ABSTRACT

Medical Donation DataBase Management System is based on the concept of of helping poor people with medicines which they can’t afford. Most of the time we have observed that people are left with medicines which remain unused. These medicines are eventually thrown into the dustbin. As we know life is so important and not all people can afford heavy expenditure, the idea of Medical Donation DataBase Management System came into picture.

Medical Donation DataBase Management System allows people to donate unused medicines.

Further these medicines will be transported to a medicine acceptor centre which will send medicines to hospitals where people who can’t afford medicines can get it for free.

Since life is so busy and people want door-to-door service and sometimes people don’t get time to travel a long distance inorder to donate something, MedDonate gives a platform where there is no need for the donor to go anywhere to do charity. Collectors will be coming to their doors to take the respective medicines . They further take those medicines to the authentication centre which is the accepting centre for the medicines. The medicines which pass through all set of test for their purity are finally sent to the hospitals where it is distributed at a very cheap cost for the people for can’t afford medication.

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Medicine Donation DataBase System

Medicine Donation DataBase Management System introduces a platform where medicines can be donated for the cause of poor people to help them fight for their life.

The name to this DataBase Management System is given “MedDonate”. MedDonate makes it easy for people to donate medicines as a collector will be provided to collect medicine from their homes. Donors have to specify the details of the medicines on the website along with their home location. There is no need to go to a particular centre to donate medicines. Medicines will be sent to the accepting centres which accepts medicines and check for its validity and purity. If some fatal issue occurs regarding medicines ,Then these medicines will be rejected. The accepted medicines will be sent to hospitals where it can reach to people who are in need.

Authentication.

MedDonate offer complete authentication of the medicines. Even the list of donors,collectors and acceptors have to undergo through an efficient set of authentication rules. This is provided by password encryption,upload of necessary documents and user authentication.

Authentication is a very necessary part when it comes to medical field so that no negligence is shown as here it’s a matter of life.

Transportation.

Since life is so busy and people want door-to-door service and sometimes people don’t get time to travel a long distance inorder to donate something, MedDonate gives a platform where there is no need for the donor to go anywhere to do charity. Collectors will be coming to their doors to take the respective medicines . They further take those medicines to the authentication centre which is the accepting centre for the medicines. The medicines which pass through all set of test for their purity are finally sent to the hospitals where it is distributed at a very cheap cost for the people for can’t afford medication.

Multiple User Handling

When several users share the data, integrating the administration of data can offer major improvements. Experienced professionals understand the nature of the data being managed and can be responsible for organizing the data representation to reduce redundancy and make the data to retrieve efficiently

The users include:

1. Donor
2. Collector
3. Acceptor

Chapter 2

LITERATURE SURVEY

2.1 Traditional File System

In the early days of computing, data management and storage was a very new concept for organizations. The traditional approach to data handling offered a lot of the convenience of the manual approach to business processes (e.g. handwritten invoices & account statements, etc.) as well as the benefits of storing data electronically.

The traditional approach usually consisted of custom built data processes and computer information systems tailored for a specific business function. An accounting department would have their own information system tailored to their needs, where the sales department would have an entirely separate system for their needs.

Initially, these separate systems were very simple to set up as they mostly mirrored the business process that departments had been doing for years but allowed them to do things faster with less work. However, once the systems were in use for so long, they became very difficult for individual departments to manage and rely on their data because there was no reliable system in place to enforce data standards or management.

Separate information systems for each business function also led to conflicts of interest within the company. Departments felt a great deal of ownership for the data that they collected, processed, and managed which caused many issues among company-wide collaboration and data sharing. This separation of data also led to unnecessary redundancy and a high rate of unreliable and inconsistent data

2.2 Pros and Cons of a Traditional File System

Pros

- Simple
    - Matched existing business processes and functions.
    - Companies were not as interested in funding complicated information systems.

- Initially low-cost
    - Early computing was not viewed as beneficial for large funding.
    - Systems were designed to be cheap in order to save on cost.

Cons

- Separated ownership
    - Business functions had a high sense of data ownership.
    - Departments unwilling to share data for fear of minimizing their superiority.
- Unmanaged redundancy
    - Multiple instances of the same data appeared throughout various files, systems, and databases.
    - Information updated in one place was not replicated to the other locations.
    - Disk space was very expensive, and redundancy had a big impact on storage.
- Data inconsistency
    - Redundant data stored in various locations was usually never stored the same way.
    - Formatting was not centrally managed.
- Lack of data sharing
    - Same data stored in multiple locations.
    - Caused unnecessary doubling of efforts for processing and managing data.
- High costs in the long run
    - Hiring data processors for each department was very expensive, and each position was typically working on the same thing just for a different area.
    - Doubling of work as well as excessive maintenance costs.

2.3 Downfall of Traditional Management System

Conceived in a relatively centralized era when software was deployed in static environments, legacy database architectures fail to support an increasingly mobile world where applications are accessed anytime, anywhere. Today software users want consistent improvements in usability and expect SaaS vendors to deliver new features and functionalities needed to achieve their business objectives.

However, legacy database technologies fall short in serving the needs of today's distributed and cloud environment for the following reasons:

- Inadequate failover capabilities.
- Latency issues.
- Insufficient provisions during peak demands.
- Lack of high availability at all times.
- Increasing operational costs.
- Inability to meet the demands of global markets.

For all of these reasons, traditional databases are unable to deliver results in a rapidly growing environment where the workload is geographically distributed across heterogeneous data centers. Upgrading to a more distributed data model is costly and complicated and your DBAs can't just sit back and give up on this situation. Hence, due to these various reasons, the downfall of the traditional system was inevitable.

2.4 Introduction to the Database Management System

A database management system (DBMS) refers to the technology for creating and managing databases. Basically, a DBMS is a software tool to organize (create, retrieve, update and manage) data in a database.

The main aim of a DBMS is to supply a way to store and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database. A datum is a unit of Data. Meaningful data combines to form Information. Hence, information is interpreted data- data provided with semantics.MS ACCESS is one of the most common examples of database management software.

Database systems are meant to handle large collection of information. Management of data involves both defining structures for storage of information and providing mechanisms that can do the manipulation of those stored information. Moreover, the database system must ensure the safety of the information stored, despite system crash or attempts at unauthorized access.

2.5 Indicative Areas for the use of a DBMS

- Airlines: Reservations, schedules etc.
- Telecom: Calls made, customer details, network usage etc.
- Universities: Registration, results, grades, etc.
- Sales: Products, purchases, customers etc.
- Banking: All transactions etc.

2.6 Advantages of a DBMS

A Database Management System has many advantages over the traditional file system used in the earlier days:

- Data independence: Application programs should be as free or independent as possible from details of data representation and storage. DBMS can supply an abstract view of the data for insulating application code from such facts.

- Efficient data access: DBMS utilize a mixture of sophisticated concepts and techniques for storing and retrieving data competently and this feature becomes important in cases where the data is stored on external storage devices.

- Data integrity and Security: If data is accessed through the DBMS , the DBMS can enforce integrity constraint on the data.

- Data administration: When several users share the data, integrating the administration of data can offer major improvements. Experienced professionals understand the nature of the data being managed and can be responsible for organizing the data representation to reduce redundancy and make the data to retrieve efficiently.

2.7 Components of a DBMS

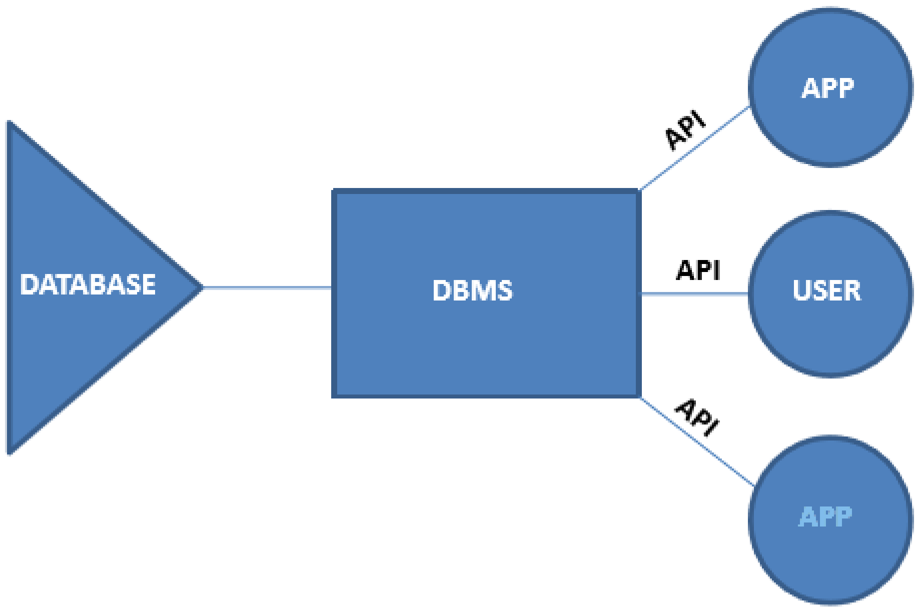


Fig 2.1 Components of a DBMS

- **Users**: Users may be of any kind, such as data base administrators, system developers or database users.
- **Database application**: Database application may be Departmental, Personal, Organizational and /or Internal
- **DBMS**: Software that allows users to create and manipulate database access.
- **Database**: Collection of logical data as a single unit.

Chapter 3

System Requirements

3.1 Hardware requirements

- Processor: Intel Core 2 Duo or above.
- RAM: 2GB or more.
- Hard Disk: 2GB or more.

