Palestine Polytechnic University

College of Engineering
Department of Electric Engineering

Project Title:

Blood Bank System

Prepared By

Ibrahim Jundi

Mohammad Talahmeh

Hadeel Ayaydeh

Supervisor

Dr. Murad Abusubaih

Submitted to the College of Engineering in fulfillment of the requirement for the bachelor degree in communication and Electronic Engineering

Hebron –Palestine

2017/2018

Palestine Polytechnic University

College of Engineering

Department of Electric Engineering

Project Title:

Blood Bank System

Prepared By

Ibrahim Jundi

Mohammad Talahmeh

Hadeel Ayaydeh

Supervisor

Dr. Murad Abusubaih

Hebron –Palestine

2017/2018

جامعة بوليتكنك فلسطين

الخليل – فلسطين

كلية الهندسة

دائرة الهندسة الكهربائية

اسم المشروع

Blood Bank System

اسماء الطلاب

ابراهيم جندي

محمد تلاحمه هديل عيايدة

بناء على نظام كلية الهندسة وإشراف ومتابعة المشرف المباشر على المشروع وموافقة اعضاء اللجنة الممتحنة , تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية وذلك للوفاء بمتطلبات درجة البكالوريوس في تخصص هندسة الاتصالات والالكترونيات.

توقيع المشرف

.....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

.....

Contents

Figures

Table

Dedication

To my dear teacher

Dr. Murad Abusubaih

Acknowledgment

Thanks to God first, to give us the strength to accomplish this project. Secondly, we thank our supervisor Dr. Murad Abusubaih for his help and care. Third, we thank our beautiful university. Fourthly, we thank our teachers. Fifthly, and we thank everyone who helped us

Abstract

Hospitals in Palestine are still working independently. They are not interconnected. Due to the shortage of resources in Palestinian hospitals, there is an urgent need to connect them together. This would allow the sharing of different recourses as well as different types of treatments and equipment that do not exist in every hospital.

In this project, the beginning of an important step towards connecting hospitals. Specifically, a system was designed to allow hospitals in Palestine to share information on the availability and types of blood units. This would allow hospitals to cooperate and save the lives of a large number of patients.

Hospitals will be linked to two types of communication technology or Internet services. The first type is through the local network available in each hospital (Ethernet), and the other is the cellular network technology currently available in Palestine (3G) through the use of the GSM-Modem device, which will connect the two technologies and this makes the possibility of communication permanent, When any of the two are disconnected, the system remains connected to the server, achieving a fairly high speed.

الملخص

لا تزال المستشفيات في فلسطين تعمل بشكل مستقل وليست مترابطة. نظرا لنقص الموارد والإمكانيات في المستشفيات الفلسطينية، هناك حاجة ملحة لربطها مع بعضها البعض. وهذا من شأنه أن يسمح بمشاركة الموارد المختلفة وكذلك أنواع مختلفة من المعدات والأدوية وغيرها، التي لا وجود لها في كل مستشفى.

في هذا المشروع، بداية خطوة هامة نحو ربط المستشفيات. وعلى وجه التحديد، تصميم نظام يسمح للمستشفيات في فلسطين بمشاركة المعلومات حول توافر وحدات الدم وأنواعها. وهذا من شأنه أن يسمح للمستشفيات بالتعاون وإنقاذ حياة عدد كبير من المرضى.

وسيتم ربط المستشفيات بنوعين من تقنيات الاتصال أو من خدمات الإنترنت. النوع الأول من خلال الشبكة المحلية المتاحة في كل مستشفى (Ethernet)، والنوع الآخر عبارة عن تقنية الشبكات الخليوية المتوافرة حاليا في فلسطين منها (3G) من خلال استخدام جهاز (GSM-Modem)، حيث سيتم توصيل التقنيتان وهذا يجعل احتمالية الاتصال دائمة، في حين انقطاع أي من هما يبقى النظام متصل بالخادم، ويحقق سرعة عالية نوعا ما.

Abbreviations

3G	**Third Generation**
HTTP	**Hyper Text Transport Protocol**
HTML	**Hyper Text Markup Language**
CSS	**Cascading Style Sheets**
PHP	**Personal Home Pages**
NET	**Network**
MySQL	**My Structured Query Language**
PC	**Personal Computer**
XHTML	**Extensible Hyper Text Markup Language**
XML	**Extensible Markup Language**
XUL	**XML-based User interface Language**
SVG	**Software Vulnerability Guide**
ECMA	**European Computer Manufacturers Association**
VLAN	**Virtual Local Area Network**
CSMA/CD	**Carrier Sense Multiple Access with Collection Detection**
IEEE	**Institute of Electrical and Electronics Engineers**
GSM	**Global System for Mobile communications**
SIM	**Subscriber Identity Module**
SMS	**Short Message Service**
MMS	**Multimedia Messaging Service**
ISDN	**Integrated Services Digital Network**
PSTN	**Public Switching Telecommunication Network**
CPU	**Central Processing Unit**
API	**Application Programming Interface**
MVC	**Model View Controller**
NPM	**Node Package Manager**
AJAX	**Asynchronous JavaScript And XML**

1

Chapter One Introduction

1.1 Motivation

1.2 Related Work

1.3 Need of the System

1.4 System Description and Solution Idea

1.5 Objectives

1.6 Expected Results

1.7 Cost

1.1 Motivation

It is known that the availability of blood units in any hospital is extremely important in order to complete the health services and care. Palestine is a special case. The low number of hospitals شي the large number of population coupled with the political situation have put a challenge for the health care. In many cases, we either hear or read announcement from different hospitals about the need of blood units.

It is not possible for the health care staff to determine where to get blood units even from neighboring other hospitals.

In this project, we plan to develop, design, and implement a system that allows hospitals to share, exchange information, and cooperate in providing better healthcare to patients. The systems allows interconnecting hospitals and provides access to a database about the available blood units in each hospital.

In the case of expected shortage of blood units, announcement for donations will be activated.

1.2 Related Work

Blood bank is responsible for the storage, processing and collection of blood. Blood plays important role in blood bank, as it is the necessity of everyone. Many researchers worked on the development of blood bank management system[1]. Some of them are given below:

1. Virtual Blood Bank Project This system implemented by using java and web applications. This allows us to find donors from their respective address, which are collected from hospital database.

2. Location based blood bank system based on cloud storage this type of system is based on mobile app which is linked to cloud server. Donor registration details and other details will be stored in the cloud. In case of an emergency, anybody can use this app to locate the donor.

3. Emergency Blood Bank directories using www.bloodbanker.com: In this type of system, www.bloodbanker.com website holds the details of hospitals and blood banks in USA. Website can be used to find the nearest blood donor.

1.3 Need of the System

The project idea has been suggested and discussed with hospitals in the city of Hebron. Specifically we have visited:

1. Alia Hospital.
2. Al-Ahli Hospital.
3. Palestine Red Crescent Society Specialized Hospital.

The technicians in all hospitals have pointed out that such system is not available yet. Further, they expect to adopt it if designed and implemented as described to them.

They explained that the availability of such system will help to safety the life of many people and will advance the healthcare services provided to Palestinians, especially due to the risk environment in Palestine.

1.4 System Description and Solution Idea

It is planned to design and implement a system that allows healthcare staff to be able to search the availability of blood units in other hospitals. There will be a central database that stores all system data. Software will be used in each hospital to access, search and edit the database. The system will interconnect hospitals over the typical internet access lines as well as 3G backup connection in urgent cases and upon failure of the Ethernet connection. The block diagram of the system as shown in figure 1.1.

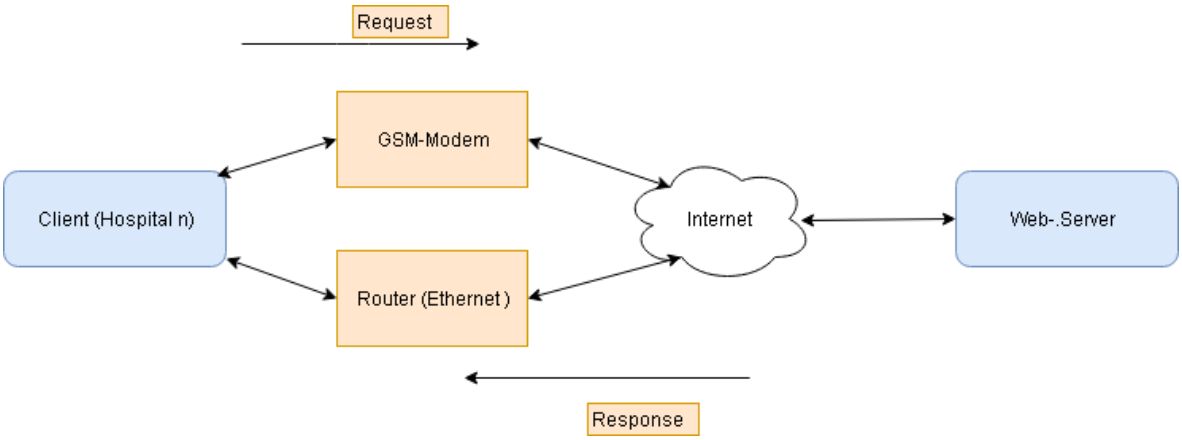


Figure 1.1: System Block Diagram

When a sick person needs a blood unit in a hospital and this unit is not available, the staff will search for this unit in the system.

1.5 Objectives

The goal of the project is to quickly and easily determine the quantity and quality of blood in several hospitals connected together.

The main objectives of this project are summarized as follows:

1) Connect Hospitals using Internet and 3G.
2) Allow search of blood units in the system and their availability.
3) Allow cooperation between hospitals.
4) Facilitate announcements for donations of blood units.

1.6 Expected Results

We expect to deliver a prototype of the system as described above. We will use it to connect hospitals in Hebron city.

1.7 Cost

Table 1-1 shows the cost of implementing the prototype.

Tools	**Number of Tools**	**Cost of one tool**	**Cost of the Tools**
Computers	2	400$	800$
Ethernet Routers	2	35$	70$
GSM-Modem	2	30$	60$
Internet Service	2	40$	80$
Total Cost			**2020$**

Table 1.1 :Cost of Implementing System Prototype

2

Chapter Two Background

2.1 Introduction

2.2 Network

2.3 Network of the System

2.4 Network side

2.5 Client side

2.6 Server side

2.7 The software used in the system

2.1 Introduction

This chapter provides a short description of the system components as well as theoretical background about the topic in general.

2.2 Network

Is a collection of computers, servers, mainframes, network devices, peripherals, or other devices connected to one another to allow the sharing of data. An excellent example of a network is the Internet, which connects millions of people all over the world. Below is an example figure 2.1 of a network with multiple computers and other network devices all connected to each other by the Internet[2].

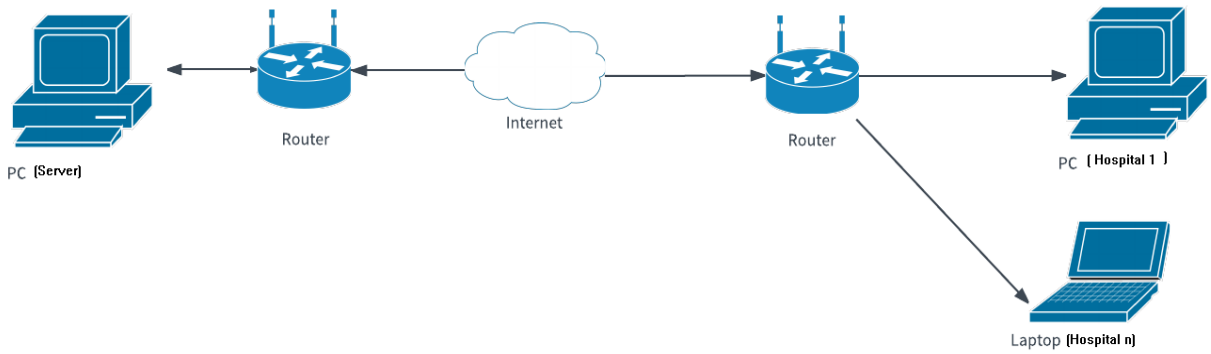


Figure 2.1: Network

2.3 Network of the System

This system is a network connecting computers with the server through the Internet, so each hospital can see the blood units of other hospitals on the same server, each hospital connects to the Internet by connecting computers to two networks, the local network and cellular network is a subscription to the data service Cell.

2.4 Network side

The network in this system is connecting the hospitals to each other via the internet using two network technologies. The first network is the local network of each hospital distributed by the distribution companies. The other network is the cellular data service that is connected via GSM modem figure 2.2 shows this system.

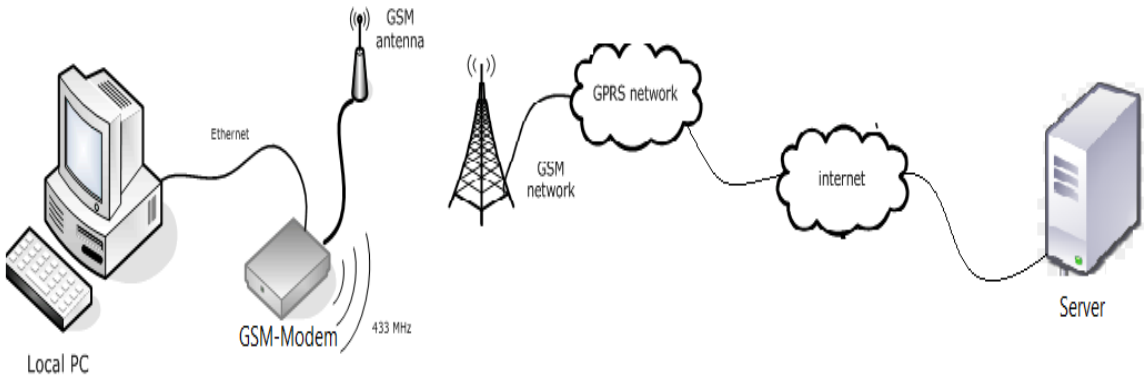


Figure 2.2: Network of the system prototype

2.4.1 Router

Small electronic devices that joins multiple computer networks together via either wired or wireless connections.

Is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet. A data packet is typically forwarded from one router to another router through the networks that constitute an internetwork until it reaches its destination node, figure 2.3; shows the one of the routers.

Figure 2.3: Router

2.4.2 GSM-Modem

Is a specialized type of modem, which accepts a SIM card, and operates over a subscription a mobile operator, just like a mobile phone. From the mobile operator perspective, a GSM modem looks just like a mobile phone.

When a GSM modem is connected to a computer, this allows the computer to use the GSM modem to communicate over the mobile network. While these GSM modems are most frequently used to provide mobile internet connectivity, many of them can also be used for sending and receiving SMS and MMS messages[3]. Figure 2.4; show the structural of the GSM-modem.

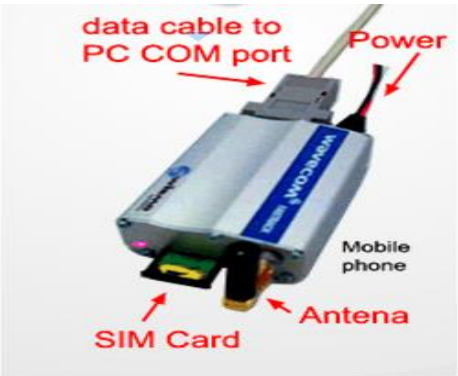


Figure 2.4: GSM Modem

- A GSM network consists of the following components:

- **A Mobile Station**

  It is the mobile phone, which consists of the transceiver, the display and the processor and is controlled by a SIM card operating over the network.

- **Base Station Subsystem**

  It acts as an interface between the mobile station and the network subsystem. It consists of the Base Transceiver Station, which contains the radio transceivers and handles the protocols for communication with mobiles. It also consists of the Base Station Controller that controls the Base Transceiver station and acts as an interface between the mobile station and mobile switching center.

- **Network Subsystem**

  It provides the basic network connection to the mobile stations. The basic part of the Network Subsystem is the Mobile Service Switching Centre, which provides access to different networks like ISDN, PSTN etc. It also consists of the Home Location Register and the Visitor Location Register, which provides the call routing, and roaming capabilities of GSM. It also contains the Equipment Identity Register that maintains an account of all the mobile equipment is wherein its own IMEI number identifies each mobile. IMEI stands for International Mobile Equipment Identity.

- **SIM Card**

  A SIM card, also known as a subscriber identity module, is a smart card that stores data for GSM cellular telephone subscribers. Such data includes user identity, location and phone number, network authorization data, personal security keys, contact lists and stored text messages. Security features include authentication and encryption to protect data and prevent eavesdropping.

2.4.3 Ethernet

Is the most widely installed local area network (LAN) technology. Ethernet is a link layer protocol in the TCP/IP stack, describing how networked devices can format data for transmission to other network devices on the same network segment, and how to put that data out on the network connection. It touches both Layer 1 (the physical layer) and Layer 2 (the data link layer) on the OSI network protocol model. Ethernet defines two units of transmission, packet and frame. The frame includes not just the "payload" of data being transmitted but also addressing information identifying the physical "Media Access Control" (MAC) addresses of both sender and receiver, VLAN tagging and quality of service information, and error-correction information to detect problems in transmission. Each frame is wrapped in a packet, which affixes several bytes of information used in establishing the connection and marking where the frame starts[4]. Figure 2.5, explain Ethernet structure.

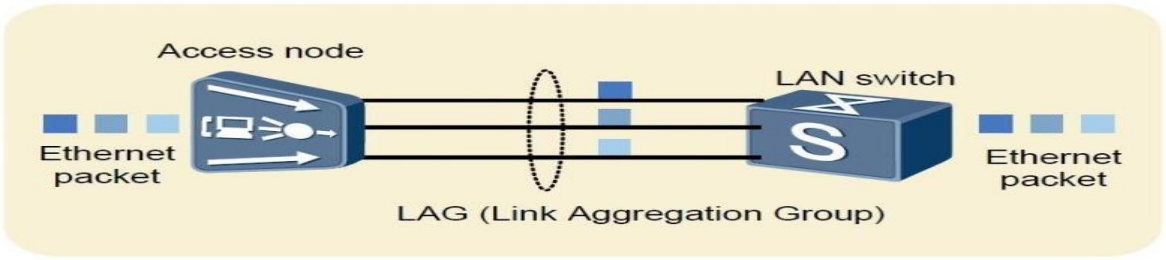


Figure 2.5: Ethernet Structure

The term Ethernet refers to the family of local-area network (LAN) products covered by the IEEE 802.3 standard that defines what is commonly known as the CSMA/CD protocol. Four data rates are currently defined for operation over optical fiber and twisted-pair cables:

- 10 Mbps—10Base-T Ethernet
- 100 Mbps—Fast Ethernet
- 1,000 Mbps—Gigabit Ethernet
- 10,000 Mbps—10 Gigabit Ethernet

Ethernet is currently used for approximately 85 percent of the world's LAN-connected PCs and workstations. Ethernet is the major LAN technology because of the following characteristics:

- Is easy to understand, implement, manage, and maintain.
- Allows low-cost network implementations.
- Provides extensive topological flexibility for network installation.
- Guarantees successful interconnection and operation of standards-compliant products, regardless of manufacturer.

  Figure 2.6, one of the twisted-pair cables type

Figure 2.6: Ethernet Cable

2.4.4 Internet

The internet is a network of global exchanges – including private, public, business, academic and government networks – connected by guided, wireless and fiber-optic technologies as figure 2.7.

Figure 2.7: Internet

The terms internet and World Wide Web are often used interchangeably, but they are not exactly the same thing; the internet refers to the global communication system, including hardware and infrastructure, while the web is one of the services communicated over the internet[5].

2.5 Client side

A client is a computer program that, as part of its operation, relies on sending a request to another computer program (which may or may not be located on another computer). For example, web browsers are clients that connect to web servers and retrieve web pages for display. Email clients retrieve email from mail servers. Online chat uses a variety of clients, which vary depending on

the chat protocol being used. Multiplayer video games or online video games may run as a client on each computer. The term "client" may also be applied to computers or devices that run the client software or users that use the client software[6].

The term was first applied to devices that were not capable of running their own stand-alone programs, but could interact with remote computers via a network. These terminals were dumb central PC clients sharing time.

In the context of the World Wide Web, commonly encountered computer languages, which are evaluated or run on the client side, include figure 2.8 explain it:

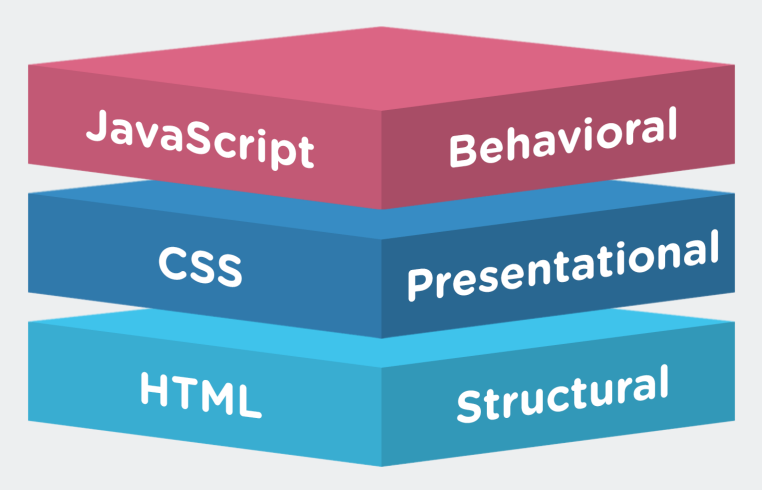


Figure 2.8: client side technologies

- **HTML**

  Is short for Hyper Text Markup Language. HTML is used to create electronic documents (called pages) that are displayed on the World Wide Web. Each page contains a series of connections to other pages called hyperlinks. Every web page you see on the Internet is written using one version of HTML code or another[7].

  HTML code ensures the proper formatting of text and images so that your Internet browser may display them as they are intended to look. Without HTML, a browser would not know

how to display text as elements, load images, or other elements. HTML also provides a basic structure of the page, upon which Cascading Style Sheets are overlaid to change its appearance. One could think of HTML as the bones (structure) of a web page, and CSS as its skin (appearance).

- **CSS**

  Is short for Cascading Style Sheet, may be used to change the font used in certain HTML element, as well as its size and color. A single CSS file may be linked to multiple pages, which allows a developer to change the appearance of all the pages at the same time.

- **JavaScript**

  Is a high-level, dynamic, weakly typed, prototype-based, multi-paradigm, and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make web pages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine[8]. Each of the many JavaScript engines represent a different implementation of JavaScript, all based on the ECMA Script specification, with some engines not supporting the spec fully, and with many engines supporting additional features beyond ECMA.

2.6 Server side

In a technical sense, a server is an instance of a computer program that accepts and responds to requests made by another program, known as a client. Less formally, any device that runs server software could be considered a server as well. Servers are used to manage network resources. For example, a user may setup a server to control access to a network, send/receive e-mail, manage print jobs, or host a website[9].

2.6.1 Type of Server

The following list contains links to various server types.

- **Application server**

  The application server is a framework, an environment where applications can run; no matter what they are or what functions, they perform. An application server can be used to develop and run web-based applications. There are a number of different types of application servers, including Java, PHP, Node.js and .NET Framework application server's figure 2.9, shown of the application server.

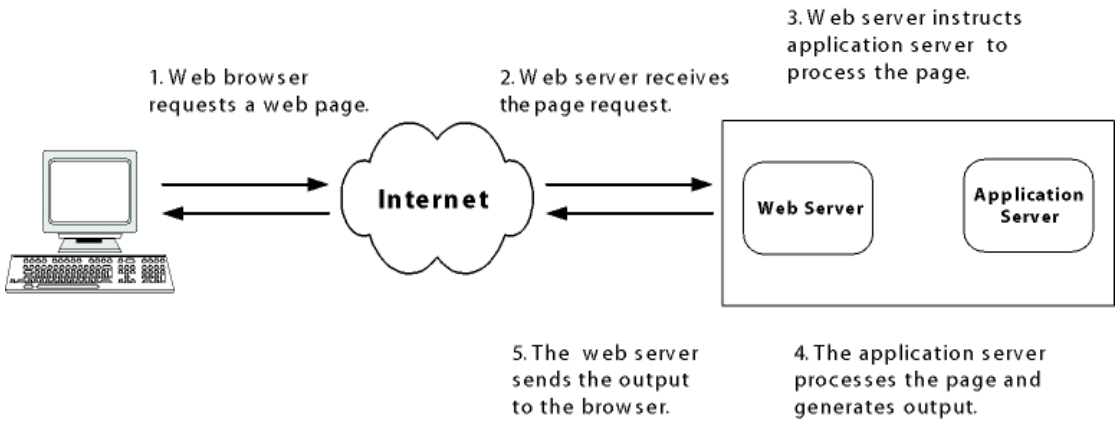


Figure 2.9: Application Server

- **Database server**

  Is a computer system that provides other computers with services related to accessing and retrieving data from a database. Access to the database server may occur via a "front end" running locally a user's machine (e.g. phpMyAdmin), or "back end" running on the database server itself, accessed by remote shell. After the information within the database is retrieved, it is outputted to the user requesting the data.

