Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Bloodhub.

Low-Level Design Report

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

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

Progress Report
Feb 12, 2018

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.

1. 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.

1.1 Object design trade-offs

1.1.1 Complexity vs Functionality

Functionality is a more important issue than complexity for our application. Because of our application is trying to address a sensitive problem, it has to be much functional for our users. There will be MySQL, PHP, Java, JavaScript, CSS and combination of them can increase the complexity of our application. Database and server operations will be programmed with MySQL and PHP. Java and JavaScript will be the main part of application. Java programming language is used while coding Android applications. CSS will be used in the admin panel. Application will make data transfers, so Mysql, Php and Android should be coded cooperatively. Because of Android is Java based, strong Java knowledge is necessary and additional tutorials for Android are compulsory. MySQL increases the complexity but if we will not use server for storing data, our application will lose most of its functionality. It is

necessary to maintain a functional system for users, so complexity will be sacrificed for this purpose.

1.1.2 Performance vs Memory

Our application will have a server side memory. All of the user information will be saved in the server. The capacity should increase according to changes in the number of people. With the increasing number of users, the performance of our application can degrade. The response times probably will increase related to increased memory. In order to keep system's performance high, the necessary memory should be allocated for the application.

1.1.3 Security vs Cost

Our application holds critical information for users such as ID number, address, location, blood type and so on. These type of data must be kept under high security. If the application doesn't seem like reliable, users will not choose using this. In order to maintain security for mobile applications, mobile applications should communicate with web services by using HTTPS. It is necessary to use Hostname Verifiers for Android version and NSURLAuthenticationMethodServerTrust for iOS version of application. So, it is very crucial to keep the data safe. It is not easy to keep information safe in today's world. It requires cost of time and money for us to maintain safety for users.

1.2 Interface documentation guidelines

This section is for demonstrating the guidelines for class interfaces of this documentation. There are sections of "class name", "description", "methods", "variables". In the "class name" section, the name of the class is written. In the "description" part, we explain what this class do. In the "methods" part, we specify the methods of the class, includes getter and setter methods. In the "variables" section, constants are specified.

1.3 Engineering standards (e.g., UML and IEEE)

In the reports, Unified Modeling Language (UML) is used for demonstrating diagrams, scenarios and use cases of the system. Structure of the system became easy to read with this technique. IEEE writing format is used for citations in the reports because it is a widely used standard.

1.4 Definitions, acronyms, and abbreviations

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.

2. Packages

2.1 Client Side Application

The client side application will be implemented as Android and iOS aplications. MVC(Model- View- Controller) design pattern will be used during the implementation of both Android and iOS Application.

2.1.1 Model Package

There will be Model classes inside this package where it will make suitable to create MVC pattern for the bloodhub. The classes will be as follows:

- **com.bloodhub.android.model.User**
  The user model class represents the each user of the system to save his/her information and provide a secure usage

- **com.bloodhub.android.model.BloodType**
  The BloodType modell class is used to hold enumerated types of blood.
- **com.bloodhub.android.model.Event**
  The Event model class represents the each event to display information of events to the user.
- **com.bloodhub.android.model.BlogPost**
  The BlogPost model class holds blog posts related to blood donation.
- **com.bloodhub.android.model.Notification**

The Notification model class represents both sent and received notifications and holds information of notifications to display to user and save.

- **com.bloodhub.android.model.Location**

  The Location model class holds information of locations of hospitals, bloodbanks and events.

2.1.2 Layout Package

There will be the XML style files that construct the visual layouts of both android and iOS project.

