**Requirements Document**

**For blood bank management system**

**The introduction:**

A blood bank management system is a software system that helps to manage the blood bank's activities, including blood donation, blood testing, blood storage, blood distribution, and blood inventory management. The system is designed to improve the efficiency of blood bank operations and ensure timely access to blood when needed. The purpose of this requirements document is to describe the requirements for the development of a blood bank management system.

**The Fonctionnel Requirements :**

1.**Regestration**:

Regular users can sign in either as donators or receivers

and hospitals may like to use our system by signing in only

one manager and possible many doctors, managers can

add doctors and regular users even if they are not signed,

doctors in other hand can add only when adding new

interaction in case user does not exist.

2.**Inventory Management**:

The system should allow hospital managers to change

bank max and min capacity for each blood type and also

update current bank capacity.

3.**Blood exchanges or interactions:**

Regular users and doctors can see their history of

Interactions. Doctors can add an interaction that they are a

part of while managers can add and delete any interaction in

their respective hospital.

Interaction increase or decrease the blood bank capacity.

4.**Blood Requests:**

Both managers and regular users can exchange requests

by process of creating, modifying, sending they can however

delete before sending it then the request passes four

different states: pending “the recipient didn’t see or respond

to the request” then either getting accepted or rejected by the

recipient then it get fulfilled meaning it became an interaction.

Request may be for taking or receiving blood at specific date.

5.**Managing Profile:**

Each user can edit his profile information in addition

regular user Have two health status attributes (check out

class diagram in the end) one he can edit freely and another

one that represents medical notes written by doctors or

hospital managers after a specific interaction with that user

also regular users can download a pdf about their past blood

exchange with visual representation.

6.**Managers Advance Search:**

Each hospital manager can search for receivers or donators

or doctors that work in his hospital and filter the results, he

can check their profiles and past interactions, doctors can

also search do the same but only for regular users.

**The Non Fonctionnel Requirements :**

1**.Securitry :**

We need to secure users authorization from attacks like

Xss and Csrf or general attack like man in middle attack by

using secure tls or ssl connections that also apply to

database connections.

**2.Performance:**

The site need to be able to handle large number of users

while the response time is relatively short furthermore the

user interface interactivity should take in consideration low

level devices.

**3.Usability:**

The site needs to respect user experience rules from

fast and easy navigation to clear layouts all that to give

smooth usage of the product.

**4.User Interface:**

The site needs to have good looking user interface

Integrating new technologies and components.

**5.Compatibilty:**

The system should be compatible with different hardware

and software platforms and support different browsers.

**The General Constraints:**

1.The site should be developed withing three months with

limited communication between developers.

2.The site is developed using limited hardware and software

support.

3.The site should be developed on English with possibility to

add more languages in the future.

4.The development cycle is iterative incremental with

possibility of adding new features throw out the process.

**System Architecture:**

1. We endure client-server architecture were front-end

and back-end code are separate and communicate using

apis.

2.Client side should have home page to describe the

website and different dashboards for regular-user

(donator/receiver), doctor, manager.

3.Server side need to have different api end-points

(controllers) for authentication, users-crud operations.

4.The used database either sql or nosql need to follow

the class diagram.

**Testing Requirements:**

We mainly focusing on end-to-end test because limited

time to do unit and integration test. This test will be after

working on one of the functional requirements mentioned

above both on front-end and back-end, if test didn’t go well,

we need to refactor our code to work as intended.