Imperial College London

EE2-PRJ E2 Project

FINAL REPORT

SUBMISSION DATE:	13 March 2016
SUPERVISOR:	Dr Balarko Chaudhuri
SUBMITTED TO:	Dr Kristel Fobelets, Mrs Esther Perea
GROUP 12:	Barnabás Bognár CID: 00941058
	Francesca R. Cavallo CID: 00981060
	Kaan Giray CID: 00938925
	Rami Kalai CID: 00869073
	Arisa Roongjirarat CID: 00973647
	Adamos Solomou CID: 00984498
	Artem Voronov CID: 00946146

Abstract

Queuing is an unavoidable experience in everyday life and it cannot be eliminated. However, it is possible to transform the time wasted in queue into time that is productive and enjoyable. eQueue aims to revolutionise the queuing experience and provide an inexpensive, universal and enjoyable solution.

To tackle this problem, the group aim to design a complete queuing system. It will be used to register customers in queues and provide real time information about the waiting time. Hence, eQueue allows the customers to use the time they would waste in queues in a productive way.

Several queuing aspects were considered and already available queuing systems were researched. Furthermore, a survey to collect information on how the queuing experience could be changed was carried out. The design criteria of the product were specified in terms of a Product Design Specification document. Based on this, three preliminary design concepts were considered and then evaluated using a decision matrix based on the most important design criteria of the project. The concept that accomplished all of the specified criteria was selected as a final design.

eQueue was then developed. The design is based on QR technology and web design in order to create a complete queuing system. eQueue combines the use of a smartphone application, a cloud based database and exchanging information over the internet to provide relevant queuing information to the customer. At the same time it gives organisation the opportunity to manage their queues more efficiently.

Finally, the group designed a working prototype which will bear the basic core for an advanced and fully functional queuing system able to compete in the demanding market.

1. Introduction

"Waiting is frustrating, demoralising, agonising, aggravating, annoying, time consuming and incredibly expensive" (Giridharadas, 2010).

Queueing theory started in Denmark, about 100 years ago, thanks to the studies of Erlang, but despite the long time since the first mathematical formulation was made, queueing mathematics has not changed much (Bhat, 2008). Instead, queueing psychology has, since "often the psychology of queueing is more important than the statistics of the wait itself" (Stone, 2012).

Queues are a negative daily experience not just because of the duration of the wait, but also because of the way one experiences that duration. For example, studies show that we are much more patient of we know roughly what the duration of the wait will be (Maister, 1985). Moreover, as a matter of fact, the time occupied is perceived as passing faster than the time spent merely waiting (Stone, 2012). Another aspect that affects the way people experience queues is fairness: seeing a person that arrived later being served earlier compromises the reputation of a business, as this is seen incapable of adequately managing its queues. Solving these issues becomes difficult in the presence of a traditional way of managing queues - for instance by means of paper numbers. However, it is crucial to find an answer to the queueing problem, as a better way of dealing with queues would improve the quality of the service and the reputation of a business.

A significant number of solutions that tackle the problem of queueing have been developed, as it was outlined in the Interim Report. However, the majority of them aims to make the queueing experience more efficient and quicker and almost none focus on the way people experience the queue and their feeling when this is not well managed. The few products that take this aspect into account provide solutions that are not attractive, and produce benefits just for the business that adopts them. More details about this are provided in the Market Research section (Appendix B, page 26) of the Appendix.

All these issues have inspired the idea of a product able to provide inexpensive and feasible solutions to make the waiting time more enjoyable and to transform the queueing experience into a good opportunity both for customers, who can invest their time differently while still reserving their place in the queue, and for businesses, that can highly benefit of the customers satisfaction. In particular, the two key aspects around which our product is developed are that occupied time goes faster than unoccupied time, and that people are more patient when they know the duration of their waiting. Therefore, a prototype a product that meets there criteria has been developed and will be illustrated in detail in the following sections.

2. Competitors & Innovation

2.1 Competitors

Electronic queuing systems are not a new concept and several products have been designed in order to tackle this problem. In order to deviate from what is already available, the group tried to innovate and add functionalities that are not included in current designs. To do that a market research has been carried out to determine what is currently available in the market. Below, the most outstanding products are briefly discussed; for more information, references and detailed comparison tables please refer to the Market Research section in Appendix B, page 26.

2.1.1 Qminder

Qminder is a queues management service that provides features such as multiple service lines and the ability to track customer traffic. However the method with which the customer's information is entered into the queueing system is still done manually. This is not a major problem if the business is not busy, but might become more problematic during peak hours. Furthermore with Qminder the customer is still required to stay within the business building to wait for their turn. If the waiting time is especially long (e.g. more than 30 minutes), as an alternative, eQueue proposes to send push notifications to the customer's phone to alert them when their turn comes. (Qminderapp.com, n.d.)

2.1.2 SmartWait

SmartWait is a more customer oriented queues management system. It is compatible with any business which has adapted eesii's (the company who owns SmartWait) queue management system. The feature which stands out the most is the real time information on waiting conditions and the user's ability to postpone their call. On the developer's website, it is mentioned that the app can be used in retail, finance, health and public sector. A potential problem with the app being openly accessible on the AppStore, is the fact that businesses have no way of telling who is a real customer. Fake customers can request a virtual ticket without the intention of ever turning up. To overcome this problem, an extra layer of security should be introduced to identify a user as a genuine customer. It could be implemented in the form of a code the customer has to enter when they request to use the virtual queueing system of a particular establishment for the first time. After the establishment has confirmed that the user is a genuine customer, the code will be send. This is especially important for the health, finance and public sector. (smart-wait.com, n.d.)

2.1.3 Ufficio postale by Poste Italiane

Poste italiane is the Italian national postal services company, and has developed a downloadable app that allows you to book your place in the queue in advance; if less than 10 people are present, the app will not allow you to book in advance but ask you to go directly to the post office. It can only be used at Italian post offices and it has a different principle from our idea as the customers book their place in addition to scanning at arrival. Moreover, the customer can book their time slot without the need of going to the post office, but there is not the possibility of tracking the state of the queue. This application provides several additional feature that, however, are only relevant in the context of post offices. (PosteApp.it, n.d.)

2.1.4 Wavetec

Wavetec is a multinational company specialised in queue management systems. Customers have to wait in the building and are entertained by advertisements. Special displays provided by the company are required: therefore, the cost for the businesses that wants to adopt this system is increased. Moreover, it does not eliminate the use of paper tickets. There is also an app that a business can decide to adopt, but does not provide any additional feature not related to the state of the queue. It allows to reschedule the time in which a customer has to be served and provides real time information on the state of the queue.

It is a very well known company, and it works with a large number of famous businesses. (Wavetec, n.d.)

2.1.5 Qless

Qless provides a list of all the queues that one might join. The program provides live updates using SMS. Customers can interact with the business by replying to the SMS: they can leave the queue, request more time if they are running late. The app compares different businesses of the same type and gives estimated waiting time to give the customer the possibility to chose where to go. The businesses can interact with the customers via voice calls, and via the QLess kiosks.It uses data from customers (which location the customer has visited, how recently the customer last visited, what days of the week the customer most often visits, what time of day the customer most often visits, How frequently the customer visits, where the customer lives, how reliably the customer ultimately shows uh) to effectuate a hyper-targeted SMS marketing strategy. This may not provide safety of personal data, but allows customers to be sent targeted discounts, coupons, news and events. Finally, it provides in-depth information regarding customers behaviour and metrics on no-shows, walk-aways, service duration and customer profiles.

It has applications in education, governmental institutions, industries, healthcare systems. (Qless, n.d.)

2.2 Why eQueue is better

eQueue combines already existing technologies like QR code, smartphone application design and web design in order to provide a complete product that tackles the specified problem. The main deflection from the existing products is that eQueue does not aim to serve a specific organisation. eQueue can be adopted by any business and this is enabled by the modular design that the group followed through the development of the product and this was one initial targets of the team. Another important deviation is that eQueue equally takes into account both types of customers (end-users and businesses). Most of the existing products are either designed to serve the end-user or the business that hosts the queue. However, eQueue considers the two type of customers by having its own designed smartphone application and a cloud based database that exchange data between each other.

Another functionality that none of the existing competitors provides is the recommendation of alternative activities during queuing time. This was one of the main innovating features that the team would like to integrate in the product. However the group did not manage to include this function in the prototype design due to the limited time that was available for the project design (For more details please refer to section 6 - Future Work, page 20).

3. Design Criteria

In the Interim Report, a set of design criteria that the product have to met were outlined. At this stage, with a working prototype, the same criteria outlined in the previous report will be discussed in more detail, and an explanation of how each one of these criteria are met by the product is provided.

3.1. Compatibility

Compatibility is the most important design criteria for the success of the product in the market. Developing a product that can be supported by different operating systems increases the number of customers that the product can reach. Moreover, a product that is able to adapt to different situations, reduces the time required for the implementation and the costs related to the installation.

Our product scores a high percentage in compatibility, as it is developed both for Android and iOS systems. In addition, the algorithm that calculates the average waiting time can be easily modified depending on the requirements of a specific business.

3.2. Customer/User-friendliness

As stated in the Interim Report, there are two types of customers that need to be taken into account during the development of this product: the organisations that decide to adopt the described queue management system; the end-user, who are the organisation's customers. For both of them, the group tried to design the product as user-friendly as possible so it can be used by anyone regarding age and education level

The system is designed to be easily adaptable to different types of businesses, especially when it comes to the variety of customer information an organisation can store which will allow them to make the queueing process even more efficient. Moreover, the dashboard, which represents the user interface that is managed by the organisation, is intuitive and presents a simple and tidy layout that avoids delays that someone new to the system might cause.

The app used by the end customers is designed to be user-friendly and has an intuitive interface. Finally, the aesthetics of the product has been taken into account during the development, in order to make the product more competitive in the market, and allow it to stand out when compared to the already existing solutions.

3.3. Complexity

The complexity of the design will affect the cost in terms of time and money when the product will be further developed in the future.

A design that is complicated even in its basic functions has less possibilities to be developed further. In fact, adding functionalities to a complex design is time consuming as it may even require the developers to modify the deepest layers. Therefore, during the development of the prototype, which may represent the first stage of a more ambitious project, the group has designed the basic functionalities of the product making sure that they are efficient and well performing.

Moreover, the product has been developed in a modular way. A modular design is easier to implement and modify, and allows an effective definition of the tasks inside the group. Modularity is an essential feature for all types of designs, and the way our product has been designed responds to the nature of all software platforms, which need to be periodically updated and adapted to different technologies and situations.

3.4. Performance

The high level description of the performance is relatively simple. The customer has the possibility to check the state of the queue in a chosen business at each time of the day. Based on the average waiting time, the customer can decide to join the queue. To do so, the customer will need to use a smartphone, where the relevant app was previously installed and configured, to check in (by scanning his/her QR code) upon arrival. After the check in, the customer will be given the time they have to wait before being served. In the meanwhile, the business will receive the relevant information about the customer and insert the new customer into the queue. The electronic queue will be managed by the person at the desk.

In the later stages, extra functionalities will be added depending on the specific necessities of different type of organisations.

3.5. Competition

Businesses who offer electronic queueing systems already exist. The group was aware that for our product to be able to compete with the rest, it would have to have a unique selling point that would stand out from the rest. This is why the group has done some extensive market research on the current competition which included breaking down the product into different categories and ranking them. [Please refer to section 2 - Competition, page 2 ,for a more detailed analysis about the competition]

3.6. Target Product Cost

Rather than being a physical product, the electronic queueing system resembles more of a service that is going to keep evolving and require frequent maintenance. This is the reason why the group intends to opt for a yearly subscription fee, instead of a one-off price and include the maintenance fee in the subscription fee.

At the current stage, the group is unable to state a concrete price for the subscription fee. In order to offer a competitive price, research about the prices of businesses offering a similar service to us would have to be done in conjunction with a survey to potential customers about how much they would be willing to pay for such a service.

3.7. Maintenance

The maintenance of the service will come in the form of bug fixes for both the app and the database / dashboard. However, due to the rapid speed at which technology is evolving, in order for the service to stay relevant, it cannot solely rely on bug fixes. After its launch, through regular feedback from the customer (i.e. organisations and end-users), the service will have to constantly evolve and adapt to their needs. For example, adding new features or updating the user interface to make it even more user-friendly. Additionally, every time there is a new operating system update for either Android or iOS, we would have to ensure that the app will remain compatible with both.

3.8. Safety/ Privacy

This area was approached from two angles.

