Bloodonate+

SOFTWARE REQUIREMENTS SPECIFICATION

v1.0

Author: Emirhan Özlen emirhanozlen@gmail.com

I. Revision history

Date	Version	Description	Author
17.08.16 17:08 GMT+03	0.1	Initial commit	Emirhan Özlen
20.08.16 01:23 GMT+03	0.2	Req#12 Socket.io added	Emirhan Özlen
21.08.16 07:17 GMT+03	1.0	Initial release	Emirhan Özlen

1. Introduction

1.1 Purpose

The purpose of this document is to provide the reader with the information of all ideas that have come up to define the system, and all its requirements with respect to all users.

This document is intended for software developers and project managers. End-users should read end-user guide. Software developers would shift their focus on requirements section of this document.

1.2 Scope

The software product named **Bloodonate+** will provide an online bridge between the patients and the volunteered blood donors. It will provide a platform such that patients can navigate through map and find locations of volunteer blood donors and retrieve their contact information.
However, this product is nothing more than a simple bridge between the volunteer and the patient. Product has no meanings of real medical operations or such medical operation provider.

1.3 Definitions

1. **Bloodonate+:** Product name, the online platform of finding blood donors.
2. **API:** Application Programming Interface
3. **MEAN:** MEAN is an acronym for MongoDB, ExpressJS, AngularJS and Node.js
4. **MongoDB:** MongoDB is an open source, document-oriented database designed with both scalability and developer agility in mind. Instead of storing your data in tables and rows as you would with a relational database, in MongoDB you store JSON-like documents with dynamic schemas
5. **Express.Js:** Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.
6. **Angular.Js:** AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly
7. **Node.js:** is an open source, cross-platform runtime environment for developing server-side and networking applications.
8. **Socket.io:** is a JavaScript library for realtime web applications. It enables real time, bidirectional communication between web clients and servers.
9. **ArcGIS:** is a geographic information system (GIS) for working with maps and geographic information.

10. **CROSSOVER:** Founded in 2014 by Andy Tryba, Crossover company provides engineering teams of exceptional quality via our platform. Crossover Teams, comprising the best engineers in the world, take on client projects in engineering, IT support, and sales.
11. **HTTP:** Hypertext Transfer Protocol.
12. **CRUD:** Acronym stands for Create, read, update and delete.
13. **SRS:** Software Requirements Specification
14. **RESTFul:** Representational state transfer (REST), is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

1.4 Overview

First section of document provides an overview of the entire SRS, purpose, what the software product will and will not do; definitions and acronyms.

Second section of document describes general factors that affect the product and its requirements though does not state specific requirements.

The third section contain all of the software features and its requirements to great level of detail sufficient to allow designers and developers to design the system to meet those requirements and testers to test the system,

2. Overall description

2.1 Product perspective

This product is independent, not a component of larger existing system. Any server operates on Windows, Mac OS X, most (if not all) Linux distribution can run Node.Js

2.1.1 User interfaces

Twitter Bootstrap will be used on general layout of the document.

Users will see a index page with a map containing pin points. Clicking or tapping on pin points will show the contact information of who left it there, on a form. They will interact with map, create pin points and leave their contact information in order to find patient to donate blood through a form. Editing or deleting pinpoints will be done on a private page access provided by a unique link.

Please see the **Appendix:** 4.1 User Interfaces section to see more about user interfaces.

2.1.2 Software interfaces

The software products and interfaces chosen here are follow up of requirements provided by **CROSSOVER**

2.1.1.1 Node.js

Source: https://nodejs.org/en/

2.1.1.2 MongoDb

Source: https://www.mongodb.com/

2.1.1.3 Angular.Js

Source: https://angularjs.org/

2.1.1.4 Express.Js

Source: https://expressjs.com/

2.1.1.5 Socket.io

Source: http://socket.io/

2.1.1.6 ArcGIS

Source: https://www.arcgis.com/

2.1.1.7 Twitter Bootstrap

Source: http://getbootstrap.com/

2.1.1.8 MEAN Stack

Source: http://mean.io/

2.1.3 Communication interfaces

HTTP used for initialization of everything, loading html documents and all other resources from server.

After that, socket.io API tries to handshake with server in order to establish **upgrade on protocol**. If server supports the protocol it returns client HTTP/1.1 101 Switching Protocols response header to client and upgrades connection to **WebSocket protocol**.

2.1.4 Hardware Interfaces

There is no hardware interface implementation needed for this software. Any device that has an internet connection and able to browse through web pages can access the system.

2.2 Product functions and user characteristics

Major functions provided by **Bloodonate**+ from user perspective are listed below as:

From donor point of view:

1. Find their location on map.
2. Leave the contact information and retrieve a link to whether update or delete whenever they want

From patent point of view:

1. Navigate through map and find donors' information based on location

2.3 Assumptions

There will be no login system:

1. Therefore, any guest can easily send data to be saved on database
2. Therefore, any guest can exploit bandwidth of the server. There will be no bandwidth limit scenarios.

