OULUN YLIOPISTO
UNIVERSITY of OULU

PERSONALISED DISPLAYS

Stage 2: Implementation

AUTHORS

Juan Camilo García, 2418810
jgarciah@student.oulu.fi

Haejong Dong, 2292191
s2haejong@gmail.com

Henri Koski, 2190426
henri.koski@student.oulu.fi

Yifei Zuo, 2443937
yifei.zuo@student.oulu.fi

University of Oulu
Applied Computing Project II
29-02-2015

Table of Contents

Introduction

Purpose of the project

Public displays are starting to become more popular every day, but they still have a low interaction rate with the potential users. Most of its use is based on advertisement, which, although it's a revenue for the owner of the display, doesn't provide all that it could to its potential consumers. This types of public spaces are very powerful since they give interactivity and dynamic functionalities to people around it, but the downside is either that the information shown is very generic so that it's useful for everyone that approaches, or it displays too much data so that anyone can find the information they want. This situation limits the richness of the interaction as well as the possible uses of a public display.

Our purpose with this project is to provide more relevant information in public displays through automatic personalization of the content is shows by leveraging the fact that most users carry a smartphone with internet connectivity at all times. This will be achieved by mimicking a bulletin board and transforming it into its digital version which is not only smart but interactive and dynamic. By being digital, more organized layouts could exist compared to the current ones seen on Figure 1 and 2 and much more data could be displayed at the appropriate time while consuming less physical space. The organization is key, since bulletin boards have a big flaw which is that finding content on them is hard. Updates are not as frequent and certainly less visible when they happen, since changing one of the posters doesn't immediately trigger a response from the user.

Figure 1. Bulletin board on the Infinite Corridor at MIT (2004) [31].

Figure 2. A bulletin board from a Starbucks [32].

With this, we would like to find if users are more likely to interact in public spaces if the content is tailored for them when they approach as well as the pros or cons the digital counterpart of a bulletin has versus it. This also gives us the opportunity to explore personalization for groups instead of individuals, as well as study how much information are consumers willing to give in exchange of more tailored information.

System Description

Our system is composed of four parts, a sensing device which recognizes devices around it. A public display which is used by the users to send information to their phones. A server which has the personalization algorithm and a mobile application which displays the content the users query from the public display.

Assumptions and Constraints

- Mac Addresses of the mobile devices are unique. Although it is very improbable that two devices appear with the same Mac Address, it is possible, so we are assuming it is not possible.
- Mac Addresses do not change. It is a constraint we placed for this phase of the project, although it is solvable by storing the new Mac Address when it changes.
- The token identifier of each public display is static. This is a constraint we have so far, since developing a dynamic one would take too much time of the schedule we had.

Glossary

This section provides a brief explanation of the terms relating to the system design document and the project in general. The terms are alphabetically ordered.

Hypertext Markup Language (HTML)

Is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page [1].

Cascading Style Sheets (CSS)

Is a style sheets that describe how HTML elements are to be displayed on screen, paper, or in other media [2].

JavaScript

Is a programming language commonly used in web development [3].

Go Programming Language (golang)

Is an open source programming language that developed with the purpose of easy, reliable, and simple software building [4].

Application Program Interface (API)

Is a set of routines, protocols, and tools for building software applications [5].

Wi-Fi

Is a wireless networking technology that allows computers and other devices to communicate over a wireless signal [6].

Airmong-ng

Is a script that can be used to enable monitor mode on wireless interfaces [7].

Go-socket.io

Is an implementation of socket.io in golang, which is a real-time application Framework.

It is compatible with latest implementation of socket.io in node.js, and supports room and namespace [8].

PostgreSQL

Is an open source relational database management system (DBMS) developed by a worldwide team of volunteers. PostgreSQL is not controlled by any corporation or other private entity and the source code is available free of charge [9].

Transport Layer Security and Secure Sockets Layer (TLS/SSL)

Both of which are frequently referred to as 'SSL', are cryptographic protocols designed to provide communications security over a computer network [10].

Implementation process

To develop the whole system, we decided to follow a parallel development strategy [11], in which each of the four members was in charge of a specific part of the whole system. We started reducing the number of features taking time constraint into account, and then split the project in four modules. With this four modules we made a backend interface design to help with the future merging of each module, and then worked in parallel on each of the modules, as shown in Figure 3. After each module had its initial version, an integration test was done and an iterative cycle began by redesigning the backend interface to add features we had missed on the initial cycle. Finally after several iterations, a Demo Release was developed and the next phase is an initial evaluation to adjust final problems of the system.

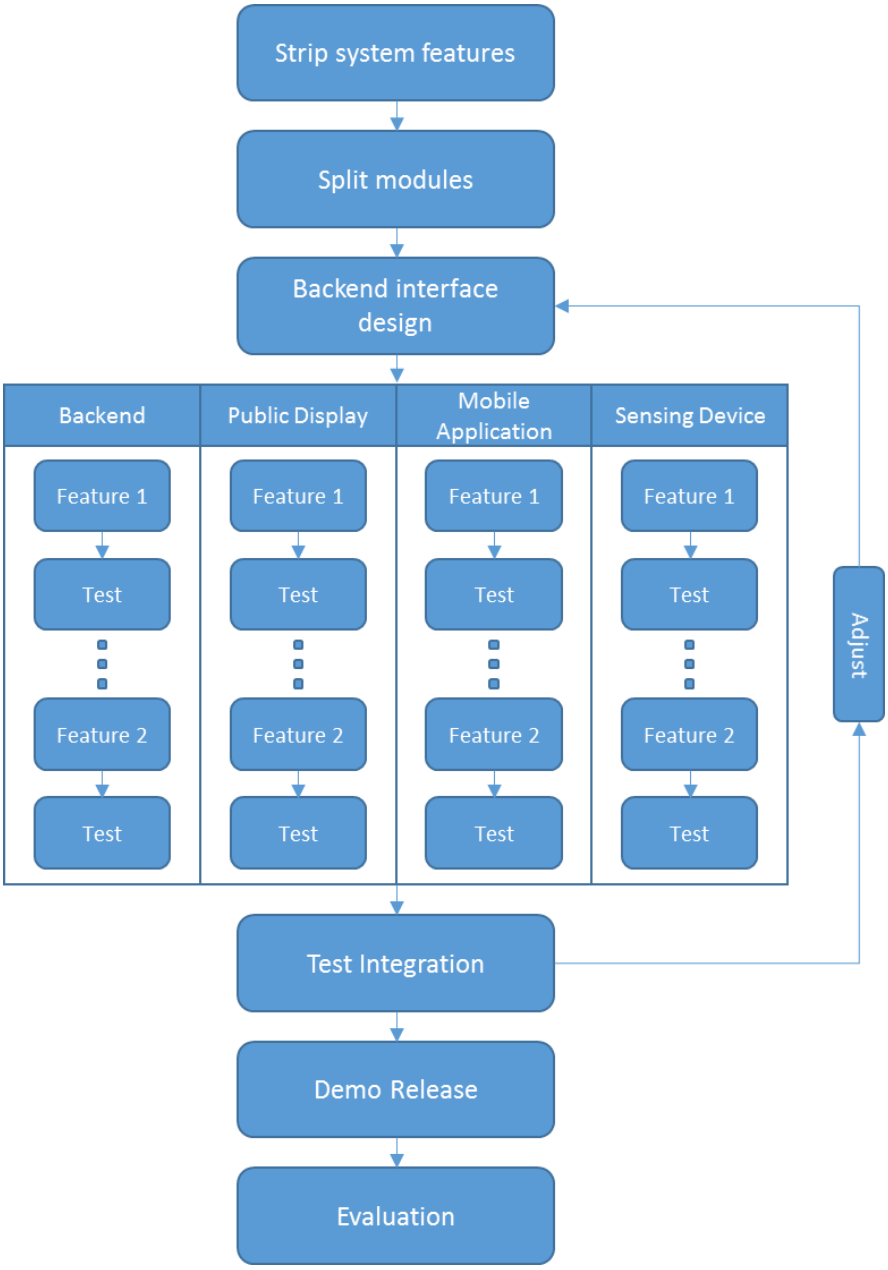

Figure 3. Implementation process flow diagram.

