PROJECT REPORT

ON

“HIBUY”

Submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications

UTTARANCHAL UNIVERSITY

Uttaranchal School of Computing Sciences (USCS)

SESSION 2022-23

Under the guidance of:

Mr. Amarjeet Rawat

Assistant Professor

USCS

Submitted by:

Rai Chatterjee

Enroll. No.UU180900384

Batch: 2021-23

ACKNOWLEDGEMENT

The far more awaited moment is progress, yet nothing can be achieved if done alone. The success is the results of many people's dedication and continual support and we applaud everybody who helped us successfully carry out this project.

I would really want to appreciate (Dr.) Sonal Sharma, Dean of Uttaranchal School of Computing Sciences (USCS), for fostering a healthy and encouraging learning environment, in particular. I am grateful to Mr. Amarjeet Rawat, Assistant Professor, USCS and my project mentor. I would also want to convey my heartfelt appreciation to them for their advice and supervision, along with for supplying vital report information and assistance in the report's completion.

It was very generous for me to accept my candidacy as the most beneficial council for improving my morals in all respects. Furthermore, I would like to thank my parents, colleagues, and well-wishers for always being there for me and encouraged me throughout the endeavor.

Rai Chatterjee

Enroll No: UU180900384

DECLARATION

I hereby declare that Rai Chatterjee submitted a project report titled "HIBUY" to the Uttaranchal School of Computing Sciences (USCS). Mr. Amarjeet Rawat, Assistant Professor at the Uttaranchal School of Computing Sciences (USCS), oversaw the project.

I also declare that the work described in this report was not submitted in whole or in part to the aforementioned University or a different institute for the awarding of another degree or a certificate.

Rai Chatterjee

Enroll No: UU180900384

CERTIFICATE OF ORIGINALITY

This is to certify Rai Chatterjee student of MCA 4th Semester, of **Uttaranchal University, Dehradun**, has completed the project **HIBUY** using the technologies HTML5, CSS3, Bootstrap 5, Angular 13, Typescript, Node JS, Express JS, and MongoDB for the Batch (2021-2023).

Under the guidance of:

Mr. Amarjeet Rawat

Uttaranchal University

Dehradun

LIST OF FIGURE

1.	Fig No. 1	Gantt Chart	8
2.	Fig No. 2	Pert Chart	9
3.	Fig No. 3	Use Case Diagram	14
4.	Fig No. 4	Entity Relation Model	15
5.	Fig No. 5	Login Page before clicking the signing button	16
6.	Fig No. 6	On successful login	17
7.	Fig No. 7	No Listing Found Page	17
8.	Fig No. 8	Login Page	18
9.	Fig No. 9	Home Page	18
10.	Fig No. 10	Listing’s Page	19
11.	Fig No. 11	Request Page	19
12.	Fig No. 12	Setting’s Page	20
13.	Fig No. 13	My Location Page	20

LIST OF TABLES

1.	Table No. 1	Software Used in the Project	**5**
2.	Table No. 2	Gantt chart	7
3.	Table No. 3	Functional Requirements Table	10
4.	Table No. 4	Non-Functional Requirements Table	11
5.	Table No. 5	Minimum Hardware Specification Table	12
6.	Table No. 6	Hardware Specification Table	12
7.	Table No. 7	Software Specification Table	13

TABLE OF CONTENTS

ACKNOWLEDGEMENT .... **I**

DECLARATION .... **II**

CERTIFICATE OF ORIGINALITY .... **III**

LIST OF FIGURES .... **IV**

LIST OF TABLES .... **V**

1. Introduction 1

2. Objectives 2

3. System Analysis 3

Problem Identification 3

Proposed Solution 3

3.1 Feasibility Study 4

3.1.1 Technical Feasibility 4

3.1.2 Economic Feasibility 5

3.1.3 Operational Feasibility 6

4. Project Planning and Scheduling 7

4.1 GANTT Chart 7

4.2 PERT Chart 9

4.3 Software Requirement Specifications (SRS) 10

4.4 Tools and Platform Used 12

4.5 Use Case Diagram 14

4.6 Entity Relational Model 15

5. Testing 16

6. Reports To Be Generated 18

7. System Security Measures 21

8. Cost Estimation Model 22

9. Future Scope Of The Project 24

10. Appendices 25

10.1 Coding 25

10.2 Bibliography 50

1. INTRODUCTION

For many of us, our pets are more than just animals. They are beloved members of our families who provide us with love, companionship, and joy. But finding the perfect pet can be a challenge[1], especially if you don't have a lot of time or money to search for pets for sale in your area. Fortunately, thanks to modern technology, finding a pet has never been easier. It all starts with a simple passion for our families and pets. For those of us who have a large[2], loving family[3], we understand the importance of looking out for one another. We also understand that our pets are just as much a part of our family as anyone else and should be treated as such[4].

When it comes to finding a new pet, one of the best places to start is the internet. There are countless websites and resources available for those who are looking to adopt a pet, including HIBUY. HIBUY is a great resource for finding local pet stores, shelters, and rescue groups. One of the best things about HIBUY is its user-friendly interface. To get started, all you need to do is create an account and enter your location. From there, you can browse through a variety of pets, including cats, rabbits, horses, birds, and more. HIBUY will automatically fetch and display pets that are available for adoption in your area, making it easy to find the perfect pet for your family.

When you find a pet that interests you, you can click on its listing to learn more. The listing will provide you with information about the pet, such as its age, breed, and whether or not it has been vaccinated. If you are interested in adopting the pet, you can click on the Request button to start the adoption process.

The great thing about HIBUY is that it connects pet owners with potential adopters. If the pet owner finds you to be a suitable candidate, they can approve your request, and you can connect with them to bring your new pet home. This not only makes the adoption process easier but also ensures that the pet is going to a loving and caring home.

Of course, adopting a pet is not a decision that should be taken lightly[5]. When bringing a new pet home, think about your lifestyle, budget, and capacity to offer a secure and caring atmosphere to your new friend. But with resources like HIBUY, finding the perfect pet for your family has never been easier. So why wait? Start your search for a new pet today and find your new best buddy!

2. OBJECTIVES

- To emphasize the value of treating pets as cherished family members and to inspire readers to give their new pet a secure and caring home.
- To encourage people to think about obtaining a pet instead of buying from pet shops or breeders by adopting from local shelters or rescue organizations.

3. SYSTEM ANALYSIS

System analysis is methods that accumulates and analyze information, identifies issues and breaks the system down into its elements. Systems analyses are carried out to examine a platform itself and components in order to establish its objectives. It is indeed a conundrum approach that strengthens the platform and ensures the efficiency of all components of the system to meet their objectives.

The first stage is to assess the initial idea to discover the needs of the project. We have studied a number of studies that are oriented to our goal. Detailed research has been done to identify the essential security flaw in the present system. Therefore, we continued with an original notion to create "HIBUY".

PROBLEM IDENTIFICATION

It is estimated that more than 1 million suitable dogs and cats are euthanized in India each year, due to the fact there are often excessive pets in shelters & too not enough individuals choose adopting when seeking a companion. Assuming greater numbers of individuals adopted pets rather than purchasing them, the amount of animals euthanized may be drastically decreased[6]. Fostering a dog or cat rescues a cherished pet by welcoming them to the family while also making shelter accommodation accessible to another animal in distress[7].

HIBUY may be able to help with these issues. It is an excellent resource for connecting with pet adoptions in your neighborhood. Either you want to adopt from a shelter or connect with a breeder or rescue group, you can find it on the internet.

PROPOSED SOLUTION

HIBUY is a free app for pet sharing. We adore camaraderie and wish to share the joy that our pets bring us, therefore we designed a pet-first application!

Pets are carriers of goodness. They have the capacity to unite humanity. HIBUY was created to foster a caring environment that will have an impact back on society and effect change. We feel that by sharing animal tales with this community, we can support animal rescue organizations and drive change. The aim is that these animals may find a forever home, or that those who already have a home would give to worthy charities.

One of the best things about HIBUY is its user-friendly interface. To get started, all you need to do is create an account and enter your location. From there, you can browse through a variety of pets, including cats, rabbits, horses, birds, and more. HIBUY will automatically fetch and display pets that are available for adoption in your area, making it easy to find the perfect pet for your family.

When you find a pet that interests you, you can click on its listing to learn more. The listing will provide you with information about the pet, such as its age, breed, and whether or not it has been vaccinated. If you are interested in adopting the pet, you can click on the Request button to start the adoption process. The great thing about HIBUY is that it connects pet owners with potential adopters. If the pet owner finds you to be a suitable candidate, they can approve your request, and you can connect with them to bring your new pet home. This not only makes the adoption process easier but also ensures that the pet is going to a loving and caring home. Of course, adopting a pet is not a decision that should be taken lightly. When bringing a new pet home, think about your lifestyle, budget, and capacity to offer a secure and caring atmosphere to your new friend. But with resources like HIBUY, finding the perfect pet for your family has never been easier.

