##### A Project report on

**DONOR MANAGER**

###### A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

**Bachelor of Technology**

**in**

**Computer Science and Engineering**

Submitted by

##### (19H51A05E3) PARINITHA PEMBARTHI

##### (19H51A05E4) SAVAN REDDY

##### (19H51A05E5) DEEPTHI

Under the esteemed guidance of

**Mr. V. NARASIMHA**

Assistant Professor



**Department of Computer Science and Engineering**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

(An Autonomous Institution under UGC & JNTUH , Approved by AICTE, Permanently Affiliated to JNTUH, Accredited by NBA.)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

#### 2019- 2023

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



#### CERTIFICATE

We are certify that the Mini Project-1 report entitled **DONAR MANAGER** being **Parinitha (19H51A05E3), Savan Reddy (19H51A05E4), Deepthi (19H51A05E5),** in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

###### The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

**Mr. V. Narasimha Dr. K. Vijaya Kumar**

Assistant Professor  **Professor and HOD**

**Dept. of CSE Dept. of CSE**

Table of contents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CHAPTER  NO. |  |  | TITLE | PAGE NO. |
|  | **LIST OF FIGURES** | | | i |
|  | **ABSTRACT** | |  | 1 |
| 1 | **INTRODUCTION** | | | 2 |
|  | 1.1 | Project Aim | | 3 |
| 2 | **BACKGROUND WORK** | | | 4 |
|  | 2. | Literature Review | | 4 |
| 3 | **PROPOSED SYSTEM** | | | 6 |
|  | 3.1 | Software specifications | | 6 |
|  | 3.2 | Conceptual Diagram | | 8 |
|  | 3.3 | Program | | 9 |
| 4 | **DESIGNING** | |  | 28 |
| 5 | **RESULTS AND DISCUSSION** | | | 30 |
| 6 | **CONCLUSION AND FUTUREWORK** | | | 31 |
| 7 | REFERENCES | |  | 32 |

#### ABSTRACT

Despite fast-evolving technology, we have hospitals struggling to provide blood and organs when the patient is in need of them. Because of this delay or scarce supply, it is leading to the death of the patient. Keeping these struggles and needs in mind we decided to create a Blood and Organ Donation website.

This website will be a common domain for both the donor and the patient in need. The donor can just visit the website and share his details and the patient can in the same manner visit the website and find a donor of his choice. The website will also be recording the medical status of the donor so that the procedure can go smoothly. The database of our website will be using MongoDB. The back end of our website will be using NodeJs and ExpressJs. For communication with the user, we will be using front end ReactJs. This website can also be used by organizations for their needs with a more customized format.

This website can be a great help for the patient in the need of the hour.

#### 

#### 

#### INTRODUCTION

The person who gives the organs is called a donor, while a person who receives the organ is called a recipient. Many patients and hospitals find donors, but most of the time the donor isn’t available. This may lead to loss of life or complications in an operation due to the delay. Sometimes the donor might be available but his health status won’t match the given requirements for the donation.

Blood donation is required during an organ transplant, accidents, cancer treatment etc. For blood donation, one needs to check for a donation camp or visit a blood bank. The Manual Blood donation system has many disadvantages which includes, it is too time consuming, often leads to error prone results, consumes a lot of manpower, lacks donor information, retrieval of data takes a lot of time, percentage of accuracy is less. In the time of emergency, it becomes difficult to approach the right donor. Rare blood groups are not available all the time at all blood banks and recipients find difficulties to track the right blood donor.

To bridge this gap of communication we decided to create a website that collects the data of a donor and encourages people to donate.  And this website connects the patient to the donor as per his requirements.

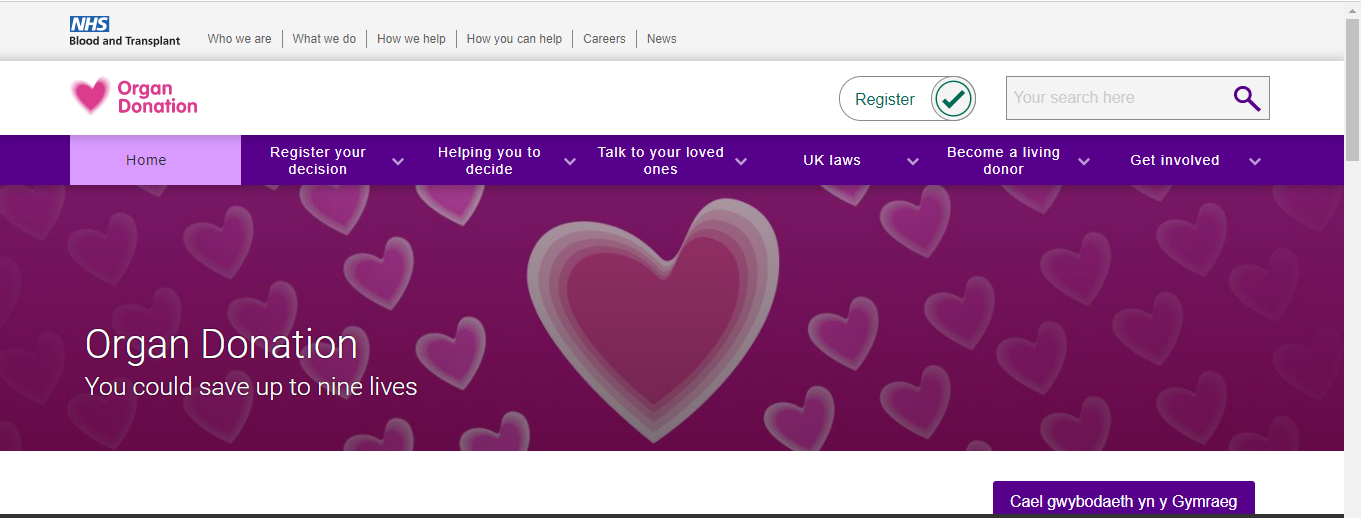
**AIM OF THE PROJECT**

1. To create a Website that is user friendly and appealing to the eye.
2. To create a Website that encourages people to donate and guide them through the donation process.
3. To create a Website  that stores the data  of the donors and  provides them to  the patients as per  their requirements
4. To create a Website that is serviced based and that uses minimal time to give or receive data.