Features kept:

- Device recognition through WiFi
- Possibility to send information from public display to the phone
- Possibility to view information on the public display
- Information shown in public display:
    - On-going courses
    - Job offers
    - Events
    - Advertisement
    - Personal events
        - Facebook
        - Calendar
- Dashboard to upload events to the database
- Possibility of multi display system using display tokens.

Features removed:

- Custom map of the University
- Special UI for each type of category on the phone
- Navigation directions on the public display
- Arrows on the public display showing the direction of a specific place
- Support for dynamic Mac Addresses

Software Architecture

Figure 4 shows the connection between the parts of the system. Basically, the public display sense devices around it, information which it sends to the server. When something is tapped on the screen, the event is sent to the server, which decides to which mobile device he has to send the information too. From the mobile device side, it registers its Mac Address with the server and sends calendar and Facebook events to the server. Aside from this, it can also register to certain topics which then are used to personalise what's visible on the public display. The server has an algorithm which will be explained later, that decides what content each public display must show.

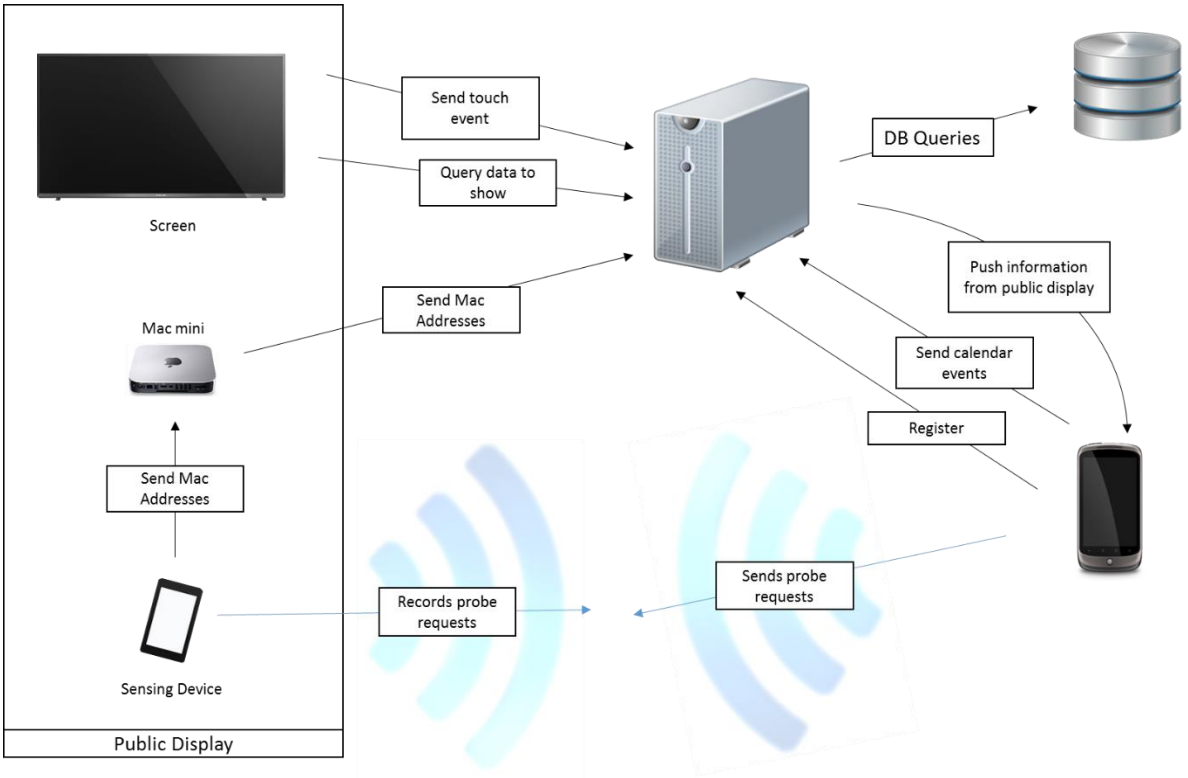

Figure 4. System architecture diagram.

The architecture is comprised by 3 parts, the Backend, the public display, and the mobile application. The Public Display is divided between Front End (UI) and Back End (device sensing capabilities). In Appendix 1 the number of lines of code and their languages can be found.

Server

Table 1 shows the services the server provides. All of the services are over SSL to ensure security. A password protected admin panel was also created to administer the database. All the backend was developed using GoLang [4].

Server		
Resource	Type	Description
For every part		
/categories	GET	Get the possible categories of an item
For Public Displays		
/display/events	POST	Request the items to display on a specific display
/display/notify	POST	Send Item to the nearest device to a specific display
For Mobile client		
/client/adduser	POST	Add a user
/client/user/facebook	POST	Add facebook id for user

/client/user/addmac	POST	Add mac address for user
/client/user/google/token	POST	Add Google Notification token for user
/client/user/addevent	POST	Add an event for user
/client/user/topic/register	POST	Register the user to a category
/client/user/topic/unregister	POST	Unregister the user from a category
For Sensing Device		
/sensor/devices	POST	Receive list of Mac addresses, with signal strength and last time seen on a specific display
For Admin Panel		
/admin/:token/valid	GET	Manage database data
/admin/:token/category	POST	
/admin/:token/category	DELETE	
/admin/:token/user/events	GET	
/admin/:token/user/event	POST	
/admin/:token/user/event	DELETE	
/admin/:token/user/categories	GET	
/admin/:token/user/category	POST	
/admin/:token/user/category	DELETE	
/admin/:token/event/categories	GET	
/admin/:token/event/category	POST	
/admin/:token/event/category	DELETE	
/admin/:token/user/macs	GET	
/admin/:token/user/mac	POST	
/admin/:token/user/mac	DELETE	
/admin/:token/user/token	POST	
/admin/:token/users	GET	
/admin/:token/user	POST	
/admin/:token/user	DELETE	
/admin/:token/events	GET	
/admin/:token/event	POST	
/admin/:token/event	DELETE	
/admin/:token/displays	GET	
/admin/:token/display	POST	
/admin/:token/display	DELETE	
/admin/:token/push	POST	

Table 3. Web services available.

The server is in charge of the personalisation algorithm for each public display. The public display queries for each category of items possible (Bubble, Job offer, Advertisement, Courses and Events) with the limit of events it can display. Meanwhile, the sensing device sends the devices seen in the last 2

minutes for a public display. The server then filters all the events of the category requested by the public display. Since each event has several topics to which users are registered, the server adds 1 to the score of each item for each visible user in that public display that is registered to the topic of the item. Then the items are ordered by highest to lowest on this score. Depending on the number of items requested by the public display, the server limits the response to the highest scores. In case of equal scores, the server chooses randomly which ones to return.

Public Display

Front End

The front end of the public display was developed as a web application. It makes queries every 5 seconds to get the stream of content it has to show. There are 5 main categories which it can query, Bubble (represents user personal events), Courses (Ongoing), Job opportunities, Adverts and Events. It queries for each of them since it should display information for all of them all the time.

It also permits the user to send information to his device by communicating the server than an event has been tapped. Figure 5 shows the general architecture.