3.1 FEASIBILTY STUDY

The feasibility assessment comprises considering all feasible methods in which the problem can be resolved. The suggested solution should meet all user needs and be flexible enough to make it easy to make future adjustments based on future requirements.

3.1.1 TECHNICAL FEASIBILITY

Technical feasibility examines the component known and projects necessary for the system development to check interoperability of the methodology developed and to see that the essential technical team is available to create the system. The proposed project name “HIBUY” is a web application. The main technology and tools associated are:

Operating System	**Windows 10**
Frontend Technologies	HTML5, CSS3, Bootstrap 5, Typescript, JavaScript, Angular 13
Backend Technologies	NodeJs, ExpressJs
API's	ImgBB (Free image hosting service), Google OAuth 2.0 (For login), Google Mapbox
Database	MongoDB
Platform	Visual Studio Code 1.55.2

Table No 1: Software Used in the Project

3.1.2 ECONOMIC FEASIBILITY

In economic feasibility, a cost-benefit analysis is performed to analyze expected expenses as well as advantages. An economic evaluation is performed to determine the cost-effectiveness of the system that is suggested. One of most significant thing is fundamental method in economic feasibility. The planned application's expense would be determined solely by the budget incurred for hardware specifications. The platform specifications can be easily met at no expense, including the installation of the free Visual Studio Code 1.55.2.

Hardware Cost

- Standalone PC: Minimum Rs- 15000/-

3.1.3 OPERATION FESIBILITY

Operational feasibility is a way of measuring how effectively the system presented addresses the problems and how everything meets the challenges outlined mostly during requirement analysis process and during the system development testing process.

We evaluated the proposed system's operational component to manual operation. We discovered that the suggested approach is more advantageous from an operational standpoint. HIBUY is a place where people post advertisements online. It is an online classified site to post pets for sale advertisements online.

Each website offers an exceptional service for those homeless pets in need, and you can also sell your pet on these websites.

Once you have found pets you are interested in or want to sell the one, take the next step and contact the owner. This website will guide you and help you in buying and selling pets.

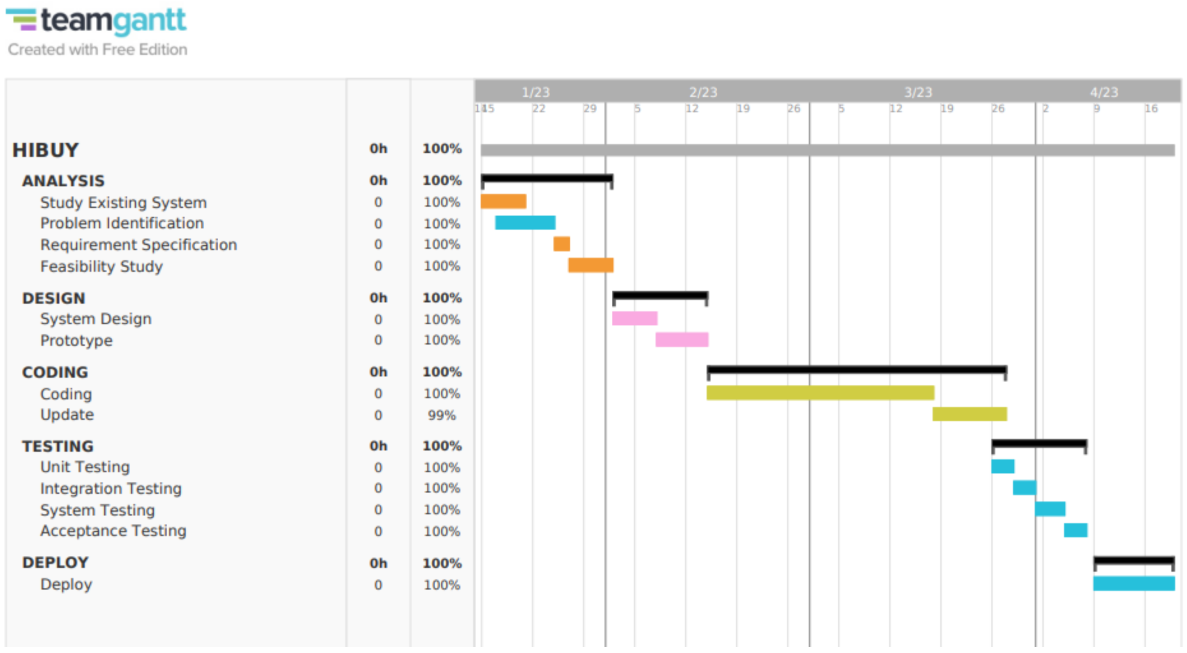
4. PROJECT PLANNING AND SCHEDULING

The project planning is unlikely to rely on deliverables solely, the start/finish dates of projects and the development constraints. A 'project plan' is therefore a complete document containing project objectives, scope, costs, risks and timetable. An estimated date and subsequent work packages will be implemented in the project timeline.

4.1 GANTT CHART

HIBUY			
TASK	START DATE	END DATE	DURATION
Task 1			
ANALYSIS			
1.1 Study Existing System	15-01-2023	20-01-2023	5
1.2 Problem Identification	17-01-2023	24-01-2023	7
1.3 Requirement Specification	25-01-2023	27-01-2023	2
1.4 Feasibility Study	27-01-2023	01-02-2023	3
Task 2			
DESIGN			
2.1 System Design	02-02-2023	07-02-2023	5
2.2 Protoype	08-02-2023	14-02-2023	6
Task 3			
CODING			
3.1 Coding	15-02-2023	16-03-2023	33
3.2 Update	16-03-2023	25-03-2023	9
Task 4			
TESTING			
4.1 Unit Testing	26-03.2023	28-03-2023	2
4.2 Integration Testing	29-03-2023	01-04-2023	2
4.3 System Testing	02-04-2023	05-04-2023	3
4.4 Acceptane Testing	06-04-2023	08-04-2023	2
Task 5			
Deploy	09-04-2023	19-04-2023	10
	TOTAL DAYS		90

Table No 2: Gantt chart



HIBUY	**0h**	**100%**
ANALYSIS	**0h**	**100%**
Study Existing System	0	100%
Problem Identification	0	100%
Requirement Specification	0	100%
Feasibility Study	0	100%
DESIGN	**0h**	**100%**
System Design	0	100%
Prototype	0	100%
CODING	**0h**	**100%**
Coding	0	100%
Update	0	99%
TESTING	**0h**	**100%**
Unit Testing	0	100%
Integration Testing	0	100%
System Testing	0	100%
Acceptance Testing	0	100%
DEPLOY	**0h**	**100%**
Deploy	0	100%

Fig No 1: Gantt chart

4.2 PERT CHART

A PERT diagram is a diagram of the project plan. A PERT or project-program evaluation method represents the main objectives and timeframe of a project. In a project timeline, that affects the completion of the project, the PERT chart reflects in evitable modifications.

