
Bilkent University

Department of Computer Engineering

# Senior Design Project

*Project short-name: Bloodhub.*

## Analysis Report

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

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

Progress Report
Nov 6, 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.

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 information. 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.

**Proposed System**

Bloodhub aims to help users that need a blood transfusion. Our main purpose to have more conscious benefits of donating blood and attract donors to get donation. However, It seems that there are some problems which are shelf life management, time and constant supply arrangement for donating blood. In order to solve these problems, we planned to develop mobile application that bring many convenience to this process. 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.

**Overview**

**Functional Requirements**

- Users should be able to confirm user agreement before entering application.
- Users should have their own unique password and user name so that users can enter application using them via "Log In" page.

- Users who want to donate blood should be able to provide some prior condition such as optimal weight, have or not to have tattoo etc.
- Users should be able to provide information about selected friend or relatives who has same blood group with them for emergency blood donation.
- The system should provide information about close blood donation tents, blood donation camps, hospitals to users who want to donate blood using their location information. Besides that the system should be able to show these health centers in Google Map.
- The system should send notifications about close blood donor to users who make a request for specific blood group if users' blood group match with close donor's blood group.
- The system should send notifications to users in order to inform blood donation campaigns. Users should be able to shut off these notifications.

**Non-Functional Requirements**

**Usability**

Our application should be used by anyone regardless of their age or knowledge about software systems. Our application should provide simple, clean and easy user interfaces .Users must not have any problems with donating blood or requesting blood donation. We can say that it will be user friendly and stable

**Reliability**

Our application should continuously collect real-time data about blood donation tents and blood donation camps. Users should not be able to change any information related to other users. Application should require an account to login.

**Portability**

Our application should be able to be used in various hardware and software platforms.

**Efficiency**

Our application should be quick to match requests and donations.

The response time of the application should be less than 100 milliseconds.

The creating  request of blood donation should take less than 10 second.

**Extensibility**

Our application should be ready to new features and functionalities.

Our application should be both extensible frontend and backend therefore new donation's functionalities can be added easily.

**Supportability**

Our application should require Internet connection

Our application should support Android and iOS. Users should be able to create their unique account using their own real name, citizenship number, blood group information, e-mail address, date of birth and telephone number. Besides that users should be able to add their home address optionally.

**Pseudo Requirements**

- System will work on PHP,CSS,JS,HTML on backend.
- Android and IOS mobile applications will be developed.
- MySQL will be used for database storage.
- Google APIs will be used.

**System Models**

**Use case Model**

### 3.5.1 Scenarios

#### 3.5.1.1 Create Account

- **Use Case Name**: Create Account
- **Actor**: User
- **Entry Condition**: User has the application and connected the Internet.
- **Exit Condition**: User has successfully created account.
- **Flow of Events**:
    - User is asked to fill the informations which are necessary to use the application.
    - User filled all of the information screen.
    - System sent authentication mail to user's e-mail address.
    - User succesfully created his or her account.
- **Alternative Flows**:
    - User entered wrong or missing information. System asked for entering their informations again.

#### 3.5.1.2 Login

- **Use Case Name**: Login
- **Actor**: User
- **Entry Condition**: User has created an account.
- **Exit Condition**: User has logged in successfully.
- **Flow of Events**:
    - User opened the application.
    - User has entered account informations.
    - Because of login informations are true, user entered the system succesfully.
- **Alternative Flows**:

- User types wrong informations. System asks again to enter account name and password.

#### 3.5.1.3 Request Blood

- **Use Case Name**: Request Blood
- **Actor**: User
- **Entry Condition**: User has logged in succesfully.
- **Exit Condition**: User has created blood emergency in the system..
- **Flow of Events**:
  - User logged in succesfully.
  - User selected blood emergency section from top menu.
  - User enters the type of blood which is urgently needed.
- **Alternative Flows**:
  - User has made a mistake and enter wrong type of blood.
  - User has updated type of blood and shared again.

#### 3.5.1.4 Search For Blood Donation Center

- **Use Case Name**: Search For Blood Donation Center
- **Actor**: User
- **Entry Condition**:  User has logged in succesfully.
- **Exit Condition**: User arrives the blood center.
- **Flow of Events**:
  - User logged in succesfully.
  - User selected “search for blood center” section from top menu.
  - A map based on user’s current location occured on the screen.
  - User selected one of the blood centers from the map.
  - System started to navigate user to the center.
  - User arrived the center.
- **Alternative Flows**:
  - There was no blood center nearby user’s location.
  - User gave up from his decision and closed the navigation.

#### 3.5.1.5 Notifying User About Blood Centers

- **Use Case Name**: Notifying User About Blood Centers
- **Actor**: System
- **Entry Condition**: User's location services are open and connected to Internet..
- **Exit Condition**: User is notified via application.
- **Flow of Events**:
    - System checks whether there is an update in locations of Kızılay squads.
    - According to location updates, application check the users nearby.
    - System send notifications these users.

#### 3.5.1.6 Notifying User About Blood Needs

- **Use Case Name**: Notifying User About Blood Needs
- **Actor**: System
- **Entry Condition**: User's location services are open and a blood request is created.
- **Exit Condition**: System sent SMS and notifications via application.
- **Flow of Events**:
    - System checks whether there is an blood request.
    - Based on request's location system scans for users in 100 km diameter area.
    - System sends notification and SMS these users.

#### 3.5.1.7 Editing Relatives

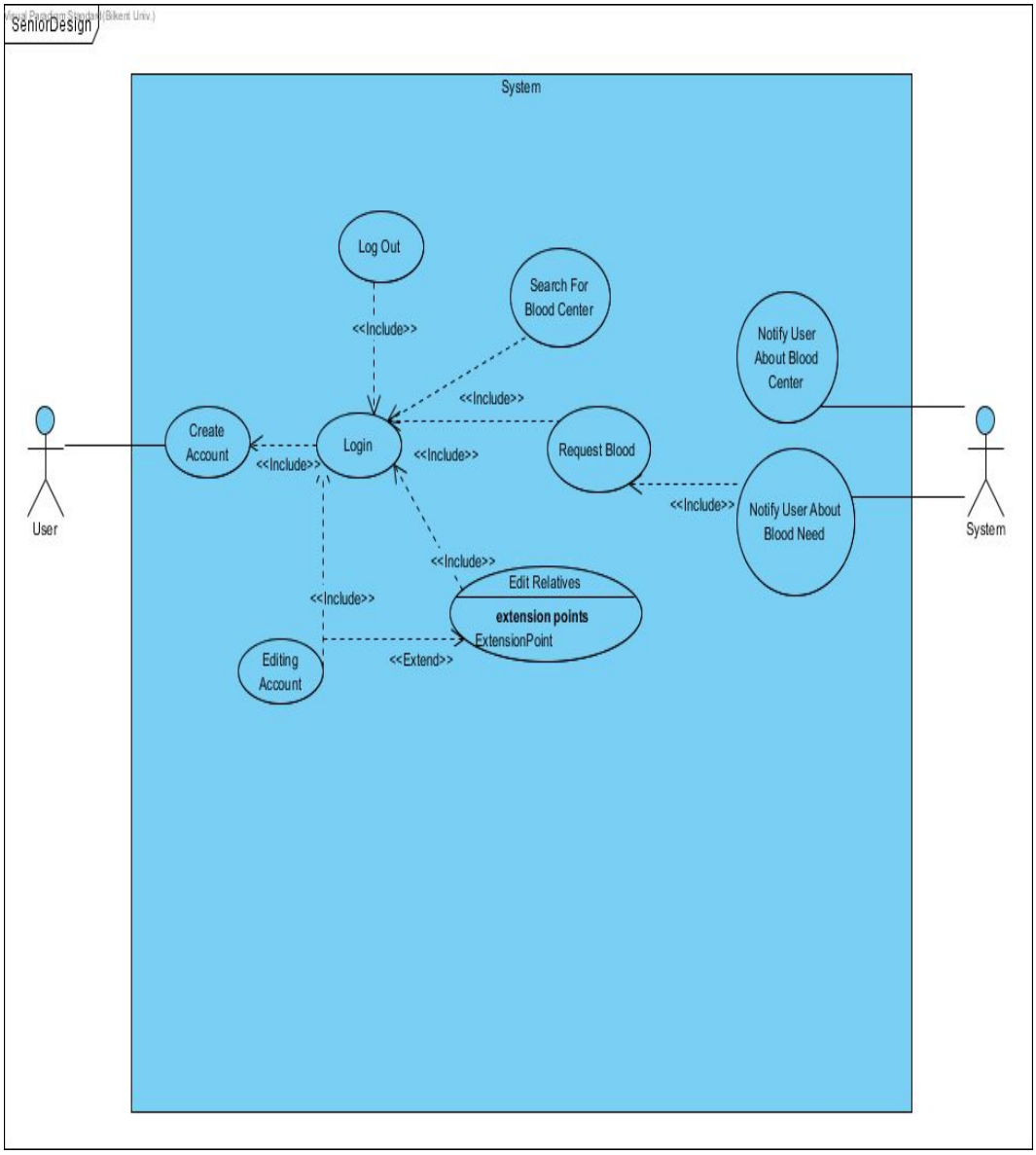
- **Use Case Name**: Editing Relatives
- **Actor**: User
- **Entry Condition**: User has an account and opened the application.
- **Exit Condition**: User has succesfully edited his relatives and these new relatives accepted the request.
- **Flow of Events**:
    - User opened the account page
    - User add people who have same blood type and easy to reach in emergency situations.
    - System send authentication SMS to these added people.
    - People accept invitations.
    - “My Relatives” section in Account page get updated.
- **Alternative Flows:** Users which are set as relative by another user didn’t accept the invitation and process terminated.