- **com.bloodhub.android.layout.Login**
  The Login layout file includes login screen user inteface elements.
- **com.bloodhub.android.layout.Register**
  The Register layout file includes register screen user inteface elements.
- **com.bloodhub.android.layout.Main**
  The Main layout file includes toolbar elements which is visible on the top of each screen except login and register screens .
- **com.bloodhub.android.layout.Profile**
  The Profile layout file includes editable text fields holding user information and other user interface elements making this screen user friendly.
- **com.bloodhub.android.layout.Blog**
  The Blog layout file includes user interface elements that hold blog contents.
- **com.bloodhub.android.layout.CreateNotification**
  The CreateNotification layout file includes user interface elements such as editable text fields and toggles that allow the user to create the desirable notification .
- **com.bloodhub.android.layout.Map**
  The Map layout file includes map fragment that allows people to find the locations of hospitals, bloodbanks and events..
- **com.bloodhub.android.layout.ReceivedNotification**
  The ReceivedNotification layout file includes the list of the notifications that have received by the user.
- **com.bloodhub.android.layout.SentNotificaton**
  The SentNotification layout file includes the list of the notifications that have sent by the user.
- **com.bloodhub.android.layout.EMFive**
  The EM5 layout file includes information of confirmed users to reach in the emergency situation , as a list view.
- **com.bloodhub.android.layout.Contact**
  The Contact layout file includes user interface elements having content of contact information.
- **com.bloodhub.android.layout.About**
  The About layout file includes user interface elements having content of information about developers of the system.
- **com.bloodhub.android.layout.Questionare**
  The Questionare layout file includes .

2.1.3 Activities Package

This package will have classes that are responsible of putting data to the respective page and the updates on the pages. The classes are as follows:

- **com.bloodhub.android.activities.LoginActivity**
  This class takes XML files of the login layout and gives functionality to these objects in the XML file.
- **com.bloodhub.android.activities.RegisterActivity**
  This class takes XML files of the login layout and gives functionality to these objects in the XML file
- **com.bloodhub.android.activities.MainActivity**
  This class merges Login and Register activities and graphics in the first screen.
- **com.bloodhub.android.activities.ProfileActivity**
  This class creates or updates data in a profile
- **com.bloodhub.android.activities.CreateNotificationActivity**
  This class creates notifications according to received data.
- **com.bloodhub.android.activities.BlogActivity**
  With using XML file creates a list. Every element of this list is a blog post.
- **com.bloodhub.android.activities.MapActivity**
  Shows Kızılay places on the Google map.
- **com.bloodhub.android.activities.MyNotificationActivity**
  This class manages the notification for a specific user.
- **com.bloodhub.android.activities.SentNotificatonActivity**
  This class sends notifiication which is created at CreateNotificationActivity class.
- **com.bloodhub.android.activities.EMFiveActivity**
  This class visualize the operations of com.bloodhub.android.api.EM5Controller class.
- **com.bloodhub.android.activities.ContactActivity**
  Visualizes the contact people from XML file.
- **com.bloodhub.android.activities.AboutActivity**
  Visualizes the info about specific activity.
- **com.bloodhub.android.activities.QuestionnaireActivity**
  This class visualize the questionnaire to the new registered users.

2.1.4 Service Package

This package will contain classes which are responsible for sending push notifications, authentication jobs (login and register which will be done by communicating with BloodHub REST-architectured API), GPS location tracker service which is Geolocation service from Android. These services will be enhanced in the final version. Classes are as follows:

2.1.4.1 Firebase Package

- **com.bloodhub.android.services.firebase.app.Config**
  This class keeps the final type variables which will be used at other Firebass classes.
- **com.bloodhub.android.services.firebase.service.MyFirebaseInstanceIDService**
  It stores the Firebase registration IDs of the both Android and iOS devices with through Helper class, SharedPreferencesManager, intermediary.
- **com.bloodhub.android.services.firebase.service.MyFirebaseMessagingService**
  It checks whether a notification message arrived and if so it handles with messages and vizualizes the notification message
- **com.bloodhub.android.services.firebase.utils.NotificationUtils**
  It is used in MyFirebaseMessagingService class to show the message to the user

2.1.4.2 Geolocation Package

- **com.bloodhub.android.services.map.Config**
- **com.bloodhub.android.services.map.GetCurrentLocation**

Also GeoJson package will be used to implementation for the Geolocation and Map of the BloodHub apps. Detailed package for geojson given below:

- **com.bloodhub.android.services.geojson.Circle**
- **com.bloodhub.android.services.geojson.Crs**
- **com.bloodhub.android.services.geojson.Feature**
- **com.bloodhub.android.services.geojson.FeatureCollection**
- **com.bloodhub.android.services.geojson.GeoJson**
- **com.bloodhub.android.services.geojson.GeoJsonObject**
- **com.bloodhub.android.services.geojson.Geometry**
- **com.bloodhub.android.services.geojson.GeometryCollection**
- **com.bloodhub.android.services.geojson.LineString**
- **com.bloodhub.android.services.geojson.LngLatAlt**
- **com.bloodhub.android.services.geojson.Main**
- **com.bloodhub.android.services.geojson.MultiLineString**
- **com.bloodhub.android.services.geojson.MultiPoint**
- **com.bloodhub.android.services.geojson.MultiPolygon**
- **com.bloodhub.android.services.geojson.Point**
- **com.bloodhub.android.services.geojson.Polygon**

2.1.4.2.1 GeoJson Adapters Package

This package will include adapter classes for geoJson package above.

- **com.bloodhub.android.services.geojson.adapter.GeoJsonObjectAdapter**
- **com.bloodhub.android.services.geojson.adapter.LngLatAltAdapter**

2.1.5 Helper Package

This package will include classes to create session like instances, handle the request from the server in general. This package can be changed in the final implementation. Classes as follows:

- **com.bloodhub.android.helper.sharedPreferences.SharedPreferencesManager**
  This class does login, logout operations on the SharedPreferences to check whether the user has logged in. If so with getUser() method returns the user logged in.
- **com.bloodhub.android.helper.sharedPreferences.FcmSharedPrefrence**
  It is used to store the registration ID in SharedPreferences and fetches from there.
- **com.bloodhub.android.helper.sharedPreferences.PrefManager**
  It is used to save registration ID of the device if it is the first time that user logs in the system.
- **com.bloodhub.android.helper.RequestHandler**
  It sends a post request to the end points of the BloodHub REST API.

2.2 Server Side Application

2.2.1 BloodHub REST API

BloodHub REST API is a framework based communication channel, application programming interface. This API is powered by PHP5 and MYSQL. This communication channel will be a connection between android, iOS devices and web server. This API will be the most important part of the project because it supplies the necessary endpoints for the android and iOS application in order them to do their job properly. This project will be highly dependent on the Bloodhub REST API.

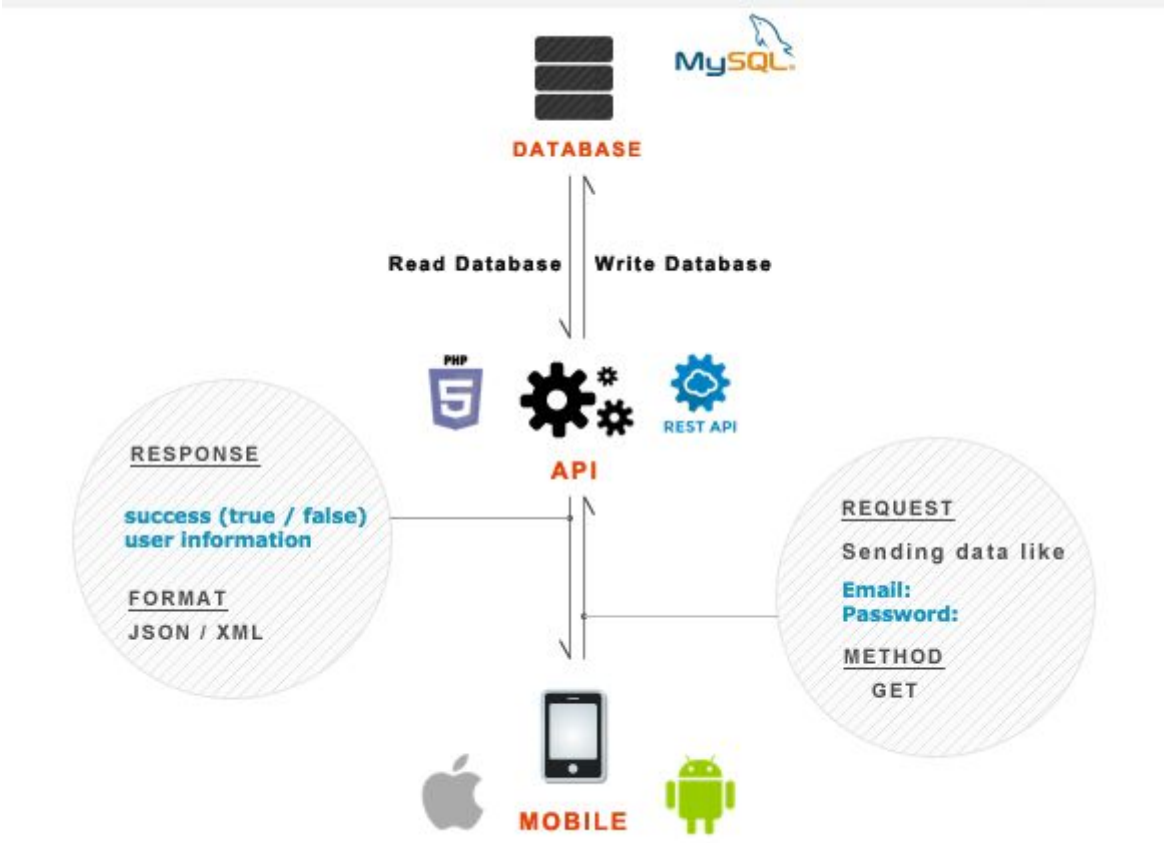
BloodHub REST API will provide various functionalities such as logging a user in, registering user to the system, sending notification in proper channels and more. These functionalities can be found on the documentation of the Bloodhub REST API in more detail.[1]

