Gather & Give
(Food Waste Donation Management System)

A PROJECT COMPONENT REPORT

Submitted by

DURAIVISHVA.R	**(Reg. No. 202004033)**
SURIYA.S	**(Reg. No. 202004153)**

for the Theory Cum Project Component

of

19CS694 – Web User Interface Design

during

VI Semester – 2022 – 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

April 2023

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

Department of Computer Science and Engineering

BONAFIDE CERTIFICATE

Certified that this project component report titled **Gather & Give** is the bonafide work of **R.DURAIVISHVA (Reg.No.202004033),** and **S.SURIYA (Reg.No. 202004153)** who carried out this work under my guidance for the Theory cum Project Component course **"19CS694 – Web User Interface Design"** during the sixth semester.

Dr.S.Karkuzhali, M.E.,Ph.D.	**Dr. J. Raja Sekar,** M.E.,Ph.D.
Assistant Professor	Professor
Course Instructor	Head of the Department
Department of Computer Science & Engg.	Department of Computer Science & Engg.
Mepco Schlenk Engineering College	Mepco Schlenk Engineering College
Sivakasi.	Sivakasi.

Submitted for viva-Voce Examination held at **MEPCO SCHLENK ENGINEERING COLLEGE (Autonomous), SIVAKASI** on **……/……/2023**

Internal Examiner **External Examiner**

ABSTRACT

Wasting food is a common problem in our society. Food waste management is a crucial since it can improve our environmental and economical sustainability. We have identified the use of websites to lessen that nourishment wastage issue through the web application that allows restaurants to donate and share their foods and leftovers with people in need to nearby old age or orphanage homes. This website will enable users (donators) to register, login, choose the orphanage. This application is using the mongodb database. Any user in need to donate the food to the orphanages which was remain to be wasted in parties or functions can donate to the needy people with the help of this application by referring to the nearby homes. The donors will just notify the receivers by sending their willingness of donating to donor through mail. Then, the receiver will contact the donor.

Overall, the Gather & Give system is a promising solution for reducing food waste, addressing food insecurity, and promoting sustainable food practices. By leveraging technology to connect donors with recipients, the system has the potential to make a significant impact on the food industry and the wider community.

ACKNOWLEDGEMENT

First and foremost, we thank the **LORD ALMIGHTY** for his abundant blessings that is showered upon our past, present and future successful endeavors.

We extend our sincere gratitude to our college management and Principal **Dr. S. Arivazhagan M.E., Ph.D.,** for providing sufficient working environment such as systems and library facilities. We also thank him very much for providing us with adequate lab facilities, which enable us to complete our project.

We would like to extend our heartfelt gratitude to **Dr. J. Raja Sekar M.E., Ph.D.,** Professor and Head, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for giving me the golden opportunity to undertake a project of this nature and for his most valuable guidance given at every phase of our work.

We would also like to extend our gratitude and sincere thanks to **Dr.S.Karkuzhali M.E., Ph.D.,** Assistant Professor, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for being our Project Mentor. She has put her valuable experience and expertise in directing, suggesting and supporting us throughout the Project to bring out the best.

Our sincere thanks to our revered **faculty members and lab technicians** for their help over this project work.

Last but not least, we extend our indebtedness towards out beloved family and our friends for their support which made the project a successful one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PERSPECTIVE

The purpose is to introduce Gather & Give for donating nourishment wastage. Gather & Give is the web application and it is responsible to provide information to the users whoever needed to donate the food waste. Here the process of donation is that whenever we're in need we shall login to the application, if we're yet to sign up then create new account and login to the application. The application holds information about homes (i.e) address, contact details,etc. Initially it asks users for selecting items and that request will be notified by the receiver. So that's how donors select their orphanage. This Gather & Give will have only one users, the donors.

Donor - They have the privilege to select the specific orphanage home and donate the waste nourishment to the needy people.

1.2 OBJECTIVES

The main objective is to help the poor and needy people to provide food and to reduce the food wastes that was wasted in any events or occasion. This application allows donator to donate food to the old age and orphanage homes. The following are the objectives,

1. Reduce Food Waste
2. Alleviate Food Insecurity
3. Increase Efficiency and Effectiveness
4. Enhance Transparency and Accountability
5. Foster Collaboration and Innovation

1.3 SCOPE

The scope of the project is to donate the nourishment to the old age home. Besides that, the scope of the project is to provide food for the poor people who lacks food.

CHAPTER 2

REQUIREMENT DESCRIPTION

2.1 FUNCTIONAL REQUIREMENTS

The functional requirement in Gather & Give is the collective information about what are the operations available in the system.

- It should provide the functionality of authentication for the valid users it maybe donor or receiver.
- It should provide the functionality of real time notification sent to the receiver if the donor is ready to donate.
- It should provide the functionality for accepting or rejecting the requests made by the donor.
- It should also provide the functionality for navigation from the current location of the user to the destination place.

2.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirement describes about platform and physical resource required for building the Gather & Give.

- The details of the donor, receiver and their login details are stored in mongodb server. So the information will be reliable and must be stored efficiently in safe manner.
- The application must be user friendly and must be very interactive to the user.
- The donor and receiver can share their location.
- The user must have the internet availability while using this application.

CHAPTER 3

SYSTEM DESIGN

3.1 ARCHITECTURE DESIGN

Architectural diagram implies the flow of the system. The flow starts whether the user has account or not. If the user had an account it directly takes to the login page and after he logged into the system, it enables the user to donate the food for which the user should send notification to receiver.

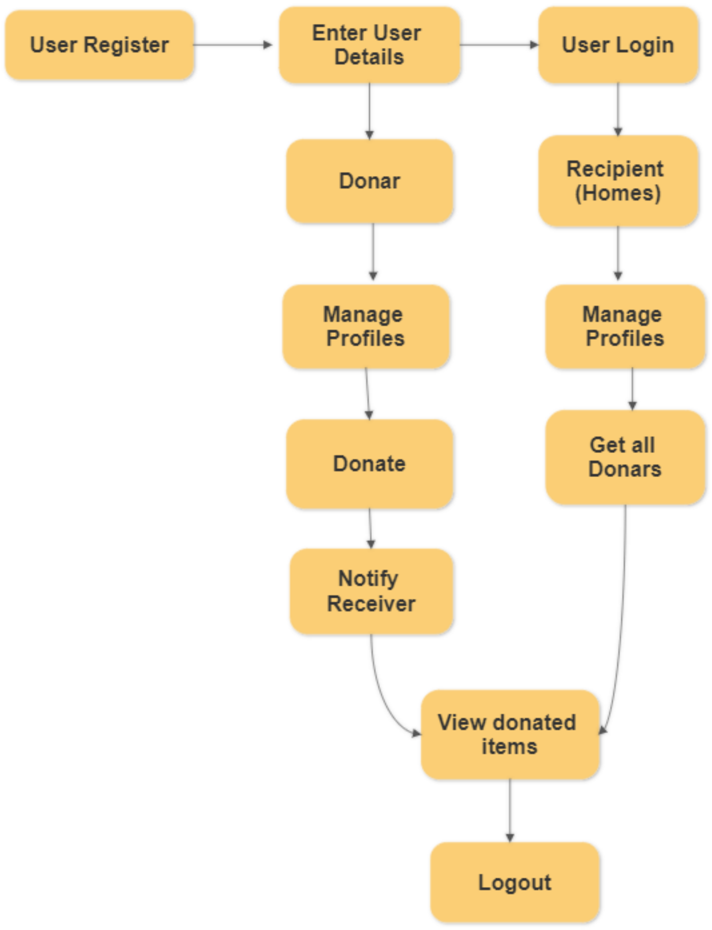


Figure 3.1: Architecture Diagram of Gather & Give

3.2 DESIGN COMPONENTS

3.2.1 Front End:

The Gather & Give uses angular.js (HTML,CSS,TS) for developing interactive pages.

3.2.2 Back End:

Uses Node.js and Mongodb for back end to store data.

3.3 DATABASE DESCRIPTION

Listed below gives a description of database document schemas used for Gather & Give.

3.3.1 Donor Structure

As shown in table 3.1, donor structure contains the details of the donors.

Table 3.1: Donor Description

Attribute Name	Type	Constraint(s)	Description
Donar Name	String	Minimum of 8 characters. May have [A-Z], [a-z],dot, hyphen or underscore. Should not start with dot or hyphen or underscore.	Name of the donor
Address	String	No Constraint	Address of the donor
Email-id	String	Must follow Email format.	Mail id of the donor
Password	String	Minimum of 8 chars. At least one lower case letter, one upper case letter andone number.	Password of the donor used for login details
Contact	Integer	Minimum of 10 Characters	Contact of the donor
Count	Integer	No Constraint	Represents the number of donations done by the donar

todonate	Array	No Constraint	Todonate contains food, cloth, money and others which have values true or false set to it.

3.3.2 Receiver Structure

As shown in table 3.2, Receiver structure contains the details of the donors.