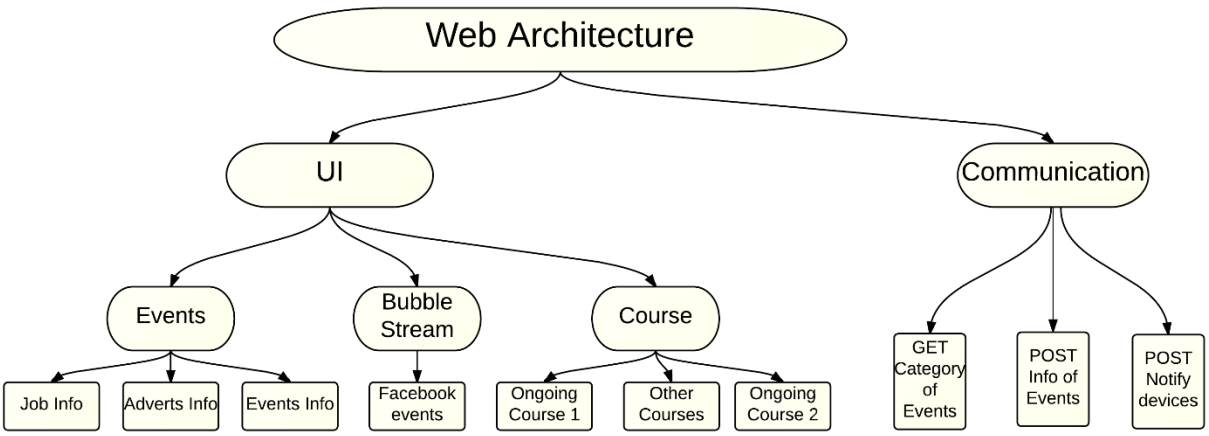

Figure 5. Web architecture.

The Web class diagram is shown in Figure 6.

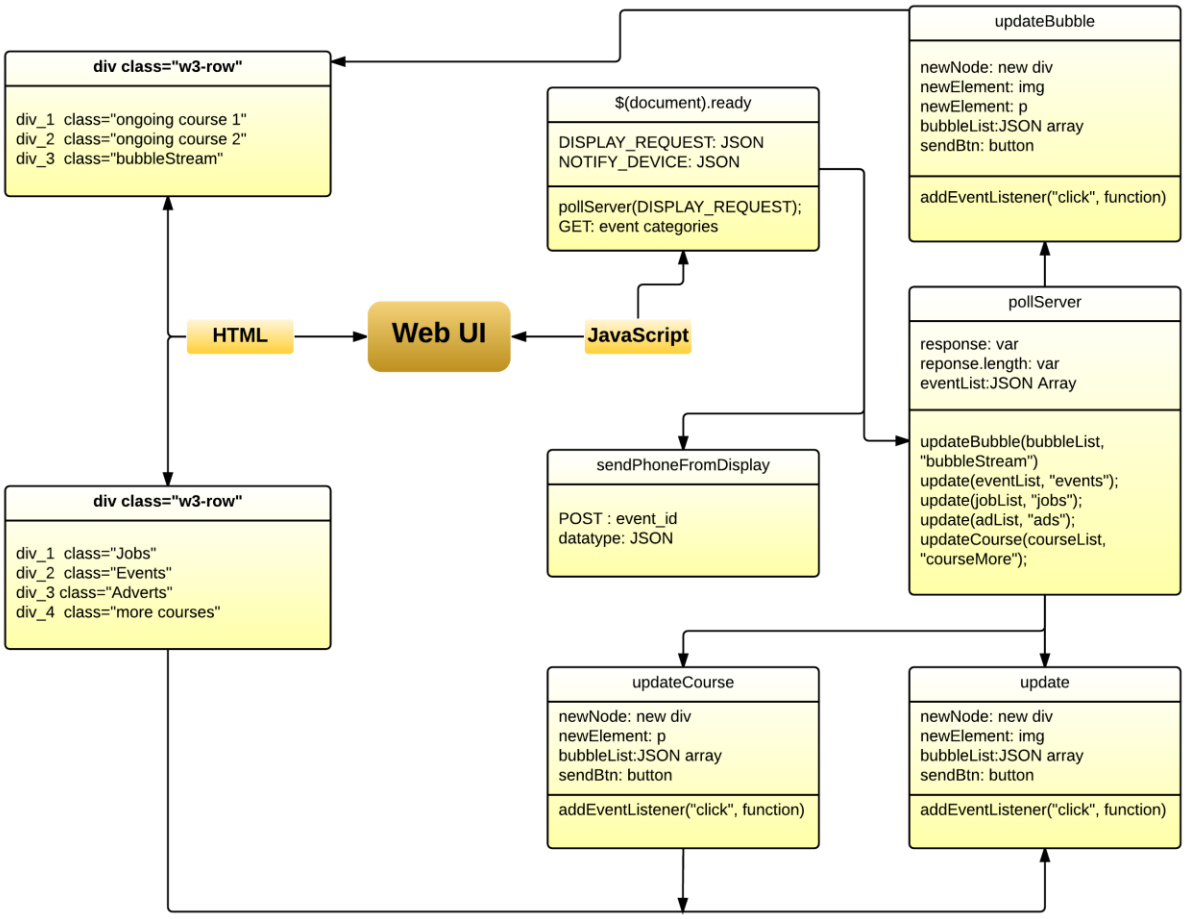

Figure 6. Web class diagram.

Back End

The back end of the public display is its device sensing capabilities. To achieve this, we developed an application for a rooted tablet, which turns the WiFi Chipset to monitor mode, using the custom kernel provided by bcmon [12] [13], which enables aircrack-ng [7] on an android device, with a specific WiFi chipset. In Figure 7, the workflow of the sensing device is shown.