**2. LITERATURE REVIEW**

**2.1   NHS Organ & Transplant**

This website is a UK based website, where one can register themselves and become an organ donor. Three people die every day across the UK due to a shortage of organs 10,000 people in the UK are in need of a transplant 20 million people have joined the NHS Organ Donor Register Fewer than 5,000 people die in circumstances where they can become donors. Families are more likely to agree if donation wishes are known one donor can help save or transform nine lives.

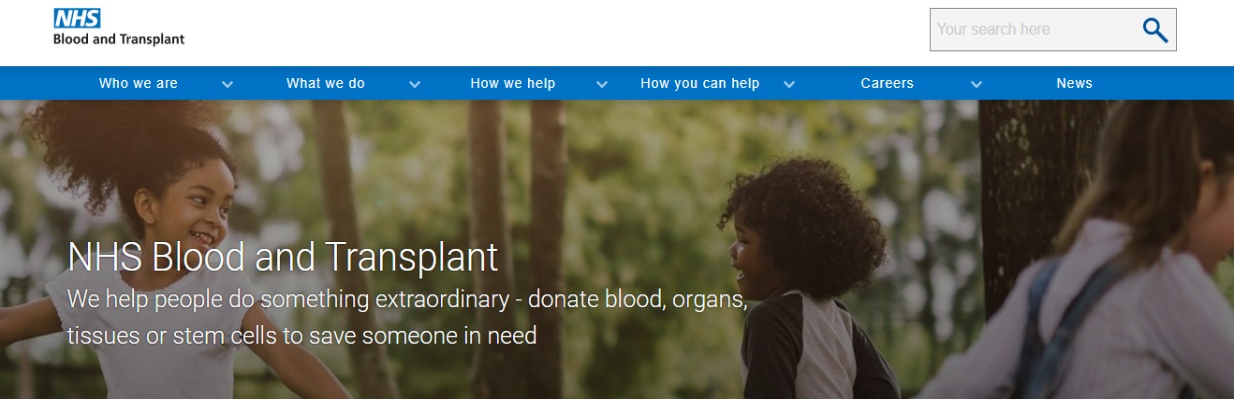


**2.2   NHS Blood & Transplant**

  This website is a UK based website, where one can register themselves and

become aBlood donor.They  provide services likeblood, transplantations,

Diagnostics and research.

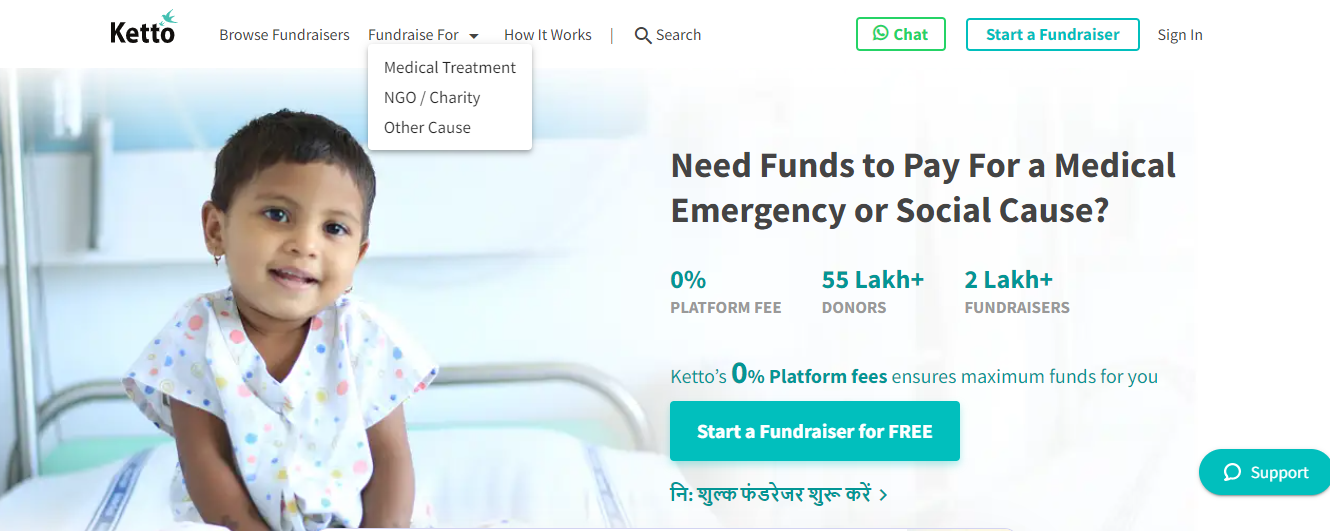


**2.3   Ketto**

This is a Indian ba ketto has    ever since assisted people, individuals and communities across India to

funds for imaginative  and personalised website that enables donors to make charitable donations and

donate money as well.



DONOR MANAGER

**3. PROPOSED SYSTEM**

Understanding our problem statement we have come up with a user friendly website that can be accessible by all. We created a website in which a Donor can upload his information and register himself as a Donor. This information will be stored in our database. When there is a requirement, they can contact our administration for the same. When our administration gets a request we sort the data as per the requirements and provide the stakeholder with the details of the donor.

***3.1 SOFTWARE SPECIFICATIONS***

* DATABASE

A **database** is an organized collection of data, so that it can be easily accessed and   managed. You can  organize data into tables, rows, columns, and index it to make it easier to find relevant information. The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data. There are many **dynamic websites** on the World Wide Web nowadays which are and led through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database. There are many **databases available** like Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

* HOSTING OF THE WEBSITE

The business of providing the storage, connectivity, and services necessary to   serve files for a website. A company that provides web hosting services is called a web host, and their detailed offers of storage, connectivity, and services are called web hosting plans. There are literally thousands of web hosting services, ranging from individuals to worldwide corporations, and many web hosts offer multiple web hosting plans WEBSITE WITH UI/UX  
UX, or user experience, is every interaction your business has with people on your website, mobile site  apps,  and online properties or services. That might sound like an exhausting number of situations to consider, but creating good UX design means focusing on the user, no matter where they are.