  Many companies utilize a database server for storage. Users can access the data by executing a query using a query language specific to the database. For example, SQL is a good example of a query language.

  This server can be used in our system for easy access and retrieval of data because it uses databases to store these data. Figure 2.10; show the database server parts with the client.

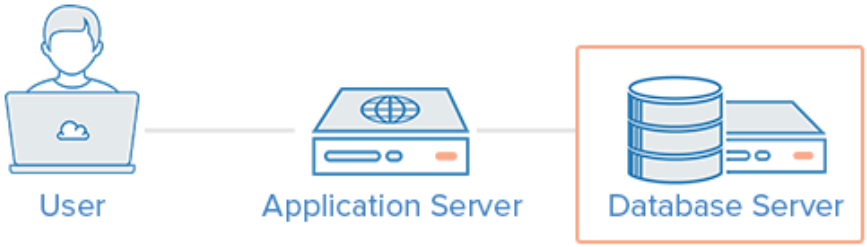


Figure 2.10: Database server

- **Web server**

  Computer or collection of computers used to deliver web pages and other content to multiple users. Figure 2.11, is shown of computer to use the web server.

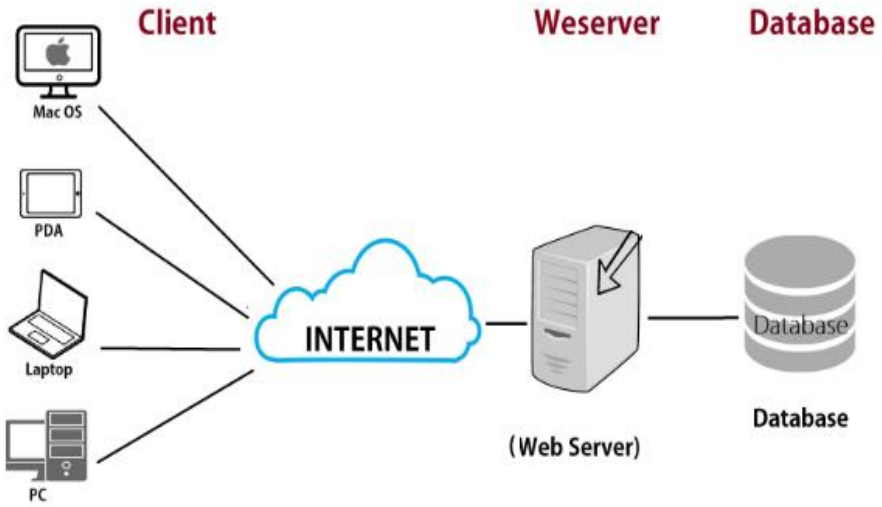


Figure 2.11: Web Server

2.6.2 Programming server

In this system we will use Web Server with Database Server, and these servers are programmed through the use of many programming languages, which we will use are:

- **REST (Representational State Transfer)**

  Is an architectural style, and an approach to communications that is often used in the development of Web services. The use of REST is often preferred over the more heavyweight SOAP (Simple Object Access Protocol) style because REST does not leverage as much bandwidth, which makes it a better fit for use over the Internet. The SOAP approach requires writing or using a provided server program (to serve data) and a client program (to request data).

- **Node.js**

  is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side. Historically, JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser[10].

- **Express - Node**

  Is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework –

  - Allows setting up middle wares to respond to HTTP Requests.
  - Defines a routing table, which is used to perform different actions based on HTTP Method and URL.
  - Allows to dynamically rendering HTML Pages based on passing arguments to templates.

- **MySQL**

  Is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

  SQL is the most popular language for adding, accessing and managing content in a database. It is most noted for its quick processing, proven reliability, ease and flexibility of use. MySQL is an essential part of almost every open source Node.js application[11].

2.7 The software used in the system

There are many programs used in web server programming and client programming. In this system, we will use Code Studio Code, and use Connectify dispatch program to integrate two networks connected to the Internet.

- **Visual Studio Code**

  Is a code editor redefined and optimized for building and debugging modern web and cloud applications.

- **WampServer**

  refers to a software stack for the Microsoft Windows operating system, created by Romaine Bourdon and consisting of the Apache web server, Open SSL for SSL support, MySQL database and PHP programming language.

- **Connectify Dispatch**

  Combines network adapters to increase speed and reliability, works by simultaneously leveraging the multiple network adapters and high-speed USB ports available on modern systems to boost bandwidth and reliability.

In the system using the connectify dispatch for integrated two networks, cellular network Ethernet network figure 2.12 shown it.

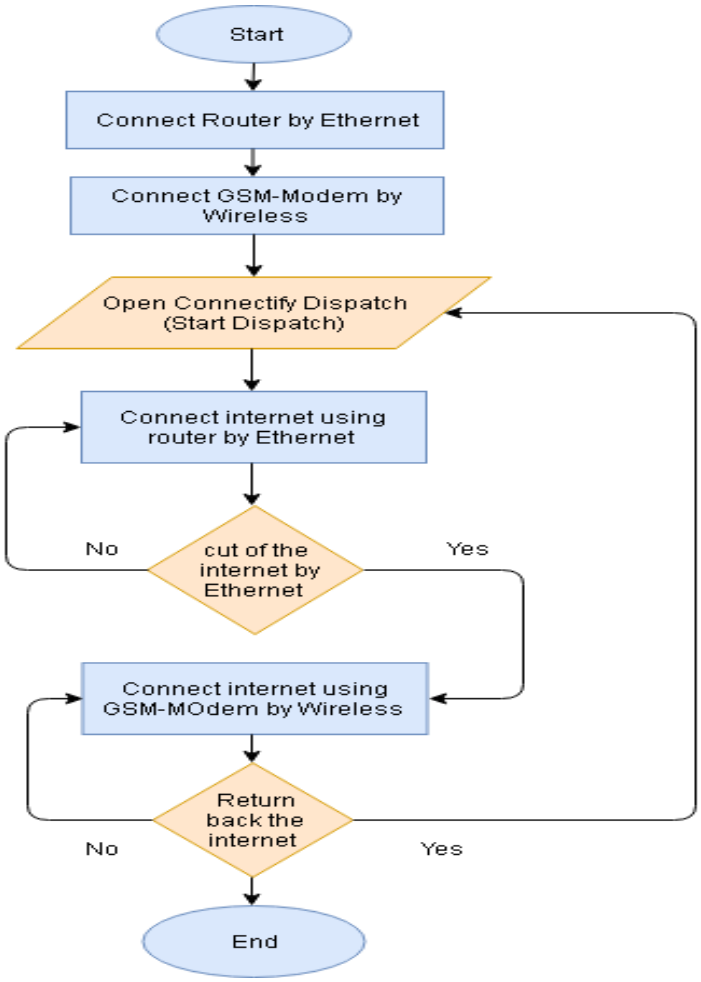


Figure 2.12: Network Connection

- When connecting the computer to the two networks, the LAN through Ethernet and the second network through the cellular network (GSM- Modem).
- Install the Connectify dispatch application.
- The two networks will appear on the Connectify dispatch application and then press start dispatch.
- If cut off the internet from router will connected by GSM-modem.

3

Chapter Three Conceptual Design

3.1 Introduction

3.2 General Block Diagram of the System

3.3 Parts of the System

3.4 Installing the programs

3.1 Introduction

This chapter describes the concepts of the system design. We will illustrate the general block diagram for main component of the system.

3.2 General Block Diagram of the System

The system general block diagram is illustrated in figure 3.1, the client is a computer, and communicates with the server (server program) throughout two line of the network (internet line from local network, and the other provided by a telecommunication network activated when first line is off), and when connected, the site content is loaded onto the client computer, and the user can use the system.

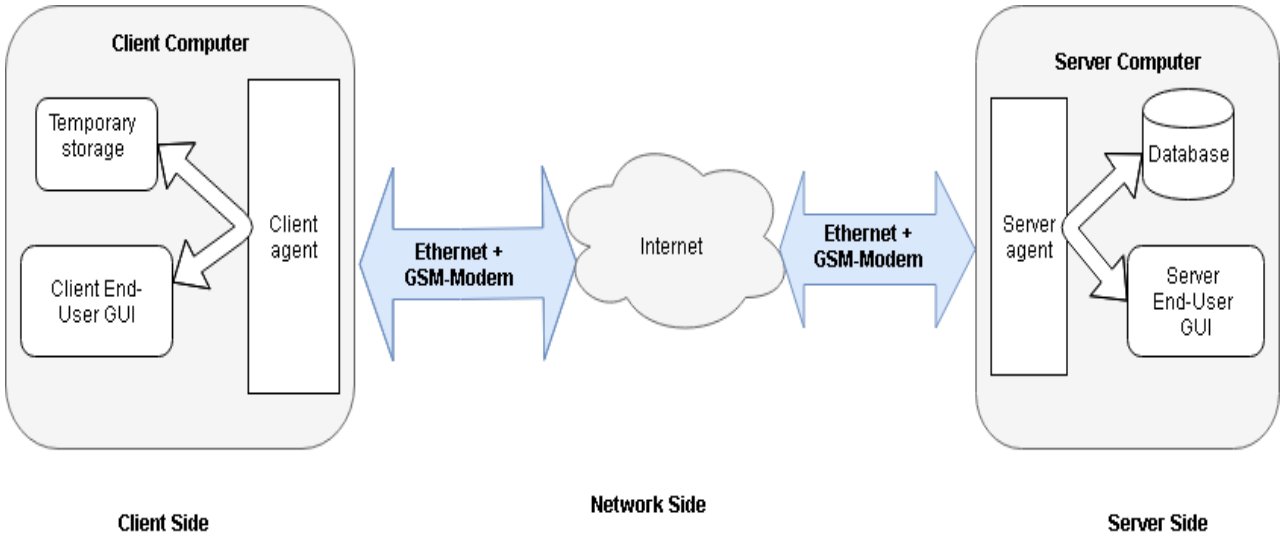


Figure 3.1: General Block diagram

3.3 Parts of the System

The system can be divided into two parts; the first part is hardware and consists of the integration of two technologies of Internet services, the local Internet service with the Internet service cell. The second part is software, contains server side programming (server program), and client side programming (website), flow chart is explain of the parts the system as shown figure 3.2:

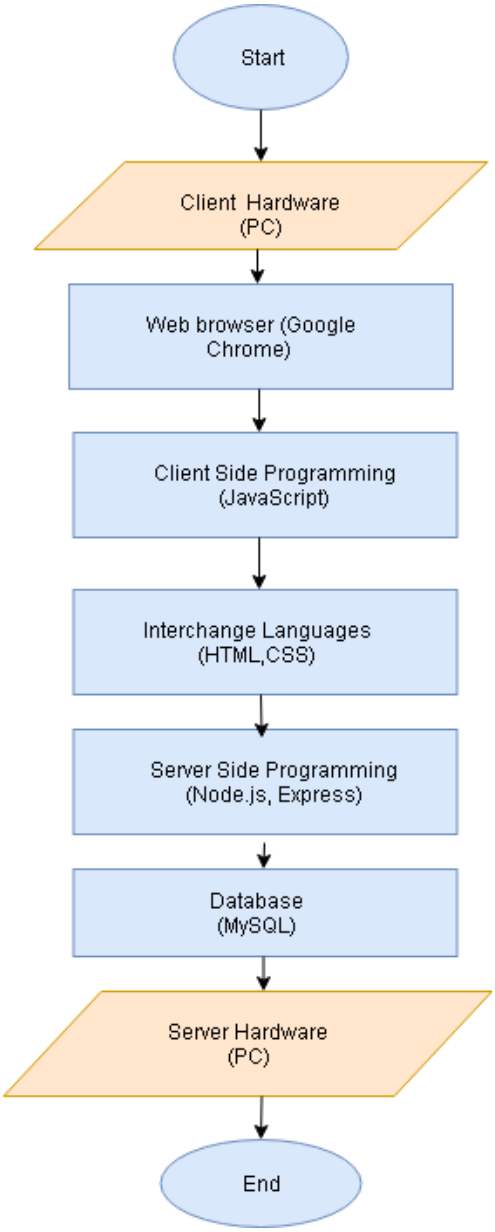


Figure 3.2: Flow chart for the system

3.3.1 Hardware of the system

3.3.1.1 Router Modem

In this system, we use Internet connection technology through land networks or local networks, which is the main part of the system to connect to the Internet via an Ethernet cable show in figure 3.3.

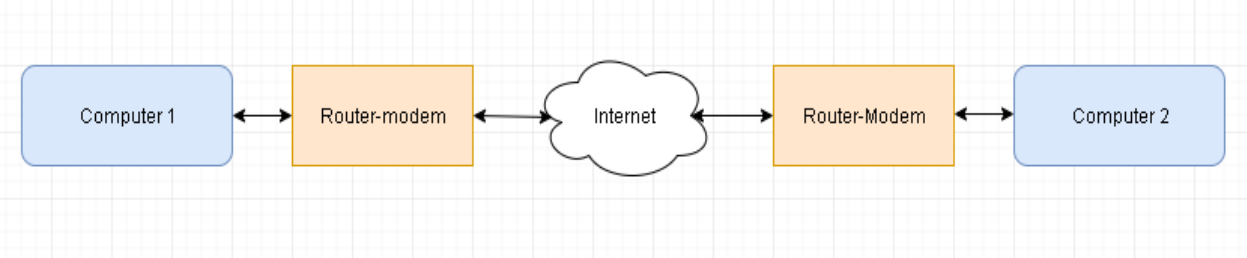


Figure 3.3: Router-Modem block diagram

3.3.1.2 GSM-Modem

GSM modem is one of the physical devices used to connect the system with hospitals; we use this modem in case the local network is off, figure 3.4 show how to connect GSM with the network. To activate the GSM modem, we send a message to the company to activate the data on the SIM card, the message including the status of the modem. After activation, we verify the Internet connection by connecting it to the computer and receive data from it. Then we integrate the GSM to work with the local network.

Figure 3.4: Block diagram of the GSM

3.3.2 Software of the system

Our software divided into two parts, the first part is server-side programming, here we use Node.js to create our web server, the second part is client-side programming, and here we use different technologies to build the website.

3.3.2.1 Server Side

We are using Node.js to create our server, Node.js provide us with Express framework which used to create a lightweight webserver, the server program is connected to MYSQL database using

MySQL-module(also provided by Node.js) as shown in figure 3.5, to make this server works online we can deploy it on Heroku server.

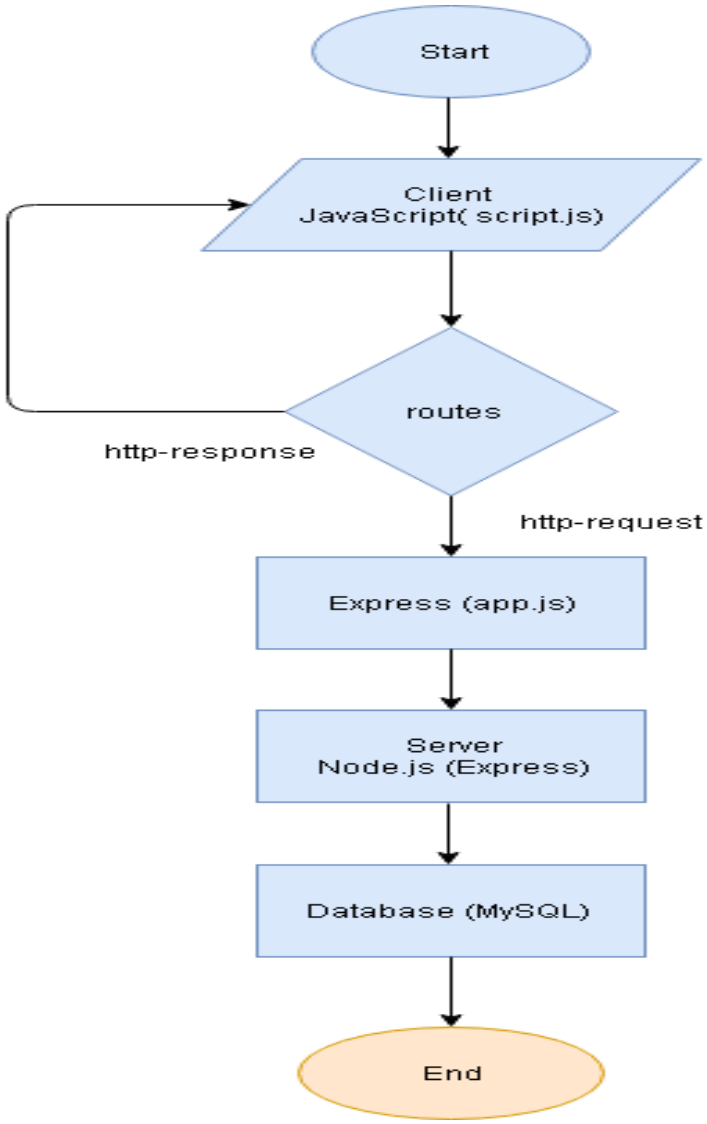


Figure 3.5: Server Programming

- Based on client request, retrieves posts from database using Nodes and MySQL.
- Uses Node to process posts into pure HTML, which it send back to client via HTTP.

- In the server, a route directs the information according to the client request and return the response to it.
- For our system implementation, a type of server that can run a Node.js application such as Heroku web server will be used.

What is Heroku?

Is a cloud platform as a service (PaaS) supporting several programming languages that is used as a web application deployment model. Heroku, one of the first cloud platforms, has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it lets the developer build, run and scale applications in a similar manner across all the languages. Simply put, Heroku is a cloud platform that lets companies/individuals build, deliver, monitor, and scale apps. It is often regarded as the fastest way to go from idea to URL, bypassing all those infrastructure headaches (i.e., you do not have to worry about infrastructure; you just focus on your application)[12].

3.3.2.2 Client Side

A website is programmed using web technologies (HTML, CSS, and JavaScript) such as the figure 3.6. HTML and CSS is used to create the interfaces, and JavaScript is used to control the logic of the system, also we use a JavaScript library called jQuery, which facilitate the process of making requests to our server.

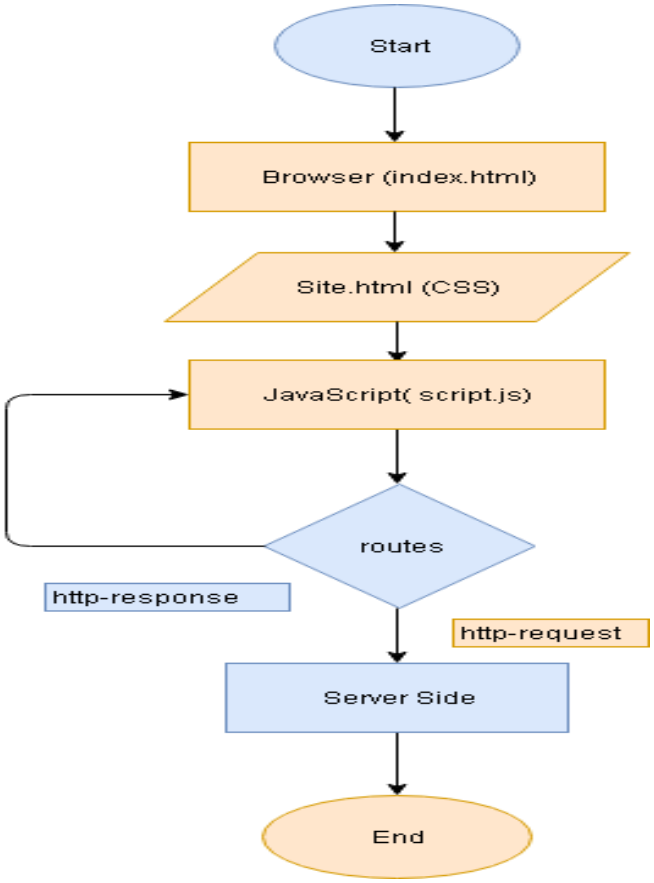


Figure 3.6: Client Side

- Browser running on your local computer/device.
- A user visits a web page hosted on the web server.
- Routing refers to how an application's endpoints (URIs) respond to client requests. For an introduction to routing, see basic routing.
- Makes HTTP requests to fetch server contents.
- Browser can run JavaScript, allowing for dynamic of interactive HTML pages.
- Server`s HTTP response contains HTML and CSS that browser renders into a webpage.
- JavaScript coding can still be run by the client`s browser to modify the HTML page.

3.3.2.3 Database for the system

A computer system that processes database queries, in our system we will use a database server that enable us to do a several kinds of database operations (insert, update and delete), such as MySQL database server (separate from webserver) as shown figure 3.7:

Our system consist of several items, which we have to store data about it, and these items are:

1. Users of the system: the user of our system refers to the person who represent a hospital, in other word, each hospital has account on the system and the user can use this account to access the system. We want to store these data about the user (hospital) to create the account ID, Email, Password, Hospital Name, Hospital Location, Hospital image, and the account is active.

2. Blood units: each hospital has different type of blood units with different quantities for these units, so for each hospital we have to store the type of each blood unit e.g. AB, O-, etc..., along with the quantity available in the hospital. These blood units can be accessed through a special opening to the hospital called the hospital_id.

3. Admins: this table store the credential of the main system admin, this admin has the ability to allow any hospital to join our system. The following is our database in table 3.1.

4. A computer system that processes a database queries, in our system we will use a database server that enable us to do a several kinds of database operations (insert, update and delete), such as MySQL database server (separate from webserver).

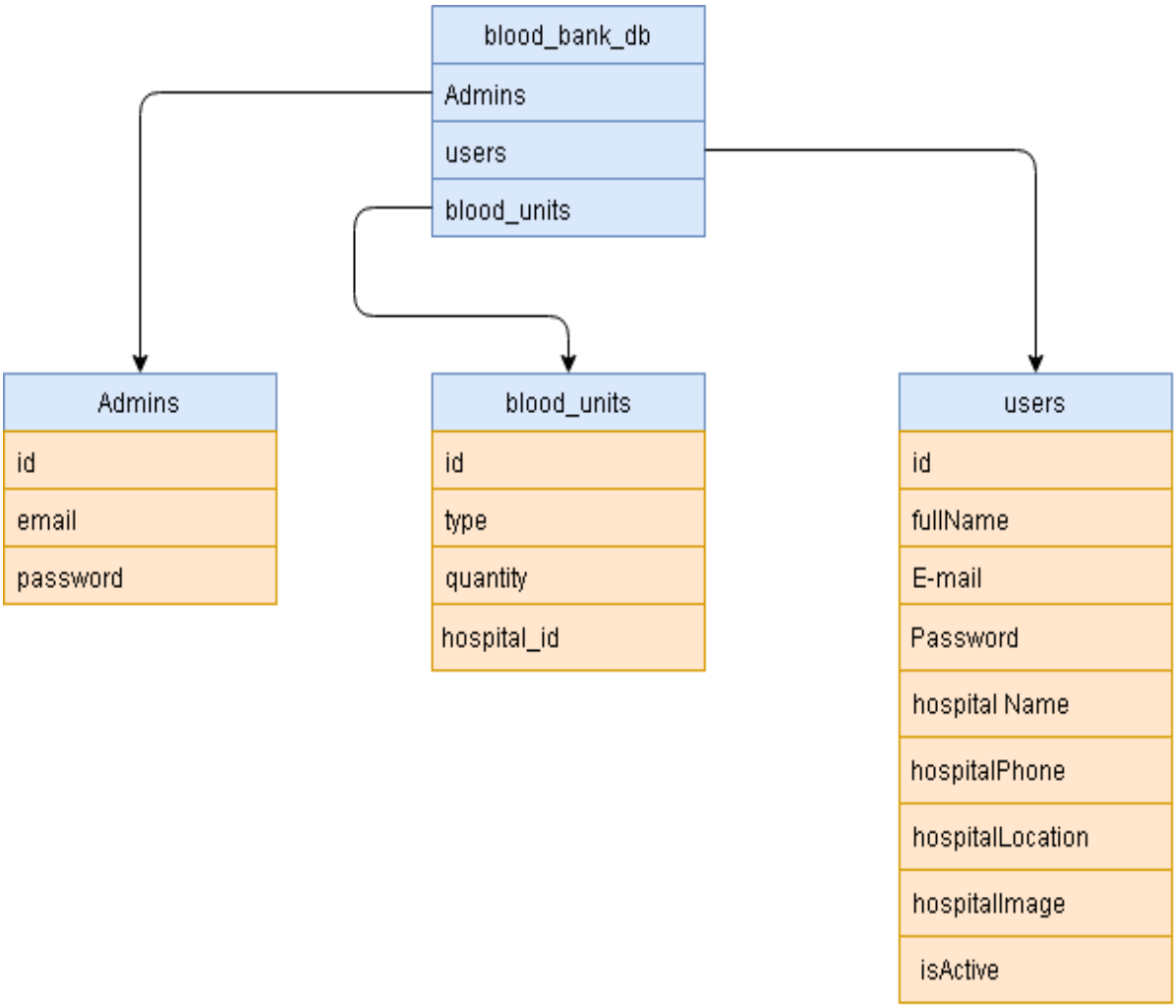


Table 3.1: Database table

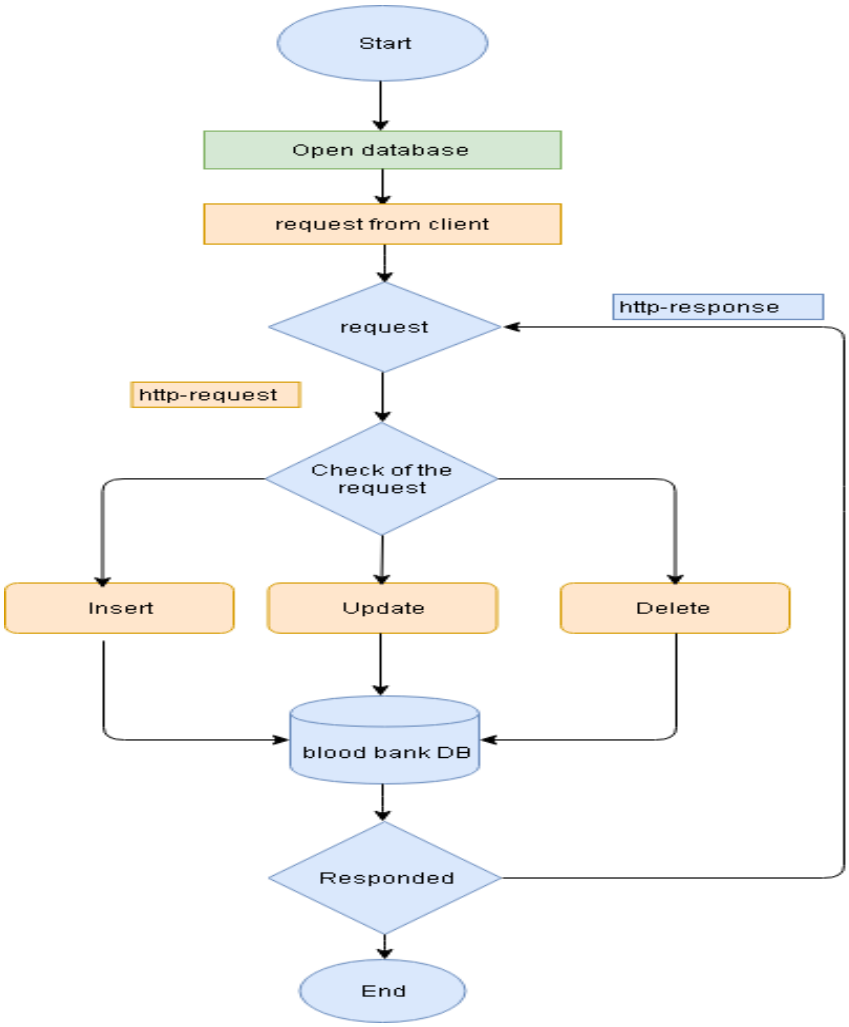


Figure 3.7: Database

- Run the server program on a server to start serving data to the clients.
- The server will start listening to the client requests on active port number specified during the development process.
- When the server receive any request, the first step is to check if the client is authorized to access data or not, then it the data is operations (insert, update and delete).
- If the user is authorized then the server will start handling the request, interact with the database to get the data using query language, and then send back a response to the client.