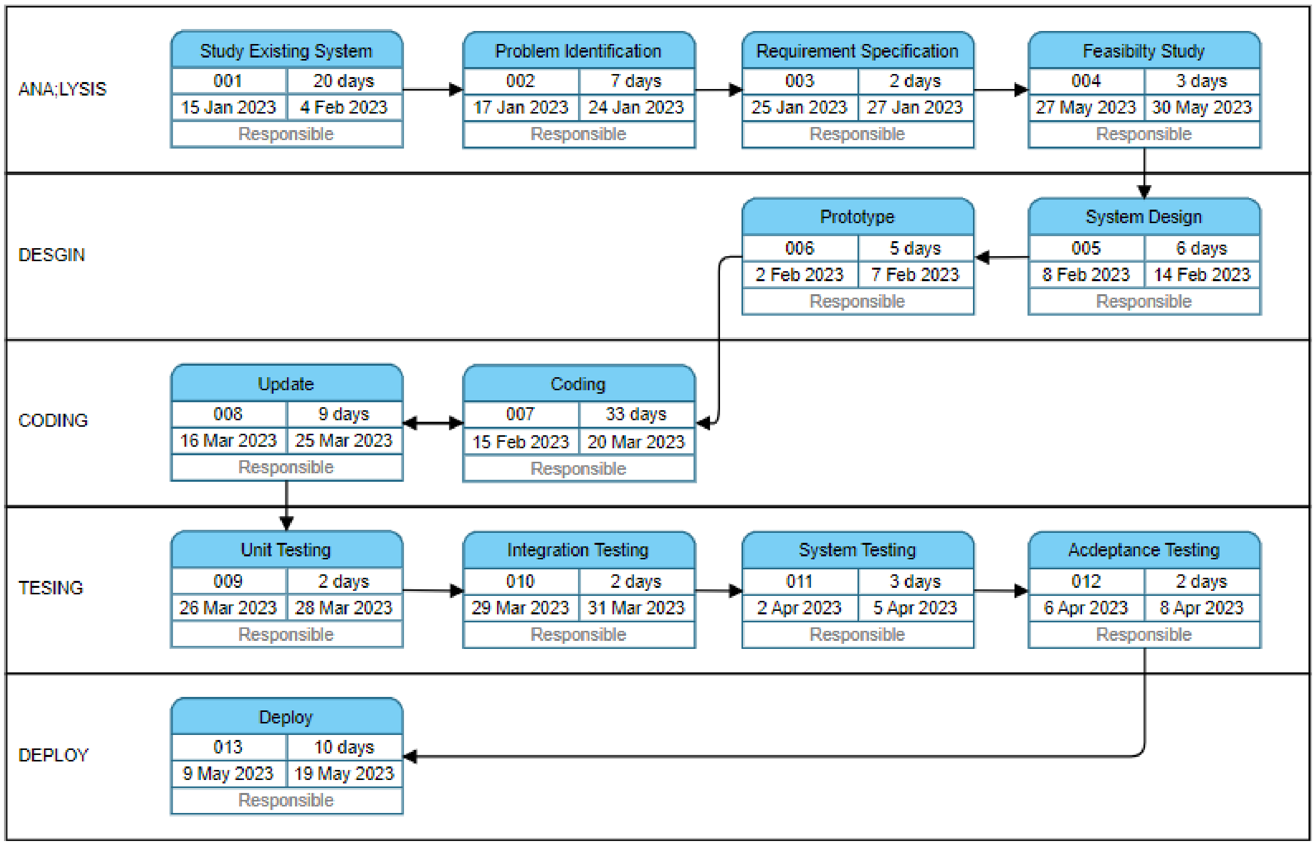


Fig No 2: Pert chart

4.3 SOFTWARE REQUIREMENT SPECIFICATION

FUNCATIONAL REQUIREMENT

The functional requirement is generally concerned with the application's design. So, the required user functional specifications for this project include:

ID	REQUIREMENT	DEPENDENCY	PRIORITY
F1	Create an account, and as well as login into it.	None	Very High
F2	Manage user profile details.	F1	High
F3	Create new pet listing as per there requirements and able to see in the all listing page.	None	Very High
F4	Able to manage listing request.	None	Very High
F5	Able to manage user added listings.	F3	Very High
F6	Manage user home location. Able to update his location too.	F1	Very High

Table No 3: Functional Requirements Table

NON- FUNCTIONAL REQUIREMENT

The non- functional requirement is generally which is not directly concerned with the application's design. So, the required user non-functional specifications for this project include:

ID	REQUIREMENT	DEPENDENCY	PRIORITY
NF1	Authenticate user using Google OAuth 2.0 (Sign in with Google) and save user details in MongoDB.	None	Very High
NF2	The entire user interface should be readable and simple so that it is easy interpretable to the end user.	None	Low
NF3	Make use of Google Mapbox API to select user home location.	NF1, NF4	Very High
NF4	Save user data to local storage, in order to avoid API call every time to fetch user details.	NF1	Very High
NF5	Make use of ImgBB (Free image hosting service) to make API request to upload profile picture while updating his profile details.	NF1, NF4	High

Table No 4: Non-Functional Requirements Table

4.4 TOOLS AND PLATFORM USED

The tools and platform used in the project used are:

MINIMUM HARDWARE SPECIFICATION

Processor	Intel Core i3
Processor Speed	250 MHz to 833 MHz
RAM	8GB
Hard Disk	500 GB

Table No 5: Minimum Hardware Specification Table

HARDWARE SPECIFICATION

Processor	Intel Core i3
Processor Speed	250 MHz to 933 MHz
RAM	8GB
Hard Disk	2TB

Table No 6: Hardware Specification Table

SOFTWARE SPECIFICATION

Operating System	Windows 10
Frontend Technologies	HTML5, CSS3, Bootstrap 5, Typescript, JavaScript, Angular 13
Backend Technologies	NodeJs, Express

API's	ImgBB (Free image hosting service), Google OAuth 2.0 (For login), Google Mapbox
Database	MongoDB
Platform	Visual Studio Code 1.55.2

Table No 7: Software Specification Table

4.5 USE CASE DIAGRAM

The primary type of system/software requirements for a new undeveloped software application is a UML use case diagram[8]. Utilization scenarios define the desired behavior (what) rather[9] than the specific way to execute it (how). Once specified, textual and graphic representation[10] (e.g., use case diagram) can be defined.