3.2 Software requirements

Technologies used:

- Front End: HTML, CSS , JSS
- Connection/Controller: Django
- Back-End Database: SQLite3

Software:

- Text Editor: Atom Text Editor
- Server: Django
- Operating System: Windows 10,Linux
- Database Support: SQLite3
- Back-End: Django

Chapter 4

System Design

4.1 Schema Diagram

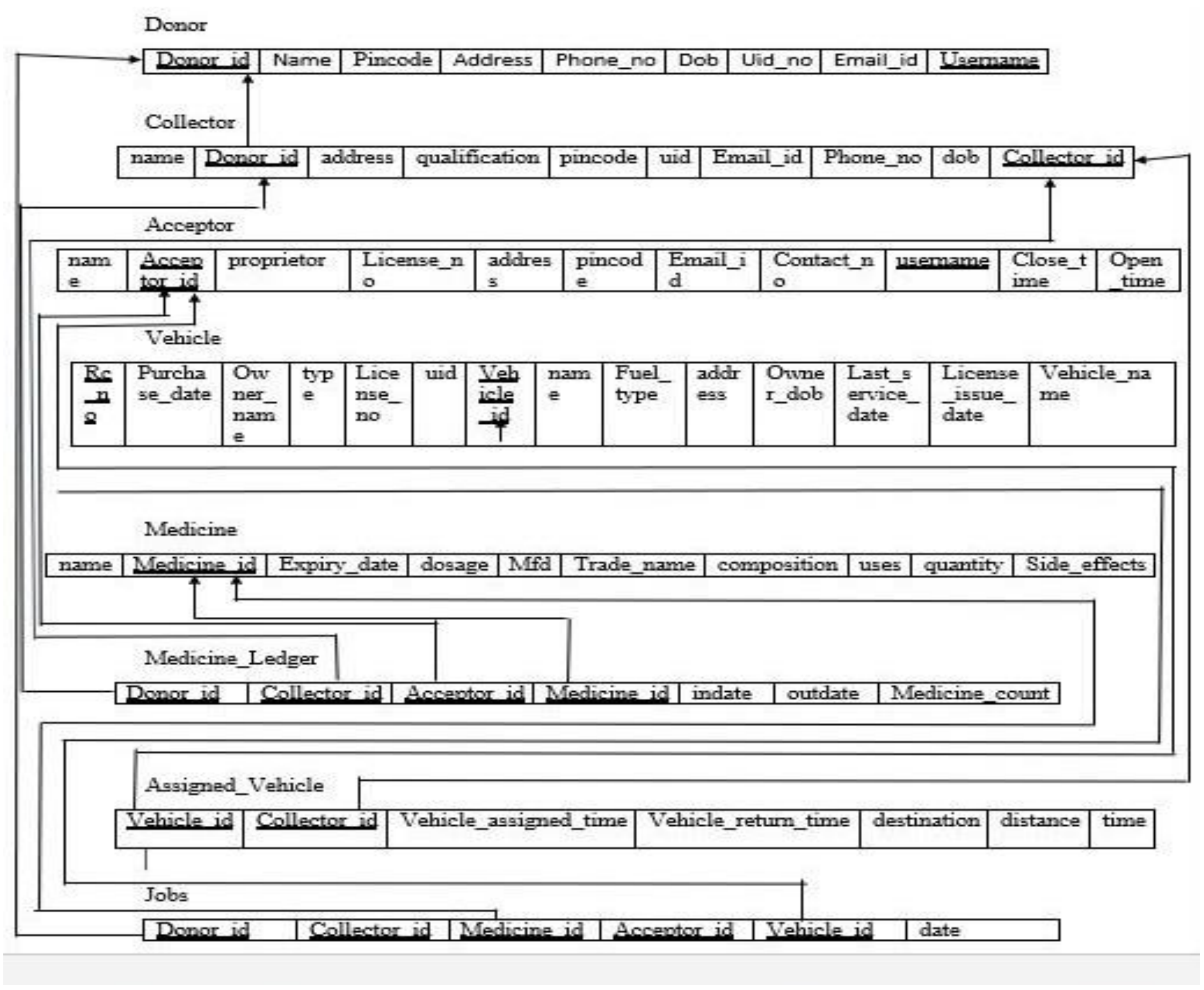

Donor
Donor_id | Name | Pincode | Address | Phone_no | Dob | Uid_no | Email_id | Username
Collector
name | Donor_id | address | qualification | pincode | uid | Email_id | Phone_no | dob | Collector_id
Acceptor
name | Acceptor_id | proprietor | License_no | address | pincode | Email_id | Contact_no | username | Close_time | Open_time
Vehicle
Rc_no | Purchase_date | Owner_name | type | License_no | uid | Vehicle_id | name | Fuel_type | address | Owner_dob | Last_service_date | License_issue_date | Vehicle_name
Medicine
name | Medicine_id | Expiry_date | dosage | Mfd | Trade_name | composition | uses | quantity | Side_effects
Medicine_Ledger
Donor_id | Collector_id | Acceptor_id | Medicine_id | indate | outdate | Medicine_count
Assigned_Vehicle
Vehicle_id | Collector_id | Vehicle_assigned_time | Vehicle_return_time | destination | distance | time
Jobs
Donor_id | Collector_id | Medicine_id | Acceptor_id | Vehicle_id | date

Fig 4.1 Schema diagram for Medicine Donation Management System

This shows the tables at a glance along with the attributes signifying the primary keys and the foreign keys present in the database of the MedDonate. Here we have 8 tables namely Donor, Collector, Acceptor, Vehicles, Assigned Vehicle, Medicine, Medical Ledger , Job, and their attributes.

4.2 ER Diagram

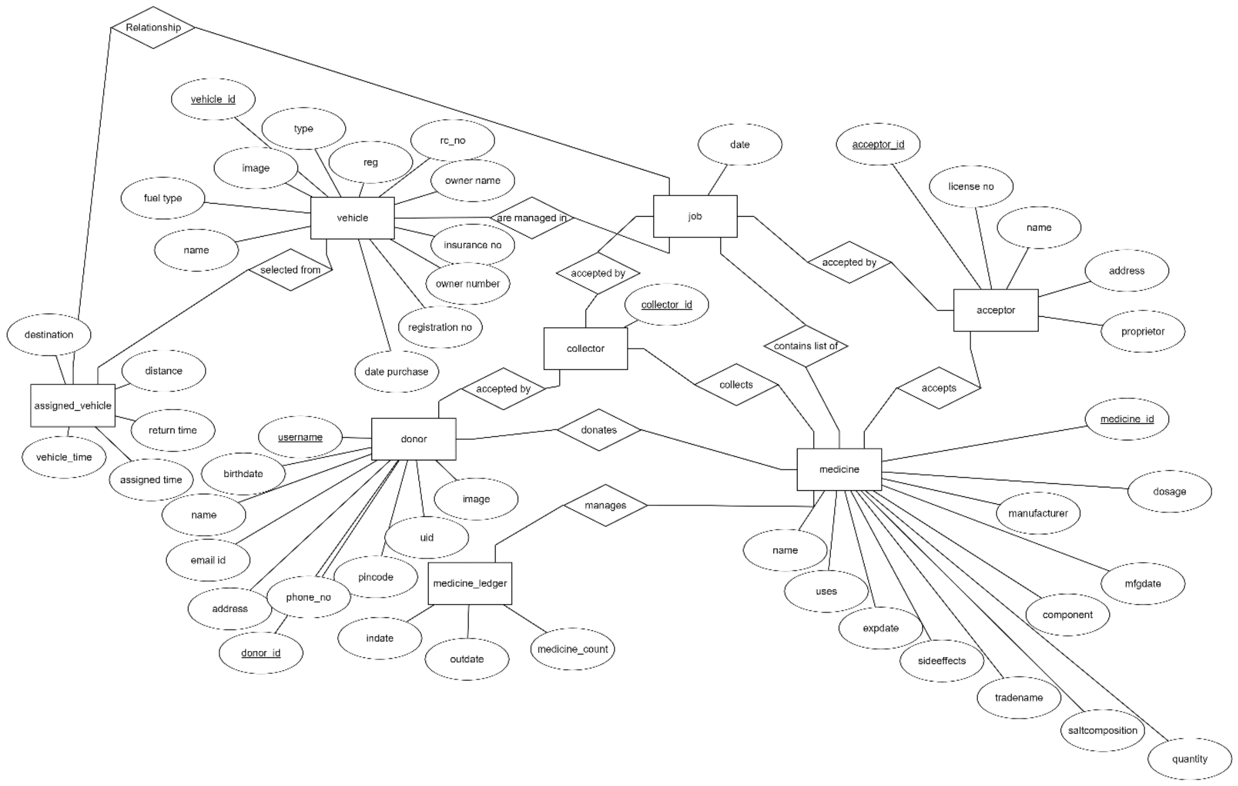


Fig 4.2 ER diagram for Medicine DataBase Management System

This ER diagram of Medical Donation Management System gives a brief idea of its existent entities and the relationships between them and the attributes associated with each entity.

Here we have 8 tables namely Donor, Collector, Acceptor, Vehicles, Assigned Vehicle, Medicine, Medical Ledger , Job, and their attributes. We could clearly see how the entities actually behave in the database.

Chapter 5

IMPLEMENTATION

5.1 HTML5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current major version of the HTML standard. It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers,etc. HTML5 is intended to subsume not only HTML4,but also XHTML1 and DOM Level 2 HTML.

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

Many new syntactic features are included. To natively include and handle multimedia and graphical content, the new <video>, <audio> and <canvas> elements were added, and support for scalable vector graphics (SVG) content and MathML for mathematical formulas. To enrich the semantic content of documents, new page structure elements such as <main>, <section>, <article>, <header>, <footer>, <aside>, <nav> and <figure>, are added. New attributes are introduced, some elements and attributes have been removed, and others such as <a>, <cite> and<menu> have been changed, redefined or standardized.

The APIs and Document Object Model (DOM) are now fundamental parts of the HTML5 specification and HTML5 also better defines the processing for, any invalid documents.

5.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML5. CSS is a cornerstone technology of the world wide web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural format.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech based browser or screen reader), and on Braille based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC2318 (march 1998). The W3C operates a free CSS validation service for CSS documents.

In addition to HTML, other mark-up languages support the use of CSS, including XHTML, plain XML, SVG, and XUL.

5.3 PYTHON

Python is an interpreted, 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 supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed..

5.4 SQLite3

SQLite is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.[5]

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.[6] SQLite has bindings to many programming language

SQLite uses an unusual type system for an SQL-compatible DBMS; instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values; in language terms it is *dynamically typed*. Moreover, it is *weakly typed* in some of the same ways that Perl is: one can insert a string into an integer column (although SQLite will try to convert the string to an integer first, if the column's preferred type is integer). This adds flexibility to columns, especially when bound to a dynamically typed scripting language. However, the technique is not portable to other SQL products. A common criticism is that SQLite's type system lacks the data integrity mechanism provided by statically typed columns in other products. The SQLite web site describes a "strict affinity" mode, but this feature has not yet been added.[11] However, it can be implemented with constraints like CHECK(typeof(x)='integer').[12] Tables normally include a hidden *rowid* index column which gives faster access.[17] If a database includes an Integer Primary Key column SQLite will typically optimize it by treating it as an alias for *rowid*, causing the contents to be stored as a strictly typed 64-bit signed integer and changing its behavior to be somewhat like an auto-incrementing column. Future versions of SQLite may include a command to introspect whether a column has behavior like that of *rowid* to differentiate these columns from weakly-typed, non-autoincrementing Integer Primary Keys.

5.5 Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Because Django was developed in a fast-paced newsroom environment, it was designed to make common Web-development tasks fast and easy. Here’s an informal overview of how to write a database-driven Web app with Django.

Like the technical community as a whole, the Django team and community is made up of a mixture of professionals and volunteers from all over the world, working on every aspect of the mission - including mentorship, teaching, and connecting people.

Diversity is one of our huge strengths, but it can also lead to communication issues and unhappiness. To that end, we have a few ground rules that we ask people to adhere to. This code applies equally to founders, mentors and those seeking help and guidance.

This isn’t an exhaustive list of things that you can’t do. Rather, take it in the spirit in which it’s intended - a guide to make it easier to enrich all of us and the technical communities in which we participate.

This code of conduct applies to all spaces managed by the Django project or Django Software Foundation. This includes IRC, the mailing lists, the issue tracker, DSF events, and any other forums created by the project team which the community uses for communication. In addition, violations of this code outside these spaces may affect a person's ability to participate within them.