- After handling the client request, the server returns to the listening phase and will repeat all the processes again when receiving a new request.

3.4 Installing the programs

Programs that can be used as mentioned above it:

3.4.1 Connectify Dispatch

In the system using this program to integrated two networks cellular and LAN networks, when connected the PC by this network and after install this program through the internet, interface of the program as figure 3.8:

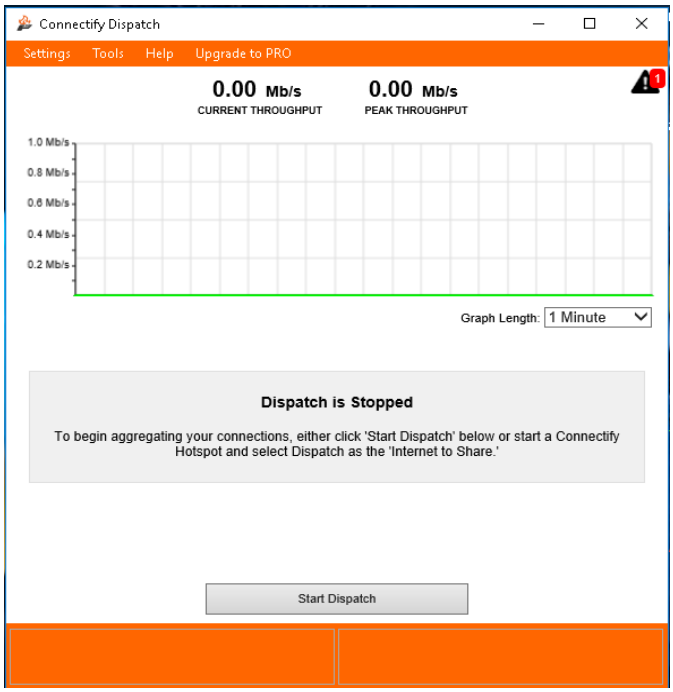


Figure 3.8: Connectify Program

3.4.2 Node.js Program

Installing Node and NPM is straightforward using the installer package available from the Node.js® web site.

Make sure you have Node and NPM installed by running simple commands to see what version of each is installed and to run a simple test program:

- **Test Node.** To see if Node is installed, open the Windows Command Prompt, Power shell or a similar command line tool, and type node -v. This should print a version number, so you will see something like this v0.10.35.
- **Test NPM.** To see if NPM is installed, type npm -v in Terminal. This should print NPM's version number so you'll see something like this 1.4.28
- **Create a test file and run it.** A simple way to test that node.js works is to create a JavaScript file: name it script.js, and just add the code console.log ('Node is installed!'); To run the code simply open your command line program, navigate to the folder where you save the file and type node script.js. This will start Node and run the code in the script.js file. You should see the output Node is installed.

3.4.3 WampServer

WampServer is a local Windows server package, which allows you to install and host Web applications that use Apache, PHP, and MySQL. Until the databases are created for these servers, the program format.

3.5 Schedule

Weeks / Process	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Search																
Server Programming																
Client programming																
Connect server with client																

Table 3.2: Schedule table

4

Chapter Four Detailed Design

4.1 Introduction

4.2 Detailed diagram of the system

4.3 Programming

4.4 Database

4.5 Routes

4.6 Security

4.7 Home Page

4.8 Steps for upload the system on the Hurok server

4.1 Introduction

The details of any system is an important part, Operation and controlling of any component in this system. This chapter explains the required software with detailed design.

4.2 Detailed diagram of the system

The system detailed diagram is illustrated in figure 4.1, explains how the client to server connectivity and access to data using certain programming languages.

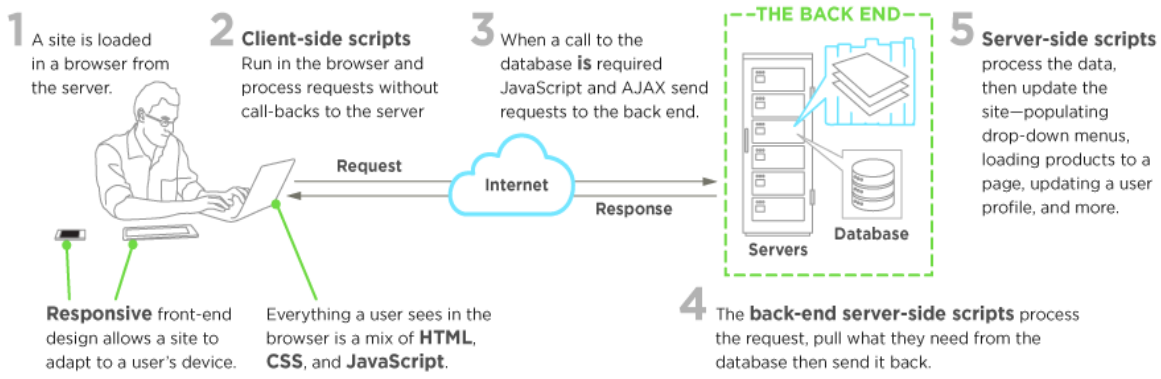


Figure 4.1: Detailed diagram of the system

4.3 Programming

At programming, a server program (REST API) will be developed, we will use Node.js to create our server program, and Node.js provide us with Express module, which facilitate the process of creating a web server program. We choose Node.js because it is easy to use and fast to learn, and we can create our server program using simple commands. Another thing that we need is a Database server, we will use MySQL server to create our database and deal with it.

4.3.1 Steps for create server

- **Step1: Install Node.Js**

  This is easy. Hit the Node.js website and click the big green Install button for the LTS (long-term-stable) version. It will detect your OS and give you the appropriate installer. Run the installer. That is it, you have installed Node.js and, equally important, NPM—Node Package Manager—which lets you add all kinds of great stuff to Node quickly and easily as in figure 4.2.

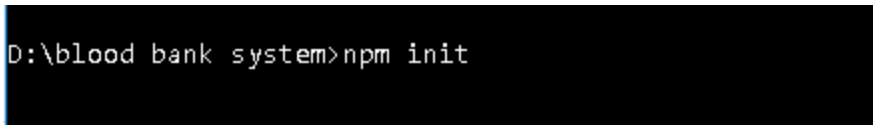


Figure 4.2: Create a package. JSon

- Open a command prompt
- Cd to the directory in which you wish to keep your test apps (for the purposes of this tutorial, C:\node).

- **Step 2: Install Express Generator**

Now that we have Node running, we need the rest of the stuff we are going to actually use to create a working website. To do that we're going to install Express, which is a framework that takes Node from a barebones application and turns it into something that behaves more like the web servers we're all used to working with (and actually quite a bit more than that). We need to start with Express-Generator, which is actually different from express itself; it is a scaffolding app that creates a skeleton for express-driven sites. In your command prompt as shown in figure 4.3:

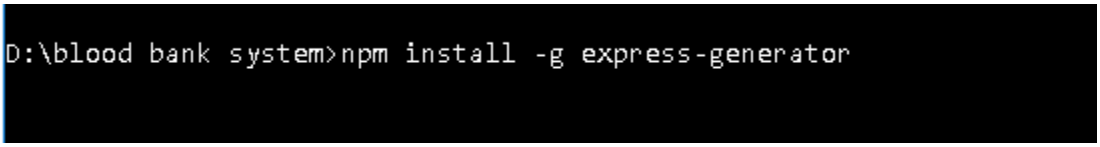


Figure 4.3: Install express generator

The generator should auto-install, and since it (like all packages installed with -g) lives in your master NPM installation directory, it should already be available in your system path. So let's use our generator to create the scaffolding for a website.

- **Step 3: Create Express**

  For the sample Local Library app, we are going to build; we will create a project named express-locallibrary-tutorial using the Pug template library and no CSS stylesheet engine.

  First, navigate to where you want to create the project and then run the Express Application Generator in the command prompt as shown in figure 4.4:

```
D:\blood bank system>express express-locallibrary-tutorial --view=pug

   create : express-locallibrary-tutorial\
   create : express-locallibrary-tutorial\public\
   create : express-locallibrary-tutorial\public\javascripts\
   create : express-locallibrary-tutorial\public\images\
   create : express-locallibrary-tutorial\public\stylesheets\
   create : express-locallibrary-tutorial\public\stylesheets\style.css
   create : express-locallibrary-tutorial\routes\
   create : express-locallibrary-tutorial\routes\index.js
   create : express-locallibrary-tutorial\routes\users.js
   create : express-locallibrary-tutorial\views\
   create : express-locallibrary-tutorial\views\error.pug
   create : express-locallibrary-tutorial\views\index.pug
   create : express-locallibrary-tutorial\views\layout.pug
   create : express-locallibrary-tutorial\app.js
   create : express-locallibrary-tutorial\package.json
   create : express-locallibrary-tutorial\bin\
   create : express-locallibrary-tutorial\bin\www

   change directory:
     > cd express-locallibrary-tutorial

   install dependencies:
     > npm install

   run the app:
     > SET DEBUG=express-locallibrary-tutorial:* & npm start
```

Figure 4.4: Create express

- **Step 4: Edit Dependencies**

  OK, now we have some basic structure in there, but we are not quite done. You will note that the express-generator routine created a file called package.json in project file directory. Open this up in a text editor and it will look like in figure 4.5:

```
{
  "name": "blood-bank-system",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.18.2",
    "bootstrap": "^4.0.0",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "express": "~4.15.5",
    "jade": "~1.11.0",
    "morgan": "~1.9.0",
    "serve-favicon": "~2.4.5"
  }
}
```

Figure 4.5: Package.json

This is a basic JSON file describing our app and its dependencies. We need to add a few things to it. Specifically, calls for MySQL and Monk. Let's make our dependencies object look like in figure 4.6:

```
  "dependencies": {
    "body-parser": "~1.18.2",
    "bootstrap": "^4.0.0",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "express": "~4.15.5",
    "jade": "~1.11.0",
    "morgan": "~1.9.0",
    "mysql": "^2.15.0",
    "serve-favicon": "~2.4.5"
  }
```

Figure 4.6: Dependencies

- **Step 5: Install Dependencies**

  Now we have defined our dependencies and we are ready to go. Note that the version numbers for those two modules are current as of the latest update, but new versions of

NPM modules are frequently rolled out. These versions are proven to work with this tutorial; if you go with the latest versions, as shown in figure 4.7:

Figure 4.7: NPM install

4.3.2 Network side

Can using two way for integrate of the two or more network by:

- **Dispatch Proxy**

  When installing Node.js we use a command to merge two networks. In this system we integrate the private LAN of the hospital (Ethernet) with the cellular network (GSM modem) and this can be used if disconnecting one of the two networks will be connected to the other network. Commend as in figure 4.8:

Figure 4.8: Merge two networks

When you connect the two networks and after you install the merge on the device through Node.js, when the merger we operate dispatch start as figure 4.9:

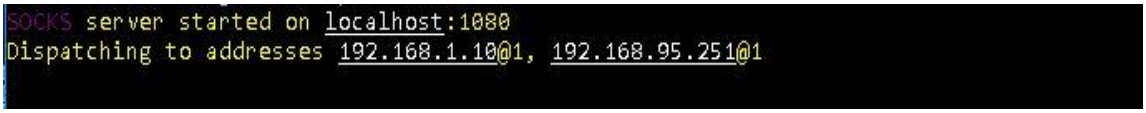


Figure 4.9: Dispatch start

- **Connectify Dispatch**

The Connectify program will open and the connected networks will appear on the device and we will start the dispatch as figure 4.10:

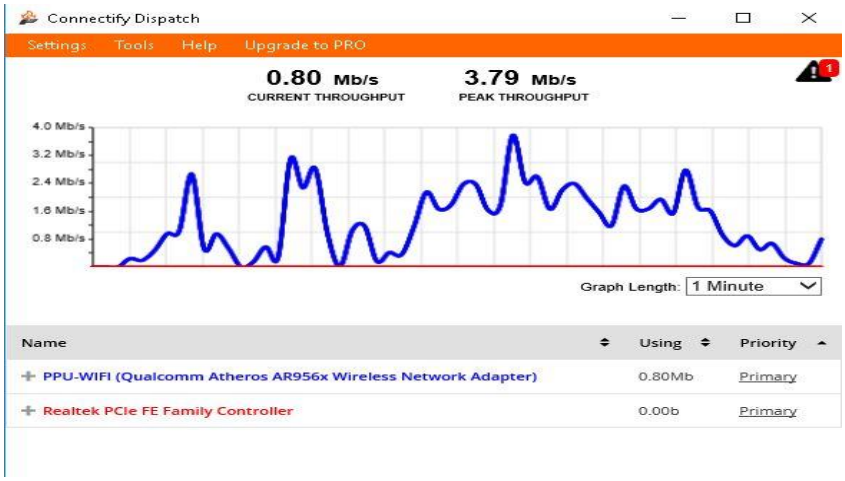


Figure 4.10: Connectify Dispatch

4.4 Database

- **Open the WampServer Program**

The phpMyAdmin page and this page needs a program to be opened is wampserver as in figure 4.11, and phpMyAdmin page as in figure 4.12 create the database.

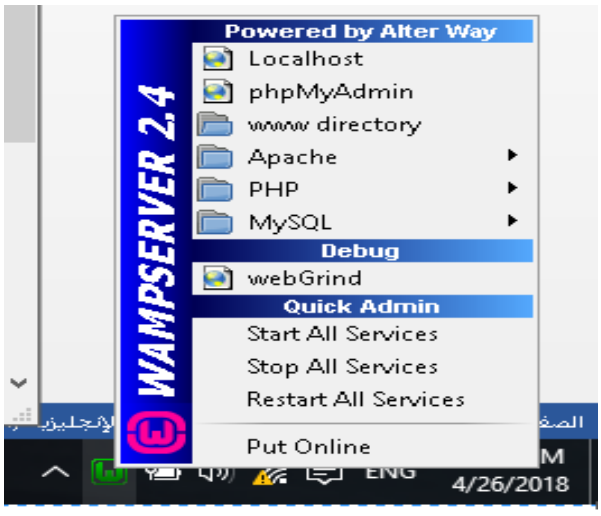


Figure 4.11: WampServer Program

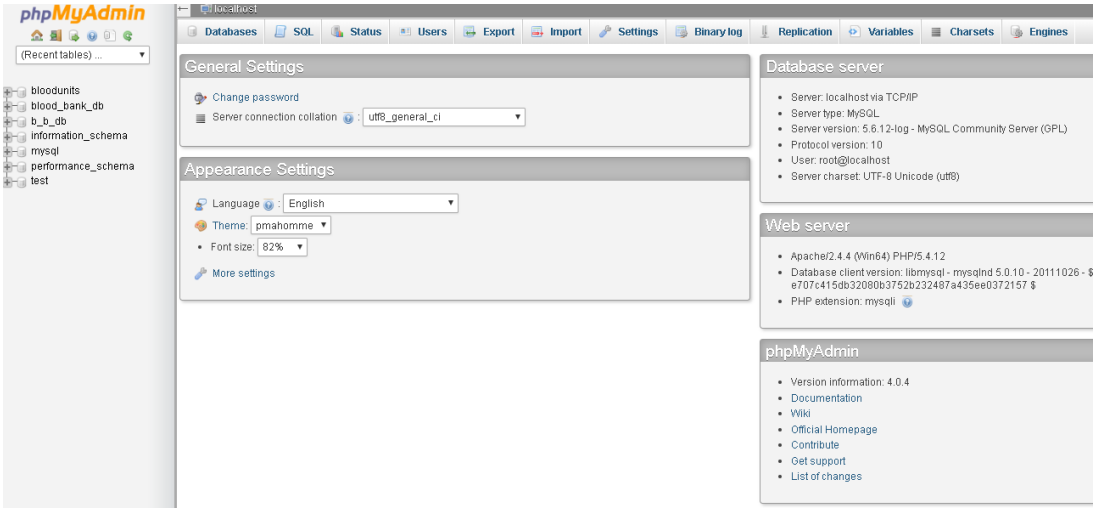


Figure 4.12: PhpMyAdmin Page

- **Creating database**

Create a database on phpMyAdmin is a popular web-based administration tool used to administrate MySQL servers and databases. This tutorial illustrates how to create a new database and a new user to access that database on a MySQL Server using phpMyAdmin.

From this phpMyAdmin, we have created a system database called blood bank, which contains a special database for the user and another for blood units, and create admin for the system as in the figure 4.13.

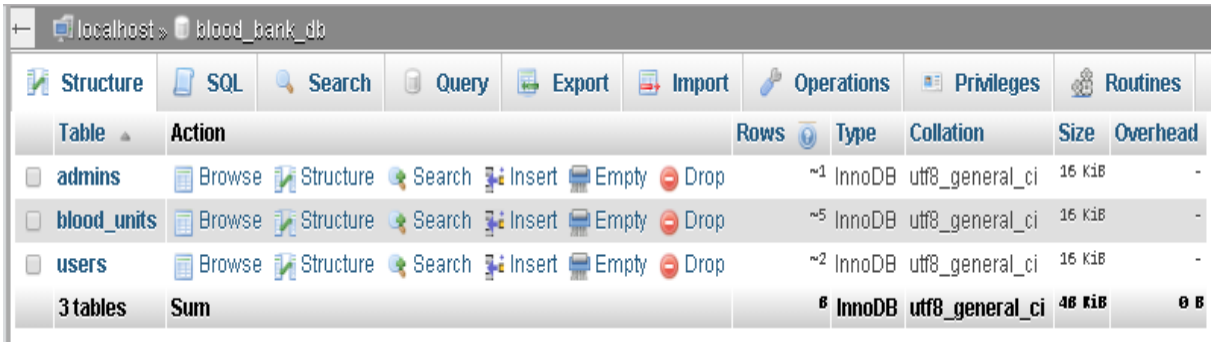


Figure 4.13: blood bank database

Database is contains users, blood units, and admins that contain the number of fields needed by the system such as ID and other as figure 4.14:

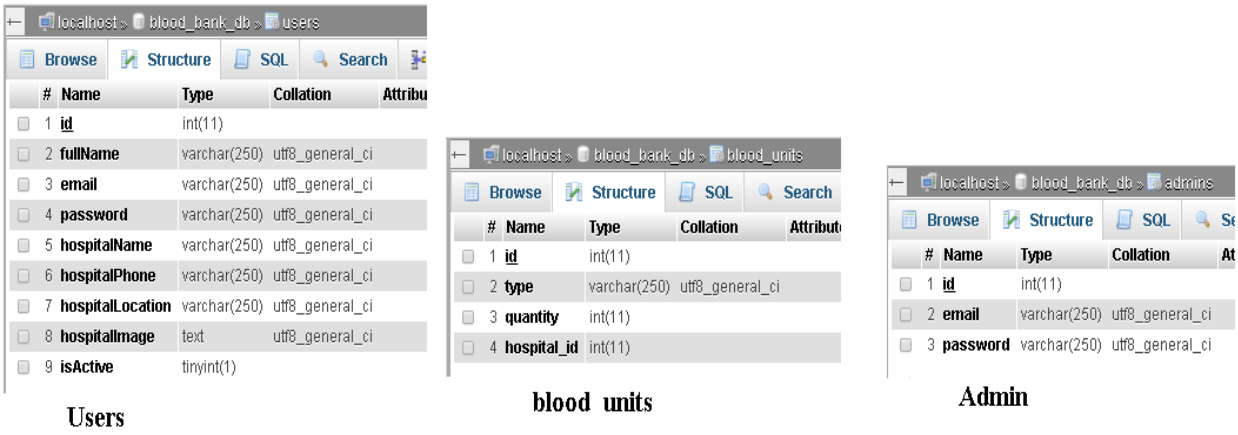


Figure 4.14: Elements of the database

- **Connect the database with the server**

Once you have MySQL up and running on your computer, you can access it by using Node.js. To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "MySQL" module, downloaded from NPM. To download and install the "MySQL" module, open the Command Terminal and execute the following as shown in figure 4.15:

```
D:\blood bank system>npm install mysql --save
```

Figure 4.15: Connection of the database

After use module MySQL, start by creating a connection to the database. Use the username and password from your MySQL database; let's see how to execute SQL queries. We will start by specifying the database name (sitepoint) in the createConnection command, the name of the database must be the same name that was created in phpmyadmin page as shown in figure 4.16:

```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "blood_bank_db"
});
```

Figure 4.16: Create Connection

Once the connection is established, we will use the connection variable to execute a query against the database table blood bank as shown figure 4.17:

```
connection.connect(function(err, conn) {
  if (err){
    console.log(err)
    throw err;
  }
  console.log("Success Connected to DataBase . . .");
});

module.exports = connection;
```

Figure 4.17: Establish of the connection

4.5 Routes

A route is a section of Express code that associates an HTTP verb (GET, POST, PUT, DELETE, etc.), an URL path/pattern, and a function that is called to handle that pattern.

There are several ways to create routes. In this system, we are going to use the express.Router middleware as it allows us to group the route handlers for a particular part of a site together and access them using a common route-prefix. We will keep all our library-related routes in a "catalog" module, and, if we add routes for handling user accounts or other functions, we can keep them grouped separately.