We will be using BloodHub REST-architectured API to communicate via server in order to provide data to the client apps , android and iOS apps.

BloodHub REST-architectured API's outputs can be selected as XML or JSON. JSON outputs are used commonly but in some cases for windows machines especially XML

are needed as output for API endpoints. This application will follow the REST Design Patterns[2] and android design pattern.

Below there is a communication flow example of register or login for a user.



User on the mobile devices , will send request to register or login to the system via proper classes, that are explained in the client side application packages, to the BloodHub REST API and from there it will communicate with database on the server, and if the result is true, which means user successfully logged in or registered to the system, BloodHub REST API response with success information and with that, logged in or registered user information is returned with the response. These responses can be both JSON or XML format.

2.2.1.1 Controller Package

This controller package will include classes that are responsible for communicating with the BloodHub REST API to receive and send the requests at endpoints[3]. The classes are as follows:

- **com.bloodhub.android.api.UserController**
  This class is responsible for login and registration. This class does these processes with the object of RequestController.
- **com.bloodhub.android.api.RequestController**
  Sends necessary data to respective API end-points, receives API servers' responses.
- **com.bloodhub.android.api.EventsController**
  Receives events or a specific event which is identified by administration to the application and creates respective event object from an event model..
- **com.bloodhub.android.api.BlogController**
  Receives blogs or a specific event which is identified by administration to the application and creates respective event object from an blog model..
- **com.bloodhub.android.api.EM5Controller**
  This class does adding, creating, deleting people processes from emergency 5 list of users.
- **com.bloodhub.android.api.FirebaseController**
  This class does creating and sending notification processes to appropriate users.

2.2.1.2 General Package

This package will show the implementation of the PHP5 REST API structure. This is a different platform that is 7 days a week, 24 hours a day running on online server and the file listing is given in a form of JSON output. It is as follows:

```
{
   "0": "api.php", // which contains all the REST endpoints.
   "1": "documentation",
   "2": "error_log",
   "3": "firebase",
   "4": "generate",
   "5": "index.php",
   "6": "login.php",
   "7": "register.php",
   "documentation": {
      "0": "index.html",
      "1": "index_files",
      "index_files": {
         "0": "base.css",
         "0": "bundle.js",
         "1": "icon.eot",
         "2": "icon.woff",
         "3": "livereload.js",
         "4": "railscasts.css",
         "5": "style.css"
      }
   },
   "firebase": {
```