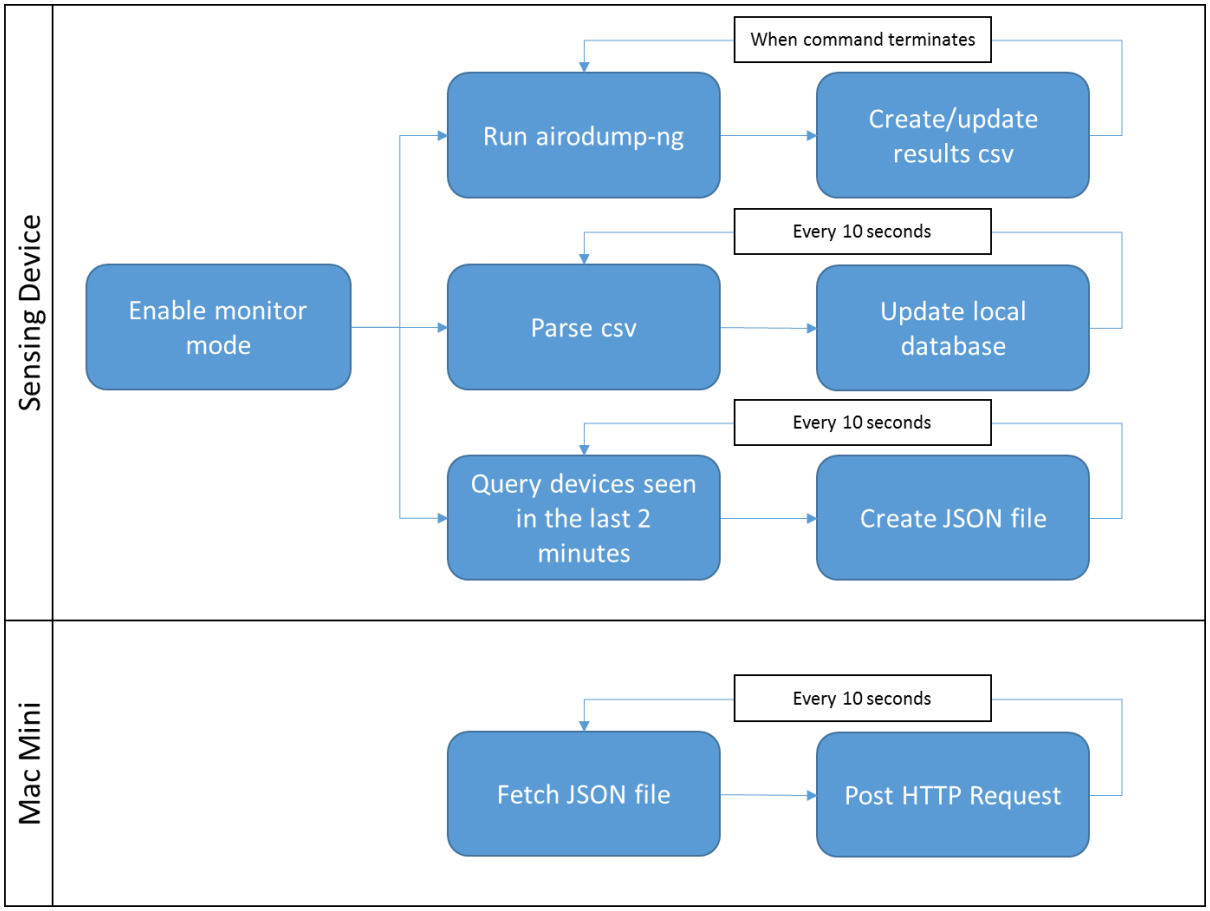

Figure 7. Sensing Device workflow.

With the WiFi Chipset in monitor mode, we read the probe requests done by devices to AP's around and store this information to a local database. A JSON file is created containing the devices seen in the last 2 minutes. The file includes their Mac Address, signal strength and last time seen.

Since the WiFi capabilities to connect to an AP for internet connection is impossible due to the monitor mode being on, a script was created for the Mac Mini, device attached to the Public Displays and the Sensing Device, which fetches the JSON file stored inside the Sensing Device and sends the information to the server using the appropriate http request.

Mobile Application

In Table 2, the different modules of the Mobile application can be seen.

Mobile Application	
User registration	It is http POST request to register mobile user to the web server. This module operated only once when user install and run the application first time, and this includes basic info of users such as name, facebook id, email and so forth.
Facebook login / logout	This module takes action if user clicked facebook login button and provided credential. In

	consequence, application would then be able to retrieve user's info from facebook by the privacy level of which the user consented when logging in.
Categories List acquisition service	This module is persistent, running in the background sending http GET request to the server by the predefined time of interval. This will continuously observe on the available list of categories that the system supports.
Calendar service	This module runs in the background however is only prompted if any changes in user's calendar schedule has been made. It posts request to the server with the data of user's calendar events.
Http request manager	It maintains all the http connection methods with the server.
Http request service	It is in responsible of http connections that require persistent operation.
Google map manager	This module provide an access to the google map plugin when user searches for the location of any events.
GCM(Google cloud messaging) [14] registration service	It issues google token for receiving push message from the server. This module is also in connection with the GCM server in which it gets actual messages from. Leveraging pubsub feature it can also subscribe to any particular broadcast messages.
GCM listener service	This is module where the messages sent from server achieved, and application take off further processes such as generating notification of message arrival.
GCM instance listener service	It is in charge of receiving notification for refreshed google token, this will notify the application of newly issued token.
Database manager	Self-explanatory module, which maintains data queries that are locally stored.
Category manager	This module maintains list of categories and provide needed methods for handling categories actions such as added/removed category, updated list of categories.
Search bar manager	This allows mobile user to search through the list of categories available and supported by the server.
Shared preference	It contains a chuck of data collected throughout the application system but not necessarily to create a table in the database thus, are stored in the device with keys.
Add category button	This button creates other buttons of categories with that of titles so user can recognize what are added in the interest.

Table2. Mobile client modules.

Figure 8 shows the flow chart of the Mobile Client.

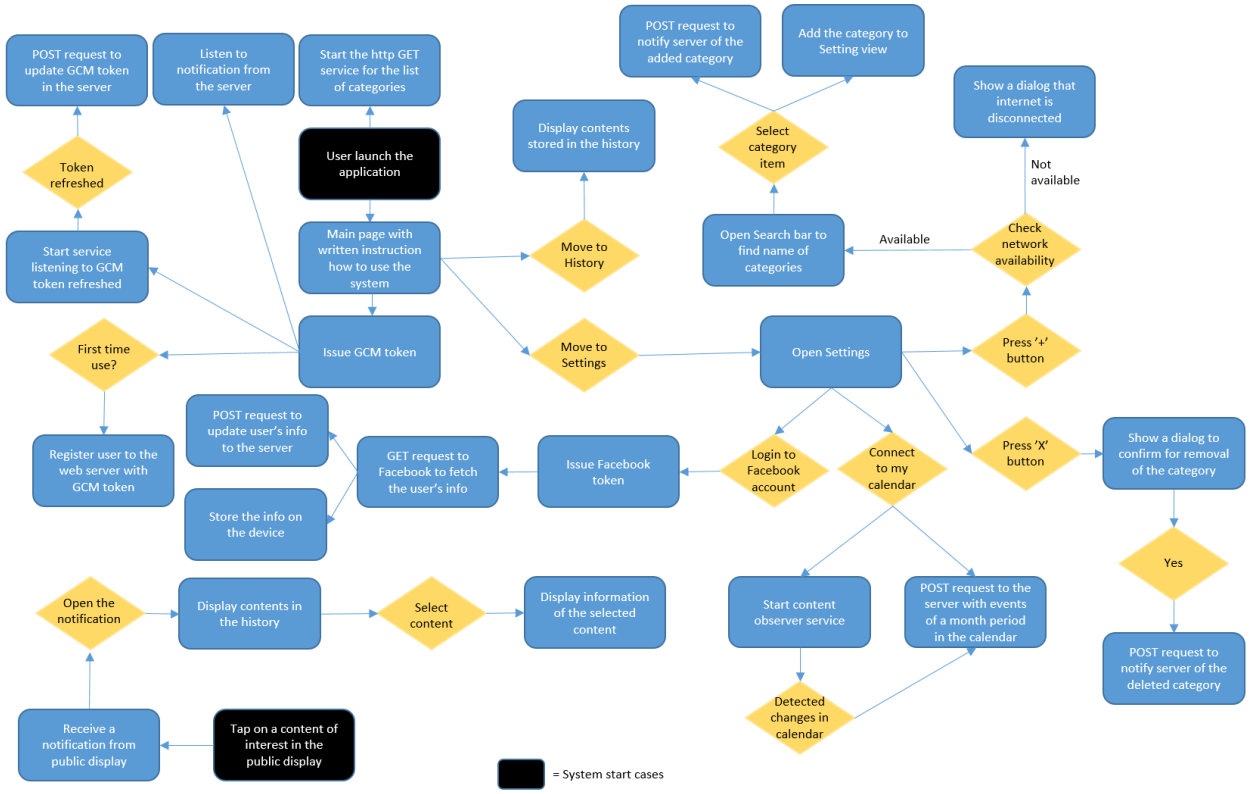

Figure 8. Mobile client flow chart.

Data Structures

Server

Figure 9 shows the database schema used for the server database.