5.5 Code snippets

5.5.1 admin.py

```
from django.contrib import admin

from .models import *

# Register your models here.

admin.site.register(Doner)

admin.site.register(Collector)

admin.site.register(Acceptor)

admin.site.register(vehicle)

admin.site.register(assigned_vehicle)

admin.site.register(medicine)

admin.site.register(medicine_ledger)

admin.site.register(job)
```

5.5.2 urls.py

```
from django.urls import path,include

from . import views

from django.conf import settings

from django.conf.urls.static import static

from django.contrib.auth.views import LoginView,LogoutView

urlpatterns =[


path("", views.index, name="index"),
path("collector-login", LoginView.as_view(template_name="MedicalDonation/Collector-login.html"), name="Collectorlogin"),

path("acceptor-login",LoginView.as_view(template_name="MedicalDonation/Acceptor-login.html"), name="Acceptorlogin"),

path("donor-login", LoginView.as_view(template_name="MedicalDonation/Donor-login.html"), name="Donorlogin"),

path("logout", LogoutView.as_view, name="logout"),

path("collector", views.collectors, name= "Collector_view"),

path("donor", views.donors, name= "Donor_view"),

path("acceptor", views.acceptors, name= "Acceptor_view"),

path("medicines", views.medicines, name= "Medicine_view"),

path("job",views.jobs, name="job_view"),



path("acceptor-home/<str:username>", views.acceptor_dash, name="acceptorhome"),

path("collector-home/<str:username>", views.collector_dash, name="collectorhome"),

path("donor-home/<str:username>", views.donor_dash, name="donorhome"),
path("add-collector", views.collector_add, name="Add_Collector"),

path("add-donor", views.donor_add, name="add_donor"),

path("add-acceptor", views.acceptor_add, name="add_acceptor"),

path("medicine-add", views.medicine_add, name="add_medicine"),



path("create_Collector", views.create_Collector),

path("create_Doner", views.create_Doner),

path("create_Acceptor", views.create_Acceptor),

path("create_Medicine", views.create_Medicine),

]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

5.5.3 models.py

```
from django.db import models

from datetime import timedelta,datetime

from .encrption import *

# Create your models here.

class Doner(models.Model):

    name = models.CharField(max_length=64)

    address = models.CharField(max_length=128,unique=True)

    pinCode = models.IntegerField()

    Phone_no = models.IntegerField()

    BirthDate = models.DateField()

    UID = models.IntegerField(unique=True)

    email = models.EmailField(max_length=64)

    username = models.CharField(max_length=16,unique=True)

    #password = models.CharField(max_length=16)

    image = models.ImageField(upload_to='doner')



    def __str__(self):

        return (f"{self.name} {self.username} {self.Phone_no}")


class Collector(models.Model):

    #to store a Image file


    name = models.CharField(max_length=64)

    address = models.CharField(max_length=128)
    pinCode = models.IntegerField()
    Phone_no = models.IntegerField()
    BirthDate = models.DateField()
    UID = models.IntegerField(unique=True)
    email = models.EmailField(max_length=64)
    username = models.CharField(max_length=16,unique=True)
    #password = models.CharField(max_length=16)
    qualification = models.CharField(max_length=64)
    Driving_License = models.CharField(max_length=32)
    Driving_License_image = models.ImageField(upload_to='photos/collector/DL')
    image = models.ImageField(upload_to='collector/image')

    def __str__(self):
        return (f"{self.name} {self.username} {self.Phone_no}")

class Acceptor(models.Model):
    #to store a Image file
    name = models.CharField(max_length=64)
    proprietor = models.CharField(max_length=64)
    license_no = models.CharField(max_length=64)
    address = models.CharField(max_length=128)
    pincode = models.PositiveIntegerField()
    UID = models.IntegerField(unique=True)
    start_time = models.TimeField(auto_now=True)
    end_time = models.TimeField(auto_now=True)
    email = models.EmailField()
    username = models.CharField(max_length=64, unique=True)
    #password = models.CharField(max_length=16)
    Phone_no = models.IntegerField()
    image = models.ImageField(upload_to='acceptor')



    def __str__(self):
        return (f"{self.name} {self.username} {self.Phone_no}")

class vehicle(models.Model):
    type = models.CharField(max_length=32)
    name = models.CharField(max_length=32)
    RC_no =models.CharField(max_length=32)
    registration_no = models.CharField(max_length=16)
    last_servicing = models.DateField()
    fuel_type = models.CharField(max_length=32)
    insurence_no = models.CharField(max_length=32)
    owner_name = models.CharField(max_length=32)
    Date_purchase = models.DateField()
    owner_number = models.IntegerField()



    def __str__(self):
        return (f"{self.type} {self.name}")

class assigned_vehicle(models.Model):
    vehicle_id = models.ForeignKey(vehicle,blank=True,on_delete=models.CASCADE)
    collector_id = models.ForeignKey(Collector,blank=True,on_delete=models.CASCADE)
    vehicle_assigned_time = models.DateTimeField(auto_now=True)

    vehicle_return_time = models.DateTimeField(auto_now=True)

    destination                                                                    =
models.ForeignKey(Doner,to_field='address',blank=True,on_delete=models.CASCADE)

    distance = models.FloatField()


    def __str__(self):

        return(f"{self.vehicle_id} {self.collector_id}")


class medicine(models.Model):

    name = models.CharField(max_length=32)

    mfgdate = models.DateField(auto_now=True)

    expdate = models.DateField(auto_now=True)

    manufacturer = models.CharField(max_length=64)

    tradename = models.CharField(max_length=64)

    composition = models.CharField(max_length=64)

    uses = models.CharField(max_length=64)

    quantity = models.PositiveIntegerField()

    sideeffect = models.CharField(max_length=64)

    saltcomposition = models.CharField(max_length=64)

    dosage = models.CharField(max_length=64)


    def __str__(self):

        return (f"{self.name} {self.tradename}")



class medicine_ledger(models.Model):
    donor_id = models.ForeignKey(Doner,blank=True,on_delete=models.CASCADE)
    collector_id = models.ForeignKey(Collector,blank=True, on_delete=models.CASCADE)
    medicine_id = models.ForeignKey(medicine,blank=True, on_delete=models.CASCADE)
    acceptor_id = models.ForeignKey(Acceptor,blank=True, on_delete=models.CASCADE)
    indate = models.DateTimeField(default=datetime.now(tz=None))
    outdate = models.DateTimeField(default=datetime.now(tz=None))
    medicine_count = models.PositiveIntegerField(default=1)



    def __str__(self):
        return (f"{self.medicine_id} {self.medicine_count} {self.indate} {self.outdate}")



class job(models.Model):
    donor_id = models.ForeignKey(Doner,blank=True,on_delete=models.CASCADE)
    collector_id = models.ForeignKey(Collector,blank=True, on_delete=models.CASCADE)
    acceptor_id = models.ForeignKey(Acceptor,blank=True, on_delete=models.CASCADE)
    medicine_id = models.ForeignKey(medicine,blank=True, on_delete=models.CASCADE)
    vehicle_id = models.ForeignKey(vehicle,blank=True,on_delete=models.CASCADE)
    date = models.DateTimeField(default=datetime.now(tz=None))

    def __str__(self):
        return(f"{self.date} {self.donor_id} {self.collector_id} {self.acceptor_id} {self.medicine_id} {self.vehicle_id}")
```

5.5.3 views.py

```
from django.shortcuts import render

from django.http import HttpResponse, Http404, HttpResponseRedirect

from django.urls import reverse

from django.contrib.auth.decorators import login_required

from .models import *

from django.views.decorators.csrf import csrf_protect

from django.contrib.auth.models import User,Group

from django.contrib.auth import login,logout,authenticate

from django.core.files.images import ImageFile

from django.core.files.storage import FileSystemStorage




def index(request):

    context= {


    }

    return render(request,"MedicalDonation/homepage.html",context)




def acceptors(request):

    context = {

    "acceptors": Acceptor.objects.all()
    }

    return render(request, "MedicalDonation/acceptor.html",context)



def collectors(request):
    context = {
    "collectors": Collector.objects.all()
    }

    return render(request, "MedicalDonation/collector.html",context)
def donors(request):
    context = {
    "donors": Doner.objects.all()
    }

    return render(request, "MedicalDonation/donor.html",context)

def medicines(request):
    context = {
    "medicines": medicine.objects.all()
    }

    return render(request, "MedicalDonation/medicines.html",context)

def jobs(request):
    context = {
```

```
    "jobs": job.objects.all()
    }

    return render(request, "MedicalDonation/job.html",context)



def collector_add(request):
    context = {
    "collectors": Collector.objects.all()
    }

    return render(request, "MedicalDonation/collector-add.html",context)




def donor_add(request):
    context = {
    "donors": Doner.objects.all()
        }

    return render(request, "MedicalDonation/donor1.html",context)



def acceptor_add(request):
    context ={
    "acceptor" : Acceptor.objects.all()
    }
    return render(request, "MedicalDonation/acceptor-add.html",context)


def medicine_add(request):
    context ={
    "medicines" : medicine.objects.all()
    }
    return render(request, "MedicalDonation/medicine-add.html",context)




def create_Collector(request):
    if request.POST:
            coll = Collector(name=request.POST['name'], address= (request.POST['address']+
request.POST['address3'] + request.POST['address4']), pinCode=request.POST['pincode'],
Phone_no=request.POST['phone_no'],BirthDate=request.POST['birth'],
UID=request.POST['uid'],
email=request.POST['email'],username=request.POST['username'],
image=request.FILES['image'],Driving_License_image=request.FILES['Driving_License_im
age'],qualification=request.POST['qualification'],Driving_License=request.POST['Driving_li
cense'])
            username=request.POST['username']
            image=request.FILES['image']
            Driving_License_image=request.FILES['Driving_License_image']
            fs = FileSystemStorage()
            filename = fs.save(username, image)
            uploaded_file_url = fs.url(filename)
            fs = FileSystemStorage()
            filename = fs.save(username, Driving_License_image)
            uploaded_file_url = fs.url(filename)
            coll.save()
            password=request.POST['password']
```

```
        user = User.objects.create_user(username=request.POST['username'],email=request.POST['email'],password=request.POST['password'])
        user.last_name = ' '
        group = Group.objects.get(name="Collector")
        group.user_set.add(user)
        user.save()
    return HttpResponseRedirect(reverse("Add_Collector"))



def create_Doner(request):
    if request.POST:
        don = Doner(name=request.POST['name'], address= (request.POST['address1'] + request.POST['address3']), pinCode=request.POST['pinCode'], Phone_no=request.POST['Phone_no'],BirthDate=request.POST['birth'], UID=request.POST['uid'], email=request.POST['email'],username=request.POST['username'], image=request.FILES['image'] )
        don.save()
        username=request.POST['username']
        password=request.POST['password']
        image=request.FILES['image']
        fs = FileSystemStorage()
        filename = fs.save(username, image)
        uploaded_file_url = fs.url(filename)
        user = User.objects.create_user(username=request.POST['username'],email=request.POST['email'],password=request.POST['password'])
        user.last_name = ' '
        group = Group.objects.get(name="Donor")
```

