

PALLIATIVE CARE PLATFORM

A MINI PROJECT REPORT

Submitted by

SIJI JOSE (TKM20MCA2037)

to

The APJ Abdul Kalam Technological University

In partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS



**Thangal Kunju Musaliar College of Engineering
Kerala**

MARCH 2022

DECLARATION

We undersigned hereby declare that the mini project report — PALLIATIVE CARE PLATFORM, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of NATHEERA BEEVI M. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Karicode

SIJI JOSE

Date: 02/03/2022

Thangal Kunju Musaliar College of Engineering
Dept. of Computer Applications



C E R T I F I C A T E

This is to certify that, the report entitled “***PALLIATIVE CARE PLATFORM***” submitted by **SIJI JOSE (TKM20MCA2037)** to the **APJ Abdul Kalam Technological University** in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor

Mini Project Coordinator

.....

.....

ACKNOWLEDGEMENT

First and foremost, I thank GOD almighty and our parents for the success of this project. We owe sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of our project.

I am extremely grateful to **Prof. Dr. Nadera Beevi S**, Head of the Department, for providing us with best facilities.

I would like to place on record my sincere gratitude to my external project guide **Joy Sebastian**, Co-Founder and Chief Executive Officer at Techgentsia Software Technologies and internal project guide **Natheera Beevi M**, Department of Computer Applications for the guidance and mentorship throughout the project.

I profusely thank all other faculty members in the department and all other members of TKM College of Engineering, for their guidance and inspirations throughout our course of study.

I owe my thanks to my friends and all others who have directly or indirectly helped us in the successful completion of this project.

SIJI JOSE

Abstract

“Palliative Care Platform” aim to provide care such as home nursing services, ambulatory services, equipment and so on. Palliative care is specialized medical care for people living with a serious illness. This type of care is focused on providing relief from the symptoms and stress of the illness. The goal is to improve quality of life for both the patient and the family. Palliative care is provided by a specially trained team of doctors, nurses and other specialists who work together with a patient’s other doctors to provide an extra layer of support. Palliative care is based on the needs of the patient, not on the patient’s prognosis. It is appropriate at any age and at any stage in a serious illness, and it can be provided along with curative treatment. The services of the service providers will be listed and published. They have a special account and will publish it. The user can filter the service based on location and service type. For example, when searching for services such as home nursing care, the companies that provide it are listed. User can subscribe and use the services according to their needs and user can review the service.

TABLE OF CONTENTS

CHAPTER.NO	TITLE	PAGE.NO
	ABSTRACT	
1.	INTRODUCTION	7
2	SYSTEM ANALYSIS	9
3	SYSTEM SPECIFICATIONS	11
4	SYSTEM DESIGN	12
5	CODING	22
6	TESTING	39
7	IMPLEMENTATION	41
8	CONCLUSION	44
9	FUTURE ENHANCEMENT	45
10	SCREENSHOTS	46
11	REFERENCES	50

CHAPTER 1

INTRODUCTION

1.1 Introduction

“Palliative Care Platform” is a web-based application. It provides various kinds of services such as home nursing services, ambulatory services, equipment and so on. Various service providers register to this platform to provide above mentioned services. The services registered by the service providers will be listed only after admin approves. The approval will be dependent on various documents corresponding to their service. Then the user can book the service based on their need and make payment to ensure service. User can make reviews related to their service so that it will help the upcoming users to get an overall idea about the services provided by various service providers. The project is developed using Python as designing tool and MySQL as database. Python is a powerful tool for web programming From Microsoft and is the front end of this project with MySQL as backend.

1.2 Objective

The goal of palliative care is to relieve suffering and provide the best possible quality of life for patients and their families. Symptoms may include pain, depression, shortness of breath, fatigue, constipation, nausea, loss of appetite, difficulty sleeping, and anxiety. The team will help you gain the strength to carry on with daily life. In short, palliative care will help improve your quality of life. And recent studies, including one published in the New England Journal of Medicine, have shown that patients with a serious illness who received palliative care lived longer than those who did not receive this care.

The key Features are:

- Improves Quality of Life
- Relieves Suffering from Symptoms and Stress
- Helps You Match Treatment Options to Your Goals
- Works Together with Your Other Doctors

1.3 Statement of the problem

This system transparently works, hand in hand with human ethics and charges are very genuine. We give a discount amount for the payment by categorizing the patients based on their financial status. Here we also make a platform for good people who wish to give a helping hand for the poor. This is by introducing a section called followers and here we coordinate the people who deserve to be helped and the people who wish to help together

CHAPTER 2

SYSTEM ANALYSIS

2.1 Present System

The present system functioning in the organization is done manually. The existing system is “Online Hospital”. This system provides treatment for bedridden patients with the help of an ambulance. This system works manually.

2.2 Limitations of Present System

- Complex updating.
- Time consuming process
- Difficulty in data retrieval.
- More chances for errors.

2.3 Proposed System

The software Palliative Care is a system which makes treatment available for bedridden patients. It promotes mobile hospital service. The system adds up’s advanced features to the existing system. Here we co-ordinates people who deserve to be helped and those who are willing to help together. It is designed keeping in mind all the drawbacks of the present system and to provide a permanent solution to the existing problems.

2.4 Advantages and Features of Proposed System

- The proposed system is of great speed and accuracy.
- It is almost error free.
- The system supports strong authentication features.
- More user friendly

2.5 Feasibility Study

The development of a computer-based system or product is more likely plagued by resources and difficult delivery data. It is both necessary and prudent to evaluate the feasibility of a project at a time months and years of effort, thousands or millions and untold professional embarrassment can be averted if an ill-convinced system is recognized early in the definition phase.

2.5.1 Economic feasibility:

Economic analysis commonly referred to as cost and benefit analysis. Economic justification is the basic consideration for most of the system. One of the factors which effect the development of a new system is the cost would require. An evaluation of development cost weighted against the ultimate income, or the benefit derived from the developed system or product.

2.5.2 Technical feasibility:

Technical feasibility checks whether the system is technically feasible to the organization. The developed system is technically feasible and easy to work with the system. It is essential that the process of analysis and definition conducted in parallel with an assessment of technical feasible. It determines the computer system and to extend it can support the proposed system. It deals with the hardware as well as software requirements.

2.5.3 Operational feasibility:

Proposed projects are beneficial only if they can be turned into information systems that will meet the operating requirements of the organization. This test of feasibility asks if the system will work when it is developed and installed. This project satisfies all the operational conditions

2.5.4 Legal feasibility:

A determination of any infringement, violation or liability that could result from the development of the system. An evolution of alternative approaches to the development of the system or product. A feasibility study is not warranted systems, in which economic justification is obvious, technical risk is low, few legal problems are expected, and no reasonable alternative exists. However, if any of the preceding conditions fail, a study of that area should be conducted. Economic justification includes a broad range of concerns that include cost-benefit analysis, long-term corporate income strategies, impact on other profit concerns or products, cost of resources needed for development and potential market growth.