3.8.1 The organisation:

At the moment the dashboard can be accessed through a login ID and a password. If we were to further to develop the service, we would look into methods of increasing the layers of security (to enforce that only authorised personnel can gain access to the dashboard) and also the encryption of the customer's data. After organisation's initial dashboard has been set up, full control will be handed over to them and there will no further intervention by the group except in the case when maintenance of the system is required.

A main concern for the group was the ability to distinguish real customers from fake ones. Since the app will be freely available on the app store, anyone with a smartphone can download the app. Initially the app had a feature where the customer was able to check in from home. However, this meant that anyone with the app could check in, even if they were not a customer. To prevent this, this feature was removed and now the customer can only check in by personally scanning their smartphone at the business.

3.8.2 The end-user:

The group's main concern for the end-user was the security of their personal information. At the moment the service is set up in such a way that it will only require minimal personal information (i.e. first name, surname and phone number) of the end-user. Before they can download the app, they are also going to be made aware of this through the terms and conditions. In the future these can be updated accordingly, as the app gains more features.

Comment on design selection

A final design to be developed was selected at the very beginning of the Spring Term (week 15). The selection process was therefore detailed in the Interim Report. Please refer to the Interim Report for information about the three concept designs considered and the selection process.

4. Concept development

The system is composed of four main modules: scanner, database, admin web app and mobile application. The module specifications have been identified as follow:

1. Scanner

It decodes the QR code, writes in a data structure to store the name, surname, phone number and email of the customer.

2. Database

3. Admin Web App

The queue manager will see how many people are in the queue, and the first three people in the queue will be shown in the screen. The queue Manager will be able to move the queue once a person has being served. The average waiting time is predicted using basin queue theory principles, and this time will be shown on the screen.

4. Mobile application

The user will input name, surname, phone number and email address and the app will encode these data in a QR code. Once the code is scanned by the relevant scanner, the app will display the remaining time and give suggestions of alternative activities based on the remaining time and geolocation.

4.1 Scanner

4.1.1 Background

Since QR scanners are sold from £200, the group decided to develop a QR scanner in order to make the product accessible to a wider range of customers. The scanner development posed many design choices.

The QR scanner is implemented with a webcam and should work with the major operating systems like Linux, OSX and Windows. The group have started to develop the software in Linux in C++ language. Developing for Linux in C++ presented several issues, as accessing real-time video from an webcam is not a straightforward task. The problems were mostly platform and driver specific.

Different methods of streaming real-time video were researched and the group decided to sue WebRTC API. "WebRTC is an open framework for the web that enables Real Time Communications in the browser. It includes the fundamental building blocks for high-quality communications on the web, such as network, audio and video components used in voice and video chat applications." (Webrtc.org, n.d.). The use of the WebRTC protocol made the system browser dependent, but it avoided any platform and device specific complexities. This trade off has enabled a more rapid development of the project.

To develop the QR scanner, the group started from existing QR decoders. The most popular one is called ZXing (Zebra Crossing)(Lazlo, 2011), which is an open source program under Apache 2.0 licence. ZXing is a Reed Solomon Error correction algorithm implemented in Python. Javascript QRCode scanner is a javascript implementation of the ZXing project and it is open sourced under Apache 2.0 License. With the methods just described, it is possible to decode only one QR code on the browser, and to scan another code the user needs to refresh the browser.

The specification for the design of the QR scanner have therefore been identified as follows:

- The device should work hands-free. The browser should not be refreshed each time.
- Once the QR code is scanned, the result should be stored in an appropriate data structure.

- The same code should not be added to the database again.

4.1.2 Storage

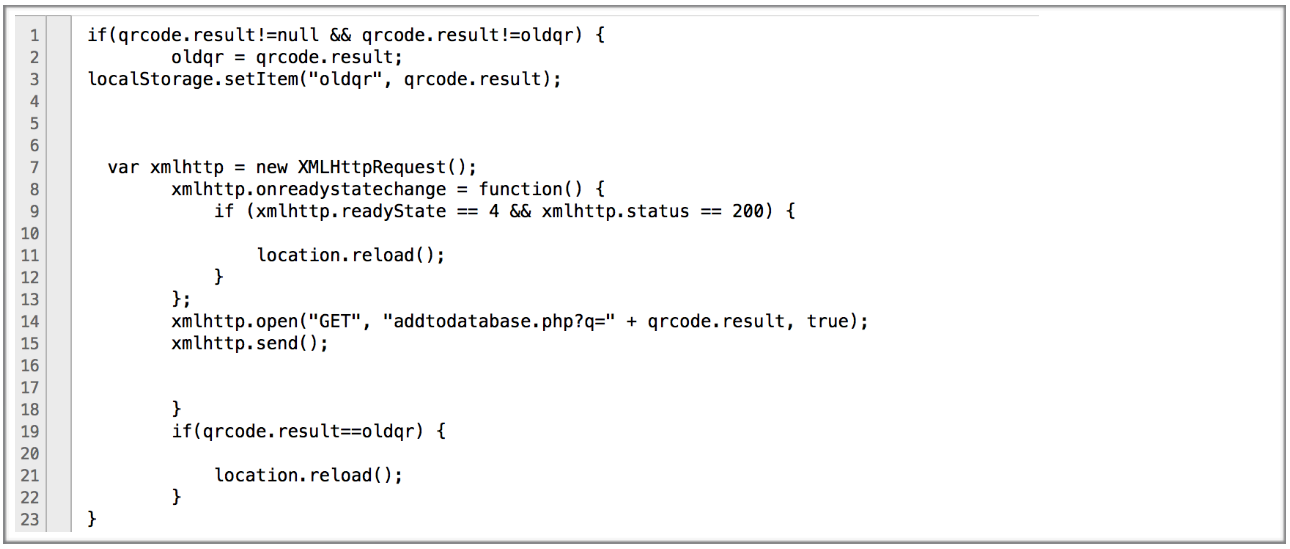
While the group was working on the C++ implementation of the project, it was decided to use linked lists as the data structure. However, developing a system for the web, it was decided to use MySQL (Structured Query Language), which is a open source relational database management system. By using MySQL and storing the data in a server, security became a real concern. A possible attack method will be discussed in the following parts. Cloud SQL Service was used to host the database.

4.1.3 Data Transfer Architecture

Once the webcam is connected to the scanner computer, a web browser should be opened and scanner web page should be loaded. The web page is coded in HTML (Hypertext Mark-up Language) and CSS (Cascading Style Sheets). HTML provides the structure and CSS provides the layout (W3.org, n.d.).

The scanning code is implemented in Javascript and encapsulated in the HTML page. Javascript is an interpreted language, and unlike C++ it doesn't need to be compiled since it is natively interpreted by the browsers. All the three languages are client side languages hence they are interpreted at the browser level.

The following code is used to transfer data into the database.



```
1   if(qrcode.result!=null && qrcode.result!=oldqr) {
2           oldqr = qrcode.result;
3   localStorage.setItem("oldqr", qrcode.result);
4
5
6
7     var xmlhttp = new XMLHttpRequest();
8           xmlhttp.onreadystatechange = function() {
9               if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
10
11                  location.reload();
12              }
13          };
14          xmlhttp.open("GET", "addtodatabase.php?q=" + qrcode.result, true);
15          xmlhttp.send();
16
17
18          }
19          if(qrcode.result==oldqr) {
20
21              location.reload();
22          }
23  }
```

Figure 1. Code for QR scanner

The first condition is used to avoid scanning the same code twice. If the scanned code is equal to the old code, the page is reloaded, otherwise the data is sent to the database and page is reloaded. The page is reloaded in any case in order to avoid looping the JavaScript code forever. A vast part of modern browsers stop JavaScript codes after a set number of iteration in order to prevent the browser from crashing; on the other hand one can reload the page many times and the browser will function. This is an essential trade-off for code to work.

Once the code is scanned, line 7 to 23 are executer. This part is very generic and it is called an AJAX call. AJAX, asynchronous JavaScript and XML, is a method for communicating with peripherals. When this query (last two lines) is sent, JavaScript code halts and waits for an answer. If the answer is success the page is reloaded (lines 19-23).

In the case of interest, the query is a PHP query. PHP is a server-side scripting language. While every process was done on the browser until now, after this point the query will be resolved at the server side. The code `("GET", "addtodatabase.php?q=" + qrcode.result, true)`, `q=qrcode.result` to the `addtodatabase.php` page is hosted in Imperial servers. Please refer to the php code on Appendix X for more complete information.

The core of the PHP page is the following code:

```
$sql = "INSERT INTO `queuebase`.`myQueue` (`Name`, `Surname`, `Phone`, `Email`, `reg_date`, `Valid`)
VALUES ('$name' , '$surname', '$phone', '$email', '$date', '1');";
$result = $conn->query($sql);
```

This lines initialise the string variable sql, and send the sql to the connection that has been established before. While the php page is hosted at Imperial servers, database is stored at Google's Europe servers. This part is implemented in structured query language and it is interpreted on the server side. The `queuebase` database and the `myQueue` table adds the scanned values to the respective fields.

When SQL is used, the most important consideration is to sanitise the SQL code against injections. Here an arbitrary example of a SQL injection is given, assuming that a user input is saved into the $Name variable. If an attacker enters:

```
Robert'); DROP TABLE STUDENTS; --
```

When the following query runs

```
INSERT INTO Students VALUES ( '$Name' )
```

MySQL parses.

```
INSERT INTO Students VALUES ( 'Robert' );  DROP TABLE STUDENTS; --' )
```

Anything after the dash is commented out: this code deletes the whole table rather than saving a name into the table. The code that the group developed is not properly sanitised due to the imposed time constraints. However, any user input is exploded at spaces and only the first four members are allowed to pass. For example, the following code provides protection against basic injection attacks even though it does not fully sanitise the code.

When the results are written into the database, a Boolean result will be sent to the `addtodatabase.php`. When

```
$arr = "Jane Doe 07564887652 jane@doe.com"

$arr = explode(" ", $q);
$name = $arr[0];
$surname = $arr[1];
$phone = $arr[2];
$email = $arr[3];
```

the php page receives the answer, it will route it to the AJAX call and the HTML page will respond.

4.2 Database

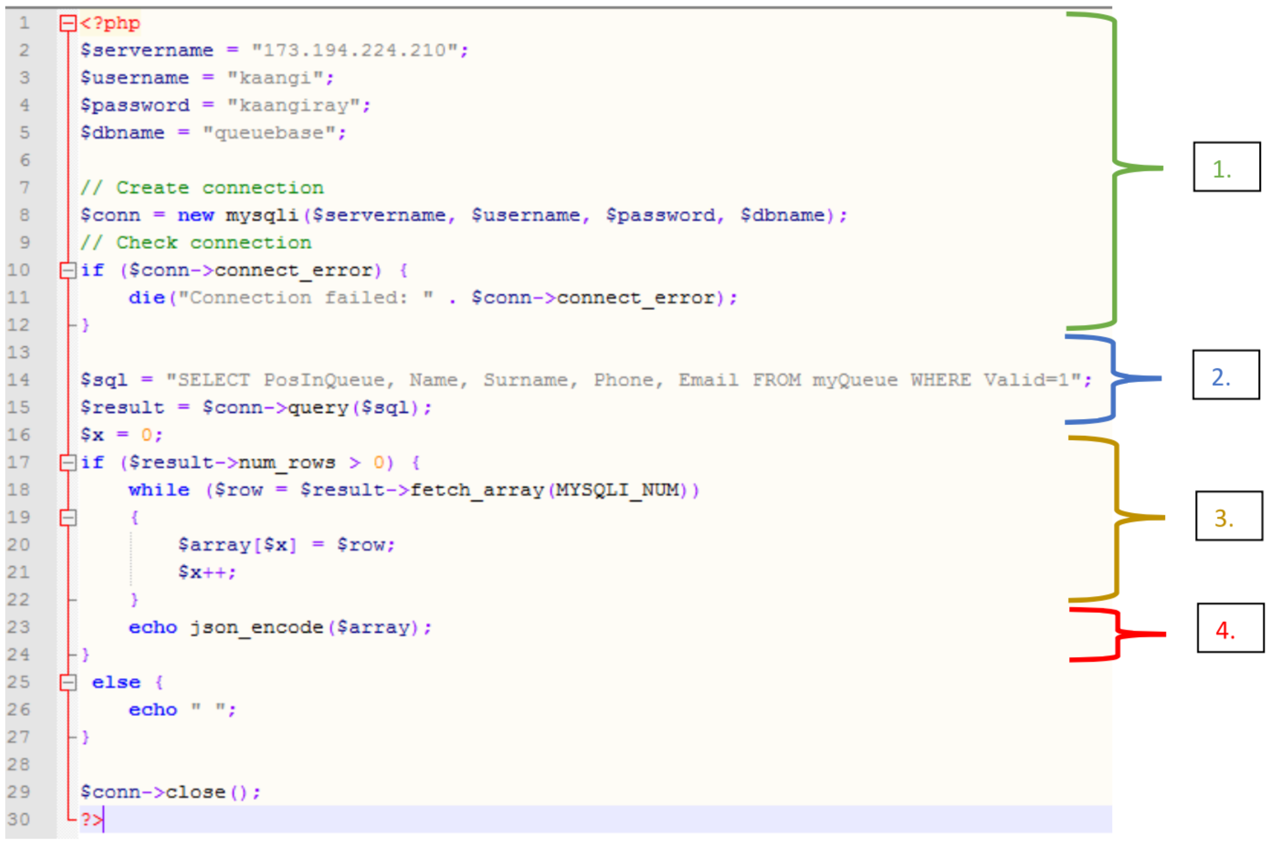
The database forms the centre of the whole system design. It brings together all the aspects of the system by acting as a link between the different user interfaces and providing a means to port data from one end to the other.