DONOR MANAGER

 Languages used in the Website:

* MongoDb

MongoDB is an open source [NoSQL](https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL) database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

MongoDB makes use of records which are made up of documents that contain a data structure composed of field and value pairs. Documents are the basic unit of data in MongoDB.

* Express.js

Express.js is a web application framework for Node.js. It provides various features that make web application development fast and easy which otherwise takes more time using only Node.js.Express.js is based on the Node.js middleware module called ***connect*** which in turn uses **http** module. So, any middleware which is based on connect will also work with Express.js

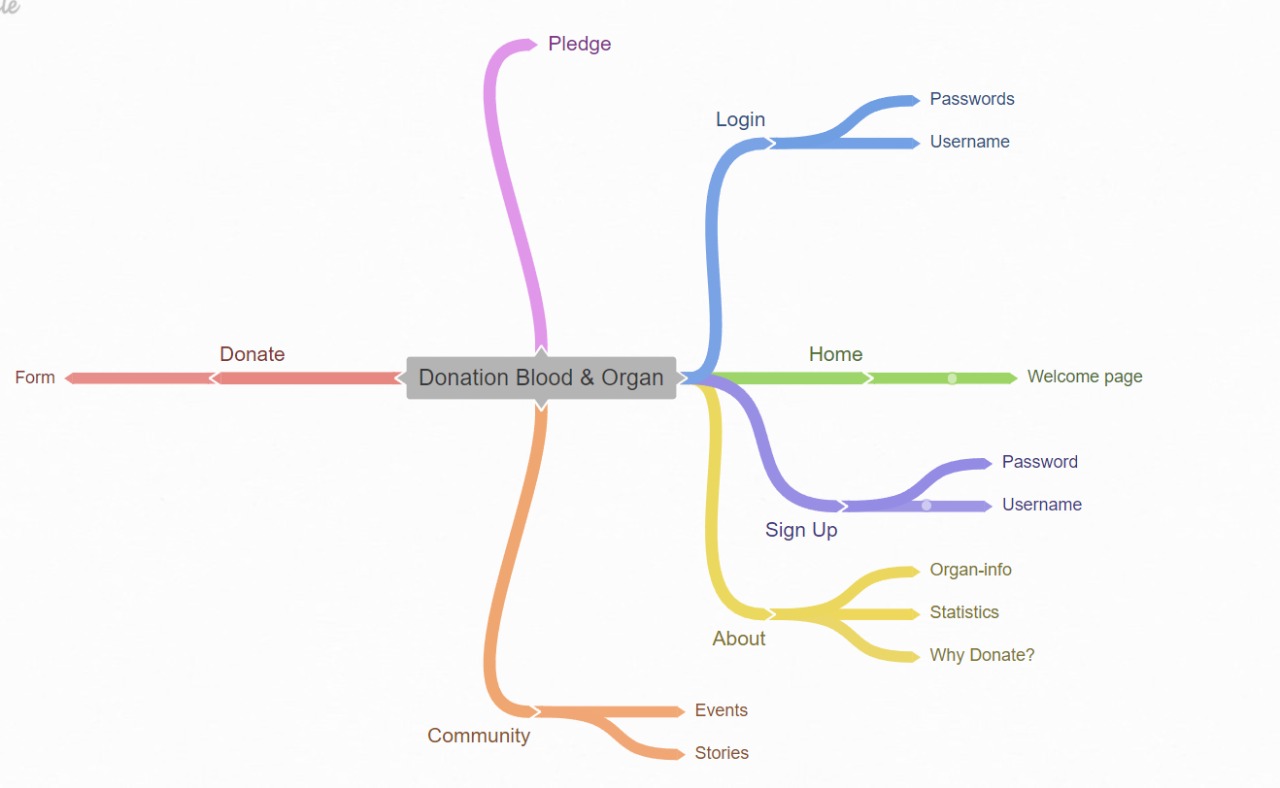
* Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of projectNode.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performance.

* React.js

ReactJS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

**3.2 Conceptual diagram**



**3.3 PROGRAMMING**

***Admin page:***

import React, { useState ,Component,useEffect} from "react";

import "./Admin.css";

import Box from "@mui/material/Box";

import Tab from "@mui/material/Tab";

import TabContext from "@mui/lab/TabContext";

import TabList from "@mui/lab/TabList";

import Tabs from "@mui/material/Tab";

import TabPanel from "@mui/lab/TabPanel";

import topBG from "../../Images/ImageFromPc/Blue-Background-2.png";

import Grid from '@mui/material/Grid';

import axios from 'axios';

import { useParams } from "react-router";

import FlipableEvent from "../../Components/FlipableEvent/FlipableEvent";

const Admin = () => {

const [value, setValue] = React.useState(1);

const [ LoadData ,setLoadData] = useState(0);

const[ xArray ,setxArray]=useState([0,1,2]);

const handleChange = (event, newValue) => {

setValue(newValue); setLoadData(newValue);

xArray[0]=newValue;

};

const [StoryStatus,setStoryStatus]=useState

const [DonationData , setDonationData] =useState([]);

const [StoriesData , setStoriesData] =useState([]);

const [EventsData , setEventsData] =useState([]);

const [UserStatus, setUserStatus] = useState('Not Auth');

const urlparams = useParams();

const UserName=urlparams.UserName;

useEffect(async ()=>{

if(UserName){

const AccessToken = localStorage.getItem(`Admin ${UserName}`);

console.log("username->");

console.log(UserName)

console.log("fetched from local->")

console.log(AccessToken);

await axios.get('http://localhost:5000/auth/TokenValidate', {headers:{"authorization" : `Bearer ${AccessToken}` }}).then(Response=>{

if(Response.data.resval === "TokenVerified")

{ setUserStatus("Auth");

console.log("Token Verified");

}

else{

window.location.replace(`/auth/Admin/Login`);

}

console.log(Response.data);

}).catch(error=>{

console.log(error);

})