#### 3.5.1.8 Editing Account Informations

- **Use Case Name**: Editing Account Informations
- **Actor**: User
- **Entry Condition**: User has an account and opened the application.
- **Exit Condition**: User has succesfully edited his account informations.
- **Flow of Events**:
    - User opened the account page
    - User clicks “Edit Account”.
    - Users change some of the informations.
    - User save changes

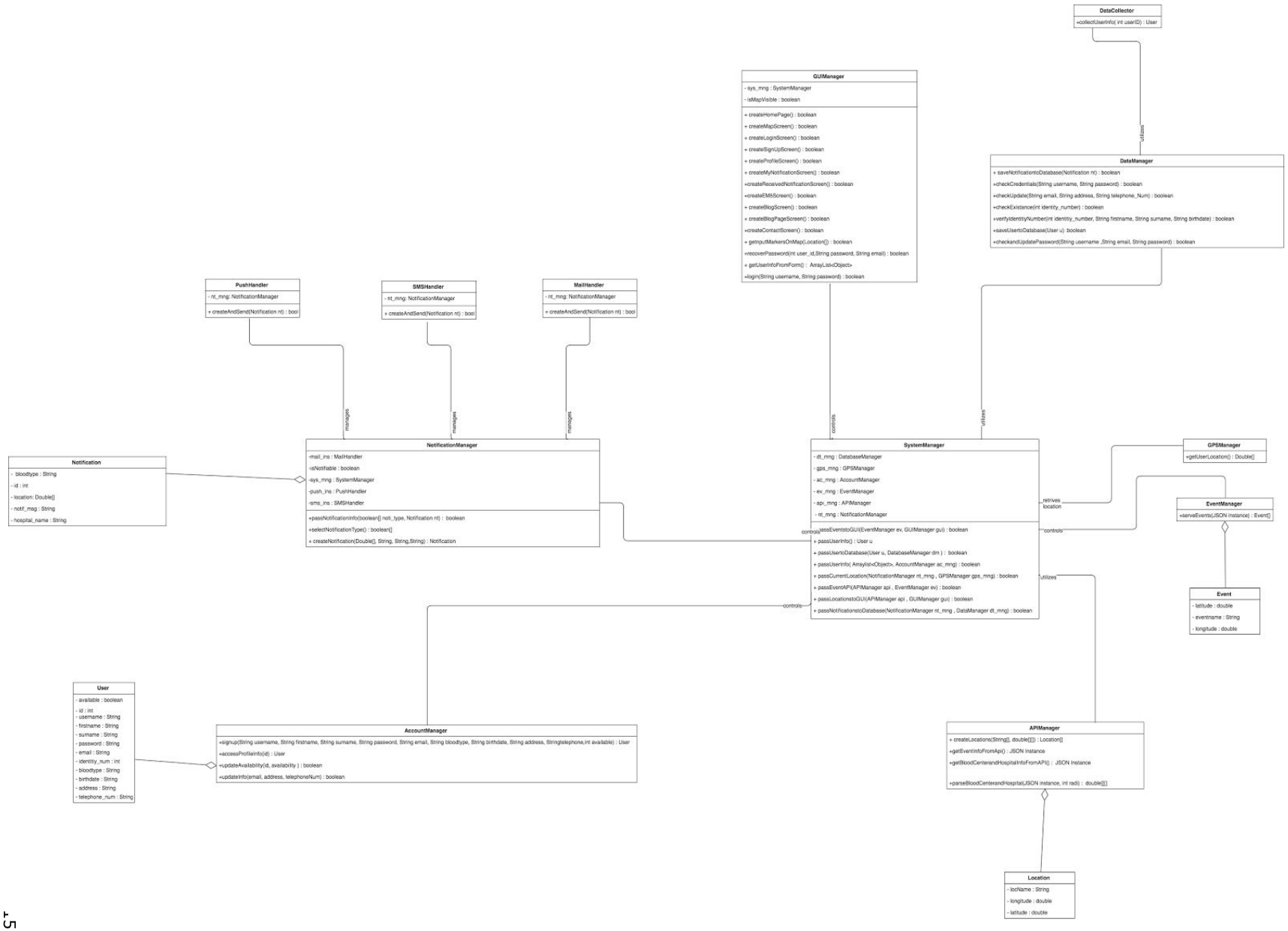
#### 3.5.1.9 Logout

- **Use Case Name**: Logout
- **Actor**: User
- **Entry Condition**: User logged in succesfully.
- **Exit Condition**: User logged out succesfully.
- **Flow of Events**:
    - User logged in the system .
    - User spent time in application.
    - User logged out the system.




**Use Case Diagram**

## Object and Class Model


DataCollector
+collectUserInfo( int userID) : User
utilizes
GUIManager
- sys_mng : SystemManager
- isMapVisible : boolean
+ createHomePage() : boolean
+ createMapScreen() : boolean
+ createLoginScreen() : boolean
+ createSignUpScreen() : boolean
+ createProfileScreen() : boolean
+ createMyNotificationScreen() : boolean
+createReceivedNotificationScreen() : boolean
+createEMSScreen() : boolean
+ createBlogScreen() : boolean
+ createBlogPageScreen() : boolean
+createContractScreen() : boolean
+ getInputMarkersOnMap(Location[]) : boolean
+recoverPassword(int user_id,String password, String email) : boolean
+ getUserInfoFromForm() : ArrayList<Object>
+login(String username, String password) : boolean
DataManager
+ saveNotificationtoDatabase(Notification nt) : boolean
+checkCredentials(String username, String password) : boolean
+checkUpdate(String email, String address, String telephone_Num) : boolean
+checkExistance(int identity_number) : boolean
+verifyIdentityNumber(int identity_number, String firstname, String surname, String birthdate) : boolean
+saveUsertoDatabase(User u) :boolean
+checkandUpdatePassword(String username ,String email, String password) : boolean
PushHandler
- nt_mng: NotificationManager
+ createAndSend(Notification nt) : bool
SMSHandler
- nt_mng: NotificationManager
+ createAndSend(Notification nt) : bool
MailHandler
- nt_mng: NotificationManager
+ createAndSend(Notification nt) : bool
manages
manages
manages
controls
utilizes
Notification
- bloodtype : String
- id : int
- location: Double[]
- notif_msg : String
- hospital_name : String
NotificationManager
-mail_ins : MailHandler
-isNotifiable : boolean
-sys_mng : SystemManager
-push_ins : PushHandler
-sms_ins : SMSHandler
+passNotificationInfo(boolean[] noti_type, Notification nt) : boolean
+selectNotificationType() : boolean[]
+ createNotification(Double[], String, String,String) : Notification
SystemManager
- dt_mng : DatabaseManager
- gps_mng : GPSManager
- ac_mng : AccountManager
- ev_mng : EventManager
- api_mng : APIManager
- nt_mng : NotificationManager
+passEventstoGUI(EventManager ev, GUIManager gui) : boolean
+ passUserInfo() : User u
+ passUsertoDatabase(User u, DatabaseManager dm ) : boolean
+ passUserInfo( ArrayList<Object>, AccountManager ac_mng) : boolean
+ passCurrentLocation(NotificationManager nt_mng , GPSManager gps_mng) : boolean
+ passEventAPI(APIManager api , EventManager ev) : boolean
+ passLocationstoGUI(APIManager api , GUIManager gui) : boolean
+ passNotificationstoDatabase(NotificationManager nt_mng , DataManager dt_mng) : boolean
retrives location
controls
controls
utilizes
GPSManager
+getUserLocation() : Double[]
EventManager
+serveEvents(JSON instance) : Event[]
Event
- latitude : double
- eventname : String
- longitude : double
controls
User
- available : boolean
- id : int
- username : String
- firstname : String
- surname : String
- password : String
- email : String
- identity_num : int
- bloodtype : String
- birthdate : String
- address : String
- telephone_num : String
AccountManager
+signup(String username, String firstname, String surname, String password, String email, String bloodtype, String birthdate, String address, Stringtelephone,int available) : User
+accessProfileInfo(id) : User
+updateAvailability(id, availability ) : boolean
+updateInfo(email, address, telephoneNum) : boolean
APIManager
+ createLocations(String[], double[][]) : Location[]
+getEventInfoFromApi() : JSON Instance
+getBloodCenterandHospitalInfoFromAPI() : JSON Instance
+parseBloodCenterandHospital(JSON instance, int radi ) : double[][]
Location
- locName : String
- longitude : double
- latitude : double

## CLASSES OF BLOODHUB

### GUIManager

This class creates all different screens of our application. It puts markers of hospitals , blood centers and events on map then provides directions to selected point by using GPS services if user prefers.It takes necessary information for creating a notification from user on relevant screen. Also , it gets user information which will affect the user access to map and be saved into database.

+ createHomePage() : boolean -Creates home page.

+ createMapScreen() : boolean -Creates map screen.

+ createLoginScreen() : boolean-Creates login screen.

+ createSignUpScreen() : boolean-Creates signup screen.

+ createProfileScreen() : boolean-Creates profile screen.

+ createMyNotificationScreen() : boolean-Creates mynotifications screen.

+createReceivedNotificationScreen() : boolean-Creates receivednotifications screen.

+createEM5Screen() : boolean-Creates EM5 screen.

+ createBlogScreen() : boolean-Creates blog screen.

+ createBlogPageScreen() : boolean-Creates blogpage screen.

+createContactScreen() : boolean-Creates contact screen.

+ getnputMarkersOnMap(Location[]) : boolean -Puts markers of locations and events on map screen.

+ getUserInfoFromForm() : ArrayList<Object> -Gets user information from questionairre.

+login(username,password) : boolean -Gets user input when user logged in.

+recoverPassword(user_id,password, email) : boolean -When the user forgets and choose ‘I forgot my password’ , takes required information (user_id,new password and email) from user.

**NotificationManager**

Creates notification objects according to information determined by user and sends an array of notifications relevant handlers depending on the notification type.

+ createNotification(Double[], String, String,String) : Notification - Creates notification depending on informations taken from user by GUI Manager

+selectNotificationType() : boolean[] - Every elements of the array that the method returns , represents a way to send notification(Push message , sms or email).User can choose some of them or all of them . Notification manager uses the instances of notification ways selected according to this array , to send notifications.

+passNotificationInfo(boolean[] noti_type, Notification nt) : boolean - Passes information to send in a specific way.

**Notification**

Holds notification information.

**PushHandler**

Transforms notifications into push messages and sends.

+ createAndSend(Notification nt) : boolean - Sends notifications as a push message.

**SMSHandler**

Transforms notifications into SMS and sends.

+ createAndSend(Notification nt) : boolean - Sends notifications as SMS.

**MailHandler**

Transforms notifications into e mails and sends.

+ createAndSend(Notification nt) : boolean - Sends notifications as an email.

**GPSManager**

Gets users current location which is requred to determine limited area in which user receives notifications from other users and sees events.

+getUserLocation() : Double[] - Returns user's current location as an array.

**EventManager**

Determines the events to be displayed on map.

+serveEvents(JSON instance) : Event[] -Creates the event holding the same information as taken from API by parsing the event that is in JSON format.

**Event**

Holds event information.

**APIManager**

Takes necessary informations from APIs to create location objects that will be used to display locations of blood centers and hospitals on map ,and required to create event objects.

+getEventInfoFromApi() : JSON Instance - Takes event information from API in JSON format.

+getBloodCenterandHospitalInfoFromAPI() : JSON Instance -Takes Blood center and hospital information from API in JSON format.

+parseBloodCenterandHospital((JSON instance, radi) : double[][] - Parses received information that is in JSON format to ,make information usable, return coordinates in an array .

+ createLocations(String[], double[][]) : Location[] - Create location objects according to information received as an array.

**Location**

Holds location information of blood centers and hospitals.

**DataManager**

Saves users registered to the system and notifications, checks accuracy of input given by user by comparing them with the values in the databases and handles changed values within system to save into database.

+ saveNotificationtoDatabase(Notification nt) : boolean - Saves notifications of users created within the system into the database.