Internally, the database is formed by a main table that contains the names of all the queues registered in the system. The table holds the relevant information about each queue, such as average waiting time per person and sample space to date. In turn this table points to two additional tables: one forms the virtual queue, and the other contains all the information related to logging in and to controlling the queue, such as admin names, emails, and passwords.

The database contains the information sent by the external modules, i.e. the web-app, the mobile app and the scanner. In addition, it holds the data computed from the provided information. All this data is continuously updated for accuracy. This is possible thanks to server-side scripting, php and database management MySQL.

PHP is a server side scripting language that stands for the recursive acronym: "PHP: Hypertext Preprocessor". It is executed on the server where the php file is stored (equeuesystems.com) and performs a connection to the database server (Google Inc.) MySQL is used to access the database, read from it and update it accordingly. This happens within the PHP script. PHP provides the link between the modules and the database and allows them to send, receive, and compute data. In all the php scripts used, there are four main steps needed to complete the required operation successfully:

1. A connection to the server hosting the database (Google Inc.)
2. An access to the database “queuebase” using MySQL
3. Evaluation or calculation (usually involves a loop)
4. Return a result



```php
<?php
$servername = "173.194.224.210";
$username = "kaangi";
$password = "kaangiray";
$dbname = "queuebase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT PosInQueue, Name, Surname, Phone, Email FROM myQueue WHERE Valid=1";
$result = $conn->query($sql);
$x = 0;
if ($result->num_rows > 0) {
    while ($row = $result->fetch_array(MYSQLI_NUM))
    {
        $array[$x] = $row;
        $x++;
    }
    echo json_encode($array);
}
 else {
    echo " ";
}

$conn->close();
?>
```

Figure 2. FetchTable.php code

A sample php file used in this project:

The above file is called “FetchTable.php” and it is used to fetch the complete queue and display it in the web-app.

Connections to the database will always look the same for the same database. A connection is comprised of setting up the ip address, username, password and database name, followed by a new connection, a check for success (or error), and finally closing the connection at the end of the php file. Accessing the database is done using MySQL. This is a structured query in the form `$sql = "SELECT PosInQueue, Name, Surname, Phone, Email FROM myQueue WHERE Valid=1";` where the capitalised words are reserved words that are interpreted by the database and dealt with accordingly. In this case, the query will select the four columns PosInQueue, Name, Surname, Phone and Email from the table myQueue where the Valid column has a value of 1. The calculation differs from php file to another. Here it is simply storing the fetched values in an array, however it can be any mathematical operation (within reason). Finally the requested information is returned via the “echo” keyword. This is a reserved keyword in php that displays the data that follows it ($array), and is read by the calling function. The FetchTable javascript function in the web-app in this case.

The database is managed using MySQL Workbench. A tool used to check the status of the server, access the database directly and update it at will.

4.3 Admin Web App

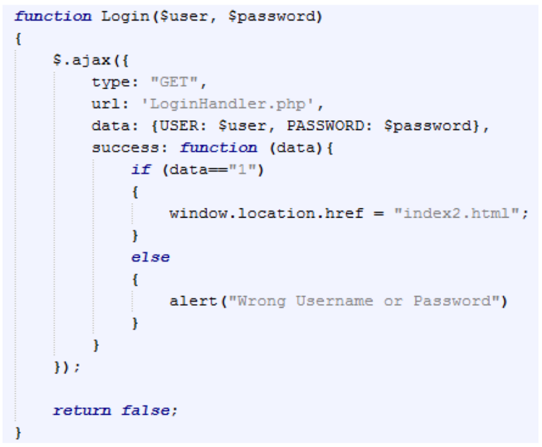
The admin web-app is based on an open-source Bootstrap Template (SB-Admin2), however, a lot has been changed in order to accommodate the required specifications. JavaScript is used throughout the web-app to perform functionalities such as performing AJAX server requests.

4.2.1 Login

This is the introductory page to the web-app. Here a user can sign-in to view and control his queue or sign-up to become an admin for a specific queue.

The page contains three JavaScript functions: `Login ($user, $password)`, `AddAdmin ($name, $surname, $email, $password)` and `LoginReset ()`.

Upon loading the page, `LoginReset ()` is called to log the previous user out of the web-app by setting the session variable to false by the `LoginReset.php` script. The user is then presented with two options: sign-in or sign-up. Clicking one of the two buttons will make a modal appear with either the sign-in form or the sign-up form. Depending on the form chosen, a specific function will be called (`Login` or `AddAdmin`).



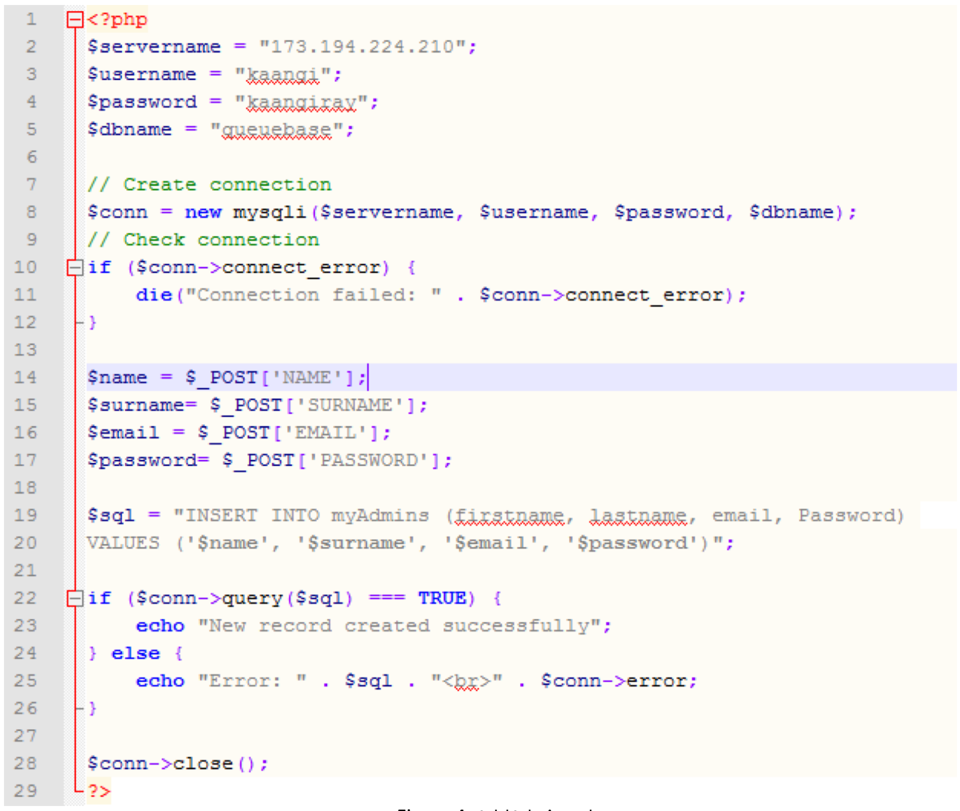
```
function Login($user, $password)
{
    $.ajax({
        type: "GET",
        url: 'LoginHandler.php',
        data: {USER: $user, PASSWORD: $password},
        success: function (data){
            if (data=="1")
            {
                window.location.href = "index2.html";
            }
            else
            {
                alert("Wrong Username or Password")
            }
        }
    });

    return false;
}
```

Figure 3. `Login ($user, $password)` function

The function `Login ($user, $password)` takes in input the user's email and password and verifies that they correspond to the correct details. This is done by performing an AJAX (Asynchronous JavaScript and XML) server request. The AJAX request then passes the inputs to a corresponding php script (LoginHandler.php) that checks them against a table called `myAdmins` in the database and sets a session variable if the details are correct. The php script returns either "1" or "0" to indicate a correct entry or not. If 1 is returner, the php will redirect the user to the dashboard where the session variable is checked (so as to allow access). If 0 is returned, the php will alert the user that a wrong username or password was entered and will prompt him to try again.

The function `AddAdmin ($name, $surname, $email, $password)` is called if the user clicks the sign-up button. The function takes in input the relevant details required to create a new admin record in the database and therefore returns a success message (again, this is done using AJAX and PHP). Upon successful completion, `Login ($user, $password)` is called with the newly created account and this function automatically sets the session variable and redirects the user to the dashboard page as explained above.



```
<?php
$servername = "173.194.224.210";
$username = "kaangi";
$password = "kaangiray";
$dbname = "queuebase";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$name = $_POST['NAME'];
$surname= $_POST['SURNAME'];
$email = $_POST['EMAIL'];
$password= $_POST['PASSWORD'];

$sql = "INSERT INTO myAdmins (firstname, lastname, email, Password)
VALUES ('$name', '$surname', '$email', '$password')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

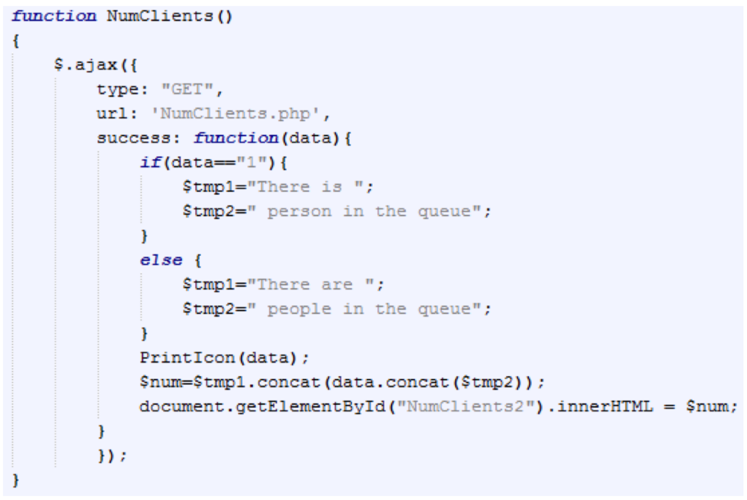
Figure 4. `AddAdmin.php`

4.2.2 Dashboard

This page is where the admin will be able to view the first three people in the queue, view details about the current waiting time and see how many people are currently part of the queue. Most importantly, the admin will able move the queue. In terms of implementation, there are two paths of execution: polling and moving the queue.

The polling function performed by this page reloads parts of the page asynchronously (using AJAX) every five seconds, in order to keep all the information up-to-date without the need to refresh the browser manually. This function is simply a call to a set of functions that will be explained below.

The function `FetchEntries ($pos)` always called with 1 as input and it recursively calls itself for positions two and three. These positions correspond to the three boxes in the dashboard that display the first three people in the queue. After returning the AJAX, `FetchEntries` calls `CustDetails ($pos)`, which in turn performs an AJAX request to fetch the Customer's details for positions one, two and three. The fetched data from the database is then displayed in the correct position by using the id of the corresponding HTML tag where the information should be printed.



