

LABORATORY RECORD

Name : BARANI R
Roll No. : 2019239003
Class : M.Sc., (Integrated -5 Years) - CS
Course Code & Title : XC5561 – SOFTWARE
DEVELOPMENT LABORATORY



DEPARTMENT OF MATHEMATICS
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI - 600 025

**DEPARTMENT OF MATHEMATICS
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI - 600 025**

BONA FIDE CERTIFICATE

Name : BARANI R

Roll No. : 2019239003

Programme :M.Sc(Integrated)

Branch : Computer Science

Semester : V

*Certified to be the bona fide record of work done by the above student in **XC5561 – SOFTWARE DEVELOPMENT LABORATORY** course during the Semester **V** of the academic year **2021-2022** submitted for the Practical Examination held on **04/02/2022.***

Lab Course Instructor

Head of the Department

INDEX

| S.NO. | DATE | TITLE | PAGE NO. | SIGNATURE |
|-------|----------|-------------------------------------|----------|-----------|
| 1 | 09.09.21 | Feasibility Report | 4 | |
| 2 | 23.09.21 | Software Requirements Specification | 5 | |
| 3 | 28.10.21 | UML diagrams | 15 | |
| 4 | 30.12.21 | Source Code Implementation | 20 | |
| 5 | 30.12.21 | Conclusion | 45 | |

Feasibility Report

Reuse:Donation Portal

This Web application is accessible on both mobiles and desktop, which deals with the concept of accepting and distributing donations so that people could be benefited more effectively. Such donations includes books, clothes and any things which may be useful. This software will make donations more easier and will help to reach more people who are in need.

Objective:

This application will provide services for two types of users:

1.Users who are in need.

They will be able to create a profile with verified information such as address,name,etc.. and accept donations from website.

2.Users who want to donate.

They will be able to donate as well as remove donations.

Client Analysis:

This project targets people who are willing to help/donate others and people who need those. This concept will help the people to get a high chance of exchanging their commodities as per their needs via a modern donation system.

Market Analysis:

On the information we gathered, we were able to find NGO management systems which revolves around, only a particular organisation. The application which we develop will connect many individual people who are willing to donate their things, which are beneficial to the others. The process of making donations comes in handy.

Economic Feasibility:

Our software does not need any materialistic resources so the cost factor is nil. The tools which we require to develop and implement this software are all open source.The project is expected to be completed on or before the proposed deadline.

Technical Feasibility:

This software is feasible in technical sense. It can be integrated with other systems as the necessary tools are available. We only require a fair amount of time to acquire a good knowledge about how to use them.

Conclusion:

This project is feasible under all aspects. The motive of this project is to promote social welfare and not for profits. However, the most important profit in this project is the learning experience and closing the gap in terms of quality of life.

SOFTWARE REQUIREMENTS SPECIFICATION

PREFACE

This Software Requirements Specification (SRS) document is intended for users of the Reuse:Donation portal and people evaluating it's potential use. This is the introductory version of Reuse:Donation portal. Though there are apps/websites for donations, we meant to make of use of products that are reusable. We designed this software for a diverse groups of acceptors. This Reuse concept may attract a wider range of population to donate as well as reuse. This web application is our beginning approach to promote social welfare through donation and environmental conservation by reuse.

INTRODUCTION

Reuse:Donation portal is a website which is developed to cater the needs of people by donating the unused, in short a conservation of resources via a Donation system. This application will make the process of donation, simple and least troubling. The application also handles services such as providing the platform for all the people to reach out to more donors who are interested to donate, so that they can be put to good use.

The launch of the application will close the gap in terms of quality of life and sow an awareness about conservation of resources among all the young minds. This application is aimed to help the society and to create a sense of solidarity through modern techniques.

GLOSSARY

1. **Acceptor** - Any person who is interested and can be benefited from the reusable materials donated.
2. **Donor** - Any individual who can/wants to donate any item which can be reused.

DONOR

Donate: Donor can donate useful things.

View/Reject Acceptance: Donor have the freedom to reject the acceptance.

View Receiver's Profile: Once the donation is accepted, the donor will be notified about acceptor's details in Email.

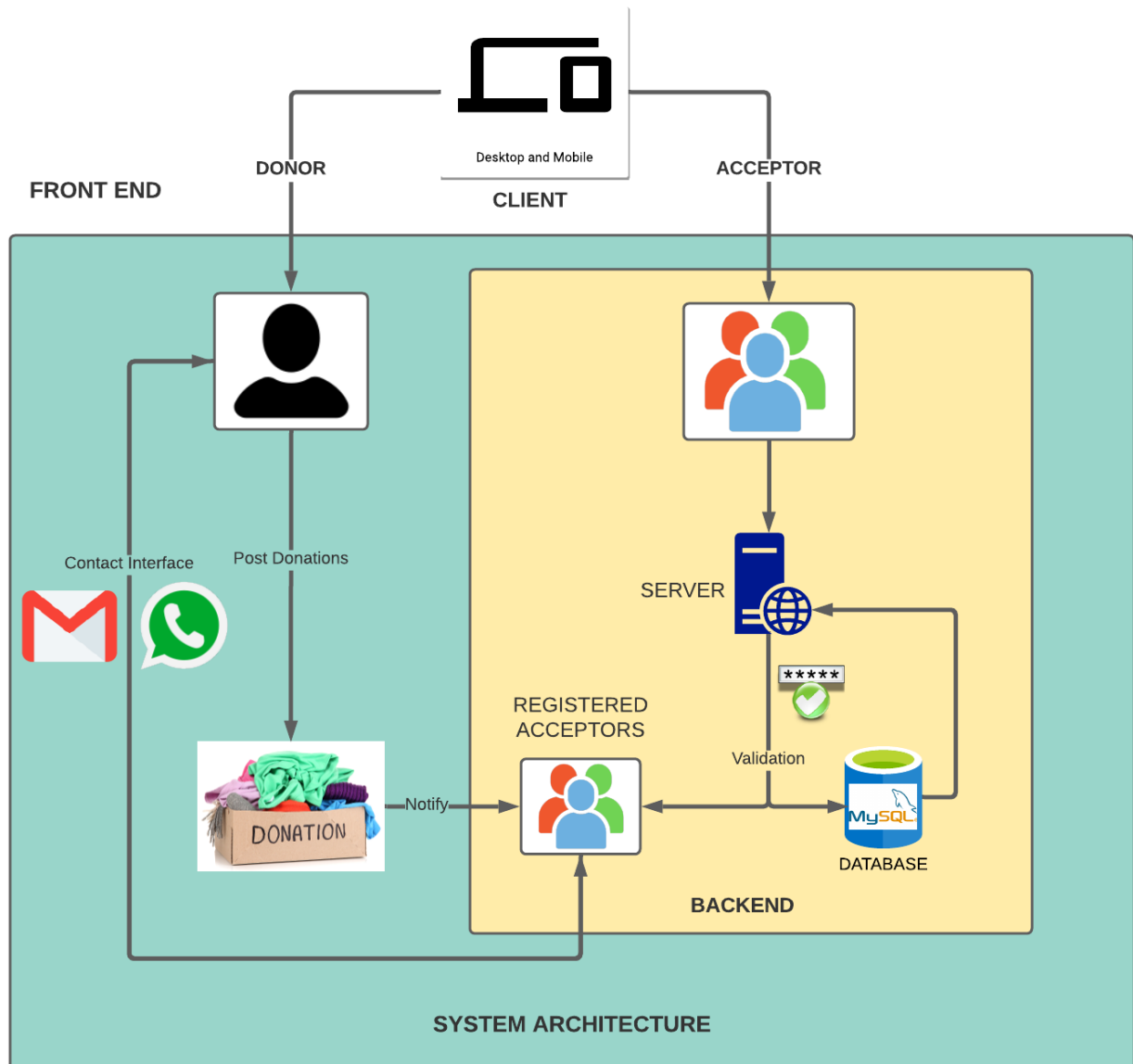
Delete Donations: Donors can remove the donation if they wish.

