Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Bloodhub.

High-Level Design Report

Mustafa Culban, İzel Gürbüz, Erdem Karaosmanoğlu, Mehmet Orçun Yalçın

Supervisor: Uğur Güdükbay
Jury Members: İbrahim Körpeoğlu, Özgür Ulusoy

Progress Report
Dec 22, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Introduction

Humankind has become the leader race of earth and got top of food chain with indisputable helps of their intelligence and technology related to it. Technology is a controversial issue because while it gives numerous blessings to humanity, it also took a lot from them. Violence, battles, fights, diseases have always been in history of humankind but with developing technology their harms have significantly increased.

Traffic accidents, surgeries, gun battles cause thousands of wounding every day. Naturally, a considerable amount of people need blood donation. Ironically, these problems are solved with technology again. Technology can take an important place in solutions to blood donating problems. With social networking voluntary blood donors and people who need blood can meet under a social media platform.

With invention of Facebook, Internet started to use its potential. People started to interact others while connected to Internet. After Facebook, “social network” has become a phenomenon. Different websites and applications followed this trend. All of these social platforms created a new power which is called as “social media”. Today, even governments forced into consider social media. Social platforms on Internet are used by millions of people, so people use it for their urgent needs too. During emergency situations users announce need for blood for their relatives or friends. With this way, announcement is heard by many people and it makes easy to find blood. Making an application just for this job can save lives.

Purpose Of the System

BloodHub aims to bring many easiness to blood donation process. There will be two types of users. First one is the users who want to donate their blood. They can see available places to donate their blood from interactive map of application. They will be notified when there is a need for blood. Application will send an notification alert to their phone. Users from the near of place where there is a need for blood will be alerted. Their location will be taken with their phone's location services. Second type of users are people who look for blood for their friends, relatives or themselves. They can interact with volunteers by using BloodHub.

For the patients who are unconscious and unable to use app, there will be tutelage system. Prioritized assigned people can use patient's account to search for volunteer. They can send notification to users who located nearby them or directly communicate them by nearby users' communication informations.

We aim to supplying blood to answer all expectations. There are many humanitarian people to donate their blood for the ones that they never met in their entire lives. We want to make processes easier for these generous people.

Design Goals

Usability

BloodHub should be easy to use in daily life and understandable enough to use without a user guide. Usage of functionalities such as sending notification , should be apparent for users in every condition to be useful even in an emergency.

Reliability

BloodHub should display an understandable error message and inform the developer when a failure occurs.. Also , no kind of failure causes loss of recorded data or reveal of protected personal information.

Security

Private personal information should be protected from both other users and third party applications . Also these private information should be saved as crypted to make invisible to even system itself.

Performance

Fast response and efficiency are very important for BloodHub. BloodHub is designed to answer people’s vital blood needs in every condition that even in an emergency. Thus , it must provide fast response time especially in terms of sending and receiving notifications.

Portability

BloodHub should work on various Android versions and web platform to reach maximum number of people possible.

Supportability

The developers should receive any comments and feedbacks of users via the application.

Definitions

Laravel: Laravel is a PHP framework for the software development.

API: Application Programming Interface.It specifies how software components interact and are used when programming graphical user interface components.

REST: Representational State Transfer. It is a trassfer style from client to the server or visa versa. It is sused for the comminication of our API service between clients and the server.

SOAP: Simple object Access Protocol . It is a protocol that is used for sending data to the server. It is like REST service.

MD5: It is a one way encription protocol. There can be only the hashed valus after the usage of this protocol. Real data cannot be returned after encription.

MySQL: It is a database structural query language design for the storage of the data into database.

Overview

Bloodhub aims to help users that need a blood transfusion. BloodHub is powered by Android for mobile and powered by Laravel for web services. Users who upload Bloodhub can see available health centers and drivers to donate blood from map of application. In this way, time for blood transfusion is reduced in case of emergency. Furthermore, users can be notified when there is a need for blood in 100 km diameter. Thus within the application Bloodhub, we meet constant blood supply and urgent needs of patients. We have a reason we choose mobile application is because apps have become integral parts of all of the people and we aim to reach users with at all ages.

Current Software Architecture

According to our research on other similar technologies and products , there are few applications developed with aim to find an easy solution for blood donation. However , none of these applications have strong security for users and this kind of delicate process. BloodHub checks and eliminates invalid registers and usage by checking citizenship number and also system protects sensitive personal information from even itself. Also we offer so much easier and faster way of informing other users about blood need which is notifications. Similar product called ‘Blood Donor’ has an appointment system which means very slower process than ours. Moreover , BloodHub offers an extra emergency precaution by letting users to make a connection up to five people. This feature called Emergency Five , lets users share their information with each other. It is very crucial feature especially in case of emergency. Other applications don’t have any feature to handle emergency needs.