```
function NumClients()
{
    $.ajax({
        type: "GET",
        url: 'NumClients.php',
        success: function(data){
            if(data=="1"){
                $tmp1="There is ";
                $tmp2=" person in the queue";
            }
            else {
                $tmp1="There are ";
                $tmp2=" people in the queue";
            }
            PrintIcon(data);
            $num=$tmp1.concat(data.concat($tmp2));
            document.getElementById("NumClients2").innerHTML = $num;
        }
        });
}
```

Figure 5. `NumClients()`

The function `NumClients()` sends a php request to calculate and return the number of people in the queue. The result is therefore printed in the correct position in the page, along with the statement "there are x people in the queue". This is because there are two cases and a static statement would not suffice.

The function `GetWaitingTime()` fetches the average waiting time for the relevant queue and then passes that data to `TotalWaitTime ($avg)`, which fetches the total number of people in the queue and multiplies these two results to obtain the total waiting time in seconds. This number is then transformed to minutes and rounded up to the nearest minute and displayed in the correct HTML tag by ID. Finally, `AssignPosition()` is called to reassign positions to the people in the queue in case someone joined or abandoned the queue.

When the button "Move Queue" is clicked, a different set of functions is executed. On clicking the button, `TimeInQueue()` is called. This function evaluates how much time the first person spent in the queue by subtracting his registration datetime stamp from the current datetime stamp and returns it. Upon the successful return of the AJAX request, the two functions `MoveQueue ($diff)` and `UpdateAvgTime ($diff)` are called.

`MoveQueue ($diff)` updates the row corresponding to the first person in the queue by storing that value in a column in the database called `TimeRemaining` and changing the Valid column to 0 – thus moving the queue along. Upon success, `FetchEntries ($pos)`, `NumClients()` and `AssignPosition()` are called.

`UpdateAvgTime ($diff)` fetches the previous average waiting time per person and the sample space, and recalculates the average by adding the time spent by the person who has just left the queue into the average and increasing the sample space by one. This function returns the new average and calls the function `UpdateAvg ($avg)`, which simply stores the value in the correct place in the database. Finally, the function `GetWaitingTime ()` is called and the the new wait time is displayed.

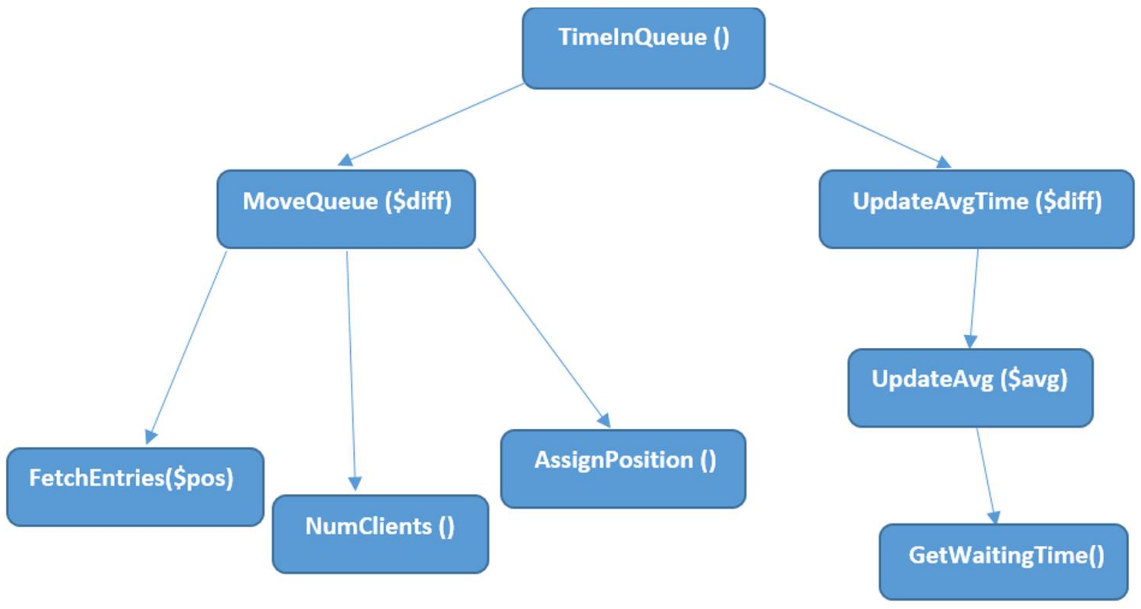


Figure 6. Low level block diagram upon clicking "move queue"

4.2.3 eQueue

This page displays a table containing all the customers in the queue along with all their relevant details such as position, name, surname, phone number and email.

This page simply fetches the information from the database using an AJAX request, where the information is in the form of a JSON (JavaScript Object Notation) encoded array. This is then decoded back into an array and printed out in the correct place in the HTML tag using the ID.

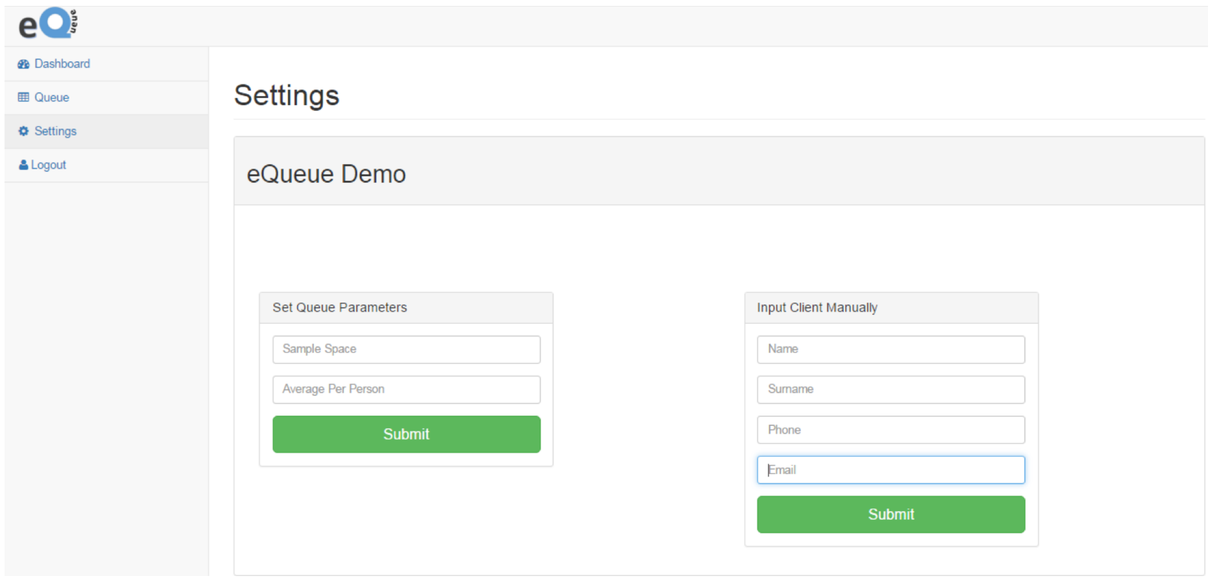
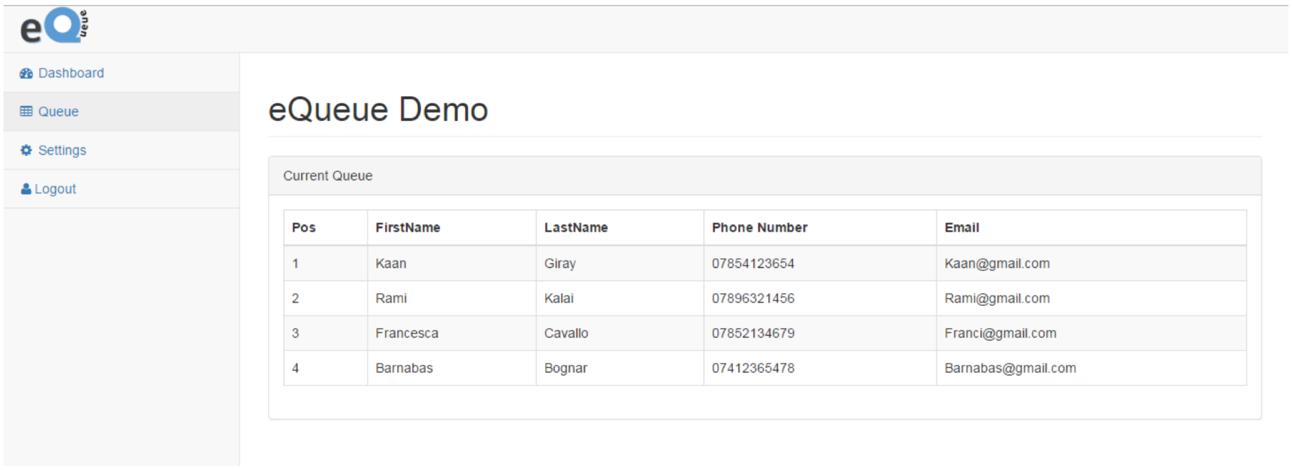


Figure 8. Settings page

4.2.4 Settings

The settings page is where the admin can manually add a customer in the queue and also update the average time per person and sample space for that queue. This is useful in case a user fails to join the queue using the scanner, or doesn't have a smartphone to use the mobile app. The queue timing parameters must only be changed if required, as they represent a constantly changing average. The purpose of this option is mainly to initialise the parameters. This page makes use of the functions `SetSampleSpaceTime ($space, $avg)` and `AddClient ($name, $surname, $phone, $email)`.



eQueue Demo

Current Queue

Pos	FirstName	LastName	Phone Number	Email
1	Kaan	Giray	07854123654	Kaan@gmail.com
2	Rami	Kalai	07896321456	Rami@gmail.com
3	Francesca	Cavallo	07852134679	Franci@gmail.com
4	Barnabas	Bognar	07412365478	Barnabas@gmail.com

Figure 7. eQueue page

4.4 iOS Application

In order to make the product user friendly, user compatible and conventional the group has developed mobile applications for both Apple iOS and Android. In this section, the iOS application will be considered.

The application creates an entry point for customer data and is currently the only module the user may interact with. The application has been designed to work with versions iOS9 for all devices in vertical mode.

4.4.1 Application Requirements

The mobile application is a vital module in the eQueue hierarchy and must perform the following basic functions:

- App must collect the required customer data from the user, such as Name, Surname, etc.;
- This data must be transmitted to the SQL database hosted on the Google Cloud, and a new customer entry must be created;
- The application must be able to connect to the database and retrieve the personal queueing time estimate (individual to each user);
- The time estimate must be printed on the screen, after the user had been registered with the database.

4.4.2 Market Constraints

A research was carried to determine the minimum iOS that the iOS application will be designed for. The results of the research are shown in figure (9) (developer.apple.com, n.d.).

The iOS app should be available for the majority of users. For this reason the application will be designed for iOS 8 and higher, making the app compatible with 95% of the devices that are registered on Apple Store.

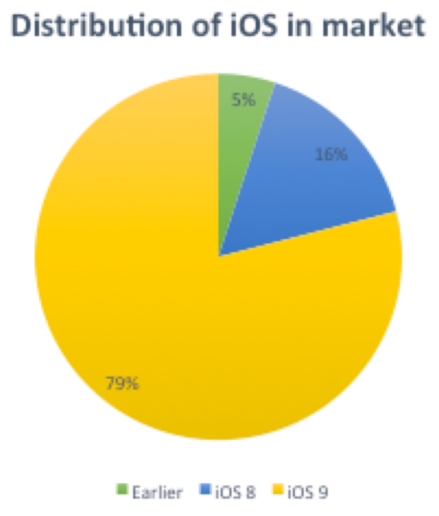
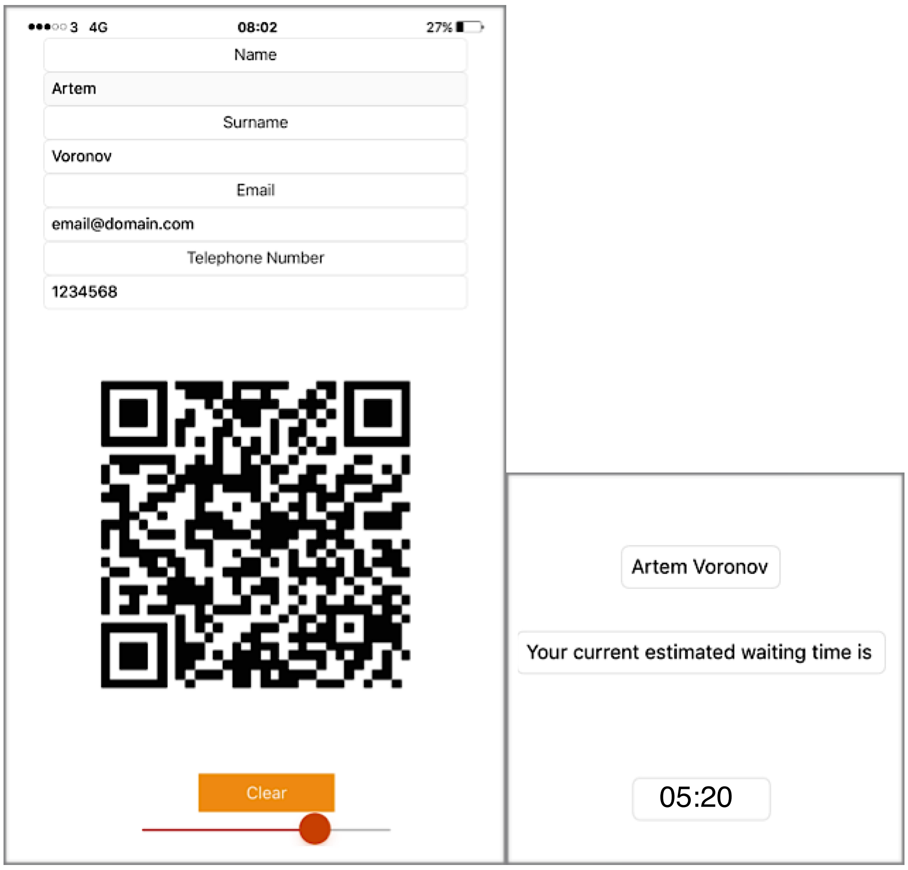


Figure 9. Distribution of iOS in the market

4.4.3 Prototype application user interface

The user application is simple and consists of two distinct tabs: Tab1 and Tab2. The user is allowed to switch between the two as they wish. Tab1 is the first screen that is displayed to the user after the application is loaded. It allows the user to enter their personal details, required for queue registration. Upon completion of the short form, this information is encoded into QR, which is displayed on the screen. The size of the QR-code may be adjusted by a slider, which is displayed after the user has generated the code. The code may not be generated without user input. After the user has scanned the QR-code at the queue registration point, they should switch to Tab2. This is the part of the app that displays how much time the user is expected to be waiting in the queue. Moreover, it shows the user's name in order to make sure that the two match and there has been no error.



Name

Artem

Surname

Voronov

Email

email@domain.com

Telephone Number

1234568

Artem Voronov

Your current estimated waiting time is

05:20

Figure 10. Tab 1 & Tab 2

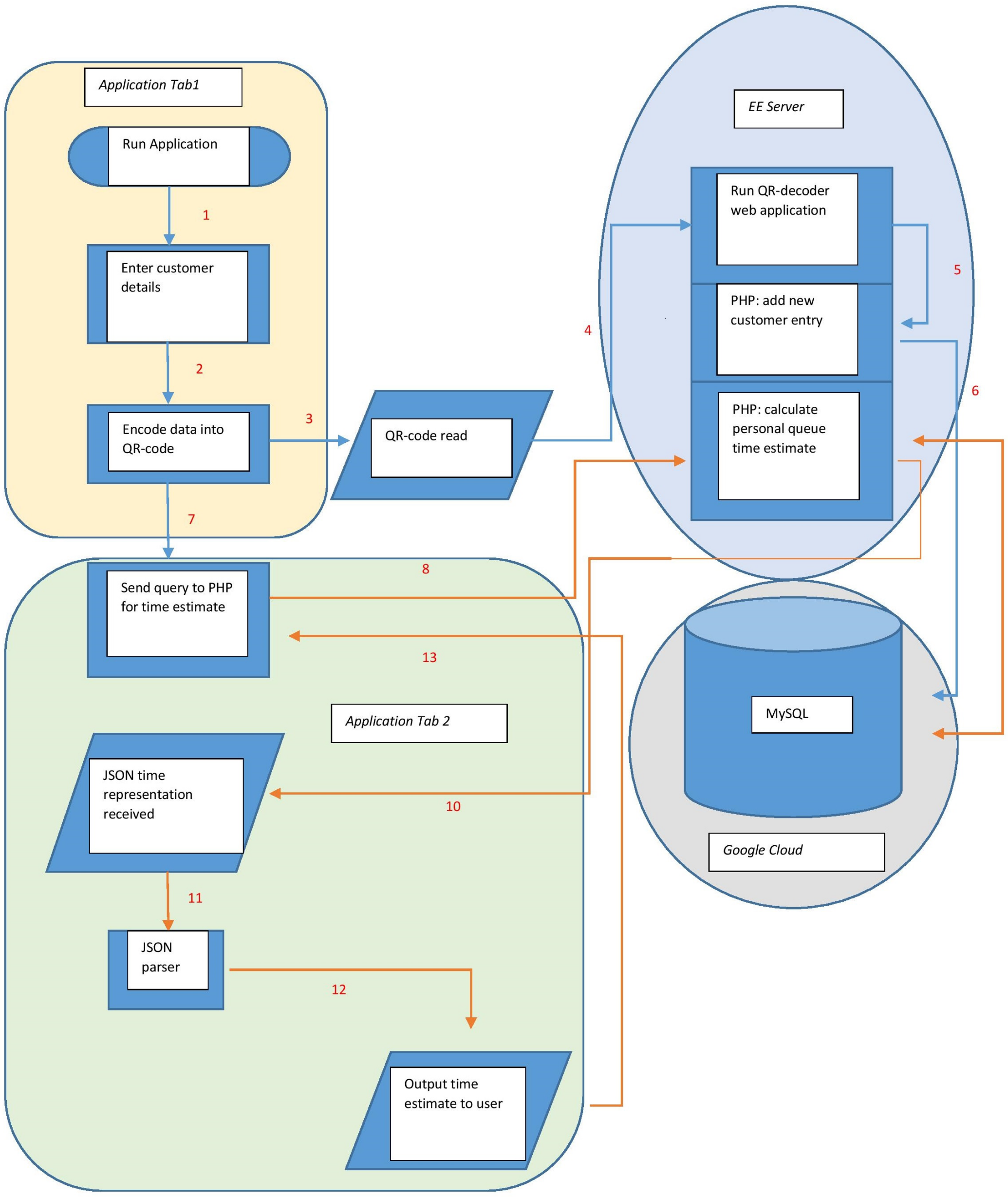
4.4.4 iOS application structure

The application structure will be described using a flowchart diagram. The chart shows the high level functionality of the application, and the way it interacts with other server-based applications and files. Each arrow represents an important transition in the state of the application. The numbering on the arrow indicated the sequence in which the transition occur, with "1" representing the first transition after the app has been initialised. Every rectangle represents a process and the parallelograms represent input/output.

The best way to understand the functionality of the app is to follow the high-level processes which occur in and out of the application, as shown on the diagram.

1. The application has been run by the user from the device's operating system;
2. Tab1 has been loaded, and the user is prompted to introduce name, phone number, email address and telephone number in the appropriate order;
3. Data is encoded into QR and displayed on the screen of the user's device. The user approaches the registration point's scanner and register the code;
4. The QR image is processed at the registration point and transmitted to the server-based QR-decoder. The user credential are decoded back into strings;
5. The user credentials are transferred to a PHP script designed to add a new customer entry in the SQL database on Google Cloud. All PHP scripts along with the web QR-decoder are hosted on the same EE server;

6. The PHP script has sent an inquiry to the database to create a new entry. A new entry is created and the user is registered on the system as a new customer;
7. Now that the customer has been registered, the application is switched from Tab1 to Tab2 by the user. A query is sent to a different PHP script, which calculates the personal estimated waiting time, based on an internal function and previous customer statistics;
8. The query is received and processed by the script;
9. The script communicates with the database and receives the time estimation in integer format;
10. The integer value is encoded into JSON representation, which is read by the application from the PHP file address. JSON representation is conventionally used for processing output on web pages. It allows the amount of received information to be easily scaled and reduces the errors that may occur when processing the textual output of the PHP script;
11. The JSON representation is parsed by the application, using a JSON parsing script to obtain the integer value which has been output by the PHP file;
12. We print the estimated queueing time on the screen for the user to see;
13. Loop back to transition 8 and keep polling. This ensures that the queueing time estimate is always up to date.



Application Tab1
Run Application
1
Enter customer details
2
Encode data into QR-code
3
QR-code read
7
EE Server
Run QR-decoder web application
5
4
PHP: add new customer entry
6
PHP: calculate personal queue time estimate
9
8
Send query to PHP for time estimate
13
Application Tab 2
MySQL
JSON time representation received
10
Google Cloud
11
JSON parser
12
Output time estimate to user

Figure 11. iOS App Flowchart Diagram

4.4.5 General comments

The current application is very basic and has been produced to test the feasibility of the project. It lacks of numerous basic features that are essential for the project due to project timing constraints. These will be added in the following versions of the system and will include the features that will be described in the section 7 - Future Work.

The low-level implementation of the application has not been discussed since it is less relevant than other material presented in this report. The XCode project, however, will be available upon demand.

