**Low-Level Architecture and Data Models**

**<P11>:<Blood Donation System>**

**<team member names & ids>**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **22100221** | **Ahmad Aslam** |
| **22100224** | **Muhammad Bilal** |
| **22100082** | **Saad Azam** |
| **22100337** | **Shehroz Atir** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Architecture diagram | 30 | 30 |
| Architecture justification | 20 | 15 |
| E/R diagram | 30 | 20 |
| E/R diagram description | 20 | 5 |
| Late submission |  |  |
| **Total** | **100** | **70** |
| **Individual Evaluation** |  |  |
| Muhammd Bilal |  | 30 |
| Shehroz Atir |  | 45 |

**Table of Contents**

1. Introduction 3
2. System Architecture 4
   1. Architecture Diagram—As it is in the prototype code 4
   2. Architecture Diagram—As it should be 7
3. Data Models 9

Description: 10

1. Tools and Technologies 10
2. Who Did What? 13
3. Review checklist 13

# Introduction

Blood Donating System is an Android, iOS and web-based app that provides a platform to blood donors and receivers for successful blood donation. It is going to make blood donation accessible to the one in need. The potential users of the app are blood donors, NGOs and blood recipients.

The overall objectives of the app are:

* Expedite the process of blood donation.
* Encourage NGOs to provide blood donors to the system.
* Through a system of ratings, points and rewards, encourage more and more donors to use the app.
* Provide an easy communication between blood donors and blood receivers.
* Make blood donation easily accessible to the one in need.

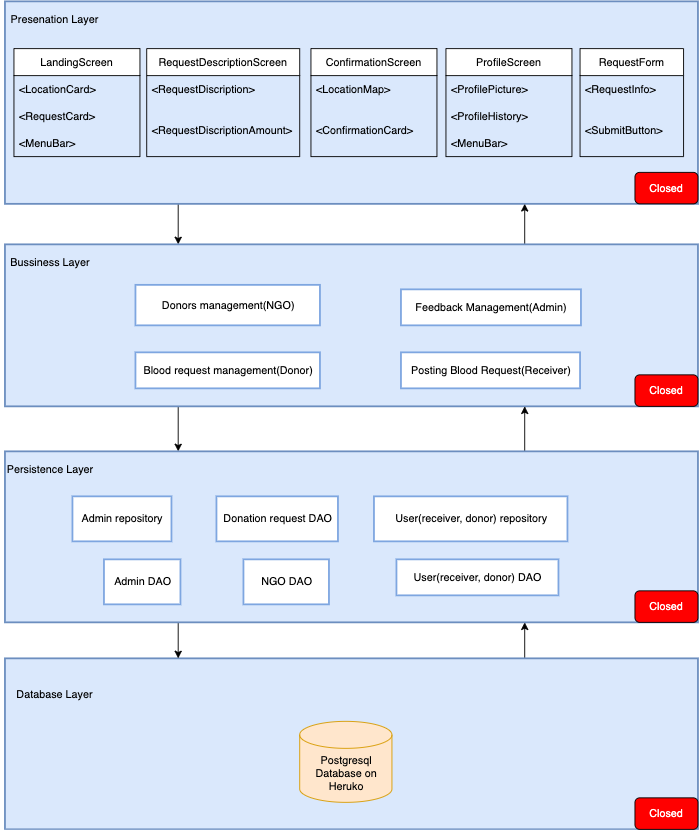
In the app, a recipient is going to ask for a donation of blood. The request will be pushed to a newsfeed where a donor will willingly respond to the request. A chat module will open between the donor and the receiver where they can communicate regarding the blood donation. On successful blood donation, the donor will be rewarded with points.

The app is going to be android, iOS and web-based. The main technologies that we are going to use for our app are Ruby on Rails, React and React Native.

# System Architecture

Where did you find the convention of using closed layers?

## Architecture Diagram—As it is in the prototype code



**Description (As in Prototype):**

**Closed layers:**

  A closed layer means that as a request moves from layer to layer, it must go through the layer right below it to get to the next layer below that one. For example, a request originating from the presentation layer must first go through the business layer and then to the persistence layer before finally hitting the database layer.

**Presentation Layer:**

Presentation layer is the top most layer of our layered architectural pattern. As the prototype consists of subsets of the use-cases so, we only have 4 screens(modules) in the presentation layer.

We have implement this layer in our project using react/react native.

**Note:** one module consist of different individual components. Exact name of the components are mention inside <> brackets.

**Business Layer:**

Business layer is the second layer of our layered architectural pattern. In our case, it contains some basic functionalities such as feedback management and donor management etc.