ACCEPTOR

Accept/Reject Donations: Once an acceptor accepts the donation, they will be provided with donor's contact details. If not required, they can reject.

SYSTEM ARCHITECTURE

SYSTEM DESIGN:



SYSTEM REQUIREMENTS SPECIFICATION

Functional Requirements:

Acceptor account management:

- Acceptor shall create a profile.
- Acceptor shall register with their Email ID and Contact number.
- Users shall log out.

Profile Management:

The Acceptor profile contains the following information:

- Name
- Contact Number
- E-mail ID.

Donation Management:

- Donors shall provide their mail-ID and mobile number to donate.
- Donors shall provide their pin-code and a landmark nearby them.
- Notifications shall be sent to the donor when the donation is accepted.

Non-functional requirements

Performance:

The application performance should be optimised, and response time should be minimized (<10 seconds).

Scalability:

The application should be highly scalable; since it is meant to be used by people who use different devices(PCs and Mobile Phones).

Extensibility:

The application should be extensible to allow adding other services in the future, and allowing integration with other APIs like Google maps to match with nearby donors and acceptors.

Integration:

The system should have the ability to extend its requirements.

Security:

The system should be highly secure since only authenticated users can have access to the server. It should respect the 3 aspects:

- **Confidentiality:**

The user's data shall be kept confidential.

- **Integrity:**

Editing data shall be authorized only to owners of the accounts.






- **Availability:**

The application shall be available 98% of the time.

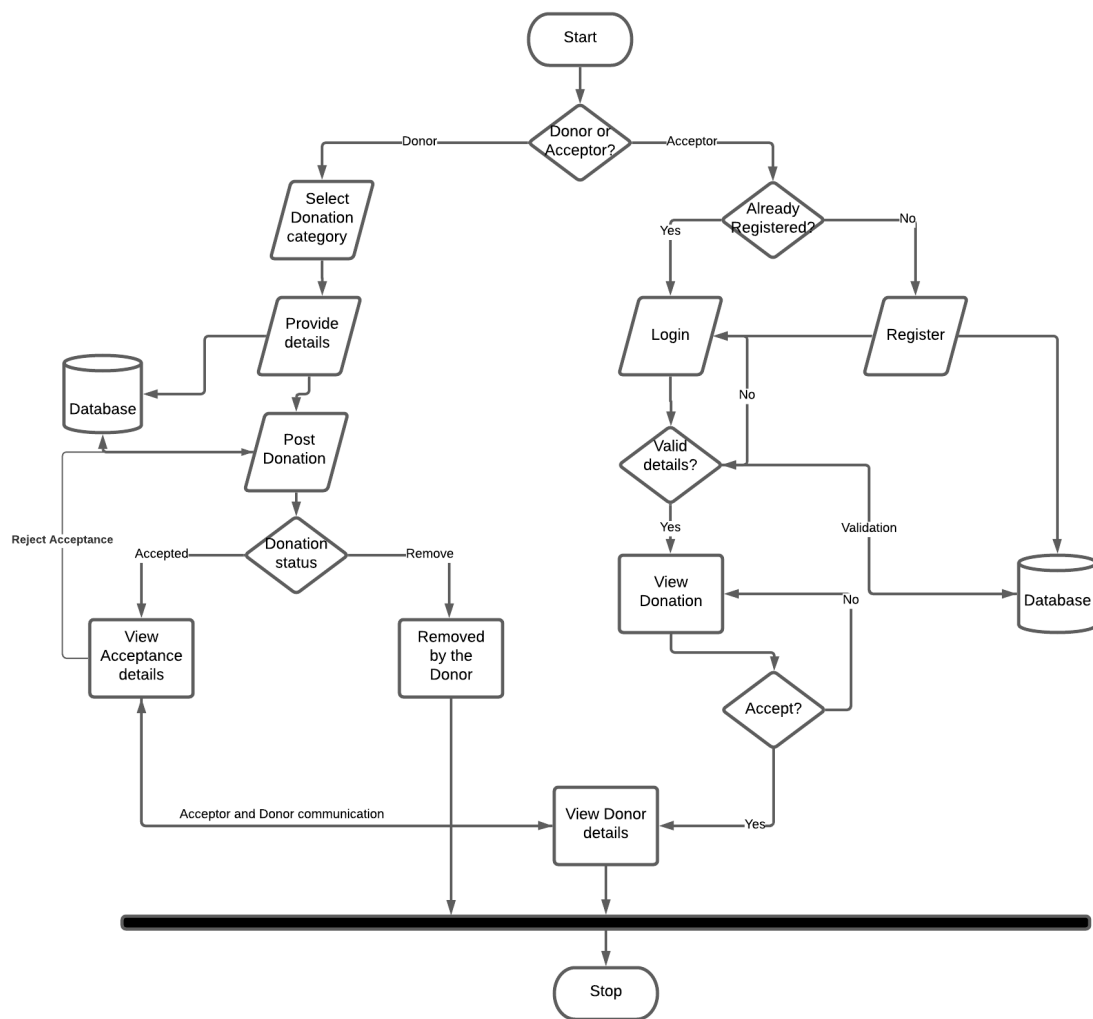
Maintainability:

The system should be easily maintainable to allow for additional upgrades that can be implemented in the future.

Tools

| | |
|---|--|
|  | Express.js is the framework used along side with node.js to implement our project |
|  | MySQL is the database used design tables and store data used in this project. |
|  | NPM (Node Package Manager) is the package manager used for providing the Node.js packages that we need in this project. |
|  | Visual Studio Code is the IDE used to develop the project. |
|  | Node.js is the Javascript run-time environment used to implement our project. |