```
            group.user_set.add(user)

            user.save()

    return HttpResponseRedirect(reverse("add_donor"))





def create_Acceptor(request):

    if request.POST:

            accep = Acceptor(name=request.POST['name'],proprietor=request.POST['proprietor'],
license_no=request.POST['license_no'],Phone_no=request.POST['Phone_no'],start_time=req
uest.POST['start_time'],end_time=request.POST['end_time'],                address=
(request.POST['address1']+                request.POST['address3']                +
request.POST['address4']),pincode=request.POST['pinCode'],UID=request.POST['UID'],emai
l=request.POST['email'],username=request.POST['username'],image=request.FILES['License
_image'])

            accep.save()

            username=request.POST['username']

            image=request.FILES['License_image']

            password=request.POST['password']

            fs = FileSystemStorage()

            filename = fs.save(username, image)

            uploaded_file_url = fs.url(filename)

            user                                                                    =
User.objects.create_user(username=request.POST['username'],email=request.POST['email'],p
assword=request.POST['password'])

            group = Group.objects.get(name="Acceptor")

            group.user_set.add(user)

            user.save()

    return HttpResponseRedirect(reverse("add_acceptor"))
```

```
def create_Medicine(request):
    if request.POST:
        med = medicine(name=request.POST['name'],mfgdate=request.POST['manufacturing_date'],expdate=request.POST['expiry_date'],manufacturer=request.POST['manufacturer'],tradename=request.POST['trade_name'],composition=request.POST['chemical_composition'],uses=request.POST['uses'],quantity=request.POST['quantity'],sideeffect=request.POST['side_effects'],saltcomposition=request.POST['salt_composition'],dosage=request.POST['dosage'])
        med.save()
    return HttpResponseRedirect(reverse("add_medicine"))



def login(request):
    context = {
    "collectors": Collector.objects.all()
    }
    return render(request, "MedicalDonation/login.html",context)
@login_required
def acceptor_dash(request,usernam):
    context ={
    "acceptor" : Acceptor.objects.filter(username=usernam),
    "collector":Collector.objects.all()
    }
    return render(request, "MedicalDonation/acceptor_home.html",context)


@login_required
def collector_dash(request,username):
    context ={
```

```
    "acceptor" : Acceptor.objects.all(),
    "donor" : Doner.objects.all(),
    "collector":Collector.objects.filter(username=username)
    }
    return render(request, "MedicalDonation/collector_home.html",context)


def donor_dash(request,usernam):
    context ={
    "donor" : Acceptor.objects.filter(username=usernam),
    "collector":Collector.objects.all()
    }
    return render(request, "MedicalDonation/donor_home.html",context)


def acceptor_login(request):
     username = request.POST['username']
     password = request.POST['password']
     user = authenticate(request, username=username, password=password)
     if user is not None:
         login(request, user)
         #acceptor_dash(request,username)
     else:
        return render(request, "MedicalDonation/failure.html",{
```

5.5.6 Stored Procedure snippet

The following lines of code serve as Stored Procedure in Django Framework.