Table 3.2: Receiver Description

Attribute Name	Type	Constraint(s)	Description
NGO Name	String	Minimum of 8 characters. May have [A-Z], [a-z],dot, hyphen or underscore. Should not start with dot or hyphen or underscore.	Name of the Receiver
Address	String	No Constraint	Address of the Receiver
Email-id	String	Must follow Email format.	Mail id of the Receiver
Password	String	Minimum of 8 chars. At least one lower case letter, one upper case letter andone number.	Password of the Receiver used for login details
Contact	Integer	Minimum of 10 Characters	Contact of the Receiver
NGO URL	String	No Constraint	URL of the Receiver

3.4 LOW LEVEL DESIGN

The following section illustrates the functionalities of the system. This includes login to the application, donating the food.

3.4.1 Registration

Table 3.3 shows the Registration details of the application.

Table 3.3 Registration Details

Files used	dsignup.component.html, dsignup.component.css, dsignup.conponent.spec.ts and dsignup.component.ts. rsignup.component.html, rsignup.component.css, rsignup.conponent.spec.ts and rsignup.component.ts.
Short Description	Allows the Donars and Receivers to Register to the application by providing Email Id.
Arguments	Username, Email Id, Password and Confirm Password.
Return	Success/Failure in Registration.
Pre-Condition	The User must have some interest to donate.
Post-Condition	Routed to the Login Page.
Exception	Invalid Username, Invalid Email Id and Invalid Password.
Actor	Donars and Receiver

3.4.2 Login

Table 3.4 shows the login details of the application.

Table 3.4 Login Details

Files used	dsignin.component.html, dsignin.component.css, dsignin.conponent.spec.ts and dsignin.component.ts. rsignin.component.html,rsignin.component.css, rsignin.conponent.spec.ts and rsignin.component.ts.
Short Description	Allows the Donars and Receivers to login to the application by providing Email Id and Password.
Arguments	Email Id, Password
Return	Success/Failure in login
Pre-Condition	The user must have an account
Post-Condition	The home page will be displayed
Exception	Invalid Email Id and password

Actor	Donors, Receiver

3.4.3 Donate

Table 3.5 shows the donation details of the application.

Table 3.5 Donation Details

Files used	donor1.component.html, donor1.component.css, donor1.conponent.spec.ts and donor1.component.ts.
Short Description	Allows the Donars to donate by selecting the items to donate
Arguments	Food, Cloth, Money, Others in Checkboxes
Return	Success/Failure to Donate
Pre-Condition	The user must have an account
Post-Condition	Notification send to the receiver
Exception	Invalid Email Id and password
Actor	Donors

3.4.4 Receive

Table 3.6 shows the receiving details of the application.

Table 3.6 Receiving Details

Files used	receiver1.component.html,receiver1.component.css, receiver1.conponent.spec.ts and receiver1.component.ts.
Short Description	Allows the Receivers to receive the notification from donors.
Arguments	Email Id, Password
Return	Success/Failure in login
Pre-Condition	The user must have an account
Post-Condition	Email Regarding donation
Exception	Invalid Email Id and password
Actor	Receiver

3.5 USER INTERFACE DESIGN

3.5.1 Home Page

Figure 3.2 provide the user interface for main activity.

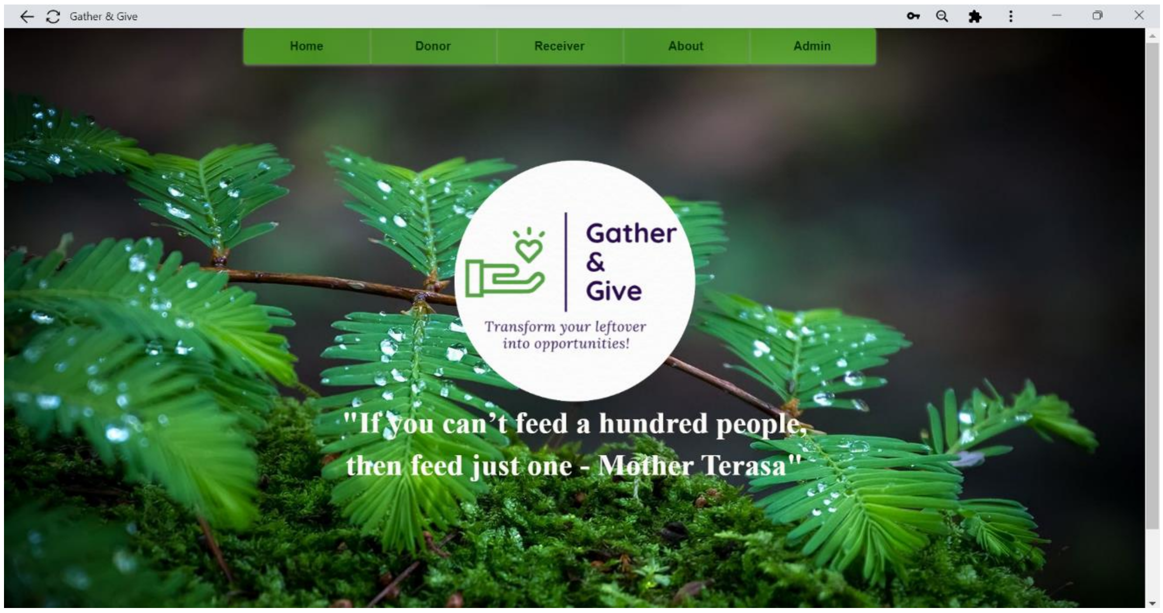
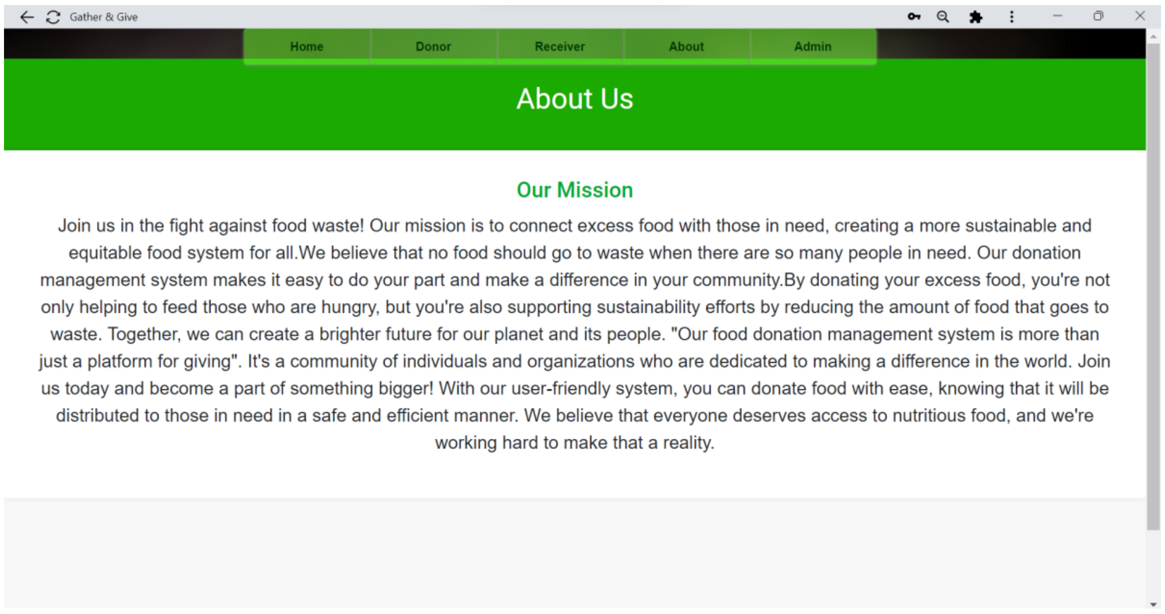

Gather & Give
Home
Donor
Receiver
About
Admin
Gather & Give
Transform your leftover into opportunities!
"If you can't feed a hundred people, then feed just one - Mother Terasa"

Figure 3.2: Home Page of Gather & Give

3.5.2 About Page

Figure 3.3 provide the about page of Gather & Give.



About Us

Our Mission

Join us in the fight against food waste! Our mission is to connect excess food with those in need, creating a more sustainable and equitable food system for all.We believe that no food should go to waste when there are so many people in need. Our donation management system makes it easy to do your part and make a difference in your community.By donating your excess food, you're not only helping to feed those who are hungry, but you're also supporting sustainability efforts by reducing the amount of food that goes to waste. Together, we can create a brighter future for our planet and its people. "Our food donation management system is more than just a platform for giving". It's a community of individuals and organizations who are dedicated to making a difference in the world. Join us today and become a part of something bigger! With our user-friendly system, you can donate food with ease, knowing that it will be distributed to those in need in a safe and efficient manner. We believe that everyone deserves access to nutritious food, and we're working hard to make that a reality.