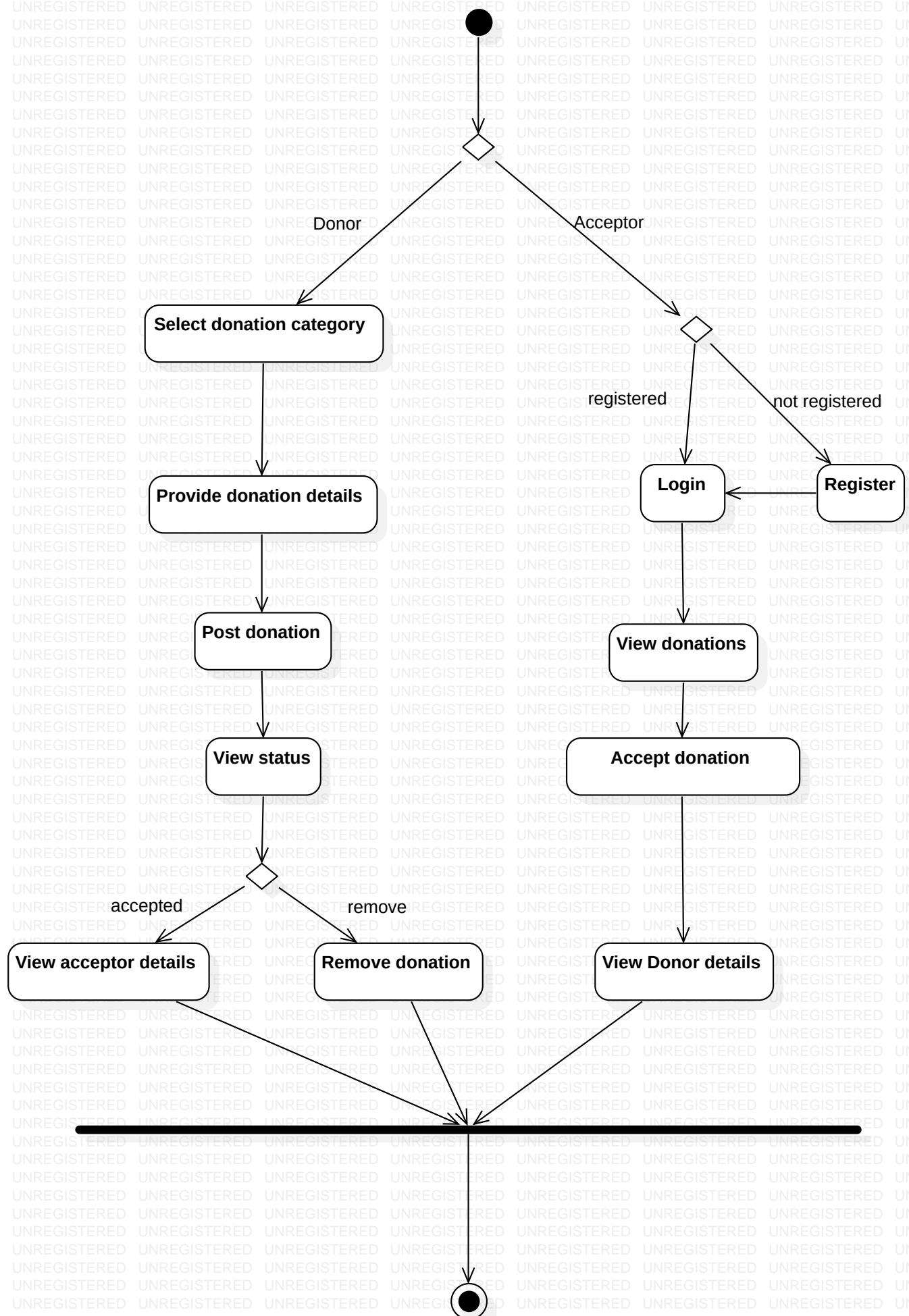
SYSTEM MODELS:



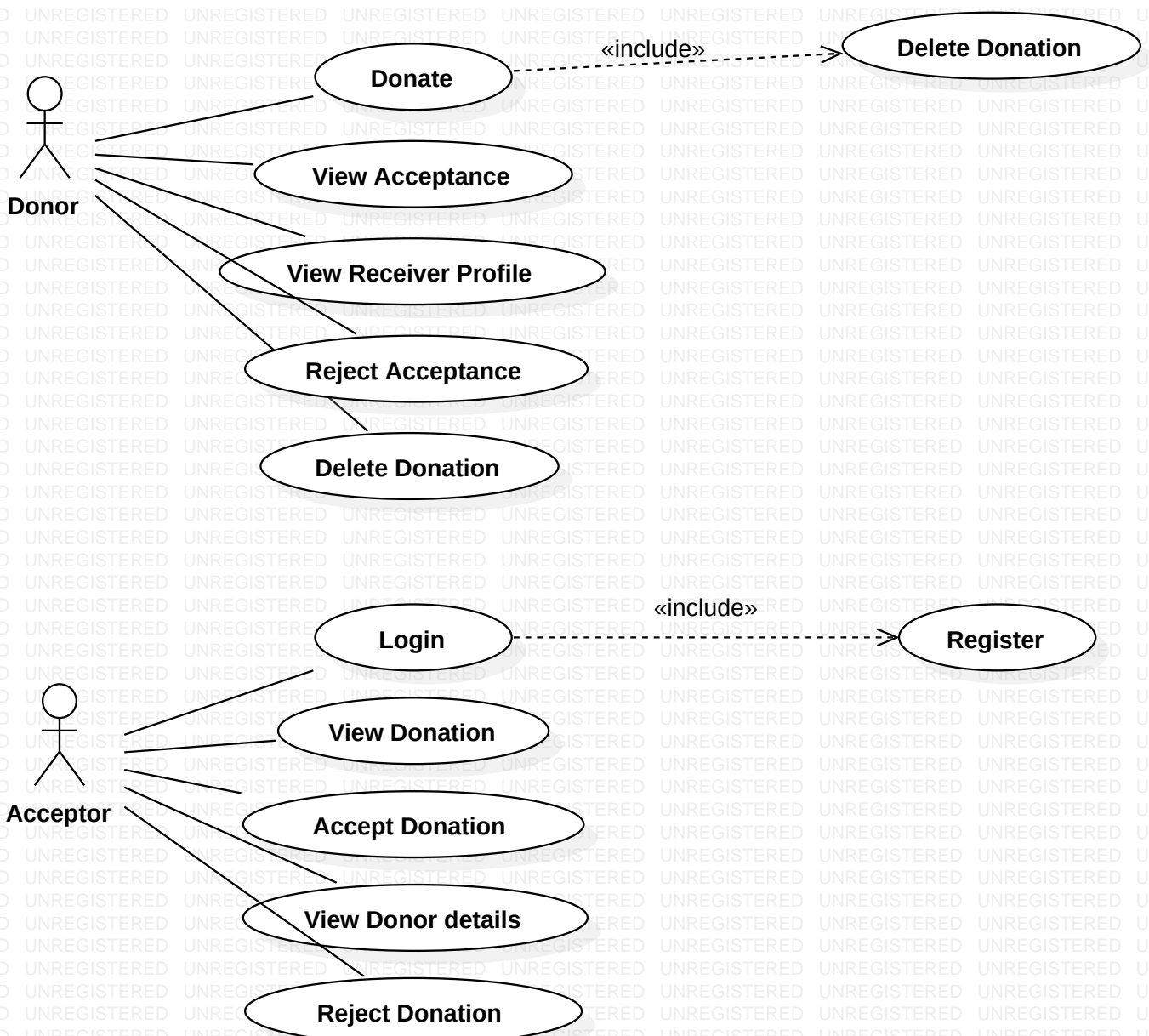
SYSTEM EVOLUTION

The idea of Reuse:Donation portal has varied possibilities with wide scope, to the future it can be integrated with mapping exploration to find nearby donors and acceptors. Also, it can be further developed to promote with navigation system for pointing people and exploring lengths of spread. In future,with desired support the mode of contact may change to text message instead of existing E-mail communication for easy and convenient access.