+checkCredentials(username, password) : boolean -When the user wants to log in , checks the accuracy of given credentials within database.

saveUsertoDatabase(User u) :boolean -Saves users registered to the system into the database

+checkUpdate(email, address, telephone_Num) : boolean - When the user changes personal information , updates database.

+checkExistance(identity_number) : boolean -Checks existence of identity number within our database, to prevent duplicates and fake accounts.

+verifyIdentitiyNumber(identitiy_number, firstname,surname, birthdate) : boolean -When a user wants to sign up , checks accuracy of information by searching given parameters in the database of government to prevent fake accounts.

+checkandUpdatePassword(username, email, password) : boolean -When the user forgets password and requests a new one, updates database according to the new password of the user.

**User**

Holds user information.

**AccountManager**

Handles account operations.

+signup(username, firstname,surname,password,email,bloodtype,birthdate,address,telephone,available) : User u-According to inputs given by the user , returns a user object .

+accessProfileInfo(id) : User -Returns the user having a specific id.

+updateAvailability(id, availability ) : boolean -When user's availability changes depending on personal information or preference of the user , updates the value of the availability.

+updateInfo(email, address, telephoneNum) : boolean - When user changes profile information , updates relevant variables.

**SystemManager**

Manages whole system by gathering instances of relevant classes to use their variables and methods together for every operations within system. That is, this class gathers all other classes to accomplish tasks of whole system. It is the most crucial class of the system.

+ passEventstoGUI(EventManager ev, GUIManager gui) : boolean -Creates interaction between Event Manager and GUIManager to pass events.

+ passUserInfo(id) : User u -Passes user among relevant classes

+ passUsertoDatabase(AccountManager, DataManager ) : boolean -Creates interaction between AccountManager and DatabaseManager to pass events.

+ passUserInfo( Arraylist<Object>, AccountManager ac_mng) : boolean -Passes user information taken from GUIManager to AccountManager

+ passCurrentLocation(NotificationManager nt_mng , GPSManager gps_mng) : boolean -Creates interaction between NotificationManager and GPSManager to pass events.

+ passEventAPI(APIManager api , EventManager ev) : boolean -Creates interaction between APIManager and EventManager to pass events.

+ passLocationstoGUI(APIManager api , GUIManager guı) : boolean -Creates interaction between APIManager and GUIManager to pass events.

+ passNotificationstoDatabase(NotificationManager nt_mng , DataManager dt_mng) : boolean

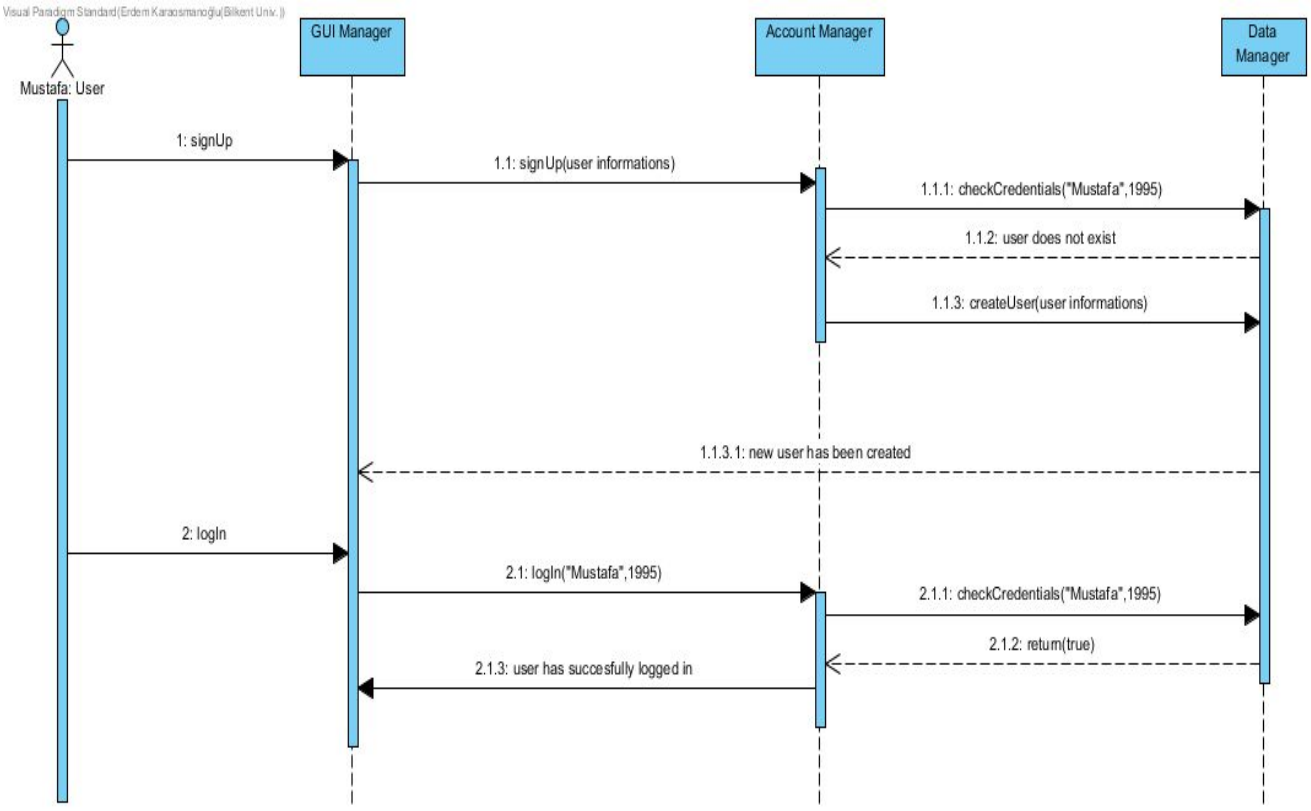
-Creates interaction between NotificationManager and DataManager to pass events.

# Dynamic Models

## Sequence Diagrams

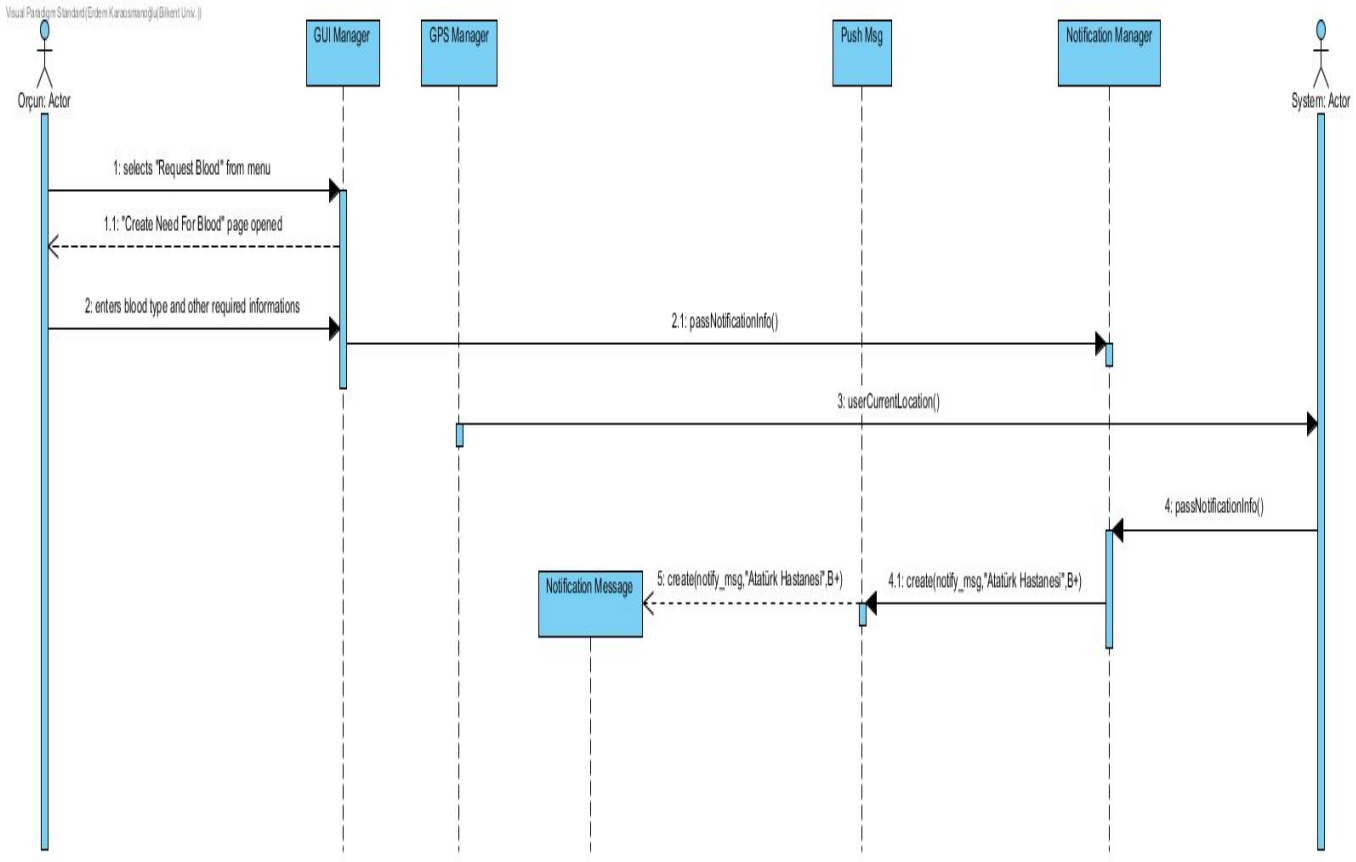
### Create Account and Login

Mustafa downloads “BloodHub” application and wants to create an account. He sets his user name as “Mustafa” and his password as “1995”. He fills the required informations. System checks and accepts his sign up process. After creating his account, Mustafa login the system and starts to use the application.



