

Blood Bank Management System Project Report



**University of Peradeniya, Faculty of Engineering
CO527 - Advanced Database Systems
Semester-06 Project**

Group Members:

- E/15/077-Dilhani K.W.A.K.K.
- E/15/211-Maduwanthi S.A.I.
- E/15/279-Premathilake L.S.W.S.

Abstract

The process of managing the blood bank that is received from the blood donation events needs a proper and systematic management. The blood bank must be handled with care and treated thoroughly as it is related to someone's life. The development of Web-based Blood Bank Management System (BBMS) is proposed to provide a management functional to the blood bank in order to handle the blood banks in Sri Lanka.

The Project describes the system blood bank management system. This report will help you to know in depth the actual work that has been done as a team work. The main objective of this application is to automate the complete operations of the blood bank. They need to maintain hundreds of thousands of records. Also searching should be very fast, so they can find required details instantly. This system is intended to provide information about the availability of blood in emergency conditions at their respective locations. Main objective is to create a system which helps the Hospital employees to complete their work faster in a simple way by using a computer, not the oldest way which is used paper. Also our project contains updated information and media gallery and many things else.

Table of Content

1. Introduction
2. Statement of the problem
3. Analysis of the Existing Blood Bank Management System in Sri Lanka
4. Requirement Analysis
5. Objectives
6. Design
 - 6.1 Logical Design of the proposed system
 - 6.1.1 ER Diagram
 - 6.1.2 Flow chart
 - 6.1.3 Use Case Diagrams
7. Implementation
 - 7.1 Front-end implementation
 - 7.2 Back-end implementation
 - 7.3 Building connection between front-end and back-end
8. Comparison between SQL and NoSQL databases
9. Interface Designing
10. Testing
11. Conclusion

1. Introduction

A blood donation is a process whereby a person voluntarily has blood drawn to be used for future transfusions when in need at hospitals for treatment procedures that require them. Donation may be of whole blood (blood drawn directly from the body) or of specific components of the blood; such as red blood cells, white blood cells, plasma, and platelets. Blood banks often participate in the process of collecting blood and other procedures such as managing stocks, approving blood requests and updating donation information.

The inspiration of this project is to improve blood banks in Sri Lanka and to develop a blood bank information system which focuses on making an online system that is accessible for administrators. The system is also developed for the administrators, who are the main authority in the system. Administrators can add, modify, delete, and query any donation information if necessary. The administrator is also responsible for responding to the hospital's blood requests and checking the stocks in the blood bank's inventory. Blood donors come to donate blood, hospital staff can check blood donor existence, blood stock management, can edit or remove blood donor details. And also managers can add new blood bags into blood stock with entered recordings and can remove stock when they supply them for recipients. All of the data entered is recorded in a real-time database.

Blood Bank Management System is a website based on JavaScript. The purpose of this project was to develop a blood management information system to assist in the management of blood donor records and ease or control the distribution of blood in various parts of the country based on the hospital demand. This project involves mainly two modules, user interface and database management. The database is real-time updated.

2. Statement of the Problem

The Blood Bank management system is not a newly created one. Even though there are such kinds of systems in hospitals, following problems arise when using a typical blood bank's existing system. Therefore our target is giving some acceptable solutions for those.

- **Personal profile accessibility**

The donor's information can only be updated by the administrators of the blood bank. A donor can update their information by calling, faxing, e-mailing, but not by themselves. This is a waste of time just for updating a piece of information and it may be troublesome for some donors.

- **Blood stock management**

Blood banks are required to maintain an account of blood bags in the inventory. This increases with each blood donation recorded in our system, and decreases as they are checked out upon hospital requests. Our system will need to keep the information up-to-date to ensure correctness of the inventory.

- **Blood result notifications**

After the process of blood donation, the donor will receive a card that only contains their name and blood type. They will not be notified of their blood result unless they request that information from the blood bank.

- **Mailing by postal system**

Blood banks will only mail donors when the donated blood is disqualified, however, this mail is sent through the postal system to the donor's given address. If the donor's address is recorded incorrectly, the mail will be sent to the wrong address and the donor will never be notified that their blood is rejected and given the reason for that.

- **Lost or damaged card**

A typical membership card can easily get damaged if it is exposed to the sunlight or weather and this causes ruin the card's barcode which is significantly important for retrieving records. If the card gets lost or stolen, the donor has to make a replacement card to keep their membership at the blood bank.

3. Analysis of the Existing Blood Bank Management System in Sri Lanka

There are two types of process in the existing system: the blood donation process by donors, and the blood request process by hospitals. In both processes, an administrator is in charge of managing the blood inventory in the blood bank.

Blood Donation Process by Donors

When a new donor comes to donate blood, they are required to fill out their personal information during the registration process before making a donation. After the donation, the donor is given a donor identification card with their name, blood type and a barcode to be used as a reference for future donations. The barcode is used to retrieve the donor's record containing their personal information, medical history and donation information, including blood results. Only blood bank administrators have the authority to access the donor's records, since the system is only available for their use within the organization. This makes it difficult for donors to make changes to their personal information within the system. That is, for donors to update their personal information, such as their phone number, mailing address, or e-mail, they cannot update the information by themselves, but have to contact the blood bank center to update their information.

At the back the card is a table that contains the number of donations, date, location, and the blood collector's signature. Existing donors can submit their donor ID cards to retrieve their personal information and donation records and start the blood donation process, and they will be given a new card after they have donated blood for a total of eight times. Having a donor ID card may be a tangible reminder to people that they are helping lives as a blood donor; however, possessing a physical card comes with drawbacks such as loss or damage. To ensure donors can still identify themselves with the system, other credentials, such as username and password, can be used as a safeguard if their donor ID card is lost or damaged. If the donated blood is disqualified, the donor will be notified through postal mail that their blood component is reactive to viruses, meaning that there is a positive result of the blood being infected, and the organization will also inform the donor to perform another blood test at the blood bank to confirm the result of blood. If the blood is qualified, the administrator then will deposit the blood into the inventory for future requests.

Blood Request Process by Hospitals

Hospitals can request for blood by calling in or e-mailing the blood bank the type of blood and the quantity that is in need. The administrator is responsible for checking the availability of the blood type according to the request. If the requested blood type is available, the administrator will withdraw the blood from the inventory and transfer it to the hospital. However, if the requested blood is unavailable, the administrator will send an email to inform the hospital.

4. Requirement Analysis

There are two internal users involved in this system. The user requirements are considered as follows:

Donor

- To be able to view their donation records, including where and when they made donations, and the blood results for each, to learn of their donated blood quality and schedule their next donations.
- To be able to view and update their personal information, including name, contact address, and phone number, to keep their donor's information record up to-date with the blood bank

Administrator

- To be able to create, update, delete, and query donor's records in order to manage donor information.
- To be able to create, update, delete, and retrieve donation records to manage information about donations made.
- To be able to deposit donated blood into stock when donations are made.
- To be able to withdraw blood from the stock and keep a record of blood stocks to always keep count of the blood bags.
- To be able to create, update, delete, and retrieve request records from hospitals to manage hospital requests for blood
- To be able to create, update, delete, and query hospital's records in order to manage hospital information.
- To be able to send emails to donors for their user account and blood results through the system.
- To be able to send email responding to hospitals for their blood requests through the system.