```
def medicine_add(request):
    context ={
    "medicines" : medicine.objects.all()
    }
    return render(request, "MedicalDonation/medicine-add.html",context)
```

5.5.7 Trigger snippet

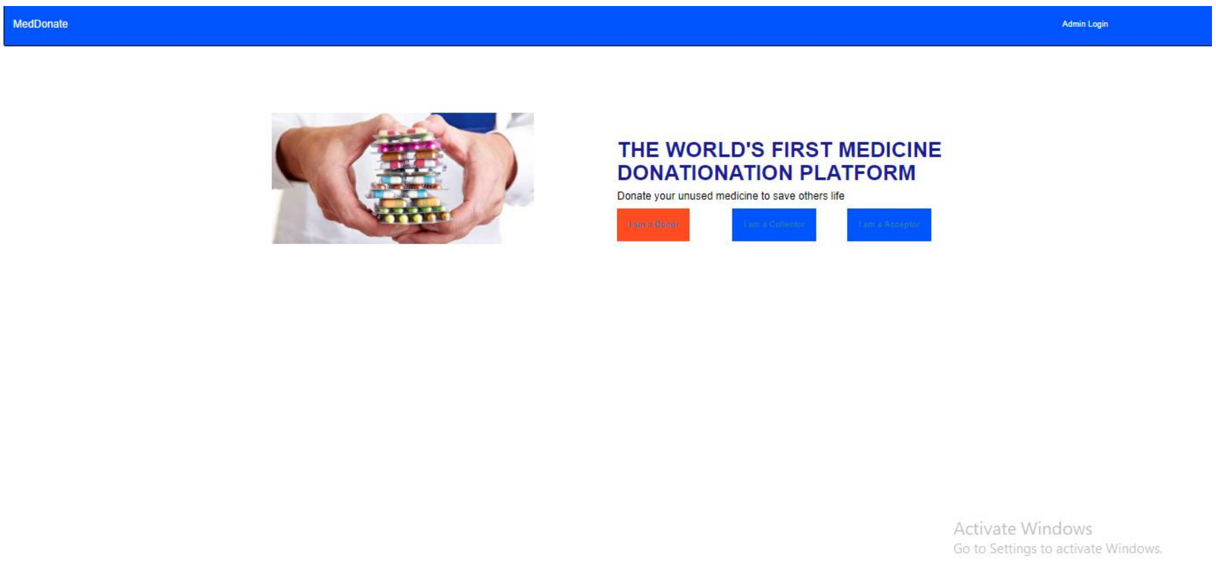
```
CREATE TRIGGER `login` AFTER INSERT ON `login-success`
 FOR EACH ROW INSERT into logs VALUES
(New.BreakName,CURRENT_TIMESTAMP);
```

‘login trigger is created on the ‘users table. When we enter the proper values in the login page and submit, the values ‘username and ‘time’ are updated in the ‘users table. While this triggers and the subsequent ‘time’ value is updated in the ‘logs’ table.

Chapter 6

SNAPSHOTS

6.1 Home Page


MedDonate
Admin Login
THE WORLD'S FIRST MEDICINE DONATIONATION PLATFORM
Donate your unused medicine to save others life
I am a Doner
I am a Collector
I am a Acceptor
Activate Windows
Go to Settings to activate Windows.

This page provides the face for the Medical Donation Management System. It gives a brief description of MedDonate and allows users to go to login or registration page.

MedDonate gives a platform where there is no need for the donor to go anywhere to do charity. Collectors will be coming to their doors to take the respective medicines . They further take those medicines to the authentication centre which is the accepting centre for the medicines. The medicines which pass through all set of tests for their purity are finally sent to the hospitals where it is distributed at a very cheap cost for the people for can’t afford medication

6.2 Registration Page

6.2.1 Donor Registration

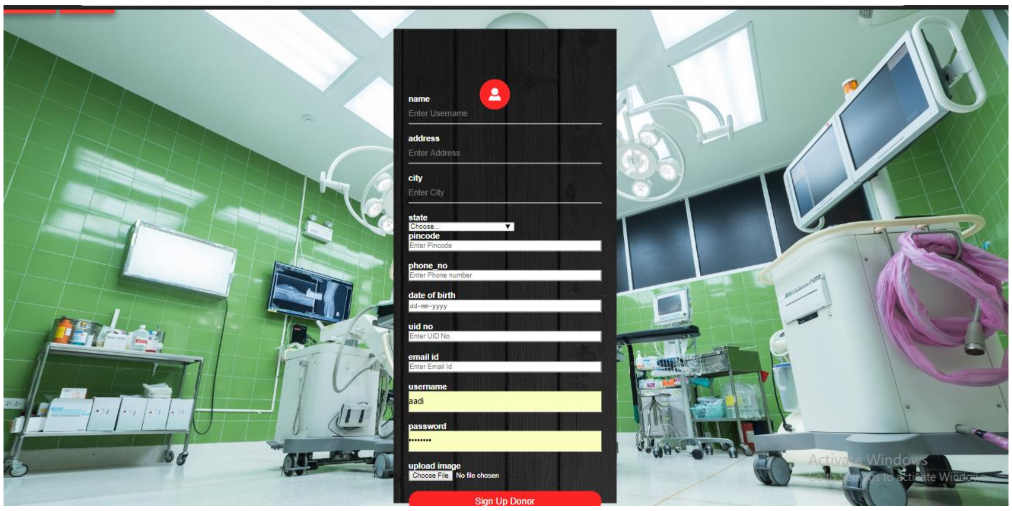


Fig 6.2.1

This is where the Donors can register to our platform, and join our therapeutic community!

6.2.2 Collector Registration

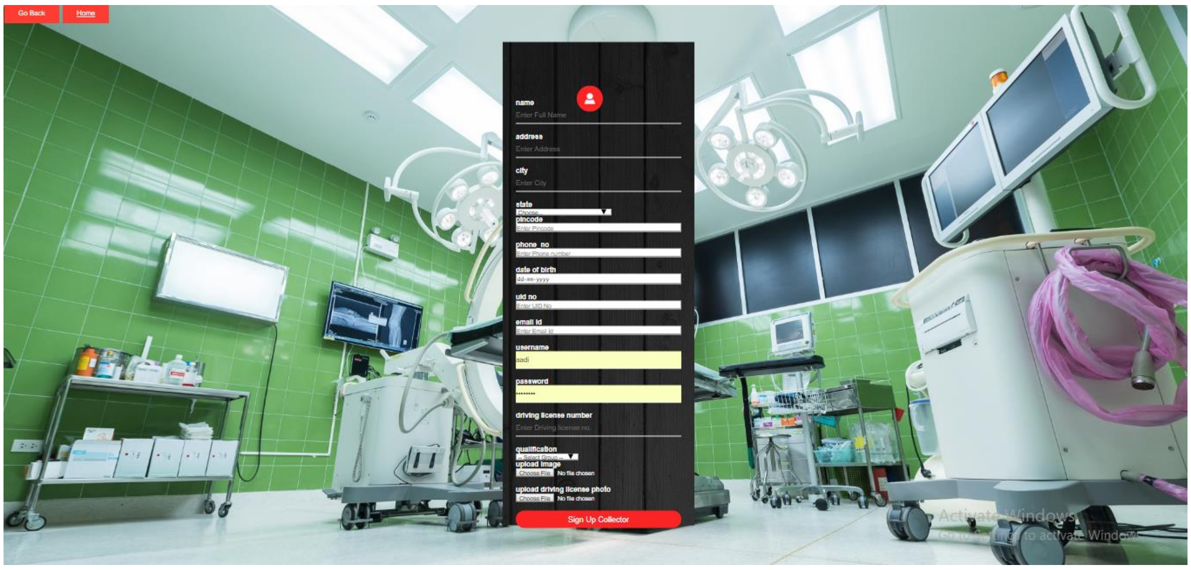
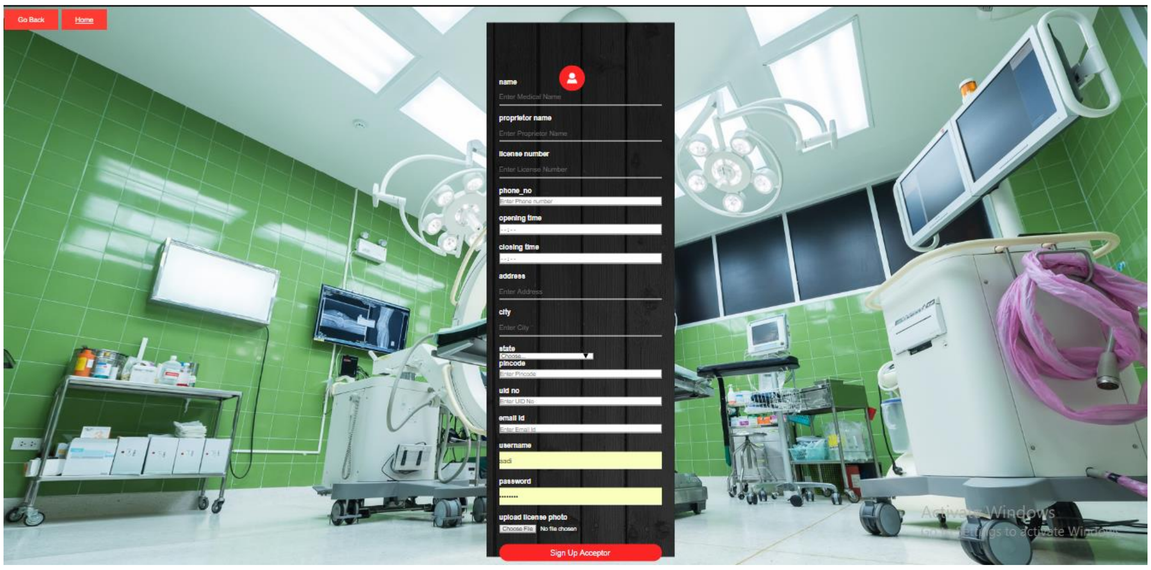


Fig 6.2.2

This is where the Collector can register to our platform, and join our therapeutic community!

6.2.3 Acceptor Registration

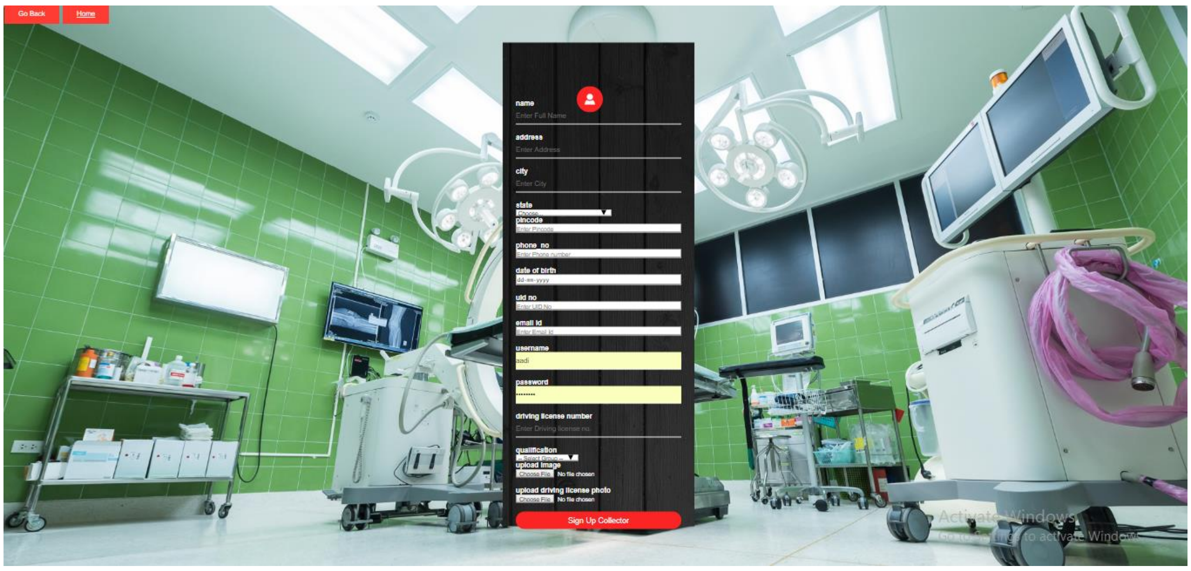


name
Enter Medical Name
proprietor name
Enter Proprietor Name
license number
Enter License Number
phone_no
Enter Phone number
opening time
--:--
closing time
--:--
address
Enter Address
city
Enter City
state
Choose...
pincode
Enter Pincode
uid no
Enter UID No
email id
Enter Email Id
username
sadi
password
••••••••
upload license photo

Fig 6.2.3

This is where the Acceptor can register to our platform, and join our therapeutic community!

6.2.4 Medicine Registration



name
Enter Full Name
address
Enter Address
city
Enter City
state
Choose...
pincode
Enter Pincode
phone_no
Enter Phone number
date of birth
dd-mm-yyyy
uid no
Enter UID No
email id
Enter Email Id
username
sadi
password
••••••••
driving license number
Enter Driving license no.
qualification
-- Select Group --
upload image
upload driving license photo

Fig 6.2.4

This is where the Donors can add medicine to our portal.

6.3 Login Page

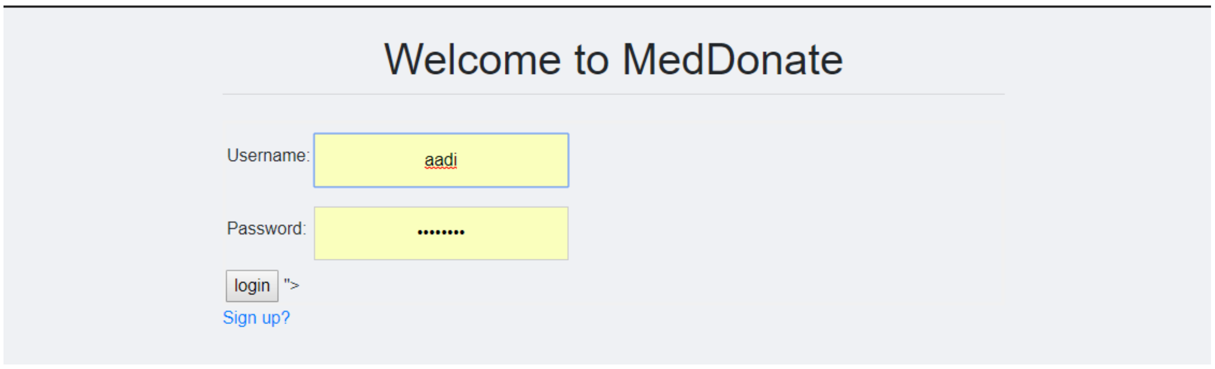


Fig 6.3

This provides the users to login either to visit their profile and view their work stats or to donate a medicine.

The same login page is used to login in all types of users, namely-

1.Donor

2.Collector

3.Acceptor

The login page performs the authentication and redirects the user if the login is a success.

If the login credentials are incorrect an error is displayed.

The Admin can log in to all the three types of login so as to ensure the functionality is proper.