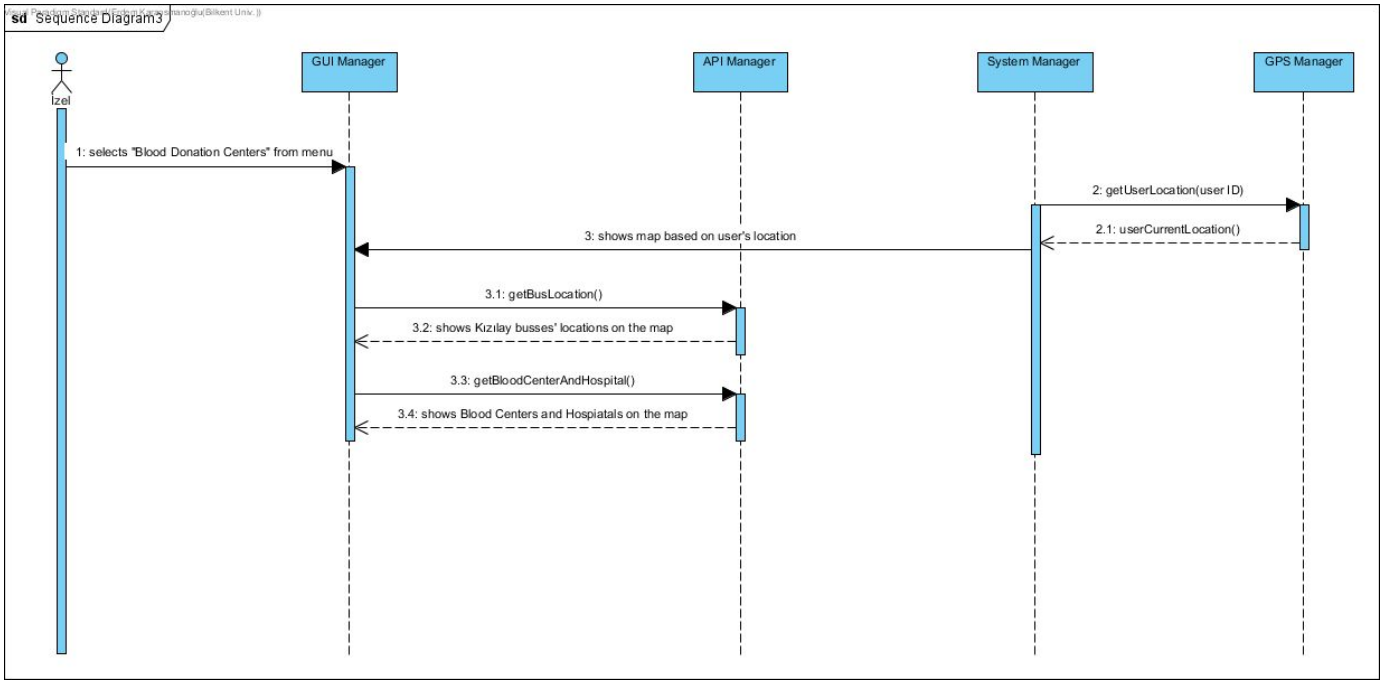
**Request Blood**

This diagram shows " Request Blood " scenario. Orçun needs for blood because of car crash at Atatürk Avenue and thus he selects " Request Blood " choice from the GUI Manager. Then, GUI Manager opens "Create Need For Blood" page for Orçun. He enters blood type(B+) and other required information for request blood donation. Then, information is passed to Notification Manager. System gets Orçun's current location from GPS Manager. Then, System pass notification information for request to Notification Manager. Notification Manager creates request by using Orçun's information and send this request to Push Msg. Finally, Push Msg creates notification message for other users.

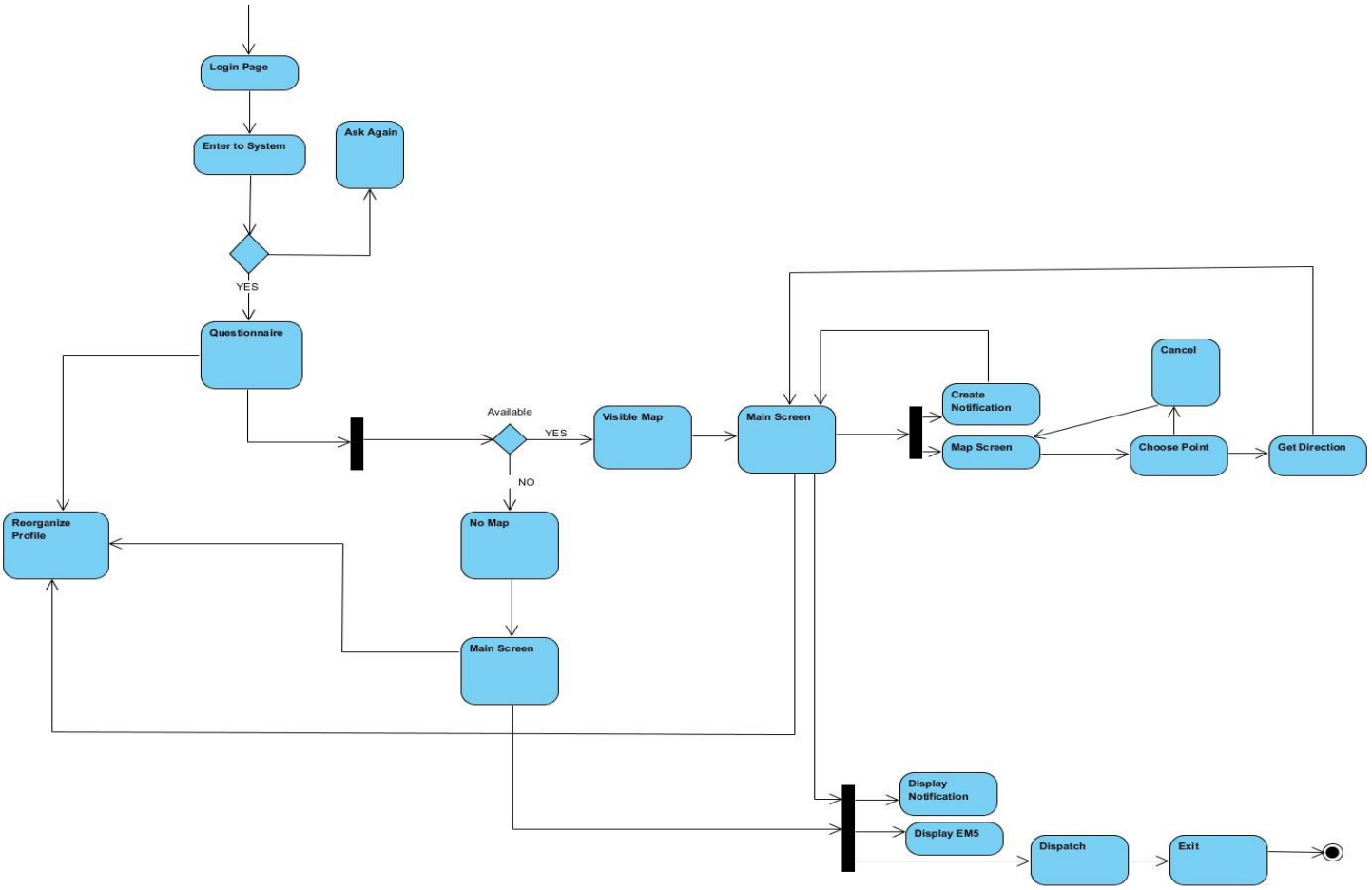


**Donate Blood**

This diagram shows " Donate Blood" scenario. İzel is influenced by advertisements of Kızılay and decides to donate blood. She opens Bloodhub and selects "Blood Donation Centers" choice from the GUI Manager. Then, System Manager gets İzel's current location by using her id from GPS Manager. Then, System Manager shows maps to İzel via GUI Manager. İzel wants to see Blood Center's or Driver's locations. API Manager shows their locations on the map. Then, because İzel is not to be sure about transportation facilities, she gets information about bus locations and API Manager shows locations of bus stops again.

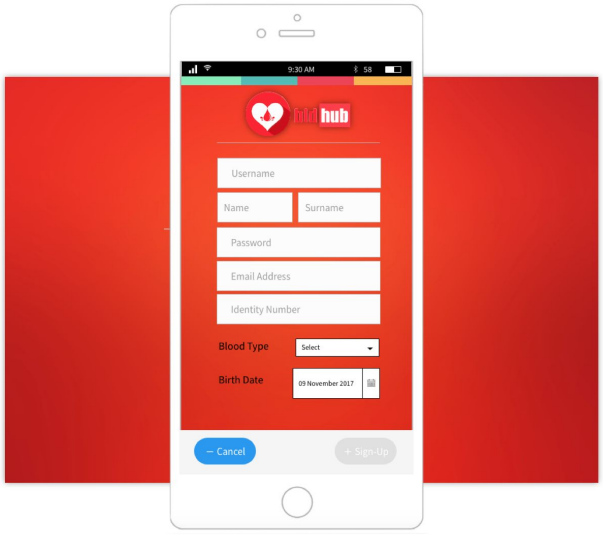


## Activity Diagram



## Mockups and UI Designs

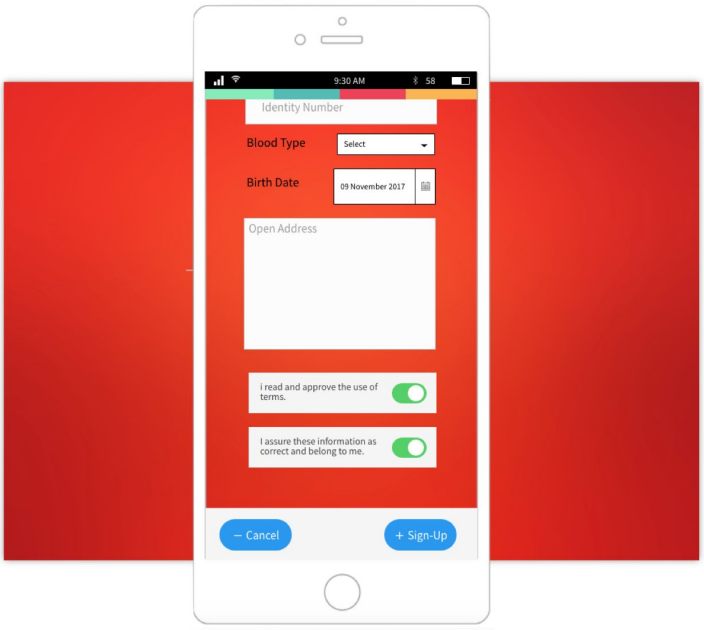
### Sign-up Page

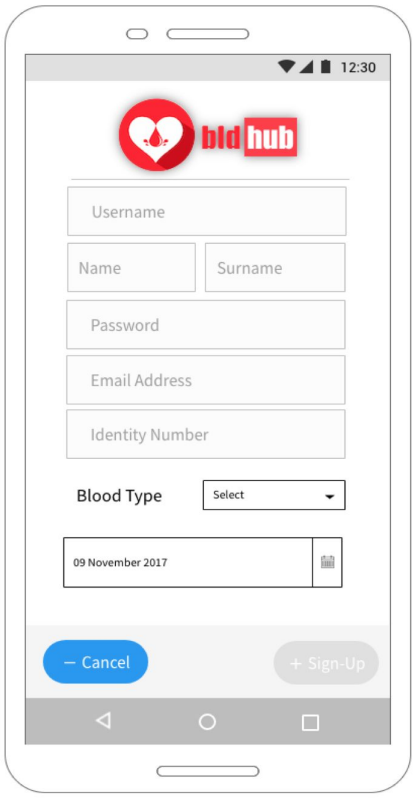



If User wants to sign-up to Bloodhub, they will use this interface. In this interface they need to supply username, name, surname, password, email, identity number, blood type, birth date, open address. He/she needs to read and accepts terms of use and assures the given information that are belong to her/him.