CHAPTER 3

SYSTEM SPECIFICATIONS

3.1 Software Requirements

OPERATING SYSTEM	:	WINDOWS XP/7
FRONT END	:	Python
BACK END	:	MySQL Server

3.2 Hardware Requirements

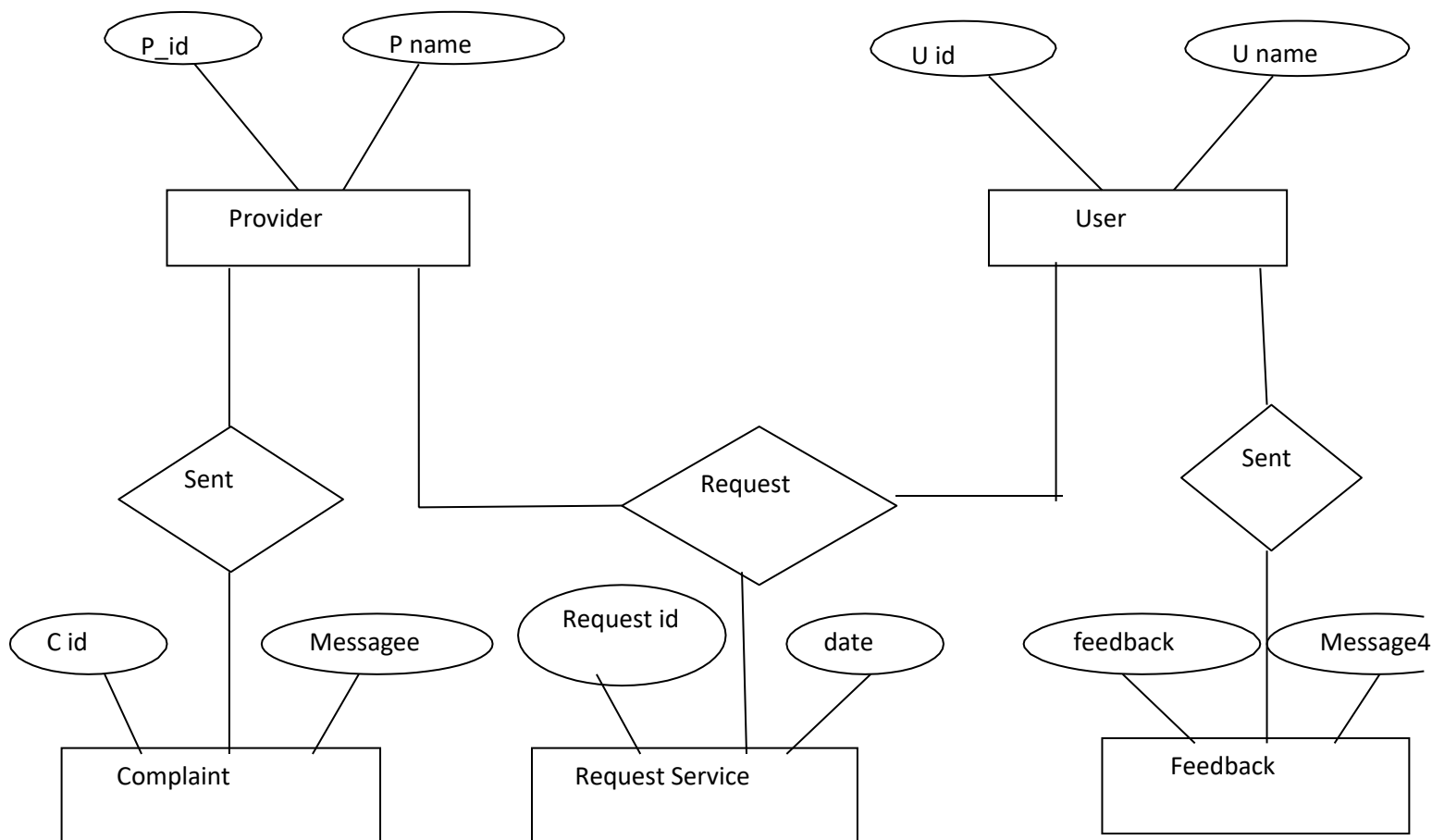
CPU	:	PENTIUM 1V
CPU CLOCK	:	1.5 GHz
RAM	:	2GB
HARD DISK	:	500 GB
KEYBOARD	:	STANDARD
MOUSE	:	NORMAL
PRINTER	:	INKJET

CHAPTER 4

SYSTEM DESIGN

Design of a system is the most creative and challenging phase of the system development lifecycle. The term design describes the final system and the process by which it is developed. The first step is to determine how the output is to be produced and in what format. Samples of the output are also presented. Second, input data and master files have to be designed to meet the requirements of the system. The final reports to implementation phase include procedural flowchart, report layouts etc.

ER DIAGRAM



4.1 Database Design

Database design forms an important part of every project. The management of data involves both the definition of structure for the storage of information and provision of mechanisms for manipulation of information. The database system must provide safety for the information stored; despite system crashes or attempts of unauthorized access the database used in this project is MYSQL.

Database constructed using relation is termed as relational database. The row of a table is referred as tuple. Each column in a table has column names. Column or a field is called an attribute. The number of tuples in a relation is called cardinality and the number of attributes is called degree. Every table must have some column or combination of columns that uniquely identify each row in a table. It is called primary key. A column in one table with a value that matches the primary key in some other table is called a foreign key. Together a primary key and foreign key create a parent-child relationship between tables.

The table has the following properties:

- Each entry in a table represents one data item.
- They are column homogeneous, in any column all items are of same kind.
- Each column is assigned a distinct name.
- All rows are distinct.

The database design is made up of three levels

- Conceptual level (High level)
- Physical level (Low level)
- View level (Representation level)

Conceptual Level

Conceptual level describes the essential features of the system data just like a DFD for system. It uses symbols and is called Entity-Relationship analysis. An entity is a conceptual representation of an object. Relationship between entities used to make the database structure.

1. A one-to-one relationship is an association between two entities.

2. A one-to-many relationship describes an entity that may have two or more entities related to it.
3. A many-to-many relationship describes entities that may have many relationships in both directions.

Physical Level

In this level the data is stored physically. That is an internal schema describes the physical storage structure of the database.

View Level

This level is used to describe how the user views the records or objects in the database.

TABLES

TABLE NAME: health_login

Field Name	Data Type	Size	Constraints	Description
<i>logid</i>	int	11	Primary Key	<i>Login id</i>
username	varchar	100	Not Null	username
password	varchar	100	Not Null	password
role	varchar	10	Not Null	role

Table Name: health_service request

Field Name	Data Type	Size	Constraints	Description
<i>request_id</i>	Int	11	Primary Key	<i>Request id</i>
service_type	Int	100	Not Null	Service Type
service_details	Varchar	1000	Not Null	Service details
reg_date	Varchar	200	Not Null	Register Date
request_date	Varchar	100	Not Null	Request Date
request_rating	Varchar	200	Not Null	Request Rating
request_status	Int	11	Not Null	Request Status
provide_login_id	int	11	Not Null	Provide Login Id

Table Name: health_feedback

Field Name	Data Type	Size	Constraints	Description
<i>Feedback_id</i>	int	11	Not Null	<i>Feedback_id</i>
Feedback_subject	varchar	100	Not Null	Feedback_subject
Feedback_message	varchar	500	Not Null	Feedback_message
Feedback_date	varchar	100	Not Null	Feedback_date
Feedback_reply	varchar	500	Not Null	Feedback_reply
user_id	int	11	Primary Key	user_id

Table Name: health_catgerory

Field Name	Data Type	Size	Constraints	Description
<i>category_id</i>	Int	11	Primary Key	<i>Category id</i>
catgerory_name	Varchar	200	Not Null	Catgerory Name
catgerory_address	varchar	200	Not Null	Catgerory Address

Table Name: health_user

Field Name	Data Type	Size	Constraints	Description
<i>user_id</i>	int	11	Primary Key	<i>User id</i>
user_name	Varchar	200	Not Null	User Name
user_address	Varchar	200	Not Null	User Address
user_email	Varchar	100	Not Null	User Email
user_phone	Varchar	100	Not Null	User Phone
user_status	Varchar	200	Not Null	User Status
log_id	int	11	Not Null	Login id
user_district	Varchar	200	Not Null	User District

Table Name: health_catgerory

Field Name	Data Type	Size	Constraints	Description
<i>category_id</i>	int	11	Primary Key	<i>Route id</i>
category_name	Varchar	200	Not Null	Route name
Category_address	Varchar	200	Not Null	Route from

Table Name: health_provider

Field Name	Data Type	Size	Constraints	Description
<i>Provide_id</i>	int	11	Not Null	Provide Id
Provide_name	varchar	100	Not Null	Provide Name
Provide_address	varchar	500	Not Null	Provide Address
Provide_email	varchar	200	Not Null	Provide Email
Provide_phone	varchar	100	Not Null	Provide Phone
Provide_status	varchar	50	Not Null	Provide Status
Provide_logid_id	int	11	NULL	Provide Login Id
catgerory_id	int	11	NULL	Category Id

Table Name: health_payment

Field Name	Data Type	Size	Constraints	Description
<i>pay_id</i>	int	11	Primary Key	Payment Id
amount	Varchar	100	Not Null	Amount
Pay_status	Varchar	100	Not Null	Payment Status
Servicerequest_id	Int	11	Not Null	Service Request Id
User_id	Int	11	Not Null	User Id

4.2 Normalization

Normalization is a process of reducing redundancies of data in a database. It is a technique that is used when designing and redesigning a database. The first step consists of transforming the data items so that a flat file is obtained.