First, we create routes for a user's in a module named users.js. The code first imports the Express application object, uses it to get a Router object and then adds a couple of routes to it using the get() method. Last of all the module exports the Router object as shown in figure 4.18:

```
var express = require('express');
var router = express.Router();
var db = require('../database/db.js');
var jwt = require('jsonwebtoken');
var multer = require('multer');


/************ file storage config ************/
var fileStorage = multer.diskStorage({
    destination: function (req, file, callback) {
        var Path = './uploads';
        callback(null, Path);
    },
    filename: function (req, file, callback) {
        callback(null, parseInt(Math.random()*10000) + file.originalname);// change the name of file when store it.
    }
});

var upload = multer({storage: fileStorage});

/* GET users listing. */
router.get('/', function (req, res, next) {
    res.send('respond with a resource');
});
```

Figure 4.18: Code of the Routes

Secondly, to use the router module in our main app file we first require() the route module (users.js). We then call use() on the Express application to add the Router to the middleware handling path, specifying an URL path of 'users' as shown in figure 4.19:

```
app.use('/', index);
app.use('/users', users);
```

Figure 4.19: Require of the route

- **Create User**

  When creating a user, we use HTML and CSS to write code to design the account creation page through which the user creates his / her own hospital. Use a <form> element to process the input. Then add inputs (with a matching label) for each field, use a element to split the page into a block until each block is placed in the field, the field and the button are designed through use a <input> element as shown in figure 4.20:

```
<form id="register-form"  name="createUserForm"  enctype="multipart/form-data"  role="form" style="display: none;" >

    <div class="form-group">
        <input type="text" name="fullName" id="user_name_new"  class="form-control" placeholder="الإسم الكامل*" value="">
    </div>
    <div class="form-group">
        <input type="email" name="email" id="user_email_new" class="form-control" placeholder="الإيميل*" value="">
    </div>
    <div class="err_msg" id="user_email_new_msg"></div>
    <div class="form-group">
        <input type="password" name="password" id="user_password_new" class="form-control" placeholder="كلمة المرور*">
    </div>
    <div class="form-group">
        <input type="password" name="confirm-password" id="user_confirm_pass" class="form-control" placeholder="تأكيد كلمة المرور*">
    </div>
    <div class="form-group">
        <input type="text" name="hospitalName" id="hospital_name_new"  class="form-control" placeholder="اسم المستشفى*" value="">
    </div>
    <div class="form-group">
        <input type="text" name="hospitalPhone" id="hospital_phone"  class="form-control" placeholder="هاتف المستشفى*" value="">
    </div>
    <div class="form-group">
        <input type="text" name="hospitalLocation" id="hospital_location"  class="form-control" placeholder="موقع المستشفى*" value="">
    </div>
    <div class="form-group">
        <label for="">تحميل صورة</label>
        <input type="file" name="hospitalImage" id="hospital_image"  class="form-control">
    </div>
    <p>* : تعني أن الحقل مطلوب.</p>
    <p id="signUpErr" class="err_msg"></p>
    <div class="form-group">
        <div class="row">
            <div class="col-sm-6 col-sm-offset-3">

                <!-- <input type="submit" name="register-submit" id="register-submit"  class="form-control btn btn-register" value="إنشاء حساب" > -->
                <input type="submit" name="register-submit" id="register-submit"  class="form-control btn btn-register" value="إنشاء حساب" >
            </div>
        </div>
    </div>
</form>
```

Figure 4.20: Code create user

When the account creation page is executed can be used multiple browsers, the browser that was used in this system is Google Chrome as shown in figure 4.21:

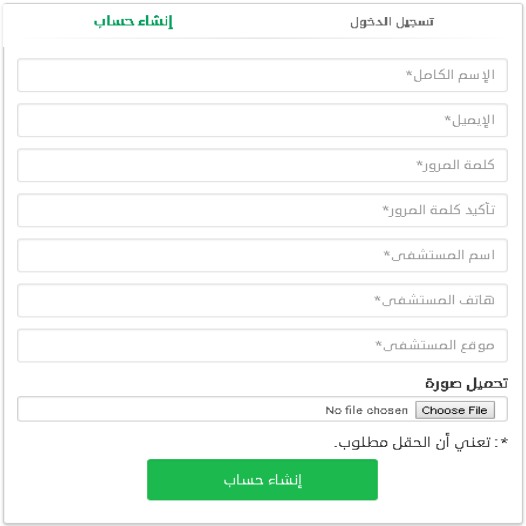


Figure 4.21: Page of the create user

After displaying the page to the user and filling out the fields and clicking the "Create Account" button, the information will be sent to the database and stored in it by creating a special path to create the account by using JavaScript and jQuery, the information is taken and placed in the matrix “userData” and load this array to the database from as you type this command (INSRT INTO users SET…) until the information is stored in the database of users, as shown in Figure 4.22:

```
router.post('/createUser', upload.single('hospitalImage'),function (req, res, next) {
console.log("req.body", req.body);
console.log("req.file", req.file);
//  return;
    var userData = {
        fullName: req.body.fullName,
        email: req.body.email,
        password: req.body.password,
        hospitalName: req.body.hospitalName,
        hospitalPhone: req.body.hospitalPhone,
        hospitalLocation: req.body.hospitalLocation,
        hospitalImage: req.file ? req.file.filename : 'hospital.jpg'
    }
    //   console.log(userData);
    db.query('INSERT INTO users SET ?', userData, function (error, results, fields) {
        if (error) {
            console.log(error)
            res.json({
                "success": false,
                "result": "error adding data"
            })
        } else {
            res.json({
                "success": true,
                "result": "success create new user"
            })

        }
    });

});
```

Figure 4.22: Route of Create user

- **Login of the system**

In entering the system, the user enters the previously created information, especially the e-mail and password. When entering the information, this information must be entered on a special login page and programmed by HTML and CSS as shown figure 4.23:

```
<form id="login-form" name="loginForm" role="form" style="display: block;" >
    <div class="form-group">
        <input type="email" name="email" id="user_email" tabindex="1" class="form-control" placeholder="الإيميل" value="">
    </div>
    <div class="form-group">
        <input type="password" name="password" id="user_password" tabindex="2" class="form-control" placeholder="كلمة المرور">
    </div>
    <p id="loginErr" class="err_msg"></p>

    <div class="form-group">
        <div class="row">
            <div class="col-sm-6 col-sm-offset-3">
                <input type="button" onclick="loginUser()" name="login-submit" id="login-submit" tabindex="4" class="form-control btn btn-login"
                value="تسجيل الدخول">
            </div>
        </div>
    </div>
    <div class="form-group">
        <div class="row">
            <div class="col-lg-12">
                <div class="text-center">
                    <a href="#" tabindex="5" class="forgot-password">نسيت كلمة المرور؟</a>
                </div>
            </div>
        </div>
    </div>
</form>
```

Figure 4.23: Login

When you finish writing of HTML code for login, it is executed through a browser so that the user can write the email and password to enter the system as shown in figure 4.24:

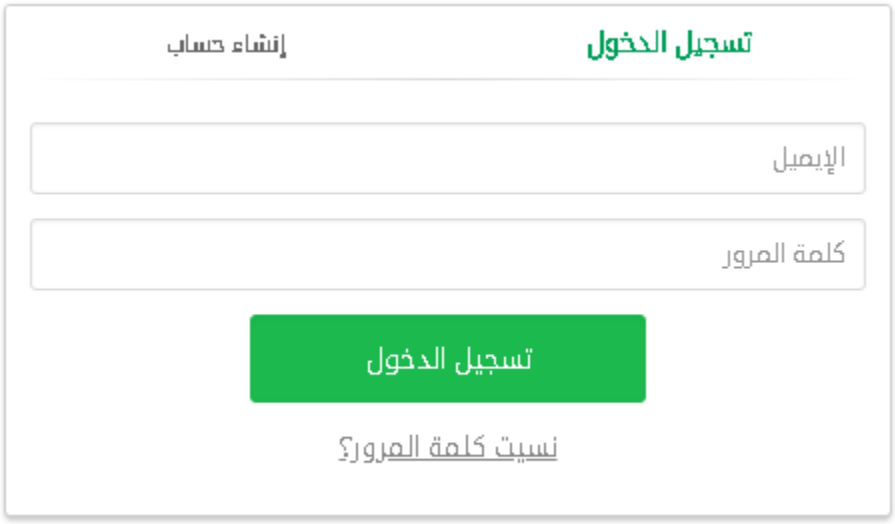


Figure 4.24: Code of implement the login page

After the user has filled in the e-mail and password to login on to the system, the system compares the previously stored information to verify whether the user is subscribed to this system or not. Through this command (SELECT *(means is all of information in this table) FROM users), the stored data is compared with the data entered if this is done by programming the Java script code and JQuery to create a path to the database as shown in Figure 4.25:

```
router.get('/loginUser', function (req, res, next) {
    var userData = {
        email: req.query.email,
        password: req.query.password
    }
    //   console.log(userData);
    db.query("SELECT * FROM users WHERE email = '" + userData.email + "' AND password = '" + userData.password +
     "' AND isActive = true", function (error, result) {
        if (error || (result.length == 0)) {
            console.log(result);
            console.log(error);
            res.json({
                "success": false,
                "result": "error user data"
            })
        } else {
            console.log(result[0].id);
            var token = jwt.sign({
                uid: result[0].id
            },
                '94Tre500*_q+esltwert#04*0$998rwertwebrbr5', {
                    expiresIn: 60 * 60 * 24 * 7 //expires in 1 week
                });

            res.json({
                "success": true,
                "result": {
                    "token": token
                }
            })
        }
    });

});
```

Figure 4.25: Route of a login

- **Admin**

Is the person who protects the site from people who do not have to log on to this system, the administrator login starts to “admin page” which is designed in HTML and CSS as shown in figure 4.26:

```
<form id="login-form" name="loginForm" role="form" style="display: block;" >
    <div class="form-group">
        <input type="email" name="email" id="user_email" tabindex="1" class="form-control" placeholder="الإيميل" value="">
    </div>
    <div class="form-group">
        <input type="password" name="password" id="user_password" tabindex="2" class="form-control" placeholder="كلمة المرور">
    </div>
    <p id="loginErr" class="err_msg"></p>

    <div class="form-group">
        <div class="row">
            <div class="col-sm-6 col-sm-offset-3">
                <input type="button" onclick="loginAdmin()" name="login-submit" id="login-submit" tabindex="4" class="form-control btn btn-login"
                value="تسجيل الدخول">
            </div>
        </div>
    </div>

</form>
```

Figure 4.26: Code HTML and CSS for Admin page

After writing the supervisor page in HTML and CSS, when executing this page as shown in the figure 4.27. The administrator then enters his / her e-mail and password. When the login button is pressed, a function “loginAdmin()”is executed through which the information is taken and placed in a matrix “userData” and sent to route of a “user/loginAdmin” as shown in the figure 4.28:

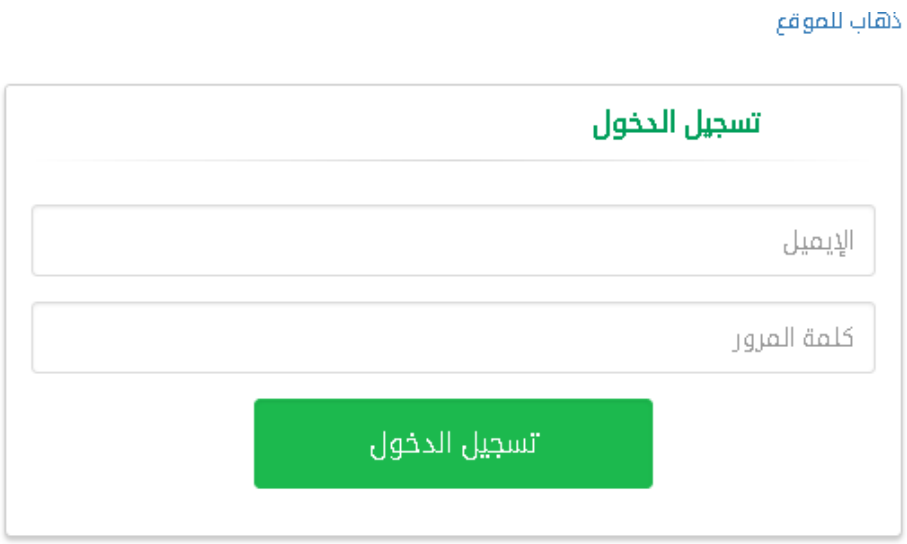


Figure 4.27: Admin page

```
function loginAdmin() {

    var loginForm = document.forms["loginForm"];
    var userData = {
        email: loginForm["email"].value || " ",
        password: loginForm["password"].value || " "
    }

    console.log(userData);

    $.get(serverAddress + 'users/loginAdmin', userData, function (data) {
        console.log(data);
        if (data.success) {
            // alert('تم تسجيل الدخول بنجاح');
            localStorage.setItem('bb_admin_token', data.result.token);

            var userData = {
                token:   data.result.token
            }

            $.get(serverAddress + 'users/getNewUsers', userData, function (data) {
                console.log(data)

                //
                var content = '<table class="table table-striped custom-table">' +
                '<thead>' +
                '<tr>' +
                '<td><h3 class="text-danger">اسم المستشفى</h3></td>' +
                '<td><h3 class="text-danger">الموقع</h3></td>' +
                '<td><h3 class="text-danger">هاتف المستشفى</h3></td>' +
                '<td><h3 class="text-danger">قبول الانضمام</h3></td>' +
                '</thead>' +
                '<tbody>';
                var rows = '';
                for (var i = 0; i < data.result.length; i++) {
                    rows = rows + '<tr data-id='+data.result[i].id+'>' +
                        '<td>' +
                        data.result[i].hospitalName +
                        '</td>' +
                        '<td>' +
                        data.result[i].hospitalLocation +
                        '</td>' +
                        '<td>' +
                        data.result[i].hospitalPhone +
                        '</td>' +
                        '<td>' +
                        '<button onclick="acceptUser('+ data.result[i].id +')">قبول</button>'+
                        '</td>' +
                        '</tr>';

                }
                content = content + rows +
                    '</tbody>' +
                    '</thead>';
                $('#admin_login_form')[0].innerHTML = '<a onclick="logoutAdmin()">تسجيل الخروج</a>'
                +'<h3 class="text-info text-center">المستخدمين الجدد</h3>' + content;
                })

                //    openDashboard();
            } else {
                // alert('فشلت عملية تسجيل الدخول،الإيميل أو كلمة المرور خطأ.');
                document.getElementById("loginErr").innerHTML = "<p>*الإيميل أو كلمة المرور خطأ</p>"
            }
        });
}
```

Figure 4.28: Function of loginAdmin

In the route "/ loginAdmin", and access the "userData" array that loads the administrator information and decrypts them. The system selects the administrator's own database through this command (SELECT * FROM admins …), so that the information is compared with the information stored in this admins table rule as shown in Figure 4.29:

```
router.get('/loginAdmin', function (req, res, next) {
    var userData = {
        email: req.query.email,
        password: req.query.password
    }
      console.log("**********************");
      console.log(userData);
    db.query("SELECT * FROM admins WHERE email = '" + userData.email + "' AND password = '" + userData.password + "'",
    function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error user data"
            })
        } else {
            console.log(result[0].id);
            var token = jwt.sign({
                uid: result[0].id
            },
                '94Tre500*_q+esltwert#04*0$998rwertwebrbr5', {
                    expiresIn: 60 * 60 * 24 * 7 //expires in 1 week
                });

            res.json({
                "success": true,
                "result": {
                    "token": token
                }
            })
        }
    });

});
```

Figure 4.29: Route of loginAdmin

Through the administrator protect the system and approve the new subscribers of the system is done through the route "/getNewUsers" that is shared before entering the information system this subscriber also sent to this route as shown in figure 4.30:

```
router.get('/getNewUsers', function (req, res, next) {

    console.log(req.decoded);
    db.query("SELECT id,hospitalName,hospitalPhone,hospitalLocation,hospitalImage  FROM users WHERE isActive = false",
    function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error)
            res.json({
                "success": false,
                "result": "error get Hospitals"
            })
        } else {
            res.json({
                "success": true,
                "result": result
            })
        }
    });

});
```

Figure 4.30: Route of new user

The supervisor approves the subscriber by pressing the Accept button upon arrival of the subscriber's information as shown in the figure 4.31. The function "acceptUser()" through which the participant's information is sent after the approval of route "user/accpetUser" is executed as shown in figure4.32:

```
function acceptUser(id){
    var userData = {
        token: localStorage.getItem('bb_admin_token'),
        userId: id
    }
    $.get(serverAddress + 'users/acceptUser', userData, function (data) {
        console.log(data);
        if(data.success){
            alert("تمت الموافقة على الطلب");

            $("[data-id="+id+"]").css("display", "none");
        }else{
            alert("لم تتم العملية، الرجاء المحاولة لاحقاً");

        }
    })
}
```

Figure 4.31: Accept user

```
router.get('/acceptUser', function (req, res, next) {
    var userData = {
        id: parseInt(req.query.userId)
    }

    db.query("UPDATE users SET isActive = true WHERE  id = '" + userData.id + "'", function (error, results, fields) {
        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error editing data"
            })
        } else {
            res.json({
                "success": true,
                "result": "success editing data"
            })
        }
    });


});
```

Figure 4.32: Route of accept user

4.6 Security

In any system or Web site, it must be secure and protected from any breaches and from the users who do not need them to turn to this system.

- **Token**

  Is also a system protection system known to discriminate users by giving each user an encrypted number before entering the system. In the route "/loginuser", this encrypted number is generated as shown in figure 4.33:

```
router.get('/loginUser', function (req, res, next) {
    var userData = {
        email: req.query.email,
        password: req.query.password
    }
    //   console.log(userData);
    db.query("SELECT * FROM users WHERE email = '" + userData.email + "' AND password = '" + userData.password +
     "' AND isActive = true", function (error, result) {
        if (error || (result.length == 0)) {
            console.log(result);
            console.log(error);
            res.json({
                "success": false,
                "result": "error user data"
            })
        } else {
            console.log(result[0].id);
            var token = jwt.sign({
                uid: result[0].id
            },
                '94Tre500*_q+esltwert#04*0$998rwertwebrbr5', {
                    expiresIn: 60 * 60 * 24 * 7 //expires in 1 week
                });

            res.json({
                "success": true,
                "result": {
                    "token": token
                }
            })
        }
    });

});
```

Figure 4.33: Route of create token

- **Middleware**

  When the user takes a Token, he wants to do any operation in the system of adding a blood unit and the other is to know from this joint through this Token, which is decoded through this route “token” as shown in figure 4.34:

```
router.use(function (req, res, next) {

  var str = url.parse(req.url).pathname;

  if (str.substr(-1) != '/') {
    str += '/';
  }
  // console.log(str)

  if (str == "/users/createUser/" || str == "/users/loginUser/" || str == "/users/loginAdmin/") { // No token is needed here.
    console.log("aaaaaaaaaaaaaaaaaaaaaaa")
    next();
    return;
  }

  // check header or url parameters or post parameters for token5
  var token = req.body.token || req.query.token || req.headers['x-access-admin_token'];

  // decode token
  if (token) {
    // verifies secret and checks exp
    jwt.verify(token, '94Tre500*_q+esltwert#04*0$998rwertwebrbr5', function (err, decoded) {
      if (err) {
        return res.json({
          success: false,
          msgCode: "TOKEN_AUTH_FAILED",
          msgText: 'Failed to authenticate token.',
          msgText: 'خطأ! الرجاء تسجيل الدخول من جديد'
        });
      } else {
        // console.log("decoded.uid", decoded.uid)
        req.decoded = decoded;
        next();
      }
    });
```

Figure 4.34: Verify of the token

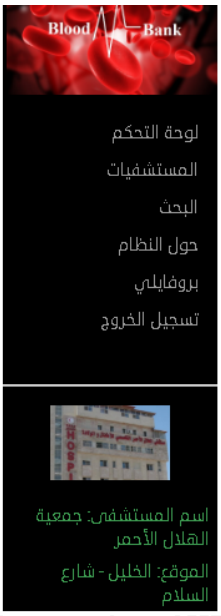
4.7 Home Page

The home page contains the main contents that the system needs examples of this: the dashboard, through which the blood units are added, the hospitals through which to view the units of each hospital on their own, the search for blood units, and others.

In addition, through HTML and CSS, this page is programmed and new elements are used, i.e., <a> element a link is converted to implement other functions, and implementation the home page at the user enter in the system as shown figure 4.35:

```html
<div id="sidebar-wrapper">
    <ul class="sidebar-nav">
        <li class="sidebar-brand">
            <img src="imgs/bloodbank.jpg" alt="" width="100%" style="margin-right: -25px">
        </li>
        <li>
            <a href="#" onclick="openDashboard()">لوحة التحكم</a>
        </li>
        <li>
            <a href="#" onclick="getHospitals()">المستشفيات</a>
        </li>
        <li>
            <a href="#" onclick="openSearchTab()">البحث</a>
        </li>
        <li>
            <a href="#" onclick="getAboutSystem()">حول النظام</a>
        </li>
        <li>
            <a href="#" onclick="openProfile()">بروفايلي</a>
        </li>
        <li>
            <a href="#" onclick="logoutUser()">تسجيل الخروج</a>
        </li>
        <div id="profileInfo"></div>
    </ul>
</div>
```

Code HTML



implementation of HTML

Figure 4.35: Home Page

4.7.1 Dashboard

Through the dashboard in the system, control of the blood units, the blood units are inserted, modified and deleted. The user clicks on the dashboard and executes the function "`openDashboard()`"as shown in figure 4.36 and implementation of the dashboard at the client side so the user executes of this process as shown figure 4.37:

```
function openDashboard() {
    $('#main_content')[0].innerHTML = '<h2 class="text-center">لوحة التحكم</h2>' +
        '<h4>أضف وحدات دم للمستشفى الخاص بك:</h4>' +
        ' <div class="col-md-6">' +
        '<form name="addBloodForm" role="form">' +
        '<div class="form-group">' +
        '<label class="text-success"><h4>فصيلة الدم</h4></label>' +
        '<select class="form-control" name="type">' +
        '<option value="A+">A+</option>' +
        '<option value="A-">A-</option>' +
        '<option value="B+">B+</option>' +
        '<option value="B-">B-</option>' +
        '<option value="O+">O+</option>' +
        '<option value="O-">O-</option>' +
        '<option value="AB+">AB+</option>' +
        '<option value="AB-">AB-</option>' +
        '</select>' +
        '</div>' +
        '<div class="form-group">' +
        '<label class="text-success" ><h4 style="margin-top: 0">الكمية</h4></label>' +
        '<input type="number" min="1" name="quantity" class="form-control">' +
        '</div>' +
        '<div id="addBloodErr"></div>' +
        '<div class="form-group">' +
        '<input class="btn btn-success" type="button" onclick="addBloodUnit()" value="أضف">' +
        '</div>' +
        '</form>' +
        '<h3 style="margin-top: 80px" class="text-info">وحدات الدم المتوفرة في المستشفى الخاص بك:</h3>' +
        '<div id="hospital_content"></div>' +
        '</div>';

    getBloodUnitsByHospitalId(null);
    getMyProfile();
}
```

Figure 4.36: Function of openDashboard

Figure 4.37: Implementation the Dashboard

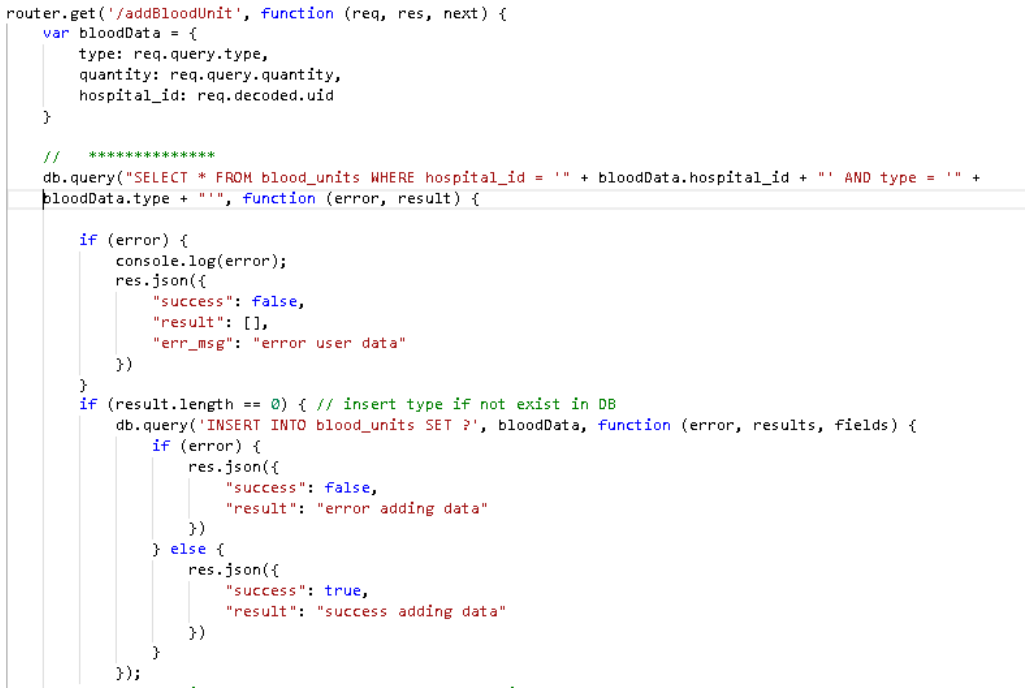
- **Insert blood units**

When a hospital needs to enter a blood unit or increase the amount of a specific blood unit, this needs access to the database for each hospital and is accessed when the user press the "Add button" to perform the function "addBloodUnit()" as in Figure 4.38. Through this function, the unit of blood is taken to the quantity of the unit and its quantity and added in the matrix "bloodData", and then sent to the path of the "users/addBloodUnit" until the unit is added and quantity in the database for each hospital.

```
function addBloodUnit() {
    var addBloodForm = document.forms["addBloodForm"];
    document.getElementById("addBloodErr").innerHTML = '';
    if (addBloodForm["quantity"].value == "") {
        document.getElementById("addBloodErr").innerHTML = "<p>* الكمية مطلوب</p>";
        return true;
    }

    var token = localStorage.getItem('bb_token');
    var bloodData = {
        token: token,
        type: addBloodForm["type"].value,
        quantity: addBloodForm["quantity"].value
    }

    // return true;

    $.get(serverAddress + 'users/addBloodUnit', bloodData, function (data) {
        console.log(data);
        if (data.success) {
            openDashboard();
        } else {
            // alert("Failed to add blood unit, try again later.");
        }
    });
}
```