On the left, first of the two parts of the registration interface is shown.

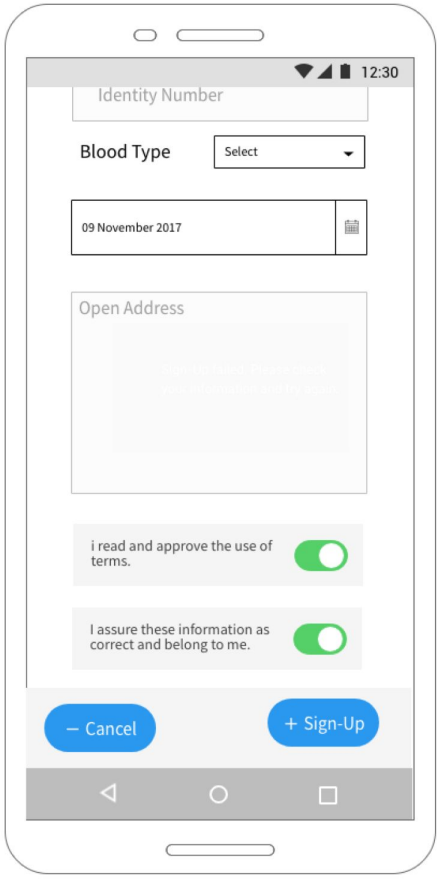
On the right, second part of the sign-up interface is shown. This 2 parts are for the iOS phones. On the next page Android versions of this interface will be shown.



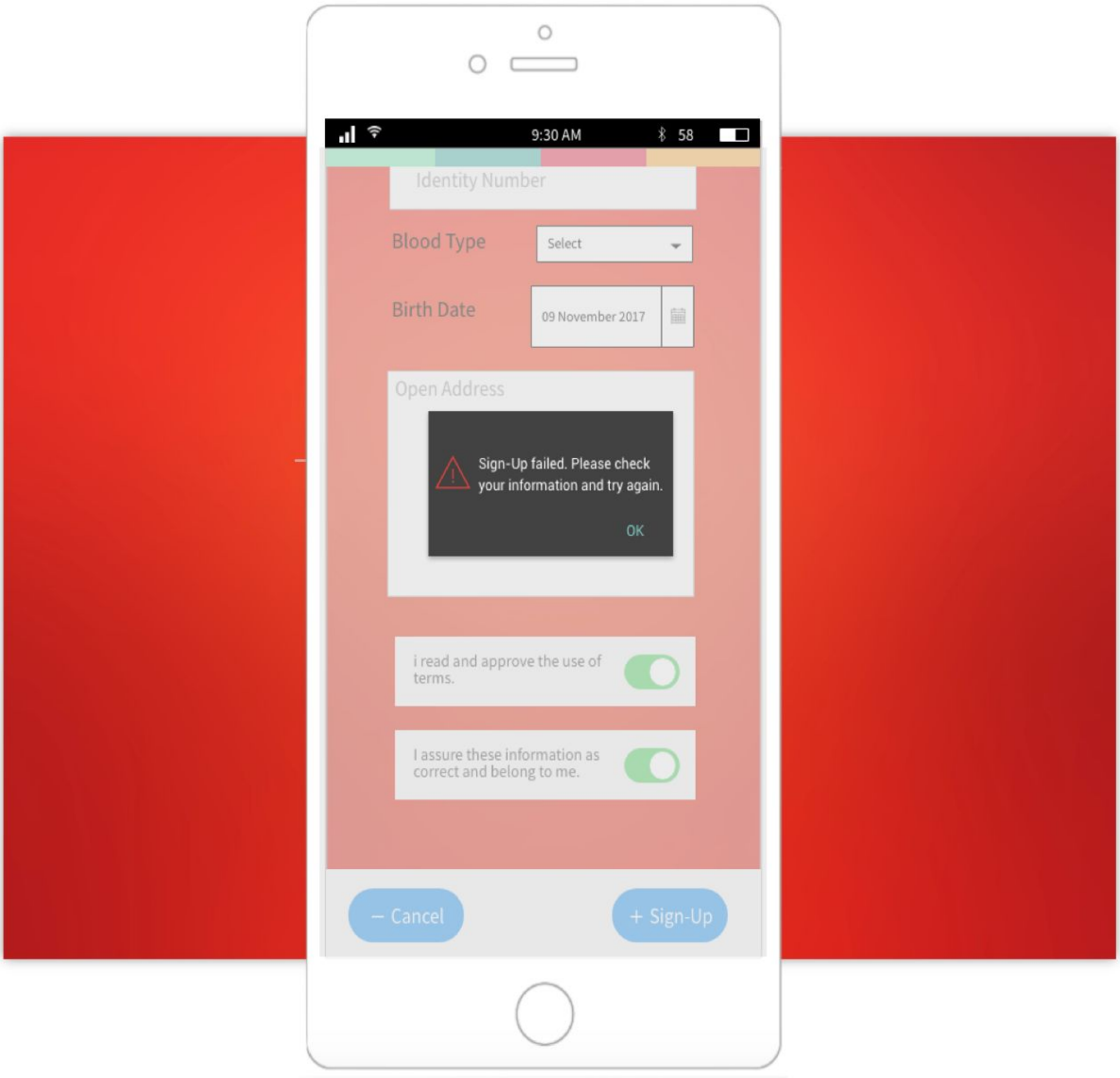



On the top, android version of first of two sign-up interface is shown. It is very similar with the iOS version.

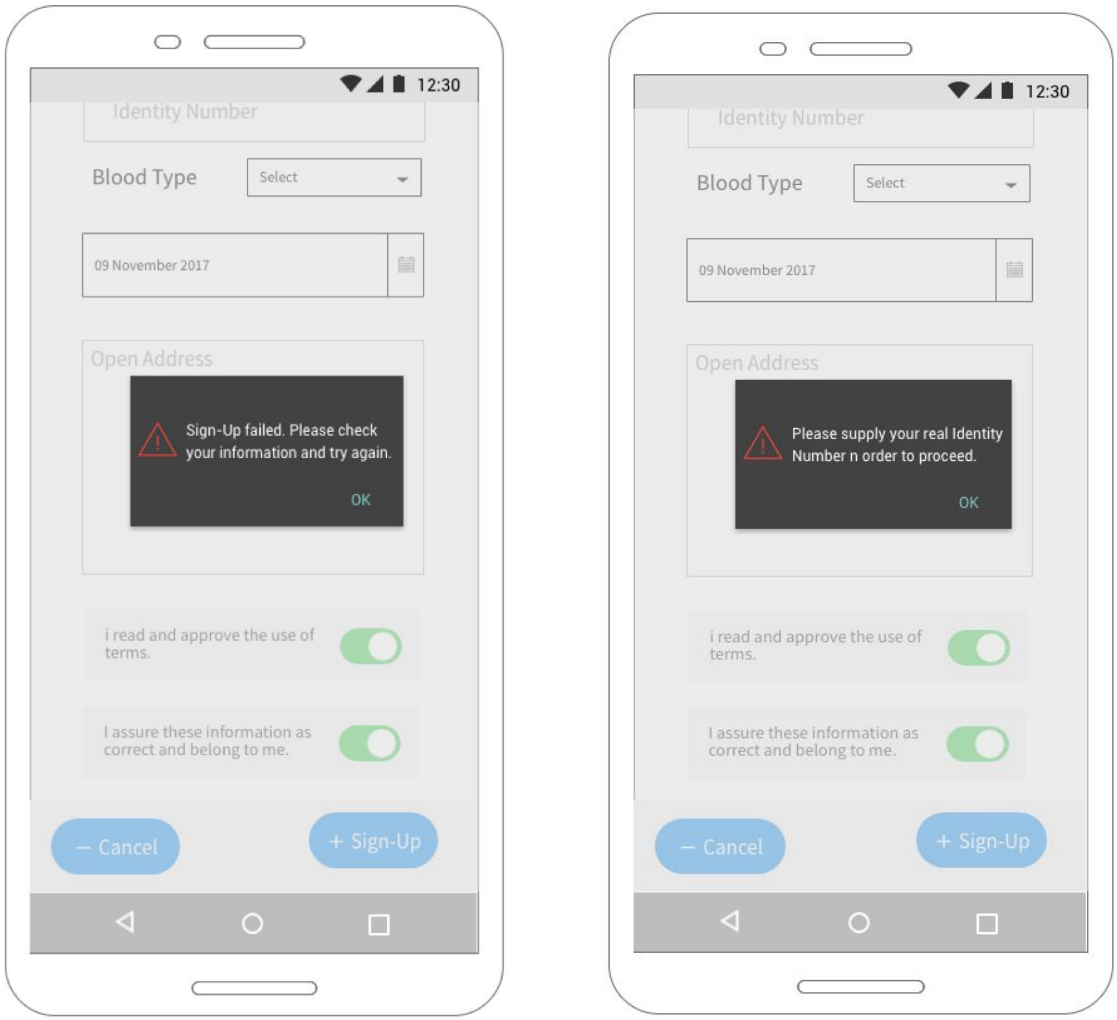
User needs to give sufficient information in order to sign-up successfully. If he/she isn't able to give enough info or, the/she gives with wrong format program will give error about that part, he/she made wrong.




On the top, android version of first of two sign-up interface is shown. It is very similar with the iOS version.