Figure 3.3: About Page of Gather & Give

3.5.2 Admin Page

Figure 3.4, 3.5, 3.6 provide the admin page of Gather & Give which includes login, get all donors and get all receivers page.

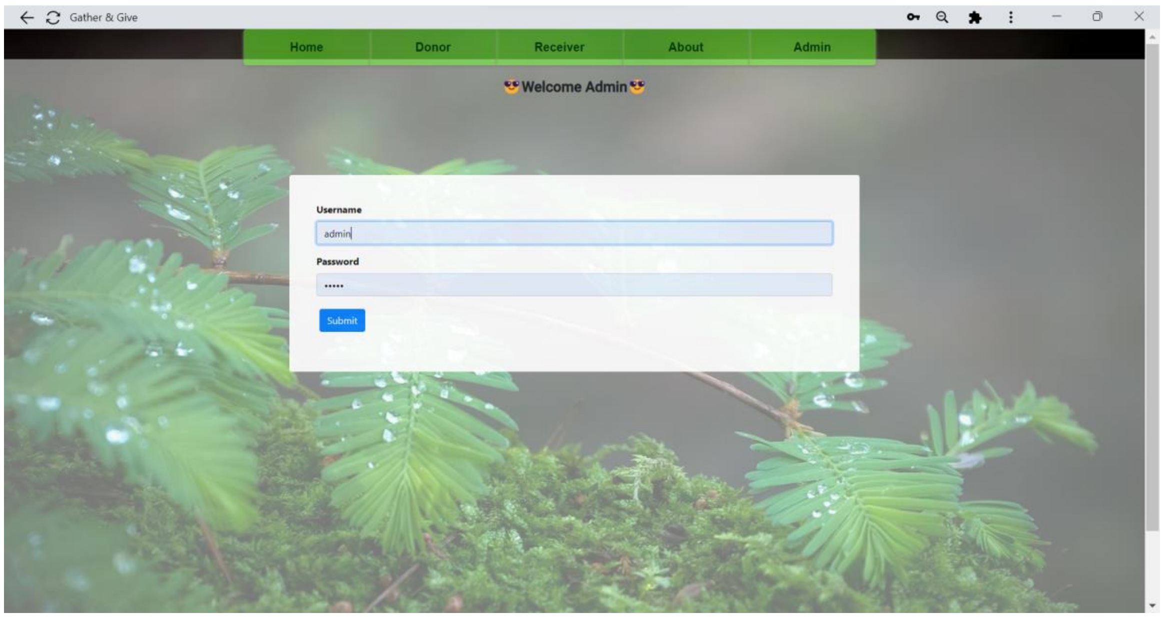
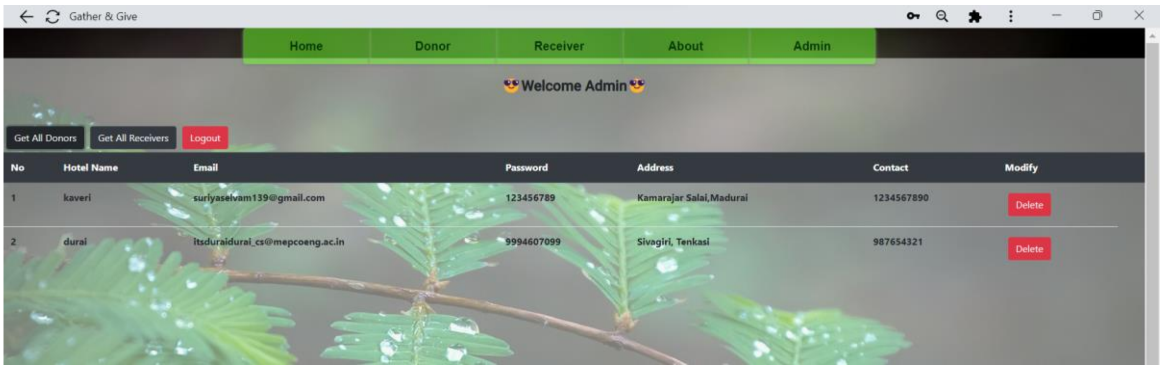
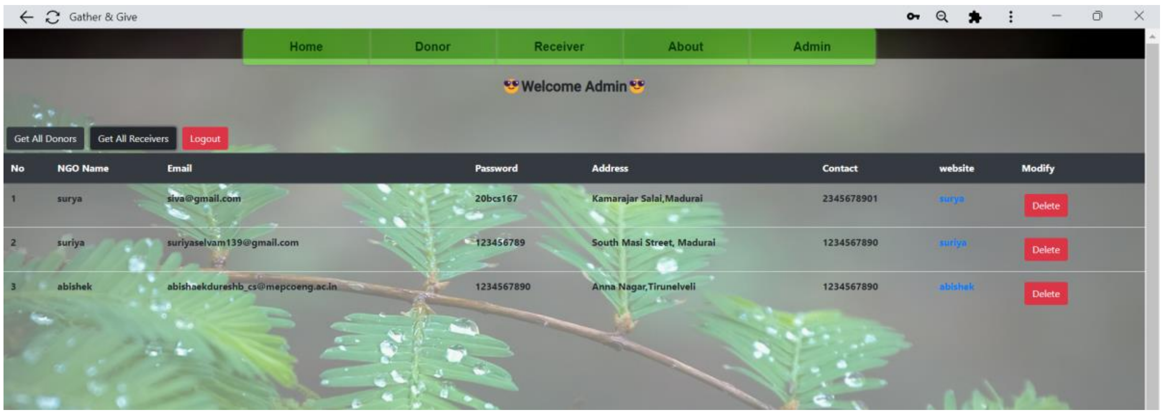


Figure 3.4: Admin Page of Gather & Give



No	Hotel Name	Email	Password	Address	Contact	Modify
1	kaveri	suriyaselvam139@gmail.com	123456789	Kamarajar Salai,Madurai	1234567890	Delete
2	durai	itsduraidurai_cs@mepcoeng.ac.in	9994607099	Sivagiri, Tenkasi	987654321	Delete

Figure 3.5: Get All Donors Page of Gather & Give

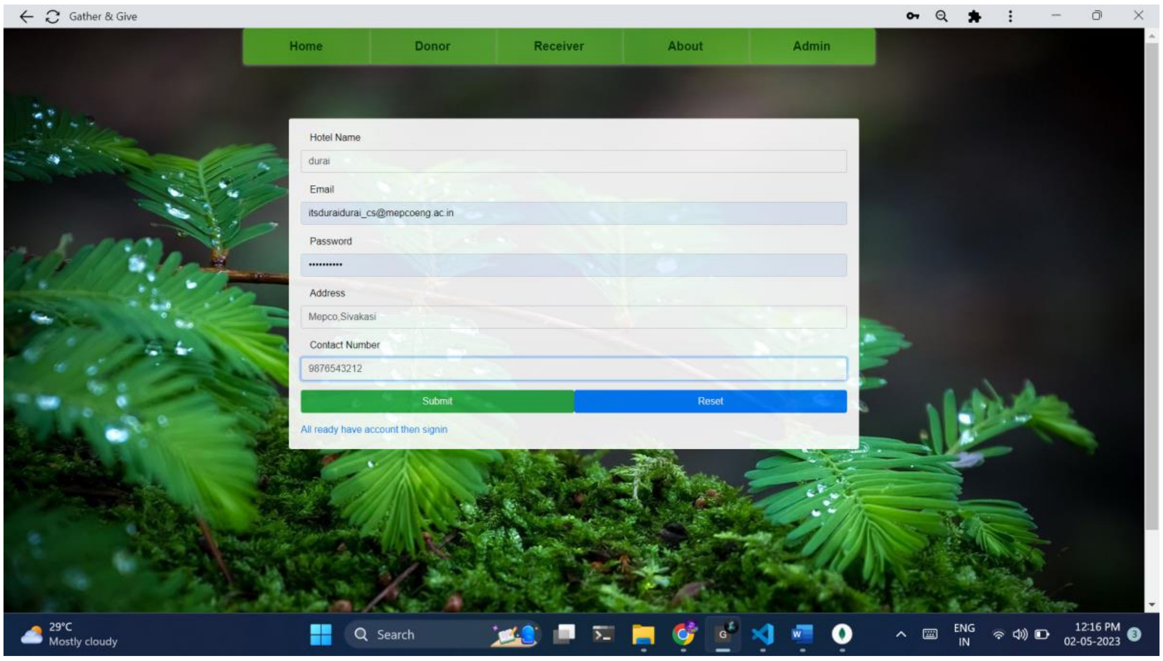


No	NGO Name	Email	Password	Address	Contact	website	Modify
1	surya	siva@gmail.com	20bcs167	Kamarajar Salai,Madurai	2345678901	surya	Delete
2	suriya	suriyaselvam139@gmail.com	123456789	South Masi Street, Madurai	1234567890	suriya	Delete
3	abishek	abishaekdureshb_cs@mepcoeng.ac.in	1234567890	Anna Nagar,Tirunelveli	1234567890	abishek	Delete

