake holders are thinking about is “How much will the new system cost”. The benefits that comes from this system must be compatible with the cost of building, maintaining and training personnel on this system. It should be easy to learn and simple so the maintenance would be as easy as possible.

The attendance system comparing with other web applications such as Facebook, YouTube, Wolt, Foodora, and other big applications is too small from the aspect of technical and hardware requirements. There are many cloud services’ providers such as Google which provides a comprehensive tool to the application management and the ability to scale up with just one click. These service providers are using the pricing type “Pay as you go” which means they will bill only the time when the system is used (GoogleCloud, 2020). This type of pricing could reduce the cost to the minimum as the average use of the system will be 8 hours/day. When the system starts to work in the actual environment and start to collect data. It will need minimum requirements to work sufficiently. Later, after the system scales up and the size of data starts to exceed, normal backend servers will need to be replaced with a bigger capacity hardware, but with the case of using Google Cloud or Amazon Web Services (AWS) that will not take more than one click to upgrade the backend hardware and it will be billed from the time was upgraded. At the same time, if the system is not working in case of summer or spring holidays, it will not be charged anything.

## 2.3 Human Interaction with System

Successful systems are more independent than the ones who need manual interaction from human each time. Because of that, this system should minimize the administrators’ rules in operating it. In other words, this system should be serving administrators, not vice versa.

The system would switch on and off automatically and during the times of lectures only. Students also should not need to choose that they are checking in or out, as the system should recognize that by itself. In this way, two goals will be reached. The first goal is

that less effort is done by users of the system. The second goal is less bugs and less problems for the system itself. For example, letting students press each time one of the two buttons (the first button is for check in and the other one is for check out) would make the depreciation faster and would also include the possibility of making a mistake by students to press the check-out button when they want to make check in and vice versa. These features are highly needed for a sufficient and smooth work of the system.

However, the idea of this project is to find a working prototype, which can be taken into pilot use and to collect feedback for further improvement. For this reason, the project specifications at this point of time were quite generic.

# 3 Available Technologies

This part discusses the available technologies in 2020 which could serve to get the best practices to solve the research problem. Section 3.1 provides an overview of what are cloud-based applications , their history, the most common hosting providers, types of cloud-based applications, and how they work. Section 3.2 provides a brief introduction of what is frontend and how does the frontend design could determine the success or failing of the whole system, this section also discusses the available tools for designing and building frontends including up-to-date statistics of what is the trend currently in 2020. Section 3.3 outlines the backend side of every system, showing the methods which could be used to build a solid and secure backend that could support all the needs of the system. It reviews the available technologies of building backend and determines the difference between server and serverless. This section also covers what kind of programming languages are used in the backend and what is the trend in 2020.

## 3.1 Cloud-Based Web Applications

A cloud application is a software placed in the cloud, usually on third-party servers such as AWS or Google hosting services accessed on the internet, system owners could have their own servers also if they want. Providing data access from anywhere is the top reason for cloud adoption. General trend is to use the cloud for creating new systems nowadays (Rouse & Shore, 2012).

Tim Berners-Lee wrote a proposal paper ”Information Management” in March 1989 which led to develop later the first website ever (http://info.cern.ch/hypertext/WWW/TheProject.html) by Tim himself. Tim’s intention was to automate information-sharing between scientists in universities and institutes around the world. Since that time and the internet development is updating in each moment of the time. Although it was not Tim’s intention to create nowadays’ internet, his idea changed the world as the internet started to be used in everything in our life (Ryan, 2010).

Cloud-Based Web Applications could be put on Public, Private, or hybrid cloud. A public cloud is a cloud service offered to multiple customers by a cloud provider. The public cloud" is used to differentiate between the original cloud model of services accessed over

the Internet and the private cloud model. Public clouds include Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) services.

In a nutshell, software as a service (SaaS) provides a single, common infrastructure and code base that is centrally maintained for users, with no need to worry about anyone accessing their personal information without authorization, a clear example of SaaS applications is the online Photoshop software (Turner, 2020). From the other side, infrastructure as a service (IaaS) provides virtual machines or storage from a provider on demand with elastic scalability and platform as a service (PaaS) provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app (CloudFlare, 2020).

In contrast, Private Cloud could refer to cloud service offered to only one organization without sharing the same cloud with others. Each one of these types has pros and cons sides for example Public cloud has lower cost, less time spent to manage servers by the company itself. From the other side, a private cloud has more privileges on the public cloud by less leakage of data because of the data management, avoid to vendor lock-in with the cloud providers who could get indirect authority on the organization who at one point should accept all the cloud providers rules and policies.

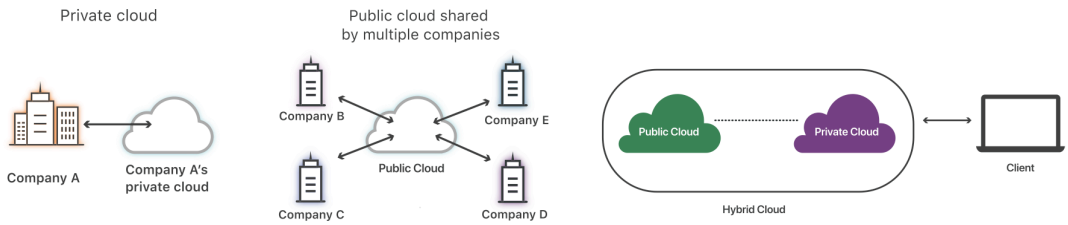


Figure 1. Types of clouds: private, public, and hybrid (CloudFlare, 2020)

From that came the term of a hybrid cloud to mix two or more of the cloud types (Public, Private or on-premises). Hybrid clouds are used if an organization want to use their private cloud for some services and their public cloud for others, or they may use the public

cloud as backup for their private cloud. They can also use the public cloud to handle periods of high traffic on their private cloud, while keeping most operations within their private cloud (Rountree & Castrillo, 2014).

Statistics show that since the launch of the internet in 1990, almost all businesses have started to rely partly or completely on the cloud services, about 80% of businesses around world have a website on the internet in 2018 statistics and it has been growing rapidly. As many as 90% of companies are on the cloud and 84% of these companies run on a multi-cloud strategy (451Research, 2018). In 2020 The global public cloud computing market is set to exceed $330 billion in 2020 (Hostingtrybunal, 2020).

From the statistics above, we can feel the importance and the rapid growth of Cloud Services for enterprises and could also conclude that companies who don't adopt the cloud services partly or completely will not survive for a long time in the market.

The cloud, as much as it is useful by globalizing the services, it also has some risks of hacking the company's systems to get their private information. That is why companies could choose between Public and Private Cloud according to the risk if they publish their information to everyone. Some companies could choose the two types of Cloud Computing to balance the need of the security and privacy with the benefits of Public cloud.

## 3.2 Frontend

A Frontend term refers to the client-side layout and the visual aspects of the web app that a user can see and experience. System layout must be well-constructed with impressive visuals, meaning that the design of the frontend must be compatible with the user experience. To design a layout of a system, the developer needs to know at least one or more of the programming languages such as HTML, CSS, and JavaScript.

Web applications used to be static and simple text sites with a humble formatting. That was thanks to CSS and HTML. But during the last 10 to 15 years, frontend development has changed considerably with the dramatic growth of JavaScript programming language, which was not as universal on the frontend as it is currently.