ACTIVITY DIAGRAM



USE CASE DIAGRAM



sd DonorSequence



DONOR: Actor

DONATIONS PAGE

DATABASE

1 : donate()

2 : viewStatus

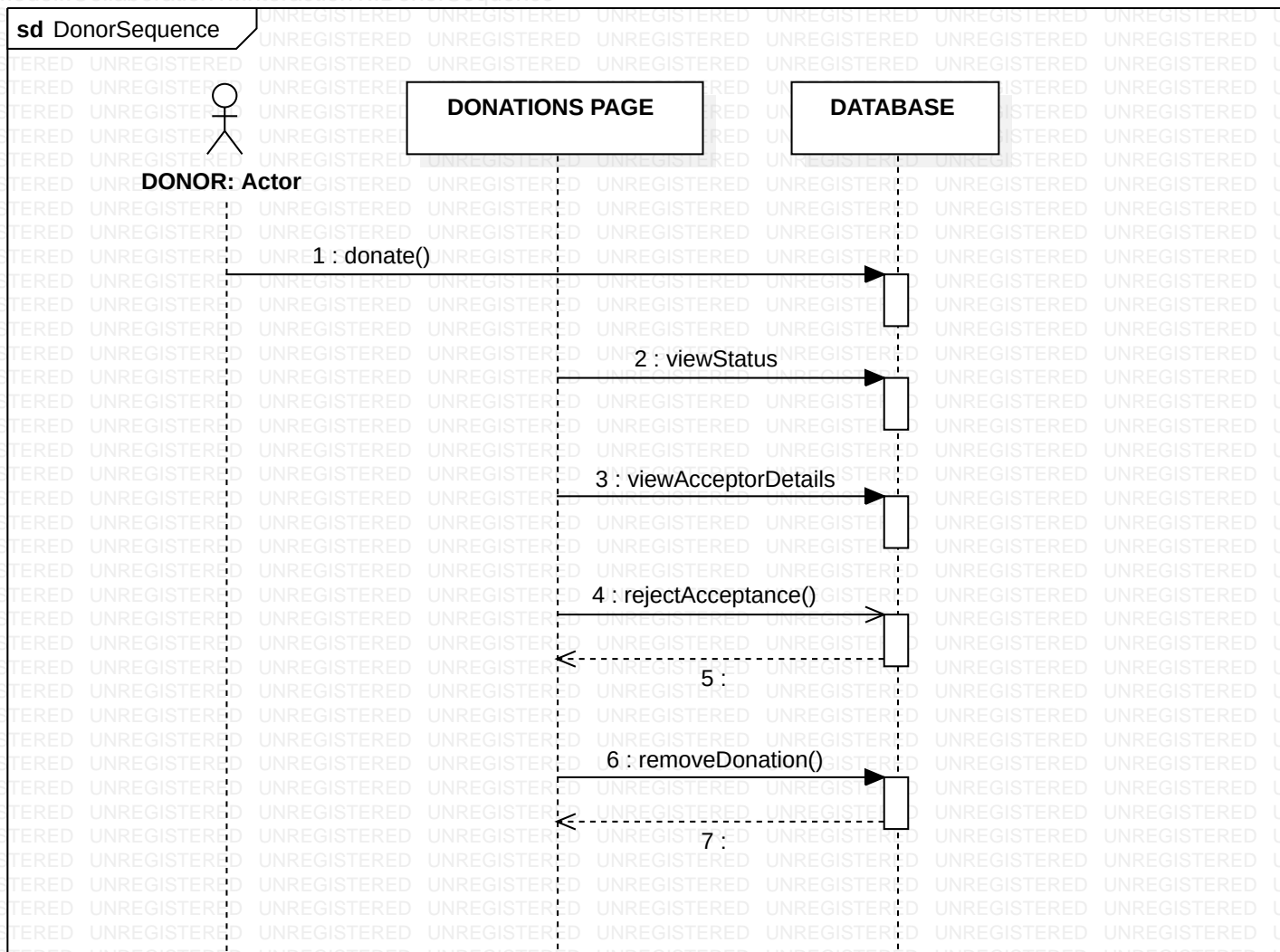
3 : viewAcceptorDetails

4 : rejectAcceptance()

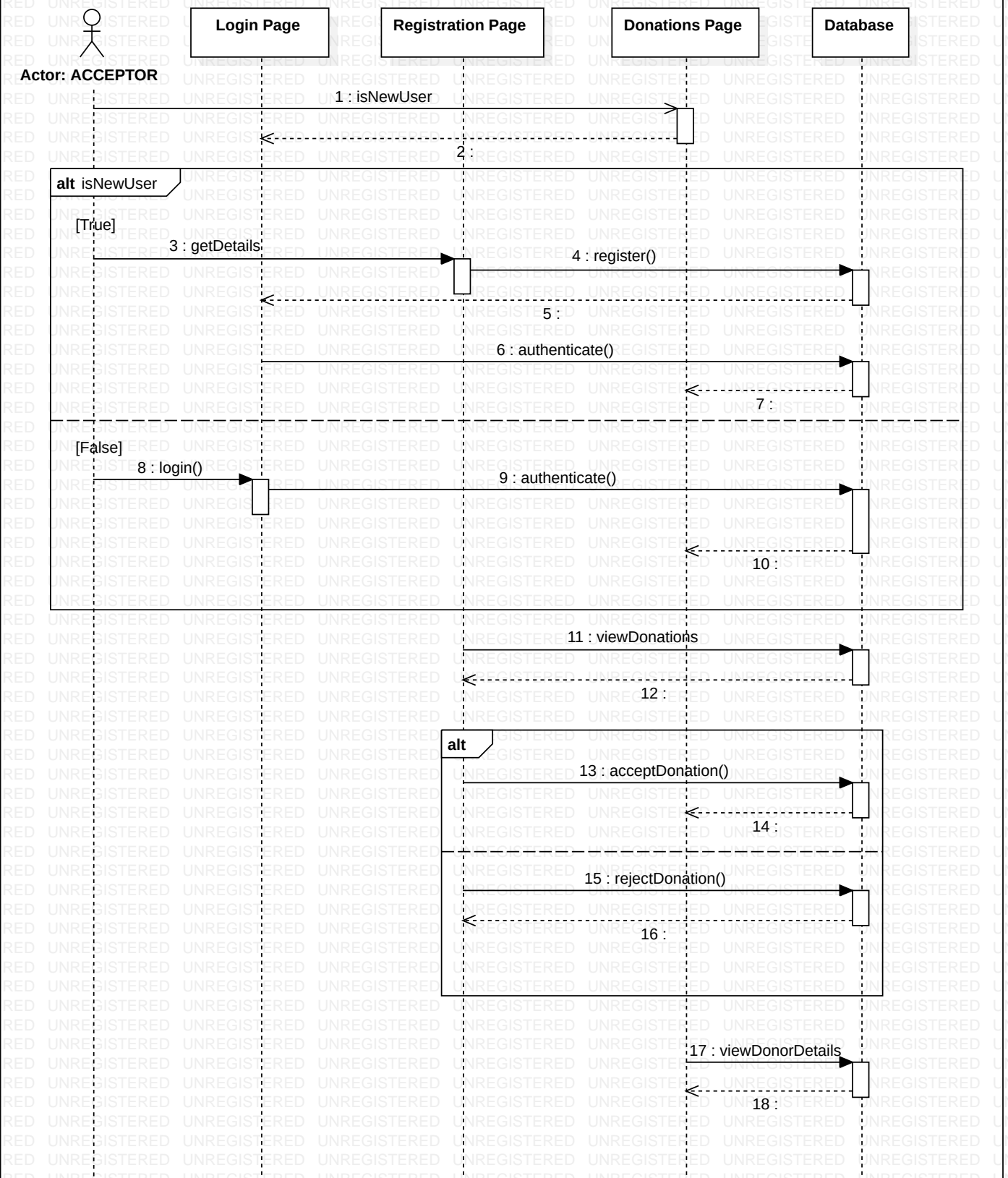
5 :

6 : removeDonation()