}

},[JSON.stringify(xArray)]);

useEffect(() => {console.log("useEffect Fired");

if(LoadData === '1'){console.log("DonationData Fetched");

axios.get('http://localhost:5000/Donate/')

.then((response=>{

console.log(" Donation response found");

setDonationData(response.data);

console.log(response.data);

}));

setEventsData([]); setStoriesData([]);

}

else if(LoadData === '2') {console.log("StoriesData Fetched");

axios.get('http://localhost:5000/Stories/')

.then((response=>{

console.log("Stories response found");

setStoriesData(response.data);

console.log(response.data);

}));

setEventsData([]); setDonationData([]); }

else if(LoadData === '3'){console.log("EventsData Fetched");

axios.get('http://localhost:5000/Events/')

.then((response=>{

console.log(" Events response found");

setEventsData(response.data);

console.log(response.data);

}));

setStoriesData([]); setDonationData([]);

}

}, [JSON.stringify(xArray)])

const DonationFormDisplay = ({Item}) => { console.log("Item->"); console.log(Item.FirstName)

return (

<Grid container spacing={6} sx={{width:'92%', justifyContent:'space-around', alignSelf:'center' , alignItems:'center' ,marginTop:'5vh' , marginBottom:'20vh', borderRadius: '10px' ,boxShadow: 8, paddingBottom:'20px',bgcolor:'hsl(0,0%,94%)'}} >

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >FirstName : <span className={Item.FirstName==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.FirstName} </span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" > Middle Name : <span className={ Item.MiddleName==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.MiddleName}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >Last Name : <span className={ Item.LastName==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.LastName}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text"

>Gender : <span className={ Item.Gender==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.Gender}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >Blood Group : <span className={ Item.BloodGroup==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.BloodGroup}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >Mobile Number : <span className={ Item.MobileNumber==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.MobileNumber}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >AadharCard Number : <span className={ Item.AadharCardNumber==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.AadharCardNumber}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" > Parent Name : <span className={ Item.ParentName==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.ParentName}</span>

</span>

</Grid>

<Grid item xs={12} sm={12} lg={12} sx={{ display:'flex', flexDirection:'row' }}>

<span className="Address-Break">Address : </span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >Apartment Name/Society Name : <span className={ Item.Address.ApartmentName\_SocietyName==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.Address.ApartmentName\_SocietyName}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" > City : <span className={ Item.Address.City==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.Address.City}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" > District : <span className={ Item.Address.District==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.Address.District}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" > State : <span className={ Item.Address.State==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.Address.State}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >Pincode : <span className={ Item.Pincode==="Not Mentioned"? " Not-Mentioned" : "Mentioned"}>

{Item.Pincode}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >occupation : <span className={ (Item.occupation==="Not Mentioned")? " Not-Mentioned" : "Mentioned"}>

{Item.occupation}</span>

</span>

</Grid>

<Grid item xs={12} sm={6} lg={4}>

<span className="Donation-Info-Component-Text" >Date & Time: <span className="Mentioned">

{Item.Date}</span>

</span>

</Grid>

</Grid>

);

};

const params= useParams();

const username=params.UserName;

const userName = username;

const UpdateStatus = async (UserName,Statusx) =>{var temp="";

console.log("username before sending put-> ");

console.log(`http://localhost:5000/Stories/update/${UserName}`);var ty="";

Statusx==="Visible"? ty="Hidden" : ty="Visible"; setStoryStatus(ty); const tr=ty; console.log("Before Updating->"); console.log({tr});

await axios.put(`http://localhost:5000/Stories/update/${UserName}`,{"Status":ty})

.then(res=>{

console.log("UI sent Story Update request to Server Successfully");

}).catch(err=>{

console.log("Error while UI sending Story Update request to Server");

});

console.log("StoryStatus After Update->")

setStoryStatus(tr);

console.log(StoryStatus);

};

console.log(StoryStatus);

const StoryDisplay = ({Story}) =>{console.log("Story->"); setStoryStatus(Story.Status);

return(

<div className="Admin-Story-Wrapper">

<div className="Admin-Story-Container">

<div className="Row-Holder1">

<img className ="ASI" src={Story.Image} alt="Broken" />

<span className="Admin-Story-Title">

{Story.Title}

</span>

</div>

<p className="Admin-Story-Description">

{Story.Description} from {Story.UserName}

</p>

<div className={"Status-Button" + (Story.Status==="Visible"? " Hidden" : " Visible" )} onClick={()=>{

UpdateStatus(Story.UserName,Story.Status);

}} >{

<span className="Status-Text">{Story.Status}</span>

}</div>

</div>

</div>

)

}

if(UserStatus==="Auth"){

return (

<div className="Admin-Wrapper">

<div className="Top-Pane">

<img src="" alt="" />

</div>

<div className="Raised-Pane">

<div className="Admin-Header">

<div className="Avatar-Holder">

<img className="Avatar Hidden" src="" alt="" />

</div>

<div className="User-Name-Holder">

<span className="User-Name">AdminAccess</span>

<span className="Basic-Info-Text">Admintest1</span>

</div>

</div>

<Box

sx={{

width: "100%",

typography: "body1",

marginTop: "10vh",

display: "flex",

flexFlow: "column nowrap",

}}

>

<TabContext value={value}>

<Box

sx={{

borderBottom: 1,

borderColor: "divider",

bgcolor: "background.paper",

}}

>

<TabList

onChange={handleChange}

aria-label="Admin-Tab-Switch"

centered

>

<Tab label="Donation Forms" value="1" sx={{fontSize:'16px',fontWeight:'320'}} />

<Tab label="Stories" value="2" sx={{fontSize:'16px',fontWeight:'320'}}/>

<Tab label="Events" value="3" sx={{fontSize:'16px',fontWeight:'320'}}/>

</TabList>

</Box>

<TabPanel value="1" sx={{display:'flex', flexFlow:'column nowrap', alignItems:'center' }}>

<span className="Count-Of-Entity">

{DonationData.length}

</span>

{

DonationData.map(Form =>{

return(

<DonationFormDisplay Item={Form}/> );

})}

</TabPanel>

<TabPanel value="2" sx={{display:'flex', flexFlow:'column nowrap', alignItems:'center' }}>