Some of the most important technologies used in frontend web development include the following:

**HTML:** The base on which every site on the web is built, it is a crucial technology in frontend development. HTML5 is the most recent HTML specification.

**CSS:** The Cascading Style Sheet (CSS) enables developers to add effects and styling to the web applications. Styles can be added globally, then masked on without changing that original styling that is applied to the entire site. CSS and the related scripts are always developing. The newest and latest version of CSS is CSS3.

CSS frontend frameworks like Bootstrap or Foundation can help to create refined web applications instantly. Several frontend developers use a CSS pre-processor to make coding much faster and to avoid repeating. User experience (UX) and user interface (UI) Design should be used carefully to bring the frontend to life. It would be careless to talk about frontend development and not mention the design aspect: these two things are closely connected in application development and websites. Many frontend developers take different roles, with UX/UI and web design skills in their sleeve. If the designer knows which things are possible with frontend code, they can design user interfaces that are more interactive and intuitive.

Nowadays, after the wide spread of mobile phones and tablets, Frontend technology was expanded to have more branches to accommodate these two new platforms and this kind of development, in turn, is divided into many sub-categories such as iOS, Android, and Windows systems development. Generally, the same languages were used in the web development. Native app developers use Swift or Objective-C for iOS applications, Java or C++ for Android applications and C# for Windows Phone applications.

**JavaScript:** It was once a supplementary tool to make web pages more interactive is now the most ubiquitous client-side technology. JavaScript is more than just a language — it's an entire ecosystem that spans frameworks, task runners, server-side development, and more.

Many Frontend open source tools were created to help with design and create web applications such as WordPress, Wix and other Joomla. Statistics show that 59% of the internet websites are made by WordPress, in the second place comes Joomla with 6.6% of the market share. Coded websites are less than 21.3% of the total amount of the websites on the internet in 2020 and the rest of the web applications are made by open source tools (Sawyer, 2018).

## 3.3 Backend

Every web app has a frontend and Backend. In the last section we talked about the frontend that is basically what the clients see when they interact with applications. Now backend or Server-side refers to all operations functionality that happens under the hood where the clients will not see. Clients can never access or see what is happening in the backend, all that the clients see is the front end.

Backend involves the programming of a computer placed mostly on the off shore and responsible for responding to what users request when interacting with the frontend side of the web application. Backend could be described basically as this example. A person enters one restaurant and ask for their menu, the customer chooses what he/she likes and gives his/her order to the waiter, after a while the customer gets his/her meal ready in front of him/her. The client did not see how this meal was prepared in the kitchen, all what they saw was the menu. This is exactly what happens in every web app, the menu is the frontend and the kitchen and chefs are the backend of the web app and the waiter who takes the orders from the client to the kitchen functions as a server in web applications.

A Server-side is written by one of the programming languages such as PHP, Python, Ruby on Rails, Java, and Net to get the backend job done. According to W3Techs (Web Technology Surveys) the most popular backend programming language is PHP with 79% of websites backend. That probably comes because the most popular open source website builder (WordPress) uses PHP as a backend. Although Python and JavaScript are growing rapidly among the coded cloud applications (W3Techs, 2020).

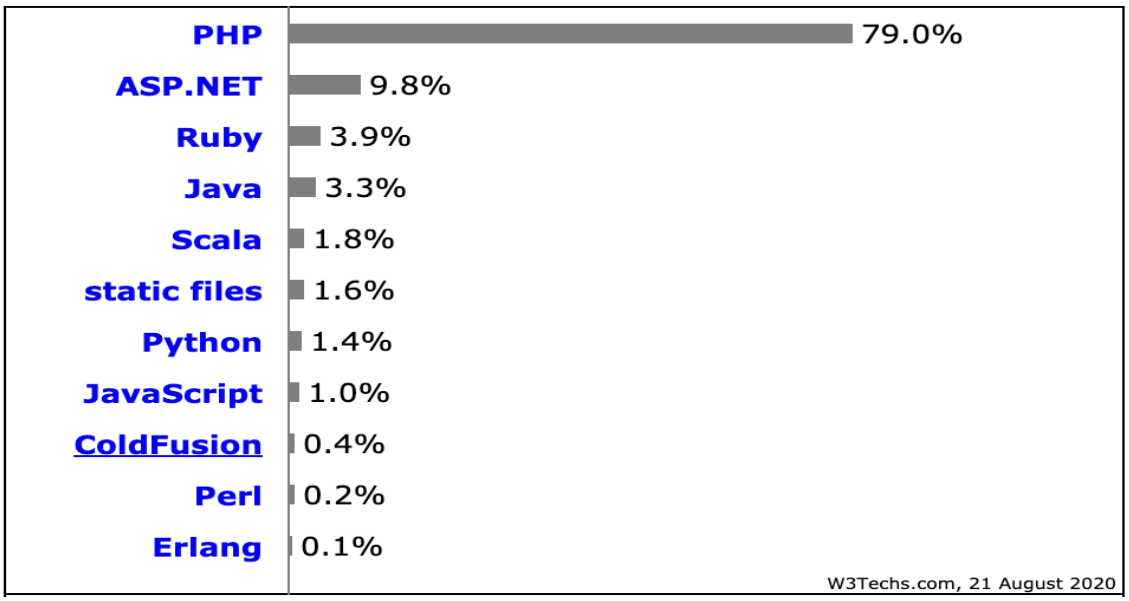


Figure 2. Percentages of websites using various server-side programming languages. (W3Techs, 2020)

Nevertheless, a new technology of managing Cloud-Based applications' backend is called Serverless technology that started to be popular since 2015. Mike Roberts, the co-founder of Symphonia which is consultancy focusing in cloud architecture explains about Serverless technology by saying "Serverless architectures are application designs that incorporate third-party "Backend as a Service" (BaaS) services, and/or that include custom code run in managed, ephemeral containers on a "Functions as a Service" (FaaS) platform." This means that foundations do not need to care about the backend infrastructure and the hardware management anymore, as it is going to be managed by a third party who provides the backend support and bill by 'pay-as-you-go' basis, meaning that developers only have to pay for the services they use. Another privilege of Serverless is that it scalable in anytime, Serverless also more productive than traditional server managed applications by simplifying the backend software development (Reberts, 2018).

The most popular serverless providers currently are Microsoft Azure, Amazon Web Services (AWS), Apache Open Whisk, Google Cloud Platform (GCP), and Cloudflare Workers. One of the new growing mobile application development platforms called "Firebase" is owned by Google is growing since 2015. Serverless is the most rapidly growing cloud service model at the moment, with an annual growth rate of 75% in 2018 (Flexera, 2020).

## 3.4 IoT (Internet of Things)

This section introduces what is IoT, how it is used in our daily life, and a few samples of devices used in the IoT projects. The Internet of Things (IoT) is inspiring our contemporary ways of life from the way we react to the way we behave. From light bulbs that the person can control with their mobile phones to smart vehicles that guide to the shortest route or the smart watch that can track the daily activities. Internet of Things (IoT) is an enormous network with connected devices collecting and sharing data about the way they are used and the environment in which they are operating. It is done by using sensors which are embedded in each physical device. It can be a smart phone, traffic lights, barcode sensors, electrical appliances, and almost any device in our daily lives. These sensors are constantly producing and sending data about the working state of the devices, but one of the main points is how do they share this enormous amount of data, and how to do get benefits out of this data.

