The Donator App

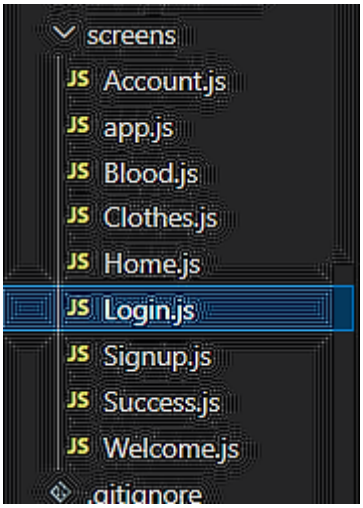
The proposed app aims to connect individuals who wish to donate with organizations that are in need of these resources. The purpose of this proposal is to suggest the development of a mobile application that will enable users to donate food, blood, money, and participate in charitable activities.

In This Report, we will discuss the stages we have been through designing this application, we chose react native to implement this application because it was challenging and we decided that we can use this chance to learn a new thing.

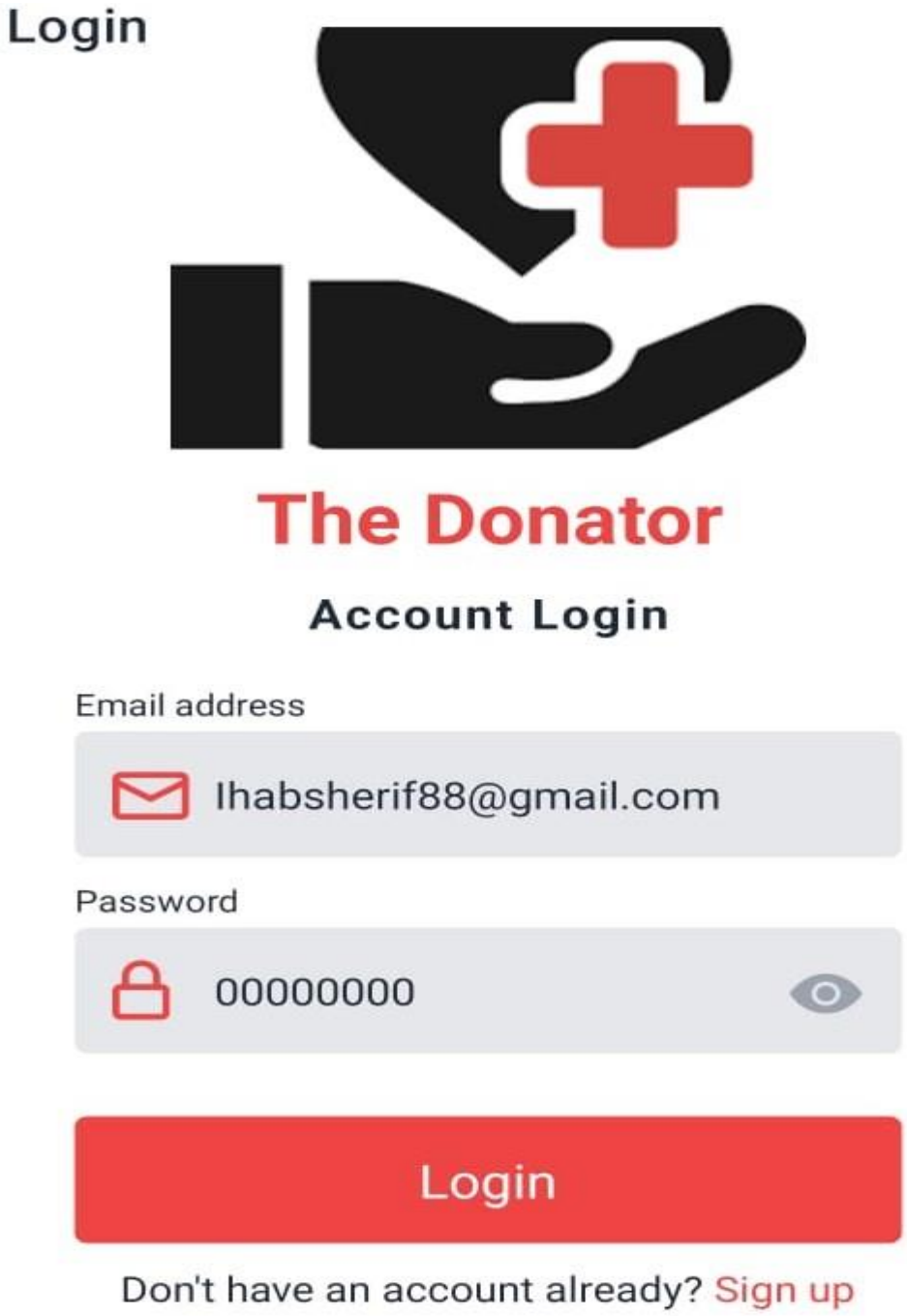
We used a mix of agile and incremental method, we decided what will we do for the design in the beginning and then we added the functionalities we need incrementally throughout the implementation by prioritizing the tasks.