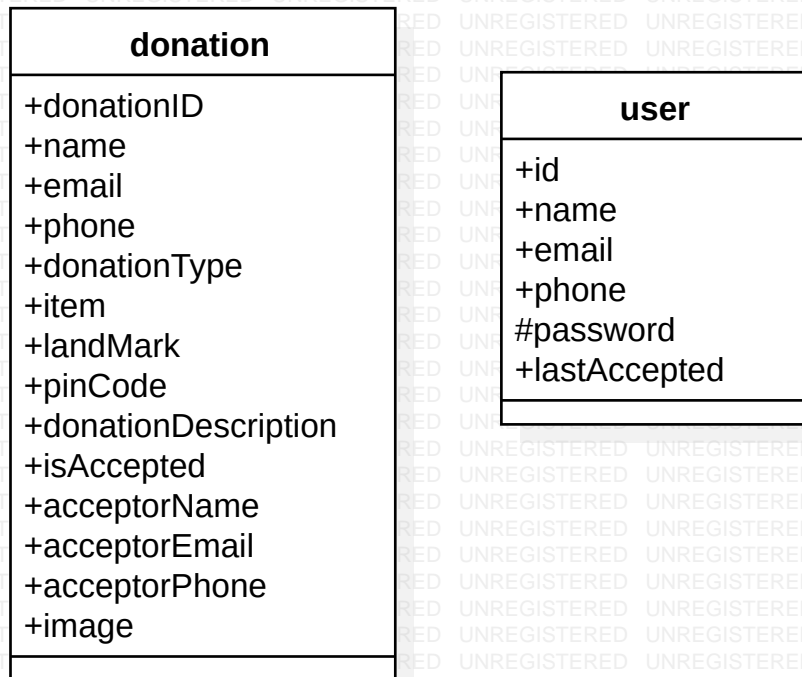
7 :



sd Acceptor Sequence



CLASS DIAGRAM



Source Code Implementation

createTable.txt:

```
CREATE TABLE `user`(  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(255) NOT NULL,  
  `email` varchar(255) NOT NULL unique,  
  `phone` bigint(10) NOT NULL,  
  `password` varchar(255) NOT NULL,  
  `lastAccepted` BIGINT NOT NULL,  
  PRIMARY KEY (`id`, `email`));
```

```
CREATE TABLE donation (  
  `donationID` varchar(255) NOT NULL,  
  `name` varchar(255) NOT NULL,  
  `email` varchar(255) NOT NULL,  
  `phone` bigint(10) NOT NULL,  
  `donationType` varchar(255) NOT NULL,  
  `item` VARCHAR(255) NOT NULL,  
  `landMark` VARCHAR(255) NOT NULL,  
  `pinCode` BIGINT(6) NOT NULL,  
  `donationDescription` VARCHAR(255),  
  `isAccepted` boolean NOT NULL,  
  `acceptorName` varchar(255) NOT NULL,  
  `acceptorEmail` varchar(255) NOT NULL,  
  `acceptorPhone` bigint(10) NOT NULL,  
  `image` LONGTEXT NOT NULL,  
  PRIMARY KEY (`donationID`));
```

config.js:

```
var mysql = require('mysql');  
var connection = mysql.createConnection({  
  host      : 'localhost',  
  user      : 'root',  
  password  : '',  
  database  : 'UserDB'  
});  
connection.connect(function(err){  
  if(!err) {  
    console.log("Database is connected");  
  } else {  
    console.log("Error while connecting with database");  
  }  
});  
module.exports = connection;
```

Index.js:

```
var express=require("express");
var bodyParser=require('body-parser');
var session = require('express-session');
var path = require('path');
var Cryptr = require('cryptr');
var nodemailer = require('nodemailer');
var cors = require('cors');
var alert = require('alert');
var multer = require('multer');
const ejsLint = require('ejs-lint');
cryptr = new Cryptr('myTotalySecretKey');
var fs = require('fs');
var https = require("https");
var connection = require('./config');

var app = express();

//app.use(cookieParser);
app.use(session({
  secret: 'secret',
  cookie: {maxAge: 600000},
  resave: true,
  saveUninitialized: true
}
));
app.use(bodyParser.urlencoded({extended : true}));
app.use(bodyParser.json());
app.use(cors());
app.set('view engine', 'ejs');
app.use(express.static(path.join(__dirname + "/images")));

var upload = multer({
  storage: multer.memoryStorage()
});

const sslServer = https.createServer(
  {
    key: fs.readFileSync(path.join(__dirname, 'ssl', 'server.key')),
    cert: fs.readFileSync(path.join(__dirname, 'ssl', 'server.crt')),
  },
  app
)
```

```
sslServer.listen(3000, () => console.log(`SSL server is running at  
https://localhost:3000`))
```

```
app.get('/', function (request, response) {  
  response.sendFile(path.join( __dirname + "/home.html" ) );  
});
```

```
app.get('/register.html', function (request, response) {  
  response.sendFile(path.join( __dirname + "/register.html" ) );  
});
```

```
app.get('/login.html', function (request, response) {  
  response.sendFile(path.join( __dirname + "/login.html" ) );  
});
```

```
app.get('/donations', function (request, response){  
  if(!request.session.email){  
    response.redirect('/login.html');  
  }  
  connection.query("SELECT * FROM donation WHERE  
isAccepted=0", (err, rows, fields)=>{  
    if (err) throw err  
    response.render(path.join(__dirname+'/views/donations'), {  
      donations : rows,  
      name : request.session.name,  
      email: request.session.email,  
      phone: request.session.phone  
    });  
  })  
});
```

```
app.get('/NEWdonations', function (request, response){  
  let donorEmail = request.session.donorEmail;  
  connection.query("SELECT * FROM donation WHERE  
email=?", [donorEmail], (err, rows, fields)=>{  
    if (err) throw err  
    else{  
      response.render(path.join(__dirname+'/views/NEWdonations'), {  
        donations : rows,  
        name : rows[0].name,  
        email: rows[0].email,  
        phone: rows[0].phone  
      });  
    }  
  });  
});
```

```

    });
}
})
});

app.get('/authOTP', function (request, response){
    if(!request.session.email){
        response.redirect('donationForm')
    }
    response.render(path.join(__dirname+'/views/authOTP'), {
        name : request.session.name,
        email: request.session.email,
        phone: request.session.phone
    });
});

app.get('/NEWauthOTP', function (request, response){
    response.render(path.join(__dirname+'/views/NEWauthOTP'), {
        name : request.session.donorName,
        email: request.session.donorEmail,
    });
});

app.get('/acceptedDonations', function (request, response){
    let acceptorEmail = request.session.email;
    if(!request.session.email){
        response.redirect('/login.html');
    }
    connection.query("SELECT * FROM donation WHERE isAccepted=1 and
acceptorEmail =?", [acceptorEmail], (err, rows, fields)=>{
        if (err) throw err
    response.render(path.join(__dirname+'/views/acceptedDonations'), {
        donationsAccepted : rows,
        name : request.session.name,
        email: request.session.email,
        phone: request.session.phone
    });
    })
});

app.get('/donationForm', function(request, response) {
    if(!request.session.email){

```

```

        response.redirect('/login.html');
    }
    response.render(path.join(__dirname + '/views/donationForm'), {
        name : request.session.name,
        email: request.session.email,
        phone: request.session.phone
    });
});

app.get('/NEWdonationForm.html', function (request, response) {
    response.sendFile(path.join( __dirname + "/NEWdonationForm.html") );
});

app.get('/NEWgetEmail.html', function (request, response) {
    response.sendFile(path.join( __dirname + "/NEWgetEmail.html") );
});

app.post('/NEWverifyEmail', function (request, response){
    let donorEmail = request.body.donorEmail;
    request.session.donorEmail = donorEmail;
    console.log("DonorEmail: " + donorEmail);
    connection.query("SELECT * FROM donation WHERE email = ?", [donorEmail],
    (error,result,fields)=>{
        if(donorEmail === result[0].email){
            response.redirect('NEWdonations');
        }
        else if(donorEmail !== result[0].email){
            alert("There is no donations in this Email ID");
            response.redirect('/NEWgetEmail.html');
        }
    })
});

app.post('/sendFeedBack', function(request,response){
    let feedEmail = request.body.Email;
    let feedName = request.body.Name;
    let feedMess = request.body.Message;
    var mailOptions = {
        from: 'reusedonation@gmail.com',
        to: 'reusedonation@gmail.com',
        subject: 'Feedback from ' + feedName,
        html: "<strong>Feedback Details</strong>"
            + "<br>Name: " + feedName
    }
});