Internet of Things (IoT) offers a shared platform for all these devices to store and process their data and a common language for all the devices to link with each other. Data is produced from various sensors and sent to IoT platform. The secure IoT platform integrates the collected data from various sources. Further analytics are performed on the data and valuable information is obtained as per requirement. The output is shared with other devices for automation, improving efficiencies and better user experience.

### 3.4.1 Raspberry Pi 4 B

The best way to describe what is Raspberry Pi is from the official source of the Raspberry Pi producer (RaspberryPi, 2020) “The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It’s capable of doing everything you would expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.”

In IoT projects, Raspberry Pi plays extremely important role to create a centralized unit to input and process data. Raspberry Pi as a single circuit board computer is used widely

nowadays for many reasons. These reasons are the productivity that these devices can do comparing with their cost and size. The organization which produces this product produced in charity based in the United Kingdom that works to give the power of digital making and computing to the people around the world in low cost. Their goal is to get the opportunity to solve problems that matters to them by using the Raspberry Pi products. Since 2012, when Raspberry Pi foundation launched their first product to the market with modest capacity, their product used a 512MB RAM, Video Core IV graphics processing unit (GPU), and had a Broadcom BCM2835 SoC which includes a 700 MHz ARM1176JZF-S processor. This was then complemented with the launch of a lower cost Model A which had less memory and USB ports.

The foundation launched their latest product with a surprise of the new features and a cheap price from 35 USD/computer. This product was given the name "Raspberry Pi 4 Model B" and has the following highlights: A 1GB, 2GB, or 4GB of LPDDR4 SDRAM, 1.5GHz quad-core 64-bit ARM Cortex-A72 CPU (~3× performance), Dual-band 802.11ac wireless networking, Full-throughput Gigabit Ethernet, Two USB 3.0 and two USB 2.0 ports, Bluetooth 5.0, Dual monitor support, supporting OpenGL ES 3.x, at resolutions up to 4K, Video Core VI graphics, 4Kp60 hardware decode of HEVC video, and a full compatibility with previous Raspberry Pi products (Upton, 2019).

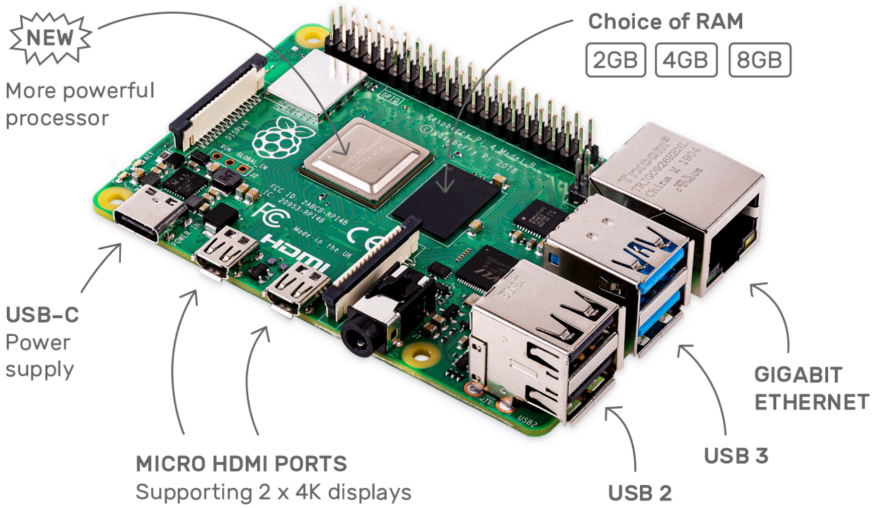


Figure 3 Raspberry Pi 4 Model B (RaspberryPi, 2020)

The foundation of Raspberry Pi has sold by the end of 2019 more than 30 million product all around the world since the launch of the foundation (Upton, 2019)

### 3.4.2 Input devices

In the IoT field, input device refers to all sensors and reading data devices such as thermometers, temperature sensors, humidity sensors, motion sensors, cameras, and many other special kinds of sensors and input devices. A sensor is a device that measures physical input from its environment and converts it into data that can be interpreted by either a human or a machine. Input devices are the major components in IoT projects as they are the first step of any IoT device (Bhatt, 2011).

Sensors are often designed to have a small effect on what it is made for, making the sensor as small as possible frequently improves this feature and may also provide other advantages. A good sensor complies with the three guidelines. First, it should be insensitive to any other property likely to be encountered in its application. Second, sensitive to the measured property. Third, it does not influence the measured property (Dincer, et al., 2019).

Barcode, NFC, and RFID sensors are widely used in IoT projects. Radio-frequency identification (RIFD) mainly uses radio waves and protocols to read and write data on electromagnetic fields cards and tags in a certain frequency band. RFID is used in many industries to read, authenticate, write, and track products. The RFID system consists of two main parts: RFID reader or interrogator and RFID tags or cards. The way it works is by continuously sending radio waves of a certain frequency. If the object on which this RFID tag is attached is within the range (0-10 CM) of this radio waves, it sends the feedback back to this RFID reader and based on this feedback, RFID reader identifies the object.

Currently, there are three different types of RFID tags commercially available: active tags, passive tags, and semi-passive tags. The active tags have a power source for their internal circuitries. They also use their own power supply for sending the response to the reader, while passive tags do not have any power supply. They used to get their power from the incoming radio waves from the readers. Semi-passive tags on the other hand do

have a power supply for internal circuitries, but they rely on the radio waves received from the reader for sending the response.


Figure. 4 RFID Reader (tomtop.com, 2020)

RFID systems are operated in three frequency bands: High-Frequency band (HF), Low-Frequency band (LF), and Ultra High-Frequency band (UHF).

High-Frequency band (HF):

HF systems provide reading distances of 10 cm to 1 m and operate in the 3 MHz to 30 MHz range. Typical applications contain electronic payment and ticketing and data transfer. Near Field Communication (NFC) technology is based on the high frequency (HF) RFID. It has been used for instance in hotel key card applications and payment cards. Other types of smart card and proximity card payment and security systems also use HF technology. Standards include, ECMA-340, ISO 15693, , ISO/IEC 14443A, ISO/IEC 18092 (for NFC) and ISO/IEC 14443 (for smart card solutions such as MIFARE) (Myerson , 2006, p. 59).

Low-Frequency band (LF):

RFID systems have a read range of up to 10 cm,  and operate in the range 30-300 KHz. Although they have a slower data read rate and shorter read range than other technologies, they do perform better in the presence of liquids or metal that can interfere with other kinds of RFID tags. ISO/IEC and ISO 14223 18000-2 are among the typical standards for low frequency RFID. Real-life examples of low frequency tags can be found in livestock tracking, access control, and other applications where a short-read range is applicable (Myerson , 2006, p. 58).

Ultra-High-Frequency band (UHF):

These systems offer read ranges up to 12 m and have a frequency range operate between 300 MHz and 3 GHz. They also have faster data transfer rates. In addition, they are more sensitive to interference from electromagnetic signals, metals and liquids. However, some of these problems have been alleviated by new design innovations. UHF tags are commonly used in pharmaceutical anti-counterfeiting, retail inventory tracking and other applications where significant amounts of tags are needed because they are much cheaper to manufacture. The EPC global Gen2/ISO 18000-6C standard is a well-known global standard for item-level tracking applications (Myerson , 2006, p. 192).