Figure 3.6: Get All Receivers Page of Gather & Give

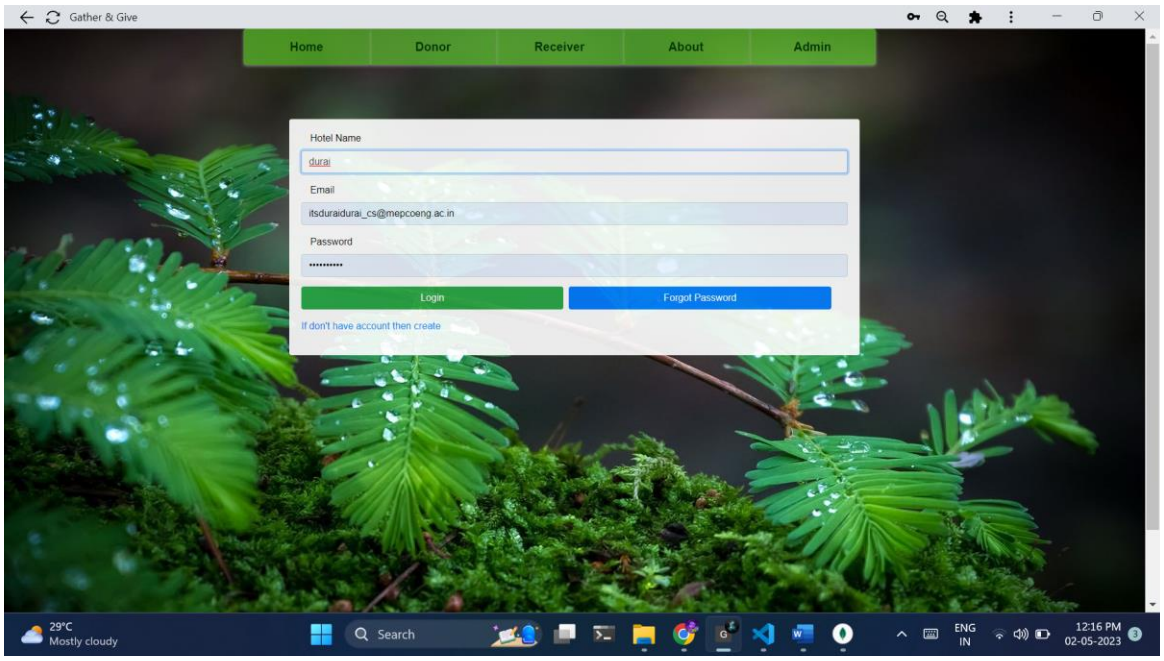
3.5.3 Donor Page

Figure 3.7, 3.8, 3.9, 3.10 provide the donor page of Gather & Give which includes register, login, select items page and donation page.



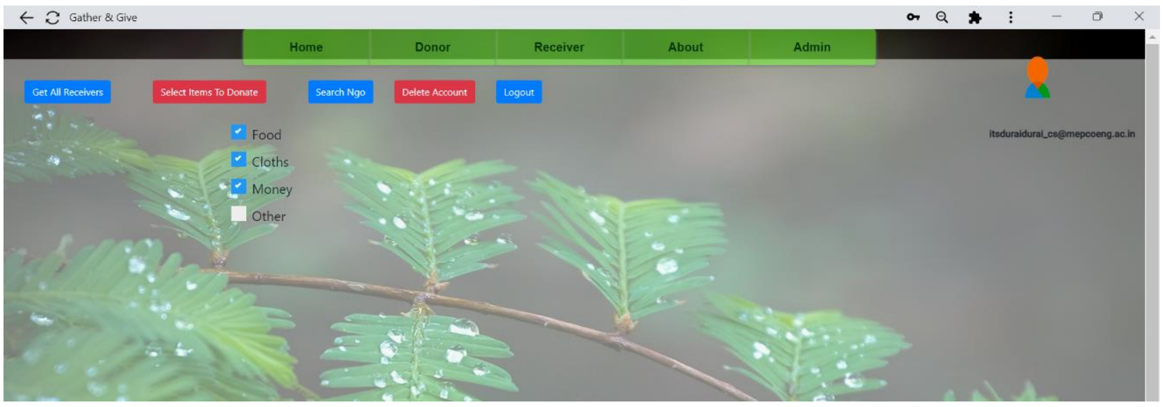
Hotel Name
durai
Email
itsduraidurai_cs@mepcoeng.ac.in
Password
••••••••••
Address
Mepco,Sivakasi
Contact Number
9876543212

Figure 3.7: Registration Page of Gather & Give for Donors



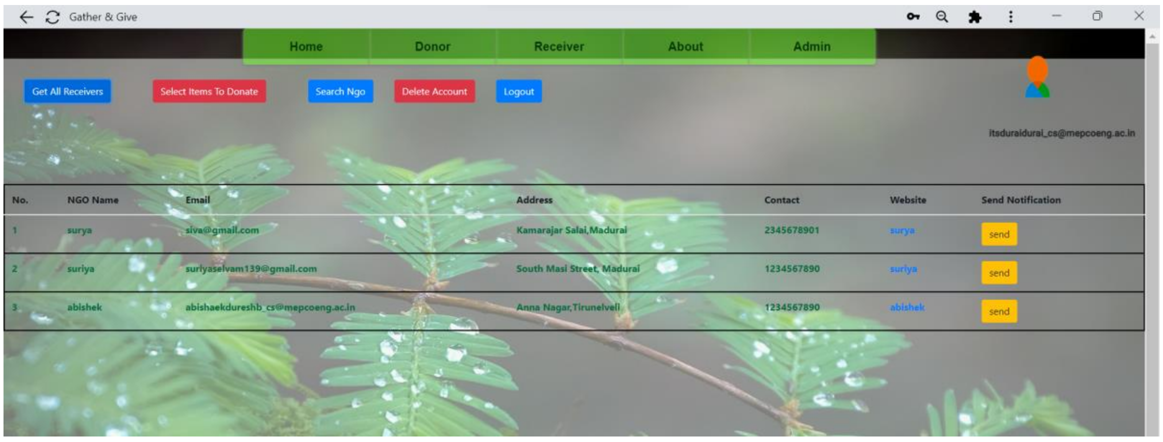
Hotel Name
durai
Email
itsduraidurai_cs@mepcoeng.ac.in
Password
••••••••••

Figure 3.8: Login page of Gather & Give for Donors



- [✓] Food
- [✓] Cloths
- [✓] Money
- [ ] Other

Figure 3.9: Select Items Page of Gather & Give for Donors

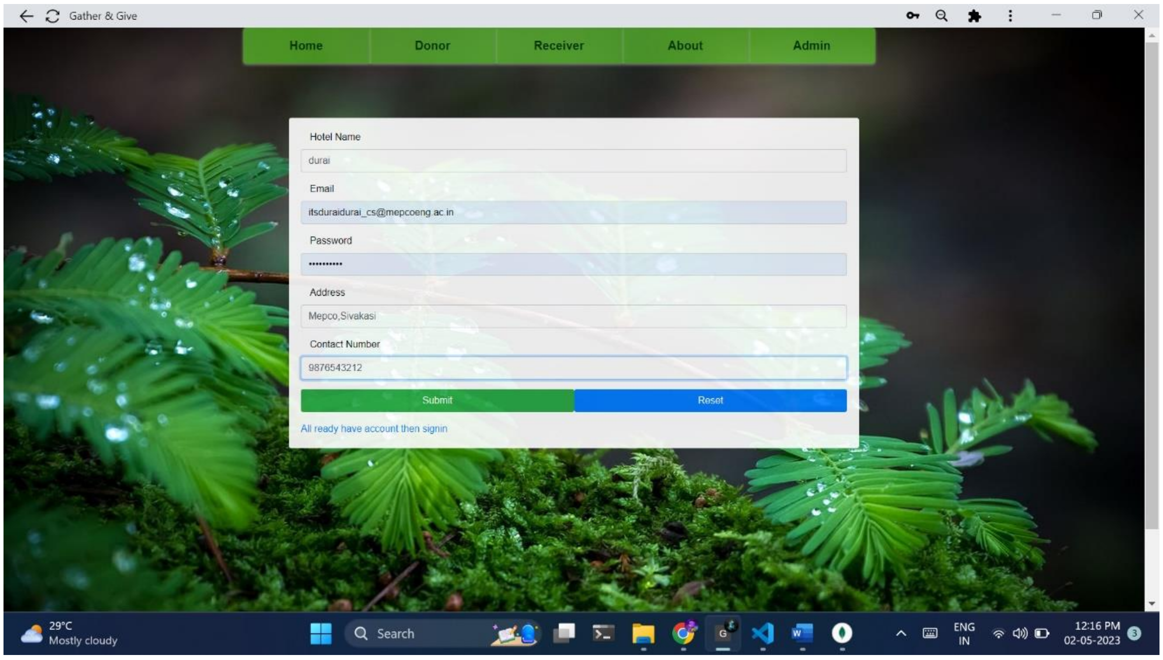


No.	NGO Name	Email	Address	Contact	Website	Send Notification
1	surya	siva@gmail.com	Kamarajar Salai,Madurai	2345678901	surya	send
2	suriya	suriyaselvam139@gmail.com	South Masi Street, Madurai	1234567890	suriya	send
3	abishek	abishaekdureshb_cs@mepcoeng.ac.in	Anna Nagar,Tirunelveli	1234567890	abishek	send