The page ensures that templates is used to three different purposes simultaneously.

6.4 The Home Pages

6.4.1 Donor Homepage

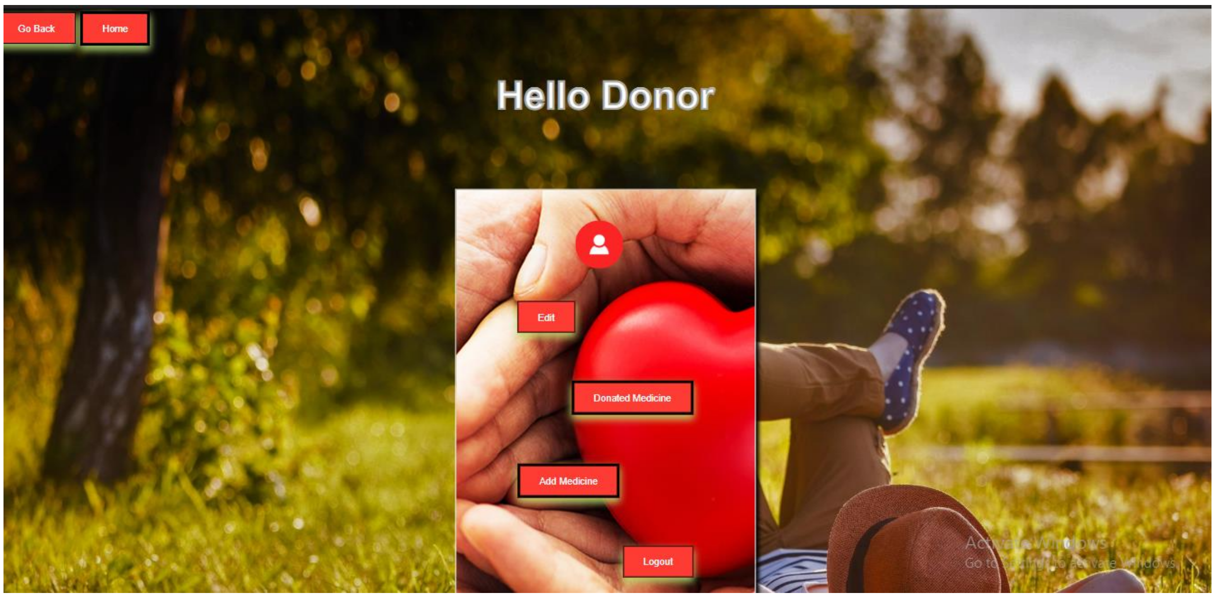


Fig 6.4.1

This is the homepage for the Donor after successful login.

6.4.2 Collector Homepage

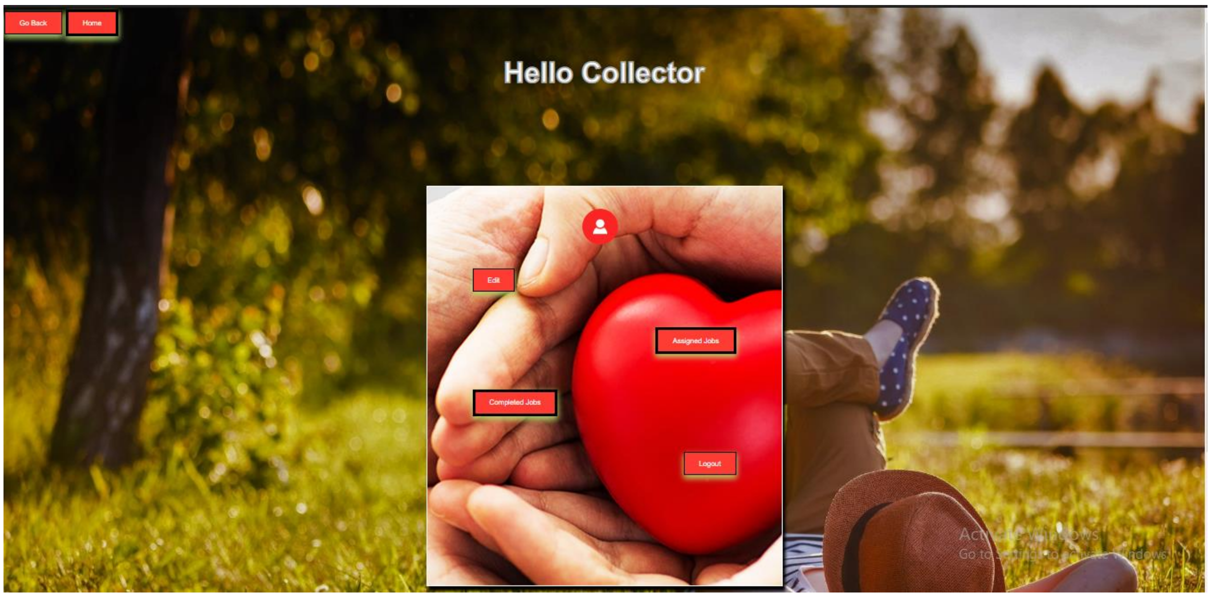


Fig 6.4.2

This is the homepage for the Collector after successful login

6.4.3 Acceptor Homepage

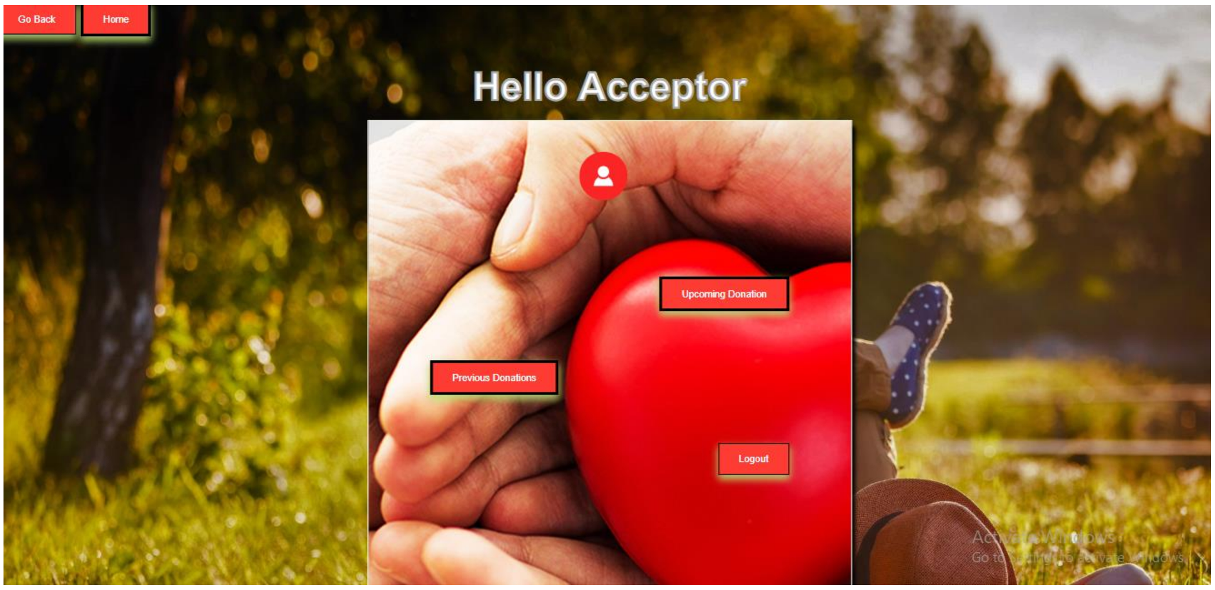
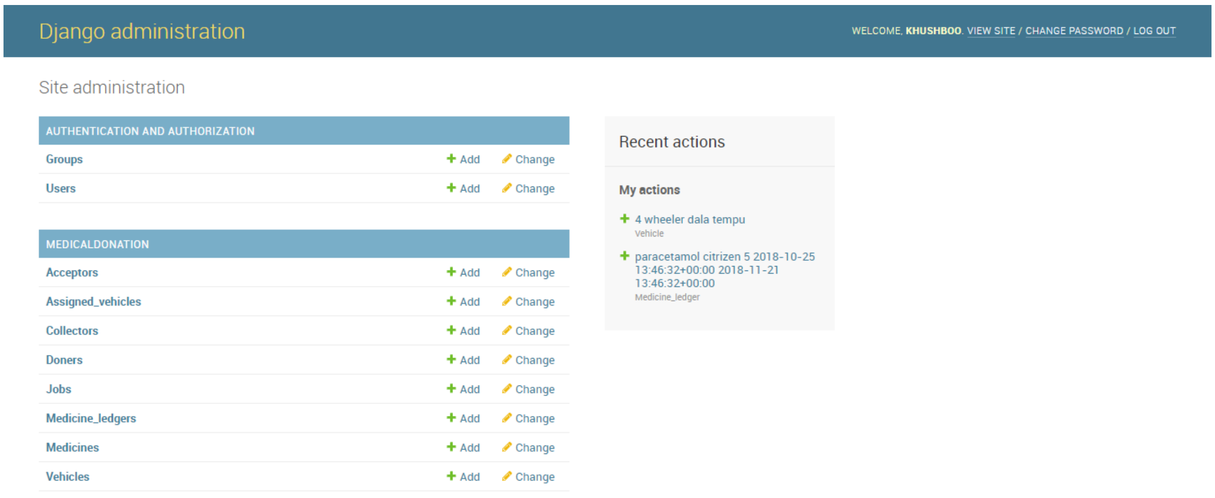


Fig 6.4.3

This is the homepage for the Acceptor after successful login.

6.5 Admin Page


Django administration

WELCOME, **KHUSHBOO**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

MEDICALDONATION		
Acceptors	+ Add	Change
Assigned_vehicles	+ Add	Change
Collectors	+ Add	Change
Doners	+ Add	Change
Jobs	+ Add	Change
Medicine_ledgers	+ Add	Change
Medicines	+ Add	Change
Vehicles	+ Add	Change

Recent actions

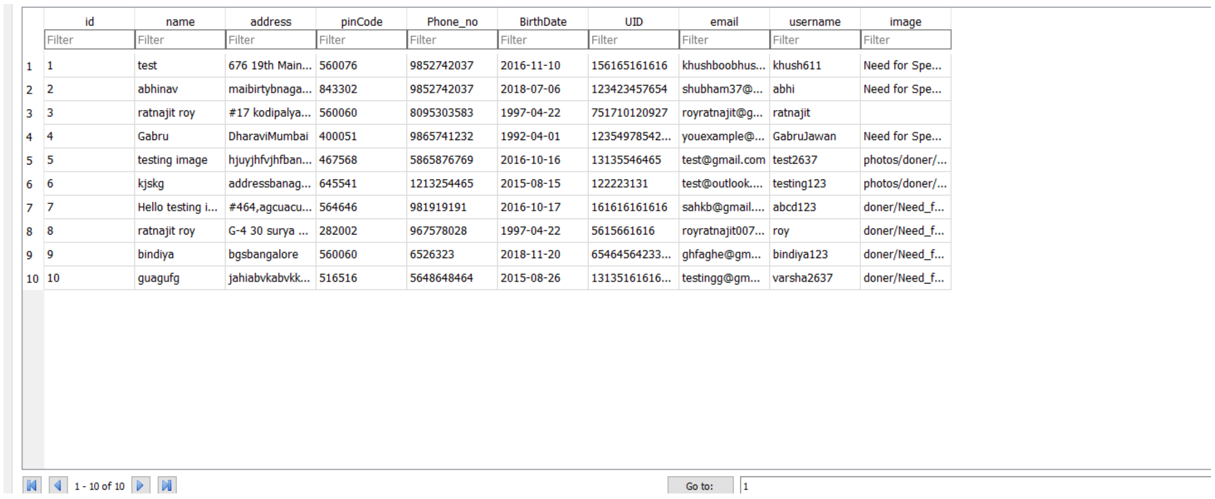
My actions

+ 4 wheeler dala tempu
  Vehicle
+ paracetamol citrizen 5 2018-10-25 13:46:32+00:00 2018-11-21 13:46:32+00:00
  Medicine_ledger

Fig 6.5

The admin handles everything from assigning vehicles to creating jobs.

6.6 Tables

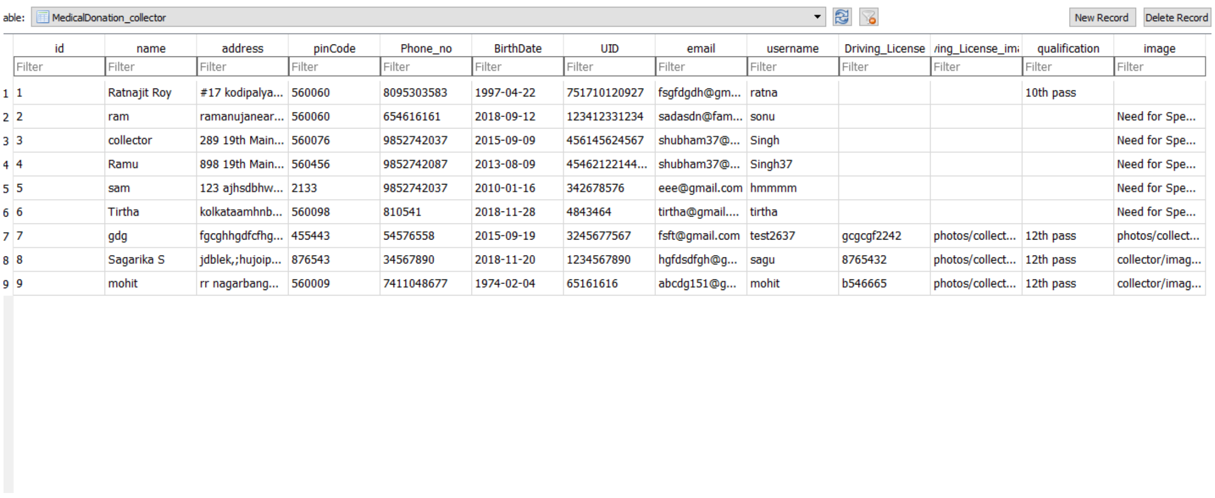


	id	name	address	pinCode	Phone_no	BirthDate	UID	email	username	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	test	676 19th Main...	560076	9852742037	2016-11-10	156165161616	khushboobhus...	khush611	Need for Spe...
2	2	abhinav	maibirtybnaga...	843302	9852742037	2018-07-06	123423457654	shubham37@...	abhi	Need for Spe...
3	3	ratnajit roy	#17 kodipalya...	560060	8095303583	1997-04-22	751710120927	royratnajit@g...	ratnajit	
4	4	Gabru	DharaviMumbai	400051	9865741232	1992-04-01	12354978542...	youexample@...	GabruJawan	Need for Spe...
5	5	testing image	hjuyjhfvjhfban...	467568	5865876769	2016-10-16	13135546465	test@gmail.com	test2637	photos/doner/...
6	6	kjskg	addressbanag...	645541	1213254465	2015-08-15	122223131	test@outlook....	testing123	photos/doner/...
7	7	Hello testing i...	#464,agcuacu...	564646	981919191	2016-10-17	161616161616	sahkb@gmail....	abcd123	doner/Need_f...
8	8	ratnajit roy	G-4 30 surya ...	282002	967578028	1997-04-22	5615661616	royratnajit007...	roy	doner/Need_f...
9	9	bindiya	bgsbangalore	560060	6526323	2018-11-20	65464564233...	ghfaghe@gm...	bindiya123	doner/Need_f...
10	10	guagufg	jahiabvkabvkk...	516516	5648648464	2015-08-26	13135161616...	testingg@gm...	varsha2637	doner/Need_f...

1 - 10 of 10 Go to: 1

Table 6-1 Donor Table

The above table shows the values in the Donor Table.



able: MedicalDonation_collector New Record Delete Record

	id	name	address	pinCode	Phone_no	BirthDate	UID	email	username	Driving_License	ving_License_im	qualification	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Ratnajit Roy	#17 kodipalya...	560060	8095303583	1997-04-22	751710120927	fsgfdgdh@gm...	ratna			10th pass	
2	2	ram	ramanujanear...	560060	654616161	2018-09-12	123412331234	sadasdn@fam...	sonu				Need for Spe...
3	3	collector	289 19th Main...	560076	9852742037	2015-09-09	456145624567	shubham37@...	Singh				Need for Spe...
4	4	Ramu	898 19th Main...	560456	9852742087	2013-08-09	45462122144...	shubham37@...	Singh37				Need for Spe...
5	5	sam	123 ajhsdbhw...	2133	9852742037	2010-01-16	342678576	eee@gmail.com	hmmmm				Need for Spe...
6	6	Tirtha	kolkataamhnb...	560098	810541	2018-11-28	4843464	tirtha@gmail....	tirtha				Need for Spe...