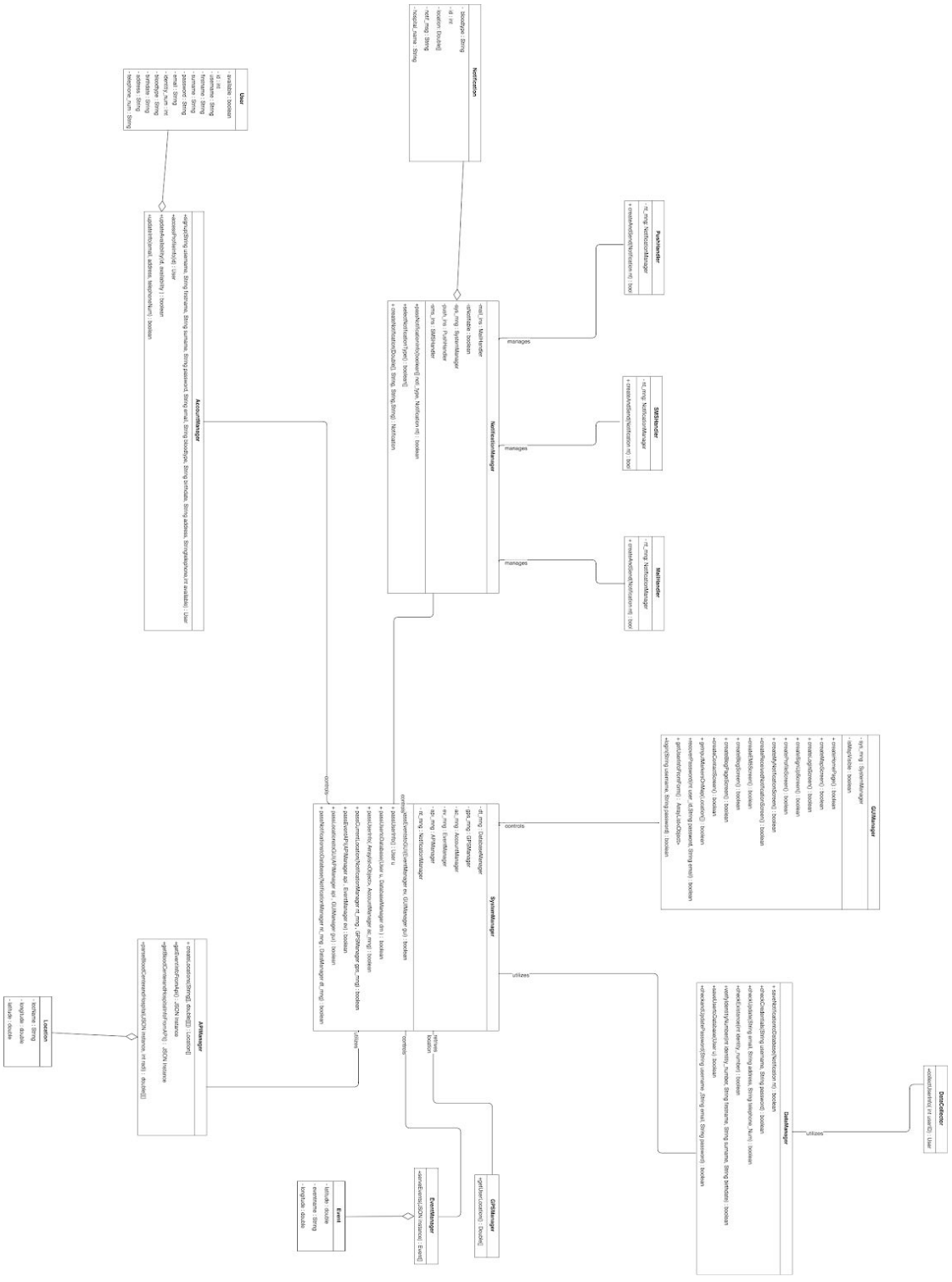
```
            "0": "config.php",
            "1": "custom",
            "2": "firebase.php",
            "3": "index.php",
            "4": "push.php",
            "custom": {
               "0": "config.php",
               "1": "firebase.php",
               "2": "index.php",
               "3": "notification.php"
            }
         },
         "generate": {
            "0": "css",
            "1": "error_log",
            "2": "index.php",
            "3": "js",
            "css": {
               "0": "style.css"
            },
            "js": {
               "0": "index.js"
            }
         }
      }
```