5. Objectives

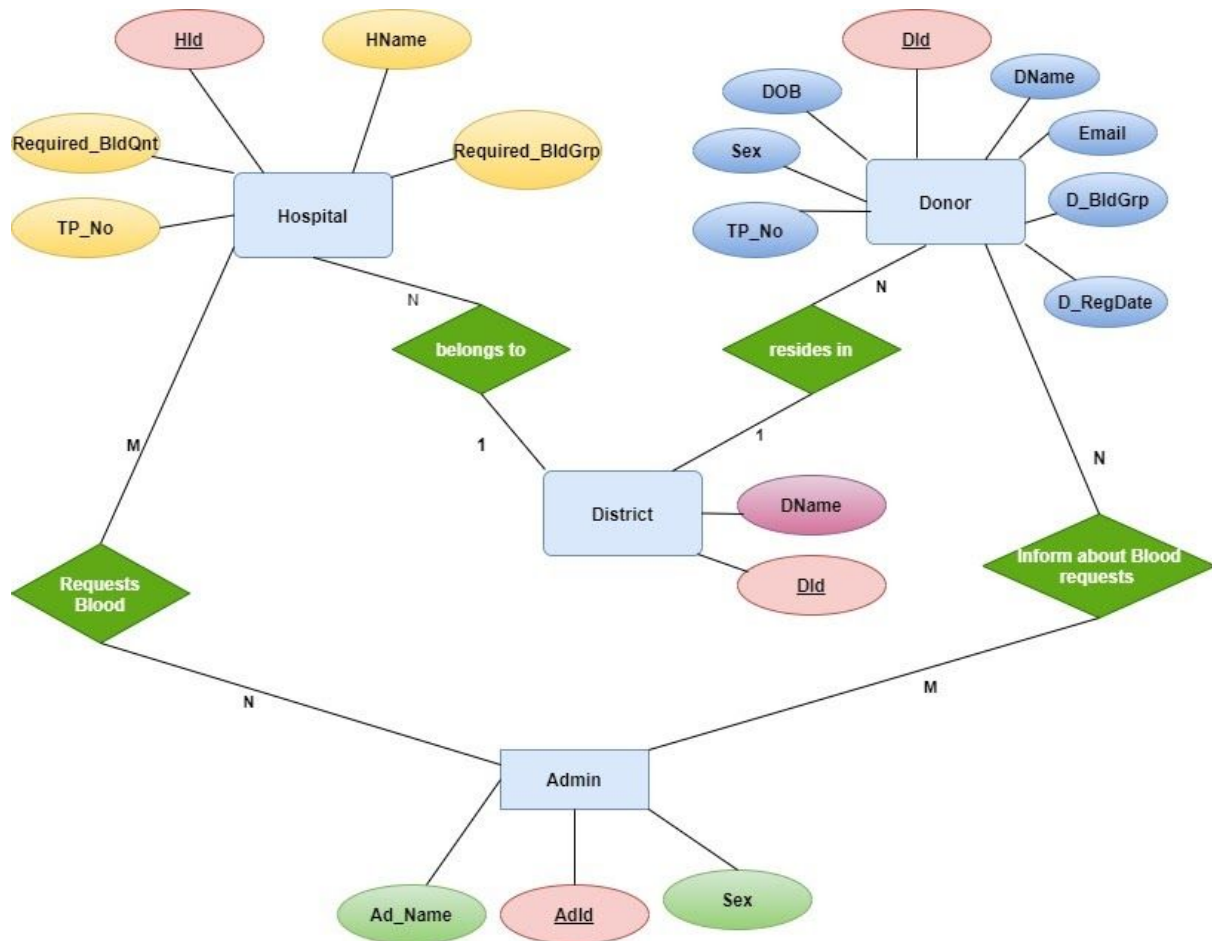
The goal of the project is to develop a web application for blood banks to manage information about their donors and blood stock. The main objectives of this website development can be defined as follows:

- To develop a system that provides functions to support donors to view and manage their information conveniently.
- To maintain records of blood donors, blood donation information and blood stocks in a centralized database system.
- To inform donors of their blood result after their donation.
- To support searching, matching and requesting for blood convenient for administrators.