2.4 Apportioning of requirements

Future versions of this product might have the features listed below:

1. Login module
2. Limiting user bandwidth
3. Captcha challenge

2.5 Operating Environment

The node.js framework itself doesn't need much RAM or CPU power per instance due to its evented rather than threaded or process based architecture. NodeJS can run on just about any operating system, Windows, Mac OSX, most (if not all) Linux distributions.

NodeJS run comfortably on system with 256MB and even 128MB ram, with as little as 1 core.

3. Specific requirements

3.1 Software product features

3.2 Functional, **3.3 Other requirements** and **4.1 User Interface** requirements are labelled below in table at second column as REQ#1, OTH#2, … UI#N are explained in great detail in their own chapters.

3.1#1 Find location on map

Donors can find their location on map

Requirements:

ArcGIS Map shown on html document	**REQ#1** **UI#1**
Form to process location text in order to **navigate** through map	**UI#2** **REQ#2**

3.1#2 Submit donor to server through form

Upon clicking on a certain point on map, a form should open and donor can add the contact information.

Requirements:

Capture click on map	**REQ#3**
Update the html document **dynamically** with a **form** in order to enter contact information	**REQ#4** **UI#3**
Input validation	**Feature#3**

3.1#3 Validate form before processing

Contact information should be validated on both client and server side. If input fields contains invalid information, a warning message will be prompted.

Requirements:

Client & Server side validation	**REQ#5**
Data structure of donor's contact information and **validation format**	**OTH#1** **OTH#2**
Update the html document dynamically with a message in order to warn user about invalid input	**REQ#4**

3.1#4 Show status message on form submit

On submitting the form, a status message should be shown to user along with donor information plus ip and geographical coordinates saved in database

Requirements:

Status messages	**OTH#3**

Update **dynamically** the html document with **dialog** indicates the status of operation and if success, along with ip, geographical coordinates and **donor information saved on database**	**REQ#4** **UI#4** **OTH#1** **REQ#6**
Will be shown along with private link	**Feature#4**

3.1#5 Generate a private link

On submitting the form, a unique private link should be generated and displayed to donor, from where donor can edit or delete his information

Requirements:

A unique private donor **page link** will be **generated** Where he/she can **edit** or **delete** his/her information	**UI#5** **REQ#7** **REQ#6**

3.1#6 Lazy load pinpoints through navigation

Pin points should be lazy loaded, so only the pins that belong to the visible area of map should be loaded.

Requirements:

Lazy load	**REQ#8**
Update pin points	**REQ#9**

3.1#7 Show donor information on pinpoint click

On clicking any pin, a popup should appear displaying the donor’s information.

Requirements:

Retrieve donor information from database upon **clicking** any pin on map after bot challenge completed	**REQ#3** **Feature#8**
Pinpoints needed	**REQ#9**

3.1#8 Email and phone number will be secured against bots

In place of email and phone number, there should be a text (click to show) to avoid bots from reading donor’s information

Requirements:

Secured donor information dialog	**UI#6**

Safely deliver donor information after bot challenge	**REQ#6**

3.1#9 Pin changes will be visible

If any pin changes (a user made change to his/her post or deleted it), the change should be visible real time to other users.

Requirements:

Update pinpoints on map	**REQ#9**
Listen to server messages to detect changes on pin points	**REQ#10**

3.2 Functional requirements

All the functional requirements listed had been beginning with words “System shall” or “System shall be” that made titles appear to wide in document. Shall statements are removed in order to simplify titles.

3.2#1 Show ArcGIS Map on HTML document

ArcGIS API should be implemented in order to show in HTML document.

3.2#2 Map navigation

There should be a module to search locations based on user’s input, and center the map if input points to somewhere valid. Requires number #1 functionality.

3.2#3 Capture click on ArcGIS map

There should be a module that is able to capture the click and tap events on ArcGIS Map and should trigger click event. This functionality is dependent on function number #1

3.2#4 Update the document dynamically

The document shall be dynamic, Angular.Js will be used to accomplish such tasks.

3.2#5 Server and client side validation

System shall check every available input from user first on client side with Angular.Js then also for security, also shall check on the server side.

3.2#6 CRUD donor information

System shall CRUD valid donor information supplied from number #5 functionality. CRUD operations will done RESTFul the number #11 functionality.

3.2#7 Unique link generation

System should generate unique links to provide a platform that donors can CRUD their data.

3.2#8 Lazy load pinpoints from server

According to zoom level of map and center point supplied from patient, system shall sent the pinpoints available in the area and patient will be able to lazy load the data from server.

3.2#9 Update pinpoints on map

There should be a module that is responsible for caching the pinpoints and updating them on the map. Number #1 ArcGIS Map is required.

3.2#10 Server shall broadcast pinpoint changes

Server will broadcast pinpoint changes to connected clients. This functionality is dependent on number #12 Socket.io.

3.2#11 CRUD operations with RESTFul API

Server will process operations related to CRUD data on database with REST API.