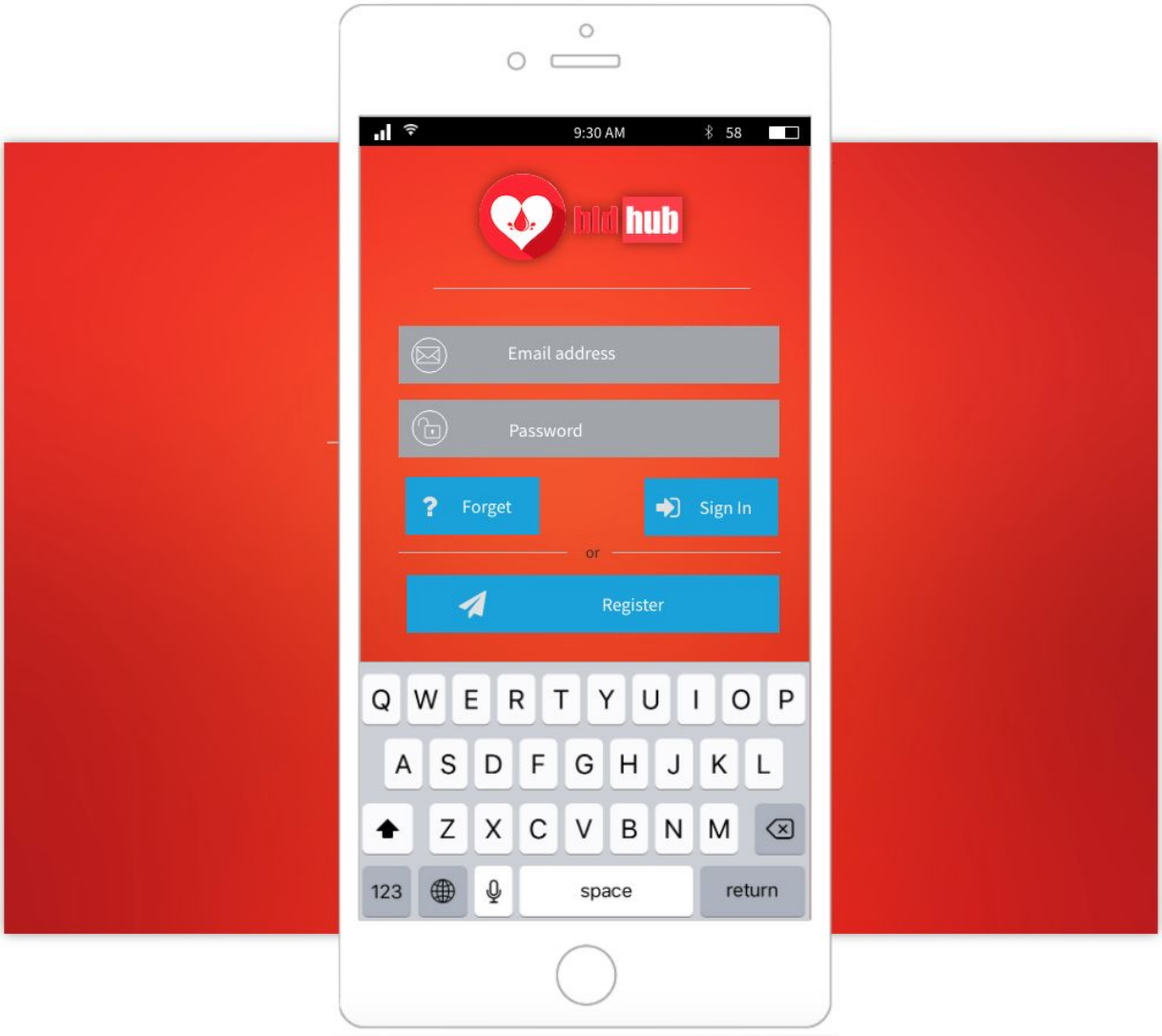

Sign-up failed due to lack of the necessary information. This error messages will be shown both on iOS and android version.



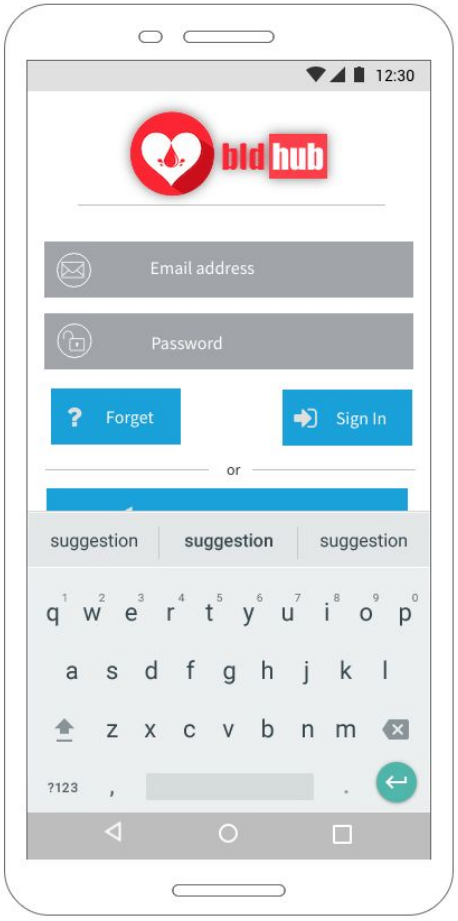

First image shows if user does not give sufficient amount of Info. Program will show an error message about that. On the second interface, program will show an error message if user gives wrong identity no to the program.

## Login Page

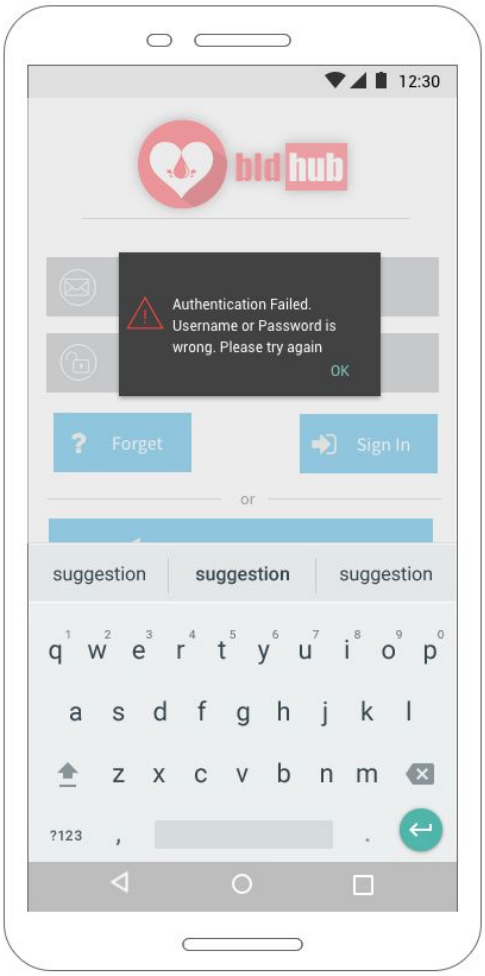
Users of both iOS and Android will be able to login Bloodhub via these interfaces.




Login page for iOS user. On the next page Android version will be shown.

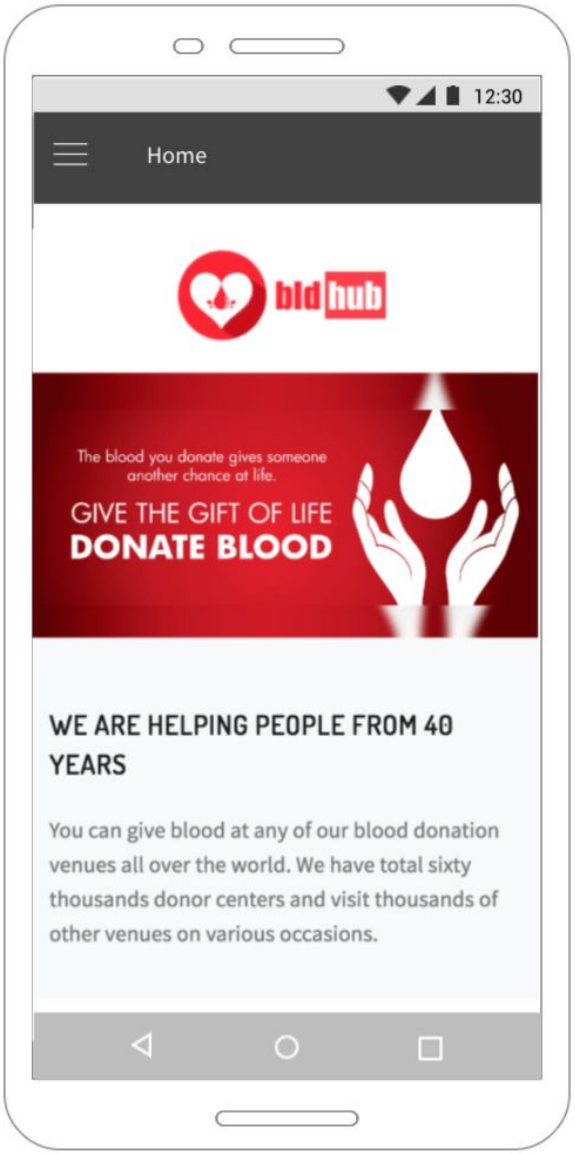



On the left, android version can be seen.
If user supply his/her credentials correctly, he/she will enter to the main screen but if not, user will see error messages about he/she gave wrong credentials.



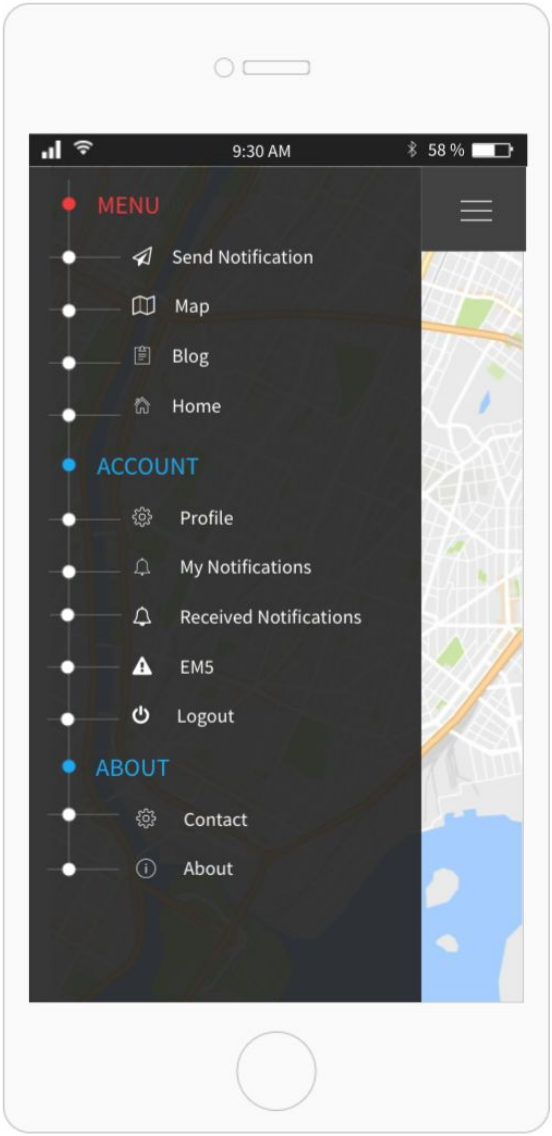

On the right, error message is given which indicates whether username or the password is wrong.

**Homepage**




Home page will be the main screen after user logins successfully. This page will contain many information about how to donate blood, what Kizilay does and other beneficial information.