As a conclusion, for a student attendance system, HF 13.56MHZ would be the most suitable RFID type to be used in reading students' cards.

# 4 Implementation

This part of the thesis discusses the implementation of the system. Section 4.1 outlines the project description and the overall structure of the system prototype. Section 4.2 provides the technology about the frontend of the system and why these technologies were chosen, including the system frontend programming language and the layout of the client side together with the dashboards. Providing screenshots of the frontend of system prototype. Section 4.3 outlines the structure of the backend, showing the method that was used to administrate the backend. It reviews whether the researcher used server or serverless technology and why this was chosen. This section also covers what kind of databases were used for the system and how is the structure of the database, providing screenshots of the frontend of system prototype. Section 4.4 provides an overview of the environment setup. It explains the various elements of the environment in which the system will work, including the physical and virtual environment.

## 4.1 Project description

The project is a cloud-based IoT application that was developed to automate the attendance taking process of students in Metropolia. After taking the attendance, many operations works under the hood to create a real-time informative dashboard that could be maintained by a personnel or lecturer. The developed platform provides a part of all required features which are mentioned in the theoretical part of this thesis. Nevertheless, more functionality can be easily added to the application. At the moment of writing this paper, the following interfaces were implemented:

• A dashboard to monitor the attendance of a class and to download the attendance of the day to CSV or .Json format. Including:

a) A set of real-time updated graphical information to show the total number of attendees, absents, and total number of students who signed in the course.

b) A bar chart shows in the real-time the total attendance of each day for the last month.

c) A data table to present the attendance of the day (check in and out). This table featured with a button to download the attendance to CSV or Json format.

• Students' interface to do the check in and out. This interface includes:

a) Input field for students to do their check in and out. This input field is contactless, students do the attendance via the RFID technology. Shows each student's ID with his/her check in and out time and has the ability to input each student's attendance to the backend (database).

b) A table to present students' check in and out status of the day. Students' names and other details are not included in this table to follow the GDPR. Instead, used an UID for each student.

• An IoT kit including (programmed Raspberry Pi, RFID reader, and 7" screen) to work as a portable device that could register the attendance using Radio Frequency Identification (RFID) cards. This kit will be fixed in the classroom.

• An organization interface to represent the features of the system and how it works.

• A system admin interface to manage the databases, authentication, rules, users, analytics, and many other rules.

## 4.2 Environment setup

The Cloud-based application was developed on a Mac running macOS Catalina version 10.15.5 operating system. Apart from the Mac and OS, the implementation required a development environment to be setup. The development environment consists of different software programs and tools to work with them. Thus, the following were installed:

ReactJS

ReactJS is the essential part for the development process of this project as it provides JavaScript library for building user interfaces and runtime environment for JavaScript code execution. It can be installed from the NPM tool (Node package manager) via this line (npx create-react-app app-name). After executing the command line, the easiest way to check if it is working properly, is to run (npm start) command in the terminal or the created folder. The command launches an internet tab works on default server http://localhost:3000 and prints a simple website that has only the React logo. The 16.9.0 version was used during the development (ReactJS.org, 2020).

Node JS

Node.js was installed as another essential part for the development process of this project as it provides the runtime environment for JavaScript code execution. Node JS package can be downloaded from the official website (https://nodejs.org/en/download/). After the installation, the easiest way to check if it is working properly, is to run node -v command in the terminal. The command prints the installed version of Node.js. The 12.18.3 version was used during the development which is recommended by most users in September 2020.

React-based dependencies

The following dependencies were installed to develop the frontend of the application: Material UI, React Bootstrap, Axios, React Chart, Firebase, Moment, History, React Date Time, React Router, and many other dependencies which come with the default react package.


Figure 5 Dependencies were used at the frontend development

Visual Studio Code

Visual Studio Code was used as a code editor to develop the frontend of the project. It is a powerful and lightweight source code editor which runs on desktop and is available for all kinds of operating systems including macOS, Linux, and Windows. It comes with a great built-in support for TypeScript, Node.js, and JavaScript and has a rich package of extensions and addons which are mostly free of charge for other languages (such as Python, PHP, Go, C++, C#, Java) and runtimes (such as Unity. and NET). This code editor made the development environment much easier and provided a comfortable tool to help with coding the project from scratch. The 1.40.0 version was used during the development (visualstudio.com, 2020).

Firebase platform

Firebase is a platform owned currently by Google for creating mobile and web applications. Firebase helps with building apps fast, without managing infrastructure as the backend will be managed by Google. Firebase is not only a hosting platform, it has much more features to increase the functionality such as analytics, databases, messaging and crash reporting. Firebase was tied with the frontend project by running (npm i firebase-tools) command in the terminal of the project (Firebase.google.com, 2020).

## 4.3 Frontend implementation

Frontend is where the user will land up or see the first time. Having a simple to navigate and to understand UI helps the user find what they are looking for quickly. Because of that, it was important to design and implement the frontend carefully. The implementation started when the researcher started to look for the available technologies to build the frontend, navigating through them and try each was will be chosen in the end. The frontend of the project was divided into three parts Students, Administrator (Lecturer), and admin interfaces. The challenge was the huge amount of available ways to create such interfaces.

At the end, coding development via JavaScript library called React and its dependencies. The reason of choosing coding the project was because the ready frontend designing tools such as Wix or WordPress were not able to fulfil the requirements. React library is one of the most common libraries to create interfaces and Single Page Applications.

### 4.3.1 Students' interface

The requirements which were set in the chapter two of this thesis, were taken into consideration. Cost effective, secure platform, easy to use and to maintain the project – these qualities made React JS to be chosen. The current situation of CODIV-19 enhanced the need to have a hygienic physical part of the attendance system. Because of that, a touchless attendance process had to be done. Below is the code which executes the function of posting the input (Student's ID) to the backend and get the response without any touch

from the user as it shows below, if the input was 7 digits long then executing the function to post data.

```
// Post data to server
const studentCheckIN = {
  [id]: { checkIN: moment().format("LTS"), checkOUT: null, ID: id }
};
const studentCheckOUT = {
  [id]: { checkOUT: moment().format("LTS") }
};
const sendDataToServer = () => {
  const dataRef = db.collection("metropolia").doc(`${datee}`);

  if (!students.map(student => student.studentID).includes(id)) {
    setMessage(
      ` ${id} does not exist in the database. Please, contact the lecturer `
    );
    setTimeout(() => setMessage(""), 5000);
  } else {
    dataRef.get().then(() => {
      if (attendance.map(value => value.ID).includes(id)) {
        dataRef.set(studentCheckOUT, { merge: true });
        setMessage(` ${id} Just checked_out `);
        setTimeout(() => setMessage(""), 3000);
      } else {
        dataRef.set(studentCheckIN, { merge: true });
        setMessage(` ${id} Just checked_in `);
        setTimeout(() => setMessage(""), 3000);
      }
    });
  }
  setId("");
};
```

Figure 6 Code which executes the function of posting the input (Student's ID) to backend.

A validation step has been added to the function as a first step when executing by calling the backend to check if the UID was input is existing in the database or not. If the UID does not exist in the database, the following message will appear on the frontend, such as the below example. This step is important also to filter out the unregistered students to make a check-in to the system.