3.2#12 Use sockets with Socket.io

Server will broadcast lazy loading and pinpoint changes in real time.

3.3 Other requirements

3.3#1 Data Structure of donor information

First name, Last name, Contact number, Email address, Blood group, IP Address, Geographical coordinates. Everything will be text as we will be saving our data on NoSQL environment

3.3#2 Validation format of donor information

First name, Last Name: Text
Phone: +xx xxx xxxx xxx or 00xx xxx xxxx xxx
Email address: example.subname@domain
Blood group: ^([ABO]|(AB))[\+\-]$
Geographical coordinates: Double

3.3#3 Status messages of validation

Decimal	Hex	Binary	Message
0	0x0	0000 0000	Success
1	0x1	0000 0001	Invalid first name
2	0x2	0000 0010	Invalid last name
4	0x4	0000 0100	Invalid contact number
8	0x8	0000 1000	Invalid email address
16	0x10	0001 0000	Invalid blood group
32	0x20	0010 0000	Invalid IP Address
64	0x40	0100 0000	Invalid geographical latitude
128	0x80	1000 0000	Invalid geographical longitude

Server can simply do OR operation on batch of message codes to indicate client has sent more than one invalid field. Couple of examples are shown below:

Decimal	Hex	Binary	Message
3	0x03	0000 0011	Invalid first and last name
42	0x2A	0010 1010	Invalid Last name, email and IP Address
255	0xFF	1111 1111	Everything is invalid

3.3#4 Socket.io Client-Server packet exchange definitions

Client-side definitions

Message		Parameters		Definition
Hex	ASCII			
0x30	0			Invalid request
0x31	1	Double[2] {x,y}	String _id	Pinpoint update
0x32	2	Array of Double[2] {x,y}		Response to request lazy load (x: lon, y; lat)
0x33	3	String _id		Pinpoint delete