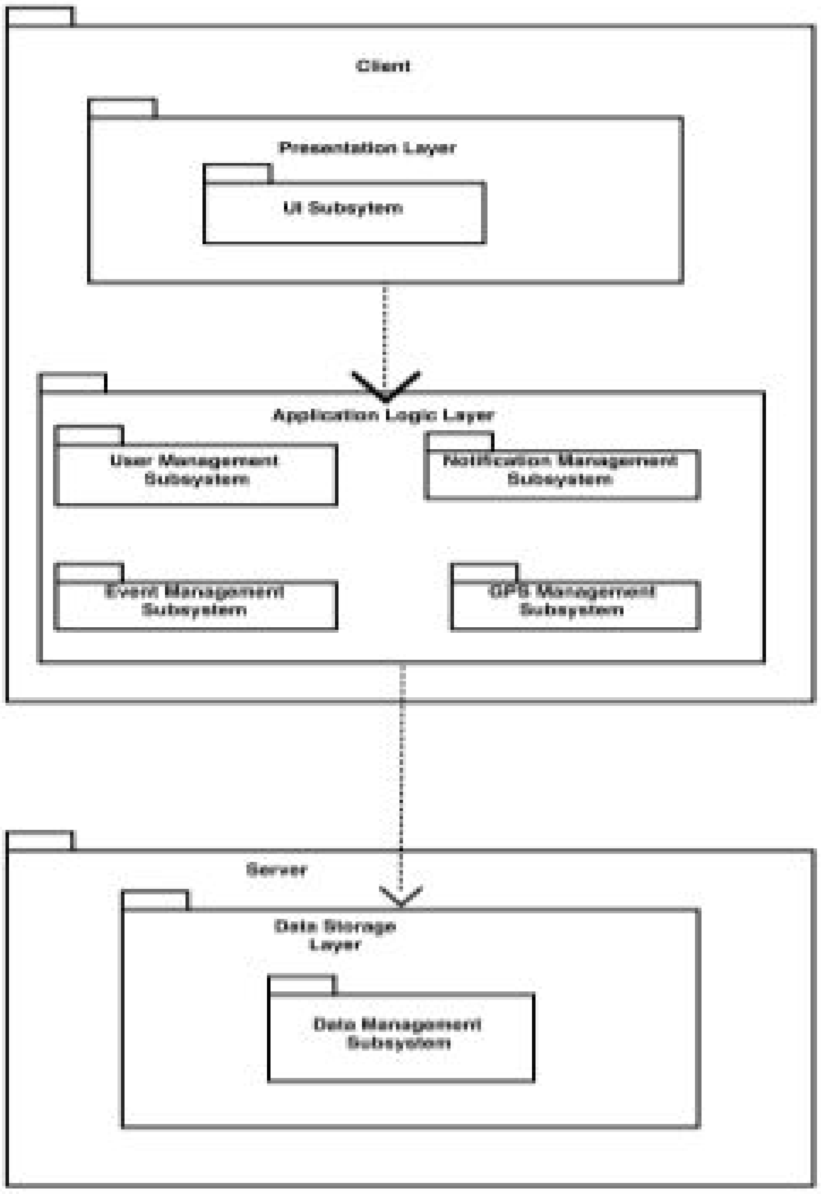
Proposed Software Architecture

Overview

We have client server model that each of the clients are interacting with the server in order to fetch the data, send the request and many other feautures. Server side of the system will be built on the PHP with using the Laravel Framework in order to make it secure Laravel supplies many features for that purpose. Client applications will be on the Aboth Android and the iOS . There will be API which will be supplied both android and iOS for taking all the processes user possibly can do to the server side. Server side of the program will be the helper and also the core for Bloodhub. We will have client system that will use these API connections to do the requests of the end users. We seperated into 2 parts as described, one for the server side that handles all the request and the other is only the request side/ client side in order to get the simplicity, to reduce the complexity and to make Bloodhub more managable.

Subsystem Decomposition

We preferred 3-tier architecture style which is a client-server architecture for our project. This architecture is the most suitable one for our system because it allows process logic , data access , data storage and UI to be developed as independent modules on different platforms.


Client
Presentation Layer
UI Subsytem
Application Logic Layer
User Management Subsystem
Notification Management Subsystem
Event Management Subsystem
GPS Management Subsystem
Server
Data Storage Layer
Data Management Subsystem

Presentation Layer

Presentation Layer contains User Interface Subsystem. This subsystem provides UI and communicates with the layers below. This layer assures sustainability and usability.

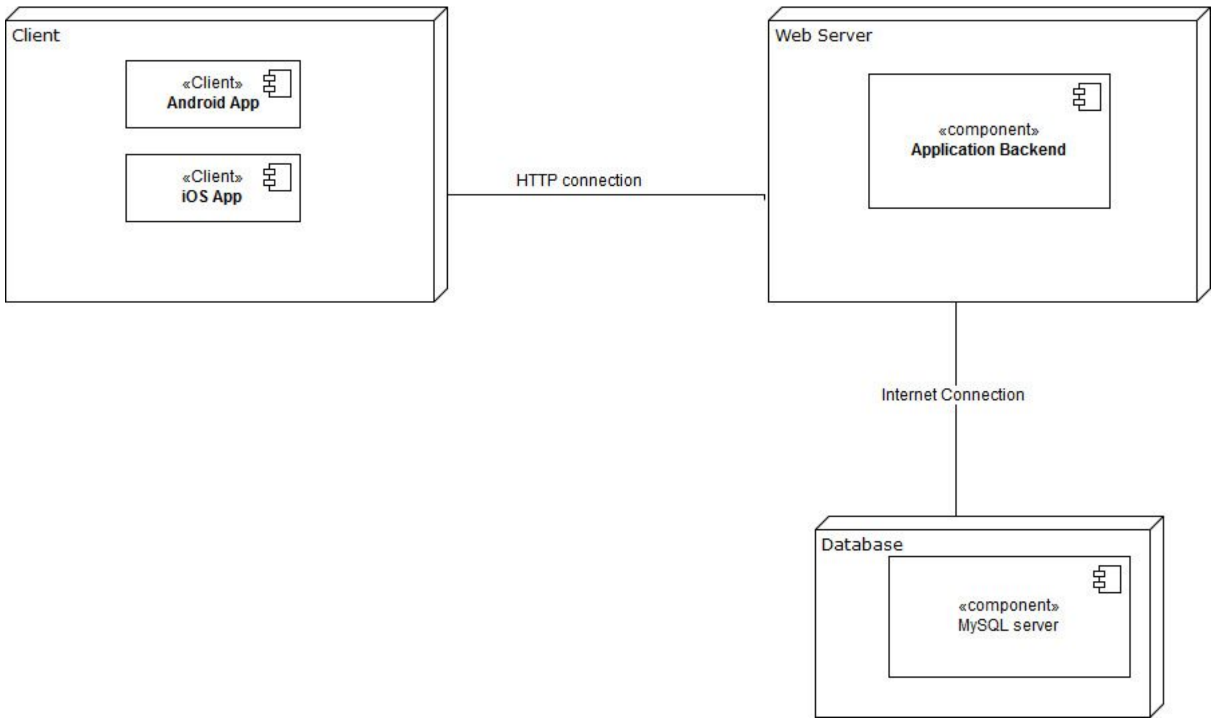
Application Layer

This layer is where application functionality is controlled. It is consists of User Management Subsystem , Notification Management Subsystem , Event Management Subsystem and GPS Management Subsystem. User Management Subsystem deals with user actions and transactions of the modifications results of these actions. Notification Management is responsible for handling transfer of the notifications. Event Management Subsystem processes event changes and informs users about current events. GPS Management Subsystem provides directions to hospital/blood center locations.

Data Storage Layer

This layer includes Data Management subsystem that stores hospital/blood center data , user data and notification records. Data Storage Layer communicates with layers above for data transfer and storage.

Hardware/Software Mapping



Our application will be developed for the Android and iOS operating systems therefore our users should use the Google Play Store or App Store to get BloodHub. For using BloodHub the Android device or iPhone which our user uses should have HTTP connection . BloodHub will use a database server to keep user and location information of traveling teams of Kızılay.. We will run a MySQL server to serve our services to users. Our server and database will be running constantly.

Persistent Data Management

Our application BloodHub depends on database and a persistent database is important for our application. The database will keep the all information about users like blood type, name, address etc. and all the location information will be kept in database and we use this information when a blood need request is made by any user. Database should work as fast as possible to decrease waiting time of our users.

Access Control and Security

To use BloodHub, all users must be register. No user can use the properties of our application without register and login. Users should have their own unique password,username, name, surname, email, identity number, blood type, birth date, open address for registering and so that users can enter application using them via "Sign In" page. Confidential information should not be shared with third party applications. User’s credentials should be encrypted with MD5 One-way Encryption Protocol. No user can be able to see other users’ personal information.