Stages of Code

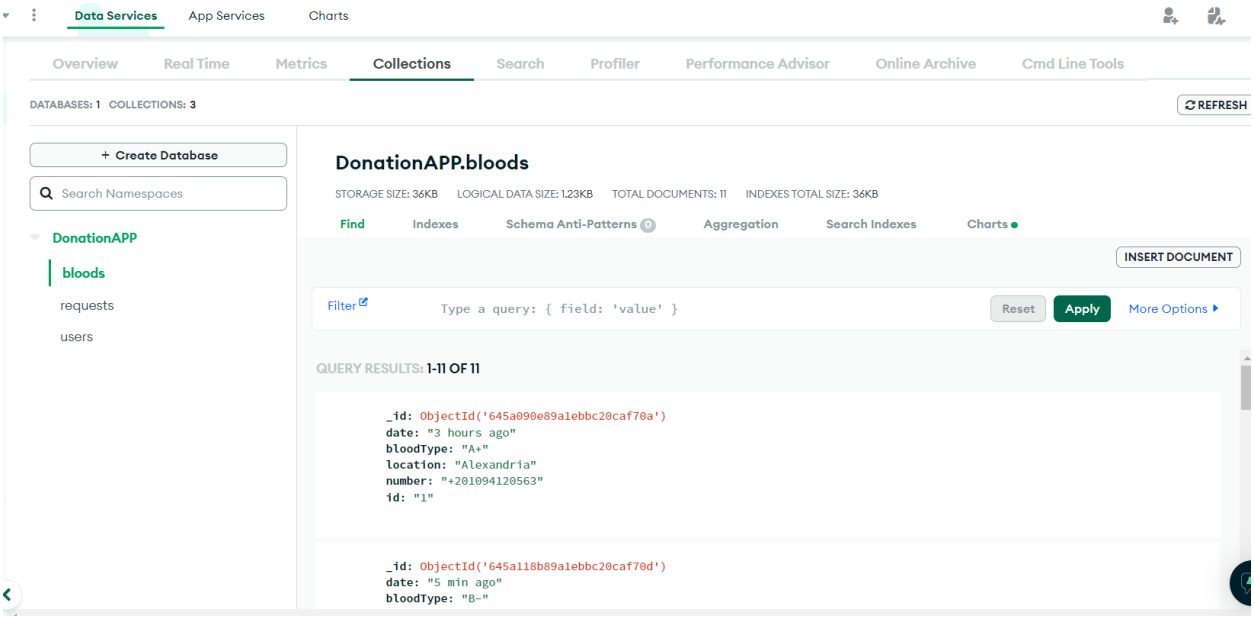
1) In order to implement multiple screens and to implement abstraction, we needed to organize our files, so we started by creating “screens” folder, which will include all the frontend of all the screen we are using.



2) We implemented the frontend of login and sign up screen.