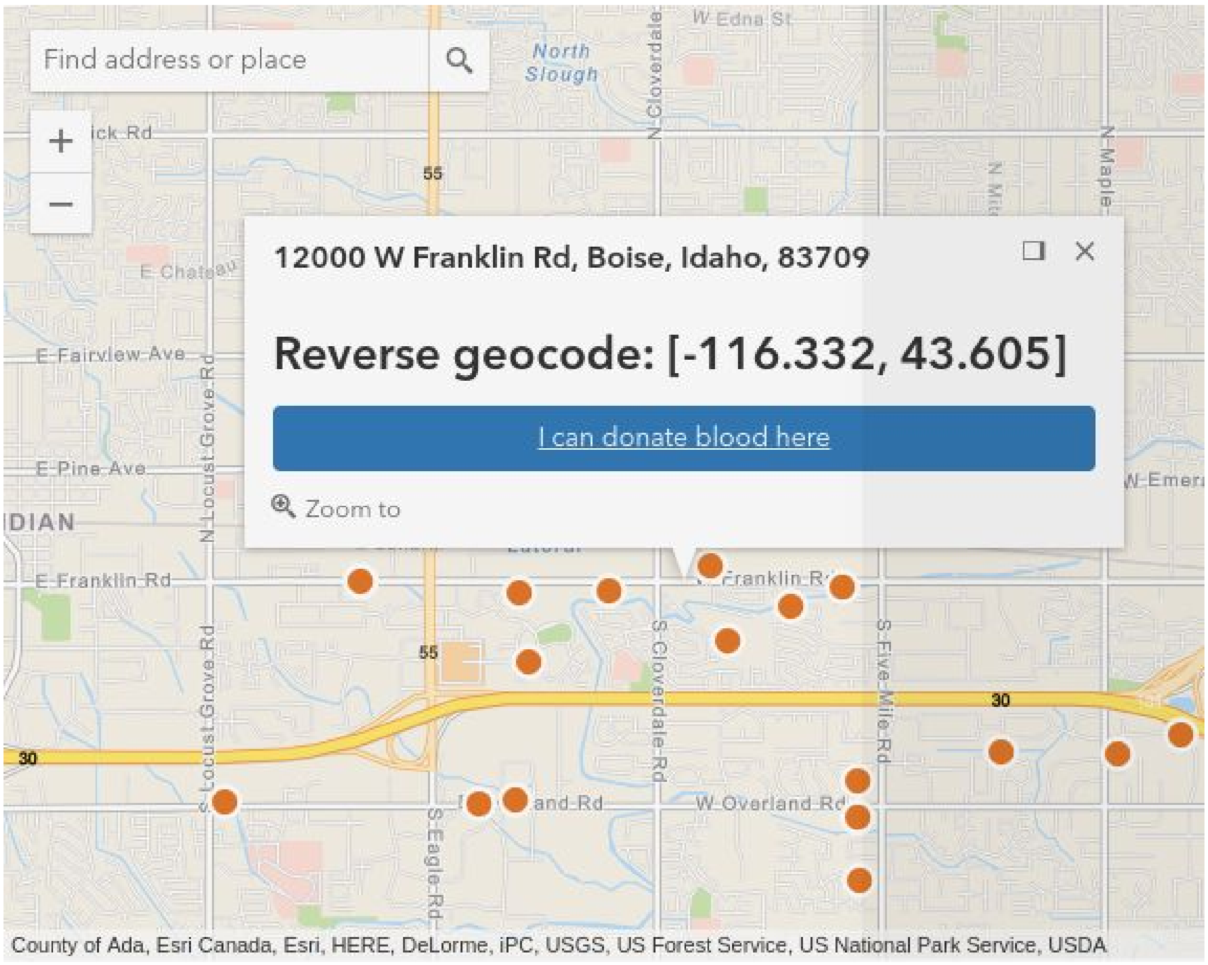
Server-side definitions

Message		Parameters		Definition
Hex	ASCII			
0x32	2	Double x1 Double x2	Double y1 Double y2	Client requesting the points in the rectangular area