<span className="Count-Of-Entity">

{StoriesData.length}

</span>

{

StoriesData.map(Story =>{

return(

<StoryDisplay Story={Story}/> );

})}

</TabPanel>

<TabPanel value="3" sx={{position:'relative',flex:'1',height:'100%', display:'flex', flexFlow : 'row wrap' ,justifyContent: 'space-around', alignItems:'center' }}>

{EventsData.length}

{

EventsData.map(Event=>{

return(

<FlipableEvent Event={Event}/>

)

})

}

</TabPanel>

</TabContext>

</Box>

</div>

</div>

);

}

else{

return(

<div className="Return-SOmething"></div>

)

}

};

export default Admin;

*Stories page:*

import React, { useState } from "react";

import "./Donate.css";

import Box from "@material-ui/core/Box";

import TextField from "@mui/material/TextField";

import InputLabel from "@mui/material/InputLabel";

import MenuItem from "@mui/material/MenuItem";

import FormControl from "@mui/material/FormControl";

import Select from "@mui/material/Select";

import axios from "axios";

import OutlinedInput from "@mui/material/OutlinedInput";

import ListItemText from "@mui/material/ListItemText";

import Checkbox from "@mui/material/Checkbox";

const Donate = () => {

const standard = "standard",

filled = "filled",

outlined = "outlined";

const [DonArr, setDonArr] = useState([]);

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

// const [] =useState('');

const ITEM\_HEIGHT = 48;

const ITEM\_PADDING\_TOP = 8;

const MenuProps = {

PaperProps: {

style: {

maxHeight: ITEM\_HEIGHT \* 4.5 + ITEM\_PADDING\_TOP,

width: 250,

},

},

};

const names = ["Heart", "Liver", "Lungs", "Pancreas", "Kidneys", "Intestine"];

const [OrganName, setOrganName] = React.useState([]);

const handleChange = (event) => {

const {

target: { value },

} = event;

setOrganName(

// On autofill we get a the stringified value.

typeof value === "string" ? value.split(",") : value

);

};

const OnSubmit = async () => {

await axios

.post("http://localhost:5000/Donate/add", {

FirstName: DonArr[1],

MiddleName: DonArr[2],

LastName: DonArr[3],

ParentName: DonArr[4],

Address: {

ApartmentName\_SocietyName: DonArr[5],

City: DonArr[6],

District: DonArr[7],

State: DonArr[8],

},

Pincode: DonArr[9],

MobileNumber: DonArr[11],

AadharCardNumber: DonArr[10],

Occupation: DonArr[12],

Gender: DonArr[13],

BloodGroup: BloodGroup,

EmergencyContactNumber: DonArr[14],

Organs: OrganName,

Date: new Date().toLocaleDateString(),

})

.then((Response) => {

const DFR = Response.data;

console.log(DFR);

})

.catch((error) => {

console.log(error);

});

};

DonArr[0] = "";

const [BloodGroup, setBloodGroup] = React.useState("");

return (

<div className="Donate-Wrapper">

<Box

sx={{

display: "flex",

flexFlow: "row wrap",

alignItems: "center",

justifyContent: "space-around",

width: "80%",

position: "relative",

padding: "30px 30px",

borderRadius: "10px",

}}

>

<TextField

autoComplete=""

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

id=""

label="First Name"

variant="outlined"

onChange={(e) => {

DonArr[1] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

//autoComplete=""

id="standard-basic"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

label="Middle Name"

variant="outlined"

onChange={(e) => {

DonArr[2] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

//autoComplete="new-password"

id="standard-basic"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

label="Last Name"

variant="outlined"

onChange={(e) => {

DonArr[3] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

//autoComplete="new-password"

id="standard-basic"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

label="Parent/Guardian Name"

variant="outlined"

onChange={(e) => {

DonArr[4] = e.target.value;

}}

sx={{ width: "100%", maxWidth: "300px", boxShadow: 4 }}

/>

<FormControl

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

>

<InputLabel

id="demo-simple-select-label"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

sx={{ background: "white" }}

>

Blood Group

</InputLabel>

<Select

labelId="demo-simple-select-label"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

id="demo-simple-select"

value={BloodGroup ?? "Undefined BloodGroup"}

label="Age"

onChange={(e) => {

setBloodGroup(e.target.value);

}}

>

<MenuItem value={"O+"}>O+</MenuItem>

<MenuItem value={"O-"}>O-</MenuItem>

<MenuItem value={"A+"}>A+</MenuItem>

<MenuItem value={"A-"}>A-</MenuItem>

<MenuItem value={"B+"}>B+</MenuItem>

<MenuItem value={"B-"}>B-</MenuItem>

<MenuItem value={"AB+"}>AB+</MenuItem>

<MenuItem value={"AB-"}>AB-</MenuItem>

</Select>

</FormControl>

<span className="Address-Identifier">Address : </span>

<TextField

autoComplete="new-password"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

id="standard-basic"

label="Apartment Name/Society Name"

variant="outlined"

onChange={(e) => {

DonArr[5] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

autoComplete="new-password"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

id="standard-basic"

label="City"

variant="outlined"

onChange={(e) => {

DonArr[6] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

// autoComplete=""

id="standard-basic"

label="District"

variant="outlined"

onChange={(e) => {

DonArr[7] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

autoComplete="new-password"

id="standard-basic"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

label="State"

variant="outlined"

onChange={(e) => {

DonArr[8] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

autoComplete="new-password"

id="standard-basic"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

label="Pincode"

variant="outlined"

onChange={(e) => {

DonArr[9] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

autoComplete="new-password"

id="standard-basic"

label="Aadhar Card Number"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

variant="outlined"

onChange={(e) => {

DonArr[10] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

autoComplete="new-password"

id="standard-basic"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

label="Mobile Number"

variant="outlined"

onChange={(e) => {

DonArr[11] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<TextField

autoComplete="new-password"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

id="standard-basic"

label="Occupation"

variant="outlined"

onChange={(e) => {

DonArr[12] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<FormControl sx={{ width: "100%", margin: "20px", maxWidth: "300px" }}>