The application is entirely dependent on the way the whole system is structured and the functionality of the application will always be dependent on the functionality of the system itself. However, the system is sophisticated enough support more advanced application features.

4.4 Android Application

4.4.1 Market Constraints

In order to determine the API level with which the Android application is be designed section a market research has been done in order to gather information about the amount of devices that use each android operating system. This is done in order to design the application such that it can be used by as many customers as possible.

The results of the research are shown in figure 4. The table shows the API level used by each operating system. The most recent operating systems use higher API levels however the most recent one (Marshmallow) only occupies 1.2% of the pie chart. An application developed using the lower API level will be able to support all devices while as the API level used increases less devices will be compatible with the application(Developer.android.com, n.d.).

The application should be compatible with more than 90% of the devices so an API level of 10 or 15 was be chosen.

Operating System	API
Froyo	8
Gingerbread	10
Ice Cream Sandwich	15
Jelly Bean	16, 17, 18
KitKat	19
Lollipop	21, 22
Marshmallow	23

Figure 12. API level of different OS

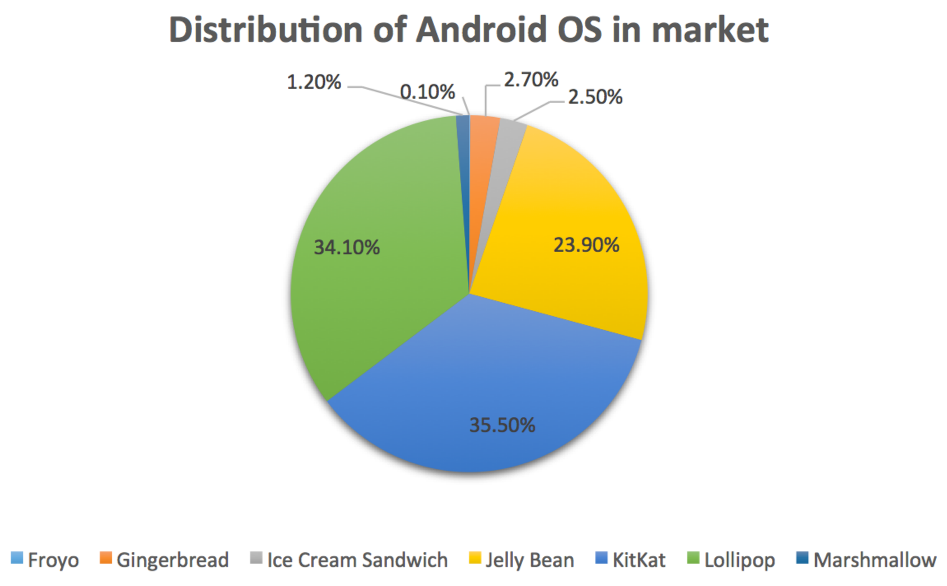


Figure 13. Distribution of Android OS in market

4.4.2 Android application development

The android application was developed using the official Android Studio environment and the SDK 23. SDK (Software Development Kit) is a set of software tools that enables the creation of applications for certain packages/platforms. The application is also designed for API 14: Android 4.0 (IceCreamSandwich) level and higher. API stands for application program interface and is a set of routines, protocols and tools for building software applications (webopedia.com, n.d.). By targeting API 14 as mentioned above, the application is able to run on approximately 97.3% of the devices that are active on the Google Play Store. The decision of using this particular API level introduces a tradeoff between the number of devices that the application can target and the features that are available for the application. By decreasing the API level application, it is possible to target more devices but at the cost of having fewer features are available. In this design, the selected API level allows us to target the vast majority of users while ensuring that all required features are enabled.

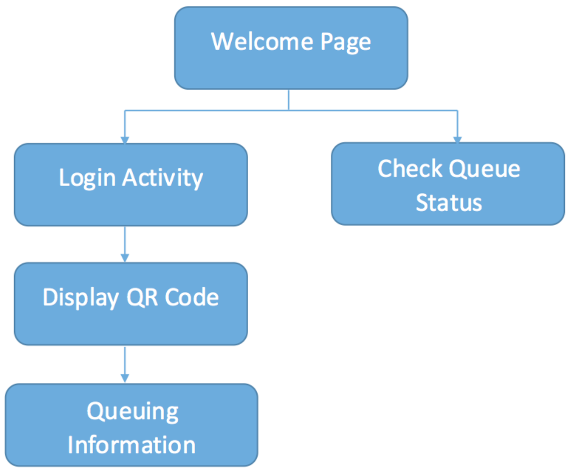


Figure 14. Android application block diagram

The application consists of five main activities: the welcome page, check status activity, login, QR page and the activity that displays to the customer, the remaining queuing time and their position in the queue. These are illustrated in figure 14.

The welcome page allows the customer to choose between the two basic operations of the application, which are the “check in” function of the store and a live update on the queue. In order to check in at the queue, the customer provides their required personal details (i.e. name, surname, phone number and email address). This information is then encoded into a QR code, that the customer needs to scan to the local scanner in order to sign in to the queue. Finally, the application uses the established communication with the database over the internet to show to the customer the remaining time and their position in the queue. Additionally, there is also the option for the customer to leave the queue after they have signed in.

In the login activity the .xml layout consists of four text fields which are used for the customer to fill in their details and a button that moves the application to the next activity. Each object in the xml file has its own ID, which is a unique identifier that can be used within the java file in order to reference that specific object and manipulate its contents. Within the `SignIn.java` file, a function called `sendMessage` is created and it is called when the Sign In button is pressed. In this function, an intent to be delivered to the QR class, is created. An intent is a type of object that provides runtime binding between separate activities within the application. Then, the four user inputs are recovered using the ID of each text field, and each piece of text is assigned to a local string variable. Since the function used later to encode the QR code only takes a single string as an input, the four strings are concatenated into a single string which is then passed to the intent. Moreover, the phone number of the user is passed separately to the intent, since this will be used in the final activity to get the average queuing time. Finally, the `startActivity` method is called with the intent as a parameter in order to move to the QR activity.

The QR code is generated in the QR activity. In the QR.java file, the `getIntent` method is used to get the intent that started this activity and retrieve part of the data contained within that intent which in this case are the concatenated user details. Then this data is assigned to a local string variable which will then be passed as an argument to the QR generating function. The QR encoder is implemented using ZXing (Zebra Crossing), which is an open source image processing library implemented in Java with Apache License (Version 2.0). The encoding function takes as input the string that will be encoded and returns a bitmap that is the generated QR code. A bitmap is a method of mapping from a source domain to bits (values of zero or one), but also gives a means of storing a binary image where each pixel is either black or white. The generated bitmap is assigned to an `imageView`, a class that can be used to display images of any source and is shown on the smartphone screen. Finally, a procedure called `getQueuingTime` is called when the analogous button is pressed by the user. In this procedure, an intent that contains the user’s phone number is created and it is delivered to the `SeeQueuingTime` activity. The `SeeQueingTime` activity starts the new and final activity of the application in which the estimated queuing time and their position in the queue is displayed. The user’s phone number that was sent over the previous two activities within the intent is retrieved and it is used as a unique identifier in order to track the customer in the database. The phone number provides a good way of identifying a customer, since two active phone numbers will never coincide. This way extra information is not generated nor stored.

The application communicates with the database over the internet through a PHP file that takes as input the phone number of the customer and returns the estimated queuing time in seconds. For this to be enabled, internet permission is added to the manifest file of the application. Then, a request for a string response is sent to the appropriate PHP file and this returns the queuing time in the form of a string. To transmit data over the network, the Volley method is used. This is an HTTP library that provides high level networking for android applications and is faster than other methods like Apache or the HttpURLConnection class (eveloper.android.org, n.d.). The Volley method requires an API level higher than 4, a requirement this design satisfies.

Since the value of the queuing time is received in the form of a string, it has to be converted to an integer using the parseInt method of the Java integer class in order to be able to be used later. A countdown timer is implemented starting from the received queuing time and counting towards zero. When the time reaches zero, a message “It’s your turn” is displayed on the screen. Using the same string request method, the position in the queue is received from a different PHP file and is also displayed to the user screen. If the request fails (the main reason for this could be that the user is not connected to internet), then the application will display the error message “Could not establish connection. Please check your internet connection and try again”.

The customer also has the option of leaving the queue by pressing the “Leave Queue” button. On the press of this button, the function `LeaveQueue` is called. In this function, another call is given to a different PHP file, changing the valid bit of the customer in the database to 0. As previously, the phone number of the customer is acting as an unique identified.

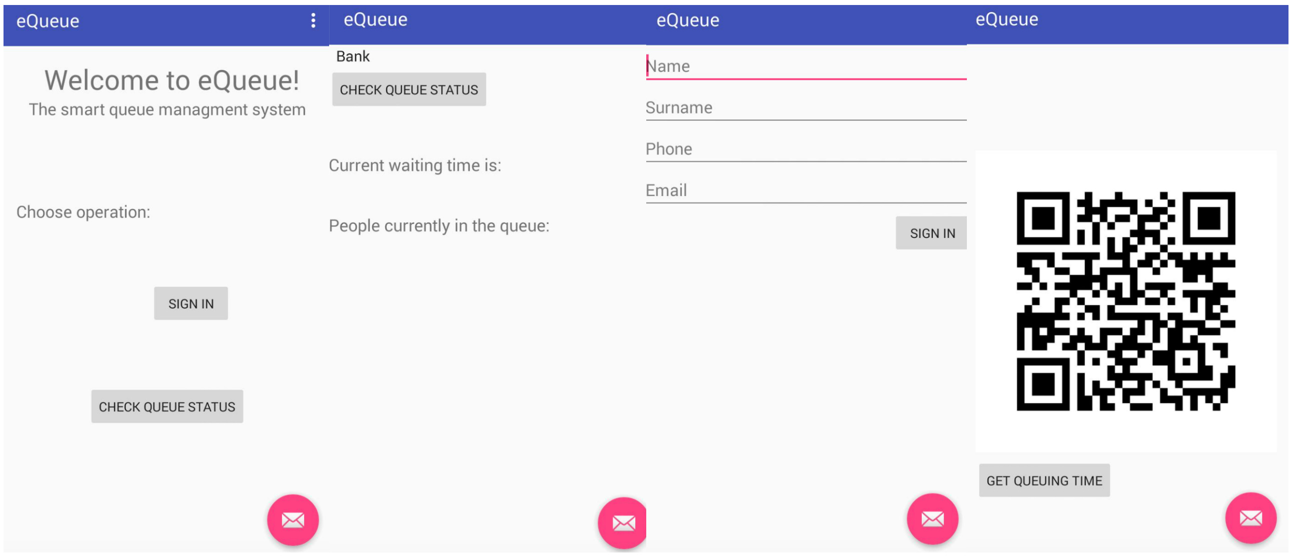


Figure 2. Several layouts of android application for different activities. Welcome page, Check queue status, Login activity, QR activity (from left to

5. Prototype design & testing

The whole design was divided into modules that were designed independently from each other. Therefore, testing was be carried out without having to wait for other modules to be designed.

The QR decoder was the first module to be designed and tested in week 17. Testing was carried out by decoding printed QR codes. At the same time the design of database was carried out and entries could be added manually to it.

The next step of the design was to connect the QR scanner and the database: when QR code was scanned its content was automatically saved to the database. The database and scanner were tested together during week 18. After completing this test, the database subgroup started the design of the dashboard to enable the control of the queue.

As it will be mentioned in the project management section, the development of the smartphone application required more time than what initially planned. The encoding of QR code using the application was carried out by week 20 and this was again tested independently by generating several QR codes and decoding their content to make sure that this function operates properly. Testing of all modules together was also carried out at the same week confirming the functionality of each module and also that the whole design operates as expected.

Finally, after the mobile application was completed (establishing the communication with database) a final testing was carried out. This included adding a user to the database by scanning the generated by smartphone QR code, automatically displaying this new entry to the dashboard and finally displaying the relevant queuing information of the user to the smartphone screen. The testing was done successfully in week 22 and completed in large degree the prototype design.