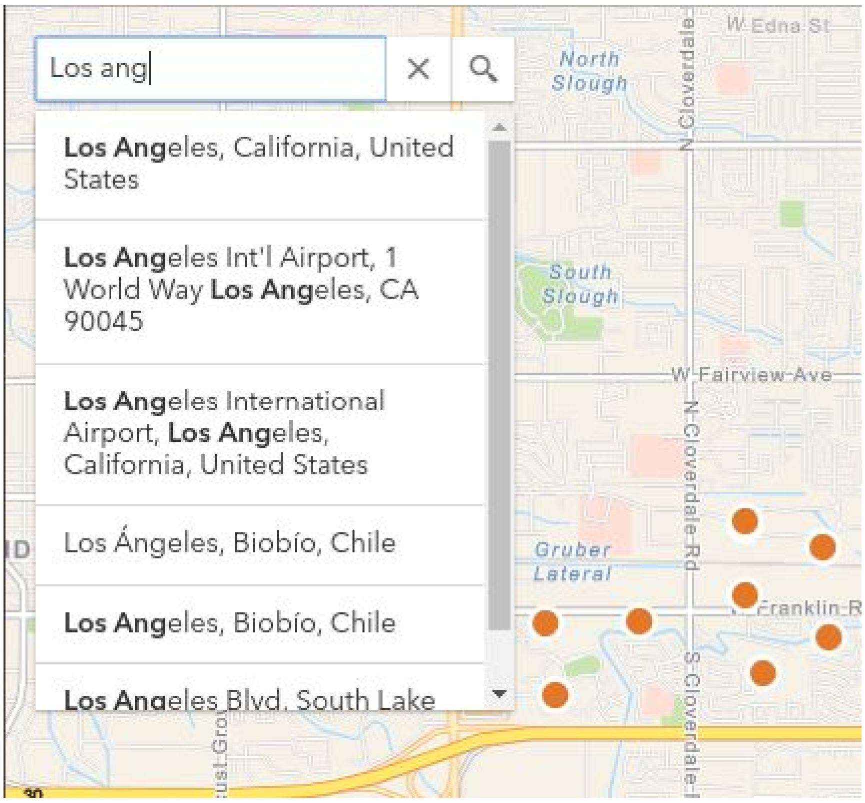
4. Appendix

4.1 User interfaces

4.1#1 ArcGIS Map layout



4.1#2 Find location form



Los ang

Los Angeles, California, United States

Los Angeles Int'l Airport, 1 World Way **Los Ang**eles, CA 90045

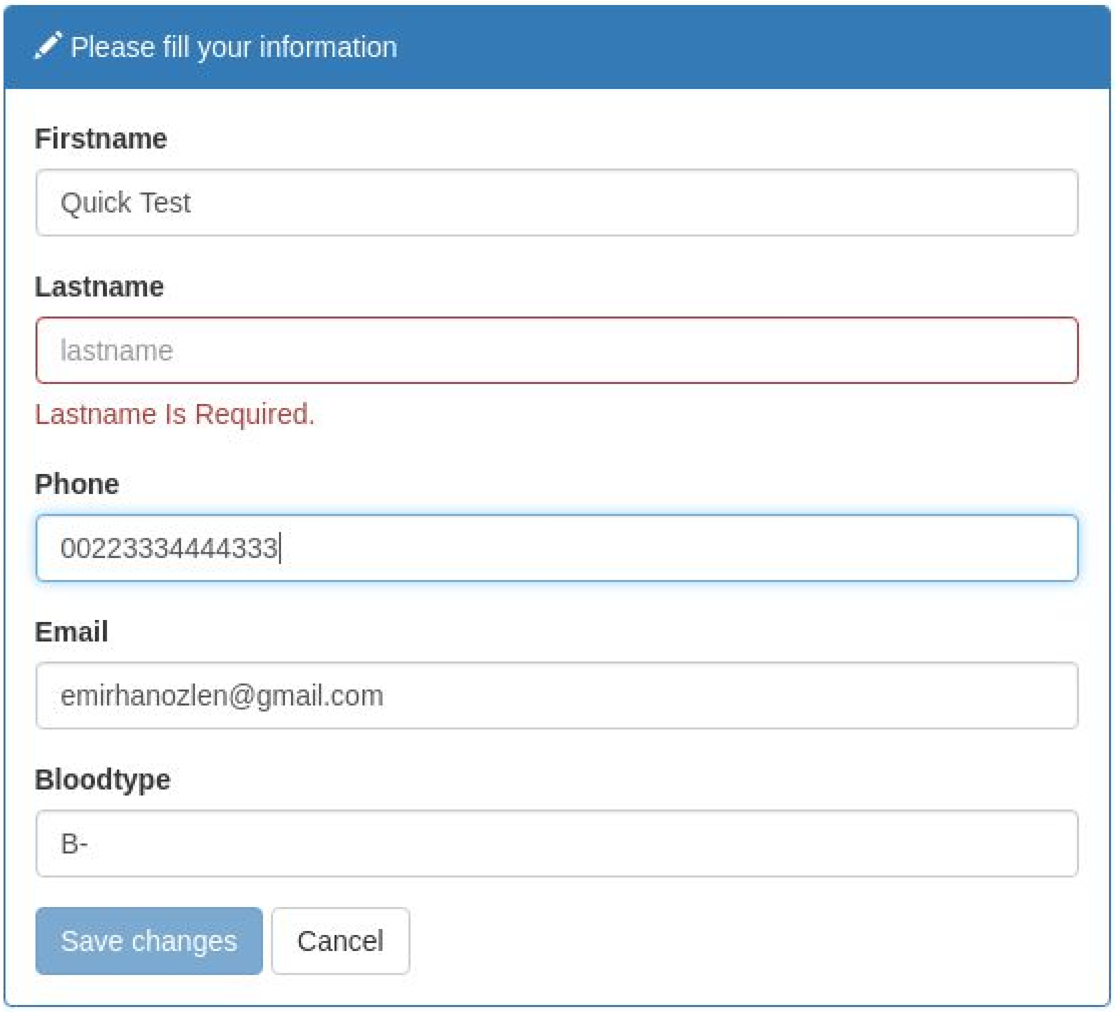
Los Angeles International Airport, **Los Ang**eles, California, United States

Los Ángeles, Biobío, Chile

Los Angeles, Biobío, Chile

Los Angeles Blvd, South Lake

4.1#3 Donor contact information form


Please fill your information

Firstname

Quick Test

Lastname

lastname

Lastname Is Required.