2.2.1.3 Configuration Package

This package will show the configuration package of the BloodHub REST API. In that package classes, proper database connections, functions that are used to provide the REST API, notifications, verifications of the identities and so on. The hierarchy of the package is as follows:

```
{
   "0": "class",//folder
   "1": "config",//folder
   "2": "database", //folder
   "3": "functions.php",
   "class": {//folder detail
      "0": "class.api.php",
      "1": "class.phpmailer.php",
      "2": "class.pop3.php",
      "3": "class.smtp.php",
      "4": "kimlikNoVerify",//folder
      "kimlikNoVerify": {//folder detail
        "0": "TcKimlikNoSorgula.php",
        "1":"YabanciKimlikNoDogrula.php"
      }
   },
   "config": {//folder detail
      "0":"config.php"
```

```
    },
    "database": {//folder detail
      "0": "dbConn.php",
      "1":"dbFunc.php"
    }
}
```



CLASS DIAGRAM OF BLOODHUB

3. Class Interfaces

3.1 Model Package

Model package includes necessary classes for modelling the elements, and the classes given in section 2.1.1.

3.1.1 User Class

Class Name: User
Description: This class will be used to create a signed in user instance to use throughout the session of a user on the apps.
Methods: String getUsername() String getEmail() String getFirstname() String getSurname() String getBloodType() String getBirthdate() String getAddress() String getTelephone() String getAvailable() String getLastLoginIP() String getLastLoginDate() String getLastLoginTime() String getLastCity() and setter methods for these getter methods.
Variables: int id String firstname String surname String bloodType String birthdate String address String telephone String available String last_login_ip String last_login_date String last_login_time String last_city

3.1.2 BloodType Class

Class Name: BloodType
Description: This model will be enum type and will include all the blood types with its visual image representation.
Methods: BloodType(String type, int imageResource) constructor String getType() int getImageResource() String toString() and Enumerated names are listed below A_PLUS("A+", R.drawable.ic_blood_type_a_plus) A_MINUS("A-", R.drawable.ic_blood_type_a_minus) B_PLUS("B+", R.drawable.ic_blood_type_b_plus) B_MINUS("B-", R.drawable.ic_blood_type_b_minus) AB_PLUS("AB+", R.drawable.ic_blood_type_ab_plus) AB_MINUS("AB-", R.drawable.ic_blood_type_ab_minus) O_PLUS("O+", R.drawable.ic_blood_type_o_plus) O_MINUS("O-", R.drawable.ic_blood_type_o_minus)
Variables: int imageResource String type

3.1.3 Event Class

Class Name: Event
Description: This class will hold the data about the each event. Each event will be an instance of Event class.
Methods: int getID() Location getLocation() String getNotificationMessage() String getHospitalName() Location getHospitalLocation() String getNotificationType() BloodType getBloodType()

and the setter methods for these
Variables: int id Location location String notification_message String hospital_name Location hospital_location String notification_type BloodType blood_type

3.1.4 BlogPost Class

Class Name: BlogPost
Description: This class will include the Blog instance as will be shown on the Blog Page.Each blog post will be an instance of this class.
Methods: int getID() String getPostName() String getTimeStamp() String getContent()
Variables: int id String post_name String timestamp String blog_content

3.1.5 Notification Class

Class Name: Notification
Description: Notifications are will be created and managed in this class.
Methods:

public void createNotification();
public void sendNotification();
public Context getContext();

Variables: String NotificationType String PatientName String BloodType

3.1.6 Location Class

Class Name: Location
Description: Location information will be saved as an instance of that class.
Methods: getters and setters
Variables: String placeName double latitude double longitude

3.1.7 BloodHubAbout Class

Class Name: BloodHubAbout
Description: This class include data for the About page for the BloodHub.
Methods: getters and setters
Variables: String companyName String website String email String phone

3.1.8 Contact Class

Class Name: Contact
Description: This class will include the details of the bloodhub. When users try to communicate with Bloodhub via the Contact Us page, settings/ models in that class will be used.
Methods: String getContact() String getEmail() String getMessage() String getName() void setContact(String str) void setEmail(String str) void setMessage(String str) void setName(String str)
Variables: String contact String email String message String name

There will be additional packages inside the model package in order to store settings for the common variables in the models and these settings are for the model packages.