7	7	gdg	fgcghhgdfcfhg...	455443	54576558	2015-09-19	3245677567	fsft@gmail.com	test2637	gcgcgf2242	photos/collect...	12th pass	photos/collect...
8	8	Sagarika S	jdblek,;hujoip...	876543	34567890	2018-11-20	1234567890	hgfdsdfgh@g...	sagu	8765432	photos/collect...	12th pass	collector/imag...
9	9	mohit	rr nagarbang...	560009	7411048677	1974-02-04	65161616	abcdg151@g...	mohit	b546665	photos/collect...	12th pass	collector/imag...

Table 6-2 Collector Table

The above table shows the values in the Collector Table.

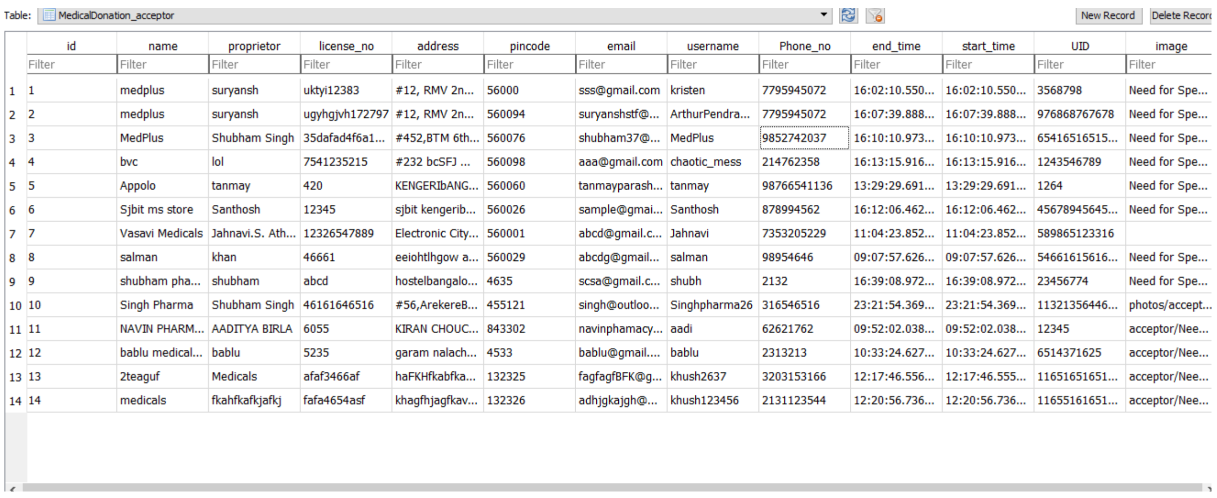


Table: MedicalDonation_acceptor

New Record | Delete Record

	id	name	proprietor	license_no	address	pincode	email	username	Phone_no	end_time	start_time	UID	image
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	medplus	suryansh	uktyi12383	#12, RMV 2n...	56000	sss@gmail.com	kristen	7795945072	16:02:10.550...	16:02:10.550...	3568798	Need for Spe...
2	2	medplus	suryansh	ugyhgjvh172797	#12, RMV 2n...	560094	suryanshstf@...	ArthurPendra...	7795945072	16:07:39.888...	16:07:39.888...	976868767678	Need for Spe...
3	3	MedPlus	Shubham Singh	35dafad4f6a1...	#452,BTM 6th...	560076	shubham37@...	MedPlus	9852742037	16:10:10.973...	16:10:10.973...	65416516515...	Need for Spe...
4	4	bvc	lol	7541235215	#232 bcSFJ ...	560098	aaa@gmail.com	chaotic_mess	214762358	16:13:15.916...	16:13:15.916...	1243546789	Need for Spe...
5	5	Appolo	tanmay	420	KENGERIbANG...	560060	tanmayparash...	tanmay	98766541136	13:29:29.691...	13:29:29.691...	1264	Need for Spe...
6	6	Sjbit ms store	Santhosh	12345	sjbit kengerib...	560026	sample@gmai...	Santhosh	878994562	16:12:06.462...	16:12:06.462...	45678945645...	Need for Spe...
7	7	Vasavi Medicals	Jahnavi.S. Ath...	12326547889	Electronic City...	560001	abcd@gmail.c...	Jahnavi	7353205229	11:04:23.852...	11:04:23.852...	589865123316	
8	8	salman	khan	46661	eeiohtlhgow a...	560029	abcdg@gmail...	salman	98954646	09:07:57.626...	09:07:57.626...	54661615616...	Need for Spe...
9	9	shubham pha...	shubham	abcd	hostelbangalo...	4635	scsa@gmail.c...	shubh	2132	16:39:08.972...	16:39:08.972...	23456774	Need for Spe...
10	10	Singh Pharma	Shubham Singh	46161646516	#56,ArekereB...	455121	singh@outloo...	Singhpharma26	316546516	23:21:54.369...	23:21:54.369...	11321356446...	photos/accept...
11	11	NAVIN PHARM...	AADITYA BIRLA	6055	KIRAN CHOUC...	843302	navinphamacy...	aadi	62621762	09:52:02.038...	09:52:02.038...	12345	acceptor/Nee...
12	12	bablu medical...	bablu	5235	garam nalach...	4533	bablu@gmail....	bablu	2313213	10:33:24.627...	10:33:24.627...	6514371625	acceptor/Nee...
13	13	2teaguf	Medicals	afaf3466af	haFKHfkabfka...	132325	fagfagfBFK@g...	khush2637	3203153166	12:17:46.556...	12:17:46.555...	11651651651...	acceptor/Nee...
14	14	medicals	fkahfkafkjafkj	fafa4654asf	khagfhjagfkav...	132326	adhjgkajgh@...	khush123456	2131123544	12:20:56.736...	12:20:56.736...	11655161651...	acceptor/Nee...

Table 6-3 Acceptor Table

The above table shows the values in the Acceptor Table.

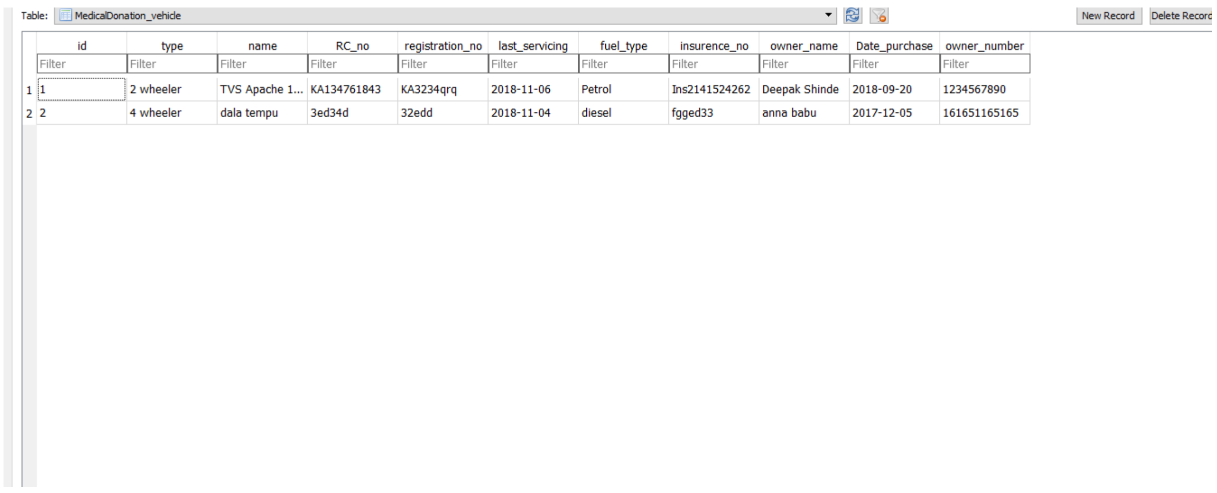


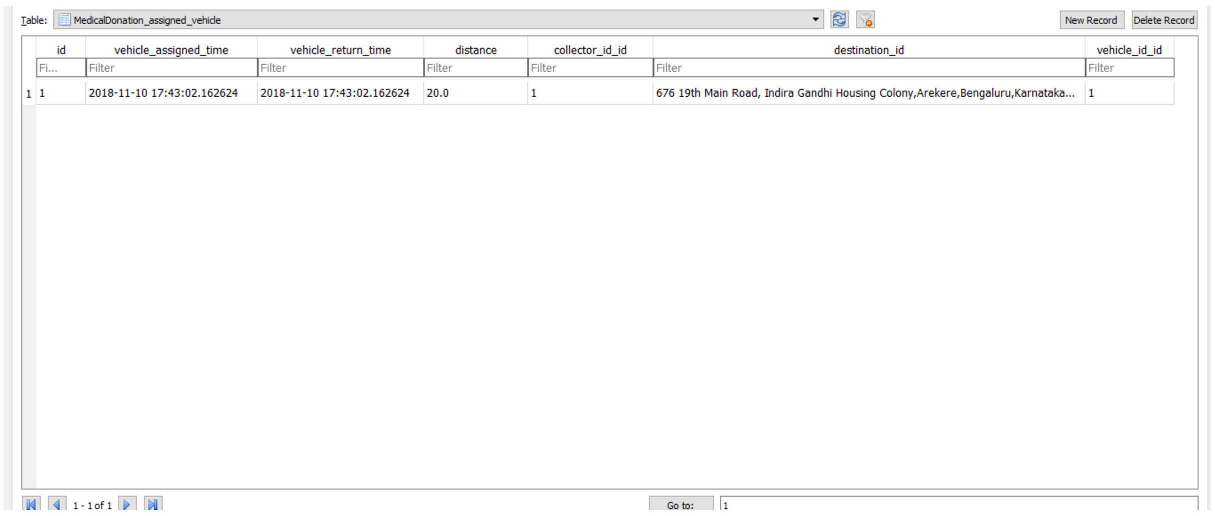
Table: MedicalDonation_vehicle

New Record | Delete Record

	id	type	name	RC_no	registration_no	last_servicing	fuel_type	insurence_no	owner_name	Date_purchase	owner_number
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	2 wheeler	TVS Apache 1...	KA134761843	KA3234qrq	2018-11-06	Petrol	Ins2141524262	Deepak Shinde	2018-09-20	1234567890
2	2	4 wheeler	dala tempu	3ed34d	32edd	2018-11-04	diesel	fgged33	anna babu	2017-12-05	161651165165