Figure 3.10: Donation Send Page of Gather & Give

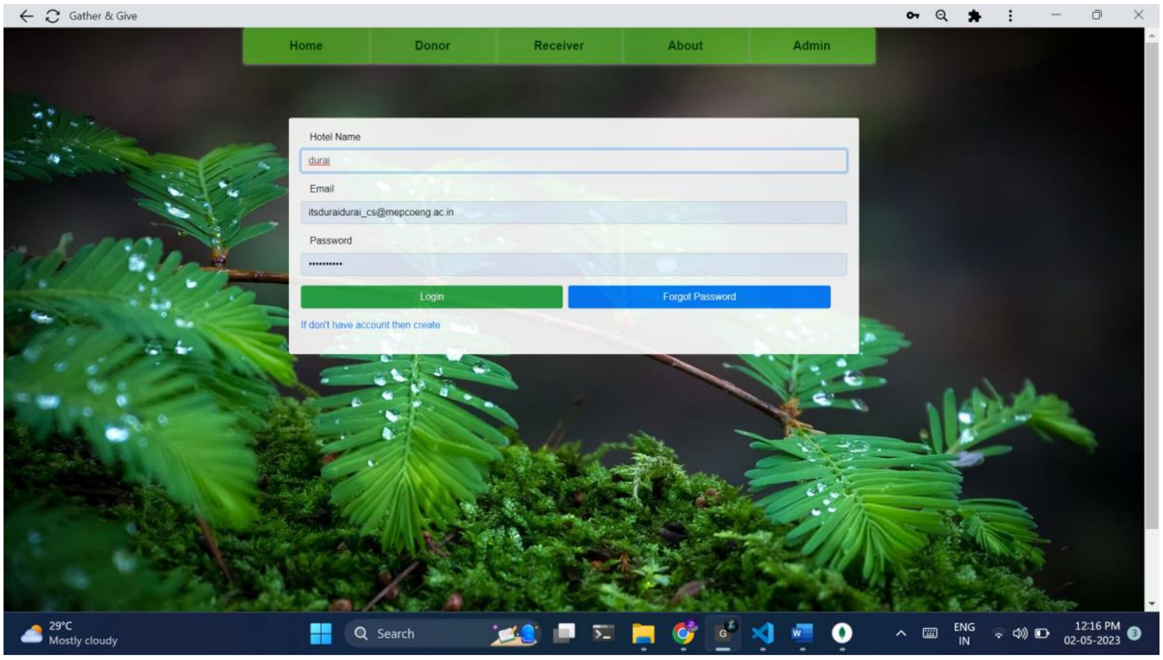
3.5.3 Receiver Page

Figure 3.11, 3.12, 3.13, 3.14, 3.15 provide the Receiver page of Gather & Give which includes register, login, get all donors page, get top donors page and email received by receiver.



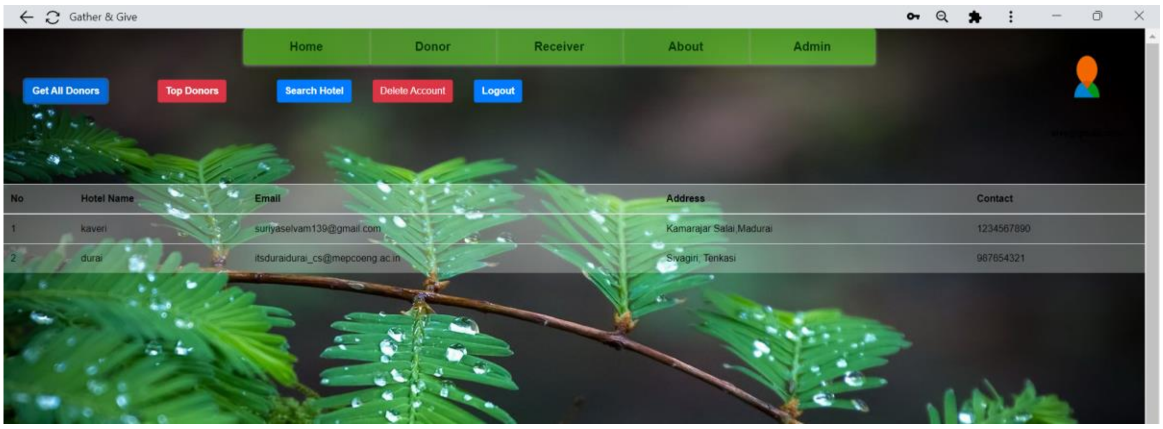
Hotel Name
durai
Email
itsduraidurai_cs@mepcoeng.ac.in
Password
••••••••••
Address
Mepco,Sivakasi
Contact Number
9876543212

Figure 3.11: Receiver Registration Page of Gather & Give



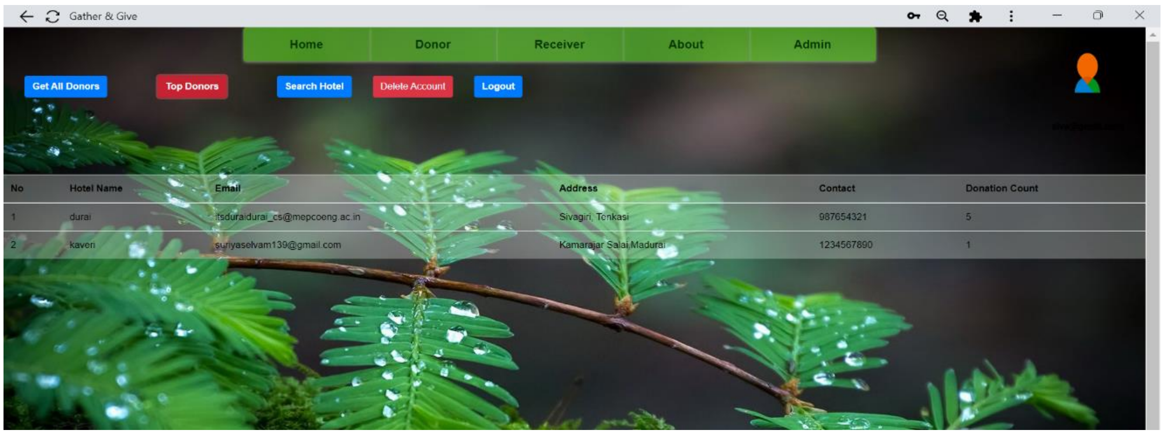
Hotel Name
durai
Email
itsduraidurai_cs@mepcoeng.ac.in
Password
••••••••••

Figure 3.12: Receiver Login Page of Gather & Give



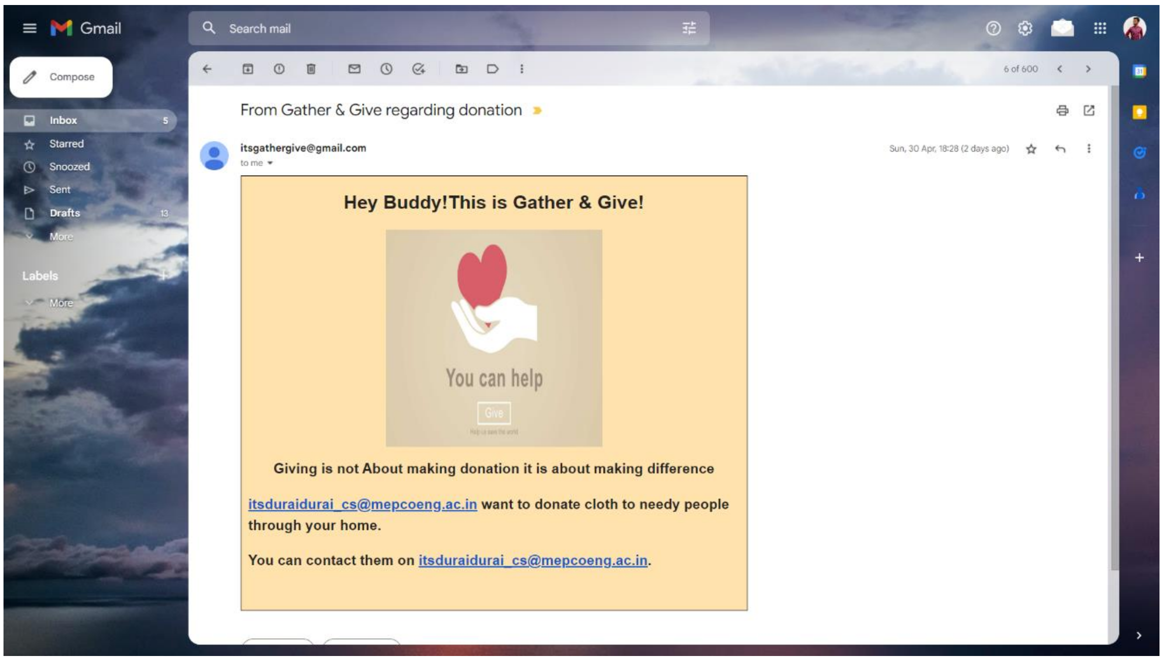
No	Hotel Name	Email	Address	Contact
1	kaveri	suriyaselvam139@gmail.com	Kamarajar Salai,Madurai	1234567890
2	durai	itsduraidurai_cs@mepcoeng.ac.in	Sivagiri, Tenkasi	987654321