First Normal Form

At the intersection of row and column, there is only one value in the tuple. Grouping of values are not allowed. A database is said to be in 1NF, if all the relations of database are in 1NF.

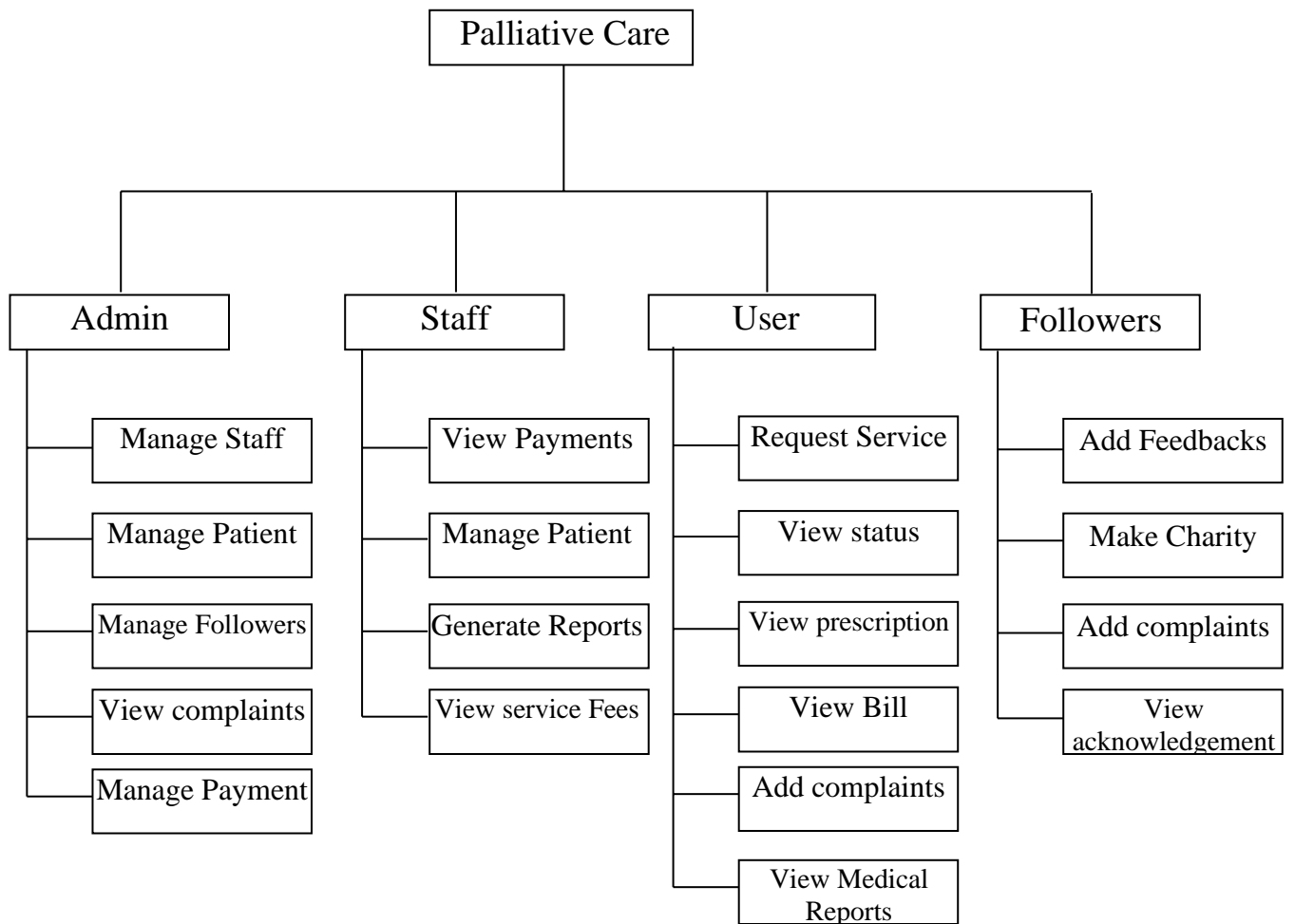
Second Normal Form

The relation is said to be in second normal form, every non-key attribute is fully functionally dependent on primary key.

Third Normal Form

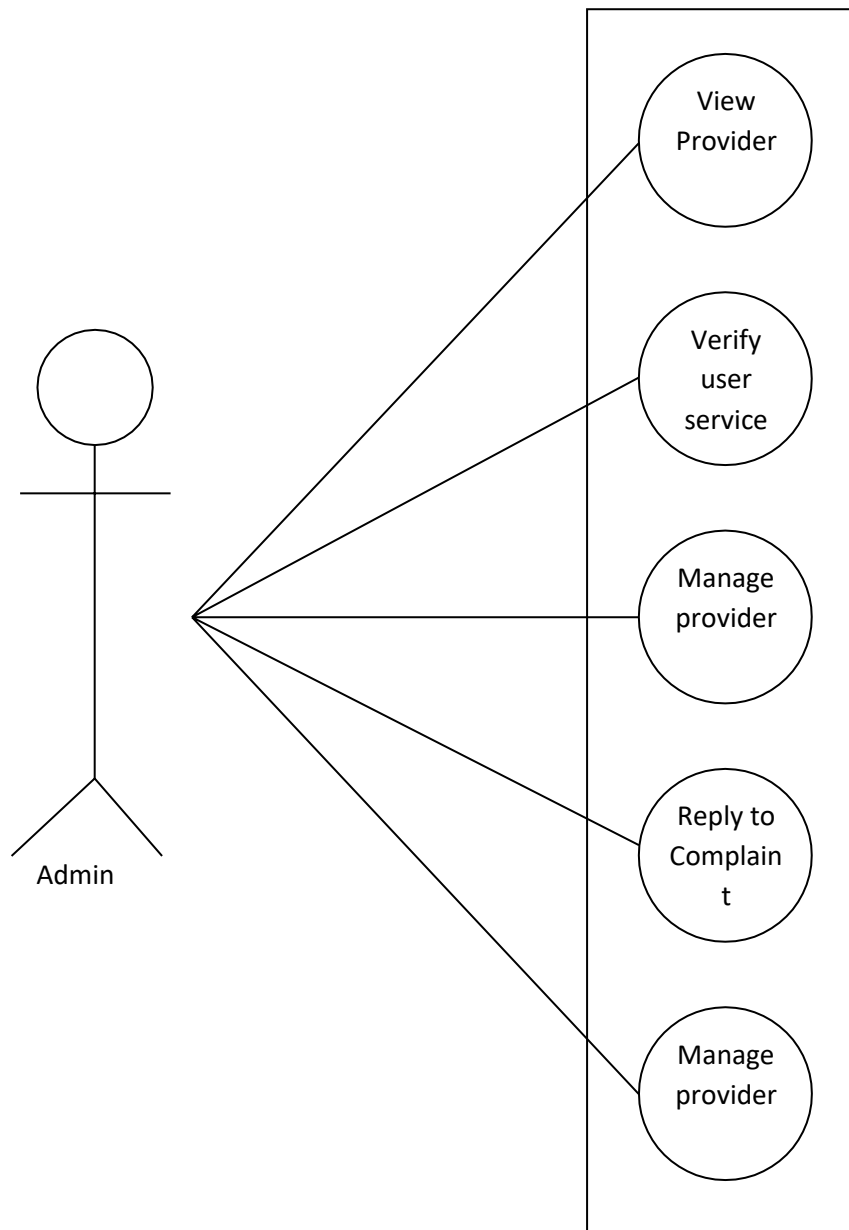
The relation is said to be in third normal form, there exist no transitive functional dependency between non-key attributes.

