**Universität des Saarlandes**

**Hochschule der Bildenden Künste Saar**

Media Informatics

Project Topic: **Checkoo**,

Smart Attendance App with IOT Technologies

**Supervisors**: **Students**:

Dr. Michael Schmitz Berina Zenuni

Dipl. Designer Ralph Schneider Agnes Nakirigya

Namrata Gohil

Saarbrücken, February 2019

**Table of Content**

1. Architecture 2

1.1. User Interface 2  
 1.2. Software Architecture 2

1.2.1. Front End Technologies 3

1.2.2. Back End Technologies 4

1.3. Tools and Implementation 4

1.4. App functions 6

1.5. The scope of the project 6

2. Use Cases 7

2.1. “Add / Delete a Course” Use Case 7

2.2. “Login / Register” Use Case 8

2.3. “Course Overview” Use Case 9

3. Pro’s and Con’s 9

3.1. Pro’s 9

3.2. Con’s 9

3.3. How to make the app better 10  
 4. Literature 11

1. **Architecture**

The project is called “Checkoo”, as in Check-in manager. Professors and students are end users. An Android app has been created that allows student to submit their presence while attending lectures and professor can view lists of attendees.

**1.1. User Interface**

Following these mock-ups (<https://ninjamock.com/s/T51XMRx>) made in Ninjamock, the new ones have been created in Adobe XD:

For review:

<https://xd.adobe.com/view/6d3fc810-03e0-4027-4eae-608ad96cdd10-1b0e/>

For development:

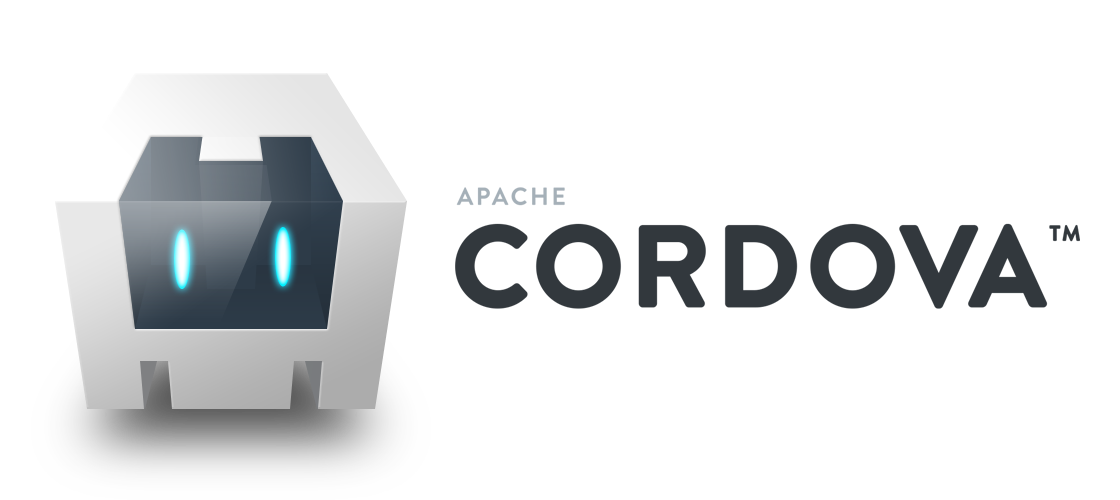
<https://xd.adobe.com/spec/6fce0166-0a0b-471d-7945-941f1275eb08-19ff/>

**1.2. Software Architecture**

* Front-end and Back-end:

Backend

Frontend



*Reference for images:*

https://cordova.apache.org/artwork/

https://www.litmos.com/marketplace/connected-apps/ios-app/

https://www.android.com/phones/

https://commons.wikimedia.org/wiki/File:Cordova-logo-by-gengns.svg

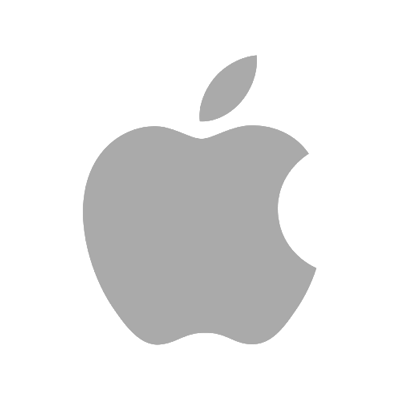
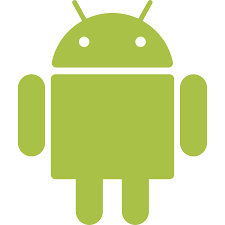
http://php.net/download-logos.php

https://en.wikipedia.org/wiki/Node.js

https://seeklogo.com/vector-logo/96581/mysql

https://medium.com/level-up-web/amazingly-useful-html-css-and-javascript-tools-and-libraries-d73b10fbae29