The whole procedure was consistent with what was planned in the beginning of the term according to the Gantt chart. The group planned to start developing the prototype by week 20 and making the final tests on prototype design by week 22-23.

6. Project Management:

During the Spring term the group has progressed from the initial research and higher level proposals to a functional prototype that completes the basic required functions that were set up in the interim report. The main tasks that the product had to perform were defined at the beginning of the Spring term, and were:

- Develop a smartphone application that enables the customers to check in at the store and returns to them relevant queuing information;
- Design a database that would store relevant information for each customer;
- Establish a communication process between the database and smartphone application so that data are exchanged between them.

As mentioned in the Interim Report, the group was split into two subgroups, one responsible for the development of the application and one for the database. These two groups worked mainly independent following the Gantt chart outline and time scales that were set at the beginning of the term (please refer to the appendix for the complete Gantt chart). Both subgroups worked separately and the whole group had weekly meetings in order to update all the members. The group didn't come across any management issues nor personal problems between members of the group and this made the project run smoothly and enjoyably for the whole team.

The two subgroups tried to follow a modular design by splitting the whole design to modules and setting weekly deliverables. For example, the application subgroup divided the complete operation of the app into two main tasks, one for generating the QR code and the other one for the connection with the database. This enabled the group to develop some extra functions that were not set initially. For instance, the group managed to design a QR decoder using a simple camera instead of using an already existing QR scanner. Moreover, the database subgroup developed a dashboard which provides a user-friendly environment to the manager of the queue. Finally, the application subgroup developed an application for both Android and iOS, even though the initial target was to design only for Android. The team achieved all of these tasks by following the set timescales. The development of the application required more time than what was estimated at the beginning. This was mainly due to the unfamiliarity with languages like Swift and Java. Nevertheless, the group managed to have a working prototype by the end of week 22 (i.e 5th of March).

7. Future Work

In order to make eQueue a complete and competitive product, the group has looked into future development areas.

One of the main future features that the group would like to develop is to provide suggestions regarding nearby places that the customer can access according to how much time the customer has to wait in the queue. This will be an innovating improvement since no other similar product provides this functionality (for reference, please refer to section 2 - Competitors&Innovation, or to Appendix B). It will also allow to make contracts with different type of companies like coffee shops, restaurants or fast food shops which will be included and recommended among the suggestions that the application will provide. Implementing this feature will provide important funding for further improvement of the product since the advertised brands will be required to pay an annual contract fee. Specifically, the group started integrating Google Maps and location properties after the basic operation was completed. However, due to lack of available time and since other parts of the project had to be completed it was decided not to include this feature in the prototype.

Additionally, the application should allow the user to create a unique personal account. This feature will allow a quicker and easier sign-in since the customers will not have to provide details at every check in. It will also add user customisation potential and improve privacy and security. Moreover, the user would be able to be registered in two separate queues in two different places at the same time. Obviously, the app will have to prevent anomalous behaviour and system abuse, such as users entering and leaving the queue continuously, or users being registered in a too large number of queues. Finally, the app should upload only the necessary information in the database, based on the type of queue that the user is currently in. Most of these features are implementable by improving and changing the application software.

In later versions of the project, the application can be made available in Apple App Store and Google Play services for iOS and Android operating systems respectively.

8. Conclusion

This report has presented and described eQueue, the product that the group managed to design and prototype during the duration of the second year project.

The product has been designed following strict constraints in order to make it interesting and competitive in a market where lots of similar products exist. The group had a careful attention for the market side during the design process, since the ability for the product to compete in the market is the key to success. The concept development proved that the idea is feasible and has the potential of being successfully developed. A working prototype has been done, and it effectively demonstrates the viability of eQueue.

Moreover, this project had a “personal development” side associated: each member of the group had the possibility to challenge and broaden their knowledge, and to give their active contribution to the success of the project. Even though the process often revealed to be hard, the group has learnt how to effectively split tasks and work as a team in order to transform an idea into a working prototype.

In conclusion, the group is satisfied with the work done and believes that many people can benefit of the improvements that eQueue may bring in the future.

9. References

Bhat, U. (2008). *An Introduction to Queueing Theory*. Boston, Mass.: BirkhaÌˆuser.

Developer.android.com. (2016). *Transmitting Network Data Using Volley | Android Developers*. [online] Available at: http://developer.android.com/training/volley/index.html [Accessed 12 Mar. 2016].

Developer.android.com. (n.d.). *Dashboards | Android Developers*. [online] Available at: http://developer.android.com/about/dashboards/index.html#Platform [Accessed 12 Mar. 2016].

Developer.apple.com. (n.d.). *App Store - Support - Apple Developer*. [online] Available at: https://developer.apple.com/support/app-store/ [Accessed 12 Mar. 2016].

Giridharadas, A. (2010). *Getting in (and Out of) Line*. [online] Nytimes.com. Available at: http://www.nytimes.com/2010/08/07/world/asia/07iht-currents.html?_r=0 [Accessed 12 Mar. 2016].

Laszlo, L. (2011). *LazarSoft/jsqrcode*. [online] GitHub. Available at: https://github.com/LazarSoft/jsqrcode [Accessed 12 Mar. 2016].

Maister, D. (1985). *The Psychology of Waiting Lines*. [online] Davidmaister.com. Available at: http://davidmaister.com/articles/the-psychology-of-waiting-lines/ [Accessed 12 Mar. 2016].

Netmarketshare.com. (2016). *Operating System Market Share*. [online] Available at: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1 [Accessed 12 Mar. 2016].

PosteApp. (n.d.). *Le applicazioni ufficiali di Poste Italiane*. [online] Available at: http://www.poste.it/app/ [Accessed 12 Mar. 2016].

Qless.com. (n.d.). *Better Customer Service | QLess*. [online] Available at: http://www.qless.com/why/better-customer-service/ [Accessed 12 Mar. 2016].

Qminderapp.com. (n.d.). *How Qminder helps your business*. [online] Available at: https://www.qminderapp.com/features/ [Accessed 12 Mar. 2016].

Smart-wait.com. (n.d.). *Avoid Waiting Lines with SmartWaitâ„¢*. [online] Available at: http://www.smart-wait.com/index-en [Accessed 12 Mar. 2016].

Stone, A. (2012). *Why Waiting in Line Is Torture*. [online] Nytimes.com. Available at: http://www.nytimes.com/2012/08/19/opinion/sunday/why-waiting-in-line-is-torture.html [Accessed 12 Mar. 2016].

W3.org. (n.d.). *HTML & CSS - W3C*. [online] Available at: https://www.w3.org/standards/webdesign/htmlcss [Accessed 12 Mar. 2016].

Wavetec. (n.d.). *Queue Management Systems & Electronic Queuing Solutions*. [online] Available at: http://www.wavetec.com/solutions/queue-management/ [Accessed 12 Mar. 2016].

Webopedia.com. (n.d.). *What is Application Program Interface (API)? Webopedia*. [online] Available at: http://www.webopedia.com/TERM/A/API.html [Accessed 12 Mar. 2016].

Webrtc.org. (n.d.). *Frequent Questions | WebRTC*. [online] Available at: https://webrtc.org/faq/#what-is-webrtc [Accessed 12 Mar. 2016].

Appendix A

Finalised PDS

1. Performance

The customer has to install the relevant application on his/her smartphone. Provide login details to the application. Scan generated QR code to the local reader.

Reader decodes QR and provides customer details to the company's database. (These include name, surname, phone number and email). Queuing time and position in queue are calculated for each customer by the database.

Smartphone application uses the established communication with the database to provide to the customer queuing time and position in queue. Customer also has the option to leave the queue at any point.

The queue can be managed electronically from a staff member using the designed dashboard.

2. Environment

The use of product is mainly indoor to avoid extreme climate conditions. However it should be able to operate in any type of climate.
Ensure that any components placed outdoors are weather proof. Customers are free to use their mobile application everywhere.

3. Life in Service

The life in service of the product can be potentially infinite, as the product is mostly software based. However, the product will be periodically updated in order to make it compatible with the future version of the mobile phones operating systems. Therefore, the actual life in service of the product cannot be predicted.

4. Maintenance

The software for both smartphone application and database / dashboard will require regular updates and bug fixes in order to ensure operation of system remains at required levels. Additional functionalities will also be developed periodically according to the feedback provided by customers (both users and businesses).

The smartphone application will need to be updated regularly based on the latest released operating system for both Android and iOS.