```



```

        + "<br>Email: " + feedEmail
        + "<br>Mess: " + feedMess
    };

    transporter.sendMail(mailOptions, function(error, info){
        if (error) throw error
    else{
        alert("Feedback submitted sucessfully. Thank you!")
        response.redirect('/');
    });
    })

app.post('/register', function(request, response){
    let initAccepted = 0;
    var encryptedString = cryptr.encrypt(request.body.password);
    var user={
        "name":request.body.name,
        "email":request.body.email,
        "phone":request.body.phone,
        "password":encryptedString,
        "lastAccepted": initAccepted
    }
    let userEmail = request.body.email;
    connection.query('SELECT * from user WHERE email = ?', [userEmail],
function (error, results, fields){
    let Email = results.email;
    console.log(Email);
    if( Email == userEmail){
        alert('Email Id already exists! Provide another Email ID or Go
to Login with the same Email ID');
        response.redirect('/register.html');
    }
    else if(!error){
        connection.query('INSERT INTO user SET ?', user, function
(error, results, fields){
            if (error) {
                console.log("status:false, message:there are some error
with query");
                response.redirect('/register.html');
            }else{
                console.log("status:true, data:results,message:user
registered sucessfully");
                response.redirect("/login.html");
            }
        }
    }
}

```

```

        }
    });
}
});
});

app.post('/authenticate', function(request, response){
    var email=request.body.email;
    var password=request.body.password;

    connection.query('SELECT * FROM user WHERE email = ?', [email], function
(error, results, fields)
    {
        if (error) {
            console.log("status:false,message:'there are some error with
query");
        }else{

            if(results.length >0){
                decryptedString = cryptr.decrypt(results[0].password);
                if(password==decryptedString){
                    console.log("status:true, message:successfully
authenticated");
                    request.session.loggedin = true;
                    request.session.email = results[0].email;
                    request.session.phone = results[0].phone;
                    request.session.name = results[0].name;
                    request.session.save();

                    console.log(request.session.email);
                    console.log(request.session.phone);
                    console.log(request.session.name);

                    response.redirect('donations');

                }else{
                    console.log("status:false,message:Email and password does
not match");
                    alert('Email and password does not match');
                    response.redirect('/login.html');
                }
            }
        }
    }
});

```

```

        else{
            console.log("status:false, message:Email does not exists");
            alert('Email-ID does not exists')
            response.redirect('/register.html');
        }
    }
});
});

var transporter = nodemailer.createTransport({
    host: "smtp.gmail.com",
    port: 465,
    secure: true,
    service: 'Gmail',
    auth: {
        user: 'reusedonation@gmail.com',
        pass: 'pragrbyesfhpbuii'
    }
});

var email;
app.post('/sendOTP',upload.single('image'), function(request, response){
    if (!request.file) {
        alert("Donation Item Image not uploaded");
        response.redirect('donations');
    } else {
        request.session.donationType = request.body.donationType;
        request.session.item = request.body.item;
        request.session.landMark = request.body.landMark;
        request.session.pinCode = request.body.pinCode;
        request.session.imageSource = request.file.filename;
        request.session.image = request.file.buffer.toString('base64');
        request.session.save();

        var otp = Math.floor(100000 + Math.random() * 99999);
        request.session.otp = parseInt(otp);
        console.log(otp);

        const d = new Date();
        let time = d.getTime();
        let dID = request.session.email+"_"+time;
        let acceptValue = 0;

```

```

let acceptorPhone = 0;
donation = {
  "donationID": dID,
  "name" : request.session.name,
  "email" : request.session.email,
  "phone" : request.session.phone,
  "donationType": request.body.donationType,
  "item": request.body.item,
  "landMark": request.body.landMark,
  "pinCode": request.body.pinCode,
  "donationDescription" : request.body.donationDescription,
  "isAccepted" : acceptValue,
  "acceptorName": 'NIL',
  "acceptorEmail": 'NIL',
  "acceptorPhone": acceptorPhone,
  "image" : request.file.buffer.toString('base64')
}
request.session.donation = donation;
email = request.session.email;
request.session.save()

var mailOptions = {
  from: 'reusedonation@gmail.com',
  to: email,
  subject: 'Donation confirmation',
  html: "<h3>Hi " + request.session.name + "</h3><h3> OTP to confirm
your donation in Donation portal is </h3>" + "<h1
style='font-weight:bold;'>" + otp + "</h1>" // html body
};

transporter.sendMail(mailOptions, function(error, info){
  if (error) {
    console.log(error);
    alert('You have not yet registered with your email: ' + email);
    response.redirect('/register.html')
  } else {
    console.log('Email sent: ' + info.response);
    console.log('Message sent: %s', info.messageId);
    console.log('Preview URL: %s', nodemailer.getTestMessageUrl(info));
    response.redirect('authOTP');
  }
});
}

```

```

});
app.post('/NEWsendOTP',upload.single('image'), function(request,
response){
  if (!request.file) {
    alert("Donation Item Image not uploaded");
  } else {
    let donorName = request.body.name;
    let donorEmail = request.body.email;
    let donorPhone = request.body.phone;
    var donationType = request.body.donationType;
    var item = request.body.item;
    var landMark = request.body.landMark;
    var pinCode = request.body.pinCode;
    var imageSource = request.file.filename;
    var image = request.file.buffer.toString('base64');
    request.session.donorEmail = donorEmail;
    request.session.donorPhone = donorPhone;
    request.session.donorName = donorName;

    var otp = Math.floor(100000 + Math.random() * 99999);
    request.session.otp = parseInt(otp);
    console.log(otp);
    const d = new Date();
    let time = d.getTime();
    let dID = donorEmail+"_"+time;
    let isAccepted = 0;
    let acceptorPhone =0

    NEWdonation = {
      "donationID": dID,
      "name" : request.body.name,
      "email" : request.body.email,
      "phone" : request.body.phone,
      "donationType": request.body.donationType,
      "item": request.body.item,
      "landMark": request.body.landMark,
      "pinCode": request.body.pinCode,
      "donationDescription" : request.body.donationDescription,
      "isAccepted" : isAccepted,
      "acceptorName": 'NIL',
      "acceptorEmail": 'NIL',
      "acceptorPhone": acceptorPhone,
    }
  }
});

```

```

    "image" : request.file.buffer.toString('base64')
  }
  request.session.NEWdonation = NEWdonation;
  donorEmail = request.body.email;
  var mailOptions = {
    from: 'reusedonation@gmail.com',
    to: donorEmail,
    subject: 'Donation confirmation',
    html: "<h3>Hi " + request.body.name +
    ",</h3><h3> OTP to confirm your donation in Donation portal is </h3>"
    + "<h1 style='font-weight:bold;'>" + otp + "</h1>"
  };

  transporter.sendMail(mailOptions, function(error, info){
    if (error) {
      console.log(error);
    }
    else {
      console.log('Email sent: ' + info.response);
      console.log('Message sent: %s', info.messageId);
      console.log('Preview URL: %s', nodemailer.getTestMessageUrl(info));
      response.redirect('NEWauthOTP');
    }
  });
}
});

app.post('/verifyOTP',function(request,response){
  if(request.body.otp == request.session.otp) {
    alert("Donated Successfully");
    response.redirect('donate');
  } else{
    alert("Incorrect OTP");
    response.redirect('authOTP');
  }
});

app.post('/NEWverifyOTP',function(request,response){
  if(request.body.otp == request.session.otp) {
    alert("Donated Successfully");
    response.redirect('NEWdonate');
  } else{
    alert("Incorrect OTP");
  }
});