After we match their password and Email address with our database, we will let them use our application.

Global Software Control

BloodHub is a client-server application. In client side, there are two mobile applications one for Android and one for iOS. On the server side there are OneSignal API for sending web push notifications, and SOAP services from the KPSPublic for the identity check. There is a constant data flow between client and server sides. Since the data flow starts with the user logs into the system, depends on user inputs. After that available health centers and drivers information are processed at the backend and according to the user’s location information are made to the user. If a change occurs in health centers or drivers’ locations, new location information will be sent to users.

Boundary Conditions

Boundary conditions have three subheadings.

- Initialization
- Termination
- Failure

Initialization

In this step, it shows how the system starts.

- When the user enters our application for the first time they will be welcomed with a login page which contains a “sign in” and “register” button.

- If user already has an account they will enter their email and password and hit the “sign in” button.

- If they have not an account they can hit the “register” button and can create an account via filling the required spaces and press the “sign-up” button

- After they sign in successfully our home page will be shown and the user can use our application and access information about how to donate a blood.

Termination

In this step, it shows how the system is terminated.

- A user can log off from the system anytime by hitting the “Log out” button at the pull-down menu.
- After that user has to login to the system again to use the application.
- A user can also directly close the application without logging out.

Failure

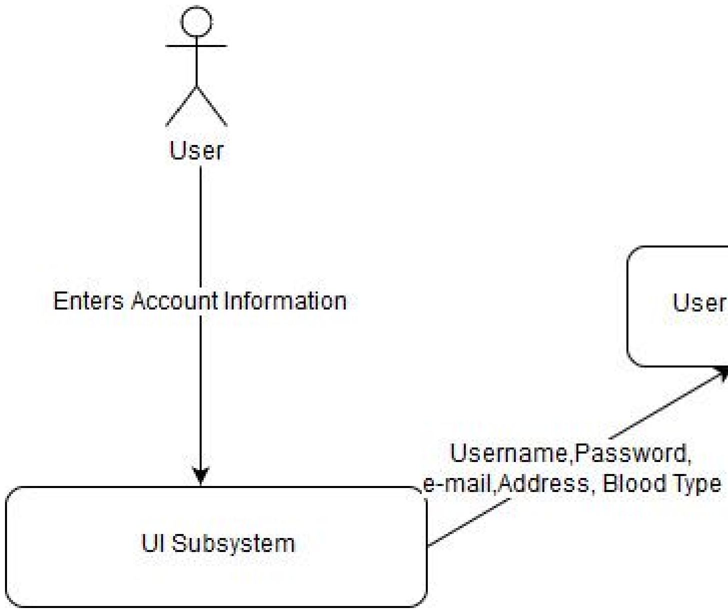
This part shows how the system reacts to the failures that occur while using it.

- Internet connection is required to use BloodHub if the connection is not available failure may occur.
- If a user does not update its application for a long time, BloodHub system cannot perform on that user’s and failure may occur.

Subsystem Services

UI Subsystem

This subsystem provides connection between user and other subsystems in Application Logic Layer. It shows information from other subsystems. It is one of the most important subsystems.


User
Enters Account Information
User Management Subsystem
Username,Password,
e-mail,Address, Blood Type
UI Subsystem

User Management Subsystem

User information is managed in this subsystem. Login and sign up processes are done in this subsystem. Additionally, user actions are interpreted in this subsystem. User’s commands and the changes they leaded are delivered to other subsytems which are at the same layer or at other layers.

We started to implement the code of this part. User can successfully login and sign up to the system via the BloodHubAPI. Below is the implementation so far:

For Login System:

```
if ((isset($_GET['email']) && isset($_GET['password'])) ||
(isset($_POST['email']) && isset($_POST['password']))) {

                  $email = '';
                  $password = '';
                // receiving the post params
                if(isset($_GET['email'])){
                  $email = $_GET['email'];
                  $password = $_GET['password'];
                  }
                  elseif(isset($_POST['email'])){
                        $email = $_POST['email'];
                  $password = $_POST['password'];
                  }

                // get the user by email and password
                $user = $db->getUserByEmailAndPassword($email,
$password);

                if ($user != false) {
                    // use is found
                    $response["error"] = "FALSE";
                    $response["user"]["id"] = $user["id"];
                    $response["user"]["username"] = $user["username"];
                    $response["user"]["firstname"] = $user["firstname"];
                    $response["user"]["surname"] = $user["surname"];
                    $response["user"]["email"] = $user["email"];
                    $response["user"]["bloodType"] = $user["bloodType"];
                    $response["user"]["birthdate"] = $user["birthdate"];
                    $response["user"]["address"] = $user["address"];
                    $response["user"]["telephone"] = $user["telephone"];
                    $response["user"]["available"] = $user["available"];
                    toXML($response);
                }
                else {
                    // user is not found with the credentials
                    $response["error"] = "TRUE";
                    $response["error_msg"] = "Login credentials are
wrong. Please try again!";
                    toXML($response);
                }
            }
            else {
        // required post params is missing
                $response["error"] = "TRUE";
                $response["error_msg"] = "Required parameters email or
password is missing!";
                toXML($response);
            }
```

Once user logins to the system, output will be formated in JSON for both iOS and the Android OS.

```
{"error":"FALSE","user":{"id":1,"username":"mustafaculban","firstname":"
Mustafa","surname":"Culban","email":"mustafaculban1@gmail.com","bloodTyp
e":"AB+","birthdate":"16041995","address":"denemeAdress","telephone":"53
96768149","available":1}}
```

If user login is failed because of the missing parameters or the wrong credentials, BloodHubAPI will display the error message in JSON format.

```
{"error":"TRUE","error_msg":"Login credentials are wrong. Please try
again!"}
```

Below is for the Registration System:

Once user wants to register to the system, the code below handles this registration process.

```
if (isset($_GET['username']) && isset($_GET['firstname']) &&
isset($_GET['surname'])&& isset($_GET['password'])&&
isset($_GET['email'])&& isset($_GET['identityNum']) &&
```

```php
isset($_GET['bloodType']) && isset($_GET['birthdate']) &&
isset($_GET['address'])  && isset($_GET['telephone'])) {

                // receiving the post params
                $username = $_GET['username'];
                $firstname = $_GET['firstname'];
                $surname = $_GET['surname'];
                $password = $_GET['password'];
                $email = $_GET['email'];
                $identityNum = $_GET['identityNum'];
                $bloodType = $_GET['bloodType'];
                $birthdate = $_GET['birthdate'];
                $address = $_GET['address'];
                $telephone = $_GET['telephone'];

                if
(!TcKimlikNoSorgula::tcKimlikNo($identityNum)->ad(($firstname))->soyad((
$surname))->dogumYili(substr($birthdate, -4))->sorgula() /*  ||  there
will be yabanci kimlik no check*/ ) {
                     $response["errorr"] = TRUE;
                    $response["error_msg"] = "This identitiy Number is
fake";
                    toXML($response);
                }
                // check if user is already existed with the same email
                else if ($db->isUserExisted($email)) {
                    // user already existed
                    $response["errorr"] = TRUE;
                    $response["error_msg"] = "User already existed with
email: " . $email. "and username: " .$username;
                    toXML($response);
                  }
                else {
                    // create a new user
                    $user = $db->storeUser($username, $firstname,
$surname, $password, $email, $identityNum, $bloodType, $birthdate,
$address, $telephone);
                    if ($user) {
                        // user stored successfully
                        $response["errorr"] = FALSE;
                        $response["user"]["id"] = $user["id"];
                        $response["user"]["username"] =
$user["username"];
                        $response["user"]["firstname"] =
$user["firstname"];
```

```
                        $response["user"]["surname"] = $user["surname"];
                        $response["user"]["email"] = $user["email"];
                        $response["user"]["bloodType"] =
$user["bloodType"];
                        $response["user"]["birthdate"] =
$user["birthdate"];
                        $response["user"]["address"] = $user["address"];
                        $response["user"]["telephone"] =
$user["telephone"];
                        $response["user"]["available"] =
$user["available"];
                        toXML($response);
                    }
                    else {
                        // user failed to store
                        $response["errorr"] = TRUE;
                        $response["error_msg"] = "Unknown error occurred
in registration!";
                        toXML($response);
                    }
                }
            }
            else {
                $response["errorr"] = TRUE;
                $response["error_msg"] = "Required parameters are
missing!";
                toXML($response);
            }
```

There is also an added class called TCKimlikNoSorgula which handles verification of the entered TC Identitiy number or Foreigner Identitiy Number.

```
<?php
class YabanciKimlikNoDogrula {
    private $kimlikNo;
    private $ad;
    private $soyad;
    private $dogumGun;
    private $dogumAy;
    private $dogumYil;
    public static function kimlikNo($kimlikNo) {
        $instance = new static;
        if(strlen($kimlikNo) !=11)
            throw new \Exception('Foreigner Identity Number Should be 11
characters');
        $instance->kimlikNo = $kimlikNo;
        return $instance;
    }

    public function ad($ad) {
        $this->ad = $this->upperCase($ad);
        return $this;
    }

    public function soyad($soyad) {
        $this->soyad = $this->upperCase($soyad);
        return $this;
    }

    public function dogumGun($dogumGun) {
        $this->dogumGun = $dogumGun;
        return $this;
    }

    public function dogumAy($dogumAy) {
        $this->dogumAy = $dogumAy;
        return $this;
    }

    public function dogumYil($dogumYil) {
        $this->dogumYil = $dogumYil;
        return $this;
    }

    private function upperCase($string) {
        return mb_convert_case($string, MB_CASE_UPPER, "UTF-8");
    }

    public function sorgula() {
        if (!isset($this->ad) || !isset($this->soyad) ||
!isset($this->dogumGun) || !isset($this->dogumAy) ||
!isset($this->dogumYil) || !isset($this->kimlikNo )) {
            throw new \Exception("To verificate information, Foreigner
```

```php
Identity No, name, surname, birth day, month and year should be
declared");
        }
        $toSend =
            '<?xml version="1.0" encoding="utf-8"?>
                <soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
                  <soap12:Body>
                        <YabanciKimlikNoDogrula
xmlns="http://tckimlik.nvi.gov.tr/WS">
                            <KimlikNo>' . $this->kimlikNo . '</KimlikNo>
                            <Ad>' . $this->ad . '</Ad>
                            <Soyad>' . $this->soyad . '</Soyad>
                            <DogumGun>' . $this->dogumGun . '</DogumGun>
                            <DogumAy>' . $this->dogumAy . '</DogumAy>
                            <DogumYil>' . $this->dogumYil . '</DogumYil>
                        </YabanciKimlikNoDogrula>
                    </soap12:Body>
                </soap12:Envelope>';
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL,
"https://tckimlik.nvi.gov.tr/Service/KPSPublic.asmx");
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_POST, true);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
        curl_setopt($ch, CURLOPT_HEADER, FALSE);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $toSend);
        curl_setopt($ch, CURLOPT_HTTPHEADER, array(
            'POST /Service/KPSPublic.asmx HTTP/1.1',
            'Host: tckimlik.nvi.gov.tr',
            'Content-Type: text/xml; charset=utf-8',
            'SOAPAction:
"http://tckimlik.nvi.gov.tr/WS/YabanciKimlikNoDogrula"',
            'Content-Length: ' . strlen($toSend)
        ));
        $response = curl_exec($ch);
        curl_close($ch);
        return strip_tags($response) == 'true';
    }
}
```

If the number is fake BloodHubAPI will give the respective error message in JSON format.

```
{"error":"TRUE","error_msg":"This identitiy Number is fake"}
```

Also create this message in XML Format:

```
<?xml version="1.0"?>
<api>
  <error>TRUE</error>
  <error_msg>This identitiy Number is fake</error_msg>
</api>
```

If user passes the identitity number verification process succesfully but if the entered email has already registered to the system for a user, following output will be displayed in JSON format.

```
{"error":"TRUE","error_msg":"User already existed with email:
mustafaculban@gmail.com and username: karamusluk"}
```

Below is in the XML format.

```
<?xml version="1.0"?>
<api>
  <error>TRUE</error>
  <error_msg>User already existed with email: mustafaculban@gmail.com
and username: karamusluk</error_msg>
</api>
```

If user passes all verification processes successfuly , registration will be done and the following output will be displayed in JSON format.

```
{"error":"FALSE","user":{"id":7,"username":"izelgurbuz","firstname":"İze
l","surname":"Gürbüz","email":"izelgurbuz1@gmail.com","bloodType":"0+","
birthdate":"11081995","address":"denemeAdress","telephone":"053178xx034"
,"available":0}}
```

Notification Management Subsystem

Notifications hold important place in this application. So we created a specific subsystem for notification processes. This subsystem creates and sends notifications to users according to data it get from other subsystems vis SMS, Push Notification or Mail.