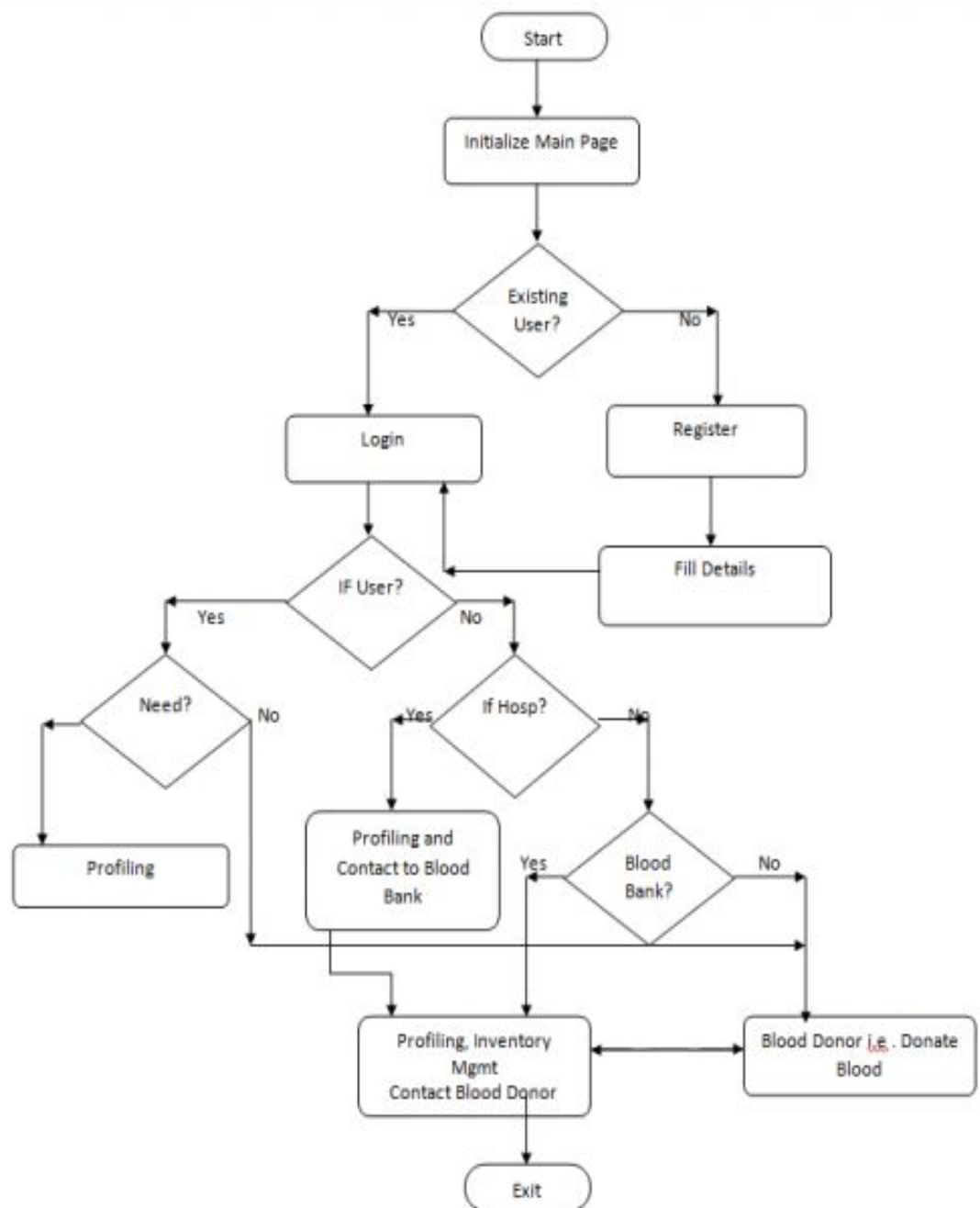
6. Design

6.1 Logical design of the proposed system

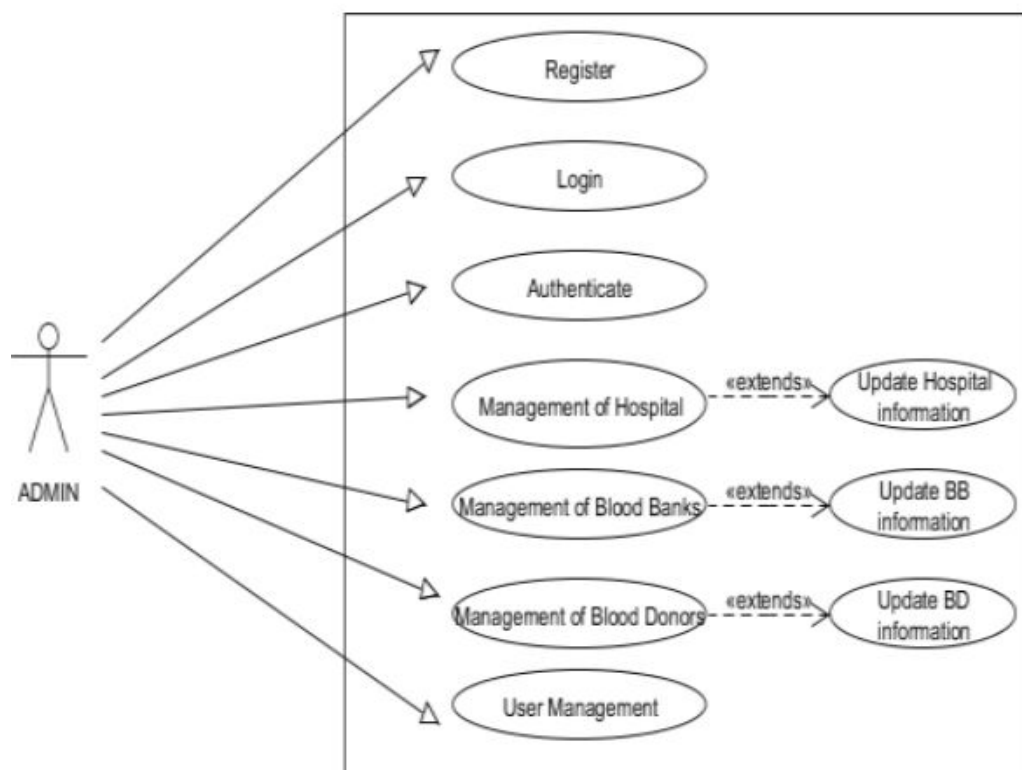
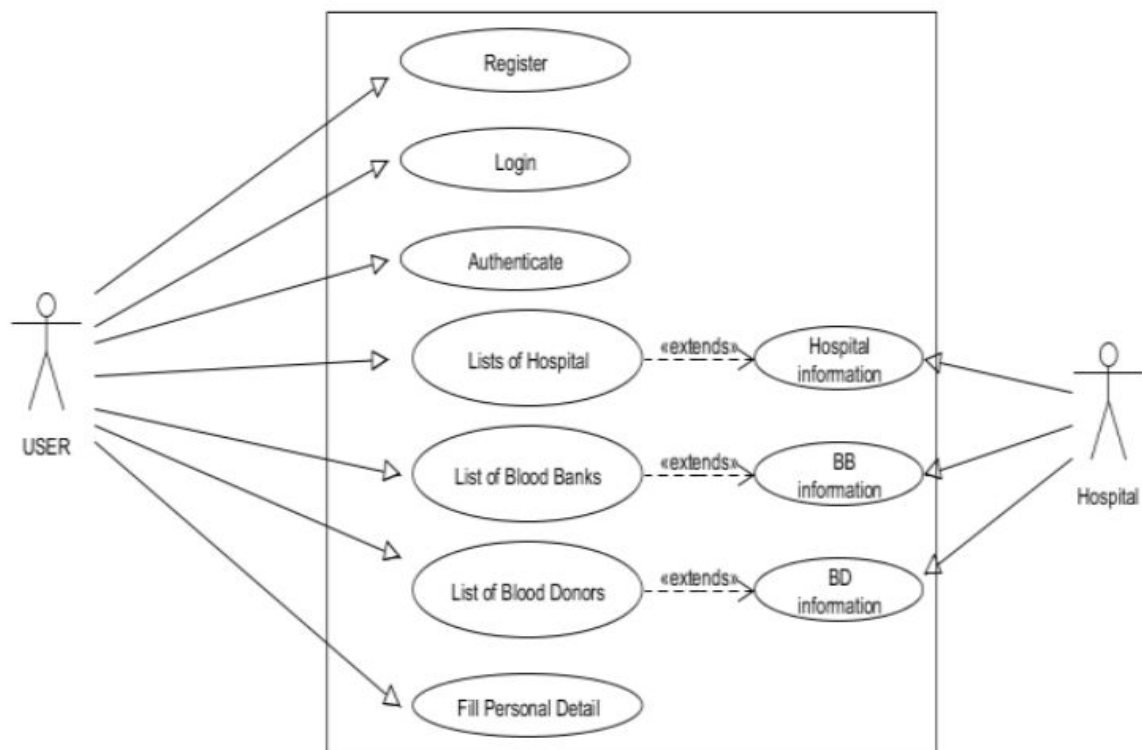
6.1.1 ER Diagram



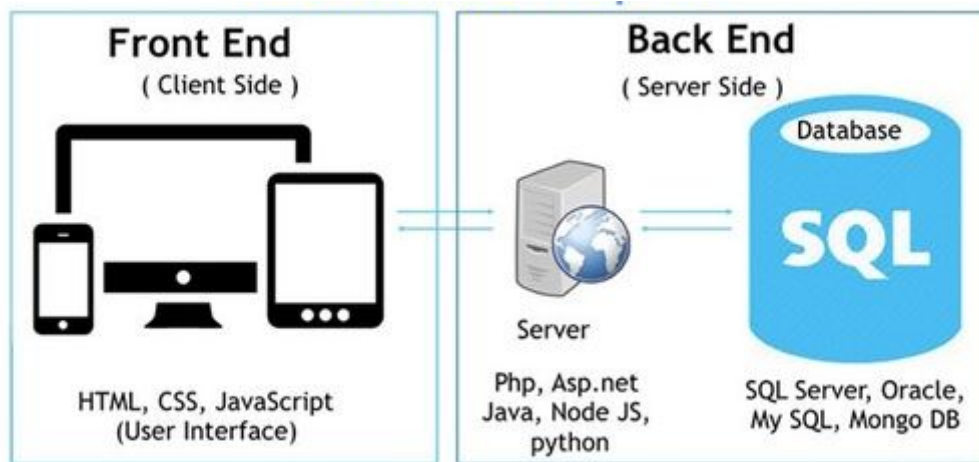
6.1.2 Flow chart



6.1.3 Use Case Diagrams



7. Implementation



7.1 Front-End Implementation

We have used **ReactJS** as the base in development of our web application which is known as an open-source JavaScript library for building user interfaces. And it is fast, scalable and simple.