Phone

00223334444333

Email

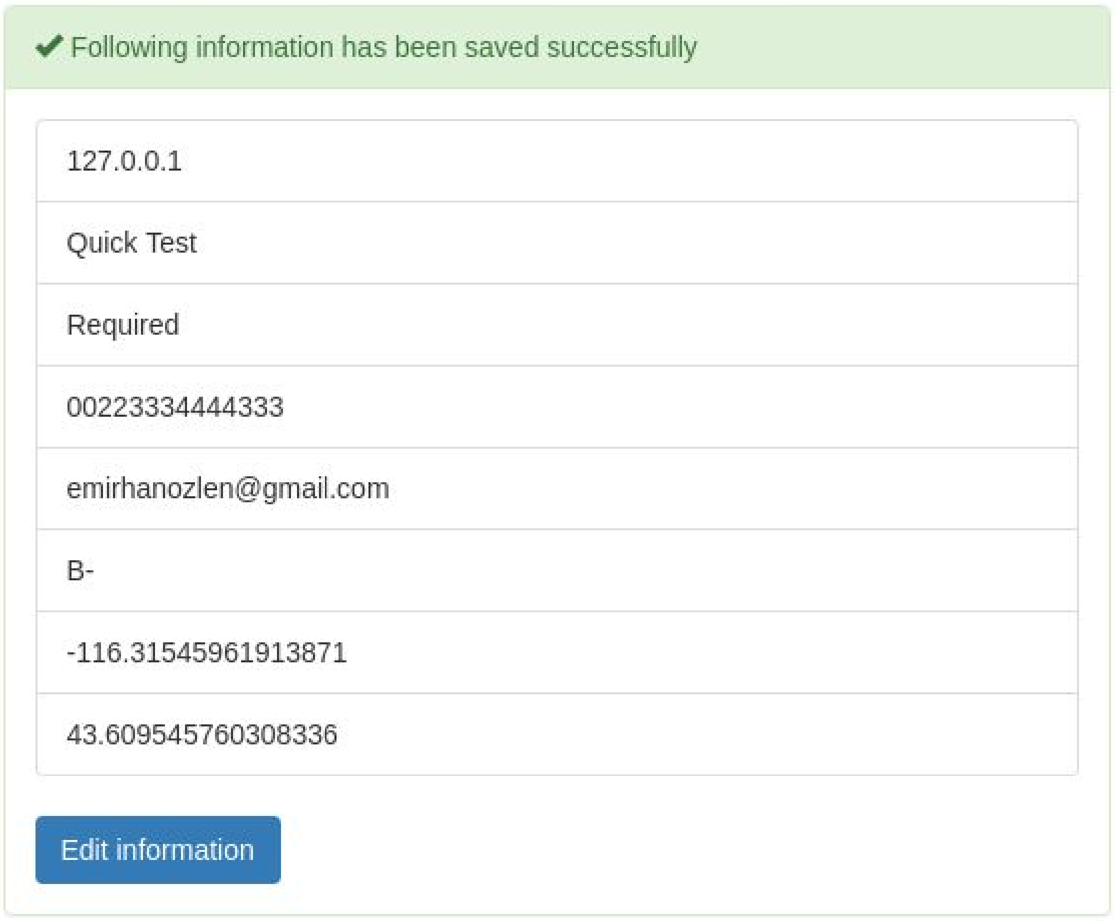
emirhanozlen@gmail.com

Bloodtype

B-

Save changes Cancel

4.1#4 Status message dialog


✓ Following information has been saved successfully

127.0.0.1

Quick Test

Required

00223334444333

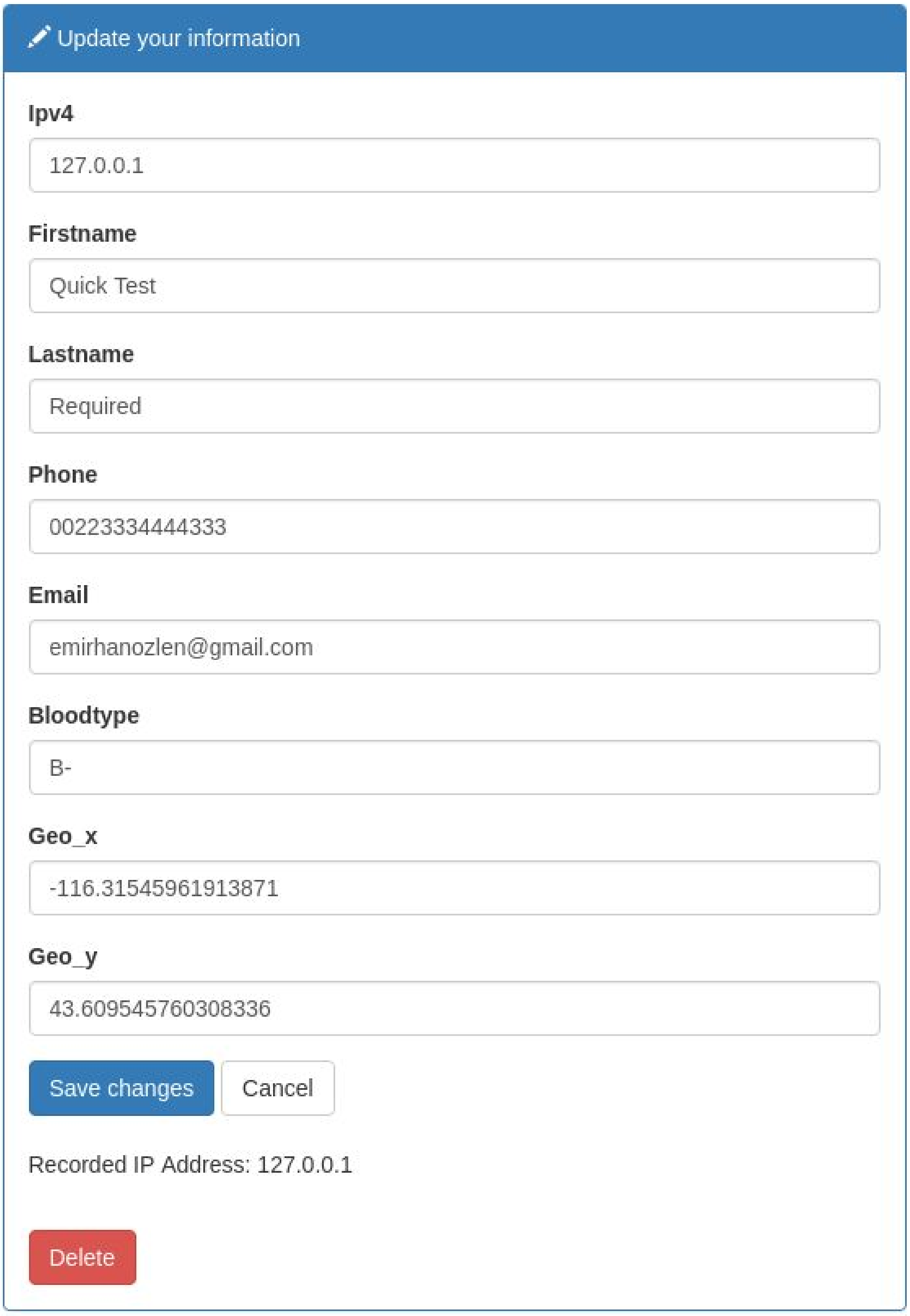
emirhanozlen@gmail.com