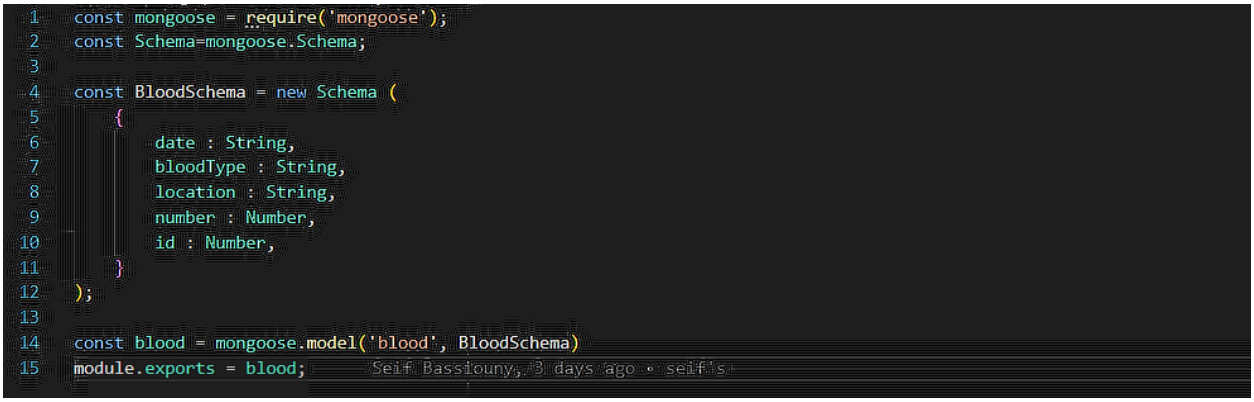
3) In order to save the user's info, we had to create a database for users, so we created a cluster on "MongoDB" to send post and get request and retrieve data.



4) Done the backend (api) of user info (login and signup) and connected it to the database and we handled inputs.

```
const User=require('./../models/user');
//sigup
router.post('/signup',(req,res)=>{
    console.log(req.body);
    let {name,email,password,birthday,bloodType}=req.body;
    name = name;
    email = email;
    password = password;
    birthday = birthday;
    bloodType=bloodType;

    if(name == "" ||email == ""||password==""||birthday==""||bloodType=="")
    {
        res.json({
            status: "FAILED",
            message:"Empty input fields!"
        });
    }else if (!/^[a-zA-Z ]*$/.test(name)){
        res.json({
            status: "FAILED",
            message:"Invalid name entered!"

        })
    }
    else if (!/^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$/i.test(email)){
        res.json({
            status: "FAILED",
            message:"Invalid Email entered!"

        })
    }
    else if (!new Date(birthday).getTime()){
        res.json({
```

5) After we finished the login and register, we started working on the main functionality which was the blood donation schema.



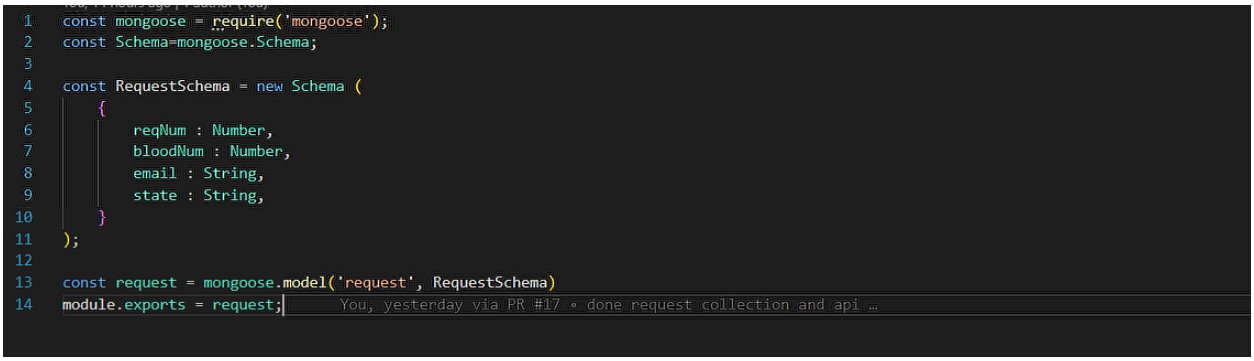
```
const mongoose = require('mongoose');
const Schema=mongoose.Schema;

const BloodSchema = new Schema (
    {
        date : String,
        bloodType : String,
        location : String,
        number : Number,
        id : Number,
    }
);

const blood = mongoose.model('blood', BloodSchema)
module.exports = blood;      Seif Bassiouny, 3 days ago • seif's
```

6) The goal was to implement a live emergency feed which the user can see and interact with.