Table 6-4 Vehicle Table

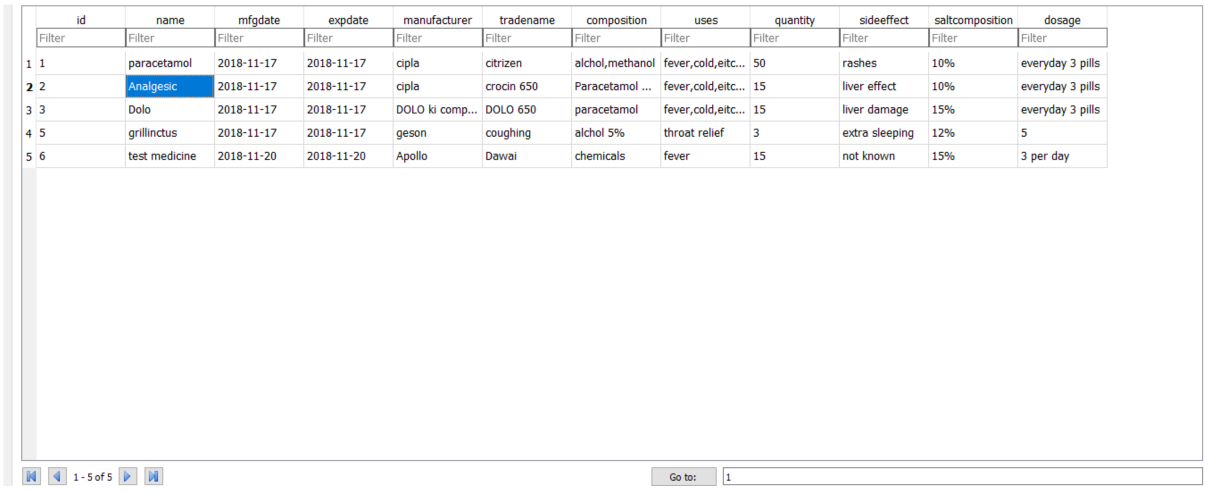
The above table shows the values in the Vehicle Table.



	id	vehicle_assigned_time	vehicle_return_time	distance	collector_id_id	destination_id	vehicle_id_id
1	1	2018-11-10 17:43:02.162624	2018-11-10 17:43:02.162624	20.0	1	676 19th Main Road, Indira Gandhi Housing Colony,Arekere,Bengaluru,Karnataka...	1

Table 6-5 Assigned Vehicle Table

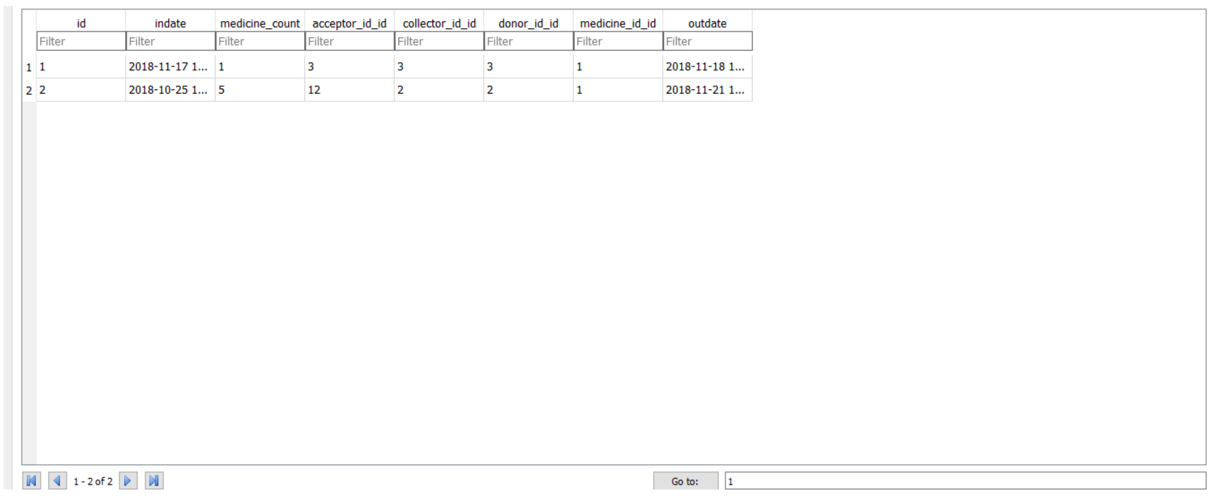
The above table shows the values in the Assigned Vehicle Table.



	id	name	mfgdate	expdate	manufacturer	tradename	composition	uses	quantity	sideeffect	saltcomposition	dosage
1	1	paracetamol	2018-11-17	2018-11-17	cipla	citrizen	alchol,methanol	fever,cold,eitc...	50	rashes	10%	everyday 3 pills
2	2	Analgesic	2018-11-17	2018-11-17	cipla	crocin 650	Paracetamol ...	fever,cold,eitc...	15	liver effect	10%	everyday 3 pills
3	3	Dolo	2018-11-17	2018-11-17	DOLO ki comp...	DOLO 650	paracetamol	fever,cold,eitc...	15	liver damage	15%	everyday 3 pills
4	5	grillinctus	2018-11-17	2018-11-17	geson	coughing	alchol 5%	throat relief	3	extra sleeping	12%	5
5	6	test medicine	2018-11-20	2018-11-20	Apollo	Dawai	chemicals	fever	15	not known	15%	3 per day

Table 6-6 Medicines Table

The above table shows the values in the Medicines Table.



	id	indate	medicine_count	acceptor_id_id	collector_id_id	donor_id_id	medicine_id_id	outdate
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	2018-11-17 1...	1	3	3	3	1	2018-11-18 1...
2	2	2018-10-25 1...	5	12	2	2	1	2018-11-21 1...

Table 6-7 Medicines Ledger Table

The above table shows the values in the Medicines Ledger Table.

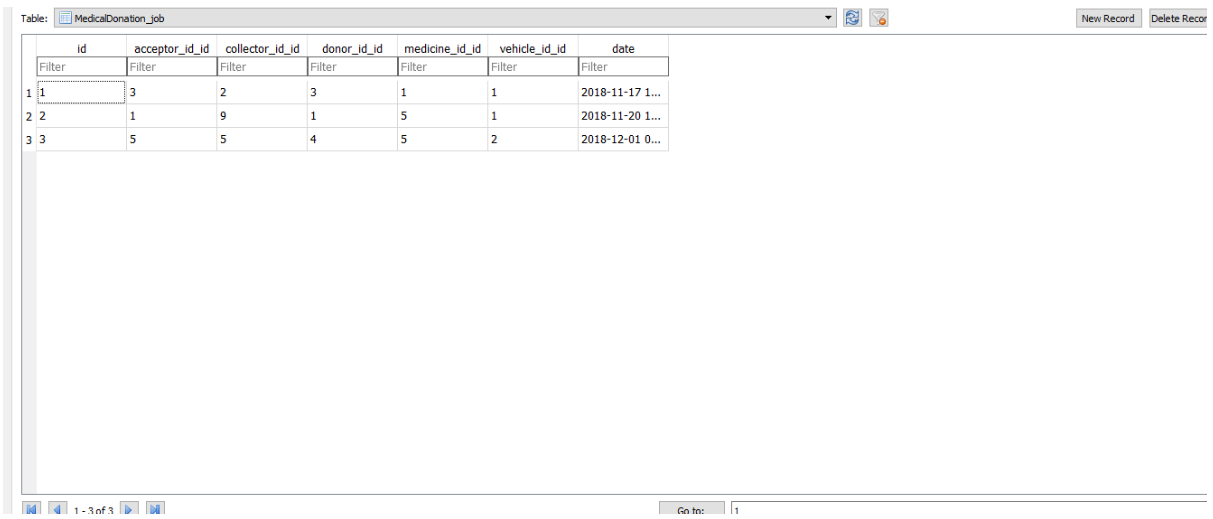


Table: MedicalDonation_job

	id	acceptor_id_id	collector_id_id	donor_id_id	medicine_id_id	vehicle_id_id	date
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	3	2	3	1	1	2018-11-17 1...
2	2	1	9	1	5	1	2018-11-20 1...
3	3	5	5	4	5	2	2018-12-01 0...

Table 6-8 Jobs Table

The above table shows the values in the Jobs Table.

Chapter 7

Conclusion

With the present theoretical inclinations of our syllabus, it becomes very essential to take the utmost advantage of any opportunity of gaining practical experience that comes along. The construction of this mini project was one of these opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts, thus making us more competent.

This project ‘Medicine Donation DataBase Management System’ is majorly about the working system of how dogs are groomed and taken care of which is shown through our database system.

Medical Donation Database Management System is based on the concept of of helping poor people with medicines which they can’t afford. Most of the time we have observed that people are left with medicines which remain unused. These medicines are eventually thrown into the dustbin. As we know life is so important and not all people can afford heavy expenditure, the idea of Medical Donation Database Management System came into picture.

References

[1]. W3Schools (HTML, CSS and PHP references) – https://www.w3schools.com/

[2]. Django Official Documentation – https://docs.djangoproject.com/en/2.1/

[3]. The SQLite3 Documentation – https://sqlite.org/docs.html

[4]. Stack Overflow – https://stackoverflow.com

[5]. Stack Exchange – https://stackexchange.com/

[6]. Udemy - https://udemy.com/

[7]. Wikipedia - https://wikipedia.org

[8]. YouTube – https://www.youtube.com

[9]. Git Hub - https://www.github.com/shubham2637/MedDonate

`