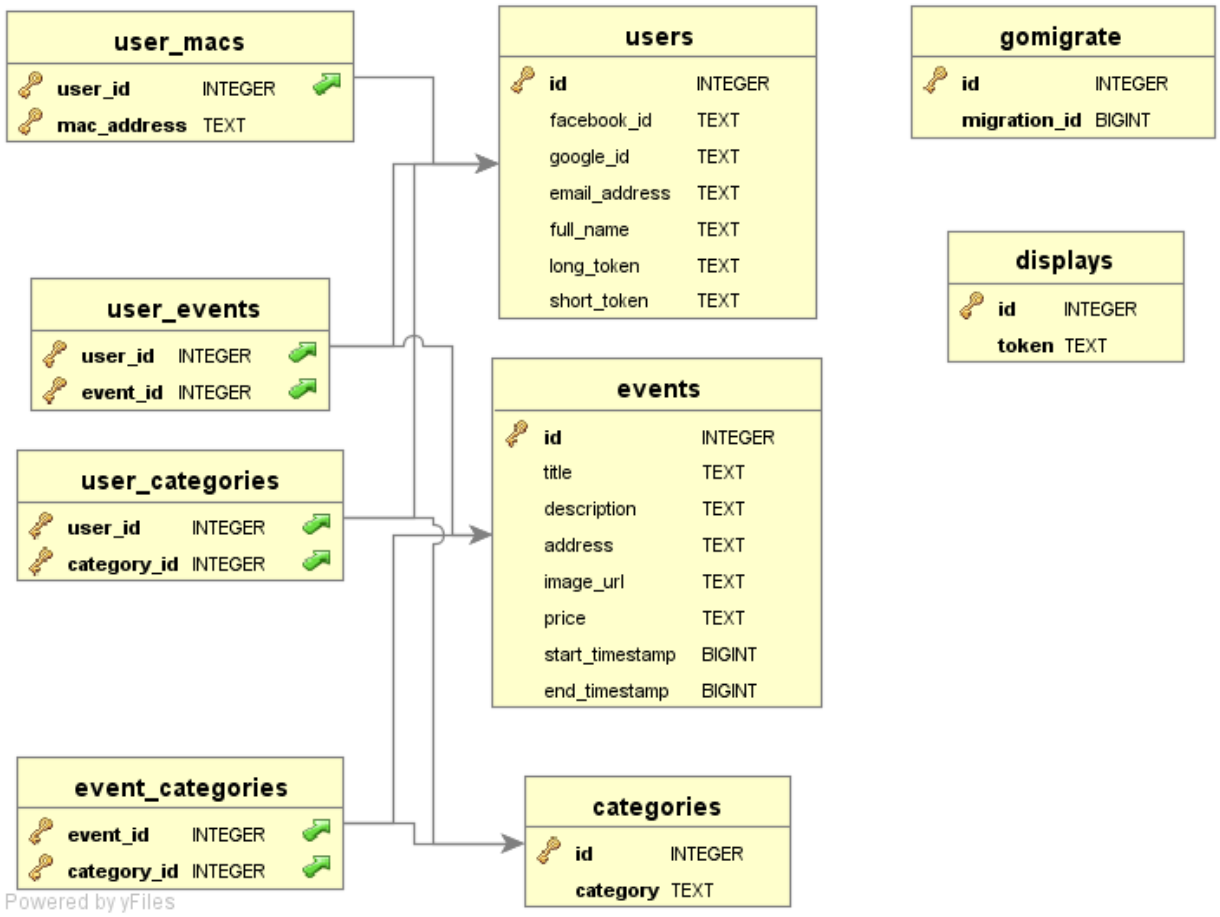

Figure 9. Server database schema.

Mobile

In Figure 10 is the Database structure in the Mobile Application.

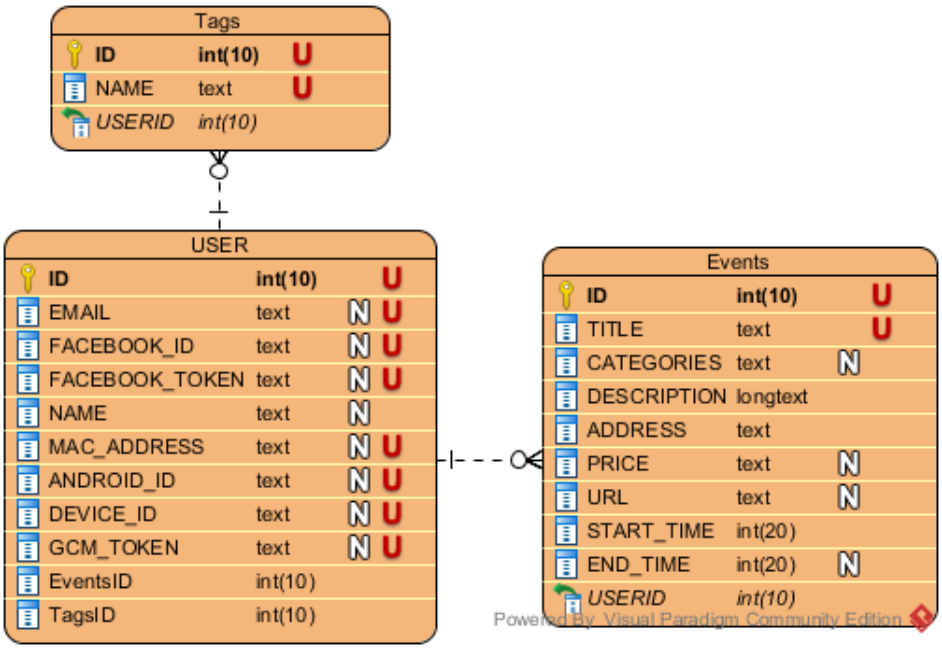

Figure 40. Mobile client database schema.

Hardware Documentation

The only very hardware dependent part of the system is the sensing device. It has to be a piece of hardware that has a WiFi Chipset that can be turned into monitor mode. On this occasion, an Andriod device was used with Aircrack-ng, a network software to sniff WiFi packets. Since this is not supported natively by Android, a custom kernel was needed. The kernel developed by Ruby Feinstein and Omri Ildis [12] [13] was used. This kernel only works on WiFi Chipsets BCM4329 and BCM4330. Because of this, a Nexus 7 (2012) which has Chipset BCM4330B2 was used. The OS on it was Cyanogen 10.1 and it was rooted to be able to access the monitor mode features.

The public display is built to work on any browser, but specifically tested in Firefox Version 44, the one that runs on the Public Displays we are deploying our system on. This public displays have a Mac Mini attached to them which also needs an active internet connection.

For the mobile application, any android device with API 17 and up (Android 4.2.2+) is supported (covering 86% of current android devices [15]), and an active internet connection is required to be able to register and unregister from topics, as well as receive new items from the public display.

For the backend, our system is on an Virtual Machine running Ubuntu Server 14.04 LTS (Trusty Tahr) with 1 core, 1GB Ram and 30GB SSD. It is located in Frankfurt and hosted by UpCloud [16]. All the web services are using a TLS certificate from Let's Encrypt [17].

User Interface

The interaction flow chart is shown as figure 11. From the web homepage, the users can get information of jobs, event, adverts, courses, bubble event and ongoing courses.

Except for the static information of ongoing courses, users could get more detailed information by tapping on the images or text. They could decide whether to download the information to their mobile phone by choosing “send to phone”, or “close” to close the window and go back to homepage. If the button of “send to phone” is triggered, there will be an alert to tell whether info has been sent successfully.