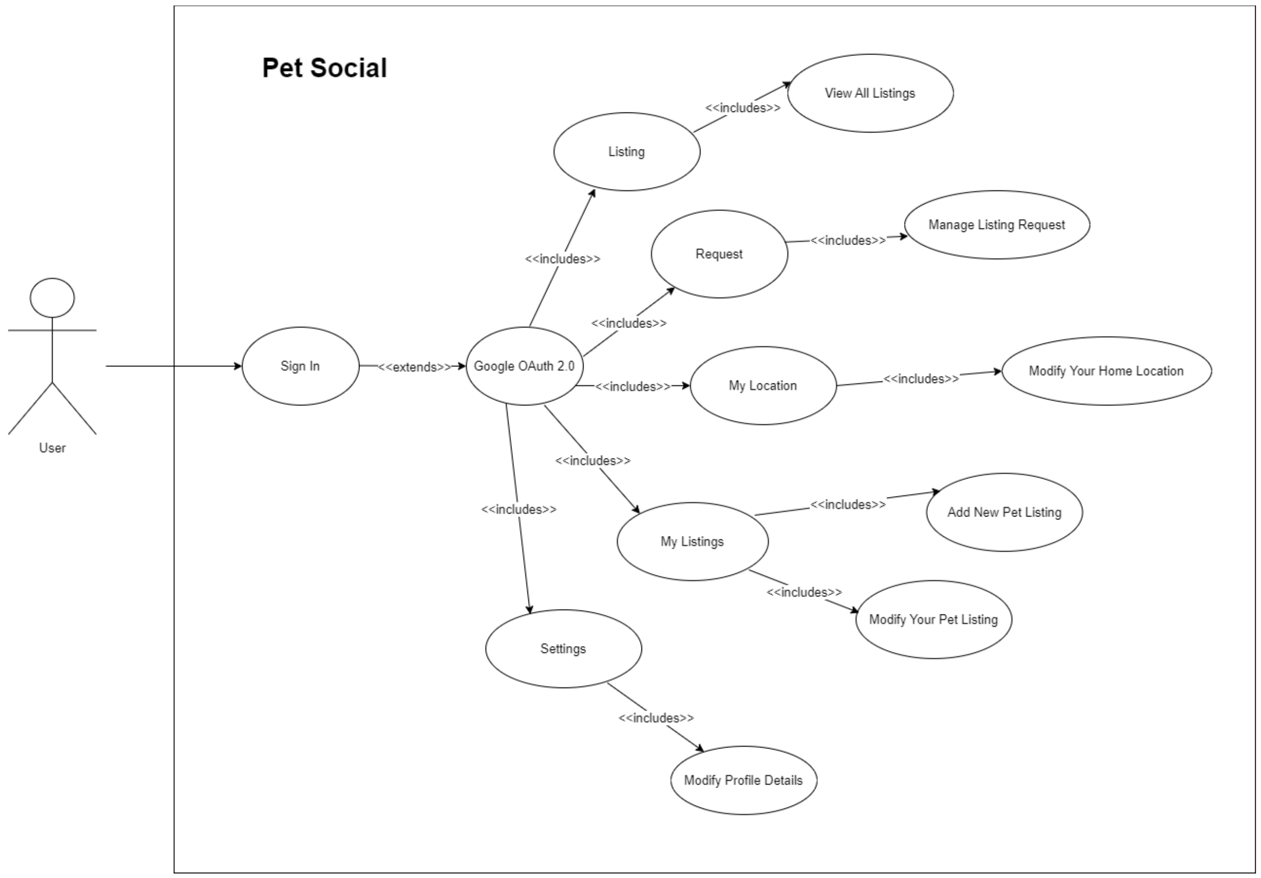


Fig. No 3: Use Case Diagram

4.6 ENTITY RELATION MODEL

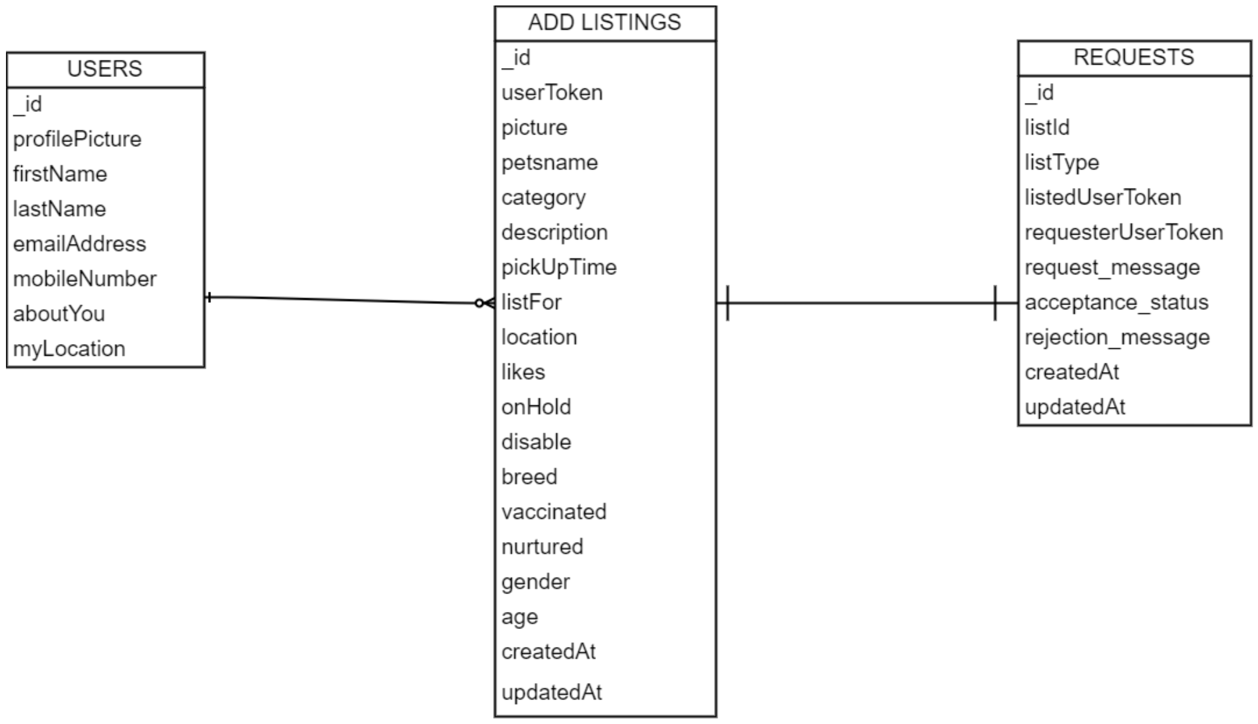


Fig. No 4: Entity Relation Model

5. TESTING

Testing[11] is a process to assess whether overall outcomes meet the predicted outcomes and if the software program is defect free[12]. It entails running an application or system component in order to evaluate one or more attributes of interest[13][14].

5.1 Test Report for On Login Success:

Test Case 1

The user needs to click on Sign in with Google button in order to authenticate and login to the system. It uses Google OAuth 2.0. On clicking the button, it connects with the Gmail account and returns the user's information. This information further is used to register the user, and login.

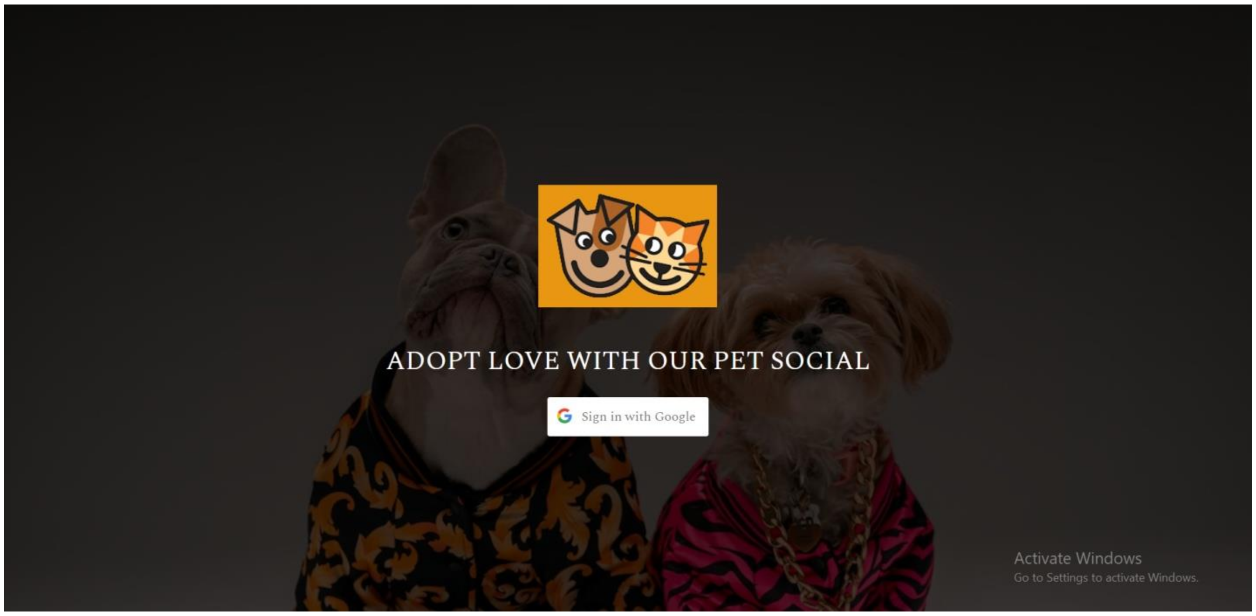
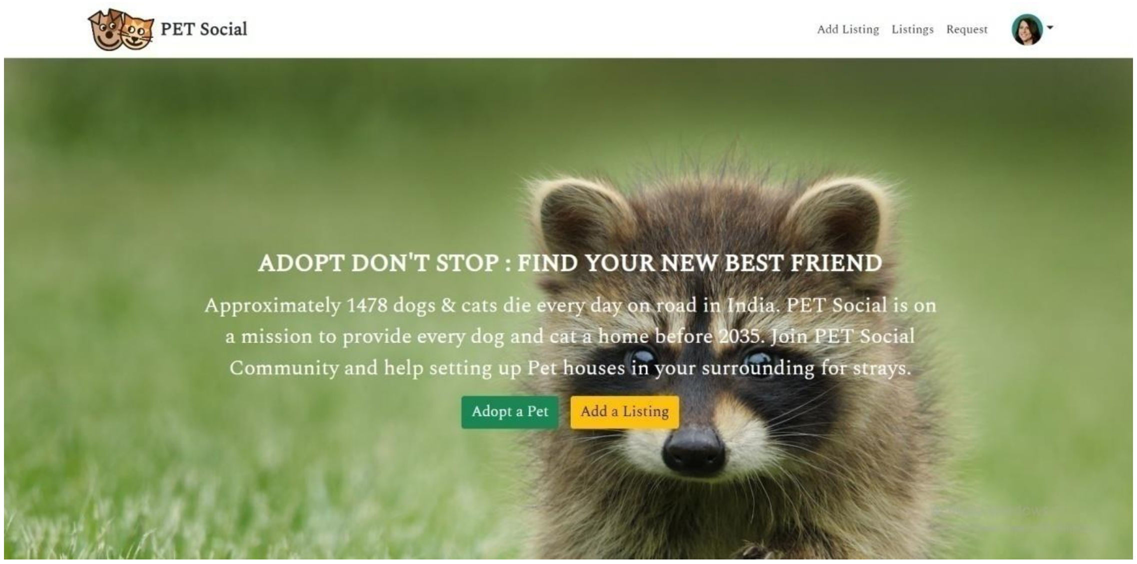

ADOPT LOVE WITH OUR PET SOCIAL
Sign in with Google
Activate Windows
Go to Settings to activate Windows.

Fig. No 5: Login Page before clicking the signing button


PET Social

Add Listing Listings Request

ADOPT DON'T STOP : FIND YOUR NEW BEST FRIEND

Approximately 1478 dogs & cats die every day on road in India. PET Social is on a mission to provide every dog and cat a home before 2035. Join PET Social Community and help setting up Pet houses in your surrounding for strays.