We started to implement the code of this part. User can successfully send notification via the BloodHubAPI. Below is the implementation so far:

The code written below handles sending notifications as SMS or email:

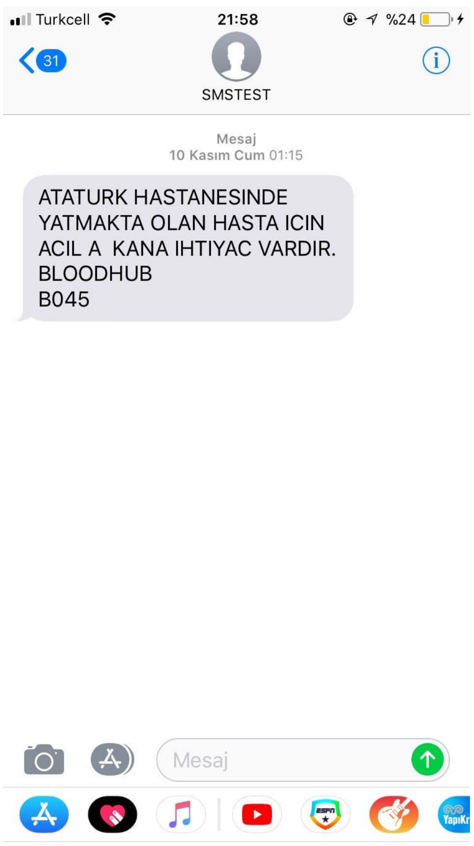
```
elseif ($func == "sendBloodRequest") {

            if (isset($_GET['notificationType']) &&
isset($_GET['bloodType']) && isset($_GET['hospitalName']) &&
isset($_GET['name_surname'])){
                  if($_GET['notificationType'] =='mail'){

$userEmails=$db->getUserWithBloodType($_GET['bloodType']);
                        if (is_array($userEmails) ||
is_object($userEmails)){
                              foreach ($userEmails as $key => $item) {

$api->sendMail($item,$_GET['name_surname'],"ACIL ".$_GET['bloodType']."
KAN GEREKMEKTEDIR.","[BAIS-ANNC:BILKENT] COK COK ACIL KAN IHTIYACI");
                              }
                        }

                  }
                  elseif($_GET['notificationType'] =='sms'){

$userPhones=$db->getPhoneWithBloodType($_GET['bloodType']);
                        foreach ($userPhones as $key => $item) {
```

```
                                    //echo $item.'<br>';

$api->sendSMS($item,$_GET['name_surname'],$_GET['bloodType'],$_GET['hosp
italName']);
                              }

                  }

                  else{

                        $response["error"] = TRUE;
                  $response['error_msg'] = "You need to give
notificationType as one of the {sms, mail, push}";
                  toXML($response);
                  }
```

The class written below is for BloodHubAPI prepared to be used for notification process:

```
class API
{
      function sendMail($email,$name_surname,$message,$subject){

                  $toMail = $email;

                  $toFullName = $name_surname;

                  $mailSubject = $subject;

                  $messageContent = $message;

                  $mail = new PHPMailer;
                  $mail->IsSMTP();
                  $mail->SMTPAuth = true;
                  $mail->Host = 'mail.cinedia.net';
                  $mail->Port = 587;
                  $mail->Username = 'no-reply@cinedia.net';
                  $mail->Password = 'xxxx';

                  $mail->SetFrom($mail->Username, '[BAIS-ANNC:BILKENT]
COK COK ACIL KAN IHTIYACI');
                  $mail->AddAddress($toMail, $toFullName);
                  $mail->CharSet = 'UTF-8';
                  $mail->Subject = $mailSubject;
                  $mail->MsgHTML($messageContent);
                  $mail->Priority = 1;
                  $mail->AddCustomHeader("X-MSMail-Priority: High");
                  $mail->AddCustomHeader("Importance: High");
                  if(isset($_GET['imageLink']))
                        $mail->AddAttachment($_GET['imageLink']);
                  if($mail->Send()) {
                      $response["error"] = FALSE;
                      $response['success'] = "Email has been sent
succesfully";
                      toXML($response);
                  } else {
                      $response["error"] = TRUE;
                  $response["error_msg"] = "Email hasn't been sent
successfully because of the reason". $mail->ErrorInfo;
                  toXML($response);

                  }
      }

      function
sendSMS($phoneNumber,$name_surname,$bloodtype,$hospitalName){
$postUrl='http://www.oztekbayi.com/panel/smsgonder1Npost.php';
            $MUSTERINO='xxxx'; //5 haneli müşteri numarası
            $KULLANICIADI='mustafaculban';
            $SIFRE='xxxxxxxx';
            $ORGINATOR="SMS TEST";

            $TUR='Normal';

            $mesaj1=$hospitalName.'DE YATMAKTA OLAN '.$name_surname.'
ADLI HASTA ICIN ACIL '.$bloodtype.' KANA IHTIYAC VARDIR.   BLOODHUB';
            $numara1=$phoneNumber;

            $xmlString='data=<sms>
            <kno>'. $MUSTERINO .'</kno>
            <kulad>'. $KULLANICIADI .'</kulad>
            <sifre>'.$SIFRE .'</sifre>
            <gonderen>'.  $ORGINATOR .'</gonderen>
            <mesaj>'. $mesaj1 .'</mesaj>
            <numaralar>'. $numara1.'</numaralar>
            <tur>'. $TUR .'</tur>
            </sms>';
            $Veriler =  $xmlString;
            $ch = curl_init();
            curl_setopt($ch, CURLOPT_URL, $postUrl);
            curl_setopt($ch, CURLOPT_POST, 1);
            curl_setopt($ch, CURLOPT_POSTFIELDS, $Veriler);
            curl_setopt($ch, CURLOPT_RETURNTRANSFER,1);
            curl_setopt($ch, CURLOPT_SSL_VERIFYPEER,0);
            curl_setopt($ch, CURLOPT_TIMEOUT, 30);
            $response = curl_exec($ch);
            curl_close($ch);
            echo $response;
            echo '<br>';

      }

}
```
The codes written for sending notification is fully and accurately functional. Screenshots of received notifications that were sent for testing the code are below:

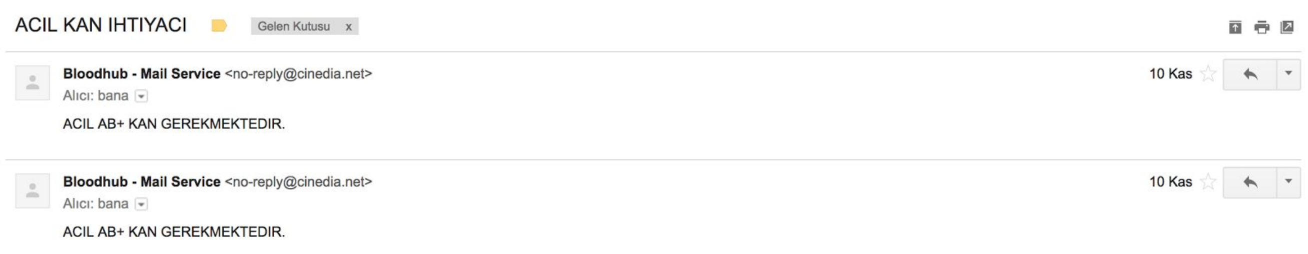
This is for SMS testing:



Mesaj
10 Kasım Cum 01:15

ATATURK HASTANESINDE
YATMAKTA OLAN HASTA ICIN
ACIL A KANA IHTIYAC VARDIR.
BLOODHUB
B045

This is for Mail Testing:



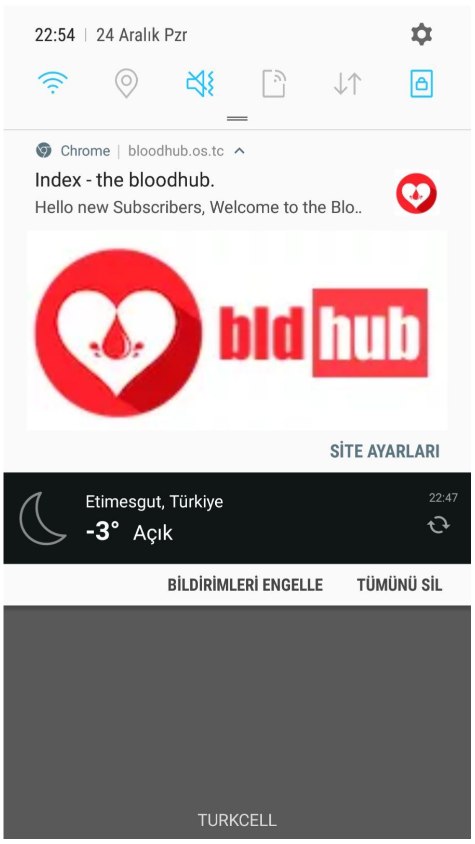
Bloodhub - Mail Service <no-reply@cinedia.net>
Alıcı: bana
ACIL AB+ KAN GEREKMEKTEDIR.
10 Kas

Bloodhub - Mail Service <no-reply@cinedia.net>
Alıcı: bana
ACIL AB+ KAN GEREKMEKTEDIR.
10 Kas

Administration of the Bloodhub also can send the notification for informing users with the statistics, updates, news and similar information via using the same functions described above. Besides, this kind of notifications will be sent to the users who didn't registered but subscribed to newsletter. The code below will handle this process:

```
<?php
if($_GET['key'] === "qwerty"){//in order to prevent outer usage who are
not authorized
    require 'oneSignal.php';
    $oneSignal = new oneSignal();
    if($_POST) {
        $text = $_POST['text'];
        $image = $_POST['image'];
        $link = $_POST['link'];
        echo '<pre style="width: 50%;margin-left: 25%">';
        print_r($oneSignal->sendMessage($text, $link,$image));
        echo '</pre>';
    }
    ?>
    <!doctype html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <title> Bloodhub Web Services || Web Push Message Sender</title>
        <script src="https://cdn.onesignal.com/sdks/OneSignalSDK.js"
async='async'></script>
        <script>
            var OneSignal = window.OneSignal || [];
            OneSignal.push(["init", {
                appId: '<?=$oneSignal->__construct();?>',
                autoRegister: false, /* Set to true to automatically
prompt visitors */
                subdomainName: 'bloodhub', // Uygulamayı oluştururken
aldığınız subDomain
                notifyButton:
                {
                    enable: true, /* Required to use the notify button
*/
                    size: 'large', /* One of 'small', 'medium', or
'large' */
                    theme: 'inverse', /* One of 'default' (red-white) or
'inverse" (white-red) */
                    position: 'bottom-left', /* Either 'bottom-left' or
'bottom-right' */
                    offset: {
                        bottom: '0px',
                        left: '0px', /* Only applied if bottom-left */
                        right: '0px' /* Only applied if bottom-right */
                    },
                    prenotify: true, /* Show an icon with 1 unread
message for first-time site visitors */
                    showCredit: false, /* Hide the OneSignal logo */
                    text: {
                        'tip.state.unsubscribed': 'Abone ol',
                        'tip.state.subscribed': "Aramıza Hoş Geldin :)",
                        'tip.state.blocked': "Bildirimlerin
Engellendi.",
                        'message.prenotify': 'Bildirimlere Abone Olmak
İçin Tıklayın.',
                        'message.action.subscribed': "Aramıza Hoş Geldin
:)",
                        'message.action.resubscribed': "Abone oldun.
Hoşgeldin :)",
                        'message.action.unsubscribed': "Abonelikten
Ayrıldın.",
                        'dialog.main.title': 'OneSignal Api',
                        'dialog.main.button.subscribe': 'Abone Ol',
                        'dialog.main.button.unsubscribe': 'Abonelikten
Ayrıl',
                        'dialog.blocked.title': 'Engelli Bildirimler.',
                        'dialog.blocked.message': "Follow these
instructions to allow notifications:"
                    }
                }
            }]);
        </script>
    </head>
    <body>
    <form style="width: 50%;margin-left: 25%" action="" method ="post"
align="center">
        <input style="width: 100%;margin-bottom: 1%" type="text"
id="text" name="text" placeholder="Mesaj" /><br>
        <input style="width: 100%;margin-bottom: 1%" type="text"
id="link" name="link" placeholder="Mesajın Gideceği Link" /><br>
        <input style="width: 100%;margin-bottom: 1%" type="text"
id="image" name="image" placeholder="Mesaj ile gondereceginiz resim"
/><br>
        <input type="submit"  />
    </form>
    </body>
    </html>
<?php }else{echo "you are not allowed to use";}?>
```

Below is the screenshot for the successfully sent notification to the user:


22:54 | 24 Aralık Pzr

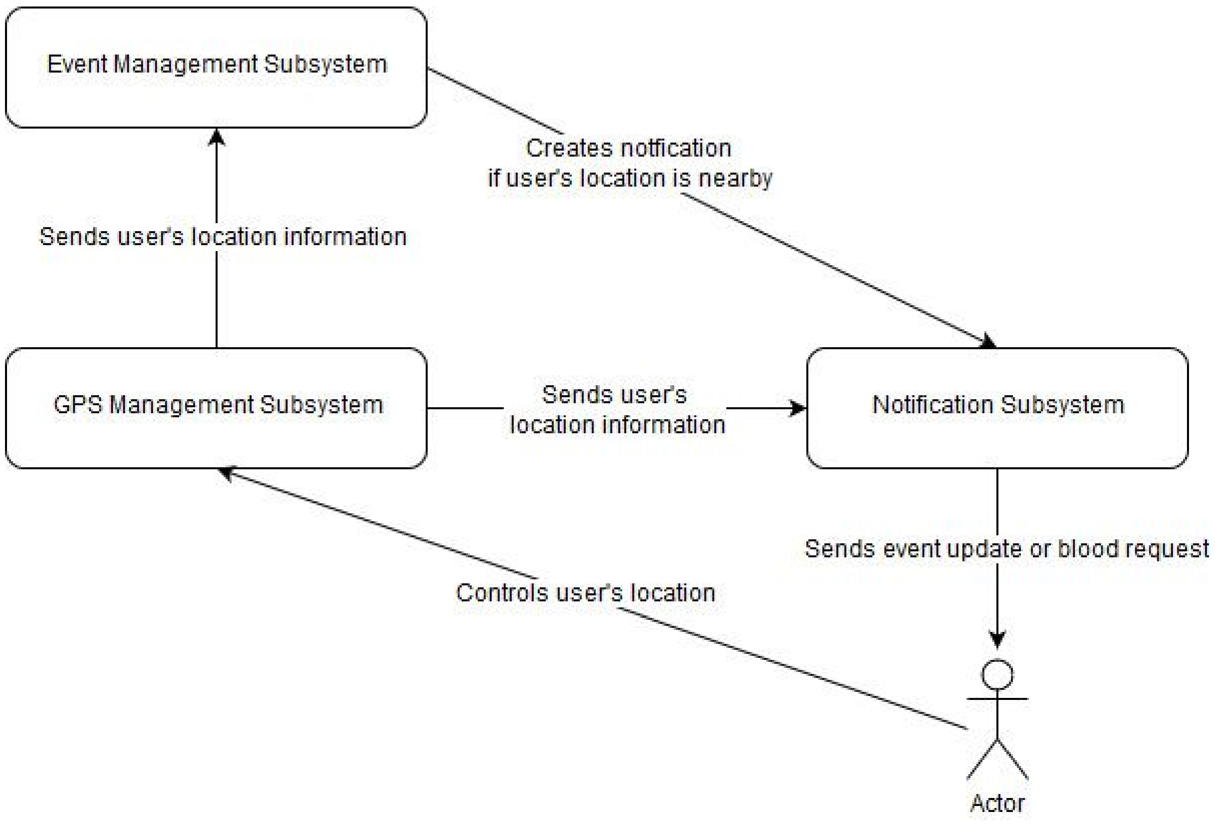
Chrome | bloodhub.os.tc

Index - the bloodhub.

Hello new Subscribers, Welcome to the Blo..

Event Management Subsystem

This subsystem is responsible for following event updates. What we called as events are organizations of Kızılay. They can organize some events in different timetables in various locations. If an user is located nearby one of these events system will notify user. Additionally, if it detects an event update it informs users immediately. This subsystem is in connection with Notification Management and Gps Management subsystem.


Event Management Subsystem
Creates notfication if user's location is nearby
Sends user's location information
GPS Management Subsystem
Sends user's location information
Notification Subsystem
Sends event update or blood request
Controls user's location
Actor

Gps Management Subsystem

This subsystem is responsible for collecting location information from users. It provides location information to Notification Management Subsystem. The user who will be notified by application will be selected with data GPS Management Subsystem provides.

Data Management Subsystem

This subsystem is at the lowest layer of the system which called as Data Storage Layer. Data Management Subsystem stores the data which will be come from other subsystems.

We started to implement the data storage part. Data will be stored in the SQL database called MySQL and will be access via the PHP for the BloodHubAPI serve purposes.

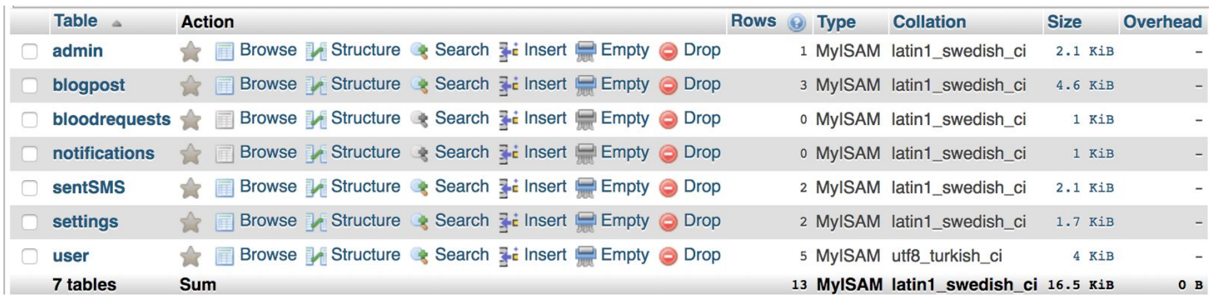


Table	Action	Rows	Type	Collation	Size	Overhead
admin	Browse Structure Search Insert Empty Drop	1	MyISAM	latin1_swedish_ci	2.1 KiB	–
blogpost	Browse Structure Search Insert Empty Drop	3	MyISAM	latin1_swedish_ci	4.6 KiB	–
bloodrequests	Browse Structure Search Insert Empty Drop	0	MyISAM	latin1_swedish_ci	1 KiB	–
notifications	Browse Structure Search Insert Empty Drop	0	MyISAM	latin1_swedish_ci	1 KiB	–
sentSMS	Browse Structure Search Insert Empty Drop	2	MyISAM	latin1_swedish_ci	2.1 KiB	–
settings	Browse Structure Search Insert Empty Drop	2	MyISAM	latin1_swedish_ci	1.7 KiB	–
user	Browse Structure Search Insert Empty Drop	5	MyISAM	utf8_turkish_ci	4 KiB	–
7 tables	Sum	13	MyISAM	latin1_swedish_ci	16.5 KiB	0 B