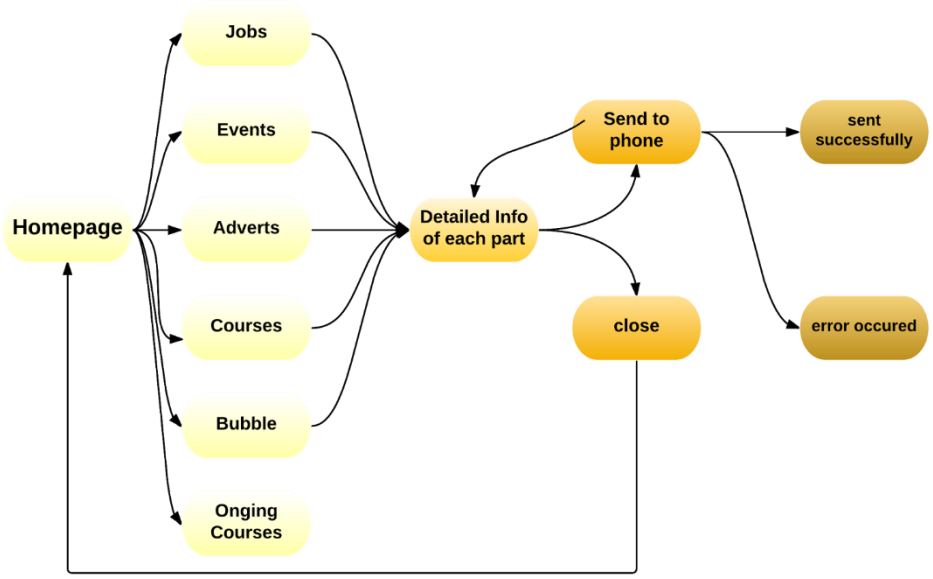

Figure 50. Web interaction flow chart.

Screenshots

In Figure 11, the web homepage provides users a clear picture of the web pages designed for them in a personalised way. Ongoing course box 1 and 2 shows location, course name and exact time of a course. The blue bubble provides information of Facebook events. Jobs, events, adverts and courses with images show users useful information they might be interested in.

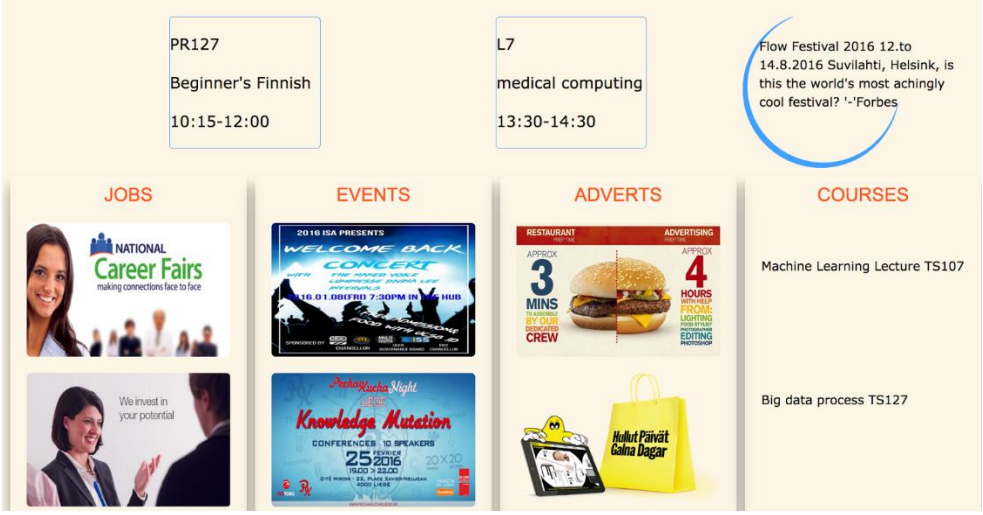

Figure 61. Web Homepage.

Figure 12 shows the bubble screenshot, which shows an example of the full information of a Facebook event. Two buttons: send to phone means after tapping, the whole info will be sent to the user's mobile phone, and if users don't want to do this, they can cancel it.

Figure 72. Web UI Bubble Content of Facebook Events.

After "send to Phone" is triggered, an alert window shows the state of whether sending has been done successfully. Figures 13 to 15 show examples of a job offer, an event and an advertisement respectively.

Figure 83. Web UI Jobs info.

Figure 94. Web UI Event info.

Figure 105. Web UI Advertisement info.

Mobile Client

In Figures 16 to 18 some screenshots of the mobile application are shown.

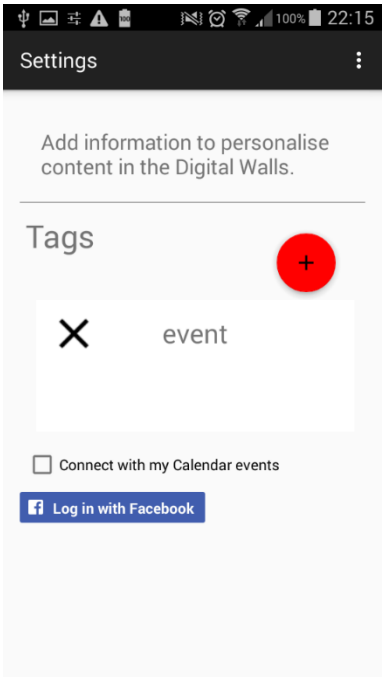

Figure 16. Settings screen.

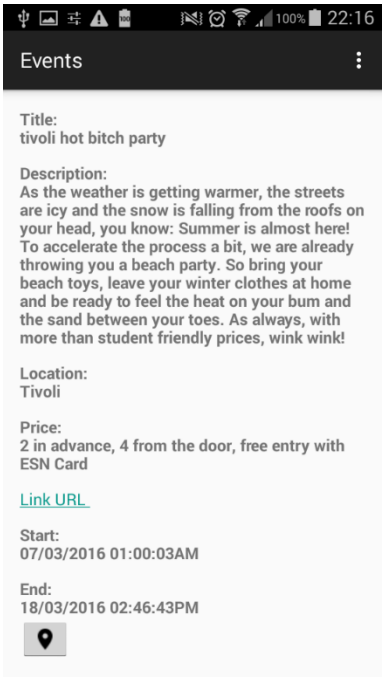

Figure 17. Event screen.

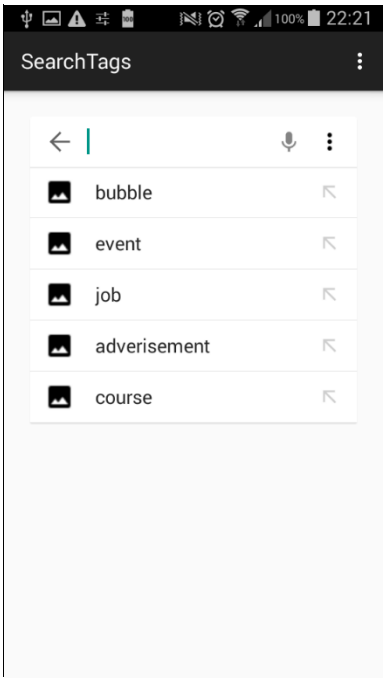

Figure 18. Register topic screen.

Tools

- w3.css: W3.CSS is a small, fast, and modern CSS framework which could speed up mobile HTML apps and provide CSS equality for all devices: PC, laptop, tablet, and mobile.
- jquery: Jquery is a JavaScript Library aiming at simplifying JavaScript programming.
- ajax: Ajax contributes the communication of web interface and the backend.