```

```

        response.redirect('NEWauthOTP');
    }
});

app.get('/donate',upload.single('image'), function(request, response) {
    connection.query('INSERT INTO donation SET ?',
request.session.donation,(error, results, fields)=> {
        if (error) throw error
        response.redirect('donations');
    });
});

app.get('/NEWdonate',upload.single('image'), function(request, response) {
    connection.query('INSERT INTO donation SET ?',
request.session.NEWdonation,(error, results, fields)=> {
        if (error) throw error
        response.redirect('NEWdonations');
    });
});

app.post('/acceptDonation', function(request, response){
    var radioID = request.body.radioID;
    var acceptorEmail = request.session.email;
    console.log(radioID, acceptorEmail);
    connection.query('SELECT * FROM donation where donationID = ?',
[radioID],(error, results, fields)=> {
        if (error) throw error
        else{
            console.log(acceptorEmail);
            connection.query('SELECT * FROM user where email = ?',
[acceptorEmail],(error, result, fields)=> {
                if (error) throw error
                else{
                    const d = new Date();
                    let timeNow = d.getTime();
                    let timeAccepted = result[0].lastAccepted;
                    let timeInterval = 3600000;
                    let timeDiff = timeNow - timeAccepted;
                    console.log(timeDiff);

                    if(timeDiff <= timeInterval){
                        alert('Sorry! You can accept donations only after an hour of
your previous acceptance');

```

```

        response.redirect('donations');
    }
    else if(timeDiff > timeInterval){

var acceptorName = request.session.name;
var accpetorPhone = request.session.phone;
console.log(acceptorEmail, acceptorName, accpetorPhone, radioID);

        connection.query("UPDATE user SET lastAccepted = ?",[timeNow],(error,
res, fields)=> {
            if (error) throw error });

        connection.query("UPDATE donation SET isAccepted=1, acceptorName=?,
acceptorEmail=?, acceptorPhone=? WHERE donationID = ?",[acceptorName,
acceptorEmail, accpetorPhone, radioID],(error, res, fields)=> {
            if (error) throw error
            else{
                var mailOptions = {
                    from: 'reusedonation@gmail.com',
                    to: results[0].email,
                    subject: "Donation item - "+results[0].item + " was accepted",
                    html: "Donation item "+results[0].item+ " which you have posted in
Donation portal have been accepted.<br><br> <strong>Acceptor
details</strong> <br> Name: "
                        + acceptorName + "<br> Email: " + acceptorEmail + "<br> Phone: "+
accpetorPhone +
                            "<br><br><br> <strong>NOTE: Please remove your donation once it
has been collected by the acceptor.</strong>"
                };
                transporter.sendMail(mailOptions, function(error, info){
                    if (error) throw error
                    else {
                        console.log('Email sent: ' + info.response);
                        console.log('Message sent: %s', info.messageId);
                        console.log('Preview URL: %s',
nodemailer.getTestMessageUrl(info));
                        response.redirect('acceptedDonations');
                    }
                })
            }
        });
    }
}

```



```

    }

  })

}

}))

});

app.post('/rejectDonations', function(request, response) {
  var radioID = request.body.radioID;
  var acceptorEmail = request.session.email;
  connection.query('SELECT * FROM donation where donationID = ?',
[radioID],(error, results, fields)=> {
    if (error) throw error
    connection.query("UPDATE donation SET isAccepted=0, acceptorEmail='NIL'
WHERE donationID = ?",[radioID],(error, res, fields)=> {
      if (error){
        response.send(error);
      }
      else{
        console.log('Rejected Sucessfully');
        connection.query('SELECT * FROM user where email = ?',
[acceptorEmail],(error, result, fields)=> {
          if (error) throw error
          else{
            let timeAccepted = result[0].lastAccepted;
            let timeInterval = 3600000;
            let timeUpdated = timeAccepted - (timeInterval+1);
            console.log(timeUpdated);

            connection.query("UPDATE user SET lastAccepted =
?",[timeUpdated],(error, res, fields)=> {
              if (error) throw error
              else{
                var mailOptions = {
                  from: 'reusedonation@gmail.com',
                  to: results[0].email,
                  subject: "Donation item - "+results[0].item + " accepted was
cancelled",

```

```

        html: "Donation item "+results[0].item+ " which was accepted
earlier has been turned down."
        + "<br> <strong>Donation Item Details</strong> <br>"
        + "Item: " + results[0].item
        + "<br>Catgory: " + results[0].donationType
        + "<br>Donation Description: " + results[0].donationDescription
    };
    transporter.sendMail(mailOptions, function(error, info){
        if (error) {
            console.log(error);
            response.write(error);
        } else {
            console.log('Email sent: ' + info.response);
            console.log('Message sent: %s', info.messageId);
            console.log('Preview URL: %s',
nodemailer.getTestMessageUrl(info));
            response.redirect('acceptedDonations');
        }
    })
    }
    });
}

});

app.post('/NEWrejectAcceptance', function(request, response) {
    var radioID = request.body.radioID;

    connection.query("UPDATE donation SET isAccepted=0, acceptorName='NIL',
acceptorEmail='NIL', acceptorPhone=0 WHERE donationID =
?",[radioID],(error, res, fields)=> {
        if (error){
            response.send(error);
        }
        else{
            console.log('Acceptance Rejected Sucessfully');
            response.redirect('NEWdonations');
        }
    }
});

```

```

    })
  });

  app.post('/NEWdeleteDonation', function(request, response) {
    var radioID = request.body.radioID;

    connection.query("DELETE FROM donation WHERE isAccepted=1 AND donationID
= ?",[radioID],(error, res, fields)=> {
      if (error){
        response.send(error);
      }
      else{
        console.log('Donation Deleted Sucessfully');
        response.redirect('NEWdonations');
      }
    })
  });

  app.post('/NEWremoveDonation', function(request, response) {
    var radioID = request.body.radioID;

    connection.query("DELETE FROM donation WHERE isAccepted=0 AND donationID
= ?",[radioID],(error, res, fields)=> {
      if (error){
        response.send(error);
      }
      else{
        console.log('Donation Removed Sucessfully');
        response.redirect('NEWdonations');
      }
    })
  });

  app.get('/logout',function(request,response){
    request.session.destroy((err) => {
      if(err){
        console.log(err);
      }
      else
      {
        response.redirect('/');
      }
    })
  })

```

```
});
```

donations.ejs:

```
<title>Donations</title>

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="icon" type="image/x-icon" href="favicon.png">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Lato">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-a
wesome.min.css">
<style>
body {font-family: "Lato", Arial, Helvetica, sans-serif}

.topnav {
  overflow: hidden;
  background-color: #333;
}

.topnav a {
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}
.topnav a:hover {
  background-color: #ddd;
  color: black;
}

.topnav a.active {
  background-color: #04AA6D;
  color: white;
}

.topnav .icon {
  display: none;
```

```

}

@media screen and (max-width: 600px) {
  .topnav a:not(:first-child) {display: none;}
  .topnav a.icon {
    float: right;
    display: block;
  }
}