Figure 4.38: Code Insert blood units

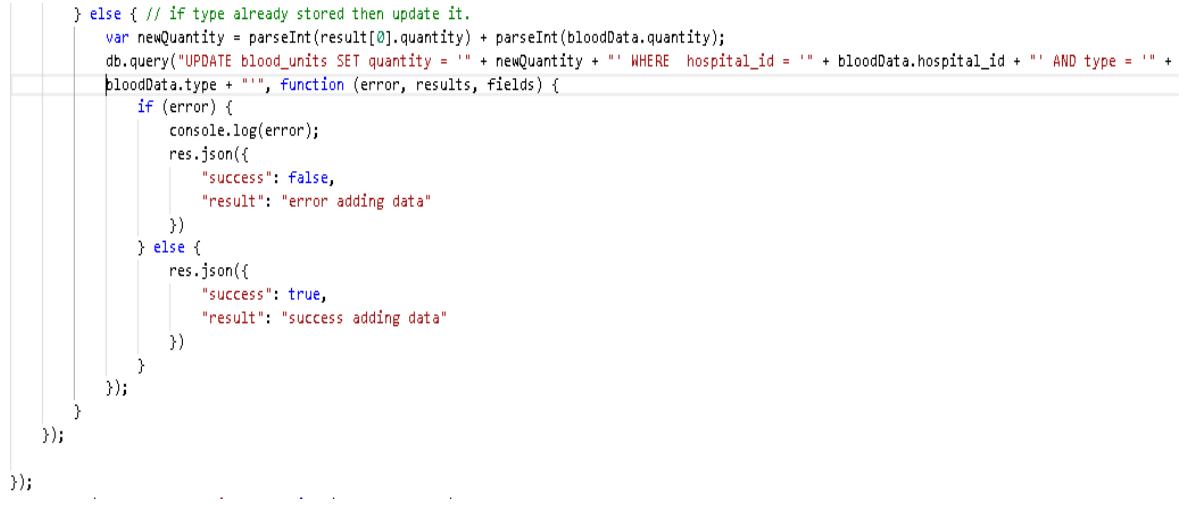
In the route of adding the blood unit at the arrival of the matrix “bloodData” containing the information to the server side is encrypted and decryption and extract “hospital_id” of the hospital that wants to add the blood unit as shown in figure 4.39:



```
router.get('/addBloodUnit', function (req, res, next) {
    var bloodData = {
        type: req.query.type,
        quantity: req.query.quantity,
        hospital_id: req.decoded.uid
    }

    //   **************
    db.query("SELECT * FROM blood_units WHERE hospital_id = '" + bloodData.hospital_id + "' AND type = '" +
    bloodData.type + "'", function (error, result) {

        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": [],
                "err_msg": "error user data"
            })
        }
        if (result.length == 0) { // insert type if not exist in DB
            db.query('INSERT INTO blood_units SET ?', bloodData, function (error, results, fields) {
                if (error) {
                    res.json({
                        "success": false,
                        "result": "error adding data"
                    })
                } else {
                    res.json({
                        "success": true,
                        "result": "success adding data"
                    })
                }
            });
```

Figure 4.39: Route of add blood units

As mentioned earlier can also increase the amount of blood added by the user through this page as shown in figure 4.40, and figure 4.41 shown implementation the insert blood units.



```
        } else { // if type already stored then update it.
            var newQuantity = parseInt(result[0].quantity) + parseInt(bloodData.quantity);
            db.query("UPDATE blood_units SET quantity = '" + newQuantity + "' WHERE  hospital_id = '" + bloodData.hospital_id + "' AND type = '" +
            bloodData.type + "'", function (error, results, fields) {
                if (error) {
                    console.log(error);
                    res.json({
                        "success": false,
                        "result": "error adding data"
                    })
                } else {
                    res.json({
                        "success": true,
                        "result": "success adding data"
                    })
                }
            });
        }
    });

});
```

Figure 4.40: Increases quantity of the blood units

Figure 4.41: Implementation of insert blood units

- **Modify of the blood units**

When adjusting the amount of blood unit by pressing the modify button, then a certain function "editBloodUnit(bloodUnitId, bloodUnitQuantity)" is performed, the system takes the value that it wants to modify and adds it in the matrix "bloodUnitData", and then sends it to the adjustment route of a "users/updateBloodUnit" until the amount of blood unit specified in the database is adjusted as shown in figure 4.42:

```
function editBloodUnit(bloodUnitId, bloodUnitQuantity) {
    console.log(bloodUnitId);
    console.log(bloodUnitQuantity);
    var newVal = prompt("Edit Quantity", bloodUnitQuantity);

    if (newVal != null) {
        var token = localStorage.getItem('bb_token');
        var bloodUnitData = {
            token: token,
            bloodUnitId: bloodUnitId,
            quantity: newVal
        }
        $.get(serverAddress + 'users/updateBloodUnit', bloodUnitData, function (data) {
            if (data.success) {
                openDashboard();
            } else {
                // alert('Failed to remove item please try again later!!');
                console.log("failed to edit !!");
            }
        });
    }
}
```

Figure 4.42: Modifying

In the route "/updateBloodUnit" of an adjustment to the quantity of the specific blood unit, the hospital's database is accessed by decoding the matrix "bloodData" and extracting the private "hospital_id of the hospital and taking the required laboratory as shown figure 4.43:

```
router.get('/updateBloodUnit', function (req, res, next) {
    var bloodData = {
        id: parseInt(req.query.bloodUnitId),
        quantity: parseInt(req.query.quantity),
        hospital_id: req.decoded.uid
    }

    db.query("UPDATE blood_units SET quantity = '" + bloodData.quantity + "' WHERE  hospital_id = '" + bloodData.hospital_id + "' AND id = '" +
    bloodData.id + "'", function (error, results, fields) {
        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error editing data"
            })
        } else {
            res.json({
                "success": true,
                "result": "success editing data"
            })
        }
    });


});
```

Figure 4.43: Update on blood units

- **Delete blood units**

In the deletion when you press the delete button and delete function "deleteBloodUnit(id)" performs as shown in the figure 4.44, through the code is taken to the desired blood unit be deleted and put them in a matrix "bloodUnitData" and sent to the route of "users/deleteBloodUnit" until access to the database.

```
function editBloodUnit(bloodUnitId, bloodUnitQuantity) {
    console.log(bloodUnitId);
    console.log(bloodUnitQuantity);
    var newVal = prompt("Edit Quantity", bloodUnitQuantity);

    if (newVal != null) {
        var token = localStorage.getItem('bb_token');
        var bloodUnitData = {
            token: token,
            bloodUnitId: bloodUnitId,
            quantity: newVal
        }
        $.get(serverAddress + 'users/updateBloodUnit', bloodUnitData, function (data) {
            if (data.success) {
                openDashboard();
            } else {
                // alert('Failed to remove item please try again later!!');
                console.log("failed to edit !!");
            }
        });
    }
}
```

Figure 4.44: Delete the blood units

In the deletion route "/deleteBloodUnit" is the access of the matrix "bloodData" to the encrypted server is decrypted and extract your "hospital_id" of the hospital with the blood unit that is deleted as in the figure 4.45, then the information that has been added, modified and deleted is displayed when the client as shown in figure 4.46:

```
router.get('/deleteBloodUnit', function (req, res, next) {
    var bloodData = {
        bloodUnitId: parseInt(req.query.bloodUnitId),
        hospital_id: req.decoded.uid
    }

    db.query('DELETE FROM blood_units WHERE id = ' + bloodData.bloodUnitId + ' AND hospital_id=' +
    bloodData.hospital_id, function (error, results, fields) {
        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error removing blood unit"
            })
        } else {
            res.json({
                "success": true,
                "result": "success removing blood unit"
            })
        }
    });

});
```

Figure 4.45: Route of the delete blood units

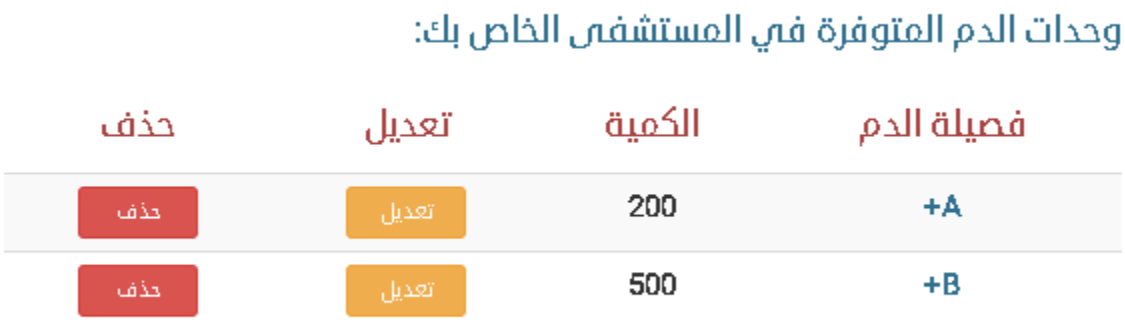


Figure 4.46: Update Page

4.7.2 Get Hospitals

On this page, all hospitals participating in this system appear, when the pressure on the hospitals is performed the function "getHospitals()" by which the call to the store Token in the "Local Storage" and send it to the route "user/getHospitals" when the access to the barrier is decoding of the Token as shown in figure 4.47:

```
function getHospitals() {
    $('#main_content')[0].innerHTML = '<h3>جاري التحميل . . .</h3>';

    var token = localStorage.getItem('bb_token');
    var authData = {
        token: token
    }
    $.get(serverAddress + 'users/getHospitals', authData, function (data) {

        if (data.result) {
            var content = "";
            for (var i = 0; i < data.result.length; i++) {

                if (data.result[i].hospitalLocation) {

                    var temp = '<div class="col-sm-3 col-xs-6">' +
                        '<div class="thumbnail" onclick="showHospitalById(' + data.result[i].id + ')">' +
                        '<div class="img_box" '+
                        'style="background-image:url(\'' + window.serverAddress + (data.result[i].hospitalImage || 'hospital.jpg') + '\')" ></div>' +
                        '<div class="caption">' +
                        '<h3>' + data.result[i].hospitalName + '</h3>' +
                        '<p>' + data.result[i].hospitalLocation + '</p>' +
                        '</div>' +
                        '</div>' +
                        '</div>';
                    content = temp + content;
                }
            }
            console.log(content);
            $('#main_content')[0].innerHTML = content;
        }
    });
}
```

Figure 4.47: Get hospitals

A route "/getHospitals" through which information from other hospitals is brought and displayed on the hospital page as shown in figure 4.48. Moreover, execute the page in the client side as shown in figure 4.49:

```
router.get('/getHospitals', function (req, res, next) {

    console.log(req.decoded);
    db.query("SELECT id,hospitalName,hospitalPhone,hospitalLocation,hospitalImage  FROM users",
    function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error)
            res.json({
                "success": false,
                "result": "error get Hospitals"
            })
        } else {
            res.json({
                "success": true,
                "result": result
            })
        }
    });

});
```

Figure 4.48: Route the get-hospital

Figure 4.49: Implementation of the get-hospital

4.7.3 Search

In the search, process is to facilitate access to hospitals and blood units, when you press the search button is performed a function “openSearchTab()” before adding the information, you want to search as shown in the figure 4.50, and in this function can implement by any browser as shown un the figure 4.51:

```
function openSearchTab() {
    var content =
        '<h4 class="jumbotron">يمكنك البحث عن أي مستشفى من خلال الإسم، وأيضاً يمكنك البحث عن وحدات الدم المتوفرة في جميع المستشفيات أو في مستشفى بعينه. </h4>' +
        ' <div class="col-md-6">' +
        '<form name="searchform" role="form">' +
        '<div class="form-group">' +
        '<label class="text-success" ><h4 style="margin-top: 0">اسم المستشفى</h4></label>' +
        '<input type="text"  name="hospitalName" class="form-control">' +
        '</div>' +
        '<div class="form-group">' +
        '<label class="text-success"><h4>فصيلة الدم</h4></label>' +
        '<select class="form-control" name="bloodType">' +
        '<option value=""></option>' +
        '<option value="A+">A+</option>' +
        '<option value="A-">A-</option>' +
        '<option value="B+">B+</option>' +
        '<option value="B-">B-</option>' +
        '<option value="O+">O+</option>' +
        '<option value="O-">O-</option>' +
        '<option value="AB+">AB+</option>' +
        '<option value="AB-">AB-</option>' +
        '</select>' +
        '</div>' +

        '<div class="text-danger" id="searchErr"></div>' +
        '<div class="form-group">' +
        '<input class="btn btn-success" type="button" onclick="searchBackend()" value="ابحث">' +
        '</div>' +

        '</form></div><div class="col-md-12">' +
        '<div style="margin-top: 50px"><hr/><div id="search_result"></div></div></div>';
    $('#main_content')[0].innerHTML = content;

}
```

Figure 4.50: Code HTML of the Search

Figure 4.51: Implementation of the search page

When a hospital searches for a blood unit or another hospital, the appropriate fields are filled in to search and a search button is executed a function “searchBackend()” is performed by which the information is stored and placed in a matrix ”bloodData” and sent to a route “user/search” as shown in figure 4.52:

```
function searchBackend() {
    $('#search_result')[0].innerHTML = '';


    var searchForm = document.forms["searchForm"];
    document.getElementById("searchErr").innerHTML = '';
    if (searchForm["hospitalName"].value == "" && searchForm["bloodType"].value == "") {
        document.getElementById("searchErr").innerHTML =
        "<p>* قم بإدخال اسم المستشفى أو وحدة الدم أو كليهما.</p>";
        return true;
    }


    var token = localStorage.getItem('bb_token');
    var bloodData = {
        token: token
    }
    if (searchForm["bloodType"].value != "") {
        bloodData.bloodType = searchForm["bloodType"].value;
    }
    if (searchForm["hospitalName"].value != "") {
        bloodData.hospitalName = searchForm["hospitalName"].value;
    }

    console.log(bloodData);

    //  return true;

    $.get(serverAddress +'users/search', bloodData, function (data) {
        console.log(data);
        if (data.success && (data.result.length > 0)) {
            if (data.result[0].hospitalName && !data.result[0].bloodType) {
                var content = "";
                for (var i = 0; i < data.result.length; i++) {
```

Figure 4.52: Function of the Search

In the route "Search", when the information you are looking for is accessed, the file is decoded and extracted from the person who searched “searchData”, and what is searched for is either a hospital by which the system by this command (SELECT id, hospitalName, hospitalPhone, hospitalLocation, hospitalImage FROM users…) compares the information

with information to be searched as shown in Figure 4.53. On the other hand, blood units and hospitals can be searched together by this command as shown in Figure 4.54:

```
router.get('/search', function (req, res, next) {
    var searchData = {
        hospital_id: req.decoded.uid,
        hospitalName: req.query.hospitalName,
        bloodType: req.query.bloodType
    }
    // bloodType: req.query.bloodType

    if (searchData.hospitalName && !searchData.bloodType) {

        db.query("SELECT id,hospitalName,hospitalPhone,hospitalLocation,hospitalImage FROM users WHERE (hospitalName LIKE '%" +
        searchData.hospitalName + "%')", function (error, result) {
            if (error) {
                console.log(error);
                res.json({
                    "success": false,
                    "result": "Filed to search term!!"
                })
            } else {
                res.json({
                    "success": true,
                    "result": result
                })
            }
        });
```

Figure 4.53: Route of the Search on hospitals

```
} else if (searchData.hospitalName && searchData.bloodType) {

    db.query("SELECT id,hospitalName,hospitalPhone,hospitalLocation,hospitalImage FROM users WHERE (hospitalName LIKE '%" +
    searchData.hospitalName + "%')", function (error, hospitalResult) {
        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": "Filed to search term!!"
            })
        } else {

            if (hospitalResult.length > 0) {
                db.query("SELECT * FROM blood_units WHERE hospital_id = '" + hospitalResult[0].id + "' AND type = '" +
                searchData.bloodType + "'", function (error, bloodResult) {
                    if (error) {
                        console.log(error);
                        res.json({
                            "success": false,
                            "result": "Filed to search term !!"
                        })
                    } else {
                        if (bloodResult.length > 0) {
                            res.json({
                                "success": true,
                                "result": [{
                                    "hospitalName": hospitalResult[0].hospitalName,
                                    "hospitalPhone": hospitalResult[0].hospitalPhone,
                                    "bloodType": bloodResult[0].type,
                                    "quantity": bloodResult[0].quantity
                                }]
                            })
```

Figure 4.54: Search on blood units and hospitals

4.8 Steps for upload the system on the heroku server

To determine how to start your app, Heroku first looks for a Procfile. If no Procfile exists for a Node.js app, we will attempt to start a default web process via the start script in your package.json. The command in a web process type must bind to the port number specified in the PORT environment variable. If it does not, the Node.js will not start. For more information, see Best Practices for Node.js Development and Heroku Node.js Support.

- **Build your system and run it locally**

  Run the npm install command in your local app directory to install the dependencies that you declared in your package.json file as shown in the figure 4.55, start your system locally using the heroku local command, which is installed as part of the Heroku as shown figure 4.56, and the system should now be running on http://localhost:3000/.

```
$ npm install
```

Figure 4.55: Run of the system

```
$ heroku local web
```

Figure 4.56: Local web

- **Deploy the system on Heroku**

  After you commit your changes to git, you can deploy system on the Heroku, to open the app in your browser, type heroku open as shown in the figure 4.57.

```
$ git add .
$ git commit -m "Added a Procfile."
$ heroku login
Enter your Heroku credentials.
...
$ heroku create
Creating arcane-lowlands-8408... done, stack is cedar
http://arcane-lowlands-8408.herokuapp.com/ | git@heroku.com:arcane-lowlands-8408.git
Git remote heroku added
$ git push heroku master
...
-----> Node.js app detected
...
-----> Launching... done
       http://arcane-lowlands-8408.herokuapp.com deployed to Heroku
```

Figure 4.57: Create heroku

By default, Heroku will install all dependencies listed in package.json under dependencies and devDependencies, after running the installation and build steps Heroku will strip out the, packages declared under devDependencies before deploying the application. You can direct Heroku to only install dependencies by setting environment variables NPM_CONFIG_PRODUCTION=true or YARN_PRODUCTION=true as shown in the figure 4.58:

```
$ heroku config:set NPM_CONFIG_PRODUCTION=true YARN_PRODUCTION=true
```

Figure 4.58: install packages

- **Customizing the build process**

While Node.js has standard preinstall and post install scripts, sometimes you may want to run scripts only before or after builds on Heroku. For instance, you may need to configure npm, git, or Ssh before Heroku installs dependencies. On the other hand, you may need to build production assets after dependencies are installed.

If the system has a build step that you would like to run when you deploy, you can use a postinstall script in package.json as the figure 4.59:

```
"scripts": {
  "start": "node index.js",
  "test": "mocha",
  "postinstall": "bower install && grunt build"
}
```

Figure 4.59: Packages.json

Heroku stores the node modules and bower components directories by default. You can override these defaults by providing a cacheDirectories array in your top-level package.json. For example, if you build inside client and server sub-directories as shown in figure 4.60:

```
"cacheDirectories": ["client/node_modules", "server/node_modules"]

"cacheDirectories": ["data"]
```

Figure 4.60: Store the system

- **Runtime behavior**

The buildpack puts node, npm, and node_modules/.bin on the PATH so they can be executed with heroku run or used directly in a Procfile, The NODE_ENV environment variable is set to 'production' by default, but you can set it to any arbitrary string as shown in the figure 4.61:

```
$ cat Procfile
web: npm start

$ heroku config:set NODE_ENV=staging
```

Figure 4.61: Runtime behavior of the system

- **Blood bank Site**

  When the system is completed deploy on the server heroku, it gives it a domain so users can access it. The domain is the system's own name. The system contains the name of this system with the server name (https://bloodbanksys.herokuapp.com), as it is in the figure 4.62.

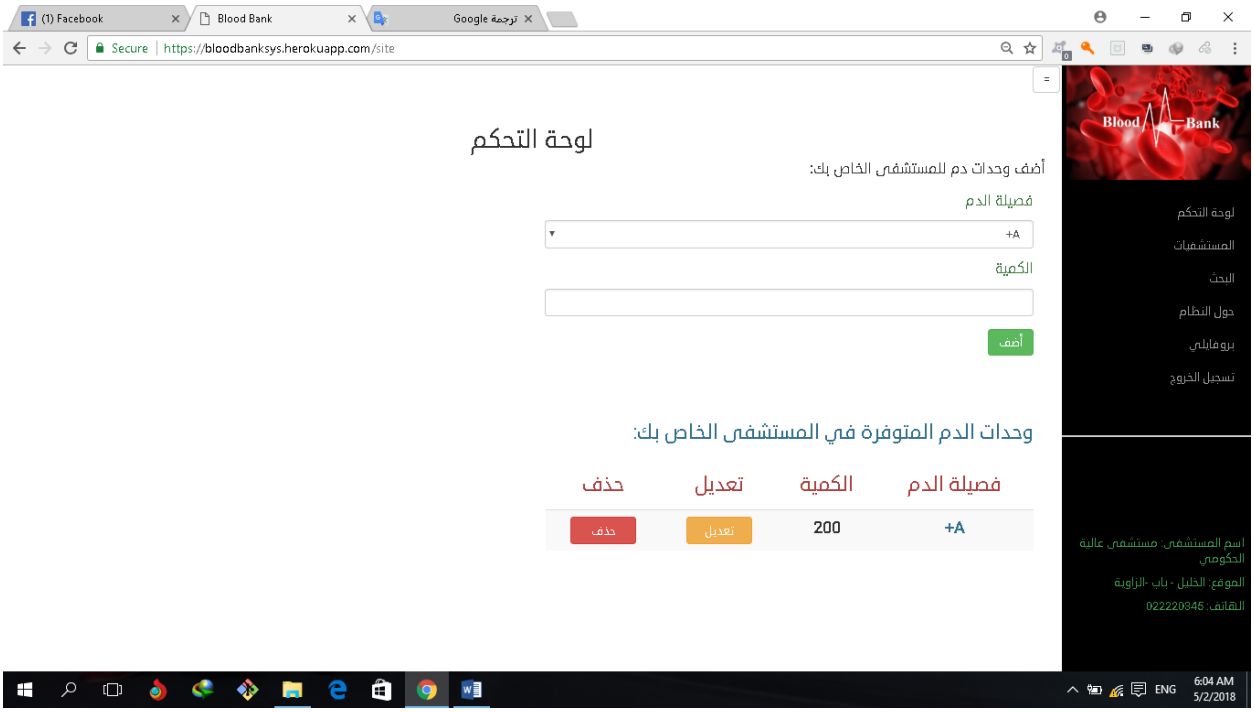


Figure 4.62: blood bank system

5

Chapter Five Implementation Design

5.1 Introduction

5.2 Testing of a Network

5.3 Testing of Client is connected to the server

5.4 Testing and result of enter the system

5.5 The result of the database

5.6 Recommendations

5.1 Introduction

The final chapter to complete the system is to test of the system to get results and performance of this system.

5.2 Testing of a Network

As previously mentioned in the section that networks will be integrated into two lines from the internet the first line in the local network of the hospital and the second-line Internet service via mobile communications via GSM- modem.