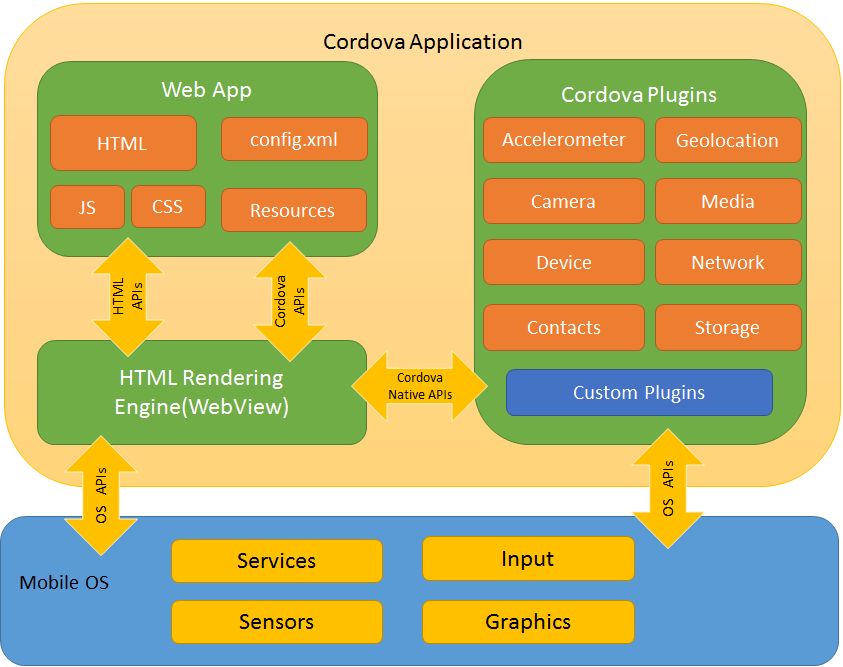
http://be-innovated.com/open-beacon-netwerk-estimote/



Database

*Picture 1. Software Architecture*

* Cordova architecture:



*Picture 2: Cordova Architecture*[[1]](#footnote-1)

The functionality of the mobile application allows access to the native mobile device services via API. Cordova has additional libraries to work with iBeacons bluetooth device ([link](https://www.npmjs.com/package/cordova-plugin-beacon)). This architecture gives a basic overview of the internal view of the application.

**1.2.1. Front End Technologies**

The FrontEnd is developed using HTML, CSS, JavaScript and AJAX and then built by Cordova to produce an apk that can be run on an android device.   
  
HTML – which is basically used to display all the information for the user.  
CSS – which is used to add styling including colors, box shapes, font sizes and making the application mobile friendly (usable on a mobile device).  
JavaScript – which is used to do the front-end application logic as well as place information on the right sections within the HTML document.  
AJAX – This is a crucial language that assists the front end to connect to the backend, send information to the backend files as well as receive information for javascript to place into the respective sections within the HTML document.  
  
  
 **1.2.2. Back End Technologies**  
There are two main backend programming languages used which include:  
PHP – This is the main programming language for the backend application logic, this enables the backend files to receive, process and send to storage any data depending on the function required.  
MySQL – This is basically used to create and manipulate the database. It is used in conjunction with PHP to enable data storage and retrieval.  
  
  
 **1.3. Tools and Implementation**Apache Cordova allows developers to build cross platform applications with the help of HTML5, CSS, and JavaScript.

The platform requirement are as follows:

* Android: Android SDK
* iOS: Swift and apple software development tools are required
* Windows: HTML5, JavaScript, CSS,
* Node.js for backend code,
* SQL for database.

We have decided to stick only with the Android version.

The user interacts with the front end which is developed using the frontend programming languages while the backend files created by the backend programming languages are uploaded to an online server where they can be accessible by the frontend. Each programming language plays its role as described below.

HTML and CSS are used to create the frontend skeleton structure and all its elements including the forms, buttons and all the sections of each page. The HTML and CSS files are separated but the CSS files are imported into the HTML files so that only the HTML files are directly accessible to the user.  
  
Ajax is then introduced into the HTML files to either send to or import data from the backend files. This is done through calling a link to the online backend file responsible for that specific page of the frontend. The link is formatted differently for each page depending on whether the frontend is to send data to the backend or just receive data.  
  
Javascript on the other hand is also placed into the HTML files to receive the data from AJAX do the necessary processing on the go then place each piece of information on the correct section of the frontend page this is also made possible since AJAX is merely an extension of JavaScript.   
  