Step 1: Installing Node and Visual Studio

Before creating a new React project, we should install NodeJS and Visual Studio in our machine.

Node.js is an open-source, cross-platform, JavaScript runtime environment that runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.) You can download NodeJs by using <https://nodejs.org/en/> this link. Then by executing **node --v** in the command prompt you can check whether it is installed successfully into the machine.

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. You can install by using <https://visualstudio.microsoft.com/vs/> this ink.

Step 2: Create React App and deploy it

After installing Node into the machine, to create a new app we should do following method:

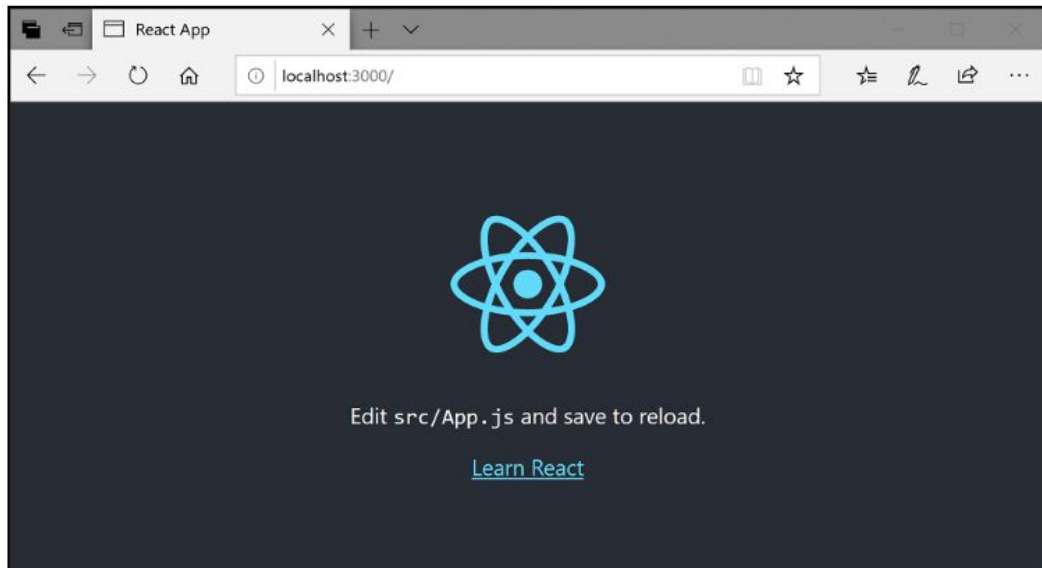
```
npx create-react-app my-app
```

It will create a directory called my-app inside the current folder. Inside that directory, it will generate the initial project structure and install the transitive dependencies.

Once the installation is done, you can open your project folder:

```
cd my-app  
npm start
```

Runs the app in development mode. Open <http://localhost:3000> to view it in the browser. The page will automatically reload if you make changes to the code. You will see the build errors and lint warnings in the console if there are.

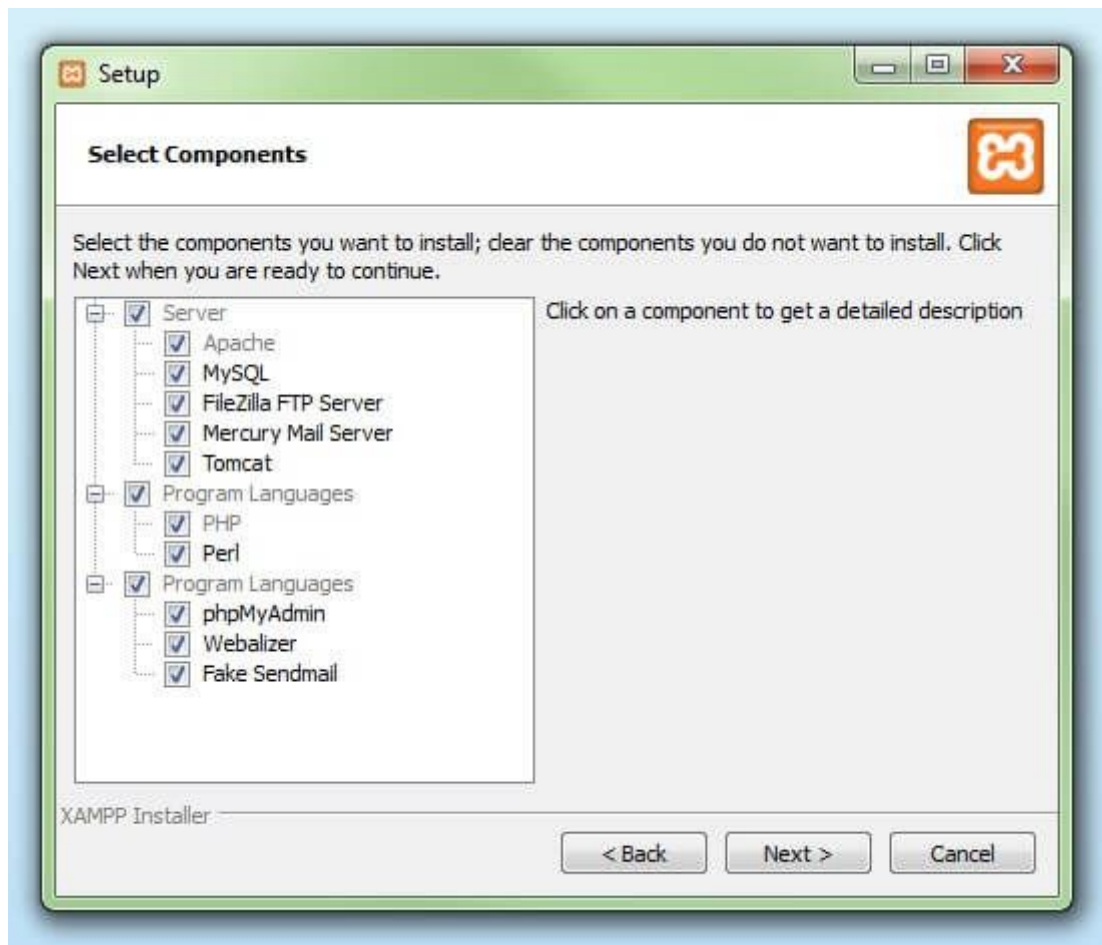


Now you can create folders, import dependencies and implement the user interface that we have previously designed.