4.3 Design of Each Subsystem

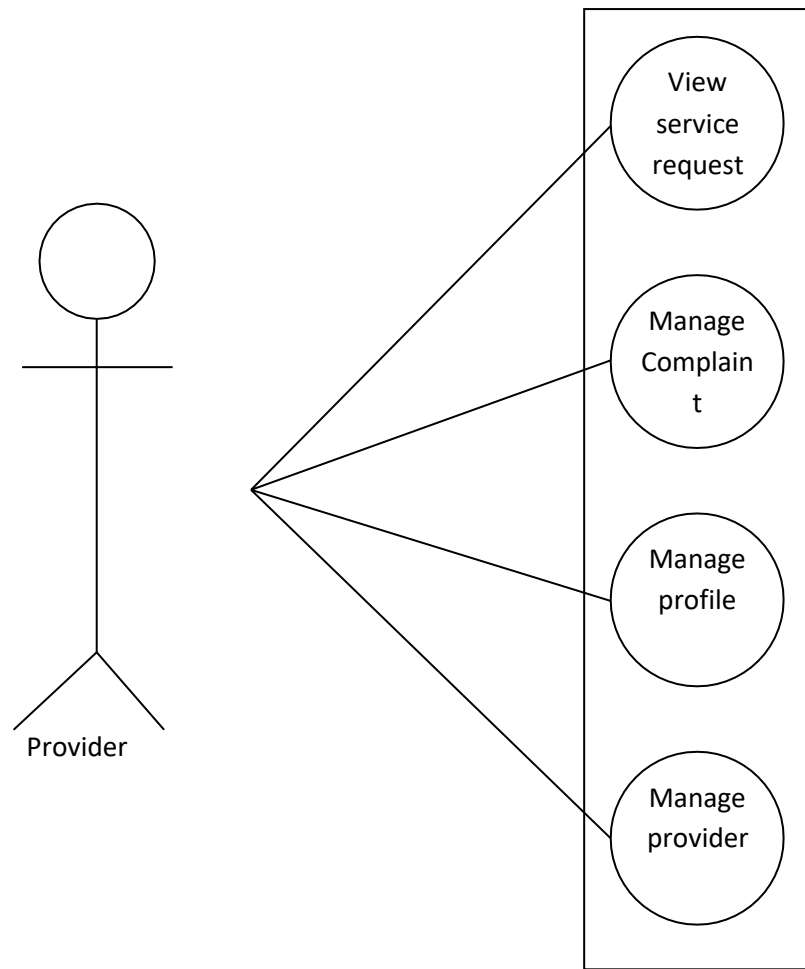


4.4 Use case Diagram

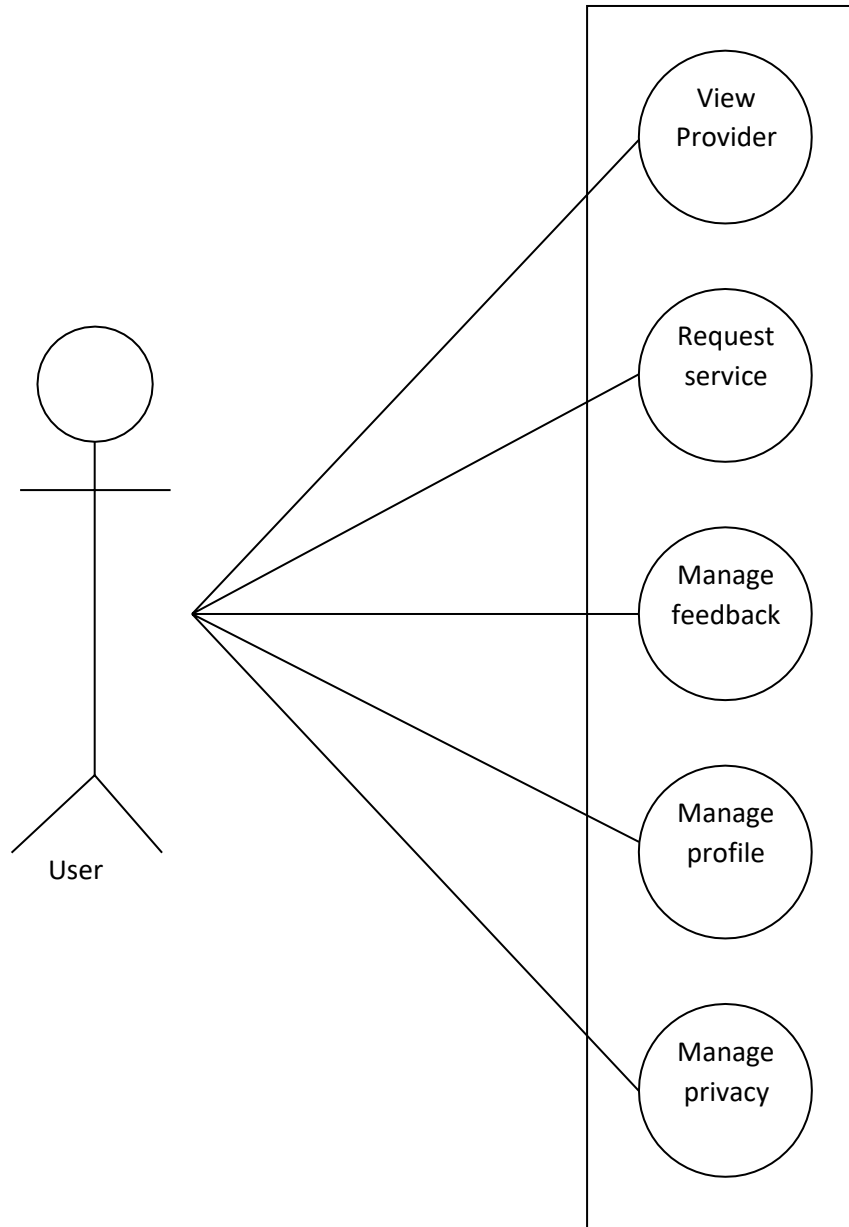
Admin



Provider



User



CHAPTER 5

CODING

5.1 Features of Language

The coding step is the process that transfer design into programming language. It translates a detail design representation of software into a programming language realization.

Python Overview

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python is dynamically typed, and garbage collected. It supports multiple programming Paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

MySQL

MySQL is the world's most popular open-source database software, with over 100 million copies of its software downloaded or distributed throughout its history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

The MySQL database is owned, developed and supported by Sun Microsystems, one of the world's largest contributors to open source software. MySQL was originally founded and developed in Sweden by two Swedes and a Finn: David Axmark, Allan Larsson and Michael "Monty" Widenius, who had worked together since the 1980's. More historical information on MySQL is available on Wikipedia

- The best and the most-used database in the world for online applications
- Available and affordable for all
- Easy to use
- Continuously improved while remaining fast, secure and reliable
- Fun to use and improve
- Free from bugs

5.2 Functional Description

Palitive code

```
from django.shortcuts import render,redirect

import datetime

from datetime import date

from django.core.files.storage import FileSystemStorage

from django.db.models import Q

from .models import user as usr,login as log,Provider as stf,Complaint as comp,Feedback as
fed,catgerory as cate,servicerequest as ser,payment as pay


# Create your views here.

def index(request):

    msg=request.GET.get("msg","")

    if(request.session.get('role', ' ')=="admin"):

        response = redirect('/admin')

        return response

    elif(request.session.get('role', ' ')=="provider"):

        response = redirect('/staff')

        return response

    elif(request.session.get('role', ' ')=="user"):

        response = redirect('/user')

        return response

    else:

        return render(request,"index.html",{"msg":msg})

def admin(request):

    if(request.session.get('role', ' ')=="admin"):

        return render(request,"adminhome.html")
```



```

else:

    response = redirect('/index'+"?msg=session expired login again")

    return response

def user(request):

    if(request.session.get('role', ' ')=="user"):

        return render(request,"userhome.html")

    else:

        response = redirect('/index'+"?msg=session expired login again")

        return response

def staff(request):

    if(request.session.get('role', ' ')=="provider"):

        return render(request,"providerhome.html")

    else:

        response = redirect('/index'+"?msg=session expired login again")

        return response

def User_Reg(request):

    return render(request,"UserReg.html")

def register_user(request):

    msg=""

    if request.POST:

        t1 = request.POST["t1"]

        t2 = request.POST["t2"]

        t3 = request.POST["t3"]

        t4 = request.POST["t4"]

        t5 = request.POST["t5"]