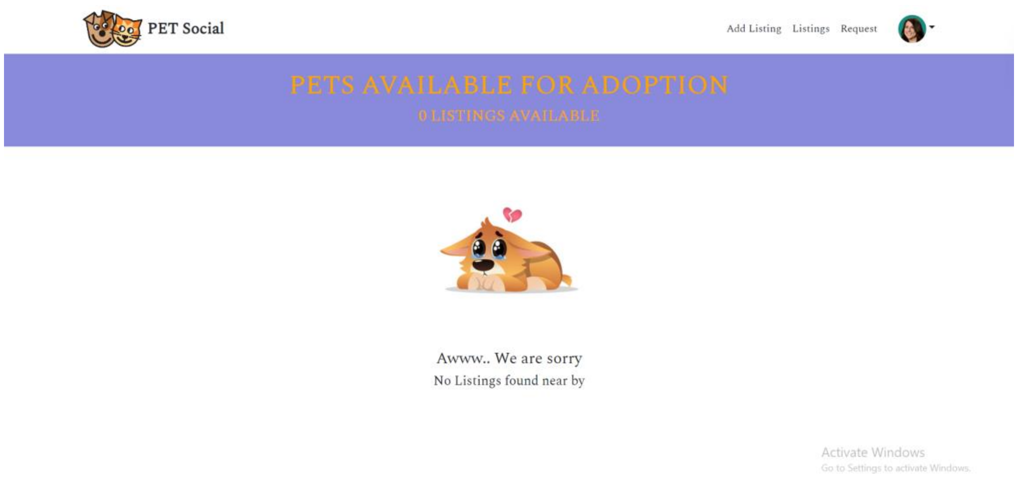
Adopt a Pet Add a Listing

Fig. No 6: On successful login

5.2 Test Report on No Listings Found

Test Case 1

If there are no listings found, it would automatically show "No Listings found near By".


PET Social

Add Listing Listings Request

PETS AVAILABLE FOR ADOPTION

0 LISTINGS AVAILABLE

Awww.. We are sorry

No Listings found near by

Activate Windows

Go to Settings to activate Windows.

Fig. No 7: No Listing Found Page

6. REPORTS TO BE GENERATED

The purpose of implementing reports with a platform, particularly for enterprise customers, is characterized as report creation. To write a report, we should first determine all content you really want acquire, where you would like to acquire it, and how you would like to represent that too. Here are few reports likely to be generated.

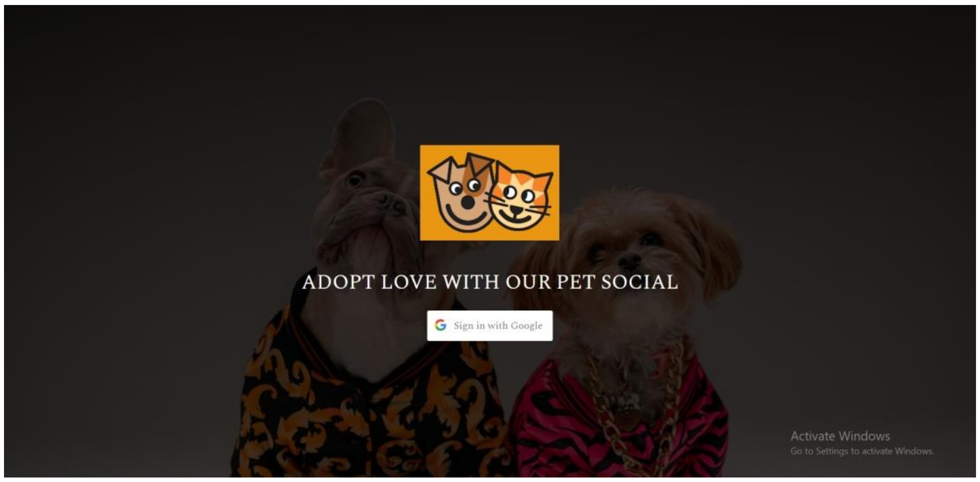
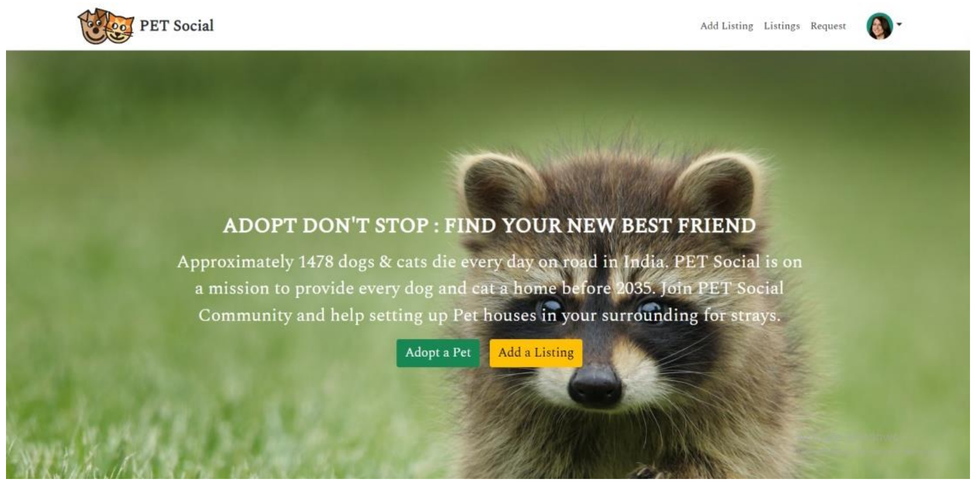
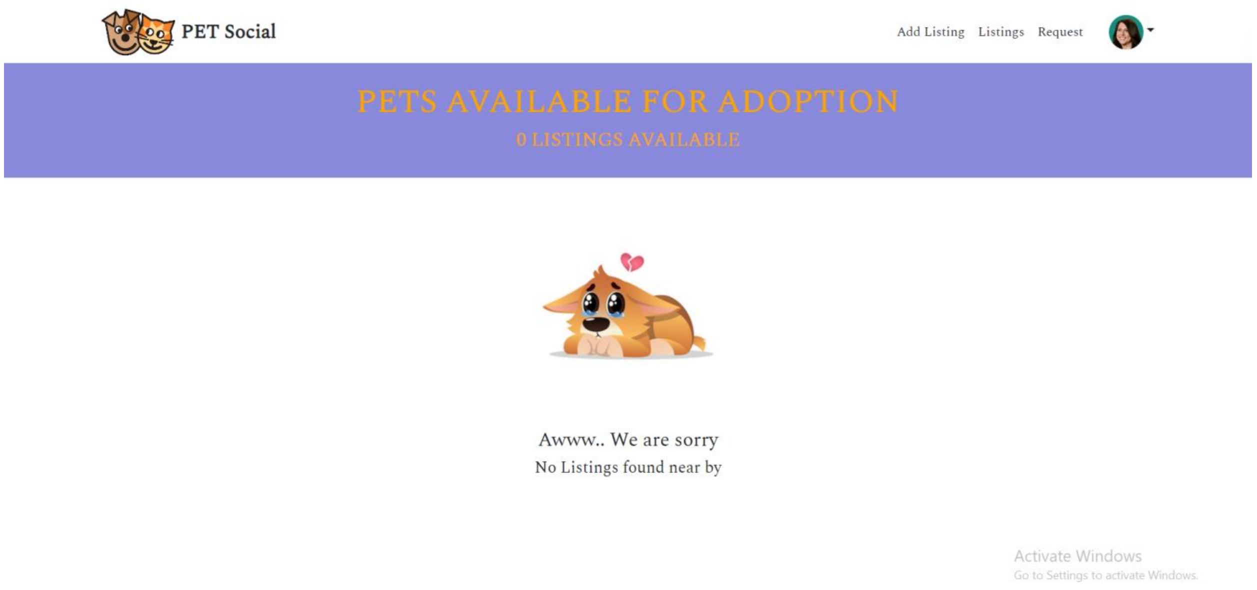

ADOPT LOVE WITH OUR PET SOCIAL
Sign in with Google
Activate Windows
Go to Settings to activate Windows.

Fig. No 8: Login Page



Approximately 1478 dogs & cats die every day on road in India. PET Social is on a mission to provide every dog and cat a home before 2035. Join PET Social Community and help setting up Pet houses in your surrounding for strays.