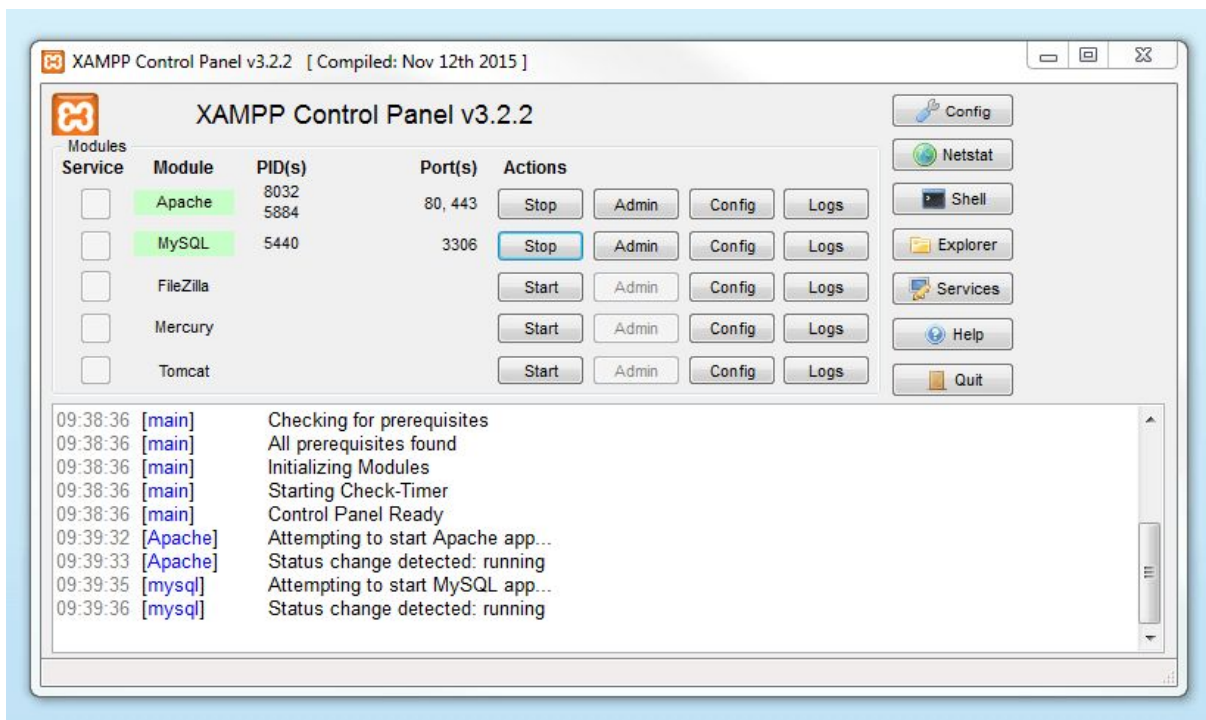
7.2 Back-End Implementation

In our project we have used a mysql database through xampp server. Before creating a database project we have to install an XAMPP in our local machine. XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy. <https://www.apachefriends.org/download.html> from this link we can download XAMPP according to our operating system. Then install your favorite apps on top of XAMPP. Bitnami provides a free all-in-one tool to install Drupal, Joomla!, WordPress and many other popular open source apps on top of XAMPP.

In the installation process we can choose which components to be installed. So there we can install mysql.




After installing XAMPP we will be getting a control panel like below. Then start running Apache and MySQL for implementing our database.



After setting up XAMPP we can open the dashboard of XAMPP's localhost.

Apache Friends
Applications
FAQs
HOW-TO Guides
PHPInfo
phpMyAdmin


XAMPP Apache + MariaDB + PHP + Perl

Welcome to XAMPP for Windows 5.6.15

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

Start the XAMPP Control Panel to check the server status.

Community






XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our Forums, adding yourself to the Mailing List, and liking us on Facebook, following our exploits on Twitter, or adding us to your Google+ circles.




Contribute to XAMPP translation at translate.apachefriends.org.

Can you help translate XAMPP for other community members? We need your help to translate XAMPP into different languages. We have set up a site, translate.apachefriends.org, where users can contribute translations.

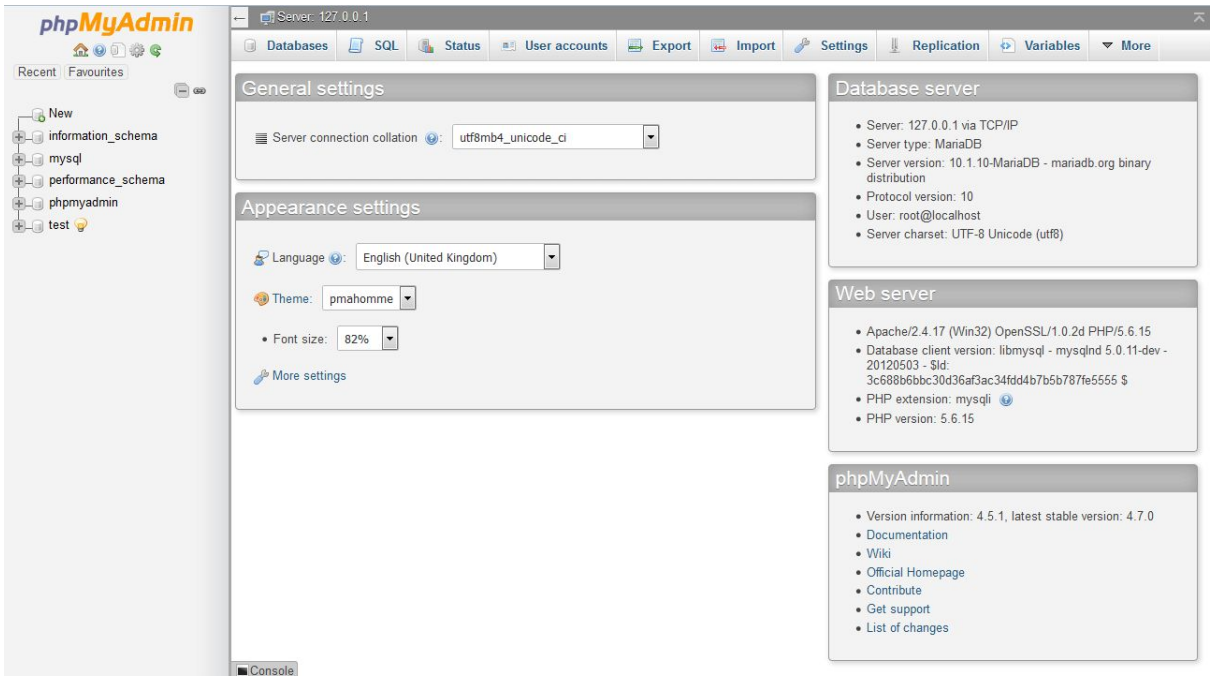
Install applications on XAMPP using Bitnami

Apache Friends and Bitnami are cooperating to make dozens of open source applications available on XAMPP, for free. Bitnami-packaged applications include Wordpress, Drupal, Joomla! and dozens of others and can be deployed with one-click installers. Visit the [Bitnami XAMPP page](#) for details on the currently available apps.

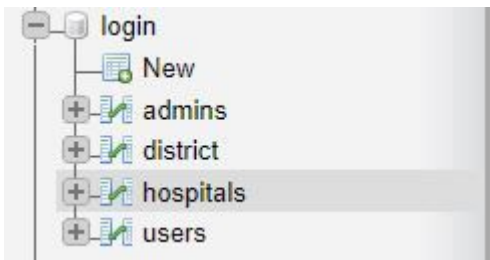



[Blog](#)
[Privacy Policy](#)
CDN provided by [fastly](#)
Copyright (c) 2015, Apache Friends

You can use the Admin button of your database module to open **phpMyAdmin**. Here, you can manage the databases of your web projects that you're testing on your XAMPP. Alternatively, you can reach the administration section of your MySQL database via localhost/phpmyadmin/.



Then we can create the required database by making tables and connect it with the web application and writing queries on it.