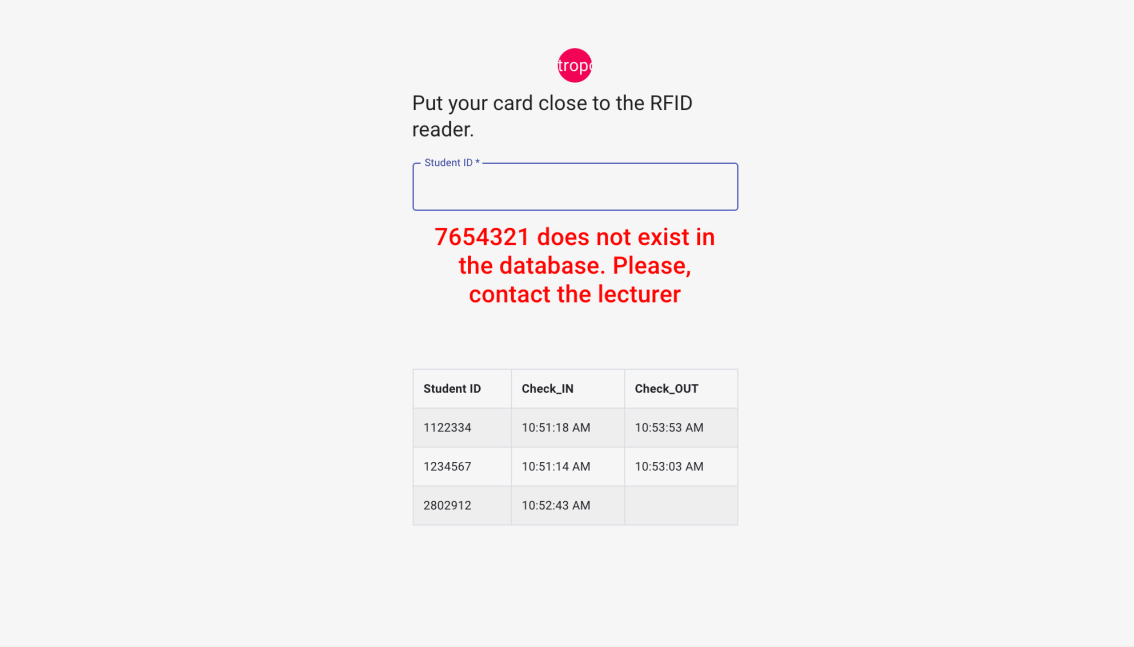


Figure 7 Students frontend layout when input irrelevant UID.

And to reduce the traffic between frontend and backend, the students' table will be updated only when there is change in the database which is called Database snapshot. This feature gives a brilliant solution to reduce the traffic between database and frontend as it is basically instead of the traditional way of calling the backend each certain time to check if there's update or not, instead, the role will be the other way around by putting the responsibility on the backend to inform the frontend by any changes happen in backend in real time.

```
/// Get data from server
useEffect(() => {
  db.collection("metropolia")
    .doc("students")
    .onSnapshot(
      docSnapshot => {
        setStudents(docSnapshot.data().students);
      },
      err => {
        console.log(`Encountered error: ${err}`);
      }
    );
  db.collection("metropolia")
    .doc(`${datee}`)
    .onSnapshot(snap => {
      snap.exists
        ? setAttendance(Object.values(snap.data()))
        : console.log("data doesn't exist");
    });
}, [datee]);
```

Figure 8 Code to fetch data from database

This is how students' interface looks. As it could be recognized, students' names or any private information will not be shown for security and legal reasons. Instead, a UID which represents a certain student will be shown.

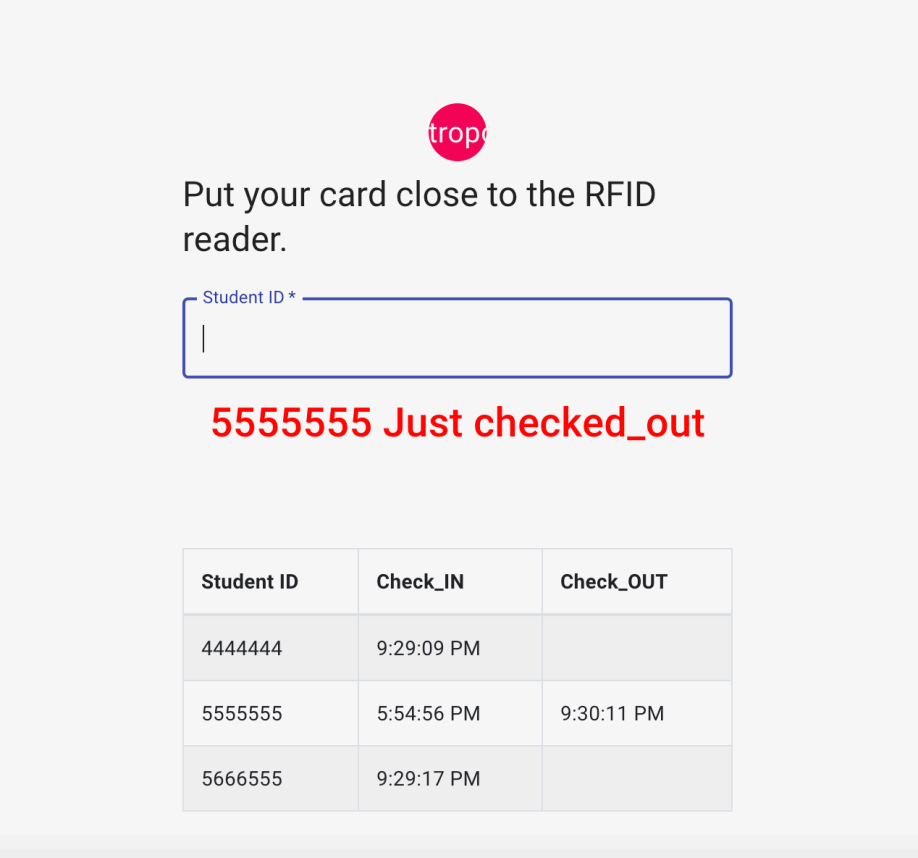


Figure 9 students' interface

The real time update and the ability to work offline when the internet connection is intermittent. With offline persistence feature enabled, the Cloud Firestore (Database) client library automatically manages offline and online data access and synchronizes local data when the device is back online. So, the system can listen to, write, read, and query the cached data. When the device comes back online, Cloud Firestore synchronizes any local changes made by the app to the Cloud Firestore backend. In case of many devices were offline and all have access to the same database collection. In this case, Firebase has a system to priories the update were made earlier by any device.

### 4.3.2 Administrator interface (Dashboard)

The attendance taker regardless whether it is the lecturer or the administrator whose work is monitoring the attendance statuses in real-time are provided with a dashboard interface which need to be authenticated to display and download the information about attendance. Material UI library and React chart were used to create the below dashboard.