B-

-116.31545961913871

43.609545760308336

4.1#5 Private donor page

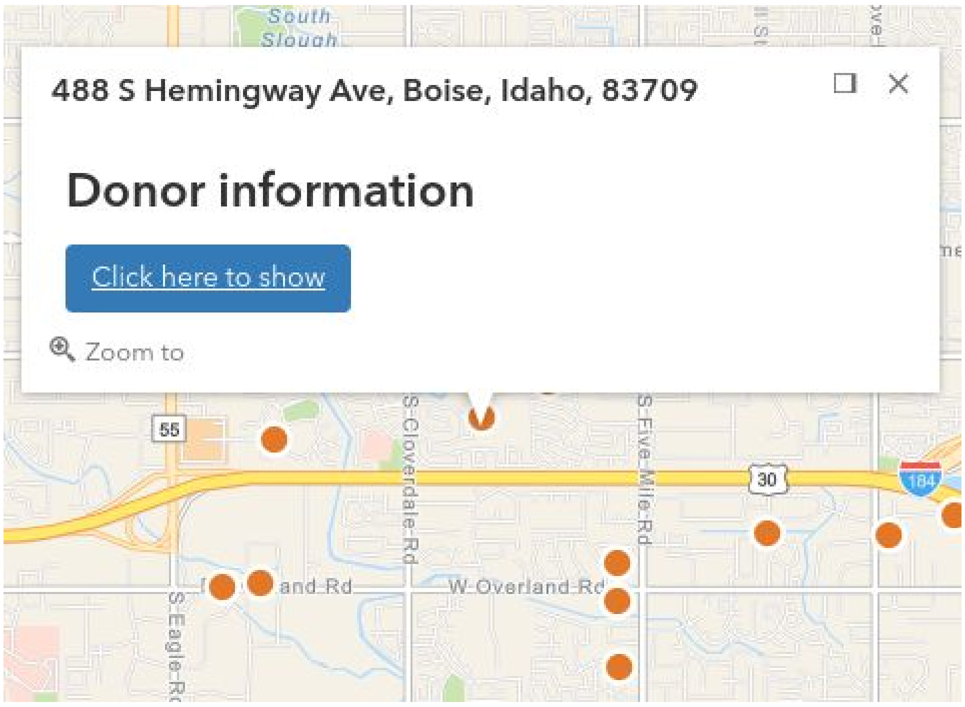


Update your information

Ipv4	127.0.0.1
Firstname	Quick Test
Lastname	Required
Phone	00223334444333
Email	emirhanozlen@gmail.com
Bloodtype	B-
Geo_x	-116.31545961913871
Geo_y	43.609545760308336