<InputLabel

id="demo-simple-select-label"

sx={{ background: "white" }}

>

Gender

</InputLabel>

<Select

labelId="demo-simple-select-label"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

id="demo-simple-select"

value={DonArr[13] ?? "Undefined Gender"}

label="Gender"

onChange={(e) => {

DonArr[13] = e.target.value;

}}

>

<MenuItem value={"Male"}>Male</MenuItem>

<MenuItem value={"Female"}>Female</MenuItem>

<MenuItem value={"Other"}>Other</MenuItem>

</Select>

</FormControl>

<FormControl sx={{ m: 1, width: 300 }}>

<InputLabel

id="demo-multiple-checkbox-label"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

sx={{ backgroundColor: "white" }}

>

Organs you wish to Donate

</InputLabel>

<Select

labelId="demo-multiple-checkbox-label"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

id="demo-multiple-checkbox"

multiple

value={OrganName ?? "Undefined organs"}

onChange={handleChange}

input={<OutlinedInput label="Organs" />}

renderValue={(selected) => selected.join(", ")}

MenuProps={MenuProps}

>

{names.map((name) => (

<MenuItem key={name} value={name}>

<Checkbox checked={OrganName.indexOf(name) > -1} />

<ListItemText primary={name} />

</MenuItem>

))}

</Select>

</FormControl>

<TextField

autoComplete="new-password"

id="standard-basic"

inputProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}} InputLabelProps={{style: {fontSize: 18,wordSpacing: 17,lineHeight: 1.5}}}

label="Emergency Contact Number"

variant="outlined"

onChange={(e) => {

DonArr[14] = e.target.value;

}}

sx={{

width: "100%",

margin: "20px",

maxWidth: "300px",

boxShadow: 4,

}}

/>

<div

className="SignUp-Button"

onClick={(e) => {

OnSubmit(e);

}}

>

Submit Form

</div>

</Box>

</div>

);

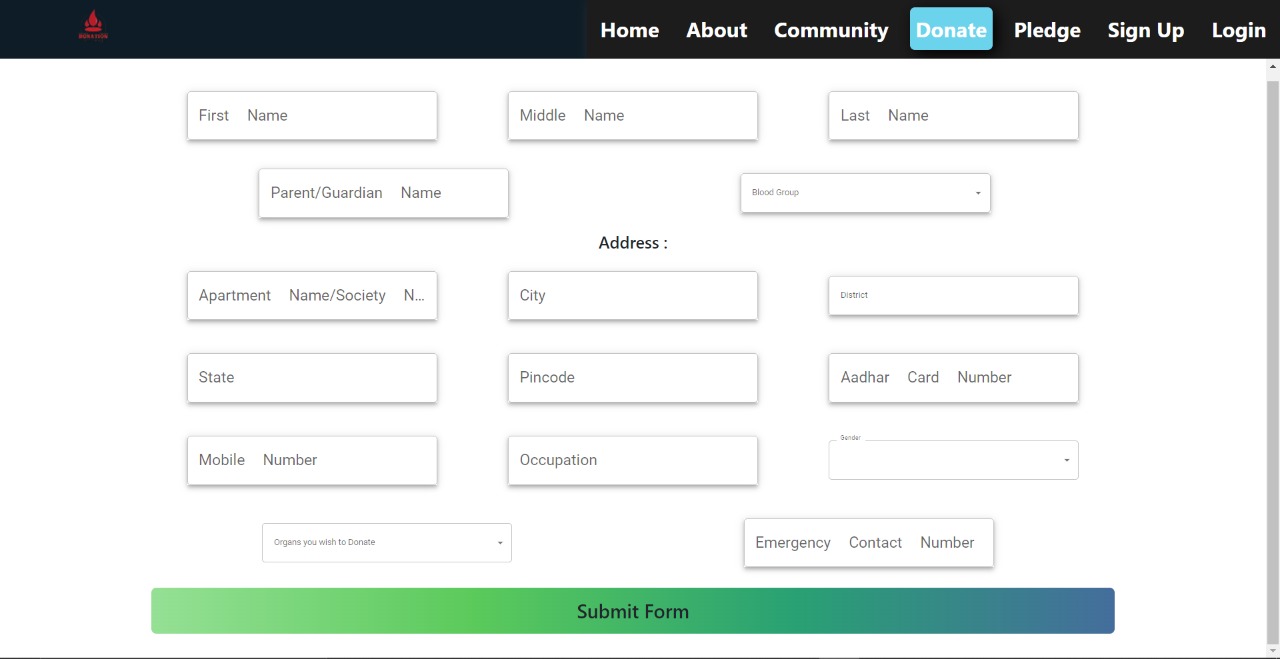
};

export default Donate;