At the backend, when a link is called by the AJAX from the frontend, the file stated in that link (which is a php file) must receive the data from the link, process it and return any relevant information in JSON format. JSON data format is very crucial because it allows the backend to effectively communicate to the frontend when the web app is converted into an android app using Cordova.  
  
At the backend, the php files need to return JSON data and therefore, after obtaining data from the database, the data is placed into an array and then converted into JSON. When the AJAX calls the link to a specific php file the php file returns only a JSON string. To make the app faster in loading JSON, each page has its own php file with its own JSON data specifically for that frontend HTML file.

**1.4. App Functions:**

The app consist of three main entry points for reaching the users’ goal. First the **student** registers on the mobile application, then logs in and continues adding and viewing their courses. Second component involves the **professor** to log in the app and add the courses during the semester. The third component is **iBeacons** (bluetooth enabled devices) that have not been implemented. The iBeacon should broadcast a message called “check-in” to the nearby devices in the range and the mobile device can check in to the course directly when clicking a button “Yes” in the mobile device. The mobile app maintains a list of all the checked-in students and then saves it to database.

**1.5. The scope of the project:**

The scope of the project was to implement a mobile app. The app based on Android OS is possible with Cordova CLI (command line interface), Java, Android SDK, Android devices to simulate the app.

**2. Use Cases**

Here are some examples of our use cases.

**2.1. “Add / Delete a Course” Use Case**

The professor’s preconditions:

- The app needs to be installed on the device,

- The professor needs to be registered and logged in,

- The professor needs to be the in charge of that course.

Scenario:

The professor registers and logs in the app. At the beginning of the semester, the professor adds courses and a basic description (which semester, type of lecture, how often, which classroom...). The professor is able to approve new students and check who attended the lectures throughout the semester (in-app calendar for each course).

The student’s preconditions:

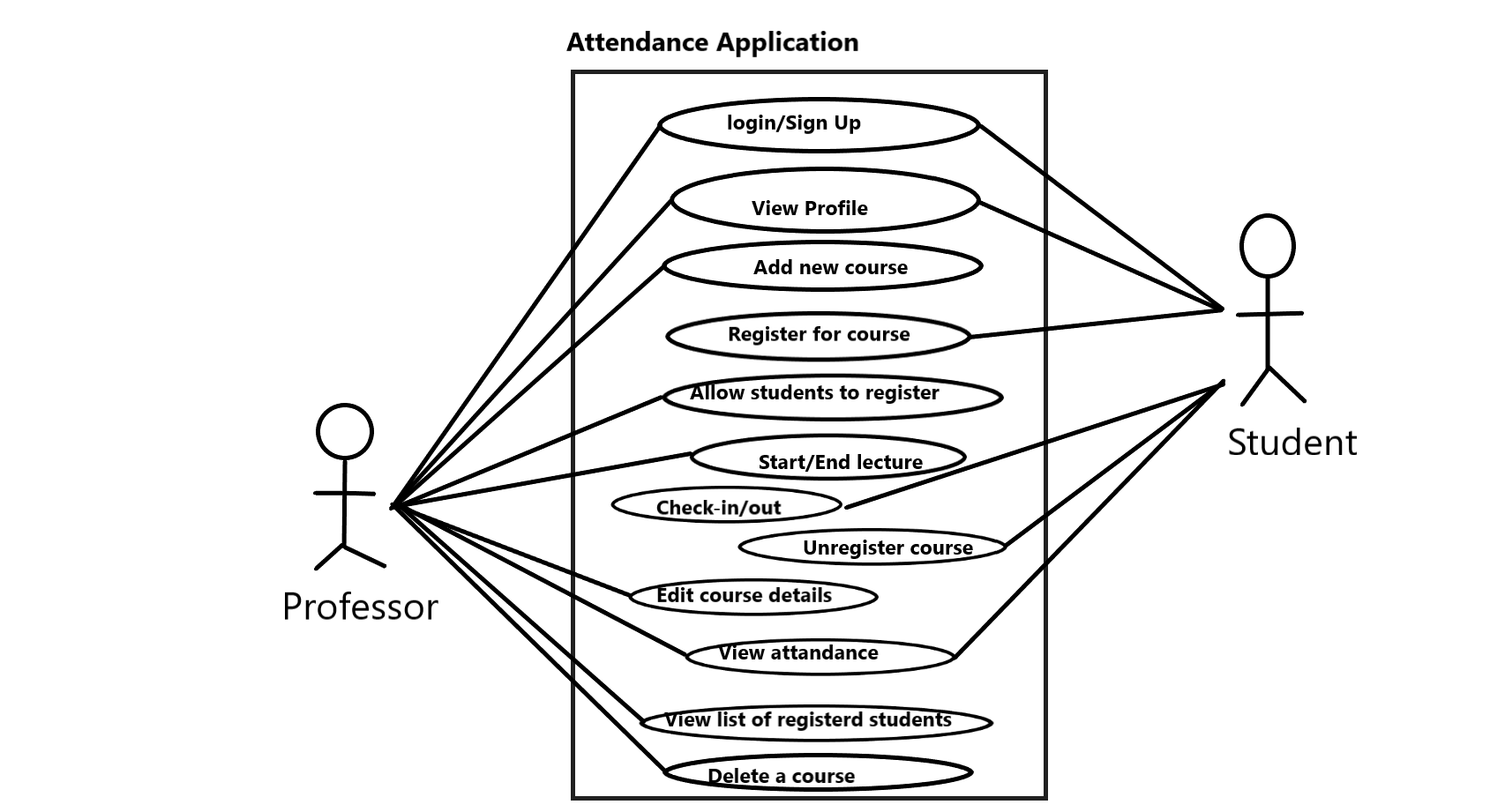
- The app needs to be installed on the device,