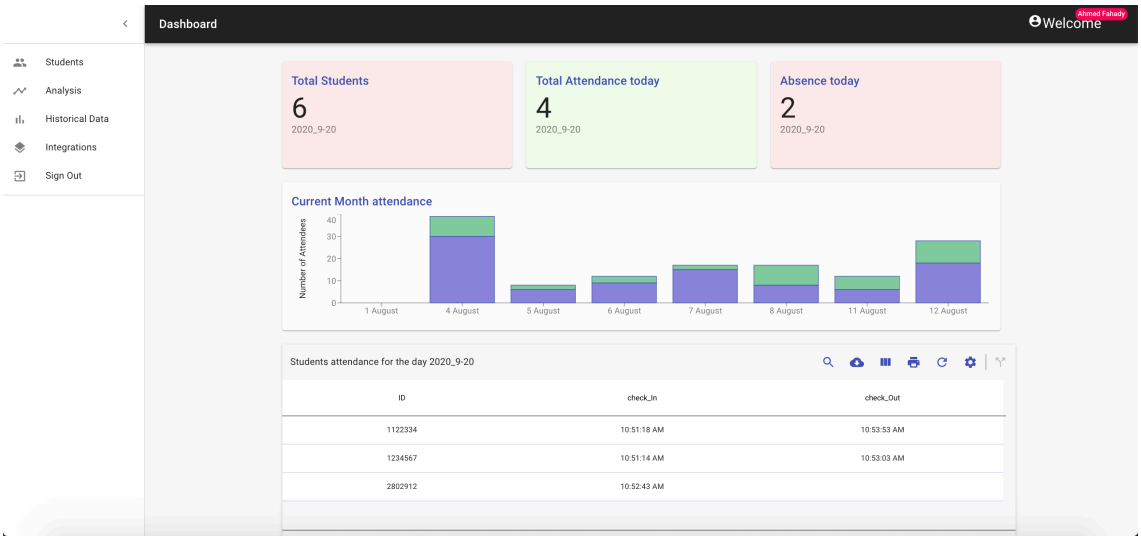


Figure 10 Administrator's dashboard

As it can be seen, the dashboard includes the total number of students in the course beside how many attendees and how many absents for the day. It has also a bar chart to represent the number of the attendants and absents of each day for a month. It has also a table to display all checked-in and checked-out students at the moment. Through the dashboard, the administrator has access to historical data and analysis which can be downloaded easily or integrated with other systems.

### 4.3.3 Admin’s interface

Admin’s interface was built on top of Firebase platform. When creating a project in Firebase console, the admin will be provided with a comprehensive monitoring dashboard to manage the project.

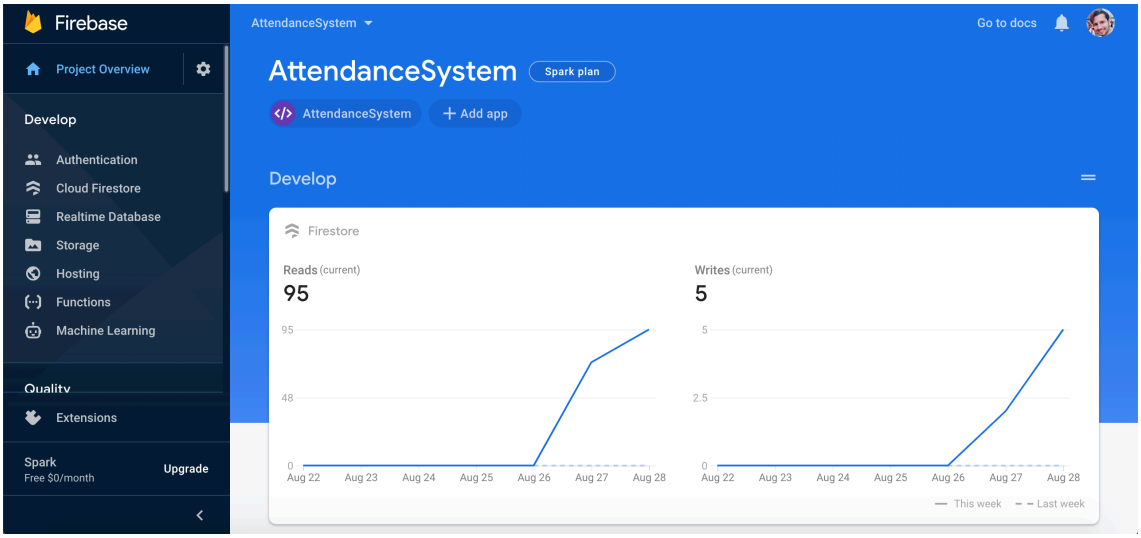


Figure 11 Admin's general interface layout (Backend)

Through the above dashboard, admins can access, create, edit, and delete databases. APIs can be created from the admin's interface or from the Command-line interface (CLI). Admins also have access to a specific information such as the number of reads writes, deletes, and snapshots of the database. This information gives clear insights about the usage.

## 4.4 Backend (Firebase)

Backend is the part of the application that is not directly accessed by the user. Instead, the backend's job is to provide the frontend with the functionality needed for working efficiently. A solid backend should be built to fulfil the purpose. To choose what kind of backend is needed, it must answer the following questions: What kind of backend could be used? Would it be server or serverless backend? What kind of databases would be the best? How the authentication will be? And considering the requirements, it ended with choosing a serverless backend on top of Firebase platform.

Firebase uses a software development kit (SDK) which supports programming in JavaScript, Angular, JavaScript/Node.js, C++, Swift, Objective-C, and Java. Backbone, Ember and React are supported through bindings to the database. The SDK was integrated in the frontend project by the following snippet:

```
const firebaseConfig = {
  apiKey: process.env.REACT_APP_API_KEY,
  authDomain: process.env.REACT_APP_AUTHDOMAIN,
  databaseURL: process.env.REACT_APP_DATABASEUR,
  projectId: process.env.REACT_APP_PROJECTID,
  storageBucket: process.env.REACT_APP_STORAGEBUCKET,
  messagingSenderId: process.env.REACT_APP_MESSAGINGSENDERID,
  appId: process.env.REACT_APP_APPID,
  measurementId: process.env.REACT_APP_MEASUREMENTID
};

firebase.initializeApp(firebaseConfig);

export const auth = firebase.auth();
export const db = firebase.firestore();
```

Figure 12 the snippet of firebase SDK integrated in the frontend

Authentication

After integrating, it became the time to develop the authentication part of the project. Firebase makes authentication easy for end users and developers. Knowing the identity of a user is needed so the application can provide a customized experience and keep the data secure. Firebase supports lots of different ways for users to authenticate. The admin has the ability to allow or block one or more of the available authentication methods such users authenticate with their email address, Facebook, Twitter, GitHub, and Google. The user information contains a unique ID which is guaranteed to be individual across all providers, and never changing for a specific authenticated user. This UID is used to detect users and what parts of the backend system they are authorized to read, write, or delete. Firebase will also manage the user session so that users will remain logged in after the browser or application restarts.