5. Target Product Cost

The cost of producing the whole product is zero after eliminating the need of buying a QR scanner.
An annual subscription fee will be required from the organisation in order to adopt the product. (Analytical subscription fees are included in the website).

6. Competition

Competition is a very important consideration when it comes to software development and more specifically to developing queue management systems as it plays an integral role in defining what makes our design more desirable as compared to a multitude of existing solutions. This promotes the development of added functionality for the users, in addition to compatibility with different platforms i.e. iPhones, Android based phones, Windows phones, etc...
Below is a list of competitors with a short description of their products:

- Poste italiane, Italian national postal services company

  Poste italiane is the Italian national postal services company, and has developed a downloadable app that allows you to book your place in the queue in advance; if less than 10 people are present, the app

will not allow you to book in advance but ask you to go directly to the post office. It can only be used at Italian post offices and it has a different principle from our idea as the customers book their place in advance rather than scanning the phone at the arrival. Moreover, it has no real life notifications.
Website: http://www.poste.it

- Wavetec

  It is a multinational company specialised in queue management systems. Customers have to wait in the building and are entertained by advertisements. Special displays provided by the company are required: therefore, the cost for the businesses that wants to adopt this system is increased. It does not eliminate the use of paper tickets.
  Website: http://www.wavetec.com

- Qless

  It provides a list of all the queues that one might join. The program provides live updates but doesn't give any suggestions.
  Website: http://www.qless.com

- Qminder

  Downloadable app that tackles the same problem. The customer's information is entered into the queueing system manually. With Qminder the customer is required to stay within the business building to wait for their turn. For more details on this, please refer to the 'Market Research' section.
  Website: https://www.qminderapp.com/

- SmartWait

  Similar to Qminder. It is compatible with any business which has adapted eesii's (the company who owns SmartWait) queue management system. Provides real time information on waiting conditions and the user's ability to postpone their call. The major problem is that 'fake' customers can request a virtual ticket without the intention of ever turning up. For more details on this, please refer to the 'Market Research' section.
  Website: https://www.esii.com/en/smartwaittm

For more information please refer to Appendix B/Market research.

7. Shipping

N/A

8. Packing

N/A

9. Quantity

N/A

10. Manufacturing Facility

N/A

11. Customer

The customer falls into two categories. The organisations that decide to adopt our queue management system. These include but are not limited to banks, government institutions, hospitals, post offices.

Also, the end-user (the organisation's customers) will be taken into account as customers, as they will have to use the app and interact with the system as a whole. This needs to be made into as pleasant an experience as possible.

When taking the organisations into consideration, the system will be designed to provide useful data for the customer such as names, phone numbers, etc… And it will have to make the queue more efficient so as to minimise congestion and to maximise efficiency in terms of resources used like the amount of employees serving the queue.

On the other hand, the app will be shaped to accommodate the end-user who will be queueing by making the app user-friendly and providing the user with added benefits such as suggestions of nearby coffee shops to wait in.

12. Size

N/A

13. Weight

N/A

14. Materials

N/A

15. Product Life Span

Usually the product life span of a product is mainly dependant on its hardware. However, because our product is software based, hardware is a non-issue. In this case the lifespan of product depends on the frequency of software updates which are released. The software updates would fix bugs and also make sure that app is compatible with newer versions of Android's/iOS' operating system to ensure the continued use of the app.

16. Aesthetics, Appearance and Finish

Product exposed to the customers will be a QR reader. Smartphone application and dashboard are designed using a user friendly environment.

17. Ergonomics

N/A

18. Standards and Specifications

N/A

19. Quality and Reliability

Quality is based on the quality of the software design. Software development techniques have been used to ensure product quality is high. For example, modular design, continuity and composability.

20. Shelf Life (storage)

N/A

21. Testing

All individual modules (QR decoder, database, dashboard, smartphone application) have been tested and are fully functional. The whole design has been also tested and is functional.

22. Processes

N/A

23. Time Scale

The time-scale for the current design and prototyping phase is 6 months

24. Safety/Privacy

Product must obey all privacy standards regarding customer/organization personal data exchange. However, this project is not set to design a new encryption method. Nevertheless, the team will ensure that the product meets the required data encryption standards. This will necessitate the use of existing encryption technologies. For example, QR technology has its own degree of encryption.

25. Company Constraints

N/A

26. Market Constraints

Due to the nature of our product (smartphone application), a major market constraint is going to be the phone operating system the user has. The two main ones are iOS and Android. According to statistics 60% of the market consists of phones using an Android and 32% iOS, while 8% uses other operating systems like Windows Phone or BlackBerry (newmarketshare.com, n.d.). So to maximise the compatibility of the app, it would have to be compatible with the two main operating systems. In order to gain a greater insight into the customer's needs, we have conducted a survey which mainly focused on what kind of additional features for the app they were interested in. The results showed that 68% of people interviewed would like to receive live updates on a business's waiting times. Furthemore, 37% of people were interested in receiving suggestions of things they can do around the area based on their waiting times. (Survey results are included in interim report).

27. Patents, Literature and Product Data

Relevant existing patents:

- http://www.google.com/patents/US20100317377
- http://www.google.com/patents/US20050271199
- Pat. No. WO2012170958 A1: "A method for managing virtual queues adapted to correspond to actual queues". Differences from our design: registration to the service via web or by phone call; the customer interacts with the system via text messages.
- U.S. Pat. No. 6,529,786 to Sim describes a queue management system which require special portable modules designed by Sim, different from mobile phones.
- U.S. Pat. No. 6,748,364 to Waytena et al. : same difference as the previous design
- U.S. Pat. No. 6,889,098 to Laval: the difference with our design lies in the fact that this design builds two different queues to the attraction based on whether a customer uses the electronic system or not. (https://www.google.com.ar/patents/US6845361)

To correctly decide whether this system is applicable for a patent, a professional opinion such as that of a lawyer's will be required as the details of existing patents need to be studied properly.

28. Legal

Application will obey all privacy standards regarding personal information shared to the company.

29. Political and Social Implications

None of the product features can create social unrest or upset, but we believe that our system can improve the way queues are managed at the moment and therefore have a positive influence on society.

In particular, the product will:

- Allow customers to use their time more effectively, without wasting time at queues
- Make the service time shorter, as the application will provide relevant information to the company database about the customer on check in.
- Prevent crowding of inside spaces, thus making emergency procedures easier and more effective.

30. Installation

The application will have to be downloaded to smartphones. This process should not take longer than a couple of minutes.

Basic information from the user can be collected for commercial purposes and to help improve the service that is provided.

Hardware to communicate with portable devices is necessary to be installed onsite (QR scanner). This depends on the decision that the adopting organisation will make but in general a simple camera can be used with the provided software.

Local reader will need to be connected to a local computer that controls data flow and the customer information.

31. Documentation

Installation and user guide for customers will be available through our website

32. Disposal

N/A

Appendix B

Market Research

The competitors analysed in section 2 of the report will be compared below.

Name	Wavetec	Comments
How it works	- The user needs to install the app on their device - The business need to install all the relevant equipment (including kiosks) - The app is not the main part of the system, but just a possibility that can be adopted	- This solution is particularly versatile, as each business can adopt a different solution (entertaining kiosks, app to book the place in the queue…)
Main features	- The user can select a nearby branch and generate a ticket based on their type of activity - The idea behind the company is to provide entertainment to customers while they are queueing - The app can provide real time information about the queue	- What if there are not nearby branches? - Providing entertainment does not really solve the queueing problem, but just makes the customers feel like the time is passing faster
OS compatibility	iOS and Android	
Price	Not specified	
Website	http://www.wavetec.com	

Name	Ufficio Postale	Comments

How it works	- The customer has to download the app - The customer has to register online providing their National Insurance Number - The customer can book their time slot, and need to check in at arrival	- Only Italian residents can use this app, and only in national postal offices - The customer can book their time slot, thus does not need to go directly to the office, but there is not the possibility of tracking the state of the queue