There is also the code for the creation of these table with dummy data in order us to test the current implementation.Below is the SQL statements of the system:

```
CREATE TABLE `admin` (
  `id` int(11) NOT NULL,
  `username` varchar(30) NOT NULL,
  `firstname` varchar(32) NOT NULL,
  `lastname` varchar(32) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(255) NOT NULL,
  `telephone` varchar(20) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;


INSERT INTO `admin` (`id`, `username`, `firstname`, `lastname`, `email`,
`password`, `telephone`) VALUES
(1, 'mustafa', 'Mustafa', 'Culban', 'mustafaculban1@gmail.com',
'xxxxxxxxx', '5396768149');

CREATE TABLE `blogpost` (
  `id` int(11) NOT NULL,
  `post_title` text NOT NULL,
  `post_text` text NOT NULL,
  `post_long_text` text NOT NULL,
  `image_link` text NOT NULL,
  `post_link` text NOT NULL,
  `date` text NOT NULL,
  `active` int(11) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO `blogpost` (`id`, `post_title`, `post_text`,
`post_long_text`, `image_link`, `post_link`, `date`, `active`) VALUES
(1, 'Blood Connects Us All in a Soul', 'In many countries, demand
exceeds supply, and blood services face the challenge of making blood
available for patient. ', '', 'images/blog_1.jpg', '', 'April 4, 2017',
1),
(2, 'Give Blood and Save three Lives', 'To save a life, you don\'t need
to use muscle. By donating just one unit of blood, you can save three
lives or even several lives.', '', 'images/blog_2.jpg', '', 'April 4,
2017', 1),
(3, 'Why Should I donate Blood ?', 'Blood is the most precious gift that
anyone can give to another person.Donating blood not only saves the life
also donors.', '', 'images/blog_3.jpg', '', 'April 4, 2017', 1);

CREATE TABLE `bloodrequests` (
  `id` int(11) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

CREATE TABLE `notifications` (
  `id` int(11) NOT NULL,
  `hastane_adi` varchar(255) NOT NULL,
  `blood_type` enum('A+','A-','B+','B-','AB+','AB-','0-','0+') NOT NULL,
  `message` text,
  `longitude` varchar(100) NOT NULL,
  `latitude` varchar(100) NOT NULL,
  `n_type` enum('push','sms','mail') NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

CREATE TABLE `sentSMS` (
  `id` int(11) NOT NULL,
  `toNo` varchar(255) NOT NULL,
  `sentDate` varchar(255) NOT NULL,
  `msgUniqueID` varchar(255) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO `sentSMS` (`id`, `toNo`, `sentDate`, `msgUniqueID`) VALUES
(1, '5396768149', '09-11-2017', '1854123229'),
(2, '5396768149', '10-11-2017', '1854123239');
```

```
CREATE TABLE `settings` (
  `about` text NOT NULL,
  `url` text NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO `settings` (`about`, `url`) VALUES
('', 'http://cs491-2.mustafaculban.net'),
('BloodHub aims to bring many easiness to blood donation process. There
will be\r\ntwo types of users. First one is the users who want to donate
their blood. They can see\r\navailable places to donate their blood from
interactive map of application. They will\r\nbe notified when there is a
need for blood. Application will send an notification alert\r\nto their
phone. Users from the near of place where there is a need for blood will
be\r\nalerted. Their location will be taken with their phone's location
services. Second type \r\n4\r\nof users are people who look for blood
for their friends, relatives or themselves. They\r\ncan interact with
volunteers by using BloodHub.', '');

CREATE TABLE `user` (
  `id` int(11) NOT NULL,
  `username` varchar(255) CHARACTER SET latin1 NOT NULL,
  `firstname` varchar(255) CHARACTER SET utf8 NOT NULL,
  `surname` varchar(255) CHARACTER SET utf8 NOT NULL,
  `password` varchar(255) CHARACTER SET latin1 NOT NULL,
  `salt` varchar(10) CHARACTER SET latin1 NOT NULL,
  `email` varchar(155) CHARACTER SET latin1 NOT NULL,
  `identitiyNum` varchar(255) CHARACTER SET latin1 NOT NULL,
  `identitySalt` varchar(10) CHARACTER SET latin1 NOT NULL,
  `bloodType` varchar(3) CHARACTER SET latin1 NOT NULL,
  `birthdate` varchar(11) CHARACTER SET latin1 NOT NULL,
  `address` varchar(255) CHARACTER SET latin1 NOT NULL,
  `telephone` varchar(15) CHARACTER SET latin1 NOT NULL,
  `available` int(1) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_turkish_ci;

INSERT INTO `user` (`id`, `username`, `firstname`, `surname`,
`password`, `salt`, `email`, `identitiyNum`, `identitySalt`,
`bloodType`, `birthdate`, `address`, `telephone`, `available`) VALUES
(1, 'mustafaculban', 'Mustafa', 'Culban',
'hpwX8Xee9eXpZNDX9V6S/JO/2b43N2UyMGM4ZWFh', '77e20c8eaa',
'mustafaculban1@gmail.com', 'xJScYofeE8AC2udPsDv8yRpPiCo5ZDNjZjg5YWUz',
'77e20c8eaa', 'AB+', '16041995', 'denemeAdress', '5396xx8149', 1),
(2, 'karamusluk', 'Mustafa', 'Culban',
'R8jB6Ra0YZPJI9MiV7+FBkaNLl5hY2Y2YzNjMWVm', 'acf6c3c1ef',
'karamusluk@gmail.com', 'bAEk/vbYJcaBpBzmE4zTsh1crXs0YWQ4NzEwMzJl',
'acf6c3c1ef', 'AB-', '16041995', 'denemeAdress', '05396xx8149', 1),
(5, 'izelgurbuz', 'Ä°zel', 'GÃ1/4rbÃ1/4z',
'hFSjvZf3XjF/Mh1woJMz02Fm64swMmE1MjY3YjRj', '02a5267b4c',
'izelgurbuz@gmail.com', 'mpNCD8HMBpY+k+DRGNv593TY3KcyMzE5ZGYzYWFl',
'02a5267b4c', 'AB+', '11081995', 'Ankara Bilkent ', '53x78xx034', 1),
(6, 'karamusluk', 'Mustafa', 'Culban',
'+HFDmKc5dSHXYX18fQ4rRNOnXLYzMDQyZjNjNGI5', '3042f3c4b9',
'mustafaculban@gmail.com', 'PesQbvCy+VrAqr7eqTUkiiWCACgyMTdlMDMxODVl',
'3042f3c4b9', '0b', '16041995', 'denemeAdress', '05396xx8149', 0),
(7, 'izelgurbuz', 'Ä°zel', 'GÃ1/4rbÃ1/4z',
'qa7P9VEPpEyD2NLvgtZJi27RC0E3OTI4NGE4NDNk', '79284a843d',
'izelgurbuz1@gmail.com', 'nHsM4A4j2DbbwgwJ6LTjxYCK4ts1ODI5NThmYTNk',
'79284a843d', '0+', '11081995', 'denemeAdress', '053x78xx034', 0);

ALTER TABLE `admin`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `blogpost`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `id_2` (`id`);

ALTER TABLE `bloodrequests`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `notifications`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `sentSMS`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `user`
  ADD PRIMARY KEY (`id`),
  ADD KEY `id` (`id`);

ALTER TABLE `admin`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;

ALTER TABLE `bloodrequests`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `notifications`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `sentSMS`
```

```
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;

ALTER TABLE `user`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;COMMIT;
```

References

[1] "Documentation", Firebase.google.com, 2017. [Online]. Available: https://firebase.google.com/docs/cloud-messaging/ [Accessed: October 8, 2017].
[2] "Application programming interface", Wikipedia.com, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Application_programming_interface [Accessed: October 8, 2017].
[3] "Code of Ethics | National Society of Professional Engineers", Nspe.org, 2017. [Online]. Available: https://www.nspe.org/resources/ethics/code-ethics. [Accessed: October 8, 2017].
[4] "OAuth 2.0", oauth.net, 2017. [Online]. Available: https://oauth.net/2/ [Accessed: October 8, 2017].
[5] " RFC 6819 - OAuth 2.0 Threat Model and Security Consideration", rfc-base.org, 2017. [Online]. Available: http://www.rfc-base.org/rfc-6819.html [Accessed: October 8, 2017].