Figure 3.13: Get all donors of Gather & Give in Receiver Page



No	Hotel Name	Email	Address	Contact	Donation Count
1	durai	itsduraidurai_cs@mepcoeng.ac.in	Sivagiri, Tenkasi	987654321	5
2	kaveri	suriyaselvam139@gmail.com	Kamarajar Salai,Madurai	1234567890	1

Figure 3.14: Get Top donors of Gather & Give in Receiver Page



From Gather & Give regarding donation

itsgathergive@gmail.com
to me
Sun, 30 Apr, 18:28 (2 days ago)

Hey Buddy!This is Gather & Give!

Giving is not About making donation it is about making difference

itsduraidurai_cs@mepcoeng.ac.in want to donate cloth to needy people through your home.

You can contact them on itsduraidurai_cs@mepcoeng.ac.in.

Figure 3.15: Email Notification of Gather & Give to Receiver

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 LOGIN IMPLEMENTATION

The login credentials are obtained. If the credentials are OK, then the user is redirected to the homepage

```
GET EmailID, Password
   IF EmailID, Password valid
          RETURN homepage
   ELSE
          TOAST Invalid Credential
```

4.2 SIGNUP IMPLEMENTATION

The form fields are obtained. If they are valid, then the user is added to the database.

```
GET requestFields
   IF requestFields valid
          RETURN added to database
   ELSE
          TOAST enter valid details
```

4.3 DONATION IMPLEMENTATION

The form fields are obtained. If they are valid, then the receiverdetails are listed.

```
GET donatingitems
IF donationitems valid
          RETURN notification to the receiver
ELSE
          TOAST not selected donation items
```

4.4 DISPLAY IMPLEMENTATION

The form fields are obtained. If they are valid, then the receiver details or donor details are listed.

```
IF button click valid
        RETURN corresponding information to the corresponding page
ELSE
        TOAST no data to be displayed
```

4.5 LOGOUT IMPLEMENTATION

The Logout Button is clicked then it will redirect to home page.

```
IF button click valid
        RETURN Redirect to Home Page
ELSE
        TOAST nothing
```

CHAPTER 5

RESULTS AND DISCUSSION

5.1 TEST CASES AND RESULTS

5.1.1 Test Cases and Results for Login function:

The Table 5.1 shows that the possible test data for the test case given below, if the user is already having account then the output is true otherwise false.

Test Id	User Interface /Method	Input Given	Expected Output	Actual Output	Testing Status
TC001a	Login form	- Your e-mail: <name@domain.com> -Password:<empty>	Error message showing “Password is required”	Error message showing “Password is required”	Pass
TC001b	Login form	- Your email:<empty> -Password: xxxx	Error message showing “email is required”	Error message showing “email is required”	Pass
TC001c	Login form	- Your email:<empty> -Password:<empty>	Error message showing "email is required." Error message showing “Password is required"	Error message showing "email is required." Error message showing "Password is required"	Pass
TC001d	Login form	- Your e-mail: <invalid_user_name> -Password:xxxx	Error message showing "Enter valid email."	Error message showing "Enter valid email ."	Pass
TC001e	Login form	- Your e-mail: <name@domain.com> -Password: xxxx	Redirected to user's dashboard page	Redirected to user's dashboard page	Pass

Table 5.1: Test Case and result for Login

5.1.2 Test Cases and Results for Registration function:

The Table 5.2 shows that the possible test data for the test case given below, if the user is already having account then the output is false otherwise true.

Test Id	User Interface /Method	Input Given	Expected Output	Actual Output	Testing Status
TC002a	Register form	- Your e-mail: <name@domain.com> -Password: xxx	Redirected to successful sign-up page	Redirected to successful sign-up page	Pass
TC002b	Register form	- Your e-mail: <empty> -Password: xxx	Error message showing "E-mail is required"	Error message showing "E-mail is required"	Pass
TC002c	Register form	-Your e-mail: <name@domain.com> -Password: xxxx -Confirm Password: yyyy	Error message showing "Passwords do not match"	Error message showing "Passwords do not match"	Pass

Table 5.2: Test Case and result for Registration

5.1.3 Test Cases and Results for Donation function:

The Table 5.3 shows that the possible test data for the test case given below, if the user is already having account then the user can able to donate using donate function.

Test Id	User Interface /Method	Input Given	Expected Output	Actual Output	Testing Status
TC003a	Donation form	- Select Items to donate: <empty>	Error message showing “select item to donate.”	Error message showing “select item to donate.”	Pass
TC003b	Donation form	- Select Items to donate: Food Cloth	Notification send successfully to NGO	Notification send successfully to NGO	Pass

Table 5.3: Test Case and result for Donate items

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT(S)

In conclusion, Gather & Give system is an effective way to reduce food waste and ensure that excess food is distributed to those in need. This system can help address the problem of food insecurity and alleviate hunger while also promoting sustainability and reducing environmental impact. Such a system can be implemented through a variety of approaches, such as partnering with local food banks, shelters, and community organizations.

Overall, a Gather & Give system can have numerous benefits, including reducing food waste, addressing food insecurity, promoting sustainability, and fostering community engagement. As such, it is a valuable initiative that should be encouraged and supported by individuals, businesses, and governments alike.

In future can be able to display the route to the location of the receiver in order to reduce the time of taking the food to the destination. This application really helps the people to donate the remaining food in parties or restaurants to the orphanage or old age homes.

APPENDIX – A

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENT:

Processor: A quad-core processor with a clock speed of 2.5 GHz or higher.

RAM: 8 GB of RAM is the minimum requirement for developing Angular applications.

Hard Drive: A solid-state drive (SSD) is used for faster read/write speeds and improved performance. However, a standard hard disk drive (HDD) with at least 7200 RPM can also be used.

SOFTWARE REQUIREMENT:

Operating System :	Linux, Windows 10,11
DBMS :	MongoDB Compass
IDE used :	Visual Studio
Angular Version:	10 and above
Node Version :	12 and above

APPENDIX – B

SOURCE CODE

login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl,ReactiveFormsModule, FormBuilder,
Form, Validators/**for validatio  purpose */ } from '@angular/forms'
import { DonorService } from '../donor.service';//taking donor service
import {Router} from '@angular/router'
@Component({
  selector: 'app-dsignin',
  templateUrl: './dsignin.component.html',
  styleUrls: ['./dsignin.component.css']
})
export class DsigninComponent implements OnInit {
  success: String;
  display:boolean=false;
  recoverpassdisplay:boolean=false;//for display reset form
  dsigninform:FormGroup//to signig form
  recoverform:FormGroup;//to reset password
  hname:String;
  email:string;
  password:String;

  ngOnInit(): void {

    sessionStorage.setItem('status', 'signin');
  }
```

```
 constructor(private fb:FormBuilder,public
donorservice:DonorService,private router:Router) {
   //singin form
   this.dsigninform=fb.group({
      hname :['', Validators.required],
      email :['',Validators.compose([Validators.required,Validators.email])],

password:['',Validators.compose([Validators.required,Validators.minLength(
6)])]
     })


    //password recover form
    this.recoverform=fb.group({
      email: ['', Validators.compose([Validators.required, Validators.email])],

password:['',Validators.compose([Validators.required,Validators.minLength(
6)])],
      confirmpassword: ['', Validators.compose([Validators.required,
Validators.minLength(6)])]


     })


   }
 //go to donor1 page
   gotoDonor1()
   {


    this.router.navigate(['donor1']);
   }


 checkData(dsigninform: any) {
  this.display = true;
  console.log('making display true')
  this.donorservice.checkDonor(dsigninform.value).subscribe((res) => {
    if (res == true)
    {
      this.email=this.dsigninform.get('email').value
      //email set to local storage
      localStorage.setItem('email',this.email);
      this.gotoDonor1();

      this.success = "Login successfully";
      console.log('sign success');

    }
    else
    {
      this.success = "Incorrect email or password";
      console.log('signin unseccessufull')