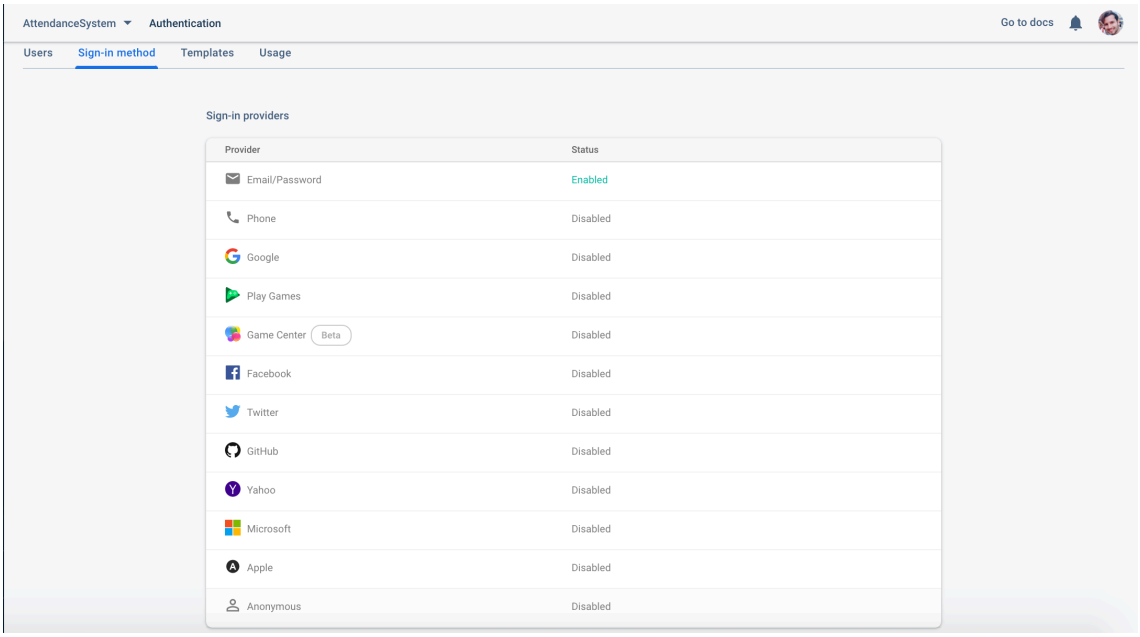


Figure 13 the available authentication methods supported by Firebase.

It also has a built-in authentication-related templates to make it easy for users to reset, change, and verify their emails and password.

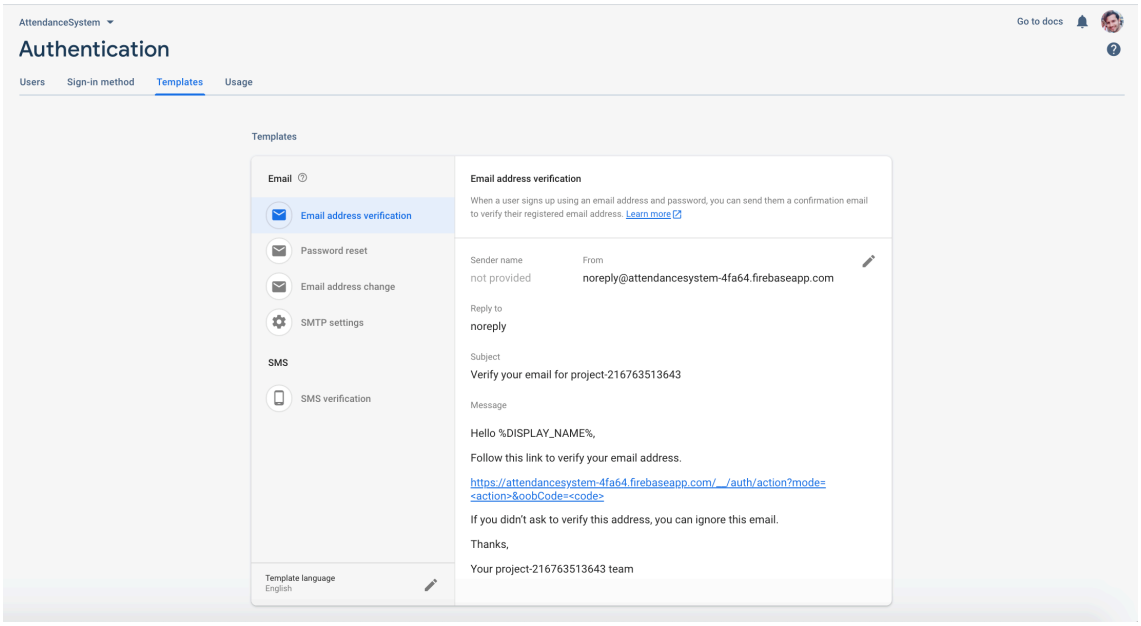


Figure 14 The built-in authentication templates of Firebase.

The application used authentication to access both the students' interface and the administrator dashboard for more secure application. Below is the code to redirect visitors if they are known or unknown by the system.

```
const Login = () => {
  console.log("current user is: ", auth.currentUser);
  const [authenticated, setAuthenticated] = useState(<Loading />);

  useEffect(() => {
    auth.onAuthStateChanged(user => {
      if (user) {
        setAuthenticated(<Dashboard />);
      } else {
        setAuthenticated(<LoginPage />);
      }
    });
  }, []);

  return <React.Fragment>{authenticated} </React.Fragment>;
};
```

Figure 15 the function which is responsible to authenticate users of the app

Database

Firebase also provided the project with a suitable database. A NoSQL database type was used by the project and was used to store all attendance, students, courses information each in a separate collection (i.e. table). The structure of the database was designed carefully to provide fully customized requests of data, the idea is to fetch only the needed information by the frontend, that results to reduce to minimum calls between frontend and backend which at the end make the application faster and cost effective. Although NoSQL databases are schemaless, still such a structure was needed to organize data inside the database. Below is how the data is stored in the database;

```
Metropolia {
        Attendance_date { Student_ID: {
                                Student_ID (string),
                                Check-in (Date and time),
                                Check-out(Date and time)},
        Students: [{courseEnrollment:
                                [ courses_name (string)],
                                Student_ID (string),
                                Student_name (string)}],

        Courses: [              {course_name (string),
                                start (Date and Time),
                                end (Date and Time),
                                lecturer (String)}]  }
```

Security Rules to secure data in Cloud Firestore were set to limit the access to only authorized persons to display and modify. The following code line was used to fill the purpose;

```
service cloud.firestore {
                        match /databases/{database}/documents {
                        match /metropolia/{metropoli} {
                        allow read, write: if request.auth.uid != null; }  } }
```

Hosting

As the application needs to be accessed at anytime from anywhere, it had to be put on the cloud. Firebase provides a smooth fast and secure hosting for the web applications, static and dynamic content, and microservices. The attendance system was deployed to the firebase hosting space via the firebase CLI by running (firebase deploy) command at the terminal of the project folder (Firebase-Hosting, 2020).

When deploying the project to firebase hosting, many features will be provided. The first time the project is deployed, it is provided with a domain name which could be later changed to the desired domain name. Admins have the ability to deploy many versions of the project and switch between them easily without losing any file. When the attendance system was deployed, it got this domain (https://attendancesystem-4fa64.firebaseapp.com/metropolia) and filled 41.6 Mega Byte (MB) of 10GB available.

Cost

Firebase has two different pricing plans, one of them is “Spark plan” (the chosen one) is free of charge if the usage does not exceed the following numbers;

- Authentication; 10K/month,
- Database storage (Firestore); 1 GB,
- Cloud functions; 125K/month,
- Hosting; 10 GB,
- Storage; 5 GB.

From what it was stated above, the project will work for many years and scale up several times more and will cost zero euros according to the pricing plan of Firebase (Firebase-Pricing, 2020). By doing so, the cost-effective goal with keeping the efficiency was reached.