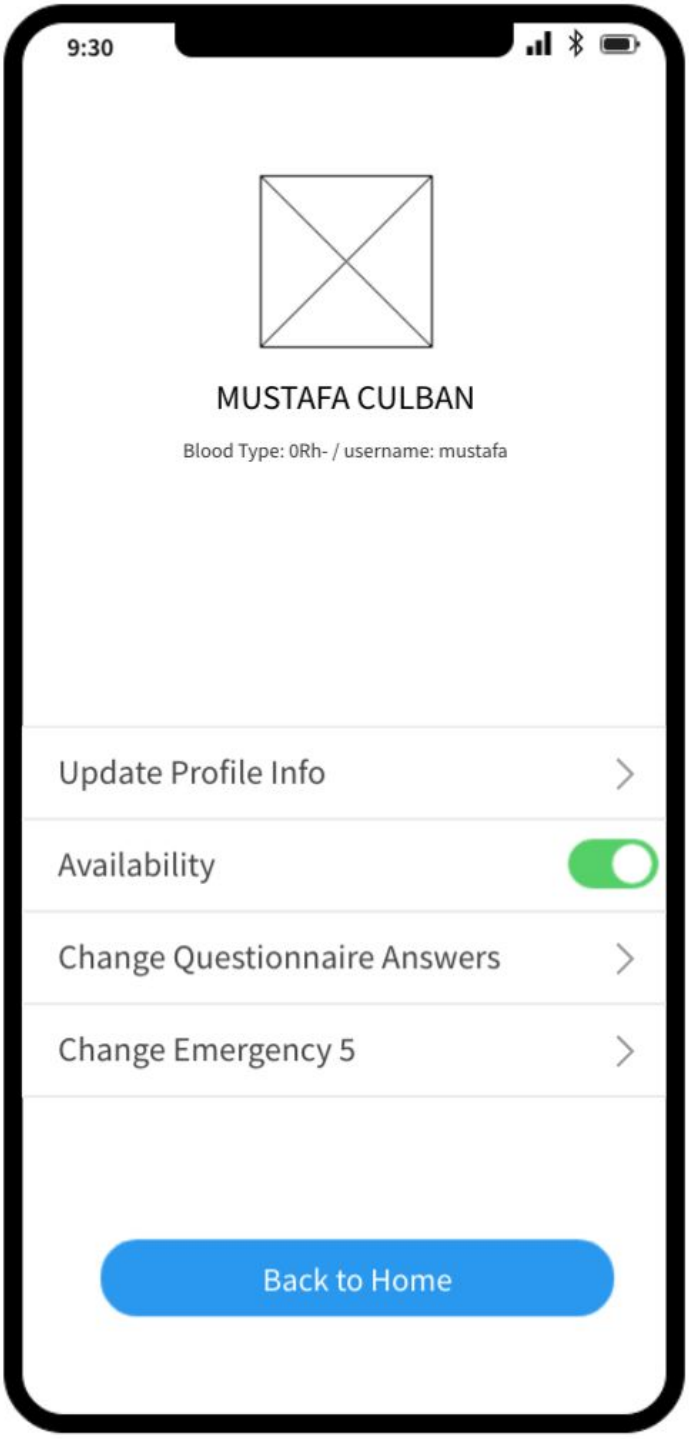
**Pull Menu**




In this pull menu interface, user will see the links to the respective pages that were created by the GUI Manager. User can send notification, see map, read on blog go to home page wherever he/she, see their profile, notifications created by them or notifications that were sent to them, he/she can reconfigure his/her emergency five people, logout, contact with us and read about of us from this menu links.

This menu will be both on iOS and Android but for simplicity only iOs version was show in this report.

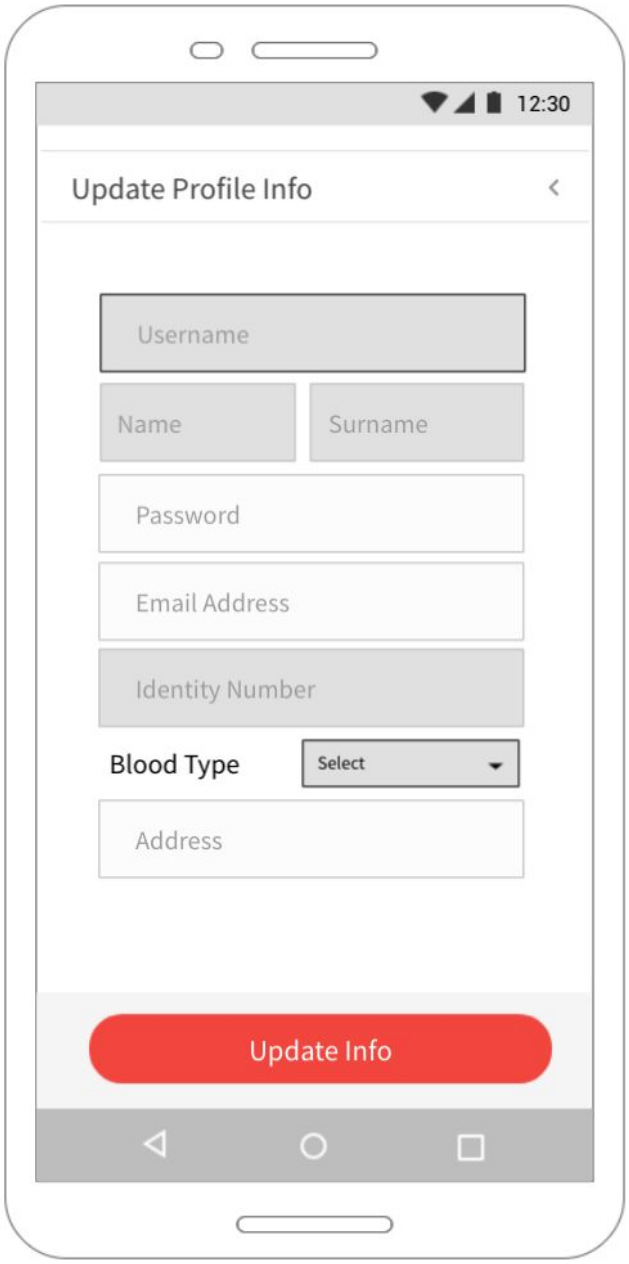
**Profile Page**




User profile information will be shown here such as blood type, name surname and username, profile image whether he/she select one of the predefined pictures.

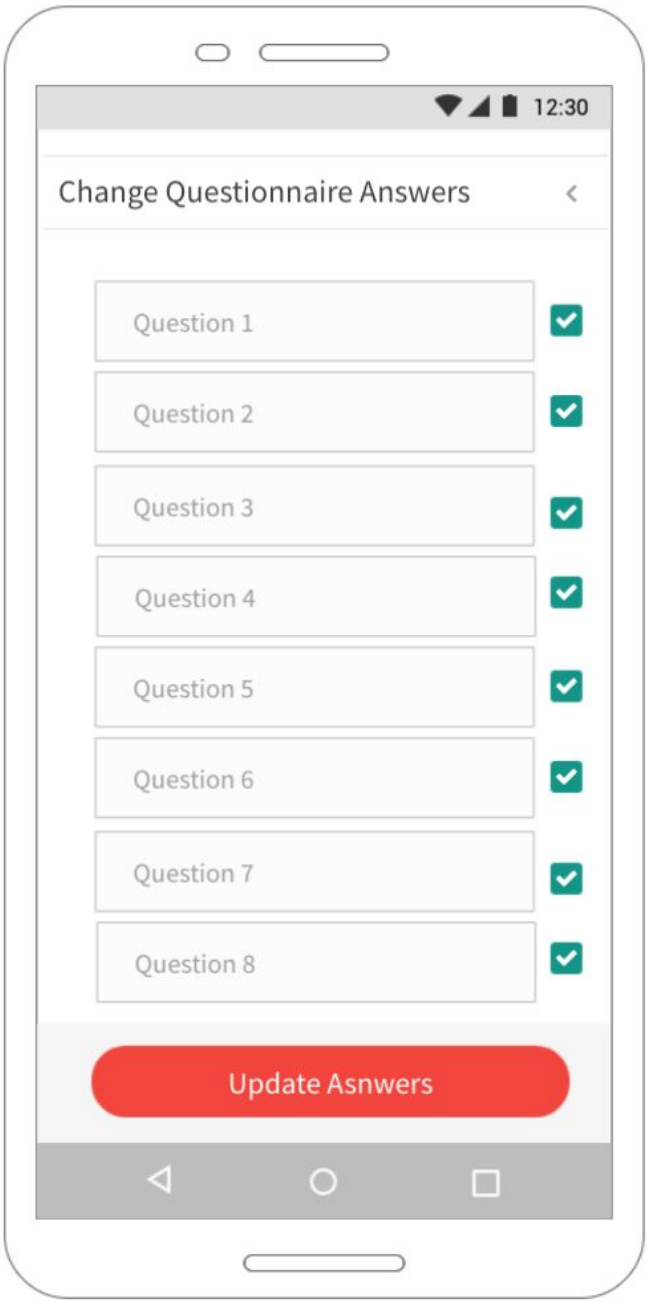
In this page user will see links to change many options such as updating profile information, changing questionnaire answers, change their Emergency Five People and their availability for wanting to take notifications from the system or not.

**Update Profile Info**



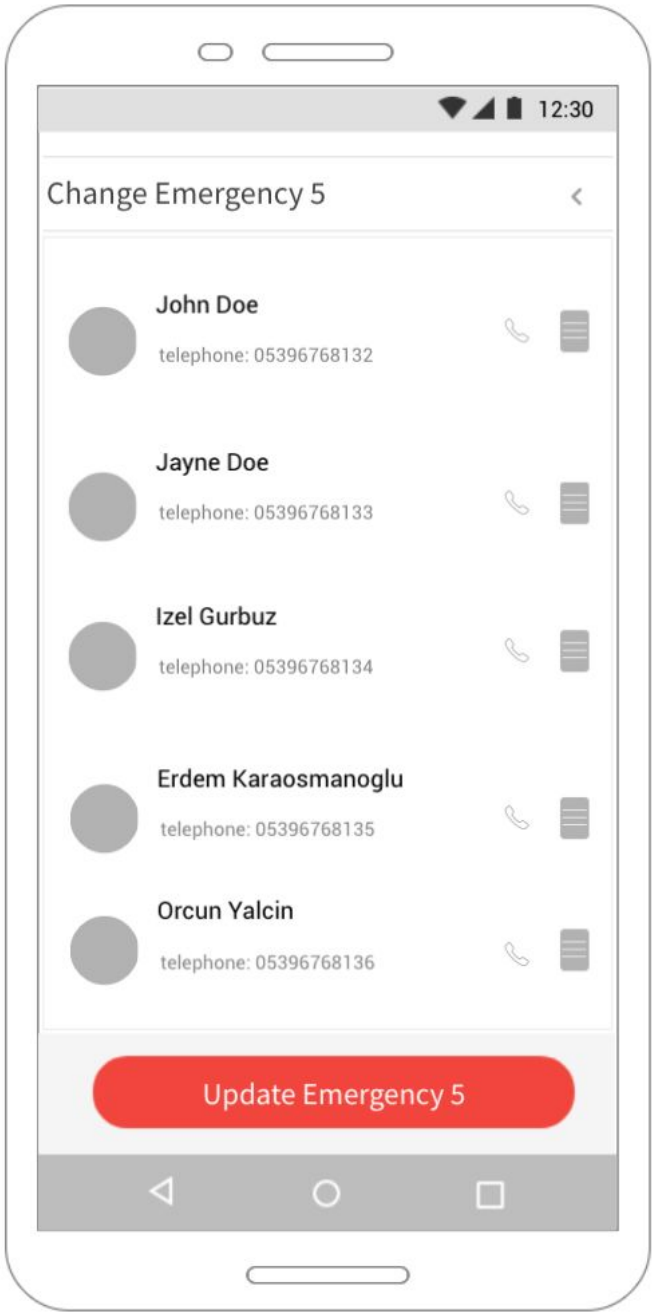

User can change their profile info on this interface but there are some instances such as Identity number, name username, surname blood type cannot be changed by the user. In order to change these variables all or some, user needs to contact with Administration.