Design Guidelines

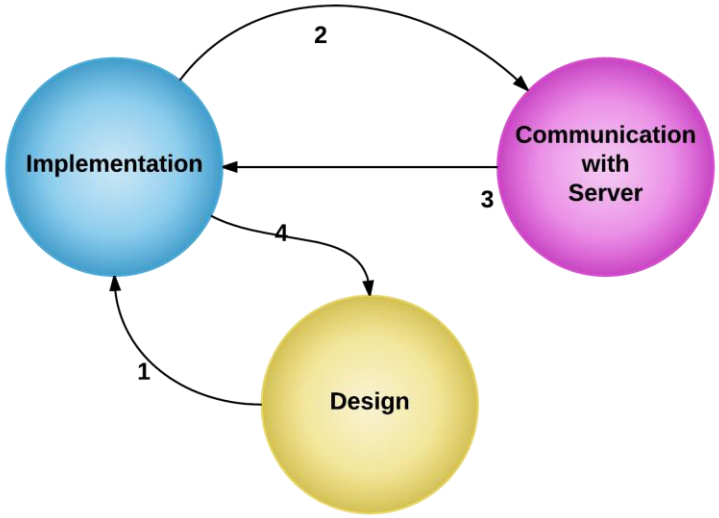

Figure 19. Design guidelines iterations.

In Figure 19, the Web UI design iteration process is shown. It follows the iterative design procedures, 1->4->1->2->3->4->1->2, and so on. Starting from the design phase, we tried to make an offline version of the Web UI to test whether the design is good or not, and then we made the web page communicate with the backend and retrieve data. With the real data, we improved the implementation and design to change colours, fonts and styles to make it more interactive and attractive.

Third Party Materials

Name	Description
GoMigrate [18]	Database Management
Bcmon [12] [13]	WiFi Monitor Mode kernel
Airmong-ng [7]	Script for Monitor Mode usage
Google Cloud Messaging [14]	Push notifications from server to mobile devices
W3.css [2]	Public Display UI
jQuery [19]	Public Display (Ajax) and Admin Panel
Bulma.io [20]	CSS framework for the Admin Panel
Httprouter [21]	Used as http multiplexer to handle each request in separate goroutine.

PostgreSQL [22]	PostgreSQL driver
Gorp.v1 [23]	ORM library
Null.v3 [24]	Library to handle optional null fields in database
Facebook-go-sdk [25]	small facebook wrapper for golang
Simple-cors [26]	simple cors handler
Okhttp [27]	Android library for http requests
gson [28]	Android library to manage json
GitHub [29]	Platform for source control while developing

Table 3. Third party materials used.

Security and Privacy

With regards to security, the connections between the admin panel, the display and the sensing devices to the server need an access token identifying the display. The connection between the mobile client and the server need the client's unique id, except when creating a new user. At the moment, the user's id is an auto incrementing id, but it will be changed to UUID v5 (standard RFC4122) [30].

All the connections are encrypted, which means the tokens can't be captured from the network. It also means, the private data sent from the mobile device to the server (personal events) is secure.

Risk assessment

This chapter contains brief description, likelihood, impact, and actions for possible risk we might encounter during the project. The risks are quite general and cover mostly the abstract for one type of risk.

Communication problems

If the team members fail to communicate with each other as much as is needed or the team fails to communicate with TA the course of project might shift and people end up doing wrong tasks. For our group this is unlikely and because of that it impacts would be minor and easily corrected. To prevent this team needs to schedule enough meetings.

Technical risks

This kind of risks usually occur when product is complex and modules are hard to integrate. These risks might lead to failure in functionality or in performance and the likelihood of it happening in this project is unlikely but impact of it might be major if not handled right away. This can be prevented with good and thorough system architecture and technological choices. When facing this kind of risk it is crucial to identify the risk as soon as possible and make the necessary changes immediately while the impact is still minor.

User acquisition risk

This kind of risk may occur if the users don't have an appealing for the system designed. It is a common risk but has a minor impact since it's easy to solve. The solution would be to offer a reward in exchange of using the system to be able to evaluate it.

Hardware failure

This kind of risk can occur in all the parts of the system. Each of them being catastrophic impacts but low likelihood. Risks that fall in this category can be Public display touch screen unresponsiveness. If this happens, a replacement of the hardware would be needed.

User overload

This risk can happen if too many users start using our system. It has a low likelihood since most of our http requests are already on a high frequency by the public displays and the sensing device, so a lot of users would be necessary to make the server crash. Its impact would be major since some downtime on the system would occur, but since the server is a virtual machine, more capacity could be added easily and start it again.

Fault sensing risks

This project has utilized WiFi signal to identify user who tapped on the content in public display by measuring the strength of WiFi signal of devices around. This may turn out inaccurate and lead to a serious privacy failure by spreading personal contents to someone else nearby. To avoid such consequences, we have developed an idea of creating a security layer in which user has to prove themselves of the right person of the content when receiving it from the server.

Network failure

On the mobile client if network connection is lost, the application is abandoned and no longer can provide expected services except ones locally stored. This also can cause a bug which the application added categories of interest without notifying the server, with current software architecture of the mobile client, we could not prevent this but over the evaluation phase this can be surely fixed with time.

References

[1] R. Margaret, “HTML (Hypertext Markup Language) definition,” September 2005. [Online]. Available: http://searchsoa.techtarget.com/definition/HTML.

[2] w3schools, “CSS Introduction,” [Online]. Available: http://www.w3schools.com/css/css_intro.asp.

[3] P. Christensson, “JavaScript Definition,” 8 August 2014. [Online]. Available: http://techterms.com.

[4] golang, “The Go Programming Language,” [Online]. Available: https://golang.org/.

[5] B. Vangie, “API - application program interface,” [Online]. Available: http://www.webopedia.com/TERM/A/API.html.

[6] P. Christensson, “Wi-Fi Definition.,” March 2014. [Online]. Available: http://techterms.com.

[7] Aircrack-ng, “Airmon-ng,” [Online]. Available: http://www.aircrack-ng.org/doku.php?id=airmon-ng.

[8] GoDoc, “Package socketio,” [Online]. Available: https://godoc.org/github.com/googollee/go-socket.io.

[9] R. Margaret, “PostgreSQL,” May 2006. [Online]. Available: http://whatis.techtarget.com/definition/PostgreSQL.

[10] E. R. T. Dierks, "The Transport Layer Security (TLS) Protocol, Version 1.2", 2008.

[11] T. Bret, “Parallel Development Strategies for Software Congifuration Management,” Congluence Systems Ltd, 2004. [Online]. Available: http://www.methodsandtools.com/archive/archive.php?id=12.

[12] R. Feinstein and O. Ildis, “Wardriving from your pocket: Using Wireshark to Reverse Engineer Broadcom WiFi chipsets,” in *RECON*, Montreal, 2013.

[13] R. Feinstein, O. Ildis and Y. Ofir, “Monitor mode for Broadcom WiFi Chipsets,” 14 7 2013. [Online]. Available: http://bcmon.blogspot.fi/2013/07/monitor-mode-reloaded_14.html.

[14] Google, “Google Cloud Messaging,” [Online]. Available: https://developers.google.com/cloud-messaging/.

[15] Google, “Android OS Distribution,” [Online]. Available: http://developer.android.com/about/dashboards/index.html#Platform. [Accessed 2 2016].

[16] UpCloud, “UpCloud,” [Online]. Available: https://www.upcloud.com/.

[17] Let's Encrypt, “Let's Encrypt,” [Online]. Available: https://letsencrypt.org/.

[18] D. Huie, “gomigrate,” [Online]. Available: https://github.com/DavidHuie/gomigrate.

[19] jQuery, “jQuery,” [Online]. Available: https://jquery.com/.

[20] BULMA, “Bulma,” [Online]. Available: http://bulma.io/.

[21] julienschmidt, “httprouter,” [Online]. Available: https://github.com/julienschmidt/httprouter.

[22] “pq,” [Online]. Available: https://github.com/lib/pq.