```

```

@media screen and (max-width: 600px) {
  .topnav.responsive {position: relative;}
  .topnav.responsive .icon {
    position: absolute;
    right: 0;
    top: 0;
  }
  .topnav.responsive a {
    float: none;
    display: block;
    text-align: left;
  }
}

```

```

table {
  border-collapse: collapse;
  border-spacing: 0;
  width: 100%;
  border: 1px solid #ddd;
}

```

```

th, td {
  text-align: left;
  padding: 8px;
}

```

```

th {
  padding-top: 12px;
  padding-bottom: 12px;
  text-align: left;
  background-color: #04AA6D;
  color: white;
}

```

```

tr:nth-child(even){background-color: #f2f2f2}

input[type=submit] {
  background-color: #04AA6D;
  color: white;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  float: right;
}
input[type=submit]:hover {
  background-color: #45a049;
}

/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}

/* Responsive layout - when the screen is less than 600px wide, make the
two columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
  input[type=submit] {
    width: 100%;
    margin-top: 0;
  }
}
</style>

<body>
<!-- Navbar -->
<div class="topnav" id="myTopnav">
  <a href="/">HOME</a>
  <a href="donations" class="active">VIEW/ACCEPT DONATIONS</a>
  <a href="acceptedDonations">ACCEPTED DONATIONS</a>

  <a href="/logout" class="w3-bar-item w3-button w3-padding-large
w3-hide-small w3-right">LOGOUT</a>
  <a href="donationForm" class="w3-bar-item w3-button w3-padding-large
w3-hide-small w3-right">DONATE</a>

```

```

</div>
<script>
function myFunction() {
var x = document.getElementById("myTopnav");
if (x.className === "topnav") {
    x.className += " responsive";
} else {
    x.className = "topnav";
}
}
</script>

<h3>Welcome! <%= name %>,</h3>
<form action="/acceptDonation" method="post">
<div style="overflow-x:auto;">

<table>
<tr>
<th>Select</th>
<th>Donation Item Image</th>
<th>Donation Item Detail</th>
<th>Donation Geographic Details</th>
</tr>
<% donations.forEach(element => { %>
<tr id="<%= element.donationID%>" name="<%= element.donationID%>">
<td>
<input type="radio" value="<%= element.donationID%>" name="radioID"
id="radioID">
</td>

<td><a href="#some-id">
<img id="<%= element.donationID %> " alt="Embedded image"
src="data:image/*;base64,<%= element.image %>" width="150" height="120">
</a>
<td>
Category: <%= element.donationType%> <br>
Item Name: <%= element.item%> <br>
<% if (element.donationDescription !== '') { %>
Description: <%= element.donationDescription%> <br>
<% } %>
</td>
<td>
Landmark: <%= element.landMark%> <br>

```

```

        Pincode: <%= element.pinCode%> <br>
    </td>
</tr>
<%}); %>
</table>
    <input type="submit" class="btn_AcceptDonation" value="Accept
Donation">
</form>

```

acceptedDonations.ejs:

```

<title>Accepted Donations</title>

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="icon" type="image/x-icon" href="favicon.png">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Lato">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-a
wesome.min.css">
<style>
body {font-family: "Lato", Arial, Helvetica, sans-serif}

.topnav {
    overflow: hidden;
    background-color: #333;
}

.topnav a {
    float: left;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}

.topnav a:hover {
    background-color: #ddd;
    color: black;
}

```



```

.topnav a.active {
  background-color: #04AA6D;
  color: white;
}

.topnav .icon {
  display: none;
}

@media screen and (max-width: 600px) {
  .topnav a:not(:first-child) {display: none;}
  .topnav a.icon {
    float: right;
    display: block;
  }
}

@media screen and (max-width: 600px) {
  .topnav.responsive {position: relative;}
  .topnav.responsive .icon {
    position: absolute;
    right: 0;
    top: 0;
  }
  .topnav.responsive a {
    float: none;
    display: block;
    text-align: left;
  }
}
table {
  border-collapse: collapse;
  border-spacing: 0;
  width: 100%;
  border: 1px solid #ddd;
}

th, td {
  text-align: left;
  padding: 8px;
}

th {

```

```

padding-top: 12px;
padding-bottom: 12px;
text-align: left;
background-color: #04AA6D;
color: white;
}

tr:nth-child(even){background-color: #f2f2f2}

/* Clear floats after the columns */
.row:after {
    content: "";
    display: table;
    clear: both;
}

/* Responsive layout - when the screen is less than 600px wide, make the
two columns stack on top of each other instead of next to each other */
@media screen and (max-width: 600px) {
    input[type=submit] {
        width: 100%;
        margin-top: 0;
    }
}

input[type=submit] {
    background-color: #04AA6D;
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    float: right;
}

input[type=submit]:hover {
    background-color: #45a049;
}

</style>

<body>
<!-- Navbar -->

```

```

<div class="topnav" id="myTopnav">
  <a href="/">HOME</a>
  <a href="donations" >VIEW/ACCEPT DONATIONS</a>
  <a href="acceptedDonations" class="active">ACCEPTED DONATIONS</a>

  <a href="/logout" class="w3-bar-item w3-button w3-padding-large
w3-hide-small w3-right">LOGOUT</a>
  <a href="donationForm" class="w3-bar-item w3-button w3-padding-large
w3-hide-small w3-right">DONATE</a>
</div>
<script>
function myFunction() {
var x = document.getElementById("myTopnav");
if (x.className === "topnav") {
  x.className += " responsive";
} else {
  x.className = "topnav";
}
}
</script>

<h3><%= name %>, here are the list of donations which you have accepted
earlier. You can reject the acceptance, if required.</h3>
<form action="/rejectDonations" method="post">
<div style="overflow-x:auto;">

<table>
  <tr>
    <th>Select</th>
    <th>Doantion Item Image</th>
    <th>Donor Details</th>
    <th>Donation Detail</th>
  </tr>
  <% donationsAccepted.forEach(element => { %>
  <tr id="<%= element.donationID%>" name="<%= element.donationID%>">
  <td>
    <input type="radio" value="<%= element.donationID%>" name="radioID"
id="radioID" >
  </td>

  <td></td>
  <td>

```

```

        Name: <%= element.name %><br>
        Email: <a
href="mailto:<%=element.email%>"><%=element.email%></a></br>
        Phone: <a
href="https://wa.me/<%=element.phone%>"><%=element.phone%></a></br>
    </td>
    <td>
        Category: <%= element.donationType%> <br>
        Item Name: <%= element.item%> <br>
        <% if (element.donationDescription !== '') { %>
        Description: <%= element.donationDescription%> <br>
        <% } %>
        Landmark: <%= element.landMark%> <br>
        Pincode: <%= element.pinCode%> <br>
    </td>
</tr>
<%}); %>
</table>
    <input type="submit" class="btn_RejectDonation" value="Reject
Donation">
</form>

</body>

<script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.js"></scri
pt>

```

CONCLUSION:

The project of ours is an integrated strategy towards a sustainable living by the concept of sharing of commodities through a modern Reuse Donation System. The main purpose of this system is to sow the concept of sharing the least used things to those who are in actual need of them. Recruiting a sufficient number of donors willing to donate reusable products is an emerging challenge. But, to help various people to reach out each other, we do hope that our platform provides support. The shortage of resources in India is due to the increase in the demands of people, but a small step towards the concept of reuse may make living better for the future. Once the satisfaction of helping is realised, we expect creation of opportunities to donate and the need to be well-informed about the need of other people.