```

```

t6=request.POST["t6"]

t7 = request.POST["t7"]

t8 = request.POST["t8"]

log.objects.create(username=t7, password=t8, role="user")

data=log.objects.last()

usr.objects.create(User_name=t1,User_email=t2,User_phone=t3,User_location=t4,User_district=t5,User_address=t6, User_status="approved",Log_id=data)

msg="Registers successfully"

response = redirect('/User_Reg')

return response

else:

    msg = ""

data1=usr.objects.all()

return render(request,"UserReg.html",{"msg":msg,"data":data1})

def Provider_Reg(request):

    msg=""

    data1=cate.objects.all()

    return render(request,"ProviderReg.html",{"msg":msg,"data":data1})

def register_provider(request):

    msg=""

    if request.POST:

        t1 = request.POST["t1"]

        t2 = request.POST["t2"]

        t3 = request.POST["t3"]

        t4 = request.POST["t4"]

```

```

t5 = request.POST["t5"]

t6=request.POST["t6"]

tf2 = request.FILES["tf1"]

t7 = request.POST["t7"]

t8 = request.POST["t8"]

t11 = request.POST["t11"]

t12 = request.POST["t12"]

log.objects.create(username=t7, password=t8, role="")

data=log.objects.last()


stf.objects.create(Provide_name=t1,Provide_email=t2,Provide_phone=t3,Provide_location=t4
,Provide_district=t5,Provide_address=t6, Provide_status="waiting",
Provide_tharif=t12,Provide_logid=data,catgerory_id=t11,Provide_doc=tf2)

msg="Registers successfuley"

response = redirect('/Provider_Reg')

return response

else:

    msg = ""

data1=usr.objects.all()

return render(request,"ProviderReg.html",{"msg":msg,"data":data1})


def Login(request):

    if request.POST:

        user = request.POST["username"]

        password = request.POST["password"]

        try:

            data = log.objects.get(username=user, password=password)

```

```

if (data.role == "admin"):

    request.session['username'] = data.username

    request.session['role'] = data.role

    request.session['id'] = data.logid

    response = redirect('/admin')

    return response

elif (data.role == "provider"):

    request.session['username'] = data.username

    request.session['role'] = data.role

    request.session['id'] = data.logid

    response = redirect('/staff')

    return response

elif(data.role == "user"):

    request.session['username'] = data.username

    request.session['role'] = data.role

    request.session['id'] = data.logid

    response = redirect('/user')

    return response

else:

    return render(request, "index.html", {"msg": "invalid account Details"})

except:

    return render(request, "index.html", {"msg": "invalid user name or password"})

else:

    response = redirect('/index')

    return response

def Profile(request):

```

```

msg=""

ids=request.session["id"]

if request.POST:

    if request.session["role"]=="provider":

        t2 = request.POST["t2"]

        t3 = request.POST["t3"]

        t4 = request.POST["t4"]

stf.objects.filter(Provide_logid=ids).update(Provide_address=t2,Provide_email=t3,Provide_ph
one=t4)

    elif request.session["role"]=="user":

        t2 = request.POST["t2"]

        t3 = request.POST["t3"]

        t4 = request.POST["t4"]

        usr.objects.filter(Log_id=ids).update(User_address=t2,User_email=t3,User_phone=t4)


if request.session["role"]=="provider":

    data1=stf.objects.get(Provide_logid=ids)

    returnpage="StaffProfile.html"

elif request.session["role"]=="user":

    data1=usr.objects.get(Log_id=ids)

    returnpage="UserProfile.html"

else:

    response = redirect('/index'+"?msg=session expired login again")

    return response

return render(request,returnpage,{"msg":msg,"data":data1})

```

```

def complaints(request):
    msg=""

    #datax=usr.objects.get(Log_id=request.session["id"])

    if request.POST:

        t1 = request.POST["t1"]

        t2 = request.POST["t2"]


        msg="updated sucessfully"

        comp.objects.filter(Complaint_id=t1).update(Complaint_reply=t2)

    data1=comp.objects.all()

    return render(request, "Answer_Queries.html",{"msg":msg,"data":data1})


def Feedback(request):
    msg=""

    #datax=usr.objects.get(Log_id=request.session["id"])

    if request.POST:

        t1 = request.POST["t1"]

        t2 = request.POST["t2"]


        msg="updated sucessfully"

        fed.objects.filter(Feedback_id=t1).update(Feedback_reply=t2)

    data1=fed.objects.all()

    return render(request, "Answer_Queries1.html",{"msg":msg,"data":data1})


def UserFeedback(request):
    msg=""

    d=request.session['id']

    data1=log.objects.get(logid=d)

```

```

datau=usr.objects.get(Log_id=data1)

#datax=usr.objects.get(Log_id=request.session["id"])

if request.POST:

    ids=request.session["id"]

    t1 = request.POST["t1"]

    t2 = request.POST["t2"]

    t3 = "2022-02-22"

    msg="updated sucessfully"

fed.objects.create(Feedback_subject=t1,Feedback_message=t2,Feedback_date=t3,Feedback_
reply="not yet Seen",user=datau)

data1=fed.objects.filter(user_id=datau).all()

return render(request, "UserFeedback.html",{"msg":msg,"data":data1})

def Logout(request):

    try:

        del request.session['id']

        del request.session['role']

        del request.session['username']

        response = redirect("/index?id=logout")

        return response

    except:

        response = redirect("/index?id=logout")

        return response

def All_Users(request):

    msg=""

    data=usr.objects.all()

```

```

    return render(request, "All_Users.html", {"msg": msg, "data": data})

def remove_usr(request):

    usr.objects.filter(Log_id=request.GET["id"]).delete()

    log.objects.filter(logid=request.GET["id"]).delete()

    response = redirect('/All_Users')

    return response

def Privacy(request):

    msg=""

    if request.POST:

        t1=request.POST["t1"]

        t2=request.POST["t2"]

        id=request.session["id"]

        data=log.objects.get(logid=id)

        if data.password==t1:

            msg="sucessfully updated"

            log.objects.filter(logid=id).update(password=t2)

        else:

            msg="invalid current password"

    returnpage="adminhead.html"

    if(request.session.get('role', ' ')=="provider"):

        returnpage="providerhead.html"

    elif(request.session.get('role', ' ')=="user"):

        returnpage="userhead.html"

    return render(request, "privacy.html", {"role":returnpage, "msg":msg})

def delete_staff(request):

```



```

stf.objects.filter(Provide_logid=request.GET["id"]).delete()

log.objects.filter(logid=request.GET["id"]).delete()

response = redirect('/List_Staff')

return response

def List_Staff(request):

    msg = ""

    data1=stf.objects.all()

    return render(request,"List_staff.html",{"msg":msg,"data":data1})

def approve_provider(request):

    t2=request.GET["id"]

    #datax=usr.objects.get(Log_id=request.session["id"])

    msg="updated sucessfully"

    stf.objects.filter(Provide_logid=t2).update(Provide_status="approve")

    log.objects.filter(logid=t2).update(role="provider")

    data1=comp.objects.all()

    response = redirect('/List_Staff')

    return response


def add_category(request):

    msg=""

    data1=cate.objects.all()

    return render(request,"AddCategory.html",{"msg":msg,"data":data1})

def add_category_process(request):

    msg=""

    if request.POST:

        t1 = request.POST["t1"]

        t2 = request.POST["t2"]

```

```

    cate.objects.create(catgerory_name=t1,catgerory_address=t2)

    msg="Registers successfuley"

    response = redirect('/ViewCategory')

    return response

else:

    msg = ""

    data1=cate.objects.all()

    return render(request,"AddCategory.html",{"msg":msg,"data":data1})

def ViewCategory(request):

    msg = ""

    data1=cate.objects.all()

    return render(request,"ViewCategory.html",{"msg":msg,"data":data1})

def delete_category(request):

    cate.objects.filter(catgerory_id=request.GET["id"]).delete()

    response = redirect('/List_Staff')

    return response


def ViewServices(request):

    msg = ""

    data1=ser.objects.all()

    return render(request,"ViewServices.html",{"msg":msg,"data":data1})


def ViewPayments(request):

    msg = ""

    id=request.GET["id"]

