****

**Cloud Computing**

**Group 15 – Phase 1**

**Blood Banking System**

|  |  |  |
| --- | --- | --- |
| ID | Student Name | SRS Group No. |
| 196735 | Mahmoud Assem Abdelkhalek | A3 |
| 196318 | Mohamed Taie | A1 |
| 203398 | Maiada | A3 |
| 195948 | Abdelrahman Hagrass | A1 |

Table of Contents

[Project description 3](#_Toc118119026)

[Constraints 3](#_Toc118119027)

[Services and their functions 4](#_Toc118119028)

[Laboratory 4](#_Toc118119029)

[Blood bank inventory: 4](#_Toc118119030)

[Administration 4](#_Toc118119031)

[Management 4](#_Toc118119032)

[Donation 4](#_Toc118119033)

[Work Plan 5](#_Toc118119034)

[Solution Methodology 5](#_Toc118119035)

[Project Components & Cloud service model 5](#_Toc118119036)

[Cloud Deployment Model: 6](#_Toc118119037)

[Cloud layers management approach 6](#_Toc118119038)

[Tenants 6](#_Toc118119039)

[Application layer: 6](#_Toc118119040)

[App data layer: 6](#_Toc118119041)

[Operating system layer: 7](#_Toc118119042)

[Hardware layer 7](#_Toc118119043)

[Tools and Technologies 7](#_Toc118119044)

# Project description

Blood Banking Management System is a web system that connects the doctors in a specific hospital with the blood bank laboratory within the hospital. The system will have a super-admin that will be responsible for registering and managing the hospitals within the system. The blood bank (in each hospital) has its own Lab Admin and Lab Manager.

The Lab Admin is responsible for requesting to insert new blood bags with its own tests to the blood bank laboratory. The Lab Manager is responsible for analyzing the information of the blood bags requested to be inserted and makes the decision of accepting to add it to the blood bank if it is valid or not. Doctors within the hospital are responsible for inserting new patients to the system and requesting blood bags on behalf of their patients. The Lab Managers are responsible for accepting the requests made by the doctors and allocating the required blood bag for them.

On the other side of the system, Donors can register an account on the system to reserve a time slot to donate blood to any of the registered hospital. Donors are to find the closest hospital to them and visit them for donation.

# Constraints

* The system Super Admin does not have access to view information about each hospital’s blood bank inventory itself.
* Each Lab Manager can view their hospital’s blood bank inventory.
* Super Admins are responsible for creating accounts for the Lab Manager to represent their hospital.
* Lab Managers are responsible for creating accounts for the Lab Admins.
* Lab Managers are responsible for creating accounts for the Doctors within the hospital.
* The Doctors are responsible for requesting blood bags on behalf of the patients in the hospital.
* Donors must be registered and submitted their medical profile before reserving a time slot to donate blood in.
* The Lab Admin must insert all information about the blood bag before making the request.

# Services and their functions

## Hospital (hospital schema)

* + - **sendNotif**

(Lab manager) Notify Super Admin that the blood bank is running low from a certain blood type

* + - **manageHospital**
    - **GenerateDonationsReport**

(Super admin) View dashboards & reports of Donors gender, donations per year

## BloodbankInventory (bloodbags scehma)

* + - **InsertBagReq**  
      (Lab admin) insert blood bags details (test results) into the hospital’s blood bank.
    - **ViewBagReq**

(lab manager) View Blood Bag Admission Requests.

* + - **ModifyBagReq**

(Lab admin) modify blood bag admission request details.

* + - **AcceptBagReq**

(Lab manager) accepts / rejects blood bad admission requests after analyzing their test results.

* + - **ViewHospitalInventory**

(Lab manager) View dashboards & reports of blood supply in his hospital

## Patients: (patient schema)

* + - **reqBloodBag**

(Doctor) Request blood bags on behalf of the patients.

* + - **viewBagReq**

(Doctor) view blood bag requests.

* + - **modifyBagReq**

(Doctor) modify blood bags requests

* + - **acceptBagReq**

(Lab manager) Accept the blood bag requests from doctors within the hospital.