Connecting a router that receives the Internet through the LAN-line to the computer and then add the slide in the GSM-modem card is also linked to the computer so we get two networks from the Internet, as in the figure 5.1:

C 

Figure 5.1: Connection network

When the system became connected to the Internet through the first two networks on the Ethernet and the other through the wireless network, these networks were integrated by the Connectify dispatch and the result was that the system is connected to the Internet through the local network connected to Ethernet show in the figure 5.2:

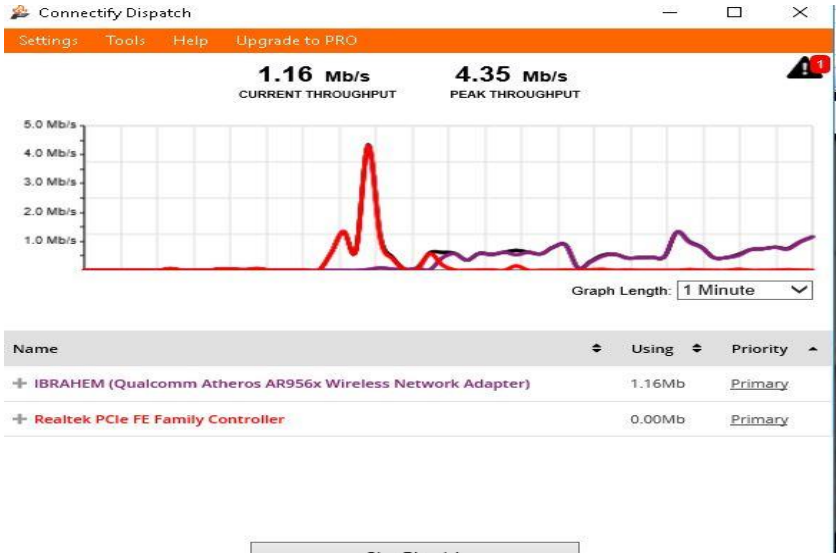


Figure 5.2: Connect two-internet network

We then disconnect the LAN system was the result that the Connectify dispatch converts Internet connection to the network wireless connected through GSM modem, which provides the Internet via cellular communications as shown in the figure 5.3:

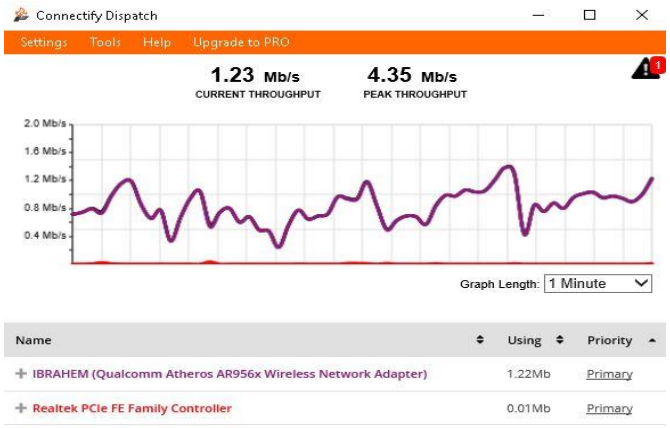


Figure 5.3: Cut of the LAN-Network

When it returned the online Connectify dispatch result was converted Internet connection from the wireless network to LAN-Network as shown in the figure 5.4:

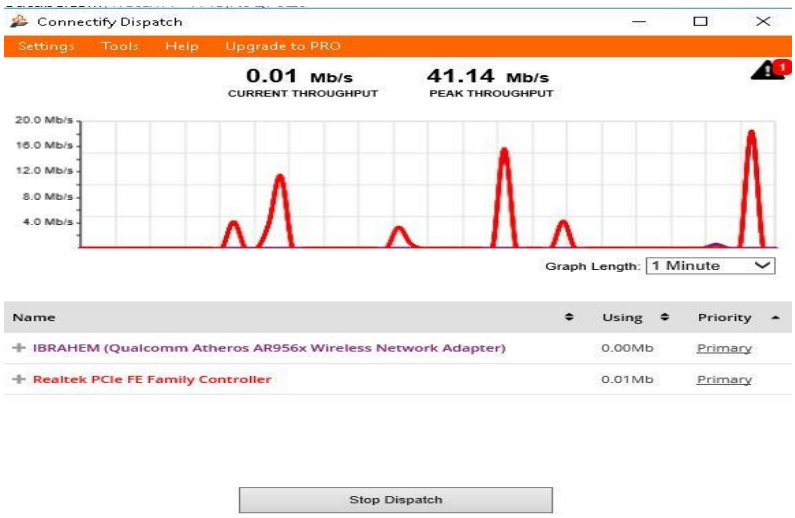
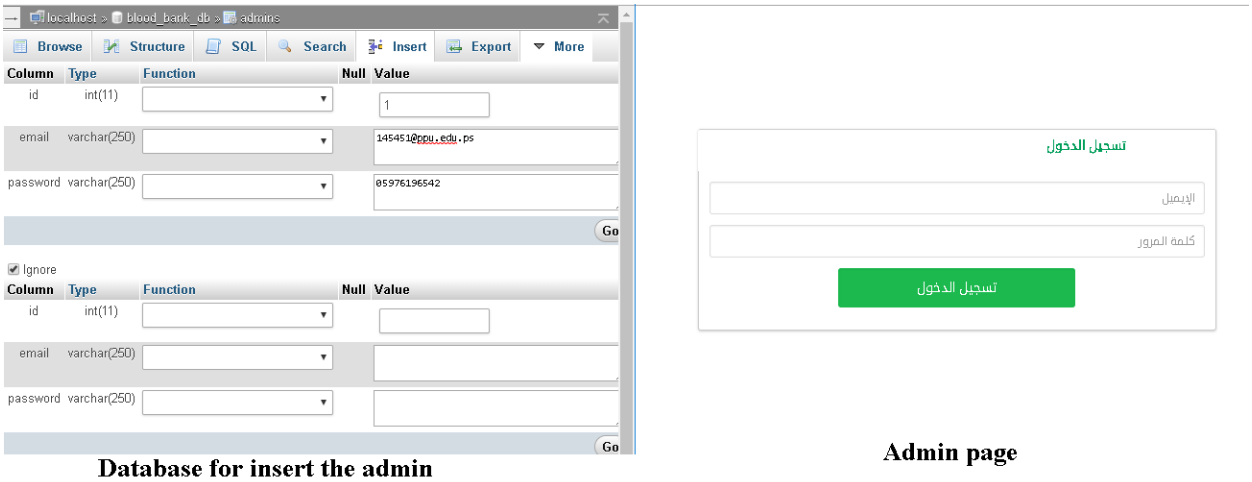


Figure 5.4: Return of the internet

5.3 Testing of Client is connected to the server

The server has needed admin for control enter process into the server, an admin is created through the admin own database in the system and logged on to the system through a special interface as shown in the figure 5.5:



Database for insert the admin

Admin page

Figure 5.5: Admin for the system

When the admin enters the system, the system allows the participants to enter the system and the subscribers are canceled as shown in the figure 5.6:

Figure 5.6: Accepting page

The client (hospitals) when connects with the server can see that other hospitals and show of the special blood units each hospital. When the hospitals create, account for the system should filling of the fields at the signup page as shown in the figure 5.7:

Figure 5.7: Create account

At the end of the hospital account creation is not access to the system only by accepting the admin for security reasons and other reasons, as shown in figure 5.8:

Figure 5.8: Accept of the hospital

When the admin approves the entry of the participant to the system, it is permissible to add special blood units to him through special interfaces for addition, and there are interfaces to see the blood units of other hospitals participating in this system and facilitates the search for any blood unit in these hospitals through the search interface:

5.4 Testing and result of enter the system

- **Home page**

  Is a page containing the main pages as we mentioned earlier, the most important is the dashboard page, through which any hospital can add special blood units and can be modified as you like cannot be modified to the special blood units of another hospital, when added is also added to the database for the system, can know the hospital that adding the blood unit through the number of special id as shown in figure 5.9:

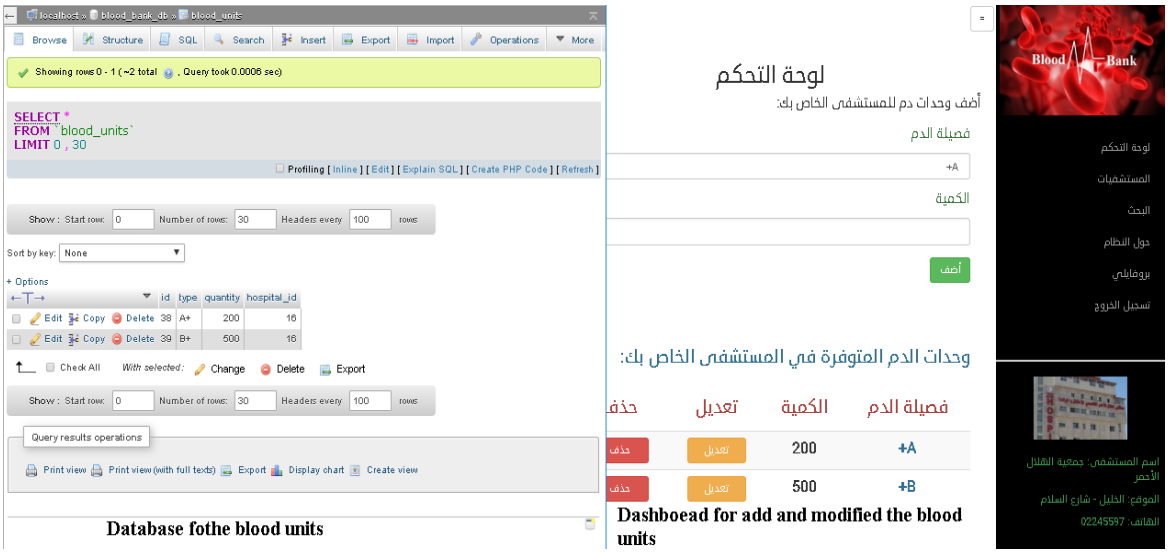


Figure 5.9: Home and dashboard page

- **Hospitals page**

On this page, when each hospital has added its own units, through this page you can see other hospitals participating in the same system and see the blood units of the participating hospital. This unit can be requested by calling him on the phone numbers of the hospital as shown in the figure 5.10:

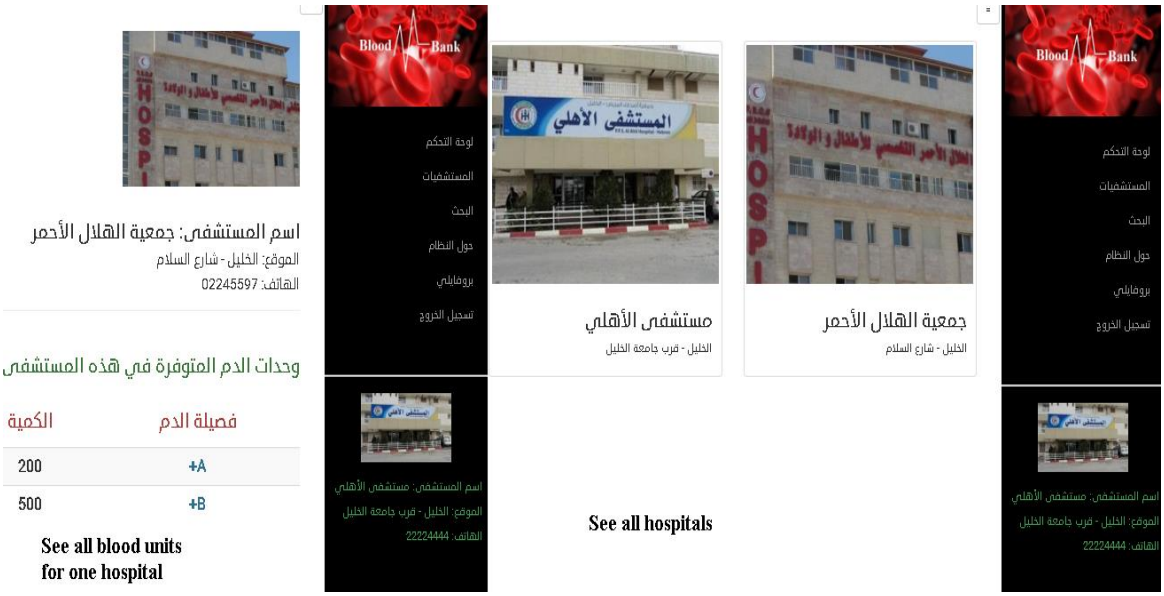


Figure 5.10: Hospitals page

- **Search page**

This search for the process of blood units, can search through hospitals only then see blood units, and can search for blood units only without the need to search for hospitals, are considered only to facilitate the search process as shown in the figure 5.11:

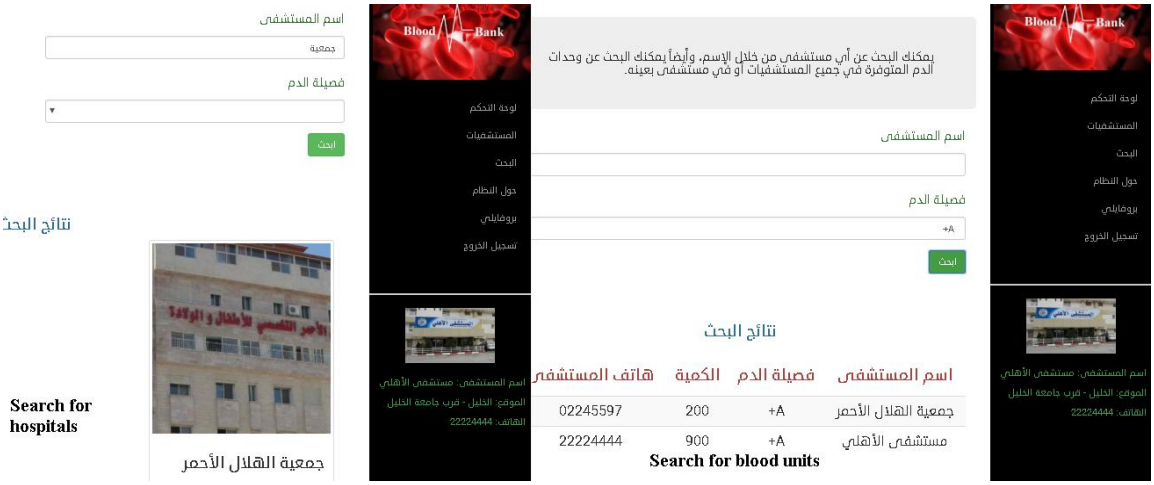


Figure 5.11: Search Page

- **Profile page**

On this page the hospital's special things change, for example the phone number, the hospital location, and the name of the hospital can be changed to this we have created this page as shown in figure 5.12:

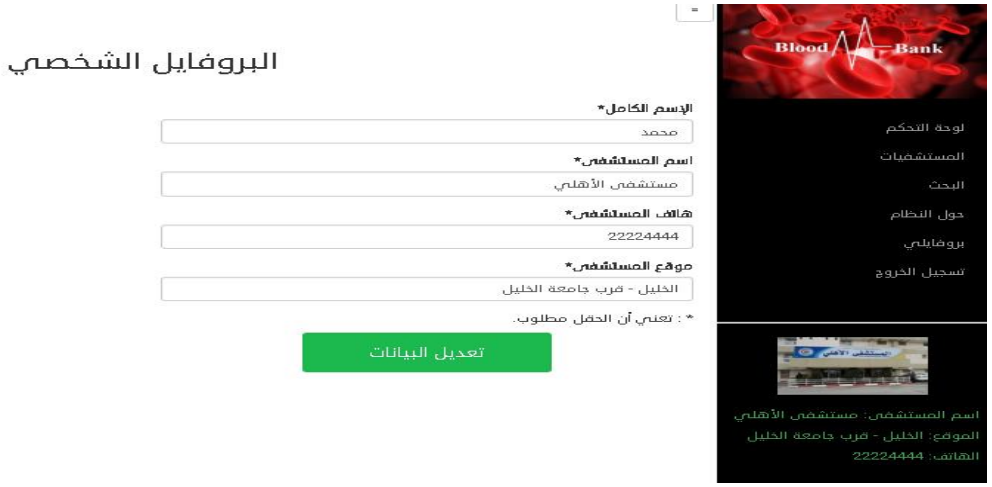


Figure 5.12: Profile page

5.5 The result of the database

The system database contains several tables including admins, users, and blood units. Can be filled out as follows:

- **Admin**

  In the admin table, it contains an id through which the admin knows if the same system is more than admin, contains E-mail for special admin, and password. For example is shown in the table 5.1, in the system, we added the admin through the private database directly to the phpmyadmin page as shown in the figure 5.13:

Id	1
E-mail	Ibrahimjundi0597@gmail.com
Password	0597619621

Table 5.1: Admin table

Figure 5.13: Database for admin

- **Users (hospitals)**

  In the users table, it contains id from the id it through which hospitals are distinguished from each other, full name is the name of user or employ of the hospital, e-mail for special the user and password, and it contains hospital phone, hospital location, hospital image, and is active. For example as shown in the table 5.2, the result of the database in the system as shown in the figure 5.14:

id	fullName	E-mail	password	hospitalName	hospitalPhone	hospitalLocation	hospitalImage	isActive
15	محمد	mmohammadratib@gmail.com	12345	مستشفى الأهلي	22224444	الخليل - قرب جامعة الخليل	jpg,اهلي2627	1
16	ibrahim	145451@ppu.edu.ps	597619621	جمعية الهلال الأحمر	2245597	الخليل- شارع السلام	5352cro.jpg	1
17	هديل	hadeel.ayaydeh@yahoo.com	112233	مستشفى عالية الحكومي	2256854	الخليل - باب الزاوية	jpg,عالية869	1

Table 5.2: Users table

id	fullName	email	password	hospitalName	hospitalPhone	hospitalLocation	hospitalImage	isActive
15	محمد	mmohammadratib@gmail.com	12345	مستشفى الأهلي	22224444	الخليل - قرب جامعة الخليل	2627اهلي.jpg	1
16	ibrahim	145451@ppu.edu.ps	0597619621	جمعية الهلال الأحمر	02245597	الخليل - شارع السلام	5352cro.jpg	1
17	هديل	hadeel.ayaydeh@yahoo.com	112233	مستشفى عالية الحكومي	02265489	الخليل - باب -الزاوية	869عالية.jpg	1

Figure 5.14: Database for users

- **Blood units**

In this table contains the type and quantity of blood units, the hospital_id units also has the same as the ID in the users' table until its own blood units are known, and it contains id is special for each blood. The result as shown table 5.3, and figure 5.15.

id	type	quantity	hospital_id
38	A+	200	16
39	B+	500	16
40	A+	900	15
41	A-	300	15
42	B+	500	15
43	A+	500	17
44	B+	300	17

Table 5.2: blood units table

	id	type	quantity	hospital_id
Edit Copy Delete	38	A+	200	16
Edit Copy Delete	39	B+	500	16
Edit Copy Delete	40	A+	900	15
Edit Copy Delete	41	A-	300	15
Edit Copy Delete	42	B+	500	15
Edit Copy Delete	43	A+	500	17
Edit Copy Delete	44	B+	300	17

Figure 5.15: Database for blood units

5.6 Recommendations

After completing the design of the system, uploading it online and testing it, the recommendations it needs in the future are:

- The first recommendation is to establish a special database of telephone numbers for people in general, and their names and blood type, when added to this system through which the work of donations to blood messages ae sent to tell them about these donations.
- That the system is convenient when you open it on smart phones, the recommendations can be made to the system Android application has a special for the system.

References

[1] A. C. Adsul and V. Bhosale, "Automated Blood Bank System using Raspberry Pi," 2017.

[2] W. M. Zintel, "Auto-configuring of peripheral on host/peripheral computing platform with peer networking-to-host/peripheral adapter for peer networking connectivity," ed: Google Patents, 2004.

[3] H. R. Tan, C. Lee, and V. Mok, "Automatic power meter reading system using GSM network," in *Power Engineering Conference, 2007. IPEC 2007. International*, 2007, pp. 465-469.

[4] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," in *ACM SIGCOMM computer communication review*, 1993, pp. 183-193.

[5] M. Castells, *The Internet galaxy: Reflections on the Internet, business, and society*: Oxford University Press on Demand, 2002.

[6] J. E. Allard, D. R. Treadwell III, and J. F. Ludeman, "Method, system and apparatus for client-side usage tracking of information server systems," ed: Google Patents, 2000.

[7] D. Raggett, A. Le Hors, and I. Jacobs, "HTML 4.01 Specification," *W3C recommendation,* vol. 24, 1999.

[8] D. Flanagan, *JavaScript: the definitive guide*: " O'Reilly Media, Inc.", 2006.

[9] D. S. Jorgenson, "System and method for maintaining consistent independent server-side state among collaborating servers," ed: Google Patents, 2006.

[10] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 281-290.

[11] A. MySQL, "MySQL," ed, 2001.

[12] M. Skirpan and T. Yeh, "Beyond the flipped classroom: learning by doing through challenges and Hack-a-thons," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, pp. 212-217.

Appendix

1. Blood Bank Server

- **Database**

db.js

```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "blood_bank_db"
});

connection.connect(function(err, conn) {
  if (err){
    console.log(err)
    throw err;
  }
  console.log("Success Connected to DataBase . . .");
});

module.exports = connection;
```

- **Routes**

1. Code of index.js

```
var express = require('express');
var router = express.Router();
var jwt = require('jsonwebtoken');
const url = require('url');

/* GET home page. */
router.get('/', function (req, res, next) {
  res.render('index', { title: 'Express' });
});



// route middleware to verify a token
router.use(function (req, res, next) {

  var str = url.parse(req.url).pathname;

  if (str.substr(-1) != '/') {
    str += '/';
  }
  // console.log(str)

  if (str == "/users/createUser/" || str == "/users/loginUser/" || str == "/users/loginAdmin/" 
    console.log("aaaaaaaaaaaaaaaaaaaaaaa")
    next();
    return;
  }

  // check header or url parameters or post parameters for token5
  var token = req.body.token || req.query.token || req.headers['x-access-admin_token'];

  // decode token
  if (token) {
    // verifies secret and checks exp
    jwt.verify(token, '94Tre500*_q+esltwert#04*0$998rwertwebrbr5',
    function (err, decoded) {
      if (err) {
        return res.json({
          success: false,
          msgCode: "TOKEN_AUTH_FAILED",
          msgText: 'Failed to authenticate token.',
          msgText: 'خطأ! الرجاء تسجيل الدخول من جديد'
        });
      } else {
        // console.Log("decoded.uid", decoded.uid)
        req.decoded = decoded;
        next();
      }
    });

  } else {
    // if there is no token
    // return an error
    return res.json({
      success: false,
      msgCode: "NO_TOKEN",
      msgText: 'No token provided.'
    });

  }
});
module.exports = router;
```