```

```

data1=pay.objects.filter(servicerequest=id).all()

return render(request,"ViewPayment.html",{"msg":msg,"data":data1})

def RequestService(request):

    msg = ""

    data1=stf.objects.all()

    return render(request,"RequestService.html",{"msg":msg,"data":data1})

def AddRequest(request):

    msg=""

    d=request.session['id']

    data1=log.objects.get(logid=d)

    datau=usr.objects.get(Log_id=data1)

    #datax=usr.objects.get(Log_id=request.session["id"])

    if request.POST:

        ids=request.session["id"]

        t1 = request.POST["t1"]

        pl=log.objects.get(logid=t1)

        t2 = request.POST["t2"]

        p2=cate.objects.get(catgerory_id=t2)

        t3 = "2022-02-22"

        s1 = request.POST["s1"]

        s2 = request.POST["s2"]

        msg="Register sucessfully"

    ser.objects.create(service_type=s1,service_details=s2,reg_date=t3,request_date=t3,request_r
ating="4star",user=datau,request_status="pending",Provide_logid=pl,catgerory=p2)

    data1=ser.objects.filter(user_id=datau).all()

```

```

        return render(request, "ServiceHistory.html", {"msg": msg, "data": data1})

def ServiceHistory(request):
    msg = ""

    d=request.session['id']

    data1=log.objects.get(logid=d)

    datau=usr.objects.get(Log_id=data1)

    data1=ser.objects.filter(user_id=datau).all()

    return render(request, "ServiceHistory.html", {"msg": msg, "data": data1})

def ServicePayments(request):
    msg = ""

    id=request.GET["id"]

    dataser=ser.objects.get(request_id=id)

    data1=pay.objects.filter(servicerequest=dataser).all()

    return render(request, "ServicePayments.html", {"msg": msg, "data": data1})

def AddPayment(request):
    msg=""

    d=request.session['id']

    data1=log.objects.get(logid=d)

    datau=usr.objects.get(Log_id=data1)

    #datax=usr.objects.get(Log_id=request.session["id"])

    if request.POST:

        ids=request.session["id"]

        t1 = 22

```

```

t2 = request.POST["t1"]

pl=ser.objects.get(request_id=t2)

t3 = 2000

s1 = request.POST["s1"]

s2 = request.POST["s2"]

msg="Register sucessfully"

pay.objects.create(amount=t3,pay_status="Payment
Complete",servicerequest=pl,user=datau)

data1=pay.objects.filter(user_id=datau).all()

return render(request, "ServicePayments.html",{"msg":msg,"data":data1})

def ProviderComplaints(request):

    msg=""

    d=request.session['id']

    data1=log.objects.get(logid=d)

    datau=stf.objects.get(Provide_logid=data1)

    #datax=usr.objects.get(Log_id=request.session["id"])

    if request.POST:

        ids=request.session["id"]

        t1 = request.POST["t1"]

        t2 = request.POST["t2"]

        t3 = "2022-02-22"

        msg="updated sucessfully"

    comp.objects.create(Complaint_subject=t1,Complaint_message=t2,Complaint_date=t3,Compl
aint_reply="not yet Seen",Provider=datau)

    data1=comp.objects.filter(Provider=datau).all()

    return render(request, "ProviderComplaints.html",{"msg":msg,"data":data1})

```

```

def providerService(request):

    msg = ""

    d=request.session['id']

    datau=log.objects.get(logid=d)


    data1=ser.objects.filter(Provide_logid=datau).all()

    return render(request,"providerService.html",{"msg":msg,"data":data1})

def UpdateRequest(request):

    t2=request.GET["id"]

    #datax=usr.objects.get(Log_id=request.session["id"])

    msg="updated sucessfully"

    ser.objects.filter(request_id=t2).update(request_status="approve")

    d=request.session['id']

    datau=log.objects.get(logid=d)

    data1=ser.objects.filter(Provide_logid=datau).all()

    response = redirect('/providerService')

    return response

def RequestPayments(request):

    msg = ""

    id=request.GET["id"]

    dataser=ser.objects.get(request_id=id)

    data1=pay.objects.filter(servicerequest=dataser).all()

    return render(request,"RequestPayments.html",{"msg":msg,"data":data1})

```

CHAPTER 6

TESTING

Testing is vital to success of the system. System testing makes logical assumption that if all parts of the system are correct, the goal will be successfully achieved. Another reason for system testing is its utility as a user- oriented vehicle before implementation. System testing is aimed at ensuring that system works accurately and efficiently before live operation commences. Testing is vital to success of the system. A series of tests are performed for the proposed system before the system is ready for user acceptance testing.

6.1 Levels of Testing

Various Testing Methods

A Software Test Plan is a document describing the testing scope and activities. It is the basis for formally testing any software/product in a project A test is a set of data that the system will process as normal input. However, the data are created with the express intent of determining whether the system will process them correctly. There are two general strategies for testing software. Code Testing must test cases that result in executing every instruction in the program or a module; but is every path through the program is tested.

i) Unit Testing

In computer programming unit testing is a software verification and validation method in which a programmer tests if individual unit of source code are fit for use. The smallest testable part of an application in procedural programming in a unit may be an individual functions or procedure. Unit testing focus on the smallest unit of software design, the module. This is also known as “Module Testing”. The modules are tested separately. This testing is carried out during the programming stage itself. In this step each modules found to be working satisfactorily. The forms in this project are tested one by one and errors are corrected.

ii) Integration Testing

Data can be lost across an interface. One module can have an adverse effect on others. Sub-functions when combined may not produce the desired major functions. The objective is to take unit tested modules and to combine them and test it. In this step all errors encountered are corrected for next testing.

iii) User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance. Since the user interface of the system is very much user friendly the acceptance of the users was gained very easily.

iv) Output Testing

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in specific format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration.

The output format of the screen is found to be correct as the format was designed in the system design phase according to the user needs. For the hard copy also, output comes out as the specified requirements by the user. Hence output testing does not result in any correction in the system. Various reports are generated in graphical output format and being pictorial representation it is found more convenient to understand by the users of the system.

v) Validation Testing

In validation testing, the requirements established as a part of software requirements analysis are validate against the software that has been constructed. Validation testing provides final assurance that the software meets all functional, behavioral and performance requirements. After each validation test case has been done, one of the two possible conditions exists:

- The function or performance characteristics conform to specifications and are accepted.
- A deviation from specification is uncovered and a deficiency list is created.

CHAPTER 7

IMPLEMENTATION

7.1 Implementation of proposed system

The implementation is the final and important phase. It involves user training, system testing in order to ensure successful running of the proposed system. Once the system design phase is over, the next stage is to implement and monitors the operation of the system to ensure that it continues the work effectively and efficiently.

The three main phases in implementation take place in series. These are the initial installation, the test of the system as a whole and evaluation maintenance and control of the system. The implementation plan and action to implement should be bound closely together. The implementation plan is a function of line management at least as far as key decision or alternative plans are concerned.

The implementation plan was to convert the existing clerical files to the computer. The implementation plan listed all sub tasks so that individuals in the organization may be assigned specific responsibilities.

The installation of the new system that is bound to replace the current one may require a major revision of computer facilities as well as completely new after space. Space planning took into account the space occupied by the people, space by equipment and the movement of people and equipment in the working investment. After conduction the initial testing the system is loaded on the client office's computer. Some of the user employees in this case are selected. These users are trained first and they run the system. A detailed documentation is prepared to this set of employees. There may be slight modifications to meet the organization.

After all modifications specified by the users in the documentation are made, the computer system is run along with manual system. Even though this kind of parallel run make extra burden to the employees and management, the system is run in parallel for the sake of checking reliability and efficiency. After this document, which compares the result of the manual system with those of the computerized is prepared. If there is any modifications are made as needed.

A procedure is developed for delivering instructions and forms to supervisors for coordination and integrating the proposal with other parts of the organization, and for working out of problems with people involved. This procedure also helped for evaluation of hardware and software. A program was developed to emphasis the nature and goals of the new system on the management and the support personnel and train operation personnel in their new tasks.

In the case of management many of whom participated in the development of the system short seminars were given. Particular attention was paid to the training of end users. The training sessions were aimed at giving the user staff the specific skills required in their new jobs. They were given practical training to have a thorough understanding of what the new system is like and how it behaves.

Education involved creating the right atmosphere and motivation of user staff. It explained the need for changes and helped to overcome the resentment caused by the feeling that computers took away the responsibility from individual departments.

Various measures have been taken by department officials in order to find suitable solutions by the following issues:

- About the skill to be acquired.
- Reduction of manpower in department
- About the new form having all required option.