Recorded IP Address: 127.0.0.1

4.1#6 Donor information dialog secured from bots



488 S Hemingway Ave, Boise, Idaho, 83709

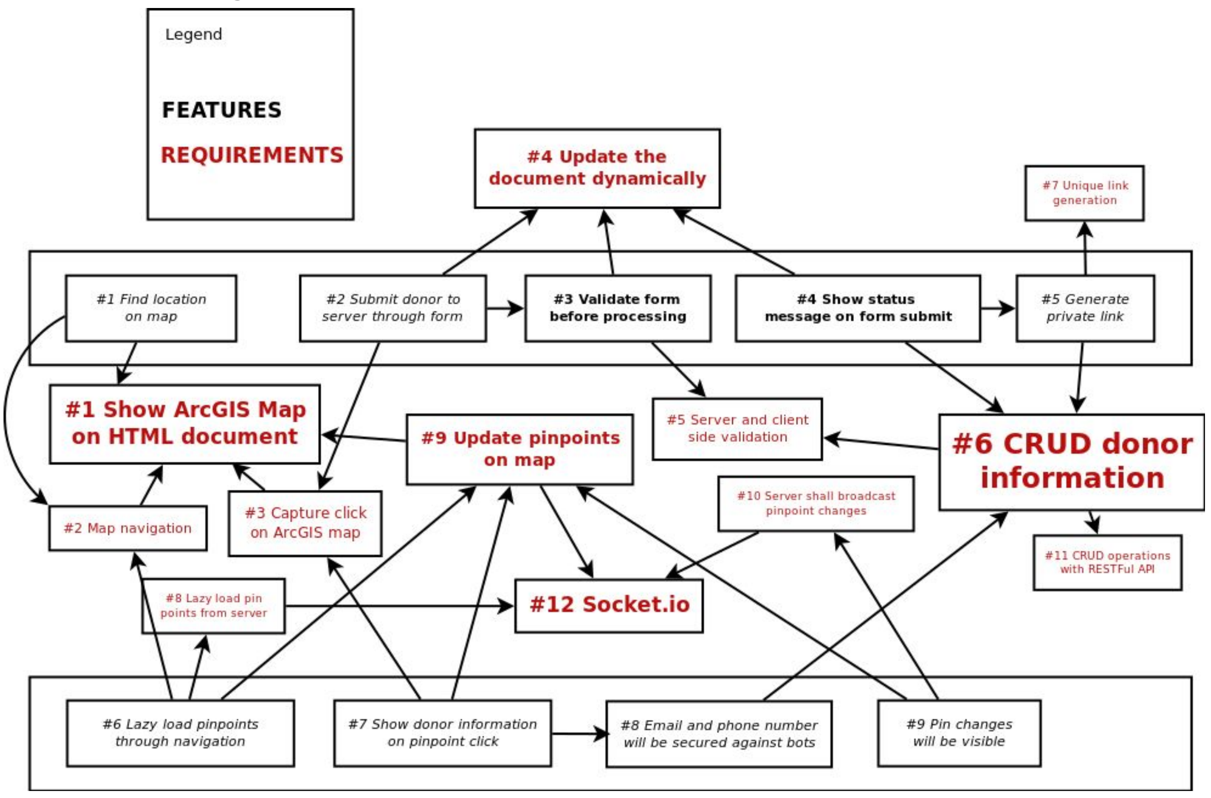
Donor information

Click here to show

Zoom to

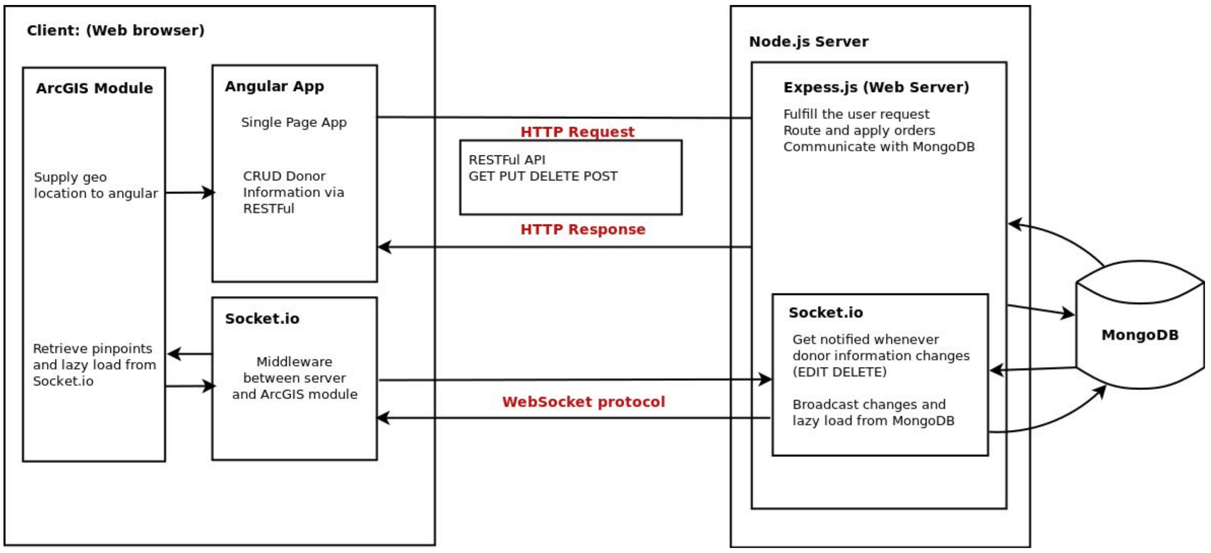
4.2 Features and requirements interaction diagram

You can find full sized image at the same directory as this document named **Features requirements.png**


Legend
FEATURES
REQUIREMENTS
#4 Update the document dynamically
#7 Unique link generation
#1 Find location on map
#2 Submit donor to server through form
#3 Validate form before processing
#4 Show status message on form submit
#5 Generate private link
#1 Show ArcGIS Map on HTML document
#9 Update pinpoints on map
#5 Server and client side validation
#6 CRUD donor information
#2 Map navigation
#3 Capture click on ArcGIS map
#10 Server shall broadcast pinpoint changes
#11 CRUD operations with RESTFul API
#8 Lazy load pin points from server
#12 Socket.io
#6 Lazy load pinpoints through navigation
#7 Show donor information on pinpoint click
#8 Email and phone number will be secured against bots
#9 Pin changes will be visible

4.3 Component Integration Diagram

You can find full sized image at the same directory as this document named **Component Integration diagram.png**


Client: (Web browser)
ArcGIS Module
Supply geo location to angular
Retrieve pinpoints and lazy load from Socket.io
Angular App
Single Page App
CRUD Donor Information via RESTFul
Socket.io
Middleware between server and ArcGIS module
HTTP Request
RESTFul API GET PUT DELETE POST
HTTP Response
WebSocket protocol
Node.js Server
Expess.js (Web Server)
Fulfill the user request
Route and apply orders
Communicate with MongoDB
Socket.io
Get notified whenever donor information changes (EDIT DELETE)
Broadcast changes and lazy load from MongoDB
MongoDB