3.2 Setting Package

3.2.1 EM5Preferences Class

Class Name: EM5Preferences
Description: This class will include the settings and the parameters for the EM5(emergency five). Emergency Five is thought to be a protocol to provide emergency communication with the desired 5 people on the phone in emergency situation. This class will be used to import constants and the setting of the EM5 to other classes for a signed in user.
Methods: String[5] getPeopleName()

int[5] getPeopleID() boolean getIsConfigured() void setPeopleName(String[5] peopleName) void setPeopleID(int[5] peopleID) void setisConfigured(boolean isConfigured)
Variables: String[5] peopleName int[5] peopleID boolean isConfigured

3.2.2 Notification Class

Class Name: Notification
Description: This class will include the settings and the parameters for the Notification
Methods: boolean isEmail() boolean isSms() PushSettings getPush() void setEmail(boolean email) void setPush(PushSettings push) void setSms(boolean sms)
Variables: boolean email PushSettings push boolean sms

3.2.3 Phone Class

Class Name: Phone
Description: This class will include the settings and the parameters for the Phone. There will be 3 types of phone such as work, home and mobile.
Methods: String getMobile() String getWork()

String getHome() void setHome(String home) void setMobile(String mobile) void setWork(String work)
Variables: String home String mobile String work

3.2.4 Push Class

Class Name: Push
Description: This class will include the settings and the parameters for the Push that are sent over the app to the API.
Methods: boolean isDonors() boolean isInsideTheArea() boolean isJourney() void setDonors(boolean donors) void setisInsideTheArea(boolean insideTheArea) void setRewards(boolean rewards) boolean canEqual(Object other) boolean equals(Object o)
Variables: boolean donors boolean insideTheArea boolean rewards

3.2.5 Settings Class

Class Name: Settings
Description: This class is created to merge other settings in to one master class.
Methods:

Notification getNotifications() Push getPushes() EM5Preferences getEM5Preferences() Phone getPhones() void setNotifications(NotificationSettings notifications) void setPushes(Push pushes) void setEM5Preferences(EM5Preferences em5) void setPhone(Phone phones) boolean equals(Object o) boolean canEqual(Object other)
Variables: Notification notifications Push pushes Phone phones EM5Preferences em5

3.3 Reward Package

3.3.1 ClaimedReward Class

Class Name: ClaimedReward
Description: This class will be used to identify and create the reward for the specific user after donation. This class extends reward class and adds additional functionalities such as delivery of the reward/badge and till when this badge/reward will stay.
Methods: int getValidDays() String toString() Delivery getDelivery() void setValidDays(int validDays) void setDelivery(Delivery delivery) boolean equals(Object o) boolean canEqual(Object other)

Variables:
Below variables are objects of the inner classes inside the Claimed Reward
protected Delivery delivery;
protected int validDays;
public enum DeliveryType {
IMAGE
}

3.3.1 Reward Class

Class Name: ClaimedReward
Description: This class will be used to identify and create the reward for the specific user after donation. This class is a basic core of the claimed reward class. In there we only have types of the rewards, categories of the rewards.
Methods: int getInventory() int getOrdinal() ArrayList<String> getCategories() void setInventory(int inventory) void setCategories(ArrayList<String> categories) void setOrdinal(int ordinal) boolean equals(Object o) boolean canEqual(Object other)
Variables: ArrayList<String> categories int inventory int ordinal

5. References

[1] "BloodHub REST API Documentation”, 2018. [Online]. Available: http://cs491-2.mustafaculban.net/api/documentation [Accessed: February 16, 2018].
[2] "Some REST Design Patterns”, 2018. [Online]. Available: http://www.jopera.org/files/SOA2009-REST-Patterns.pdf [Accessed: February 14, 2018].
[3] " BloodHub API Documentation", 2018. [Online]. Available: http://goo.gl/F3iXvQ. [Accessed: February 16, 2018].