7.2 Installation Procedure

To install the system, the primary need is web-based environments without which the system will not have a proper utilization. To install the system, it is must to setup a centralized server which can hold social networking website including the user's information database. The database is accessed through web pages using browser at the client end. In order to have the server setup for the device information system, the following components are needed at the server end.

1. PHP

2. MySQL

3. A preferable operating system like Windows 7 or Windows 8

Copy all the required files to the web server root folder and create the database.

In the field of computer software, the term software build refers either to the process of converting source code file in to stand alone software artifacts that can be run on a computer, or the result of doing so.

CHAPTER 8

CONCLUSION

The new system has overcome most of the limitations of the existing system and works according to the design specification given. The developed systems dispense the problem and meet the needs of by providing reliable and comprehensive information. All the requirements projected by the user have been met by the system.

The newly developed system consumes less processing time, and all the details are updated and processed immediately. Since the screen provides online help messages and is very user-friendly, any user will get familiarized with its usage. Modules are designed to be highly flexible so that any failure requirements can be easily added to the modules without facing many problems.

CHAPTER 9

FUTURE ENHANCEMENT

Enhancement means adding, modifying or developing the code to support the changes in the specification. It is the process of adding new capabilities such as report, new interface without other systems and new features such as better screen or report layout.

Every module in the system is being developed carefully such that the future enhancements do not affect the basic performance of the system. In future we can add any links or services to the System very easily.

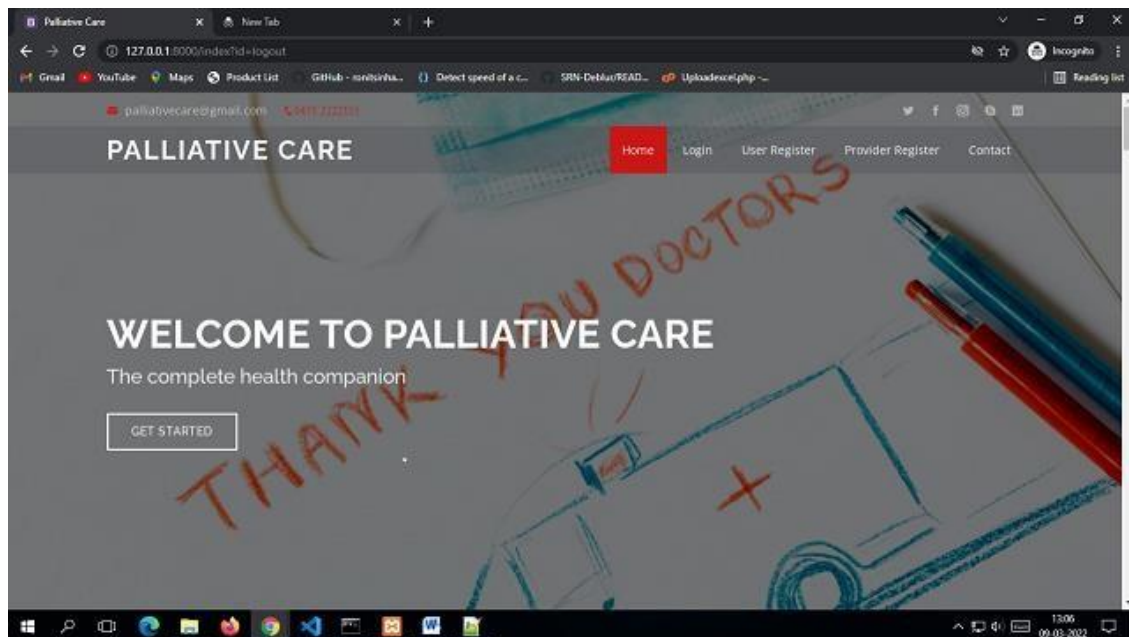
Moreover, due to limited time allotted for the project, there are features, which I couldn't implement. Thus, the system offers the scope of future enhancement.

As this software is reliable to use, any modification in accordance with the necessity of the user can be done for the future use. Any additional feature can be implemented very easily. So, what we call this software also a user friendly. Some of the future developments that can be incorporated in this software are:

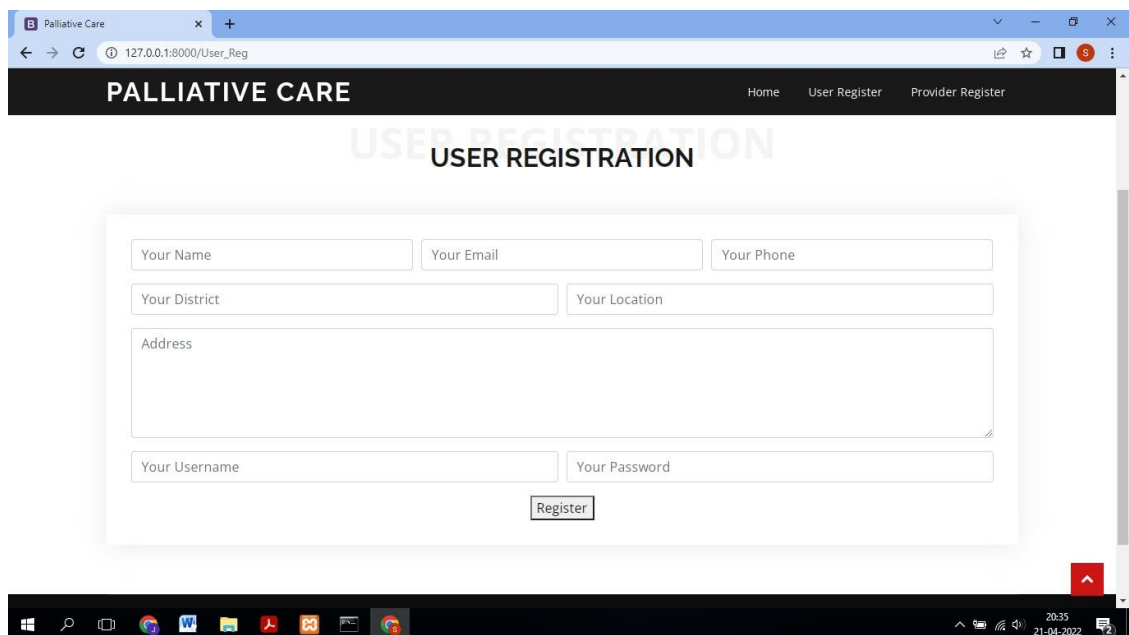
- The current system is only applicable in two, three and four-wheeler vehicles.in future we can implement this concept to heavy vehicles.
- In current system payment process is semi computerized we can implement it total computerized.
- This application is implementing all over the world.

SCREENSHOTS

PALLIATIVE CARE



USER REGISTRATION



PROVIDER REGISTRATION

PALLIATIVE CARE Home User Register **Provider Register**

PROVIDER REGISTRATION

VIEW SERVICE REQUEST

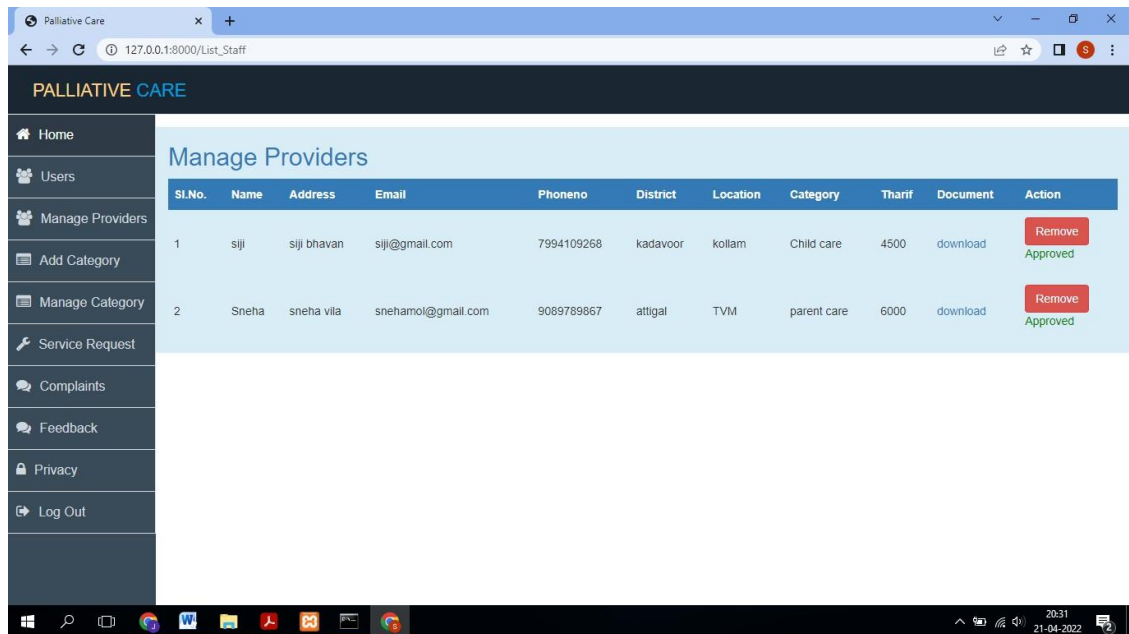
PALLIATIVE CARE

Home Users Manage Providers Add Category Manage Category Service Request Complaints Feedback Privacy Log Out