- The student needs to be registered and logged in,

- The student needs to be approved for that course.

Scenario:

The student registers and logs in the app. At the beginning of the semester, the student adds courses and gets approved. The student is able to check in and out of lectures and view their own attendance list throughout the semester (in-app calendar for each course).



*Picture 3. “Add/Delete a Course” Use-case*

**2.2. “Login / Register” Use Case**

* The Register Use Case

Scenario:

This use case starts when a system user is not logged in to the system and goes to the login page. The system prompts the user for a username and password or register new account. If the user selects the registration option, the system prompts the user for registration information, user name, password, etc.

* The Log in Use Case

Scenario:

The use case begins when the user types their name and password in the login form. The system validates the user’s password and logs them into the system. The system displays the main page (course overview).

**2.3. “Course Overview” Use Case**

The use case starts when the user logs in to the system and gets redirected to the main page (course overview). The user can then click the view button to view the attendance list. That means that the professor is able to view which students have attended the lectures, and the student is able to see when she/he has attended.

**3. Pro’s and Con’s**

Here are some of the pro’s and con’s that we have encountered while working on the project.

**3.1. Pro’s:**

* The application is easier to develop than using the native mobile app development languages,
* Easier to integrate a web app with an android app by use of Cordova,
* No paperwork or manual work for attendance,
* Professor can easily keep track of history of attendance and student records.

**3.2. Con’s:**

* Database maintenance needs to be done periodically and remove dead records is a big workload,
* No verification functionality available for fake users,
* Dependency on beacons or any other physical device technology.
* The mobile app tends to be slower to load information,
* Integrating mobile device hardware like Bluetooth or WiFi receivers into the app is challenging.

**3.3. How to make the app better**

After the experience in developing the application, it would be necessary to mention the following to enable future developers to better develop this app:

* Application works better when it does not have any dependency on other physical device (Estimote stickers). iBeacons are effective and efficient but there is a dependency when batteries are dead or if signal jam happens.
* Even using Estimotes Stickers (iBeacons) works on specific API. It is hard to write an independent API which works for every beacons.
* This application is limited to Android users only. It could be cross-platform and could work on web as well, so students can do check-in using laptops or tablets.
* A better database maintenance is required.
* This application is developed as activity centered and has no security level corrections. The application should maintain fault tolerance.
* It would be much better if the app is to be developed by native mobile app languages.
* Proper planning and feasibility and UX study should be carried out before undertaking the project.
* Members of the team should have a substantial skill sets in their areas of development. Increasing the number of teammates would be a good idea too.

**4. Literature**

|  |  |  |
| --- | --- | --- |
| 1 | Backend and Front-end | <https://www.upwork.com/hiring/development/a-beginners-guide-to-back-end-development/> |
| 2 | Cross platform app | <https://theappsolutions.com/blog/development/dev-cross-platform-apps/> |
| 3 | Research paper-1 | <https://www.ijarcce.com/upload/2017/april-17/IJARCCE%2085.pdf> |
| 4 | Cordova for beginners | <https://cordova.apache.org/docs/en/latest/guide/overview/> |
| 5 | Ibeacons getting started | <https://evothings.com/doc/examples/ibeacon-scan.html> |
| 6 | Cordova installation for Android | <https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html>  <https://evothings.com/doc/build/cordova-install-windows.html> |
| 7 | Selenium | <https://www.seleniumhq.org/> |
| 8 | XAMPP & MYSQL |  |
| 9 | Node JS | <https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm> |
| 10 | Beacons API | <https://github.com/futomi/node-beacon-scanner#BeaconScanner-startScan-method> |

1. https://stackoverflow.com/questions/26648378/phonegap-basic-architecture [↑](#footnote-ref-1)