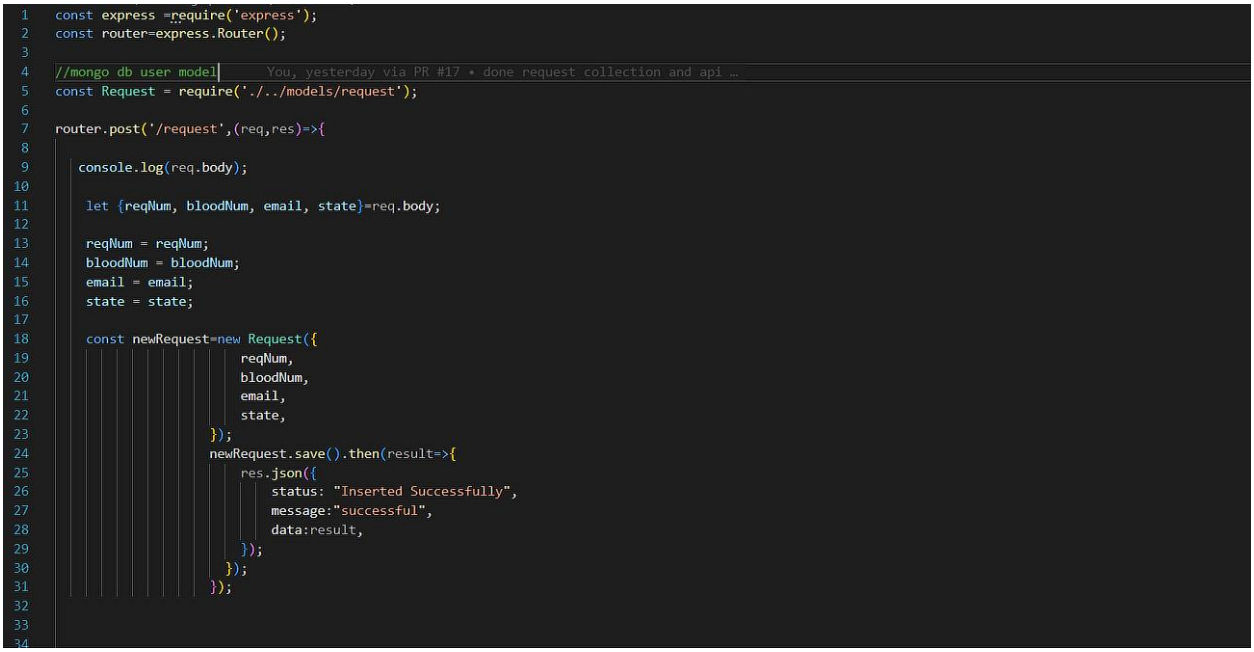
So we started by retrieving the data of the blood needers with location of the hospital and the blood type needed. The screen showed only the blood types the user can donate to.

7) The goal of this emergency feed, is to use the donate button which uses post function to register a request in the database. Each user donating is given a unique request number which he will use when he goes to the hospital to donate.