Main features	- Simple booking system - Lots of additional features, but only relevant for a post office	- The app is very easy to use, but does not provide information about the queue - The app is significant only for one situation, and cannot be adapted to different organisations
OS compatibility	iOS and Android	
Price	Free	
Website	https://play.google.com/store/apps/details?id=com.posteitaliane.spim&hl=it	

Name	Qless	Comments
How it works	- The business collects the phone number of its customers into a database, and use these to contact the customer about the state of the queue - The customer does not need to download any app	- Very adaptable solution, as it works via SMS, thus no smartphone is required - The business keeps personal data of the customer (phone number associated with name)
Main features	- Allows to track the state of the queue - The customer can request additional time - Trends are outlined using statistics collected from the queues - The businesses carry out a targeted marketing	- It is very easy for the business to adapt the features to the type of business - The targeted marketing can result unpleasant, and some people may prefer queueing rather than receiving spam via SMS
OS compatibility	Any	It uses SMS
Price	Not specified	
Website	http://www.qless.com	

Name	Qminder	Comments

How it works	- Cloud based service - The business owner has to download the app to their iPad (only iOS compatible) - Sync the iPad with the Qminder dashboard in his web browser by entering a 4-digit code from the app - Personalise the customer interface e.g. how are the customers going to be addressed? By name or number?	- Advantage of a cloud based service: - Data is backed up in the cloud - In the case of a business with multiple branches, all the data will be in one place - Disadvantage of cloud based service: - How safe is the data? This would be especially important if the business are banks. - Disadvantage of OS compatibility: - It is only compatible with iOS so if a business wants to use their service, they need to have an iPad
Main features	- Call customers out of order or skip of needed and address them by their name if they have given one - Multiple service lines i.e. different queue for different types of services - Remote access of the queue management system - Track customer traffic which includes: average hourly visits, daily visitors and customers served. - Track customer waiting times	- The customer's information still has to be manually entered. If the business is very busy a queue might form in front of the iPad. - The use of multiple service lines is a good idea and makes the product more flexible - The ability to track customer traffic is a very useful additional feature which allows a business to judge their performance.
OS compatibility	iOS	
Price	$249 per location/month	
Website	https://www.qminderapp.com/	

Name	SmartWait	Comments
How it works	- Download the app onto smartphone	
Main features	- Virtual queueing – Replace the paper queueing ticket with a virtual one your phone - Appointment booking – additional feature besides virtual ticketing - Real time information on waiting conditions - Postpone their call – The customer can postpone their call if they are not available	
OS compatibility	iOS and Android	

Price	Free	The app for the customer is for free, however the business has to purchase the software which is compatible with the app. The price for the software is not listed on the website
Website	https://www.esii.com/en/smartwaittm	

Final Comparison

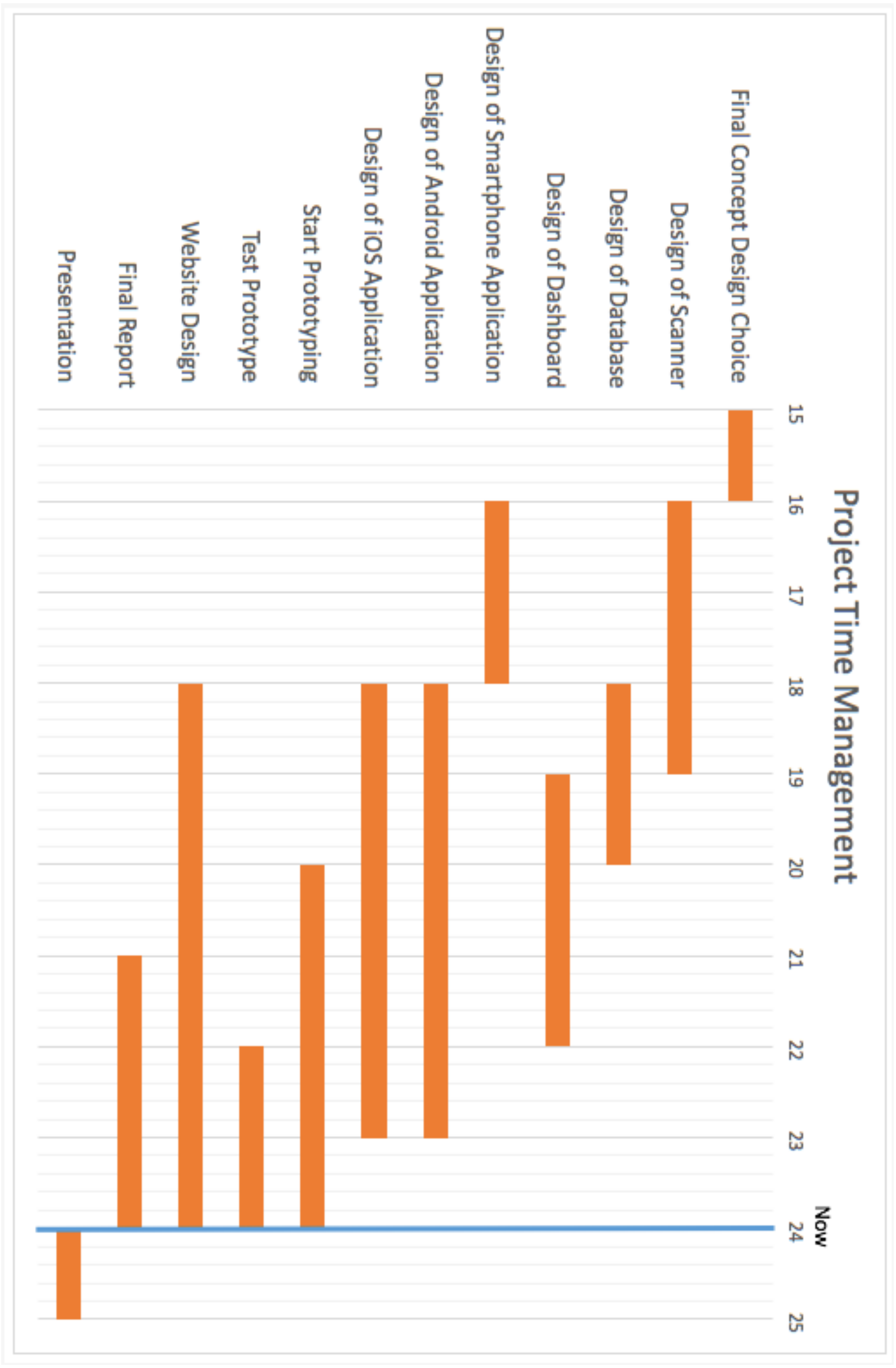
Below, a table that compares the products just described is provided. From the comparison, it is possible to see the Qless is the best product in the category, which is also the most similar to our design.

	Weight	Qminder	SmartWait	Poste Italiane	Wavetec	Qless
Compatibility	8	5	5	6	6	10
User-friendliness	7	4	7	6	7	7
Adaptability	7	6	6	6	8	7
Complexity	8	7	4	1	8	8
Safety	8	6	8	9	7	4
Additional features	5	8	5	9	8	8
Total score		254	252	257	313	314

Appendix C

Project Management

Here, the Gantt Chart used to set deliverables and manage the project.



Below, the minutes taken at each group meeting.

Database group	App Group
Minutes – 22.01.2016 Present: All <u>Time: 10.00 – Meeting starts</u> • Discussion on how the app is going to interface with the database. • Established that the input of the database will be a QR signal - Further research is required on what this exactly means - The database will store: Customer name, phone number and unique customer ID • How will the information be send back to the customer? - Push notifications (Requires further research) - Information send back has to contain (1) the state of the queue and (2) the current waiting time <u>Time: 10.45</u> Discuss the type of milestones that have to be set and their respective dates (create a Gantt chart) <u>Time: 11.00 – Meeting ends</u> **Action Points:** • Barnabas: Find out how to send info from the database back to the device. • Arisa: Research how exactly QR works • Rami and Kaan: Figure out how to use the QR library (C ++) and make it work. <u>Next meeting: 28.01.2016</u>	**Minutes – 20.01.2016** Present: All <u>Time: 15.30 – Meeting starts</u> Goal definition: • Develop a simple app that allows the user to insert their personal information (name, phone number, email etc.) • Send this information to the database (of the other group) • The app has to be able to receive real time notifications regarding the state of the queue (The software for this, will also be provided by the other group) <u>Time: 16.00</u> Split final goal into smaller deliverables: • Research how to write an Android App and download the necessary software (Android Studio) • Research how to connect the app and the other software • Development of the app code • Connection between app and software and testing <u>Time: 16.30 - Meeting ends</u> **Action Points:** All: Research how to write an Android App and download the necessary software (Android Studio). Start writing the sign-in page. <u>Next meeting: 03.02.2016</u>
Minutes – 28.01.2016 Present: All <u>Time: 10.00 – Meeting starts</u> • Update everyone on the research done • Rami and Kaan: Using C++ (for the database) might not be possible and an alternative has to be found • Barnabas: Possibility of using a PHP plugin that is linked to WordPress • Arisa: List of open source applications which can be used to decode the QR code <u>Time: 10.45 – Meeting ends</u> **Action Points:** • Barnabas: Continue with own research topic • Arisa: Determine the most suitable application for our project and also how to integrate it to the database • Rami and Kaan: Find an alternative for the database and a method to implement this <u>Next meeting: 10.02.2016</u>	**Minutes – 03.02.2016** Present: All <u>Time: 15.30 – Meeting starts</u> • Discussion of difficulties encountered: • None of the group members did manage to have working a sign-in page, due to the fact that no one is familiar with Java. • Re-evaluate approach <u>Time: 15.45</u> • Break down tasks even further: • Create an app for both OS (iOS and Android) [Note: iOS uses Swift and Android uses Android Studio] • Research about the possibility of doing sign in via Google Sign in and about the connection with the program that the other group is developing <u>Time: 16.30 - Meeting ends</u> **Action Points:** • Artem: Create the iOS app • Adamos: Create the Android app • Francesca: Carry out the research <u>Next meeting: 16.02.2016</u>

Minutes – 10.02.2016
Present: All
Time: 11.00 – Meeting starts
- Update everyone on the research done
- Rami and Kaan: Use of MySQL (for the database) and PHP to link MySQL to website
    - A local host was used to simulate the website
    - Get the server space provided by the department working, so that everything can be moved over
- Demonstration on what both of them have done so far
    - Barnabas: Possibility of using a PHP plugin that is linked to WordPress which is linked to the app
    - Discuss this with the app group

Time: 12.00 – Meeting ends
Action Points:
- Barnabas: Sort out the server space assigned by the department
- Rami and Kaan: Complete the prototype, ready for the big group meeting with the whole group so that it can be integrated with the app

Next meeting: 24.02.2016 (Whole group meeting)

Minutes – 16.02.2016
Present: All
Time: 16.00 – Meeting starts
- The group members update each other:
- Artem and Adamos made progress with the app
- To connect the app and to other modules, PHP query is needed.

Time: 16.30 - Meeting ends

Action Points:
- Artem and Adamos: Continue with the development of the app
- Francesca: Research about the PHP language and talk to the other group for this purpose.

Next meeting: 24.02.2016 (Whole group meeting)

Whole Group

Minutes – 24.02.2016
Present: All
Time: 11.00 – Meeting starts
- Demonstration of the database by the database sub-group
- Demonstration of the app by the app sub-group
- Discussion on the integration of both

Time: 11.45
Brain storm on the general layout of the website (content and structure)
Time: 12.15 – Meeting ends
Action Points:
- Rami, Adamos and Artem: Work on the integration of the database and the app
- Kaan: Work on the fixed scanner (raspberry pi)
- Francesca, Barnabas and Arisa: Create some design layouts for the website

Next meeting: 02.03.2016

Minutes – 02.03.2016
Present: All
Time: 11.00 – Meeting starts
- Successful demonstration of using the Android app with the Web app (i.e. dashboard)
- Successful demonstration of using the iOS app with the Web app (i.e. dashboard)

Time: 11.30
Discussion on how the website will look like and its content.
Time: 12.00
Kaan updates everyone on the progress made with the raspberry pi.
Time: 12.15 – Meeting ends
Action Points:
- Barnabas: Continue working on the website so that the basic structure is going to be ready for the next meeting.
- Kaan and Rami: Work on the scanner
- Francesca and Arisa: Start working on the general outline of the final report. Have a clear idea on what will be needed for the report and how the work is going to be split up.

Next meeting: 07.03.2016

Minutes – 07.03.2016
Present: All
Time: 11.00 – Meeting starts
- Assign respective parts for the report
- Refer to the word document ‘Structure’ on Google Drive for their parts

Time: 11.30 – Meeting ends
Action Points:
All: Write their respective part of the report. Ready to be compiled on the 11th of March.

Note: All the code is available on request.