Fig. No 9: Home Page



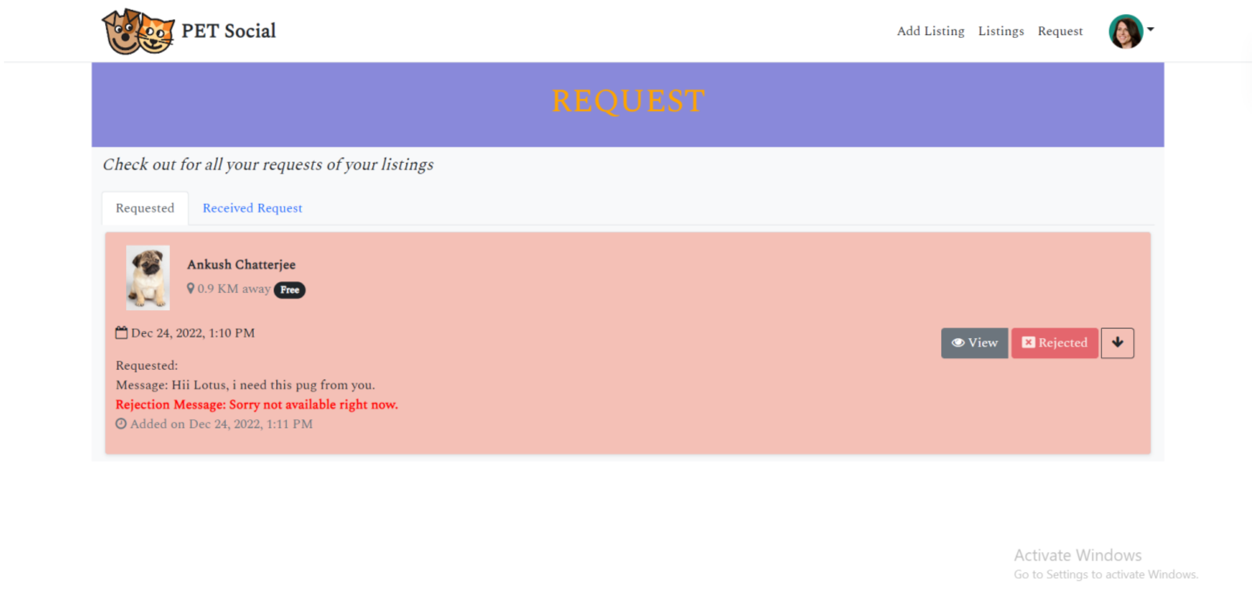
PETS AVAILABLE FOR ADOPTION

0 LISTINGS AVAILABLE

Awww.. We are sorry

No Listings found near by

Fig. No 10: Listing's Page



REQUEST

Check out for all your requests of your listings

Ankush Chatterjee

0.9 KM away Free

Dec 24, 2022, 1:10 PM

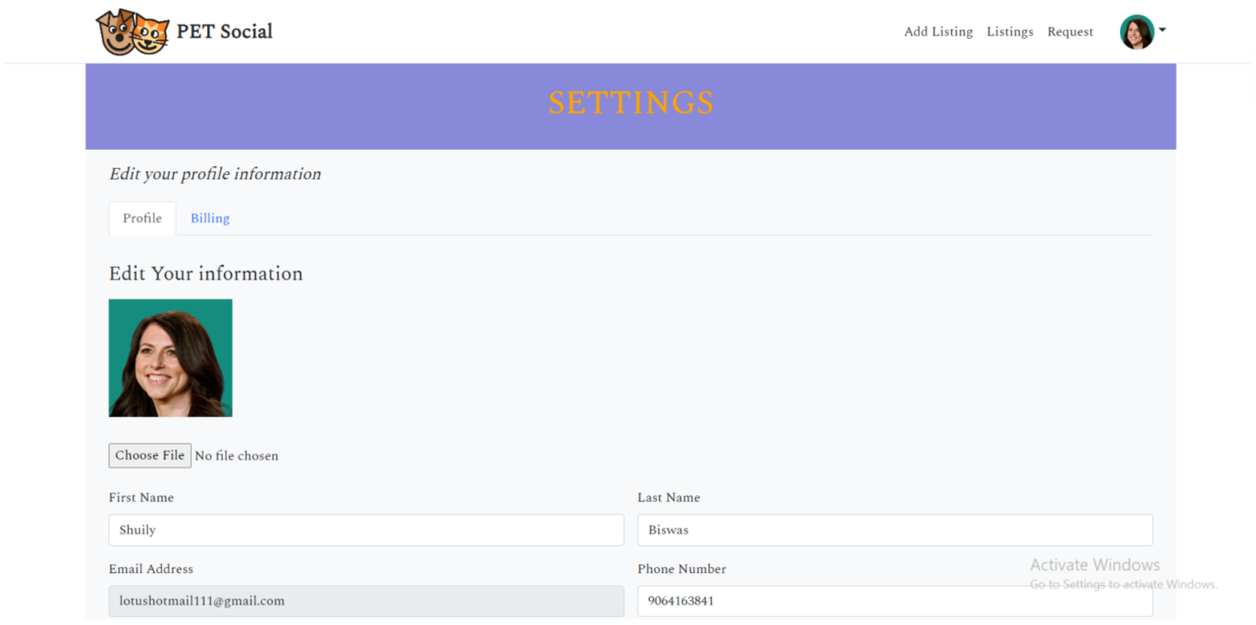
Requested:

Message: Hii Lotus, i need this pug from you.

Rejection Message: Sorry not available right now.

Added on Dec 24, 2022, 1:11 PM

Fig. No 11: Request Page



Edit your profile information

Profile | Billing

Edit Your information

No file chosen

First Name: Shuily
Last Name: Biswas
Email Address: lotushotmail111@gmail.com
Phone Number: 9064163841

Fig. No 12: Setting's Page

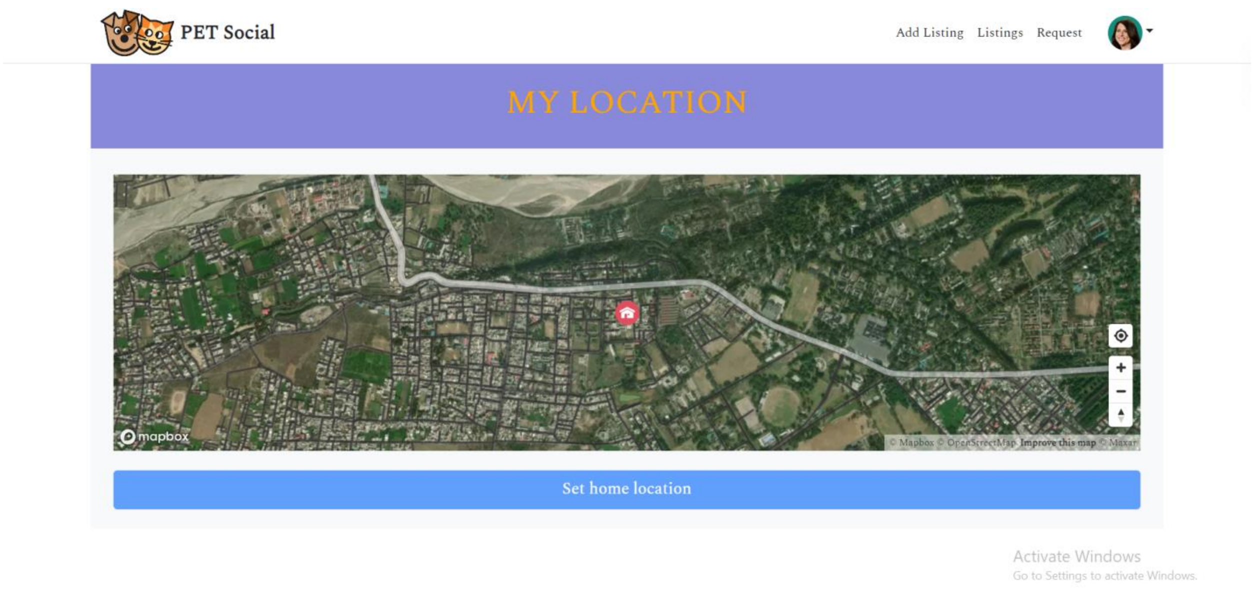


Fig. No 13: My Location Page

7. SYSTEM SECURITY MEASURES

We have focused to keep user’s data secured. We have defined and implemented various parameters to secure our application from exploits attacks. To validating user credentials, authentication with Google accounts, local storage, unique user token and secure API routes.

Proper message on successful events and errors, have been implemented so that the end user is aware of all the transactions and activities going across the application. We have unit tested every feature of the application, to find vulnerabilities and to fixed most of the minor and major bugs.