2. Code of the Users

```
var express = require('express');
var router = express.Router();
var db = require('../database/db.js');
var jwt = require('jsonwebtoken');
var multer = require('multer');


/*********** file storage config ***********/
var fileStorage = multer.diskStorage({
    destination: function (req, file, callback) {
        var Path = './uploads';
        callback(null, Path);
    },
    filename: function (req, file, callback) {
        callback(null, parseInt(Math.random()*10000) + file.originalname);
    }
});

var upload = multer({storage: fileStorage});

/* GET users listing. */
router.get('/', function (req, res, next) {
    res.send('respond with a resource');
});

router.post('/createUser', upload.single('hospitalImage'),
function (req, res, next) {
console.log("req.body", req.body);
console.log("req.file", req.file);
//  return;
    var userData = {
        fullName: req.body.fullName,
        email: req.body.email,
        password: req.body.password,
        hospitalName: req.body.hospitalName,
        hospitalPhone: req.body.hospitalPhone,
        hospitalLocation: req.body.hospitalLocation,
        hospitalImage: req.file ? req.file.filename : 'hospital.jpg'
    }
    //   console.log(userData);
    db.query('INSERT INTO users SET ?', userData,
     function (error, results, fields) {
        if (error) {
            console.log(error)
            res.json({
                "success": false,
                "result": "error adding data"
            })
        } else {
            res.json({
                "success": true,
                "result": "success create new user"
            })

        }
    });

});
router.get('/loginUser', function (req, res, next) {
    var userData = {
        email: req.query.email,
        password: req.query.password
    }
    //   console.log(userData);
    db.query("SELECT * FROM users WHERE email = '" + userData.email +
    "' AND password = '" + userData.password +"' AND isActive = true",
     function (error, result) {
        if (error || (result.length == 0)) {
            console.log(result);
            console.log(error);
            res.json({
                "success": false,
                "result": "error user data"
            })
        } else {
            console.log(result[0].id);
            var token = jwt.sign({
                uid: result[0].id
            },
                '94Tre500*_q+esltwert#04*0$998rwertwebrbr5', {
                    expiresIn: 60 * 60 * 24 * 7 //expires in 1 week
                });

            res.json({
                "success": true,
                "result": {
                    "token": token
                }
            })
        }
    });

});

router.get('/getProfile', function (req, res, next) {
    //   console.log(userData);
    db.query("SELECT id,email,hospitalName,fullName,hospitalPhone,"
    ,'hospitalLocation,hospitalImage FROM users WHERE id = '" +
     req.decoded.uid + "'", function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error)
            res.json({
                "success": false,
                "result": [],
                "err_msg": "error user data"
            })
        } else {
            res.json({
                "success": true,
                "result": result
            })
        }
    });

});

router.get('/getHospitals', function (req, res, next) {

    console.log(req.decoded);
    db.query("SELECT id,hospitalName,hospitalPhone,hospitalLocation,hospitalImage  FROM users",
```

```
    function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error)
            res.json({
                "success": false,
                "result": "error get Hospitals"
            })
        } else {
            res.json({
                "success": true,
                "result": result
            })
        }
    });

});

router.get('/getHospitalById', function (req, res, next) {
    var userData = {
        hospitalId: req.query.hospitalId
    }
    //   console.log(userData);
    db.query("SELECT * FROM users WHERE id = '" + userData.hospitalId + "'",
    function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error)
            res.json({
                "success": false,
                "result": "error user data"
            })
        } else {
            res.json({
                "success": true,
                "result": result
            })
        }
    });

});

router.get('/getBloodUnitsByHospitalId', function (req, res, next) {
    var hospitalId;
    if (!req.query.hospital_id) {
        hospitalId = req.decoded.uid;
    } else {
        hospitalId = req.query.hospital_id;
    }


    //   console.log(userData);
    db.query("SELECT * FROM blood_units WHERE hospital_id = '" + hospitalId + "'",
    function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error)
            res.json({
                "success": false,
                "result": [],
                "err_msg": "error user data"
            })
        } else {
            res.json({
                "success": true,
                "result": result
            })
        }
    });

});
router.get('/addBloodUnit', function (req, res, next) {
    var bloodData = {
        type: req.query.type,
        quantity: req.query.quantity,
        hospital_id: req.decoded.uid
    }

    //   **************
    db.query("SELECT * FROM blood_units WHERE hospital_id = '" +
    bloodData.hospital_id + "' AND type = '" +
    bloodData.type + "'", function (error, result) {

        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": [],
                "err_msg": "error user data"
            })
        }
        if (result.length == 0) { // insert type if not exist in
            db.query('INSERT INTO blood_units SET ?', bloodData,
            function (error, results, fields) {
                if (error) {
                    res.json({
                        "success": false,
                        "result": "error adding data"
                    })
                } else {
                    res.json({
                        "success": true,
                        "result": "success adding data"
                    })
                }
            });
        } else { // if type already stored then update it.
            var newQuantity = parseInt(result[0].quantity) + parseInt(bloodData.quantity);
            db.query("UPDATE blood_units SET quantity = '" + newQuantity +
            "' WHERE  hospital_id = '" + bloodData.hospital_id + "' AND type = '" +
            bloodData.type + "'", function (error, results, fields) {
                if (error) {
                    console.log(error);
                    res.json({
                        "success": false,
                        "result": "error adding data"
                    })
                } else {
                    res.json({
                        "success": true,
                        "result": "success adding data"
                    })
                }
            });
        }

    });

});
router.get('/updateBloodUnit', function (req, res, next) {
    var bloodData = {
        id: parseInt(req.query.bloodUnitId),
        quantity: parseInt(req.query.quantity),
        hospital_id: req.decoded.uid
    }

    db.query("UPDATE blood_units SET quantity = '" + bloodData.quantity +
    "' WHERE  hospital_id = '" + bloodData.hospital_id + "' AND id = '" +
    bloodData.id + "'", function (error, results, fields) {
        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error editing data"
            })
        } else {
            res.json({
                "success": true,
                "result": "success editing data"
            })
        }
    });


});
router.get('/deleteBloodUnit', function (req, res, next) {
    var bloodData = {
        bloodUnitId: parseInt(req.query.bloodUnitId),
        hospital_id: req.decoded.uid
    }

    db.query('DELETE FROM blood_units WHERE id = ' + bloodData.bloodUnitId +
    ' AND hospital_id=' +
    bloodData.hospital_id, function (error, results, fields) {
        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error removing blood unit"
            })
        } else {
            res.json({
                "success": true,
                "result": "success removing blood unit"
            })
        }
    });

});


router.get('/search', function (req, res, next) {
    var searchData = {
        hospital_id: req.decoded.uid,
        hospitalName: req.query.hospitalName,
```

```
        bloodType: req.query.bloodType
    }
    // bloodType: req.query.bloodType

    if (searchData.hospitalName && !searchData.bloodType) {

        db.query("SELECT id,hospitalName,hospitalPhone,hospitalLocation,"
        ,'hospitalImage FROM users WHERE (hospitalName LIKE '%" +
        searchData.hospitalName + "%')", function (error, result) {
            if (error) {
                console.log(error);
                res.json({
                    "success": false,
                    "result": "Filed to search term!!"
                })
            } else {
                res.json({
                    "success": true,
                    "result": result
                })
            }
        });
    } else if (searchData.hospitalName && searchData.bloodType) {

        db.query("SELECT id,hospitalName,hospitalPhone,hospitalLocation,"
        ,'hospitalImage FROM users WHERE (hospitalName LIKE '%" +
        searchData.hospitalName + "%')", function (error, hospitalResult) {
            if (error) {
                console.log(error);
                res.json({
                    "success": false,
                    "result": "Filed to search term!!"
                })
            } else {

                if (hospitalResult.length > 0) {
                    db.query("SELECT * FROM blood_units WHERE hospital_id = '" +
                    hospitalResult[0].id + "' AND type = '" +
                    searchData.bloodType + "'", function (error, bloodResult) {
                        if (error) {
                            console.log(error);
                            res.json({
                                "success": false,
                                "result": "Filed to search term !!"
                            })
                        } else {
                            if (bloodResult.length > 0) {
                                res.json({
                                    "success": true,
                                    "result": [{
                                        "hospitalName": hospitalResult[0].hospitalName,
                                        "hospitalPhone": hospitalResult[0].hospitalPhone,
                                        "bloodType": bloodResult[0].type,
                                        "quantity": bloodResult[0].quantity
                                    }]
                                })
                            } else {
                                res.json({ // if blood type not exist
                                    "success": true,
                                    "result": [{
                                        "hospitalName": hospitalResult[0].hospitalName,
                                        "hospitalPhone": hospitalResult[0].hospitalPhone,
                                        "bloodType": searchData.bloodType,
                                        "quantity": 0
                                    }]
                                })
                            }
                        }
                    });

                } else {
                    res.json({
                        "success": false,
                        "result": "Filed to search term!!, Hospital not found."
                    })
                }

            }
        });

    } else if (searchData.bloodType) {
        // db.query("SELECT * FROM blood_units WHERE type = '"+ searchData.blood
        db.query("SELECT users.hospitalName, users.hospitalPhone'"
        , "blood_units.type AS bloodType,'"
        ,"blood_units.quantity'"
        ,"FROM users JOIN blood_units ON users.id ='"
        ,"blood_units.hospital_id WHERE blood_units.type = '" +
        searchData.bloodType + "'", function (error, result) {
            if (error) {
                console.log(error)
                res.json({
                    "success": false,
                    "result": [],
                    "err_msg": "error user data"
                })
            } else {
                res.json({
                    "success": true,
                    "result": result
                })
            }
        });
    }

});

router.get('/updateProfile', upload.single('hospitalImage'),
function (req, res, next) {

    var userData = {
        hospital_id: req.decoded.uid,
        fullName: req.query.fullName,
        hospitalName: req.query.hospitalName,
        hospitalPhone: req.query.hospitalPhone,
        hospitalLocation: req.query.hospitalLocation
    }

    // console.log(userData);
    // return

    db.query("UPDATE users SET fullName = '" + userData.fullName + "', hospitalName = '" +
    userData.hospitalName + "', hospitalPhone = '" + userData.hospitalPhone
    + "', hospitalLocation = '" +
    userData.hospitalLocation + "'  WHERE  id = '" + userData.hospital_id + "'",
    function (error, results, fields) {
        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error editing data"
            })
        } else {
            res.json({
                "success": true,
                "result": "success editing data"
            })
        }
    });


});

router.get('/loginAdmin', function (req, res, next) {
    var userData = {
        email: req.query.email,
        password: req.query.password
    }
      console.log("**********************");
      console.log(userData);
    db.query("SELECT * FROM admins WHERE email = '" +
    userData.email + "' AND password = '" + userData.password + "'",
    function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error user data"
            })
        } else {
            console.log(result[0].id);
            var token = jwt.sign({
                uid: result[0].id
            },
                '94Tre500*_q+esltwert#04*0$998rwertwebrbr5', {
                    expiresIn: 60 * 60 * 24 * 7 //expires in 1 week
                });

            res.json({
                "success": true,
                "result": {
                    "token": token
                }
            })
        }
    });

});

router.get('/getNewUsers', function (req, res, next) {

    console.log(req.decoded);
    db.query("SELECT id,hospitalName,hospitalPhone,hospitalLocation'"
    ,"hospitalImage  FROM users WHERE isActive = false",
```

```
    function (error, result) {
        if (error || (result.length == 0)) {
            console.log(error)
            res.json({
                "success": false,
                "result": "error get Hospitals"
            })
        } else {
            res.json({
                "success": true,
                "result": result
            })
        }
    });

});

router.get('/acceptUser', function (req, res, next) {
    var userData = {
        id: parseInt(req.query.userId)
    }

    db.query("UPDATE users SET isActive = true WHERE  id = '" +
    userData.id + "'", function (error, results, fields) {
        if (error) {
            console.log(error);
            res.json({
                "success": false,
                "result": "error editing data"
            })
        } else {
            res.json({
                "success": true,
                "result": "success editing data"
            })
        }
    });


});

module.exports = router;
```

- **Code of the app.js**

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');

var index = require('./routes/index');
var users = require('./routes/users');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

app.use(express.static('uploads'));
// uncomment after placing your favicon in /public
//app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

// Add headers
app.use(function (req, res, next) {

  // Website you wish to allow to connect
  res.setHeader('Access-Control-Allow-Origin', '*');

  // Request methods you wish to allow
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');

  // Request headers you wish to allow
  res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');

  // Set to true if you need the website to include cookies in the requests sent
  // to the API (e.g. in case you use sessions)
  res.setHeader('Access-Control-Allow-Credentials', true);

  // Pass to next layer of middleware
  next();
});


app.use('/', index);
app.use('/users', users);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

- **Packages.json**

```json
{
  "name": "blood-bank-server",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.18.2",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "express": "~4.15.5",
    "jade": "~1.11.0",
    "jsonwebtoken": "^8.2.0",
    "morgan": "~1.9.0",
    "multer": "^1.3.0",
    "mysql": "^2.15.0",
    "serve-favicon": "~2.4.5"
  }
}

```

2. Blood Bank Server

- **Code of script.js**

```javascript
$(document).ready(function () {
    console.log("Ready !!!!!!!!!!!!!!!!!!!");
    /******* Login and sign up form */
    $('#login-form-link').click(function (e) {
        $("#login-form").delay(200).fadeIn(200);
        $("#register-form").fadeOut(200);
        $('#register-form-link').removeClass('active');
        $(this).addClass('active');
        e.preventDefault();
    });
    $('#register-form-link').click(function (e) {
        $("#register-form").delay(200).fadeIn(200);
        $("#login-form").fadeOut(200);
        $('#login-form-link').removeClass('active');
        $(this).addClass('active');
        e.preventDefault();
    });

    /********** sidebar  *****/
    $("#menu-toggle").click(function (e) {
        e.preventDefault();
        $("#wrapper").toggleClass("toggled");
    });

});
window.serverAddress = 'http://192.168.1.4:3000/';

// ************************ create user account *****
$('#register-form').on('submit', function (e) {
    e.preventDefault();


    if (!validateCreateNewUser()) {
        return;
    }
    console.log("SUBMITTTED THE POST")

    // console.log($.ajax);
    $.ajax({
        url: window.serverAddress + 'users/createUser',
        type: "POST",
        data: new FormData(document.forms['register-form']),
        processData: false,
        contentType: false,
        success: function (data) {
            // $("#form_output").html(data);
            console.log(data)
            if (data.success) {
                $('#succ_create_msg').css({ 'display': 'block' }).text("تم إنشاء الحساب بنجاح، الرجاء تسجيل الدخول.");
                $('#login-form-link').click();
            }
        },
        error: function (jXHR, textStatus, errorThrown) {
            console.log(errorThrown)
            // alert(errorThrown);
        }
    });
});

function validateCreateNewUser() {

```

```
    // validate signup form
    var signUpForm = document.forms["createUserForm"];



    document.getElementById("signUpErr").innerHTML = '';
    if (signUpForm["fullName"].value == "") {
        document.getElementById("signUpErr").innerHTML = "<p>* الإسم الكامل مطلوب!</p>";
        return false;
    }
    if (signUpForm["email"].value == "") {
        document.getElementById("signUpErr").innerHTML = "<p>* الإيميل مطلوب!</p>";
        return false;
    }
    if (signUpForm["password"].value == "") {
        document.getElementById("signUpErr").innerHTML = "<p>* كلمة المرور مطلوب</p>";
        return false;
    }
    if (signUpForm["confirm-password"].value == "") {
        document.getElementById("signUpErr").innerHTML = "<p>* تأكيد كلمة المرور مطلوب</p>";
        return false;
    }
    if (signUpForm["password"].value != signUpForm["confirm-password"].value) {
        document.getElementById("signUpErr").innerHTML = "<p>* كلمة المرور غير متطابقة</p>";
        return false;
    }
    if (signUpForm["hospitalName"].value == '') {
        document.getElementById("signUpErr").innerHTML = "<p>* إسم المستشفى مطلوب</p>";
        return false;
    }
    if (signUpForm["hospitalPhone"].value == '') {
        document.getElementById("signUpErr").innerHTML = "<p>* هاتف المستشفى مطلوب</p>";
        return false;
    }
    if (signUpForm["hospitalLocation"].value == '') {
        document.getElementById("signUpErr").innerHTML = "<p>* موقع المستشفى مطلوب</p>";
        return false;
    }

    return true;

}
// ********************* login user *************************
function loginUser() {
    // validate signup form
    var loginForm = document.forms["loginForm"];
    document.getElementById("loginErr").innerHTML = '';
    if (loginForm["email"].value == "") {
        document.getElementById("loginErr").innerHTML = "<p>* الإيميل مطلوب!</p>";
        return;
    }
    if (loginForm["password"].value == "") {
        document.getElementById("loginErr").innerHTML = "<p>* كلمة السر مطلوب</p>";
        return;
    }

    var userData = {
        email: loginForm["email"].value,
        password: loginForm["password"].value
    }
```

```
    // console.log(userData);
    $.get(serverAddress + 'users/loginUser', userData, function (data) {
        console.log(data);
        if (data.success) {
            // alert('تم تسجيل الدخول بنجاح');
            localStorage.setItem('bb_token', data.result.token);
            //console.log("Test")
            window.location.href = './site.html';
            //    openDashboard();
        } else {
            // alert('فشلت عملية تسجيل الدخول،الإيميل أو كلمة المرور خطأ.');
            document.getElementById("loginErr").innerHTML = "<p>*الإيميل أو كلمة المرور خطأ!</p>";
        }
    });
}

// ********************* get Hospitals *************************
function getHospitals() {
    $('#main_content')[0].innerHTML = '<h3>جاري التحميل. . . .</h3>';

    var token = localStorage.getItem('bb_token');
    var authData = {
        token: token
    }
    $.get(serverAddress + 'users/getHospitals', authData, function (data) {

        if (data.result) {
            var content = "";
            for (var i = 0; i < data.result.length; i++) {
                if (data.result[i].hospitalLocation) {

                    var temp = '<div class="col-sm-3 col-xs-6">' +
                        '<div class="thumbnail" onclick="showHospitalById(' + data.result[i].id + ')">' +
                        '<div class="img_box" ' +
                        'style="background-image:url(\'' + window.serverAddress +
                        (data.result[i].hospitalImage || 'hospital.jpg') + '\')" ></div>' +
                        '<div class="caption">' +
                        '<h3>' + data.result[i].hospitalName + '</h3>' +
                        '<p>' + data.result[i].hospitalLocation + '</p>' +
                        '</div>' +
                        '</div>' +
                        '</div>';
                    content = temp + content;
                }
            }
            console.log(content);
            $('#main_content')[0].innerHTML = content;
        }
    });
}

// ********************* get Hospitol Info by id *************************
function showHospitalById(id) {
    var token = localStorage.getItem('bb_token');
    var authData = {
        token: token,
        hospitalId: id
    }
```

```
    $('#main_content')[0].innerHTML = '<h3>جاري التحميل. . . .</h3>';
    $.get(serverAddress + 'users/getHospitalById', authData, function (data) {
        console.log(data);

        var content = '<div class="col-sm-9">' +
            '<div >' +
            '<div class="hospital_img" style="background-image:url(\'' +
            window.serverAddress + data.result[0].hospitalImage + '\')"></div><br>' +
            '<h3>اسم المستشفى: ' + data.result[0].hospitalName + '</h3>' +
            '<h4>الموقع: ' + data.result[0].hospitalLocation + '</h4>' +
            '<h4>الهاتف: ' + data.result[0].hospitalPhone + '</h4>' +
            '</div><hr><h3 style="margin-top: 50px;" class="text-success">وحدات الدم المتوفرة في هذه المستشفى</h3>' +
            '<div id="hospital_content"></div>' +
            '</div>';


        console.log(content);
        $('#main_content')[0].innerHTML = content;
        // get blood units
        console.log(id)
        getBloodUnitsByHospitalId(id);
    });
}


// ********************* Open Dashboard *************************
function openDashboard() {
    $('#main_content')[0].innerHTML = '<h2 class="text-center">لوحة التحكم</h2>' +
        '<h4>أضف وحدات دم للمستشفى الخاص بك:</h4>' +
        ' <div class="col-md-6">' +
        '<form name="addBloodForm" role="form">' +
        '<div class="form-group">' +
        '<label class="text-success"><h4>فصيلة الدم</h4></label>' +
        '<select class="form-control" name="type">' +
        '<option value="A+">A+</option>' +
        '<option value="A-">A-</option>' +
        '<option value="B+">B+</option>' +
        '<option value="B-">B-</option>' +
        '<option value="O+">O+</option>' +
        '<option value="O-">O-</option>' +
        '<option value="AB+">AB+</option>' +
        '<option value="AB-">AB-</option>' +
        '</select>' +
        '</div>' +
        '<div class="form-group">' +
        '<label class="text-success" ><h4 style="margin-top: 0">الكمية</h4></label>' +
        '<input type="number" min="1" name="quantity" class="form-control">' +
        '</div>' +
        '<div id="addBloodErr"></div>' +
        '<div class="form-group">' +
        '<input class="btn btn-success" type="button" onclick="addBloodUnit()" value="أضف">' +
        '</div>' +
        '</form>' +
        '<h3 style="margin-top: 80px;" class="text-info">وحدات الدم المتوفرة في المستشفى الخاص بك:</h3>' +
        '<div id="hospital_content"></div>' +
        '</div>';

    getBloodUnitsByHospitalId(null);
    getMyProfile();
}
```

```
function getMyProfile() {
    var token = localStorage.getItem('bb_token');
    var userData = {
        token: token
    };
    $.get(serverAddress + 'users/getProfile', userData, function (data) {
        console.log("**********", data);
        if (data.success) {
            var content = '<div class="sidebar_img" style="background-image:url(\'' +
            window.serverAddress + data.result[0].hospitalImage + '\')"></div><br>' +
                '<p>اسم المستشفى: ' + data.result[0].hospitalName + '</p>' +
                '<p>الموقع: ' + data.result[0].hospitalLocation + '</p>' +
                '<p>الهاتف: ' + data.result[0].hospitalPhone + '</p>';
            document.getElementById("profileInfo").innerHTML = content;
        } else {
            // alert("Failed to add blood unit, try again later.");
        }
    });
}

function addBloodUnit() {
    var addBloodForm = document.forms["addBloodForm"];
    document.getElementById("addBloodErr").innerHTML = '';
    if (addBloodForm["quantity"].value == "") {
        document.getElementById("addBloodErr").innerHTML = "<p>* الكمية مطلوب</p>";
        return true;
    }

    var token = localStorage.getItem('bb_token');
    var bloodData = {
        token: token,
        type: addBloodForm["type"].value,
        quantity: addBloodForm["quantity"].value
    }

    // return true;

    $.get(serverAddress + 'users/addBloodUnit', bloodData, function (data) {
        console.log(data);
        if (data.success) {
            openDashboard();
        } else {
            // alert("Failed to add blood unit, try again later.");
        }
    });
}


function getBloodUnitsByHospitalId(id) {
    var token = localStorage.getItem('bb_token');
    var HospitalData = {
        token: token,
        hospital_id: id
    }

    $('#hospital_content')[0].innerHTML = '<h3>جاري التحميل . . .</h3>';
    $.get(serverAddress + 'users/getBloodUnitsByHospitalId', HospitalData, function (data) {
        console.log(data);

        if (data.result && data.result.length > 0) {

            var content = '<table class="table table-striped custom-table">' +
                '<thead>' +
                '<tr>' +
                '<td><h3 class="text-danger">فصيلة الدم</h3></td>' +
                '<td><h3 class="text-danger">الكمية</h3></td>';
            var extraData = '';
            if (id == null) {
                extraData = '<td><h3 class="text-danger">تعديل</h3></td>' +
                    '<td><h3 class="text-danger">حذف</h3></td>';
            }
            content = content + extraData +
                '</tr>' +
                '</thead>' +
                '<tbody>';
            var rows = '';
            for (var i = 0; i < data.result.length; i++) {
                rows = rows +
                    '<tr>' +
                    '<td><b class="text-info">' + data.result[i].type + '<b></td>' +
                    '<td><b>' + data.result[i].quantity + '</b></td>';
                var extraData = '';
                if (id == null) {
                    extraData = '<td><button class="btn btn-warning" onclick="editBloodUnit(' + data.result[i].id +
                        ',' + data.result[i].quantity + ')">تعديل</button></td>' +
                        '<td><button class="btn btn-danger" onclick="deleteBloodunit(' + data.result[i].id + ')">حذف</button></td>';
                }
                rows = rows + extraData +
                    '</tr>';
            }
            content = content + rows +
                '</tbody>' +
                '</table>';

        } else {
            var content = '<h4 class="text-danger">لا يتوفر وحدات دم</h4>';
        }
        $('#hospital_content')[0].innerHTML = content;
    });

}

function editBloodUnit(bloodUnitId, bloodUnitQuantity) {
    console.log(bloodUnitId);
    console.log(bloodUnitQuantity);
    var newVal = prompt("Edit Quantity", bloodUnitQuantity);

    if (newVal != null) {
        var token = localStorage.getItem('bb_token');
        var bloodUnitData = {
            token: token,
            bloodUnitId: bloodUnitId,
            quantity: newVal
        }
        $.get(serverAddress + 'users/updateBloodUnit', bloodUnitData, function (data) {
            if (data.success) {
                openDashboard();
            } else {
                // alert('Failed to remove item please try again later!!');
                console.log("Failed to edit !!");
            }
        });
    }
}

function deleteBloodunit(id) {
    console.log(typeof id);
    var response = confirm("Are you sure you want to delete?");
    if (response == true) {
        var token = localStorage.getItem('bb_token');
        var bloodUnitData = {
            token: token,
            bloodUnitId: id
        }
        $.get(serverAddress + 'users/deleteBloodUnit', bloodUnitData, function (data) {
            console.log(data)
            if (data.success) {
                openDashboard();
            } else {
                // alert('Failed to remove item please try again later!!');
            }
        });
    }
}

function checkAuth() {
    if (!localStorage.getItem('bb_token')) {
        console.log(localStorage.getItem('bb_token'));
        window.location.href = './index.html';
    }
    openDashboard();
    console.log("Arrrr _______ Arrr");
}

function logoutUser() {
    localStorage.removeItem('bb_token');
    window.location.href = './index.html';
}


// ********** Search **********
function openSearchTab() {
    var content =
        '<h4 class="jumbotron">يمكنك البحث عن أي مستشفى من خلال الإسم، وأيضاً يمكنك البحث عن وحدات'+
        ' الدم المتوفرة في جميع المستشفيات أو في مستشفى بعينه. </h4>' +
        ' <div class="col-md-6">' +
        '<form name="searchForm" role="form">' +
        '<div class="form-group">' +
        '<label class="text-success" ><h4 style="margin-top: 0">اسم المستشفى</h4></label>' +
        '<input type="text"  name="hospitalName" class="form-control">' +
        '</div>' +
        '<div class="form-group">' +
        '<label class="text-success"><h4>فصيلة الدم</h4></label>' +
        '<select class="form-control" name="bloodType">' +
        '<option value=""></option>' +
        '<option value="A+">A+</option>' +
        '<option value="A-">A-</option>' +
        '<option value="B+">B+</option>' +
        '<option value="B-">B-</option>' +
        '<option value="O+">O+</option>' +
        '<option value="O-">O-</option>' +
        '<option value="AB+">AB+</option>' +
        '<option value="AB-">AB-</option>' +
        '</select>' +
        '</div>' +
```

```
        '</div>' +
        '<div class="text-danger" id="searchErr"></div>' +
        '<div class="form-group">' +
        '<input class="btn btn-success" type="button" onclick="searchBackend()" value="ابحث">' +
        '</div>' +
        '</form></div><div class="col-md-12">' +
        '<div style="margin-top: 50px"><hr/><div id="search_result"></div></div></div>';
    $('#main_content')[0].innerHTML = content;

}

function searchBackend() {
    $('#search_result')[0].innerHTML = '';


    var searchForm = document.forms["searchForm"];
    document.getElementById("searchErr").innerHTML = '';
    if (searchForm["hospitalName"].value == "" && searchForm["bloodType"].value == "") {
        document.getElementById("searchErr").innerHTML =
        "<p>* الرجاء إدخال اسم المستشفى أو وحدة الدم أو كليهما.</p>";
        return true;
    }


    var token = localStorage.getItem('bb_token');
    var bloodData = {
        token: token
    }
    if (searchForm["bloodType"].value != "") {
        bloodData.bloodType = searchForm["bloodType"].value;
    }
    if (searchForm["hospitalName"].value != "") {
        bloodData.hospitalName = searchForm["hospitalName"].value;
    }
```

```
    }

    console.log(bloodData);

    // return true;

    $.get(serverAddress +'users/search', bloodData, function (data) {
        console.log(data);
        if (data.success && (data.result.length > 0)) {
            if (data.result[0].hospitalName && !data.result[0].bloodType) {
                var content = "";
                for (var i = 0; i < data.result.length; i++) {

                    if (data.result[i].hospitalLocation) {
                        var temp = '<div class="col-sm-4">' +
                            '<div class="thumbnail" onclick="showHospitalById(' + data.result[i].id + ')">' +
                            '<div class="img_box" style="background-image:url(\'' + window.serverAddress +
                            (data.result[i].hospitalImage || 'hospital.jpg') + '\')" ></div>' +
                            '<div class="caption">' +
                            '<h3>' + data.result[i].hospitalName + '</h3>' +
                            '<p>' + data.result[i].hospitalLocation + '</p>' +
                            '</div>' +
                            '</div>' +
                            '</div>';
                        content = temp + content;
                    }
                }
                $('#search_result')[0].innerHTML = '<h3 class="text-info text-center">نتائج البحث</h3>' + content;
            } else if (data.result[0].hospitalName && data.result[0].bloodType) {
                var content = '<table class="table table-striped custom-table">' +
                    '<thead>' +
                    '<tr>' +
                    '<td><h3 class="text-danger">اسم المستشفى</h3></td>' +
                    '<td><h3 class="text-danger">فصيلة الدم</h3></td>' +
                    '<td><h3 class="text-danger">الكمية</h3></td>' +
```

```
                    '<td><h3 class="text-danger">هاتف المستشفى</h3></td>' +
                    '</thead>' +
                    '<tbody>';
                var rows = '';
                for (var i = 0; i < data.result.length; i++) {
                    rows = rows + '<tr>' +
                        '<td>' +
                        data.result[i].hospitalName +
                        '</td>' +
                        '<td>' +
                        data.result[i].bloodType +
                        '</td>' +
                        '<td>' +
                        data.result[i].quantity +
                        '</td>' +
                        '<td>' +
                        data.result[i].hospitalPhone +
                        '</td>' +
                        '</tr>';

                }
                content = content + rows +
                    '</tbody>' +
                    '</thead>';
                $('#search_result')[0].innerHTML = '<h3 class="text-info text-center">نتائج البحث</h3>' + content;
            }
        } else if (data.success && (data.result.length == 0)) {
            $('#search_result')[0].innerHTML = '<h3 class="text-danger text-center">لا يوجد نتائج Y</h3>'

        } else {
```

```
                $('#search_result')[0].innerHTML = '<h3 class="text-danger text-center">فشل عملية البحث</h3>'
            }
        });
    }
    /********************* Show Profile ***************/
    function openProfile() {
        // $('#main_content')[0].innerHTML


        var token = localStorage.getItem('bb_token');
        var userData = {
            token: token
        };
        $.get(serverAddress + 'users/getProfile', userData, function (data) {
            console.log("**********", data);
            if (data.success) {
                var content = '<h2 class="text-center">البروفايل الشخصي</h2>' +
                    '<div class="row"><div class="col-md-6">' +
                    '<form  name="editProfileForm" id="editProfileForm"  role="form" class="mt-40">' +
                    '<div class="form-group">' +
                    '<label>الإسم الكامل*</label>' +
                    '<input type="text" name="fullName" id="user_name_new"  class="form-control"'+
                    'placeholder="الإسم الكامل*" value="' + data.result[0].fullName + '">' +
                    '</div>' +
                    '<div class="err_msg" id="user_email_new_msg"></div>' +
                    '<div class="form-group">' +
                    '<label>اسم المستشفى*</label>' +
                    '<input type="text" name="hospitalName" id="hospital_name_new"  class="form-control"'+
                    'placeholder="اسم المستشفى*" value="' + data.result[0].hospitalName + '">' +
                    '</div>' +
```

```
                    '<div class="form-group">' +
                    '<label>هاتف المستشفى*</label>' +
                    '<input type="text" name="hospitalPhone" id="hospital_phone"  class="form-control"'+
                    'placeholder="هاتف المستشفى*" value="' + data.result[0].hospitalPhone + '">' +
                    '</div>' +
                    '<div class="form-group">' +
                    '<label>موقع المستشفى*</label>' +
                    '<input type="text" name="hospitalLocation" id="hospital_location"  class="form-control"'+
                    'placeholder="موقع المستشفى*" value="' + data.result[0].hospitalLocation + '">' +
                    '</div>' +

                    '<input type="hidden" name="token"  class="form-control">' +
                    '<p>* : تعني أن الحقل مطلوب.</p>' +
                    '<p id="signUpErr" class="err_msg"></p>' +
                    '<div class="form-group">' +
                    '<div class="row">' +
                    '<div class="col-sm-6 col-sm-offset-3">' +
                    '<input type="button"  class="form-control btn btn-register" value="تعديل البيانات" onclick="editProfile()">' +
                    '</div>' +
                    '</div>' +
                    '</div>' +
                    '</form>' +
                    '</div></div>';
                $('#main_content')[0].innerHTML = content;
            } else {
                // alert("Failed to add blood unit, try again later.");
                $('#main_content')[0].innerHTML = '<h4 class="text-danger">فشل عملية جلب معلومات المستخدم !!</h4>';
            }
        });