[23] “gopkg.in/gorp.v1,” [Online]. Available: http://gopkg.in/gorp.v1.

[24] “gopkg.in/guregu/null.v3,” [Online]. Available: http://gopkg.in/guregu/null.v3.

[25] heppu, “facebook-go-sdk,” [Online]. Available: https://github.com/heppu/facebook-go-sdk.

[26] heppu, “simple-cors,” [Online]. Available: https://github.com/heppu/simple-cors.

[27] OkHttp, “OkHttp,” [Online]. Available: http://square.github.io/okhttp/.

[28] Google, “gson,” [Online]. Available: https://github.com/google/gson.

[29] GitHub, “GitHub,” [Online]. Available: https://github.com/about.

[30] “A universally Unique IDentifier (UUID) URN Namespace,” [Online]. Available: https://tools.ietf.org/html/rfc4122.

[31] “Bulletin Board,” 10 11 2015. [Online]. Available: https://en.wikipedia.org/wiki/Bulletin_board.

[32] “Bulletin Board @ 15th Ave Coffee & Tea,” 6 11 1009. [Online]. Available: https://www.flickr.com/photos/gumption/4083521070.

[33] “Usabilityfirst. introduction to user-Centered Design,” [Online]. Available: http://www.usabilityfirst.com/about-usability/introduction-to-user-centered-design/.

[34] NGINX, “Welcome to NGINX Wiki’s documentation,” [Online]. Available: https://www.nginx.com/resources/wiki/.

[35] R. Margaret, “Apache definition,” September 2005. [Online]. Available: http://searchsoa.techtarget.com/definition/Apache .

[36] P. Christensson, “Bluetooth Definition,” 2006. [Online]. Available: http://techterms.com.

[37] T. Kevin, “GATT.,” March 2014. [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt .

[38] R. Margaret, “PayPal definition,” [Online]. Available: http://searchsoa.techtarget.com/definition/PayPal.

[39] R. Margaret, “Redis,” April 2013. [Online]. Available: http://whatis.techtarget.com/definition/Redis.

[40] gorillatoolkit, “Package mux,” [Online]. Available: http://www.gorillatoolkit.org/pkg/mux.

[41] P. Christensson, “HTTP Definition,” May 2015. [Online]. Available: http://techterms.com.

Contributions

Date	Start time	End time	Total time (hours)	Description	Yifei	Haejong	Henri	Camilo
8/2/2015	14:00	16:00	2	Device sensing research				X
10/2/2015	16:00	18:00	2	Device sensing research				X
12/2/2015	18:00	20:00	2	Device sensing research				X
6/2/2016	10:00	22:00	12	coding be			X	
7/2/2016	14:00	22:00	8	coding be			X	
8/2/2016	20:00	21:04	1.066666667	Web UI	X			
8/2/2016	19:00	24:00:00	5	coding be			X	
9/2/2016	17:00	18:32:00	1.533333333	Web UI	X			
11/2/2016	14:08	14:40	0.5333333333	Goup Meeting	X	X		X
11/2/2016	14:00	21:45	7.75	coding be			X	
12/1/2016	14:00	15:00	1	Meeting	X	X		X
12/1/2016	17:20	17:35	0.25	Online meeting			X	X
13/2/2016	13:00	20:30	7.5	coding be			X	
14/01/2016	14:00	16:10	2.166666667	Device Sensing module				X
14/01/2016	16:55	17:35	0.6666666667	Device Sensing module				X
14/2/2016	16:00	21:30	5.5	Coding be			X	
15/01/2016	15:30	17:35	2.083333333	Web UI	X			
16/2/2016	14:08	16:54	2.766666667	Web UI	X			
17/02/2016	10:46	12:05	1.316666667	Web UI	X			
17/02/2016	16:00	16:45	0.75	Web UI	X			
17/2/2016	10:00	13:25	3.416666667	coding		X		
17/2/2016	15:00	20:00	5	coding		X		
18/2/2015	20:00:00	23:59	3.983333333	Device sensing research				X
18/2/2016	10:00	12:00	2	coding		X		
18/2/2016	13:30	17:50	4.333333333	coding		X		
19/2/2015	19:25	23:30	4.083333333	Device sensing developement				X
19/2/2016	16:00	17:48	1.8	Goup Meeting	X	X	X	X
19/2/2016	15:00	22:00	7	Coding be			X	

19/2/2016	7:25	11:30	4.083333333	coding		X		
19/2/2016	13:00	15:35	2.583333333	coding		X		
20/2/2015	0:10	5:00	4.833333333	Device sensing developement				X
20/2/2016	13:00	19:00:00	6	Coding be				
20/2/2016	9:45	11:40	1.916666667	coding		X		
20/2/2016	14:30	21:00	6.5	coding		X		
21/2/2015	9:00	14:00	5	Device sensing developement				X
21/2/2016	17:10	9:00:00 PM	3.833333333	Web UI	X			
21/2/2016	9:15	12:15	3	coding		X		
21/2/2016	13:20	19:00	5.666666667	coding		X		
22/02/2016	16:00	18:54	2.9	Group Meeting	X	X	X	X
22/2/2015	15:00	21:00:00	6	Device sensing developement				X
22/2/2015	21:23	23:45	2.35	Device sensing developement				X
22/2/2015	14:00	21:00	7	coding		X		
22/2/2016	10:15	13:00	2.75	coding		X		
23/02/2016	10:20	11:00	0.6666666667	traced from slack	X			
23/02/2016	15:46	19:56	4.166666667	traced from slack	X			
24/02/2016	10:43	15:04	4.35	traced from slack	X			
24/02/2016	21:13	21:23	0.1666666667	traced from slack	X			
25/02/2016	9:58	19:19	9.35	traced from slack	X			
26/02/2016	11:17	14:44	3.45	traced from slack	X			
27/02/2016	14:45	17:45	3	traced from slack	X			
27/2/2015	15:00	20:00	5	coding		X		
28/02/2016	11:00	19:56	8.933333333	traced from slack	X			
29/02/2016	10:03	19:01	8.966666667	graph drawing	X			
29/02/2016	19:40	22:50	3.166666667	report writting	X			
29/2/2015	13:00	22:00	9	coding, writing report		X		
29/2/2016	19:00	0:00	5	db schema, raport related stuff			X	
29/2/2016	12:00	23:59	11.98333333	report writting				X
30/01/2016	12:01	13:05	1.066666667	Web UI	X			

Total Hours					67	68	63	54

Appendixes

Appendix 1

Device sensing				
Language	**Files**	**Blank**	**Comments**	**Code**
Bourne Shell	2	0	0	12
Java	6	103	18	554
XML	14	8	6	259
Bourne Again Shell	1	19	20	121
DOS Batch	1	24	2	64
Groovy	4	7	3	43
Prolog	1	2	0	15
Total	**29**	**163**	**49**	**1068**

Mobile Client				
Language	**Files**	**Blank**	**Comments**	**Code**
Java	49	1302	1044	5319
XML	54	192	233	1144
Groovy	10	37	15	172
Bourne Again Shell	1	20	21	123
JSON	1	0	0	73
DOS Batch	1	24	2	64
Prolog	3	6	0	45
Total	**119**	**1581**	**1315**	**6940**

Backend				
Language	Files	Blank	Comments	Code
Go	16	351	51	1949
JavaScript	1	54	1	723
HTML	1	18	0	382
SQL	9	218	199	279
Total	**27**	**641**	**251**	**3333**

Web Frontend				
Language	**Files**	**Blank**	**Comments**	**Code**
JavaScript	1	28	15	325
HTML	1	17	70	147
Total	**2**	**45**	**85**	**472**