    }
  })
  console.log('making display false')
  this.display = false;
}gotosignup()
{
  this.router.navigate(['dsignup']);
}

recoverPass()
```

```
  {
    this.recoverpassdisplay=true;
}
  //to update password
  updatePass(recoverform:any)
  {
    console.log(this.recoverform)
    if (this.recoverform.controls['password'].value !=
this.recoverform.controls['confirmpassword'].value)
    {
      alert(' Your both passwords must match')
      this.recoverform.controls['password'].setValue('');
      this.recoverform.controls['confirmpassword'].setValue('');
    }
    else
    {
      this.donorservice.recoverPass(recoverform.value).subscribe((res)=>{
        if(res==true)
        {
          console.log('updated');
          this.recoverpassdisplay = false;
          //to set again all values to blank
          this.recoverform.controls.email.setValue('');
          this.recoverform.controls.password.setValue('');
          this.recoverform.controls.confirmpassword.setValue('');

        }
        else
          console.log('email not exist');
    })
   } }}
```

signup.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, FormBuilder, Form, Validators/**for
validatio  purpose */ } from '@angular/forms'
import { DonorService } from '../donor.service';//taking donor service
import { Router } from '@angular/router'

@Component({
  selector: 'app-dsignup',
  templateUrl: './dsignup.component.html',
  styleUrls: ['./dsignup.component.css']
})
export class DsignupComponent implements OnInit {

  validation: string;
  dsignupform: FormGroup;



  constructor(private fb: FormBuilder, public donorservice: DonorService,
public route: Router) {
    this.dsignupform = fb.group({
      hname: ['', Validators.required],
      email: ['', Validators.compose([Validators.required,
Validators.pattern("^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$"),
Validators.email])],
      password: ['', Validators.compose([Validators.required,
Validators.minLength(6)])],
      address: ['', Validators.required],
      contact: ['',
Validators.compose([Validators.required,Validators.minLength(10),
Validators.maxLength(10), Validators.pattern("^((\\+91-?)|0)?[0-
9]{10}$")])]
    })
  }

  ngOnInit(): void {
    sessionStorage.setItem('status', 'signup');
  }

  //submit data
  submitData(dsignupform: any) {

    this.donorservice.postDonor(dsignupform.value).subscribe((res) => {
      if (res == true)//data found
      {
        this.validation = "Data All ready exist";
        window.alert('Your Entered data allready exist please signin');
      }
      else {
        console.log('donor signup success')
        this.validation = "Sucessfully inserted";
        this.resetform();
        this.route.navigate(['dsignin'])
      }

    });
  }
```

```
 get f() {
   return this.dsignupform.controls;
 }
 resetform() {
   this.dsignupform.reset();
 }

 gotosignin()//if all ready having account go to signin
 {
   this.route.navigate(['dsignin'])
 }}
```

donor.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Receiverdata } from 'Models/receiverdata.model';
import { ReceiverService } from 'src/app/receiver.service';
import { Donordata } from 'Models/donordata.model';
import {DonorService } from 'src/app/donor.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-donor1',
  templateUrl: './donor1.component.html',
  styleUrls: ['./donor1.component.css']
})
export class Donor1Component implements OnInit {
  donoremail:string;

  plotbar:boolean=false;

  showreceivers:boolean=false;//to display all receivers

  donate:boolean=false;//to select items to donate

  showlocation:boolean=false;//to search ngo

  showdelete:boolean=false;

  // isUserdonated=false;//

  receiverdata: Receiverdata;

  receiverarray: Receiverdata[];

  food:boolean=false;

  cloth:boolean=false;

  money:boolean=false;

  other:boolean=false;

  d:Donordata;


  email:string

  constructor(public receiver: ReceiverService, public donorservice: DonorService, private router: Router) { }


  ngOnInit(): void {

    sessionStorage.setItem('status', 'on_page');

    //taking email from local storage and set in session storage

    sessionStorage.setItem('email',localStorage.getItem('email'));
    this.email=sessionStorage.getItem('email');


  }


  getallreceivers() {
    if (!this.showreceivers)//if false
    {
      console.log(this.email);
      //first set othe false so it will not display other componets
      this.showlocation=false;
      this.donate=false;
      this.showreceivers = true;
      this.receiver.getReceivers().subscribe((res) => {
        this.receiverarray = res as Receiverdata[];
      });
         this.showreceivers=true;
    }
    else {
      this.showreceivers = false;
    }}
displayDonateItems()
  {
    if(!this.donate)//if not display then display
    {
      console.log(this.email);
      //first set othe false so it will not display other componets
      this.showlocation = false;
      this.showreceivers= false;
      //set true and  display this component
      this.donate = true;
    }
    else
      this.donate=false;
  }
 donateToThisNgo(receiver:Receiverdata)
  {
    //  this.donoremail = sessionStorage.getItem('email');
    this.d = new
Donordata(this.email,receiver.email,this.food,this.cloth,this.money,this.other
);
if(this.food ||  this.cloth || this.money || this.other)//if selected
    {
      this.donorservice.sendEmail(this.d).subscribe((res) => {
        if (res == true) {
          window.alert('Notification send to NGO ' + receiver.ngoname);
          //After donation all items set to false again
          this.food=false;
          this.cloth=false;
          this.money=false;
          this.other=false;
 }
        else
          console.log('error in sending notification');
      })
      sessionStorage.setItem('id','yes');
    }
    else
      window.alert('first select Items to donate');
  }
//for map
  searchLocation() {
    if (!this.showlocation)
    {
      //first set othe false so it will not display other componets
      this.donate = false;
      this.showreceivers = false;
      //set true and  display this component
      this.showlocation = true;
    }
    else
      this.showlocation = false;
  }

  //to delete account
  deleteAaccount()
  {
    if(window.confirm('Are you sure want to delete'))
    {
    var donordata=new Donordata(this.email,'',true,true,true,true);
      this.donorservice.deleteByPermission(donordata).subscribe((res)=>{
          if(res==true)
          {
            // sessionStorage.clear();
            sessionStorage.setItem('logout', 'yes')
            this.router.navigate(['home']);

          }
          else
            console.log('error in delete donor inside particulat donor')
      })}
    else
      console.log('not');
      }
```

```
//for Logout
  Logout()
  {
      sessionStorage.setItem('logout','yes')
      this.router.navigate(['home'])
  }}
```

receiver.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Donordata } from 'Models/donordata.model';
import { DonorService } from 'src/app/donor.service';
import {ReceiverService} from 'src/app/receiver.service';
import { Router } from '@angular/router';
import { Receiverdata } from 'Models/receiverdata.model';
@Component({
  selector: 'app-receiver1',
  templateUrl: './receiver1.component.html',
  styleUrls: ['./receiver1.component.css']
})
export class Receiver1Component implements OnInit {

  showdonors: Boolean = false;
  showtopdonors:Boolean=false;
  showlocation: boolean = false;
  donorarray: Donordata[];

  topdonorarray:Donordata[];

  email:string;

  constructor(public donor: DonorService, private router: Router,private
recerver:ReceiverService) { }


  ngOnInit(): void {

    sessionStorage.setItem('status', 'on_page');

   //taking email from local storage and set in session storage

    sessionStorage.setItem('email', localStorage.getItem('email'));

 this.email = sessionStorage.getItem('email');


  }

getalldonors() {

    if (!this.showdonors)//if false

    {

      //making other false

      this.showlocation=false;

      this.showtopdonors=false;

    this.showdonors = true;//set true and display

      this.donor.getDonors().subscribe((res) => {

        console.log(res);

        this.donorarray = res as Donordata[];

      })
    }
    else if (this.showdonors)//if display then not display
    {
      this.showdonors = false;
    } }
  getIndescendingOrder()
  {
    if(!this.showtopdonors)
    {
      //making other false
      this.showlocation = false;
      this.showdonors = false;
     this.showtopdonors=true;
      this.donor.getDonors().subscribe((res) => {
        this.topdonorarray=res as Donordata[];
        // console.log(this.topdonorarray);
        this.topdonorarray.sort((a,b)=>(a.count < b.count) ? 1 : -1 );
      })
    }
    else if(this.showtopdonors)
    {
      this.showtopdonors=false;
    }
```

```
 }
 searchLocation() {
   if(!this.showlocation)
   {
     //making other false
     this.showdonors = false;
     this.showtopdonors = false;
     this.showlocation = true;
   }
     else
     this.showlocation=false;
 }
deleteAaccount() {
   if (window.confirm('Are you sure want to delete')) {
    var receiver = new Receiverdata(this.email);
     this.recerver.deleteByPermission(receiver).subscribe((res) => {
       if (res == true) {
         sessionStorage.setItem('logout', 'yes')
         this.router.navigate(['home']);
       }
       else
         console.log('error in delete donor inside particulat donor')
```

```
      });
}
    else
      console.log('not');
 }
 //for Logout
  Logout() {
    sessionStorage.setItem('logout', 'yes')
    this.router.navigate(['home'])
  }}
```