8) The donate button register an entry in the request schema which takes user donation requests. By default the user request is “pending” and the user we will for his request to be “confirmed by the admin.



```
const mongoose = require('mongoose');
const Schema=mongoose.Schema;

const RequestSchema = new Schema (
    {
        reqNum : Number,
        bloodNum : Number,
        email : String,
        state : String,
    }
);

const request = mongoose.model('request', RequestSchema)
module.exports = request;       You, yesterday via PR #17 • done request collection and api …
```

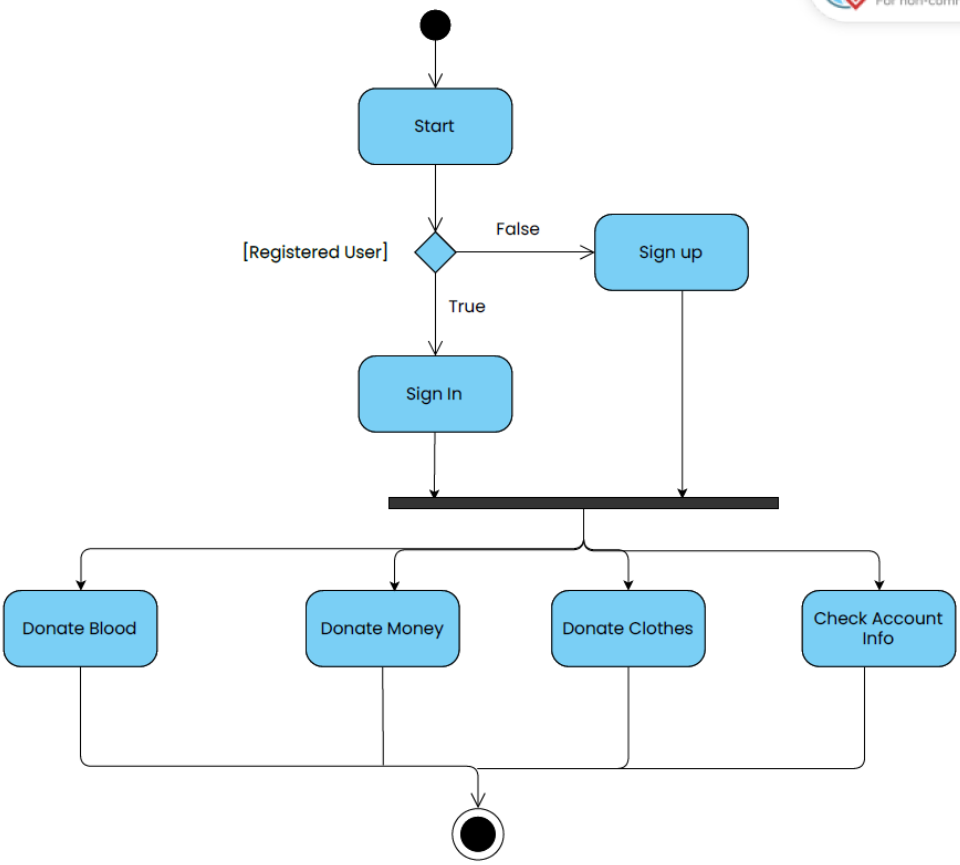


```
const express =require('express');
const router=express.Router();

//mongo db user model       You, yesterday via PR #17 • done request collection and api …
const Request = require('./../models/request');

router.post('/request',(req,res)=>{

    console.log(req.body);

    let {reqNum, bloodNum, email, state}=req.body;

    reqNum = reqNum;
    bloodNum = bloodNum;
    email = email;
    state = state;

    const newRequest=new Request({
                        reqNum,
                        bloodNum,
                        email,
                        state,
                    });
                    newRequest.save().then(result=>{
                        res.json({
                            status: "Inserted Successfully",
                            message:"successful",
                            data:result,
                        });
                    });
                });
```