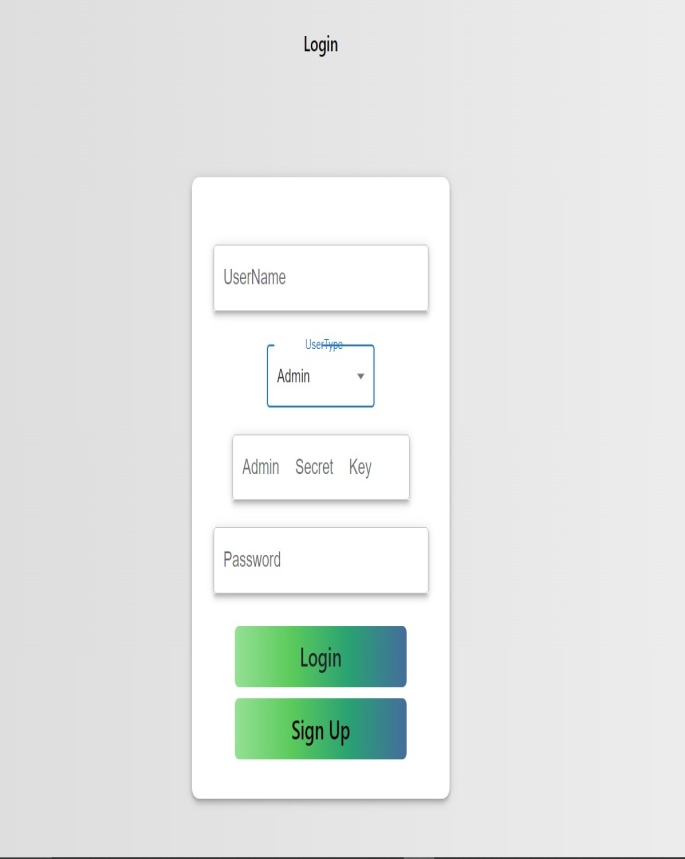
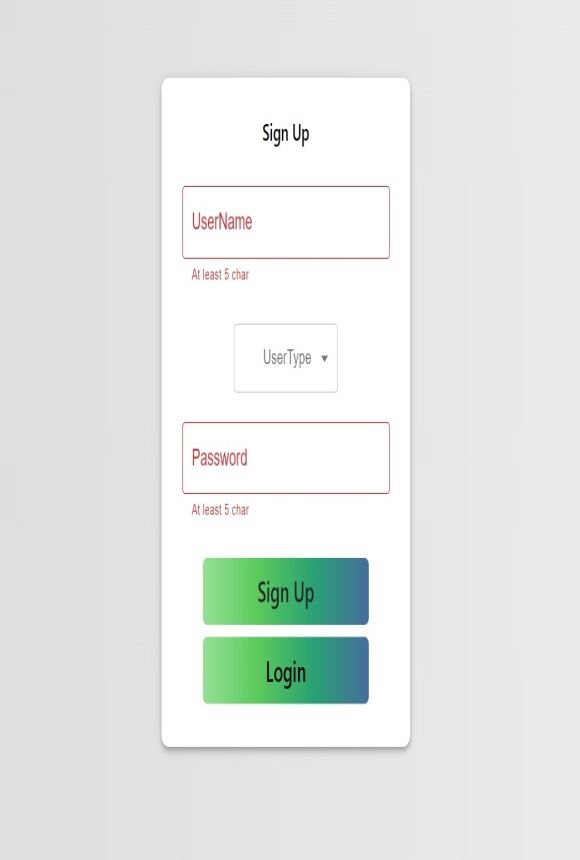
DESIGNING

**SYSTEM IMPLEMENTATON SCREENSHOTS**

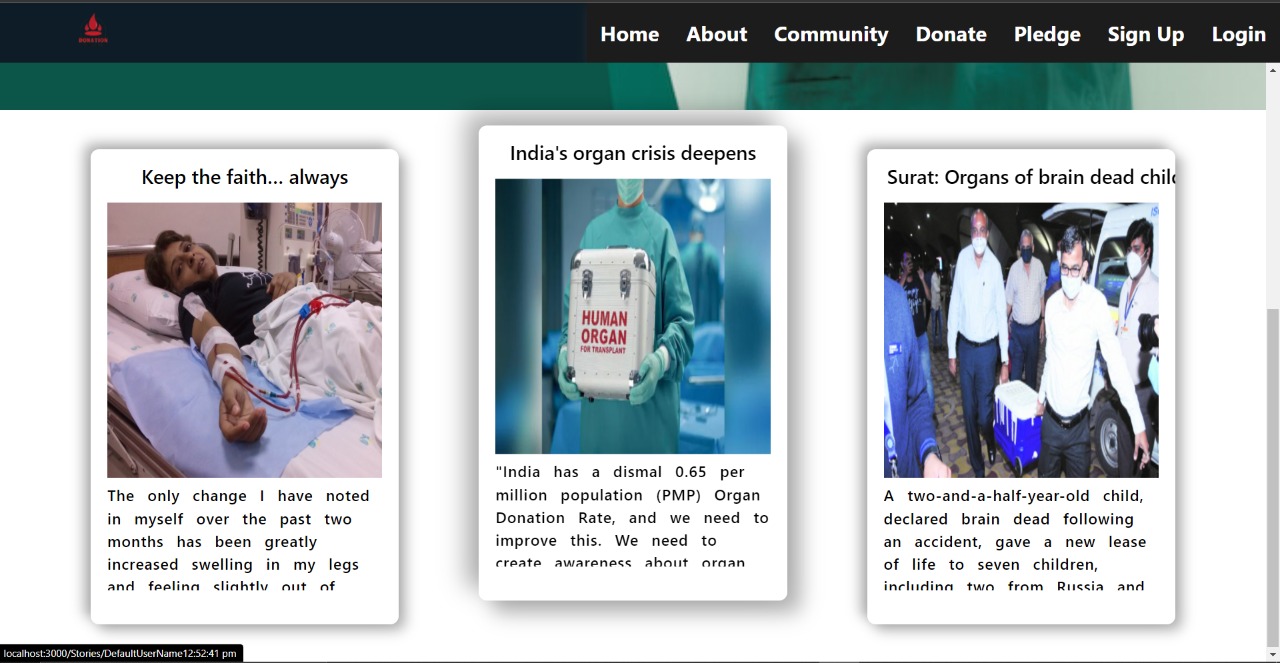
Home Page:



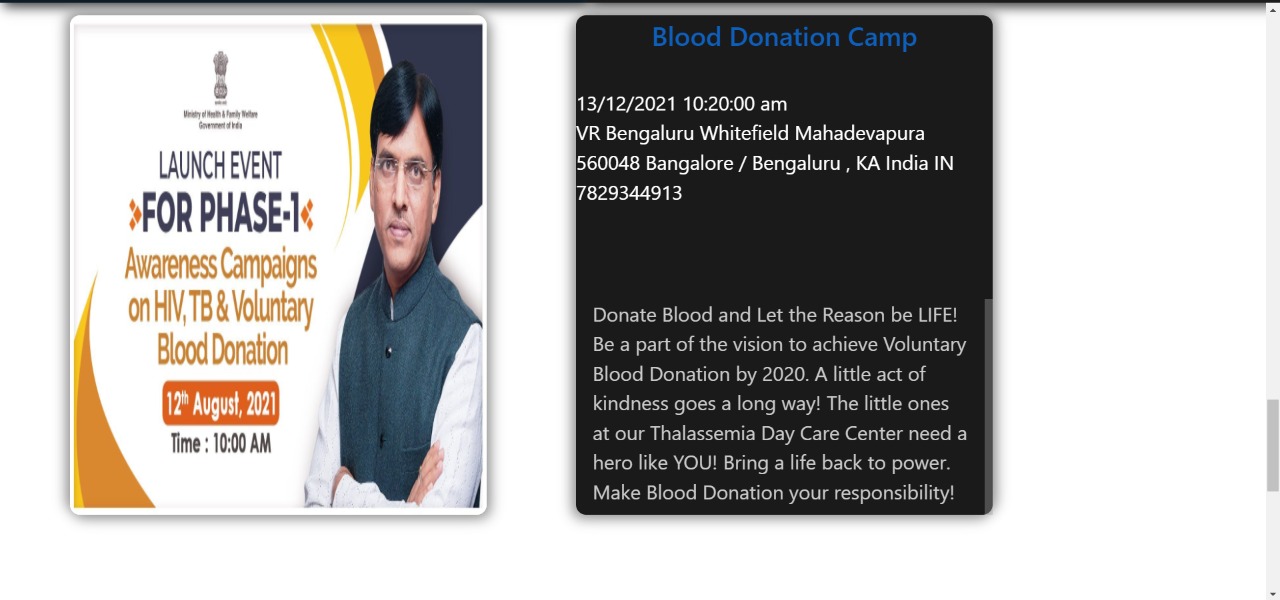
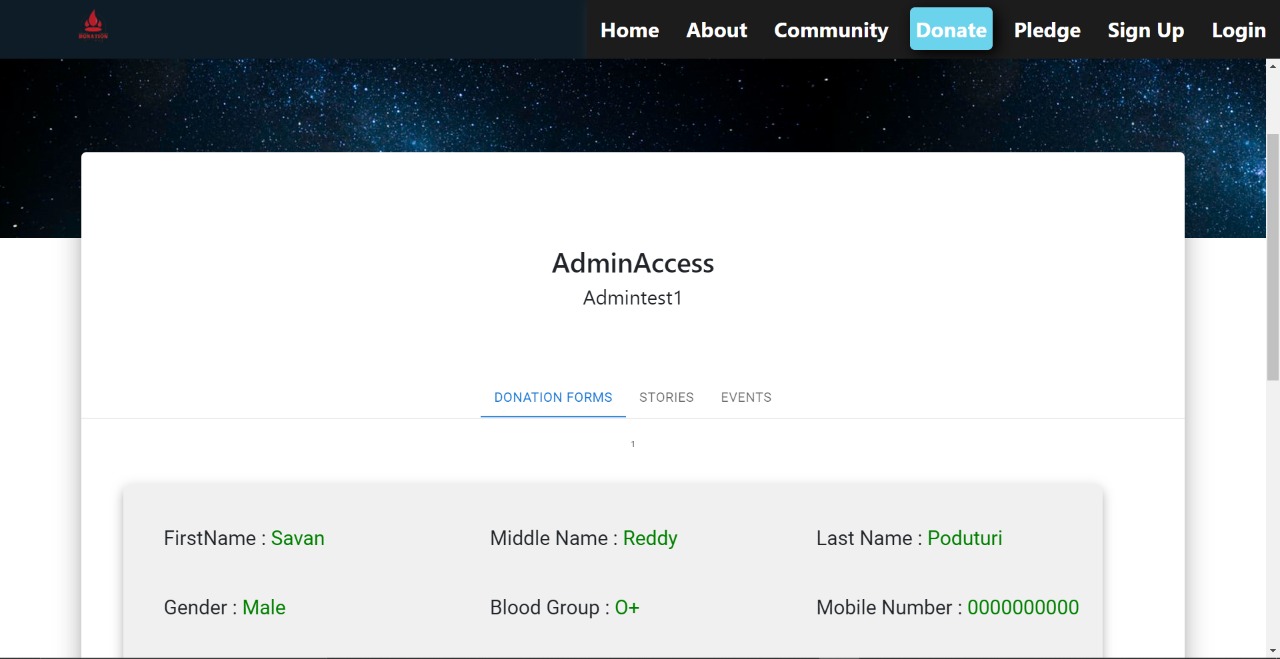
Sign up page: Login page:



Stories page:



Admin page: Event page:



**RESULTS AND DISCUSSIONS**

Understanding our problem statement we have come up with a user friendly website that can be accessible by all.We created a website in which a donor can upload his information an register himself as a donor. This information will be stored in ou database.When there is a requirement,they can contact our administration for the same.When our administration gets a request we sort the data as per the requirements and provide the stakeholder with details f the donor.

By uploading your information and registering yourself on our website one can save a person’s life. One can even share their donation and survival stories on our website and inspire others to register themselves. We have also provided information about why should one donate and the procedure of each organ donation. Finally the donor information is stored in our database and is secure. This information is accessible by the administration as per the requirements.

**CONCLUSION AND FUTUREWORK**

The old-fashioned way of approaching our blood-related relatives and family members for organs and blood works in a few cases but mostly fails. With our solution one can survive and receive the required organs and blood in a short span of time.

By uploading your information and registering yourself on our website one can save a person’s life. One can even share their donation and survival stories on our website and inspire others to register themselves.We have also provided information about why should one donate and the procedure of each organ donation. Finally the donor information is stored in our database and is secure. This information is accessible by the administration as per the requirements.

We hope that our system will bring a significant change in our locality. Anew, updated and expanded edition of our project is to implement thecloud computing function to increase the rise of blood donors, and to translate the system into Regional language because it is local for every state so it can have dual languages. This work is not a one-time job but is continuous work to be adopted for further research and the system can be used in various “what-if” scenarios.  This work may be extended to interconnect all the blood donors societies in India.

**REFERENCES**

1. <https://www.organdonation.nhs.uk/>
2. <https://www.nhsbt.nhs.uk/>
3. <https://www.organindia.org/>
4. <https://www.notto.gov.in/>
5. <https://dghs.gov.in/content/1353_3_NationalOrganTransplantProgramme.aspx>