**Change Questionnaire Answers Page**



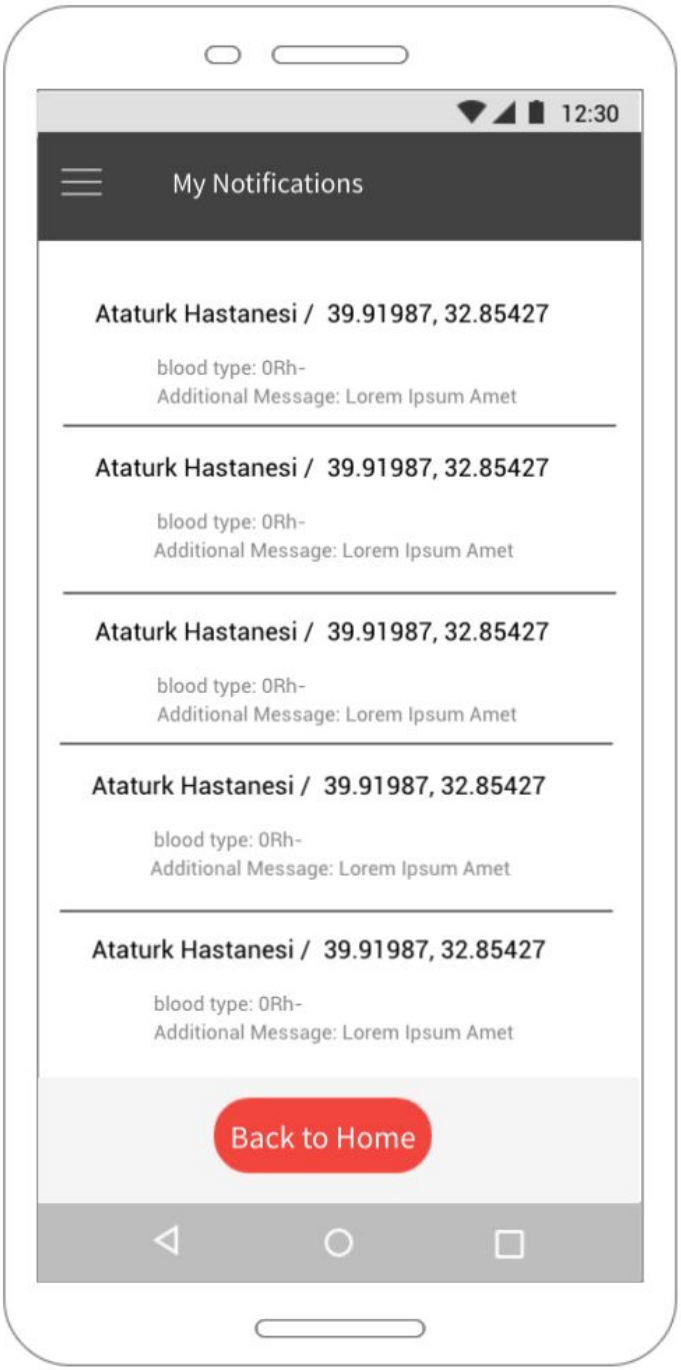

User will be able to change their questionnaire answer to take notification from the system. These question will identify whether user is qualified to donate blood or not. SOme of the questions can be their weight, height, whether they are using antibiotics or not etc.

**Change Emergency Five**




User will be able to view and change their Emergency five to contact with them in an urgent time. He/she can arrange the order to prioritize them for sending notification when user create blood request. User will see their name , pre-chosen images and their phone. On the right of their name there will be phone button to call them.

**My Notifications**

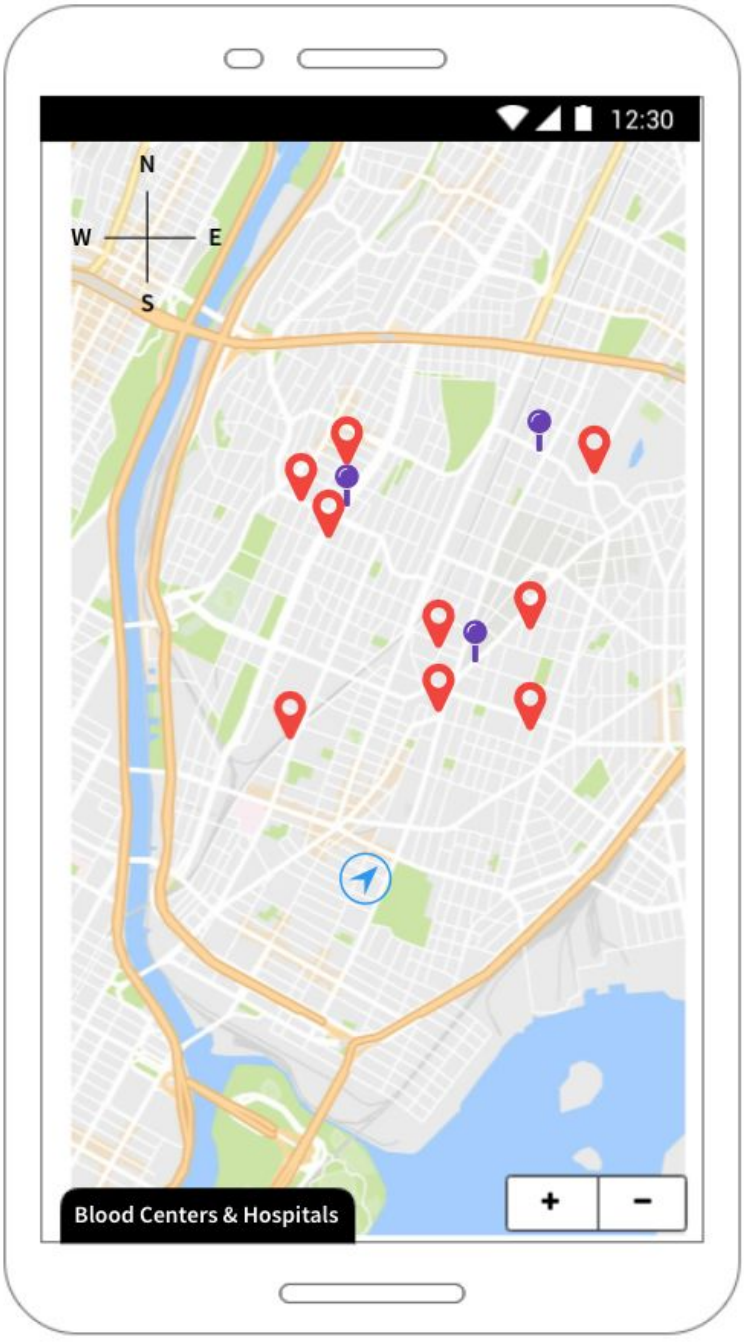



User will see his/her notifications where he/she requested blood and the coordination of that, blood type and additional messages he/she wrote in this notification.

**Received Notifications**


User will see notifications that have been send to them while they were in a range of the notification sender . There will be only the ten recent notifications which mean user will see only recent ten.

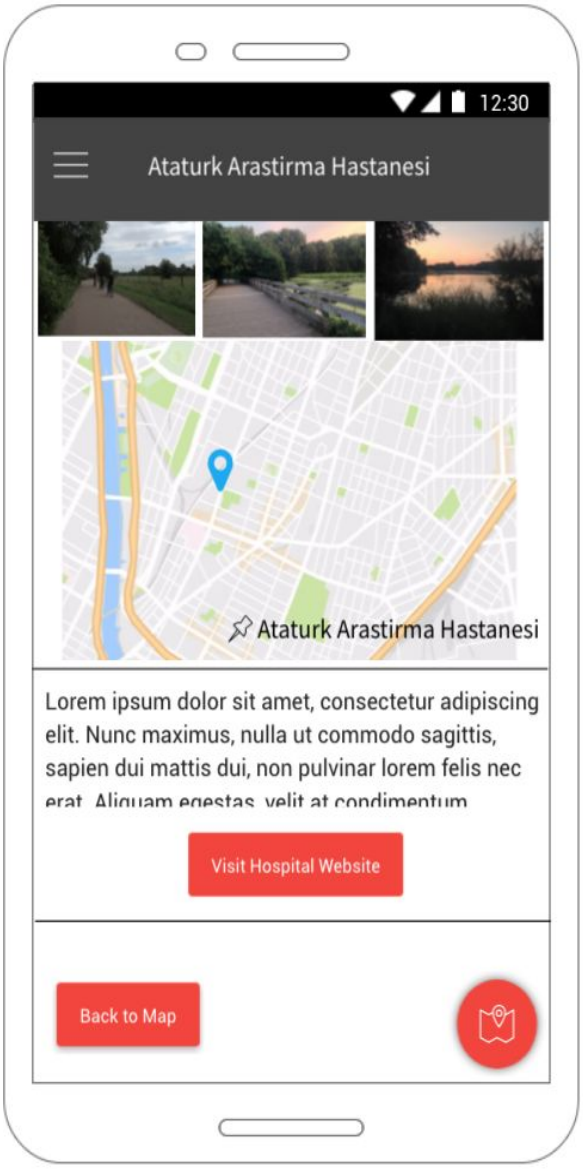
**Map**




User will see the locations of the hospitals, blood centers, Kizilay Blood Events with his/her location. User will see these locations with pre-defined range for example hospitals in 10 km range.

If user wants to learn about something about hospital, blood centers, or events he/she will click and detailed information will be shown to the user.

**Specific Information about Events/Hospitals/Blood Centers**




User will see detailed information about the events/hospitals/blood centers which explained one Interface above. There will be a map which shows respective location of the place, a short information about events/hospitals/blood centers, a button for a website if it has any, back to map button and on the right bottom corner a button which takes user to the selected location, in other words a button to create a navigation to selected location.

**Blog Page**


User will be able to see and read the blog entries that was written by Administration which can include some news, new functionalities or social responsibilities. There will be infinite scroll for user to see all the posts in one interface and if user select one of the post, he/she will see the blog post.

## Contact Page


User will communicate and tell their problems to the Administration via this page. He/she will also need to inform Administration via this page in order to change some mistakes about his/her personal information(blood type etc.).

There are some page in iOS interface and some in Android interface but all the interfaces will be in both Android and iOS. Just for simplicity, in this report some of the user interfaces are made for iOS and some in Android.

# 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].