admin.component.ts

```
import { Component, OnInit } from '@angular/core';
import  {DonorService} from '../donor.service'
import {ReceiverService} from '../receiver.service'
import { Donordata } from 'Models/donordata.model';
import { Receiverdata } from 'Models/receiverdata.model';
import { Router } from '@angular/router';
import { FormGroup, FormControl, FormBuilder, Form, Validators/**for
validation  purpose */ } from '@angular/forms'
@Component({
  selector: 'app-admin',
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.css']
})
```

```
export class AdminComponent implements OnInit {
  pageTitle:String="😎Admin😎"
  adminform:FormGroup
  id: Number;
  donordata: Donordata;
  receiverdata: Receiverdata;
  hname: Donordata["hname"];
  email: string
  showdonors: Boolean = false;
  showreceivers: Boolean = false;
  valid: boolean = false;
  donorarray: Donordata[];
  receiverarray: Receiverdata[];
  ngOnInit(): void {
    sessionStorage.setItem('status','admin');
 }
  constructor(public donor: DonorService, private router: Router,public
receiver:ReceiverService,private fb:FormBuilder) {
    this.adminform = fb.group({
      adminname: ['', Validators.required],
      adminpassword: ['',Validators.required]
    })
   }
submit(f:any)
```

```
{
 var username=this.adminform.get('adminname').value;
 var pass=this.adminform.get('adminpassword').value;
 if(username=="admin" && pass=="admin")
 {
   this.valid = true;
   sessionStorage.setItem('status', 'on_page');
 }
 else
 {
   this.valid = false;
   window.alert('Please check user name and password');
 }}
getalldonors()
 {
     if(!this.showdonors)//if false
     {
       this.showreceivers=false;
       this.showdonors=true;//set true and display
         this.donor.getDonors().subscribe((res)=>{
         console.log(res);
         this.donorarray=res as Donordata[];
       })
     }
     else if(this.showdonors)//if display then not display
     {
       this.showdonors=false;
     }    }
getallreceivers()
 {
   if(!this.showreceivers)//if false
   {
     this.showdonors=false;
     this.showreceivers=true;//set true and display
     this.receiver.getReceivers().subscribe((res) => {
       this.receiverarray = res as Receiverdata[];
     });
   }
   else
   {
     this.showreceivers=false;
   }}
 deleteDonor(donordata)
 {
   console.log(donordata);
   this.donor.deleteDonor(donordata).subscribe((res)=>{
     if(res==true)
     {
       console.log('deleted');
       this.showdonors=!this.showdonors;
       this.getalldonors();
     }
     else
       console.log('not deleted')

   })
 }
 deleteReceiver(receiverdata) {
   console.log(receiverdata);
   this.receiver.deleteReceiver(receiverdata).subscribe((res) => {
     if (res == true) {
       console.log('deleted');
       this.showreceivers=!this.showreceivers;
       this.getallreceivers();
     }
     else
       console.log('not deleted')
   })
 }
```

```
 //for Logout
  Logout() {
    sessionStorage.setItem('logout', 'yes')
    this.router.navigate(['home'])
}}
```

app.component.ts

```
import { Component, OnInit} from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  title = '';
  constructor() { }
  ngOnInit() {
    sessionStorage.clear();
    sessionStorage.setItem('status','app-component');
}}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { Routes, RouterModule, Router } from '@angular/router';
import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { DsigninComponent } from './dsignin/dsignin.component';

import { DsignupComponent } from './dsignup/dsignup.component';

import { RsignupComponent } from './rsignup/rsignup.component';

import { RsigninComponent } from './rsignin/rsignin.component';

//for reactive form module

import { FormsModule,ReactiveFormsModule } from '@angular/forms';

//both below for  serveices

import { DonorService } from './donor.service';

import { ReceiverService } from './receiver.service';

import { AdminComponent } from './admin/admin.component';

//for http request response

import {HttpClientModule} from '@angular/common/http';

//for google map

import { AgmCoreModule } from '@agm/core';

import { Receiver1Component } from
'./Receiver/receiver1/receiver1.component';

//for session managment

import { NgxWebstorageModule } from 'ngx-webstorage';

import { Donor1Component } from './Donor/donor1/donor1.component';

// import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';

import { MatCheckboxModule } from '@angular/material/checkbox';
import { Donor1Guard } from './donor1.guard';

import { ThankyouComponent } from './thankyou/thankyou.component';

import { HomeComponent } from './home/home.component';

import { AboutComponent } from './about/about.component';

import { DeactiveGuard } from './deactive.guard';

import { FooterComponent } from './footer/footer.component';

import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';

import { MapsComponent } from './maps/maps.component';

import { LogoutGuard } from './logout.guard';

@NgModule({

  declarations: [

    AppComponent,

    DsigninComponent,

    DsignupComponent,

    RsignupComponent,

    RsigninComponent,

    AdminComponent,

    Receiver1Component,

    Donor1Component,

    ThankyouComponent,

    HomeComponent,

    AboutComponent,
    FooterComponent,
    MapsComponent ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
    RouterModule,
    BrowserAnimationsModule,
    MatCheckboxModule,
    AgmCoreModule.forRoot({
      apiKey: '',
      libraries: ['places']
    }),
],
  providers: [DonorService, ReceiverService, Donor1Guard,
DeactiveGuard,LogoutGuard],//registering services
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { DsigninComponent } from './dsignin/dsignin.component';

import { DsignupComponent } from './dsignup/dsignup.component';

import { RsigninComponent } from './rsignin/rsignin.component';

import { RsignupComponent } from './rsignup/rsignup.component';

import { AdminComponent } from './admin/admin.component';

import { Receiver1Component } from
'./Receiver/receiver1/receiver1.component';

import { Donor1Component } from './Donor/donor1/donor1.component';

import { Donor1Guard } from './donor1.guard';

import { ThankyouComponent } from './thankyou/thankyou.component';

 import { HomeComponent } from './home/home.component';

import { AboutComponent } from './about/about.component';

import { DeactiveGuard } from './deactive.guard';

import { LogoutGuard } from './logout.guard';

const routes: Routes = [

  {path:'',component:HomeComponent},

  { path: 'admin', component: AdminComponent, canActivate:
[Donor1Guard,LogoutGuard]},

  { path: 'home', component: HomeComponent, canActivate:
[Donor1Guard,LogoutGuard]},

  { path: 'about', component: AboutComponent, canActivate:
[Donor1Guard,LogoutGuard]},

  { path: 'dsignin', component: DsigninComponent, canActivate:
[Donor1Guard,LogoutGuard]},

  { path: 'dsignup', component: DsignupComponent, canActivate:
[Donor1Guard,LogoutGuard]},
  { path: 'rsignin', component: RsigninComponent, canActivate: 
[Donor1Guard,LogoutGuard]},

  { path: 'rsignup', component: RsignupComponent, canActivate: 
[Donor1Guard,LogoutGuard]},

  { path: 'donor1', component: Donor1Component, canActivate: 
[Donor1Guard,LogoutGuard]},

  { path: 'receiver1', component: Receiver1Component, canActivate: 
[Donor1Guard,LogoutGuard]},

  { path:'thankyou',component:ThankyouComponent}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }
```

app.component.html

```
<div class="menu">

<nav class="nav">

        <ul>

                <li><a [routerLink]="['home']">Home</a></li>

                <li><a [routerLink]="['home']">Donor</a>

                        <ul>

                                <li><a [routerLink]="['dsignup']">Sign 
Up</a></li>
```

```
                    <li><a [routerLink]="['dsignin']">Sign In</a></li>


               </ul>
          </li>
          <li><a [routerLink]="['home']">Receiver</a>
               <ul>
                    <li><a [routerLink]="['rsignup']">Sign
Up</a></li>
                    <li><a [routerLink]="['rsignin']">Sign In</a></li>


               </ul>
          </li>
          <li><a [routerLink]="['about']">About</a></li>
          <li><a [routerLink]="['admin']">Admin</a></li>
     </ul>
</nav>
</div>
```

REFERENCES

1. https://angular.io/tutorial

2. https://www.w3schools.com/angular/default.asp

3. https://www.tutorialspoint.com/nodejs/index.html

4. P.J. Deitel, H.M. Deitel, “Internet and World Wide Web – How to program”, Fifth Edition, Pearson Education Publishers, 2009

5. Amol Nayak, “MongoDB Cookbook Paperback” , November 2014

6. Krasimir Tsonev, “Node.js by Example Paperback”, May 2015

7. Nate Murray, Felipe Coury, Ari Lerner and Carlos Taborda, “ng-book, The Complete Book on Angular 4” September 2016

8. Jeffrey C. Jackson, "Web Technologies - A Computer Science Perspective", Pearson Education, 2011

9. David Herron, “Node.js Web Development: Create real-time server-side applications with this practical, step-by-step guide”, 3rd Edition, 2016

10. Agus Kurniawan, “AngularJS Programming by Example”, First Edition, Kindle, 2014