8. COST ESTIMATION MODEL

Project Name: HIBUY

Project Size: 10,000 Lines of Code (KLOC)

Development Schedule: 4 months

Team Size: 1 developer

CALCULATIONS:

Effort = a * (KLOC)^b * (sum(Ti))^(c) * (sum(Ej))^(d)

Assuming:

a = 2.4

b = 1.05

c = 0.38

d = 0.35

Effort = 2.4 * (10)^1.05 * (sum(Ti))^0.38 * (sum(Ej))^0.35

Effort = 31.8 Person-Months

Productivity = Project Size / Effort

Productivity = 10,000 / 31.8

Productivity = 314.5 Lines of Code per Person-Month

Development Time = Effort / Team Size

Development Time = 31.8 / 1

Development Time = 31.8 Months

Cost = Effort * Cost per Person-Month

Assuming:

Cost per Person-Month = $6,000

Cost = 31.8 * $6,000

Cost = $190,800

RESULTS:

Effort Required: 31.8 Person-Months

Productivity Rate: 314.5 Lines of Code per Person-Month

Development Time: 31.8 Months

Development Cost: $190,800

9. FUTURE SCOPE OF THE PROJECT

There are few ideas which we have planned in the near future. We are planning of building Email and Phone message notification alert automation, of popular or nearby listings and when a request has been made on the listings. So that it can keep the donor and sender notified of the status of the listings and request.

Other than this, we have planned to build a chatting gateway for the donor and the requester, so that they can exchange talks. In order to stay updated or for any other queries, they can resolve it through chats. For now, we only have these future enhancements of the project, can we believe implementing these would make our application more effective and would help the users as well for better experience.

10. APPENDICES

10.1 CODING

index.js

```
const express = require('express')

const mongoose = require('mongoose')

const cors = require('cors');

const bodyParser = require('body-parser')

const port  = process.env.PORT || 3000;


// Initliaze express server

const app = express();app.use(cors());

app.use(bodyParser.urlencoded({ extended: true }))


app.use(bodyParser.json())



const url = 'mongodb+srv://lotusbiswas:lotusbiswas@cluster0.1zfsoap.mongodb.net/hibuy'


mongoose.set('strictQuery', false);

mongoose.connect(url, {useNewUrlParser: true})


const con  = mongoose.connection
```

```
con.on('open', ()=>{
    console.log('connected');
})


// Router
const userRouter = require('./routes/user');
const listing = require('./routes/listing');
const requestRouter = require('./routes/request')


app.use('/user', userRouter);
app.use('/user/listing', listing);
app.use('/user/request', requestRouter);




app.get('/',(req,res)=>{
    res.send({status:"running"});
})
app.listen(port,()=>{
    console.log( `App listing at ${port}`);
})
```

package.json

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "lotusbiswas",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.0",
    "cors": "^2.8.5",
    "express": "^4.18.1",
    "mongodb": "^4.10.0",
    "mongoose": "^6.6.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.20"
  }
}
```

routes/user.js

```
const express = require('express')
const router = express.Router();
const User = require('../models/user')
```

```
router.get('/', async (req, res) => {
   try {
       const users = await User.find();
       res.json(users)


   } catch (err) {
       res.send('Error ' + err);
   }
})


// ADD A USER
router.post('/addUser', async (req, res) => {


   // Pre User if already exist or not
   const userExist = await User.findOne({emailAddress:req.body.emailAddress});


   if(userExist)
   {
       return res.status(200).json(userExist);
   }else{
       const user = new User({
           firstName: req.body.firstName,
           lastName: req.body.lastName,
           emailAddress: req.body.emailAddress
```

```
        })


        try {
            const u1 = await user.save();
            res.status(200).json(u1);
        } catch (err) {
            res.status(502).send('Error ' + err);
        }
    }


})


//  UPDATE USER MY LOCATION
router.patch('/updateMyLocation/:id', async(req, res)=>{
    try{
        const user = await User.findById(req.params.id);
        user.myLocation ={"lng":req.body.lng,"lat":req.body.lat};
        const u1 = await user.save();


        res.status(200).json(true);


    }catch (err) {
        res.status(502).send('Error ' + err);
    }
```

```
});


//  UPDATE USER PROFILE PICTURE

router.patch('/updateMyProfilePicture/:id', async(req, res)=>{

   try{

       const user = await User.findById(req.params.id);

       user.profilePicture =req.body.profilePicture;

       const u1 = await user.save();



       res.status(200).json(true);



   }catch (err) {

       res.status(502).send('Error ' + err);

   }



});



//  UPDATE USER DATA

router.put('/updateUserData/:id', async(req, res)=>{

   try{

       const user = await User.findById(req.params.id);

       user.firstName =req.body.firstName;

       user.lastName =req.body.lastName;
```

```
        user.mobileNumber =req.body.mobileNumber;

        const u1 = await user.save();


        res.status(200).json(true);


    }catch (err) {

        res.status(502).send('Error ' + err);

    }


});




//  GET USER LOCATION

router.get('/getMyLocation/:id', async(req, res)=>{

    try{

        const user = await User.findById(req.params.id);

        const lng = user.myLocation.lng;

        const lat =  user.myLocation.lat;


         res.status(200).json(

            {

            "status": true,

             "lng":lng,

             "lat":lat
            }
            );

    }catch (err) {
        res.status(502).send('Error ' + err);
    }

});



// GET USER BY ID
router.get('/getUserDataById/:id', async (req, res) => {
    try {
        const user = await User.findById(req.params.id);
        res.json([user]);

    } catch (err) {
        res.send('Error ' + err);
    }
})

// HIGHEST REWARD POINT

router.get("/getHighestRewardPointMember", async(req,res)=>{
    try{

        const user = await User.find({}).sort({rewardPoints:-1}).limit(1);

        res.json(user);


    }catch(err)

    {

        res.status(502).send('Error ' + err);

    }

})


// GET USER BY ID

router.get('/:id', async (req, res) => {

    try {

        const user = await User.findById(req.params.id);

        res.json(user)


    } catch (err) {

        res.send('Error ' + err);

    }

})


// UPDATE

router.patch('/:id', async(req,res)=>{

try{
    const user = await User.findById(req.params.id);

    user.aboutYou = req.body.aboutYou;

    const u1 = await user.save();



    res.json(u1);



}catch(err){

    res.send('Error ' + req);

}

})



//DELETE

router.delete('/:id', async(req,res)=>{

    try{

        const user = await User.findById(req.params.id);

        const u1 = await user.remove();

        res.json("success");



    }catch(err){

        res.send('Error '+ err );

    }

})



module.exports = router;
```

routes/request.js

```
const express = require('express')

const router = express.Router();

const Request = require('../models/request');



const FreeListing = require('../models/freeListing');



router.post('/add/newRequest', async (req, res) => {

   try {

       const newRequest = new Request({

           listId: req.body.listId,

           listType:req.body.listType,

           listedUserToken: req.body.listedUserToken,

           requesterUserToken: req.body.requesterUserToken,

           request_message: req.body.request_message,

           acceptance_status: req.body.acceptance_status,

           rejection_message: ""

       })



       const r1 = await newRequest.save();

       res.status(200).json(true);



   } catch (err) {

       res.status(200).send('Error ' + err);
```

```
    }
})


// REJECTION MESSAGE
router.patch('/add/rejectionMessage/:id', async (req, res) => {
    try {
        const requestOne = await Request.findOne({
            _id: req.params.id,
        });
        requestOne.rejection_message = req.body.rejection_message;
        requestOne.acceptance_status = req.body.acceptance_status;


        const r1 = await requestOne.save();
        res.status(200).json(true);


    } catch (err) {
        res.status(200).send('Error ' + err);
    }
})


// ACCEPTANCE
router.patch('/update/acceptanceStatus/:reqId/:listId', async (req, res) => {
    try {
        const requestOne = await Request.findOne({
```

```
    _id: req.params.reqId,
});
const freeList = await FreeListing.findOne({
    _id:req.params.listId
});
if(req.body.listType == 'listing')
{
    if(req.body.acceptance_status == 'delivered' )
    {
        requestOne.acceptance_status = req.body.acceptance_status;
        freeList.onHold = false;
        freeList.disable = true;
        const r1 = await requestOne.save();
        const f1 = await freeList.save();
    }

    else{
        requestOne.acceptance_status = req.body.acceptance_status;
        const r1 = await requestOne.save();
    }
}
```

```
        res.status(200).json(true);


    } catch (err) {

        res.status(200).send('Error ' + err);

    }

})




router.get('/get/allRequest/requested/token/:token1', async (req, res) => {

    try {

        const allRequest = await Request.find({


            requesterUserToken: req.params.token1


        })


        res.status(200).json(allRequest);


    } catch (err) {

        res.status(200).send('Error ' + err);

    }

})
```

```
router.get('/get/allRequest/received_request/token/:token1', async (req, res) => {
    try {
        const allRequest = await Request.find({

            listedUserToken: req.params.token1

        })

        res.status(200).json(allRequest);