Tables are created as shown in below.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	ad_id	int(15)		No	None			Change Drop More
<input type="checkbox"/>	2	ad_name	text	utf16_croatian_ci	No	None			Change Drop More
<input type="checkbox"/>	3	sex	text	utf16_croatian_ci	No	None			Change Drop More
<input type="checkbox"/>	4	tp_no	text	utf16_croatian_ci	No	None			Change Drop More
<input type="checkbox"/>	5	password	text	utf16_croatian_ci	No	None			Change Drop More

+ Options						
		ad_id	ad_name	sex	tp_no	password
<input type="checkbox"/>	Edit Copy Delete	12564	wathsari	female	0710312706	\$2b\$10\$LkF9/YpSiQPVg9W77JiYOU7oSVhV0BAW6se44XzmcL...
<input type="checkbox"/>	Edit Copy Delete	45698	ishani	female	0765588751	\$2b\$10\$3QIm23Jye1lSu5jXv7KflujHODUzSSQ/niN0H3ws9fO...
<input type="checkbox"/>	Edit Copy Delete	564789	kshithija	female	0714568532	\$2b\$10\$WGES.4.DAH9uegStM9Z77esC70SFCqchXZGiWCGcXV7...

Figure : Admins table

Server: 127.0.0.1 » Database: login » Table: users											
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers											
+ Options											
		id	first_name	last_name	NIC	dob	sex	blood_group	tp_no	email	password
<input type="checkbox"/>	Edit Copy Delete	1	wathsari	premathilaka	956922577v	1995-10-07	female	A	710312706	wath@test.com	\$2b\$10\$Fum1lvi7ysfjo8rV
<input type="checkbox"/>	Edit Copy Delete	2	ishani	madhuwanthi	957844622v	1995-02-03	female	AB	725689754	ish@test.com	\$2b\$10\$3Y7uVDRWNs2r1t
<input type="checkbox"/>	Edit Copy Delete	3	kshithija	wanniarachchi	956899245v	1996-12-04	female	O+	765894232	kshithi@test.com	\$2b\$10\$kJ0jo.H1orAdFeJM
<input type="checkbox"/>	Edit Copy Delete	4	mihiri	premathilaka	9356887441v	1993-04-03	female	O-	702122618	mihiri@test.com	\$2b\$10\$WjPdXdfzTDehHig
<input type="checkbox"/>	Edit Copy Delete	5	piyuma	kavinda	987544224v	1998-11-01	male	AB	785698852	piyu@test.com	\$2b\$10\$LkF9/YpSiQPVg9V
<input type="checkbox"/>	Edit Copy Delete	6	chamini	kavindi	200055445v	2000-01-15	female	A	768524569	chami@test.com	\$2b\$10\$3QIm23Jye1lSu5j
<input type="checkbox"/>	Edit Copy Delete	7	nuwan	nirmala	902575841v	1990-08-02	male	O-	716205687	nuwan@test.com	\$2b\$10\$VnEWGQgHko38ly
<input type="checkbox"/>	Edit Copy Delete	8	bilani	somarathna	945689255v	1994-04-02	female	AB	714562369	bilani@test.com	\$2b\$10\$39zPYyHQ26rbm6
<input type="checkbox"/>	Edit Copy Delete	31	vishaka	perera	9587466522v	1995-02-05	female	A	714589357	vishaka@gmail.com	\$2b\$10\$WGES.4.DAH9ueg
<input type="checkbox"/>	Edit Copy Delete	32	dilini	madumali	958677422v	1995-08-12	female	A	768542395	dilini@test.com	\$2b\$10\$LPqThiUsdH7xyS.

Figure : Users table

Server: 127.0.0.1 » Database: login » Table: hospitals

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Table structure

Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	h_id	int(11)			No	None			Change Drop More
<input type="checkbox"/> 2	hospital_name	text	utf32_croatian_ci		Yes	NULL			Change Drop More
<input type="checkbox"/> 3	district	text	utf32_croatian_ci		Yes	NULL			Change Drop More
<input type="checkbox"/> 4	d_id	int(11)			No	None			Change Drop More
<input type="checkbox"/> 5	blood_type	text	utf16_croatian_ci		No	None			Change Drop More
<input type="checkbox"/> 6	blood_amount	int(11)			No	None			Change Drop More
<input type="checkbox"/> 7	Contact number	text	utf16_croatian_ci		No	None			Change Drop More

☐ Show all | Number of rows:

25

 | Filter rows:

Search this table

 | Sort by key:

None

+ Options

		h_id	hospital_name	district	d_id	blood_type	blood_amount	Contact number
<input type="checkbox"/>	Edit Copy Delete	1	Ampara General Hospital	Ampara	1	A+	10	0375784073
<input type="checkbox"/>	Edit Copy Delete	2	Kurunegala General Hospital	Kurunegala	14	B+	12	0375756273
<input type="checkbox"/>	Edit Copy Delete	3	Anuradhapura Teaching Hospital	Anuradhapura	2	O-	2	0375784856
<input type="checkbox"/>	Edit Copy Delete	4	Kandy Teaching Hospital	Kandy	11	AB	3	0815784856
<input type="checkbox"/>	Edit Copy Delete	5	Mannar District Hospital	Mannar	15	B+	5	0366584856
<input type="checkbox"/>	Edit Copy Delete	6	Mathale District Hospital	Mathale	16	O+	6	0369874123
<input type="checkbox"/>	Edit Copy Delete	7	Jaffna District Hospital	Jaffna	15	AB	5	0366698756
<input type="checkbox"/>	Edit Copy Delete	8	Gampaha District Hospital	Gampaha	7	A-	4	03865789541

☐ Check all

With selected:

Edit Copy Delete Export

Figure : Hospital table

Server: 127.0.0.1 » Database: login » Table: district

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)
[Export](#)
[Import](#)
[Privileges](#)
[Operatic](#)

[Table structure](#)
[Relation view](#)

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	d_id	int(11)			No	None			Change Drop More
<input type="checkbox"/> 2	district_name	text	utf16_croatian_ci		No	None			Change Drop More

Server: 127.0.0.1 » Database: login » Table: district

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)

+ Options

				d_id	district_name
<input type="checkbox"/>	Edit	Copy	Delete	1	Ampara
<input type="checkbox"/>	Edit	Copy	Delete	2	Anuradhapura
<input type="checkbox"/>	Edit	Copy	Delete	3	Badulla
<input type="checkbox"/>	Edit	Copy	Delete	4	Batticalo
<input type="checkbox"/>	Edit	Copy	Delete	5	Colombo
<input type="checkbox"/>	Edit	Copy	Delete	6	Galle
<input type="checkbox"/>	Edit	Copy	Delete	7	Gampaha
<input type="checkbox"/>	Edit	Copy	Delete	8	Hambanthota
<input type="checkbox"/>	Edit	Copy	Delete	9	Jaffna
<input type="checkbox"/>	Edit	Copy	Delete	10	Kaluthara
<input type="checkbox"/>	Edit	Copy	Delete	11	Kandy
<input type="checkbox"/>	Edit	Copy	Delete	12	Kegalle
<input type="checkbox"/>	Edit	Copy	Delete	13	Kilinochchi
<input type="checkbox"/>	Edit	Copy	Delete	14	Kurunegala
<input type="checkbox"/>	Edit	Copy	Delete	15	Mannar
<input type="checkbox"/>	Edit	Copy	Delete	16	Matale
<input type="checkbox"/>	Edit	Copy	Delete	17	Matara
<input type="checkbox"/>	Edit	Copy	Delete	18	Monaragala
<input type="checkbox"/>	Edit	Copy	Delete	19	Mullativu
<input type="checkbox"/>	Edit	Copy	Delete	20	Nuwaraeliya
<input type="checkbox"/>	Edit	Copy	Delete	21	Deleggeruwa

[Console](#)