View Service Request

Sl.No.	Service Type	category	Details	Request Date	User Details	Provider Details	Status	Rating	Action
1	Child care	Child care	child care	2022-02-22	Helna		approve	4star	<input type="button" value="View Payments"/>
2	parent care	parent care	urgent	2022-02-22	Sona		approve	4star	<input type="button" value="View Payments"/>
3	Child care	Child care	ff	2022-02-22	Helna		approve	4star	<input type="button" value="View Payments"/>
4	Child care	Child care	jb kn	2022-02-22	Helna		pending	4star	<input type="button" value="View Payments"/>

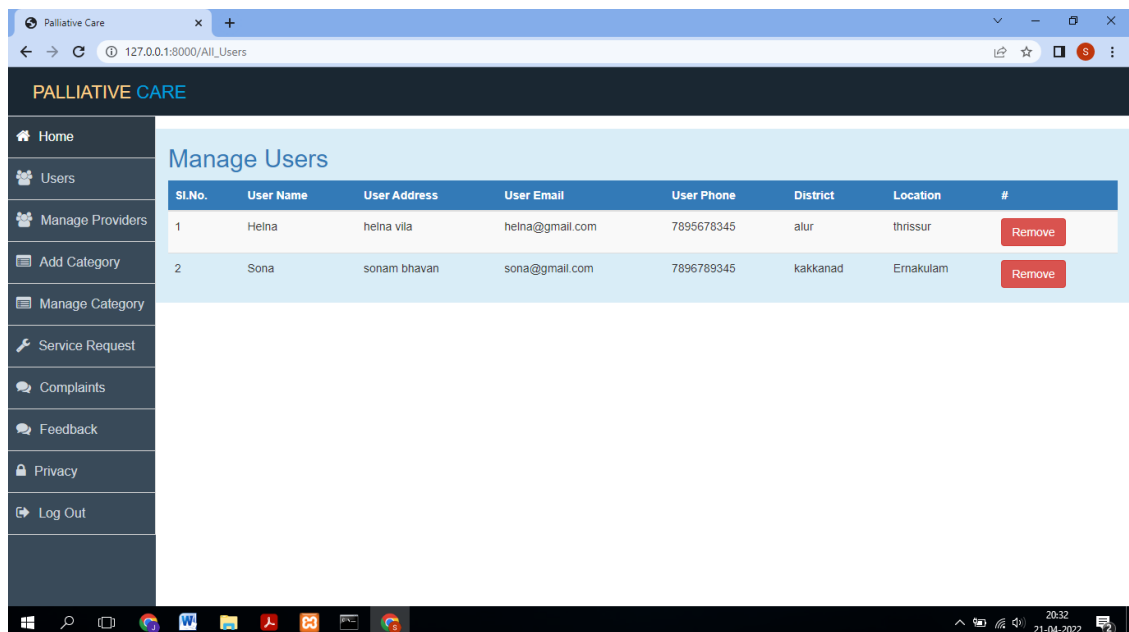
MANAGE PROVIDERS



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/List_Staff". The application title is "PALLIATIVE CARE". The sidebar on the left contains the following menu items: Home, Users, Manage Providers, Add Category, Manage Category, Service Request, Complaints, Feedback, Privacy, and Log Out. The main content area is titled "Manage Providers" and contains a table with the following data:

Sl.No.	Name	Address	Email	Phoneno	District	Location	Category	Tharif	Document	Action
1	siji	siji bhavan	siji@gmail.com	7994109268	kadavoor	kollam	Child care	4500	download	Remove Approved
2	Sneha	sneha vila	snehamol@gmail.com	9089789867	attigal	TVM	parent care	6000	download	Remove Approved

MANAGE USERS



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/All_Users". The application title is "PALLIATIVE CARE". The sidebar on the left contains the following menu items: Home, Users, Manage Providers, Add Category, Manage Category, Service Request, Complaints, Feedback, Privacy, and Log Out. The main content area is titled "Manage Users" and contains a table with the following data:

Sl.No.	User Name	User Address	User Email	User Phone	District	Location	#
1	Helna	helna vila	helna@gmail.com	7895678345	alur	thrissur	Remove
2	Sona	sonam bhavan	sona@gmail.com	7896789345	kakkanad	Ernakulam	Remove

SERVICE REQUEST

The screenshot shows a web browser window with the URL `127.0.0.1:8000/RequestService`. The page displays a 'Service Request' modal form. The form includes a 'Category' dropdown menu with 'Child care' selected, a 'Service Type' text input field, and a 'Service Details' text area. A green 'Submit' button is at the bottom left of the form, and a 'Close' button is at the bottom right. The background shows a sidebar with navigation links: Home, Request Service, Service History, Feedback, Profile, Privacy, and Log Out.

VIEW SERVICE REQUESTS

The screenshot shows a web browser window with the URL `127.0.0.1:8000/providerService`. The page displays a 'View Service Requests' table. The table has the following columns: S.No., Service Type, category, Details, Request Date, User Details, Provider Details, Request Status, Rating, and Action. The table contains three rows of data. The Action column contains buttons: 'View Payments' for the first two rows and 'Approve Request' and 'View Payments' for the third row.

S.No.	Service Type	category	Details	Request Date	User Details	Provider Details	Request Status	Rating	Action
1	Child care	Child care	child care	2022-02-22	Heina	siji	approve	4star	<button>View Payments</button>
2	Child care	Child care	ff	2022-02-22	Heina	siji	approve	4star	<button>View Payments</button>
3	Child care	Child care	fb km	2022-02-22	Heina	siji	pending	4star	<button>Approve Request</button> <button>View Payments</button>

REFERENCES

[1] Akshay Kumar¹, Pankaj Kumar Meena², Debiprasanna Panda³, Ms. Sangeetha⁴

CHATBOT IN PYTHON International Research Journal of
Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 06
Issue: 11 | Nov 2019

www.irjet.net p-ISSN: 2395-0072

[2] Sasa Arsovski 1*, Adrian David Cheok 2, Muniru Idris 3, Mohd Radzee
Bin Abdul Raffur4 ANALYSIS OF THE CHATBOT OPEN-SOURCE
LANGUAGES AIML AND CHATSCRIPT: A Review

[3] Jincy Susan Thomas¹, Seena THOMAS² CHATBOT USING GATED
END-TO-END MEMORY NETWORKS International Research Journal
of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume:
05 Issue: 03

| Mar-2018 www.irjet.net p-ISSN: 2395-0072

[4] Aafiya Shaikh¹, Dipti More², Ruchika Puttoo³, Sayli Shrivastav⁴,
Swati Shinde⁴ A SURVEY PAPER ON CHATBOTS International Research
Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN:2395-0072

[5] Mikic, Fernando A, Juan C Burguillo, Martín Llamas, Daniel A
Rodríguez, and Eduardo Rodríguez (2009). "Charlie: An aiml-based
chatterbot which works as an interface among machines and humans". In:
EAAEE Annual Conference, 2009. IEEE.

[6] Lue Lin, Luis Fdo. D'Haro, and Rafael Banchs. A Web based Platform
for Collection of Human Chatbot Interactions; Paper accepted in HAI
2016 to appear in Oct. 2016.

[7] Deryugina, OV (2010). "Chatterbots". In: Scientific and Technical
Information Processing 37.2, pp. 143–147.