We implement the logic of system in this layer. For example, if donor wants to join NGO, it will be selected on basis of its health reports. So, logic of donor selection will be implemented in this layer.

**Persistence Layer:**

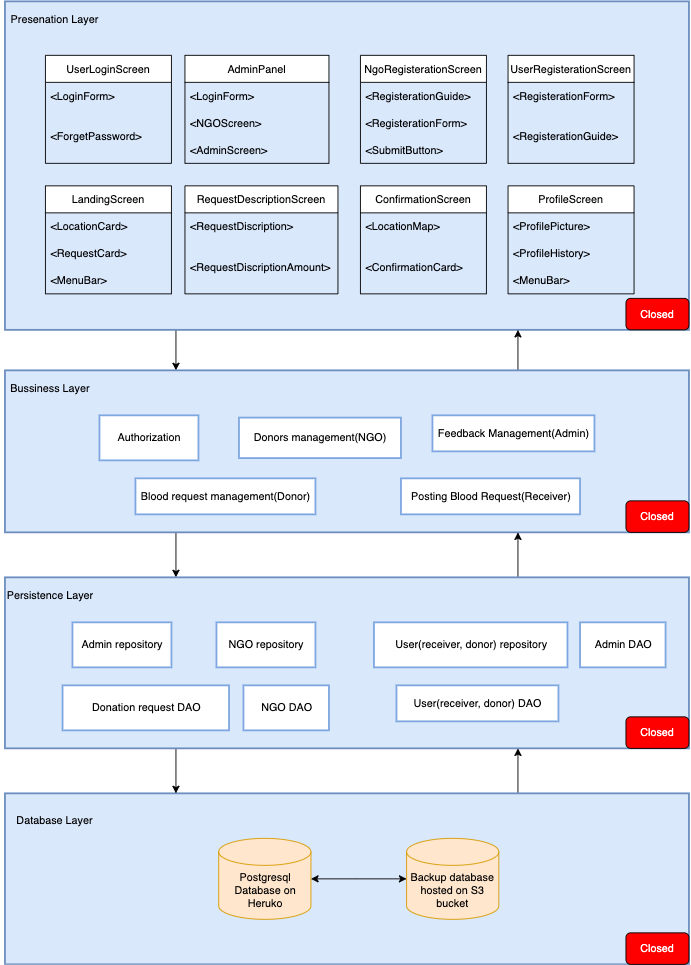
Persistence layer is the third layer of our layered architectural pattern. In our case, it contains user repositories and data access objects etc. This layer will access data from our database using repositories and data access objects and will pass it to business layer above it. For example, if donor wants to view his profile, it will interact with UI, this interaction will be forwarded to business layer and then to persistence layer, persistence layer will then read data of donor from database. Persistence layer will send this read data to business layer and from there it will be sent to presentation layer, where data will be shown to user.

**Database Layer:**

Database layer is the part of layered architectural pattern, where all user data is stored. Persistence layer access data from database using data access objects and repositories to handle user interactions.

We will use PostgreSQL hosted on the heroku.

## Architecture Diagram—As it should be



**Description:**

Main changes are in presentation and Database Layer.

**Presentation Layer:**

Presentation layer is the top most layer of our layered architectural pattern. This layer contain all the UI which is divided into different screens (modules).

We have implement this layer in our project using react/react native.

**Note:** one module consist of different individual components. Exact name of the components are mention inside <> brackets.

**Database Layer:**

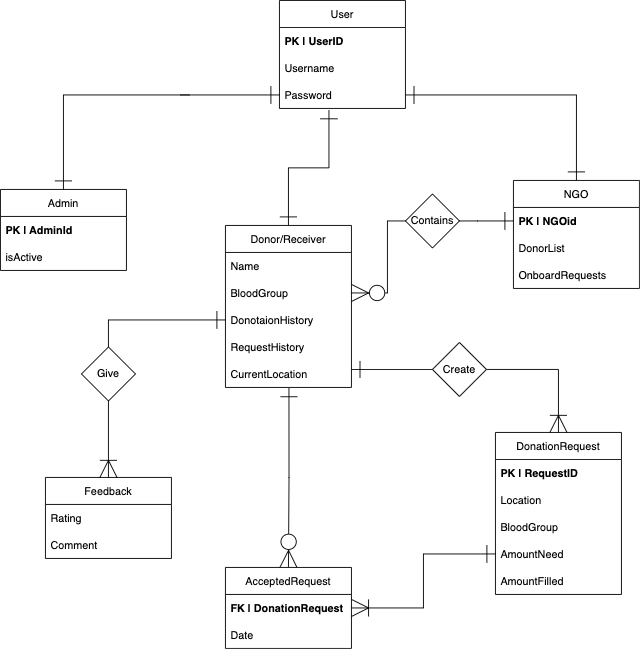
Database layer is the part of layered architectural pattern, where all user data is stored. Persistence layer access data from database using data access objects and repositories to handle user interactions. The use of multiple databases on different hosting platform is to ensure the safety of data in case of any data lost hazard.