Figure : District table

Here, our BBMS - Blood Bank Management System is a multi user database system. In our system there are different users and their access roles are also different. (i.e donors and administrators)

7.3 Building connection between front-end and Back-end

```
const Sequelize = require('sequelize')
const db = {}
const sequelize = new Sequelize('login', 'root', '', {
  host: 'localhost',
  dialect: 'mysql',
  operatorsAliases: false,

  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000
  }
})

db.sequelize = sequelize
db.Sequelize = Sequelize

module.exports = db
```

```

1  var express = require('express')
2  var cors = require('cors')
3  var bodyParser = require('body-parser')
4  var app = express()
5  var port = process.env.PORT || 5000
6
7  app.use(bodyParser.json())
8  app.use(cors())
9  app.use(
10     bodyParser.urlencoded({
11         extended: false
12     })
13 )
14
15 var Users = require('./routes/Users')
16
17 app.use('/users', Users)
18
19 app.listen(port, function() {
20     console.log('Server is running on port: ' + port)
21 })

```

Then we can run the server as shown in below.

```

> login@1.0.0 start E:\Academic\6th Semester\login
> node server.js

Server is running on port: 5000

```

To run the front-end;

```

> client@0.1.0 start E:\Academic\6th Semester\login\client
> react-scripts start

Starting the development server...

```

Here we have used mysql-sequelize Package to connect the database instead of mysql - mysql package. Following are some examples for the queries we have used in database implementation.

ex:

```
users.post('/login', (req, res) => {  
  User.findOne({  
    where: {  
      email: req.body.email  
    }  
  })  
})
```

This part is equal to **"SELECT * FROM users WHERE email = email.body"** in normal query

```
User.findOne({  
  where: {  
    id: decoded.id  
  }  
})
```

This part is equal to **"SELECT * FROM User WHERE id = decoded.id"** in normal query

Features of the database

For the better performance of the database indexes are used. A good transaction processing is done to maintain the database integrity in the database. Otherwise when multiple users are accessing the same at the same time it will lead to inconsistent state of the database. And also the query optimization is also done to get the better performance of the system. As system security the BBMS has introduced user authentication features.

8. Comparison between SQL and NoSQL databases

In our system we use MySQL database. It is a SQL base database. Even though we have used SQL databases, there are also no SQL databases. Lets compare the SQL and no SQL database models.



- SQL databases are primarily called as Relational Databases (RDBMS); whereas NoSQL databases are primarily called as non-relational or distributed databases.
- SQL databases are table based databases whereas NoSQL databases are document based, key-value pairs, graph databases or wide-column stores. This means that SQL databases represent data in form of tables which consist of n number of rows of data whereas NoSQL databases are the collection of key-value pairs, documents, graph databases or wide-column stores which do not have standard schema definitions which it needs to adhere to.
- SQL databases have predefined schema whereas NoSQL databases have dynamic schema for unstructured data.
- SQL databases are vertically scalable whereas the NoSQL databases are horizontally scalable. SQL databases are scaled by increasing the horse-power of the hardware. NoSQL databases are scaled by increasing the databases servers in the pool of resources to reduce the load.
- SQL databases use SQL (structured query language) for defining and manipulating the data, which is very powerful. In NoSQL databases, queries are focused on collection of documents. Sometimes it is also called UnQL (Unstructured Query Language). The syntax of using UnQL varies from database to database.

- SQL database examples: MySQL, Oracle, SQLite, Postgres and MS-SQL. NoSQL database examples: MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb.
- For complex queries: SQL databases are good fit for the complex query intensive environment whereas NoSQL databases are not good fit for complex queries. On a high-level, NoSQL don't have standard interfaces to perform complex queries, and the queries themselves in NoSQL are not as powerful as SQL query language.
- For the type of data to be stored: SQL databases are not best fit for hierarchical data storage. But, NoSQL databases fit better for hierarchical data storage as it follows the key-value pair way of storing data similar to JSON data. NoSQL databases are highly preferred for large data sets (i.e for big data). Hbase is an example for this purpose.
- For scalability: In most typical situations, SQL databases are vertically scalable. You can manage increasing load by increasing the CPU, RAM, SSD, etc, on a single server. On the other hand, NoSQL databases are horizontally scalable. You can just add a few more servers easily in your NoSQL database infrastructure to handle the large traffic.
- For high transactional based application: SQL databases are best fit for heavy duty transactional type applications, as it is more stable and promises the atomicity as well as integrity of the data. While you can use NoSQL for transaction purposes, it is still not comparable and stable enough in high load and for complex transactional applications..