```

```
function editProfile() {
    // validate signup form
    var editProfileForm = document.forms["editProfileForm"];
    document.getElementById("signUpErr").innerHTML = '';
    if (editProfileForm["fullName"].value == "") {
        document.getElementById("signUpErr").innerHTML = "<p>* الإسم الكامل مطلوب!</p>";
        return;
    }

    if (editProfileForm["hospitalName"].value == '') {
        document.getElementById("signUpErr").innerHTML = "<p>* إسم المستشفى مطلوب</p>";
        return ;
    }
    if (editProfileForm["hospitalPhone"].value == '') {
        document.getElementById("signUpErr").innerHTML = "<p>* هاتف المستشفى مطلوب</p>";
        return ;
    }
    if (editProfileForm["hospitalLocation"].value == '') {
        document.getElementById("signUpErr").innerHTML = "<p>* موقع المستشفى مطلوب</p>";
        return;
    }


    // if the user fill out all field
    // send data to backend server
    var token = localStorage.getItem('bb_token');
    var userData = {
        fullName: editProfileForm["fullName"].value,
        hospitalName: editProfileForm["hospitalName"].value,
        hospitalPhone: editProfileForm["hospitalPhone"].value,
```

```
        hospitalLocation: editProfileForm["hospitalLocation"].value,
        token: token
    }
    $.get(serverAddress + 'users/updateProfile', userData, function (data) {

        if (data.success) {
            getMyProfile();
            alert("تم تعديل البيانات بنجاح");
        } else {
            alert('فشلت عملية تعديل البيانات، الرجاء المحاولة لاحقاً');

        }
    });
}
function getAboutSystem(){
    var content = '<div class="box">'+
    '<h3 style="color:red;">بنك الدم</h3>'+
    '<p>لا تزال المستشفيات في فلسطين تعمل بشكل مستقل،وليست مترابطة، وأيضا نقص الموارد المتاحة لدى المستشفيات الفلسطينية</p>'+
    ',لهذا منال حاجة ملحة لربط المستشفيات معا للمشاركة بالموارد المغتلفة والموارد العلاجية .</p>'+
    '<p>في هذا النظام ويكون بداية خطوة هامة لربط المستشفيات معا بالمصاح للمستشفيات للعصول علي معلومات حوله توفر'+
    ',وحدات الدم وأنواعها وكمياتها لتسهيل العملية العلاجية في كل مشتشفي للمشاركة في انقاذ المرضى</p>'+
    '<img src="imgs/natalie_portman_actress_face_brunette_110931_3840x2160.jpg" alt="..." width="410px"/>'+
    '<h4>هذا النظام كمتطلب لتخرج طلاب مندمة الإتصالات في جامعة بوليتكنك فلسطين بإشراف الدكتور:</h4>'+
      '<ul>'+
      '<li>د.مراد أبو سبيح</li>'+
      '</ul>'+
        ' <h4>الطلاب:</h4>'+
        '<ol>'+
        '<li>ابراهيم العجلة</li>'+
        '<li>معمد نلاحمة</li>'+
        '<li>مديل عيابدة</li>'+
        '</ol>'+
'</div>';
    $('#main_content')[0].innerHTML = content;
}
// ********************* login admin ************************
function loginAdmin() {

    var loginForm = document.forms["loginForm"];
    var userData = {
        email: loginForm["email"].value || " ",
        password: loginForm["password"].value || " "
    }

    console.log(userData);

    $.get(serverAddress + 'users/loginAdmin', userData, function (data) {
        console.log(data);
        if (data.success) {
            // alert('تم تسجيل الدخول بنجاح');
            localStorage.setItem('bb_admin_token', data.result.token);

            var userData = {
                token:   data.result.token
            }

            $.get(serverAddress + 'users/getNewUsers', userData, function (data) {
                console.log(data)

                //
```

```
                var content = '<table class="table table-striped custom-table">' +
                '<thead>' +
                '<tr>' +
                '<td><h3 class="text-danger">اسم المستشفى</h3></td>' +
                '<td><h3 class="text-danger">الموقع</h3></td>' +
                '<td><h3 class="text-danger">هاتف المستشفى</h3></td>' +
                '<td><h3 class="text-danger">قبول الانضمام</h3></td>' +
                '<theads>' +
                '<tbody>';
            var rows = '';
            for (var i = 0; i < data.result.length; i++) {
                rows = rows + '<tr data-id='+data.result[i].id+'>' +
                    '<td>' +
                    data.result[i].hospitalName +
                    '</td>' +
                    '<td>' +
                    data.result[i].hospitalLocation +
                    '</td>' +
                    '<td>' +
                    data.result[i].hospitalPhone +
                    '</td>' +
                    '<td>' +
                    '<button onclick="acceptUser('+ data.result[i].id +')">قبول</button>'+
                    '</td>' +
                    '</tr>';

            }
            content = content + rows +
                '</tbody>' +
                '</thead>';
```

```
                $('#admin_login_form')[0].innerHTML = '<a onclick="logoutAdmin()">تسجيل الخروج</a>'+
                '<h3 class="text-info text-center">المستعدمين الجدد</h3>' + content;
                })

                //    openDashboard();
            } else {
                // alert('فشلت عملية تسجيل الدخول،الإيميل أو كلمة المرور خطأ.');
                document.getElementById("loginErr").innerHTML = "<p>*الإيميل أو كلمة المرور خطأ</p>";
            }
        });
}

function acceptUser(id){
    var userData = {
        token: localStorage.getItem('bb_admin_token'),
        userId: id
    }
    $.get(serverAddress + 'users/acceptUser', userData, function (data) {
        console.log(data);
        if(data.success){
            alert("تمت الموافقة على الطلب");

            $("[data-id="+id+"]").css("display", "none");
        }else{
            alert("لم تتم العملية، الرجاء المعاولة لاحقاً");

        }
    })
}

function logoutAdmin(){
    localStorage.removeItem('bb_admin_token');
    window.location.href = "./system_admin.html";
}
```

- **Code CSS**

```css
/********** general ****************/

@font-face {
    font-family: 'Roboto';
    src: url('Roboto-Regular.ttf');
}
@font-face {
    font-family: 'DroidSansArabic';
    src: url('./DroidSansArabic.ttf');
}

body {
    font-family: 'Roboto','DroidSansArabic', 'sans-serif';
    font-size: 18px;
}
.mt-20{
    margin-top: 20px;
}
.mt-40{
    margin-top: 30px;
}
/********** Login and signup form ************/
.panel-login {
    border-color: #ccc;
    -webkit-box-shadow: 0px 2px 3px 0px rgba(0,0,0,0.2);
    -moz-box-shadow: 0px 2px 3px 0px rgba(0,0,0,0.2);
    box-shadow: 0px 2px 3px 0px rgba(0,0,0,0.2);
}
.panel-login>.panel-heading {
    color: #00415d;
    background-color: #fff;
    border-color: #fff;
    text-align:center;
}
.panel-login>.panel-heading a{
    text-decoration: none;
    color: #666;
    font-weight: bold;
    font-size: 15px;
    -webkit-transition: all 0.1s linear;
    -moz-transition: all 0.1s linear;
    transition: all 0.1s linear;
}
.panel-login>.panel-heading a.active{
    color: #029f5b;
    font-size: 18px;
}
.panel-login>.panel-heading hr{
    margin-top: 10px;
    margin-bottom: 0px;
    clear: both;
    border: 0;
    height: 1px;
    background-image: -webkit-linear-gradient(left,rgba(0, 0, 0, 0),rgba(0, 0, 0, 0.15),rgba(0, 0, 0, 0));
    background-image: -moz-linear-gradient(left,rgba(0,0,0,0),rgba(0,0,0,0.15),rgba(0,0,0,0));
    background-image: -ms-linear-gradient(left,rgba(0,0,0,0),rgba(0,0,0,0.15),rgba(0,0,0,0));
    background-image: -o-linear-gradient(left,rgba(0,0,0,0),rgba(0,0,0,0.15),rgba(0,0,0,0));
}
.panel-login input[type="text"],.panel-login input[type="email"],.panel-login input[type="password"] {
    height: 45px;
    border: 1px solid #ddd;
    font-size: 16px;
    -webkit-transition: all 0.1s linear;
    -moz-transition: all 0.1s linear;
    transition: all 0.1s linear;
}
.panel-login input:focus {
    outline:none;
    -webkit-box-shadow: none;
    -moz-box-shadow: none;
    box-shadow: none;
    border-color: #1CB94E;
}
.btn-login {
    /* background-color: #59B2E0; */
    background-color: #1CB94E;
    outline: none;
    color: #fff;
    font-size: 18px;
    height: auto;
    font-weight: normal;
    padding: 14px 0;
    border-color: #1CB94E;
}
.btn-login:hover,
.btn-login:focus {
    color: #fff;
    background-color: #1CB94E;
    border-color: #1CB94E;
}
.forgot-password {
    text-decoration: underline;
    color: #888;
}
.forgot-password:hover,
.forgot-password:focus {
    text-decoration: underline;
    color: #666;
}

.btn-register {
    background-color: #1CB94E;
    outline: none;
    color: #fff;
    font-size: 18px;
    height: auto;
    font-weight: normal;
    padding: 14px 0;
    border-color: #1CB94A;
}
.btn-register:hover,
.btn-register:focus {
    color: #fff;
    background-color: #1CA347;
    border-color: #1CA347;
}
.err_msg{
    color: #f54242;
    font-size: 16px;
}

/************** sidebar style ************/
/*!
 * Start Bootstrap - Simple Sidebar HTML Template (http://startbootstrap.com)
 * Code licensed under the Apache License v2.0.
 * For details, see http://www.apache.org/licenses/LICENSE-2.0.
 */

/* Toggle Styles */

#wrapper {
    padding-right: 0;
    -webkit-transition: all 0.5s ease;
    -moz-transition: all 0.5s ease;
    -o-transition: all 0.5s ease;
    transition: all 0.5s ease;
}

#wrapper.toggled {
    padding-right: 250px;
}

#sidebar-wrapper {
    z-index: 1000;
    position: fixed;
    right: 250px;
    width: 0;
    height: 100%;
    margin-right: -250px;
    overflow-y: auto;
    background: #000;
```

```
    -webkit-transition: all 0.5s ease;
    -moz-transition: all 0.5s ease;
    -o-transition: all 0.5s ease;
    transition: all 0.5s ease;
    overflow: hidden;
}

#wrapper.toggled #sidebar-wrapper {
    width: 250px;
}

#page-content-wrapper {
    width: 100%;
    position: absolute;
    padding: 15px;
}

#wrapper.toggled #page-content-wrapper {
    position: absolute;
    margin-right: -230px;
}

#menu-toggle{
    position: relative;
    left: 18px;
    top: -19px;
}


/* Sidebar Styles */

.sidebar-nav {
    position: absolute;
    top: 0;
    width: 230px;
    margin: 0;
    padding: 0;
    list-style: none;
}

.sidebar-nav li {
    text-indent: 20px;
    line-height: 40px;
}

.sidebar-nav li a {
    display: block;
    text-decoration: none;
    color: #999999;
}

.sidebar-nav li a:hover {
    text-decoration: none;
    color: #fff;
    background: rgba(255,255,255,0.2);
}

.sidebar-nav li a:active,
.sidebar-nav li a:focus {
    text-decoration: none;
}

}

.sidebar-nav > .sidebar-brand {
  margin-bottom: 20px;
}



@media(min-width:768px) {
    #wrapper {
        padding-right: 230px;
    }

    #wrapper.toggled {
        padding-right: 0;
    }

    #sidebar-wrapper {
        width: 230px;
    }

    #wrapper.toggled #sidebar-wrapper {
        width: 0;
    }

    #page-content-wrapper {
        padding: 20px;
        position: relative;
    }

    #wrapper.toggled #page-content-wrapper {
        position: relative;
        margin-right: 0;
    }
}

.thumbnail{
    cursor: pointer;
}
.thumbnail:hover{
    background: #eee;
}
.thumbnail .img_box{
    width: 100%;
    height: 276px;
    background-size: cover;
    background-position: center;
}

.sidebar_img{
    width: 60%;
    height: 80px;
    margin: 0 auto;
    background-size: cover;
    background-position: center;
}
.hospital_img{
    width: 250px;
    height: 180px;
    background-size: cover;
    background-position: center;
}
/********************** available blood units table **************/
.custom-table{
    text-align: center;
    font-size: 20px;
}
.custom-table button{
    min-width: 80px;
}
#profileInfo{
    margin-top: 50px;
    padding: 20px 8px;
    border-top: 2px solid #ccc;
    color: #3d9a51
}
#profileInfo img{
    display: block;
    width: 50%;
    margin: 0 auto 20px;
}
```

- **Code index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Register</title>
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/bootstrap-rtl.min.css">
    <link rel="stylesheet" href="css/custom.css">
</head>
<body dir="rtl">

<div class="container" style="margin-top: 20vh">
    <h3 id="succ_create_msg" class="jumbotron text-center"
    style="display:none;color: green"></h3>
    <div class="row">
        <div class="col-md-6 col-md-offset-3">
            <div class="panel panel-login">
                <div class="panel-heading">
                    <div class="row">
                        <div class="col-xs-6">
                            <a href="#" class="active"
                            id="login-form-link">تسجيل الدخول</a>
                        </div>
                        <div class="col-xs-6">
                            <a href="#" id="register-form-link">إنشاء حساب</a>
                        </div>
                    </div>
                    <hr>
                </div>
                </div>
                <div class="panel-body">
                    <div class="row">
                        <div class="col-lg-12">
                            <form id="login-form" name="loginForm"
                            role="form" style="display: block;" >
                                <div class="form-group">
                                    <input type="email" name="email" id="user_email" tabindex="1"
                                    class="form-control" placeholder="الإيميل" value="">
                                </div>
                                <div class="form-group">
                                    <input type="password" name="password"
                                    id="user_password" tabindex="2" class="form-control"
                                    placeholder="كلمة المرور">
                                </div>
                                <p id="loginErr" class="err_msg"></p>

                                <div class="form-group">
                                    <div class="row">
                                        <div class="col-sm-6 col-sm-offset-3">
                                            <input type="button" onclick="loginUser()"
                                            name="login-submit" id="login-submit" tabindex="4"
                                            class="form-control btn btn-login"
                                            value="تسجيل الدخول">
                                        </div>
                                    </div>
                                </div>
                                <div class="form-group">
                                    <div class="row">
                                        <div class="col-lg-12">
                                            <div class="text-center">
                                                <a href="#" tabindex="5"
                                                class="forgot-password">نسيت كلمة المرور؟</a>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </form>

                            <!-- <form id="register-form" method="POST" action="http://192.168.215.2:3000/users/cre
                            <form id="register-form"  name="createUserForm"
                             enctype="multipart/form-data"  role="form" style="display: none;" >

                                <div class="form-group">
                                    <input type="text" name="fullName" id="user_name_new"
                                     class="form-control" placeholder="الإسم الكامل*" value="">
                                </div>
                                <div class="form-group">
                                    <input type="email" name="email" id="user_email_new"
                                    class="form-control" placeholder="الإيميل*" value="">
                                </div>
                                <div class="err_msg" id="user_email_new_msg"></div>
                                <div class="form-group">
                                    <input type="password" name="password"
                                    id="user_password_new" class="form-control" placeholder="كلمة المرور*">
                                </div>
                                <div class="form-group">
                                    <input type="password" name="confirm-password"
                                    id="user_confirm_pass" class="form-control" placeholder="تأكيد كلمة المرور*">
                                </div>
```

```
                                    <div class="form-group">
                                        <input type="text" name="hospitalName" id="hospital_name_new"
                                            class="form-control" placeholder="اسم المستشفى*" value="">
                                    </div>
                                    <div class="form-group">
                                        <input type="text" name="hospitalPhone" id="hospital_phone"
                                            class="form-control" placeholder="هاتف المستشفى*" value="">
                                    </div>
                                    <div class="form-group">
                                        <input type="text" name="hospitalLocation" id="hospital_location"
                                            class="form-control" placeholder="موقع المستشفى*" value="">
                                    </div>
                                    <div class="form-group">
                                        <label for="">تحميل صورة</label>
                                        <input type="file" name="hospitalImage" id="hospital_image"
                                            class="form-control">
                                    </div>
                                    <p>* : تعني أن الحقل مطلوب.</p>
                                    <p id="signUpErr" class="err_msg"></p>
                                    <div class="form-group">
                                        <div class="row">
                                            <div class="col-sm-6 col-sm-offset-3">

                                                <!-- <input type="submit" name="register-submit" id="register-submit"
                                                <input type="submit" name="register-submit" id="register-submit"
                                                    class="form-control btn btn-register" value="إنشاء حساب" >
                                            </div>
                                        </div>
                                    </div>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

- **Code Side.html**

```
<!DOCTYPE html>

<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=0.1">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Blood Bank</title>
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/bootstrap-rtl.min.css">
    <link rel="stylesheet" href="css/custom.css">
</head>

<body dir="rtl" onload="checkAuth()">


    <div id="wrapper">

        <!-- Sidebar -->
        <div id="sidebar-wrapper">
            <ul class="sidebar-nav">
                <li class="sidebar-brand">
                    <img src="imgs/bloodbank.jpg" alt=""
                    width="100%" style="margin-right: -25px">
                </li>
                <li>
                    <a href="#" onclick="openDashboard()">لوحة التحكم</a>
                </li>
                <li>
                    <a href="#" onclick="getHospitals()">المستشفيات</a>
                </li>
                <li>
                    <a href="#" onclick="openSearchTab()">البحث</a>
                </li>
                <li>
                    <a href="#" onclick="getAboutSystem()">حول النظام</a>
                </li>
                <li>
                    <a href="#" onclick="openProfile()">بروفايلي</a>
                </li>
                <li>
                    <a href="#" onclick="logoutUser()">تسجيل الخروج</a>
                </li>
                <div id="profileInfo"></div>
            </ul>
        </div>
        <!-- /#sidebar-wrapper -->

        <!-- Page Content -->
        <div id="page-content-wrapper">
            <a href="#menu-toggle" class="btn btn-default" id="menu-toggle">=</a>
            <div class="container-fluid">
                <div class="row" id="main_content"></div>
            </div>
        </div>
        <!-- /#page-content-wrapper -->

    </div>
    <!-- /#wrapper -->

    <!-- Menu Toggle Script -->


    <script src="js/jquery.min.js"></script>
    <script src="js/script.js"></script>
</body>

</html>

```

- **Admin.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Register</title>
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/bootstrap-rtl.min.css">
    <link rel="stylesheet" href="css/custom.css">
</head>
<body dir="rtl">

<div class="container" style="margin-top: 20vh">

    <div class="row">
        <div class="col-md-6 col-md-offset-3" id="admin_login_form">
            <div class="panel panel-login">
                <div class="panel-heading">
                    <div class="row">
                        <div class="col-xs-6">
                            <a href="#" class="active" id="login-form-link">تسجيل الدخول</a>
                        </div>
                    </div>
                    <hr>
                </div>
                <div class="panel-body">
                    <div class="row">
                        <div class="col-lg-12">
                            <form id="login-form" name="loginForm"
                            role="form" style="display: block;" >
                                <div class="form-group">
                                    <input type="email" name="email"
                                    id="user_email" tabindex="1" class="form-control"
                                    placeholder="الإيميل" value="">
                                </div>
```

```
                                    placeholder="الإيميل" value="">
                                </div>
                                <div class="form-group">
                                    <input type="password" name="password" id="user_password"
                                    tabindex="2" class="form-control" placeholder="كلمة المرور">
                                </div>
                                <p id="loginErr" class="err_msg"></p>

                                <div class="form-group">
                                    <div class="row">
                                        <div class="col-sm-6 col-sm-offset-3">
                                            <input type="button" onclick="loginAdmin()" name="login-submit"
                                            id="login-submit" tabindex="4" class="form-control btn btn-login"
                                            value="تسجيل الدخول">
                                        </div>
                                    </div>
                                </div>

                            </form>


                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>



    <script src="js/jquery.min.js"></script>
    <script src="js/script.js"></script>
</body>
</html>
```