    } catch (err) {
        res.status(200).send('Error ' + err);
    }
})

module.exports = router;
```

routes/listing.js

```
const express = require('express');
const router = express.Router();
const FreeListing = require('../models/freeListing');
const User = require('../models/user');
const Request = require('../models/request');
// GET ALL FREE LISTING
```

```
router.get('/get/freeListing/', async (req, res) => {
    try {
        const allList = await FreeListing.find();
        res.status(200).json(allList);
    } catch (err) {
        res.status(502).send('Error ' + err);
    }
});

// GET LISTING BY USER TOKEN

router.get('/get/freeListing/userToken/:id', async (req, res) => {
    try {
        const allList = await FreeListing.find({
            userToken: req.params.id
        });
        res.status(200).json(allList);
    } catch (err) {
        res.status(502).send('Error ' + err);
    }
});

// GET LISTING BY LISTING ID
```

```
router.get('/get/freeListing/listingId/:listId/:userToken', async (req, res) => {
    try {
        const list = await FreeListing.find({
            _id: req.params.listId,
            userToken: req.params.userToken
        });

        res.status(200).json(list);
    } catch (err) {
        res.status(502).send('Error ' + err);
    }
});

router.get('/get/listings/nearBy/:lng/:lat', async (req, res) => {
    try {

       const longitude = Number(req.params.lng);
        const latitude = Number(req.params.lat);
       const find_nearBy= await FreeListing.find({
        location:{
            $near:{
                $geometry:{
                    type:"Point",
                    coordinates:[
                        longitude,
                        latitude
                    ]
                },

                $maxDistance : 25000,

            }
        }
       })



        res.status(200).json(find_nearBy);
    } catch (err) {
        res.status(502).send('Error ' + err);
    }
})

// ADD FREE LISTING
router.post('/add/freeListing', async (req, res) => {

    try {
        const freeList = new FreeListing({
            userToken: req.body.userToken,

            picture: req.body.picture,

            petsname: req.body.petsname,

            breed: req.body.breed,

            gender: req.body.gender,

            vaccinated: req.body.vaccinated,

            age: req.body.age,

            nurtured: req.body.nurtured,

            age: req.body.age,

            category: req.body.category,

            description: req.body.description,

            pickUpTime: req.body.pickUpTime,

            listFor: req.body.listFor,

            location: { "lng": req.body.lng, "lat": req.body.lat }

        })


        const f1 = await freeList.save();

        res.status(200).json(true);


    } catch (err) {

        res.status(200).send('Error ' + err);

    }

});
```

```
// UPDATE FREE LISTING ITEM


router.put('/update/freeListing/:listId/:userToken', async (req, res) => {

  try {

    const freeList = await FreeListing.findOne({

      _id: req.params.listId,

      userToken: req.params.userToken

    });

    freeList.picture = req.body.picture;

    freeList.petsname= req.body.petsname,

    freeList.breed= req.body.breed,

    freeList.gender= req.body.gender,

    freeList.vaccinated= req.body.vaccinated,

    freeList.age= req.body.age,

    freeList.nurtured= req.body.nurtured,

    freeList.age= req.body.age,

    freeList.category = req.body.category;

    freeList.description = req.body.description;

    freeList.pickUpTime = req.body.pickUpTime;

    freeList.listFor = req.body.listFor;

    freeList.location = { "lng": req.body.lng, "lat": req.body.lat };


    const u1 = await freeList.save();

    console.log(u1);
```

```
        res.status(200).json(true);

    } catch (err) {

        res.status(502).send('Error ' + err);

    }

});


//  UPDATE FREE LISTING  PICTURE

router.patch('/update/freeListingPicture/:listId/:userToken', async (req, res) => {

    try {

        const freeList = await FreeListing.find({

            _id: req.params.listId,

            userToken: req.params.userToken

        });

        freeList.picture = req.body.picture;

        const l1 = await freeList.save();


        res.status(200).json(true);


    } catch (err) {

        res.status(502).send('Error ' + err);

    }


});
```

```
// ON HOLD

router.patch('/update/onHoldListing/:id', async (req, res) => {

    try {

        const list = await FreeListing.findOne({

            _id: req.params.id,

        });

        list.onHold = req.body.onHold;


        const l1 = await list.save();

        res.status(200).json(true);


    } catch (err) {

        res.status(200).send('Error ' + err);

    }

})




//  DISBALE LISTING VIEW STATUS

router.patch('/update/disableStatusFreeListing/:listId/:userToken', async (req, res) => {

    try {

        const freeList = await FreeListing.findOne({

            _id: req.params.listId,

            userToken: req.params.userToken

        });
        freeList.disable = req.body.disableStatus;

        const l1 = await freeList.save();



        res.status(200).json(true);



    } catch (err) {

        res.status(502).send('Error ' + err);

    }



});



//  UPDATE ADD LIKE TO LISTING

router.patch('/update/addLikeFreeListing/:listId', async (req, res) => {

    try {

        const freeList = await FreeListing.findByIdAndUpdate(req.params.listId,



            { $push: { likes: { "listId": req.params.listId, "userToken": req.body.userToken } } },



            { 'upsert': true });



        const l1 = await freeList.save();



        res.status(200).json(true);
    } catch (err) {
        res.status(502).send('Error ' + err);
    }

});

// REMOVE LIKE FROM LISTING
router.patch('/update/removeLikeFreeListing/:listId', async (req, res) => {
    try {
        const freeList = await FreeListing.findByIdAndUpdate(req.params.listId,

            { $pull: { likes: { "listId": req.params.listId, "userToken": req.body.userToken } } });

        const l1 = await freeList.save();

        res.status(200).json(true);

    } catch (err) {
        res.status(502).send('Error ' + err);
    }

});

// DELETE FREE LISTING
```

```
router.delete('/delete/freeListing/:listId/:userToken', async (req, res) => {
    try {
        const freeList = await FreeListing.findOne({
            _id: req.params.listId,
            userToken: req.params.userToken
        });

        const requestOne = await Request.findOne({
            listId: req.params.listId,
            listedUserToken: req.params.userToken
        });

        const f1 = await freeList.remove();
        const r1 = await requestOne.remove();
        res.status(200).json(true);

    } catch (err) {
        res.status(200).json('Error ' + err);
    }
})

module.exports = router;
```

10.2 BIBLIOGRAPHY

[1] "Urgent Need for Pet Adoption - Find Dogs & Cats & More | Petfinder." https://www.petfinder.com/ (accessed May 06, 2023).

[2] "The Importance of Animal Shelters | Blog | Richell USA." https://www.richellusa.com/the-importance-of-animal-shelters/ (accessed May 06, 2023).

[3] "Adopt a dog or cat today! Search for local pets in need of a home." https://www.adoptapet.com/ (accessed May 06, 2023).

[4] "10 Good Reasons To Adopt & Not Shop!" https://www.veganfirst.com/article/10-good-reasons-to-adopt-not-shop- (accessed May 06, 2023).

[5] "5 Reasons You Should Adopt From An Animal Shelter - Animals Matter To Me." https://www.amtmindia.org/5-reasons-you-should-adopt-from-an-animal-shelter/ (accessed May 06, 2023).

[6] "Why do animals need shelter?" https://www.vedantu.com/question-answer/why-do-animals-need-shelter-class-11-biology-cbse-6110850fd608bc68885bc5d8 (accessed May 06, 2023).

[7] "Animal Shelters FAQ: What to Know Before Adopting a 'Pet.'" https://www.peta.org/features/animal-shelters/ (accessed May 06, 2023).

[8] T. Point, "UML - Activity Diagrams," *www.tutorialspoint.com*, 2018. https://www.tutorialspoint.com/uml/uml_activity_diagram.htm

[9] J. M. Almendros-Jiménez and L. Iribarne, "Describing use-case relationships with sequence diagrams," *Comput. J.*, vol. 50, no. 1, pp. 116–128, Jan. 2007, doi: 10.1093/COMJNL/BXL053.

[10] "Unified Modeling Language (UML) | An Introduction - GeeksforGeeks." https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ (accessed May 06, 2023).

[11] “Software Testing - Quick Guide.” https://www.tutorialspoint.com/software_testing/software_testing_quick_guide.htm (accessed May 06, 2023).

[12] “What is Software Testing? Definition, Types and Importance.” https://www.techtarget.com/whatis/definition/software-testing (accessed May 06, 2023).

[13] “What is Software Testing? Definition.” https://www.guru99.com/software-testing-introduction-importance.html (accessed May 06, 2023).

[14] I. Chana and P. Chawla, “Testing Perspectives for Cloud-Based Applications,” pp. 145–164, 2013, doi: 10.1007/978-1-4471-5031-2_7.