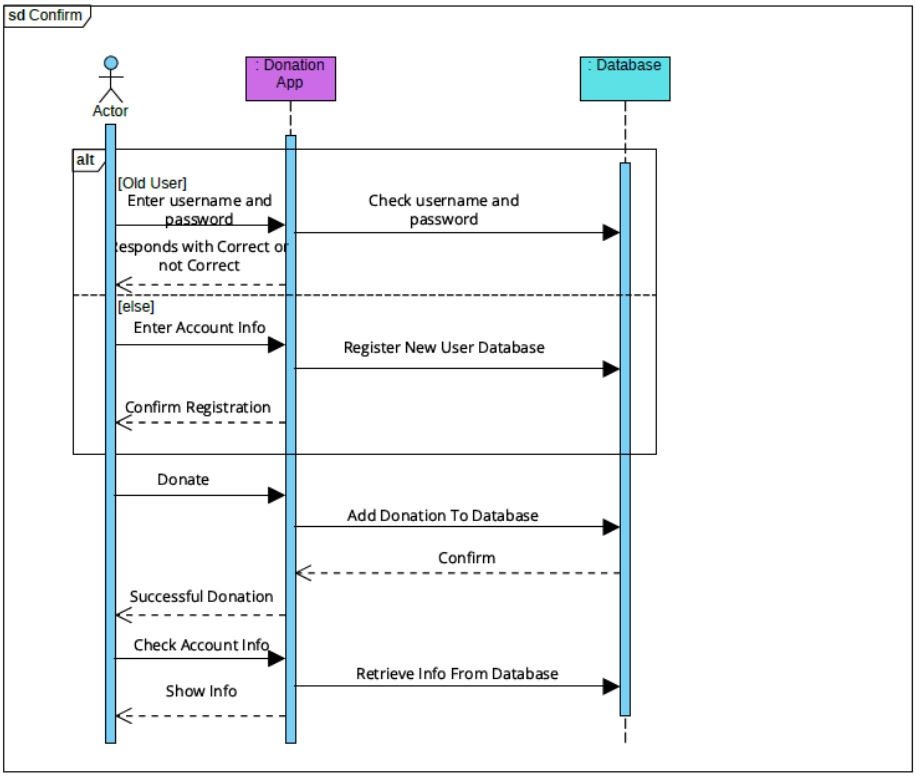
9) We started to implement a screen for clothes and for money also, but didn’t finish it.

10) We added styling for the screens + plus user info screen.

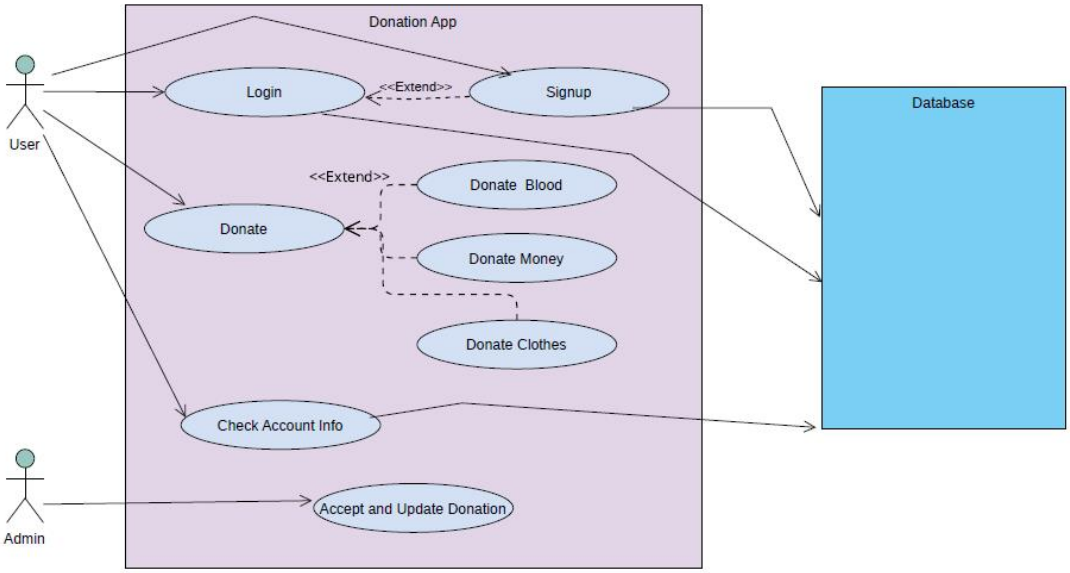
1) Activity Diagram



2) Sequence Diagram



3) Use Case Diagram



4) State Machine Diagram