9. Interface Designing



Figure : Home page

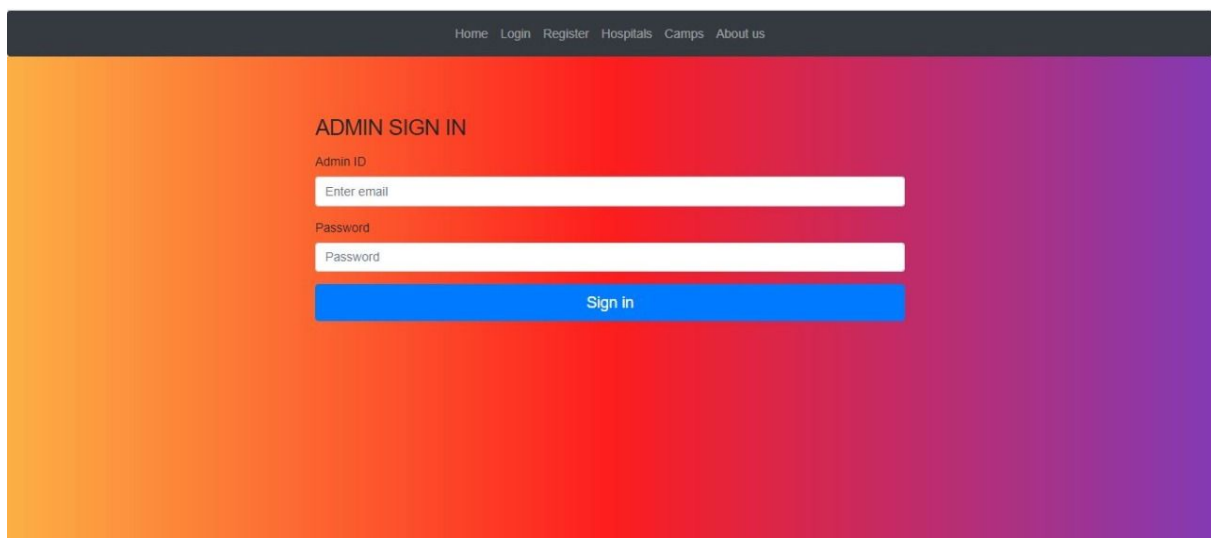


Figure : Admin sign-in page

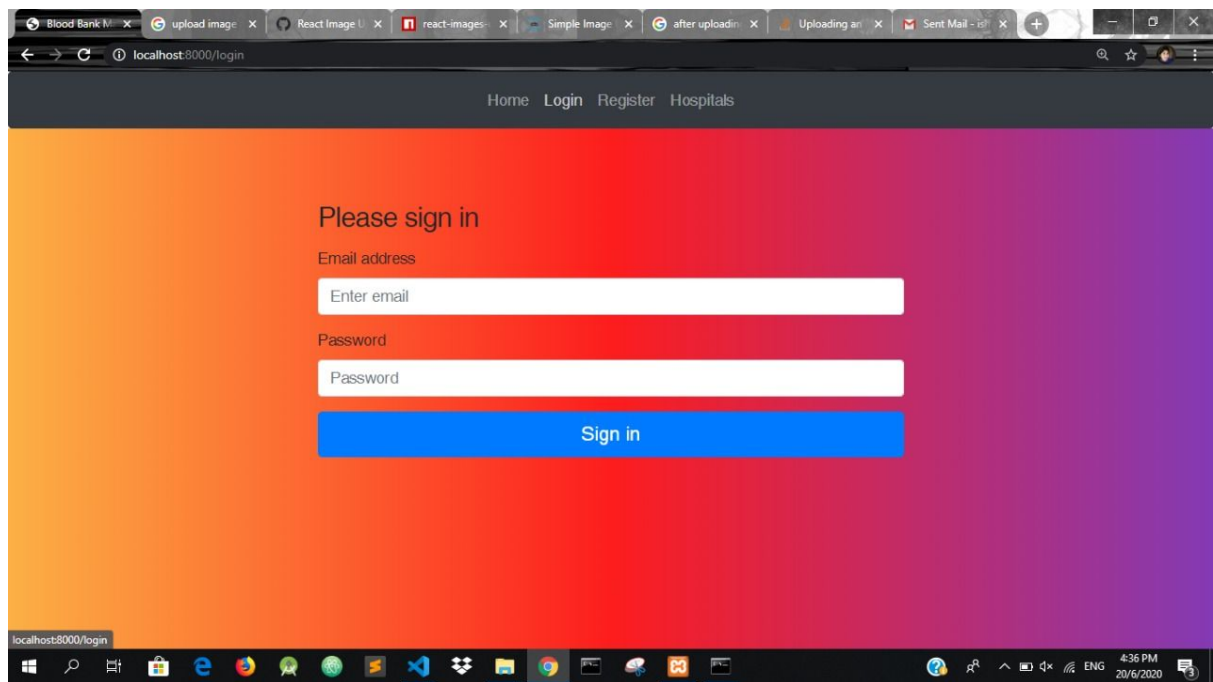


Figure : Donor sign-in page

After sign-in to the system they can view their profile in the system.

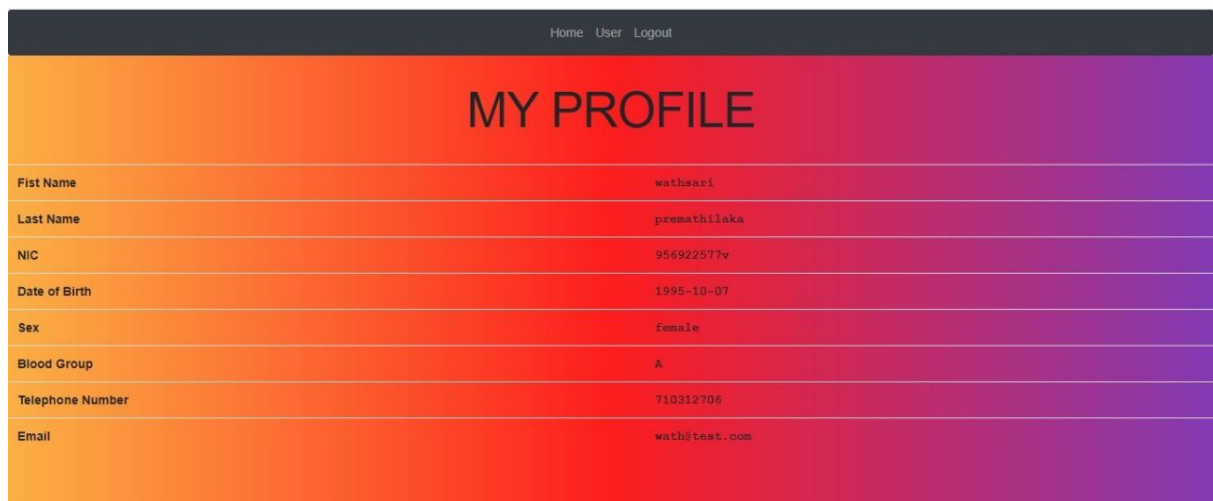


Figure : My profile page

Home Login Register Hospitals Camps About us

Register

First name
Enter your first name

Last name
Enter your lastname

NIC
Enter your NIC

Date of Birth
mm/dd/yyyy

Sex

Blood Group
Enter your Blood Group

Telephone Number
Enter your Telephone number

Email address
Enter email

Password
Password

Figure : Donor Registration page

Home Login Register Hospitals

WELCOME!

Blood Bank Management System
Under the authority of Ministry of Health Department

Services we provided

1. Donors can donate bloods in blood donation centers of your area.
2. Recipients can request for bloods with the permission of any hospital.
3. Blood donation campaigns can be organized in your area, firms by the participation of responsible persons.(doctors/nurses)
4. You can contact us in any problem using our department contacts.

Branches we have across the country

- >>Colombo
- >>Kurunegala
- >>Gampaha

Figure : About us page

10. Testing

Testing is essential for checking the BBMS(Blood Bank Management System) for potential bugs before it is made live and is accessible to the general public. Web Testing checks for functionality, usability, security, compatibility, performance of the web application or website.

Functionality Testing

- testing all the links in the web page
outgoing links, internal links, mail to links ,anchor links
- Testing forms whether they work as expected
Scripting checking , checking where default values are populated, submitting process,

Usability Testing

- Test the site Navigation
Menus, buttons or Links to different pages on the site should be easily visible and consistent on all webpages
- Test the content
Content should be legible with no spelling or grammatical errors

Interface Testing

Three areas to be tested here are - Application, Web and Database Server

- **Application:** Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the application and must be only shown to the administrator and not the end user.
- **Web Server:** Test Web server is handling all application requests without any service denial.
- **Database Server:** Make sure queries sent to the database give expected results

Database Testing

Database is one critical component of the web application and stress must be laid to test it thoroughly. Testing activities will include-

- Test if any errors are shown while executing queries
- Data Integrity is maintained while creating, updating or deleting data in the database.
- Check response time of queries and fine tune them if necessary.
- Test data retrieved from your database is shown accurately in your web application

Performance Testing

This will ensure the site works under all loads. Software Testing activities will include but not limited to -

- Website application response times at different connection speeds
- Load test the web application to determine its behavior under normal and peak loads
- Stress test your web site to determine its break point when pushed to beyond normal loads at peak time.
- Test if a crash occurs due to peak load, how does the site recover from such an event
- Make sure optimization techniques like gzip compression, browser and server side cache enabled to reduce load times

Security Testing

- Test unauthorized access to secure pages should not be permitted
- Restricted files should not be downloadable without appropriate access
- Check sessions are automatically killed after prolonged user inactivity
- On use of SSL certificates, websites should redirect to encrypted SSL pages.

Crowd Testing

This is a multi user database system. Therefore multiple users with different access roles are there. The large number of people may access the database and execute the queries. This should be done without any defect. Hence this should be tested.

11.Conclusion

Technology is introducing new innovations day by day, thus reducing the time required to do things. The proposed system can be used to reduce the time required to deliver required blood to the needy in cases of emergency. All in all we hope that the Sri Lankan people benefit from our system. We hope all the information we get is what they want. Actually, all the challenges and problems that faced us was nothing. We say that after we finished our project. We recommend hospitals, customers and employment in hospitals to use this system which will help them to get the most details about the blood bank management system. Our system is beneficial for spending less effort and time on blood donation.