## Donation

* + - (Donor) **ViewDonationHistory**.
    - (Donor) **ReserveTimeSlot**.

## UserAccount

* + - (Super admin) Create Lab Manager account to represent the hospital within the system.
    - (Lab manager) Creating Lab Admin accounts to be responsible for inserting blood bag admission requests.
    - (Lab manager) Creating Doctor accounts to handle patient requests.
    - (Doctor) Insert new patients
    - (Doctor) Manage patients
    - (Donor) Register Account with medical profile info.
    - (Donor) Manage Account details

# Work Plan

|  |  |
| --- | --- |
| **Name** | **Service Assigned** |
| **Mahmoud** | * **BloodBankInventory Service** (ALL) * **UserAccount Service**   (Lab Admin inserts blood bag insertion request to be approved by the Lab Manager. Lab Admin is responsible to insert all lab tests for the blood bag that they are requesting to add to the inventory.) |
| **Mohamed** | * **Donation service** (ALL) * **UserAccount Service**   ((Doctor) Insert new patients and (Doctor) Manage patients) |
| **Maiada** | * **Hospital service** (ALL) * **UserAccount Service**   (Super admin creates Lab Manager account to represent the hospital within the system.) |
| **Abdelrahman** | * **Patient service** (ALL) * **UserAccount Service**   ((Lab manager) Creating Lab Admin accounts to be responsible for inserting blood bag admission requests.) |

# Solution Methodology

## Project Components & Cloud service model

We are ***building a SAAS product*** that serves users (sysSuperAdmin – Hospitals Lab Managers – Hospitals Lab Admins – Hospitals Doctors - Donors), that will be ***deployed on Microsoft Azure***.

Our system components are mainly back-end and front-end. For the back end we are building a ***REST API*** backend server to store and retrieve the data from our cloud deployed **Mongo DB server**. For the front-end there will be a ***React web application*** acting as a GUI for our system's end-users (sysSuperAdmin – Hospitals Lab Managers – Hospitals Lab Admins – Hospitals Doctors - Donors).

The 2 components are communicating via HTTP requests (GET for requesting data, POST for inserting data, PUT for modifying data, and DELETE for removing data). We will be using **Express** for handling those requests and return JSON data as a response.

For automatic restart of the application, to maintain having updated data without the need to refresh, we will be using ***Nodemon*** package.

Our app is initially connected to the DB with the aid of ***Mongoose*** package. We chose to use the ***MVC Layered architecture*** model that mainly consist of a “Router”, “Controller”, “Service” and a “Model” to build our REST APIs.

Besides the APIs mentioned above, we are going to be ***using cloud services*** that are ***based on PAAS like Microsoft Azure Maps*** for the donors to find the nearest blood donation camp for hospital to donate blood there, ***Google Cloud Platform based on PaaS along with NodeMailer API*** to send confirmation emails between the users (Accepting, refusing blood bag insertion requests performed by Lab Admins – Notifying Donors), and ***Microsoft PowerBI external API*** to embed the needed dashboards in our web-application.

## Cloud Deployment Model:

The deployment of the application as a whole will be ***private*** since that the system contains ***sensitive data*** about the users that use the system, and it also carries information about the blood bags donated by donors using the system.

## Cloud layers management approach

### Tenants

* + sysSuperAdmin
  + Hospitals Lab Managers
  + Hospitals Lab Admins
  + Hospitals Doctors
  + Donors

### Application layer:

* **Consumer**: End Users
* **Provider**: Our System
* **Administrative**: Our System
* **Owner**: Our System

### App data layer:

* **Consumer**: End Users
* **Provider**: Our System
* **Administrative**: Our System
* **Owner**: Our System

### Operating system layer:

* **Consumer**: Our System
* **Provider**: Microsoft Azure
* **Administrative**: Microsoft Azure
* **Owner**: Microsoft Azure

### Hardware layer

* **Consumer**: Our System
* **Provider**: Microsoft Azure
* **Administrative**: Microsoft Azure
* **Owner**: Microsoft Azure

Tools and Technologies

* + Visual Studio Code
  + REACT
  + JavaScript
  + JSON
  + REST API
  + Express
  + MongoDB
  + Microsoft Azure