# 5 Results and Analysis

The application was developed using React JS and Firebase. Serverless rendering with Material UI and Bootstrap were used to create single page application user interfaces and Raspberry Pi, RFID reader, and 7" screen was used in the IoT part of the project. The code was separated into different modules, such as views, assets, components, and config. Therefore, leaving the code clean and clear and very easy to maintain and scale up in the future.

However, the developed application is not ideal. The application lacks integration with the organization system where students' data are stored. Because of the limited time, an android/IOS app for students to view their attendance status and enabling them to do the attendance via their phone NFC feature without the need to carry the student card with them was not developed.

As the project is described in the paper, the single page application (SPA) method was used in the project development as SPA is a very popular approach nowadays and it has many advantages. Therefore, the routing and backend calls are in the frontend part. This way made it is very easy to modularize the components with the help of React JS and Firebase admin SDK.

On the 7th of September 2020, this system was tested with the students of Tends in ICT course in Metropolia. The 29 participants used their own RFID-based travel cards as a UID. Immediately after that, they answered a few questions in a survey form. Participants provided their feedback about the system. Although the survey was not my main focus, it still gave a lot of useful information about the attendance system and the participants' thoughts for the future improvement. The survey and answers are attached in the appendix.

This survey was designed carefully in a cooperation between the researcher and four of his classmates to form up clear and fruitful questions and how they could be asked was taken into deep consideration, ending up with six questions for students and seven questions for lecturers. At the end, the system got a high review from the samples of Metropolia students. However, some unexpected errors or behavior might occur while using the

application. Notwithstanding the fact that the project resulted in the clean, organized and functional prototype, there are always multiple different technologies and approaches that could be used, as well as space for improvements.

# 6 Discussions and Conclusions

The core target of this thesis was to study the technologies involved in the serverless full stack JavaScript development project and to create a working IoT-and cloud based application to monitor electrically students attendance in the courses. The research took a long time, but as a result, all the required technologies, such as React JS, Firebase, Linux, and IoT sensors were thoroughly analyzed.

After a detailed research of a theoretical side of the project, the practical part was carried out. Careful architecture design was completed first and then the actual implementation of the application was done. This work resulted in a working prototype of a platform for Metropolia students' attendance system. The purpose of the development was to find the best way to get students' attendance to different courses and though improve the traditional way of collecting signatures. The project can be further improved with a number of features such as integrating this system with Metropolia's own students management system and adding the ability to be used in the online lectures to transform it into a production ready system.

After a deep study and an actual implementation of the project using React Js and Firebase tools, it is possible to make several conclusions. First of all, it requires the knowledge of only one programming language, which can be used to develop the frontend and leave the backend to be managed by a trusted foundation such Google. Secondly, agile development is the master of the market among all other ways of development, as the constantly changing environment always needs continuous optimizations of the applications. Thirdly, when thinking about developing a new application, it is mandatory to think if the used technologies are scalable or not, as the environment is changing fast. However, making a use of the services that Firebase is offering currently will bring a big benefit to the project from many aspects such as being easy to scale up, cost-effective, and secure environment for the project.

# 7 References

451Research, 2018. *451Research.* [Online] Available at: https://451research.com/images/Marketing/press_releases/Pre_Re-Invent_2018_press_release_final_11_22.pdf [Accessed 31 08 2020].

Bhatt, A., 2011. *Engineers Garage.* [Online]
Available at: https://www.engineersgarage.com/article_page/sensors-different-types-of-sensors [Accessed 31 08 2020].

Bourg, D. M. & Seemann, G., 2004. *AI for Game Developers.* Sebastopol(California): O'Reilly Media.

CloudFlare, 2020. *The difference between a public cloud, a private cloud and a hybrid cloud,* s.l.: s.n.

CloudFlare, 2020. *What Is Platform-as-a-Service (PaaS)?*. [Online]
Available at: https://rb.gy/fer8rc [Accessed 15 09 2020].

Dincer, C. et al., 2019. Disposable Sensors in Diagnostics, Food, and Environmental Monitoring. *Wiley Online Library,* 31(30).

Firebase.google.com, 2020. *Firebase.* [Online]
Available at: https://firebase.google.com/ [Accessed 31 08 2020].

Firebase-Hosting, 2020. *Firebase.* [Online]
Available at: https://firebase.google.com/docs/hosting [Accessed 31 08 2020].

Firebase-Pricing, 2020. *Firebase.* [Online]
Available at: https://firebase.google.com/pricing?authuser=0 [Accessed 31 08 2020].

Flexera, 2020. *Flexera.* [Online]
Available at: https://info.flexera.com/SLO-CM-REPORT-State-of-the-Cloud-2020 [Accessed 31 08 2020].

gdpr-info.eu, 2016. *GDPR.EU.* [Online] Available at: https://gdpr-info.eu/ [Accessed 31 8 2020].

GoogleCloud, 2020. *Google Cloud.* [Online] Available at: https://cloud.google.com/maps-platform/pricing/sheet [Accessed 31 08 2020].

Hostingtrybunal, 2020. *Hostingtrybunal.* [Online] Available at: https://hostingtribunal.com/blog/cloud-computing-statistics [Accessed 15 09 2020].

Myerson , J. . M., 2006. *RFID in the Supply Chain: A Guide to Selection and Implementation.* illustrated ed. s.l.:CRC Press.

RaspberryPi, 2020. *RaspberryPi.* [Online]
Available at: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/ [Accessed 31 08 2020].

ReactJS.org, 2020. *React official website.* [Online]
Available at: https://reactjs.org/ [Accessed 31 08 2020].

Reberts, M., 2018. *Serverless Architectures.* [Online]
Available at: https://martinfowler.com/articles/serverless.html [Accessed 31 08 2020].

Rountree, D. & Castrillo, I., 2014. *The basics of cloud computing.* Waltham: Elsevier.

Rouse, M. & Shore, J., 2012. *SearchCloudComputing.* [Online] Available at: https://searchcloudcomputing.techtarget.com/definition/cloud-application [Accessed 31 08 2020].

Ryan, J., 2010. *A History of the Internet and the Digital Future.* Dublin: Reaktion Books.

Sawyer, L., 2018. *CMSC Media.* [Online]
Available at: https://www.cms-connected.com/News-Archive/November-2018/CMS-Top-Players,-Trends,-and-Market-Share-Growth [Accessed 2020 08 31].

tomtop.com, 2020. *pinterest.com.* [Online]
Available at: https://rb.gy/itvuf8 [Accessed 15 09 2020].

Turner, B., 2020. *TechRadar.* [Online]
Available at: https://www.techradar.com/news/what-is-saas [Accessed 2020 09 15].

Upton, E., 2019. *RaspberryPi.* [Online]
Available at: https://www.raspberrypi.org/blog/raspberry-pi-4-on-sale-now-from-35/ [Accessed 31 08 2020].

Upton, E., 2019. *Twitter.* [Online]
Available at: https://twitter.com/EbenUpton/status/1205646606504275968?s=19 [Accessed 31 08 2020].

W3Techs, 2020. *W3Techs.* [Online] Available at: https://w3techs.com/technologies/overview/programming_language [Accessed 31 08 2020].

visualstudio.com, 2020. *Visual Studio Code.* [Online] Available at: https://code.visualstudio.com/ [Accessed 31 08 2020].

Appendices:

Appendix 1. Survey for lecturers
Appendix 2. Survey for students
Appendix 3. Responses and analysis.