We will use PostgreSQL hosted on the heroku and S3 bucket.

The layered architecture of the system ensures the separation of concern. The updated layered architecture also have closed layers which ensures the flow the data through predictable path and makes the implementation of security measures easy and effective. Moreover, our frontend technology react native uses individual and independent components which ensures reusability. In the layered architecture along with MVC approach is it easy to identify the specific layer and component that is causing faults which ensures the maintainability of the system. The updated layered architecture of the system also includes the backup database hosted on different hosting platform to ensure security of data.

# Data Models

* Are these the only entities in your system? Take a look at the class diagram of your system as well.
* Missing proper descriptions. Also format description, you may use a table.



# Description:

We have a general user class which contains username and password of the user. In the context of data model User can be of three types.

1. Blood receiver and donor

2. NGO which have one to many relationship with blood donor and receiver.

3. Admin which is the most privileged user of the system

**Data Types:**

username | varChar

password | varChar (hashed)

Name | varChar

isActive | bool

BloodGroup | char

DonationHistory | SET(val1, val2, val3,…)

RequestHistory | SET(val1, val2, val3,…)

CurrentLocation | varChar

DonorList | SET(val1, val2, val3,…)

OnboardRequest| SET(val1, val2, val3,…)

Rating | INT

Comment | varChar

Location | varChar

AmountNeeded | INT

AmountFilled | INT

# Tools and Technologies

**Tech stack:** React Js, React Native, NodeJs, PostgreSQL

**React Native:**

For **frontend** we will be using React Native. React native is an open source UI software framework created by Facebook. It allows developers to develop applications for Android, iOs and web. Difference between react Js and react native is that, react Js is library and react native is a framework which use native components to develop responsive mobile applications.

Advantages of using react native for frontend development are:

* + It is really famous and backed by a very large community which increases the availability of solutions we might face while development.
  + Its uses Modular and intuitive architecture which allows developers to create independent reusable component. This approach contributes in clean and reusable code, enhanced flexibility and saving time and money.
  + It allows developers to create an app with cross platform support.

**Version:** we will be using latest react native version 0.66

**Express.js:**

For **backend** we will be using ExpressJS. Express.js, or simply Express, is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.Ruby on Rails or Rails is a server-side web application framework written in Ruby. Ruby is highly used for the social app development.

Advantages of using ExpressJs as back end development are:

* + It makes the app scalable. It is easy to learn and also decreases the developer cost to maintain the app.
  + Express.JS is supported with the Google V8 engine with the help of which you would be able to get higher performance without any lag or error in the processing.
  + Express.js supports the caching feature, and the advantage of the catch is that you would not have to re-execute the codes again and again. Moreover, it will help web pages to load faster than ever.

**Version:** we will be using current stable version  4.17.1.

**PostgreSQL:**

We will be using PostgreSQL as **Database.** PostgreSQL is a powerful, open source object-relational database system. It is really mature database which provides following advantages:

* + High speed
  + Secure
  + reliability

**Version:** we will be using current version 13.3.

**Styling and Responsiveness:**

To enhance the usability of the app, responsiveness is really important. Bootstrap is CSS framework which provides grid layout for easier incorporation of responsiveness in the app. For better styling and responsiveness we will be using Bootstrap 5 along with HTML5 and CSS(2.1) .

**Design:**

For designing the screens and prototypes we will be using **Figma** version 102.9.

**Testing:**

For testing the backend API’s we will be using **Postman** (8.

**Coding Environment:**

We will be using **vsCode**(v.1.61) as Integrated Development Environment.

**Version Control:**

To track progress and version control, we will be using **GitHub.**

**Task Management:**

We will be using **Trello** for task management.

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Ahmad Aslam | E-R Diagram, Architecture diagram as in prototype |
| Muhammad Bilal | Tools and technologies |
| Saad Azam | Architecture diagram as it should be |
| Shehroz Atir | Diagram descriptions |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| E-R diagram | Ahmad Aslam Muhammad Bilal |
| Architecture diagram | Saad Azam Shehroz Atir |
| Tools and technologies | Ahmad Aslam |