Engineering Project in Community Service

Guide details

Project Supervisor	Dr. K. Sripriyan
Reviewer1	Dr. Pradeep Kumar Kashyap
Reviewer2	Dr. Subhash Chandra Patel

Team Members

20BAI10085	Ayush Porwal
20BAI10299	Rushil Bhatnagar
20BAI10327	Yashaswi Patel
20BCE10498	Drishtavya Gupta
20BCE10892	Deepanshu Pandey
20BCY10197	Aayushi Srivastava
20BCE10845	Akshat Dadhich
20BHI10033	Shivam Sharma

Introduction

Traditional job portals connect businesses and skilled workers. However, there is a need for a platform that connects local experts with people who need their services. Our project will create such a platform, where local experts can find part-time or on-demand work. We will verify the experts and make sure that they are qualified to do the work. Customers can then book the services they need from the experts in their area. It Connects local experts with people who need their services, provies part-time or on-demand work for local experts, verifies the experts to ensure quality and makes it easy for customers to book services

Motivation

Plumbers, cleaners, carpenters, and other home service providers are essential for our daily needs. Most likely, people manually obtain these services or make reservations through websites that do so, but only in a few cities and with a small workforce. Currently, accessing these services is limited to a few cities and a small workforce, primarily through manual requests or limited websites. However, there is a significant demand for job opportunities in our area, with individuals willing to work for lower rates than companies charge.

Our project aims to create a platform that enables these individuals to connect directly with customers, ensuring fair and prompt bookings. It seeks to empower the local workforce and bridge the gap between service providers and customers in a mutually beneficial way.

Objective

To create a portal for local and unemployed workers, without any fancy qualifications and people looking for part-time jobs which will futher provide customers with an on-demand portal to get locally available home services at their convenience.

Existing Work

90 percent of the 50 crore workers who work in both organised and unorganised sectors are unorganised. In terms of regulation with employers, overtime, exploitation, casual work culture, and many other issues, this unorganised sector faces numerous difficulties and challenges. Currently, industries use labour brokers to locate unskilled labourers; in return for putting the two together, they pay a commission based on the wages of these workers.

There are many job portals that facilitate direct communication between employees and employers by posting job openings etc. The employee only needs to enter his or her degree, experience, prior employment information, skills, and so forth. However, unorganised workers who lack a degree corresponding to their skills, like technicians, plumbers, joiners, and painters, cannot access such facilities.

Introduction to the System

Online Home Services

This Online Home Services project will deal with an online system designed for booking serviceman (electrician, carpenter, plumber, painter etc.) as per the requirements of the customers at their convenience.

The current system is manual mostly and it is time-consuming. It is also cost-ineffective, and the average return is low and diminishing.

Goal

- To make a web portal which provides bookings to local servicemen.
- To make a database that is consistent, reliable and secure.
- To provide correct, complete, ongoing information.
- To develop a well-organized information storage system.
- To make good documentation so as to facilitate possible future enhancements.

Hardware and Software Specification

Software Requirements:

- Technology: Python Django
- IDE : Spyder / VsCode
- Client Side Technologies: HTML, CSS, JavaScript , Bootstrap
- Server Side Technologies: Python
- Data Base Server: SQLite
- Operating System: Microsoft Windows/Linux

Hardware Requirements:

- Processor: Pentium-III (or) Higher
- Ram: 4GB (or) Higher
- Hard disk: 256GB (or) Higher

System Overview:

The key features that are required in the system are as follows:

Admin Module

- **Manage Servicemen**
- **Manage Users**
- **Manage Bookings**

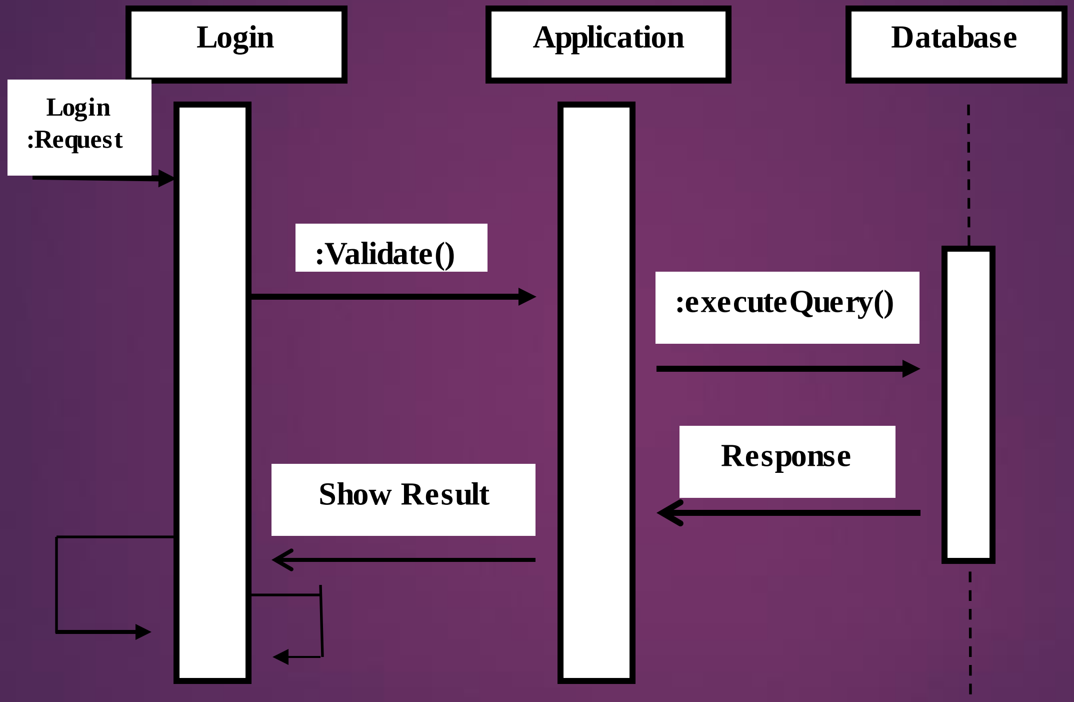
Servicemen Module

- Create Profile/Edit Profile
- View Bookings

User Modules

- Search Servicemen
- View Servicemen Profiles
- Book Servicemen

Sequence Diagram

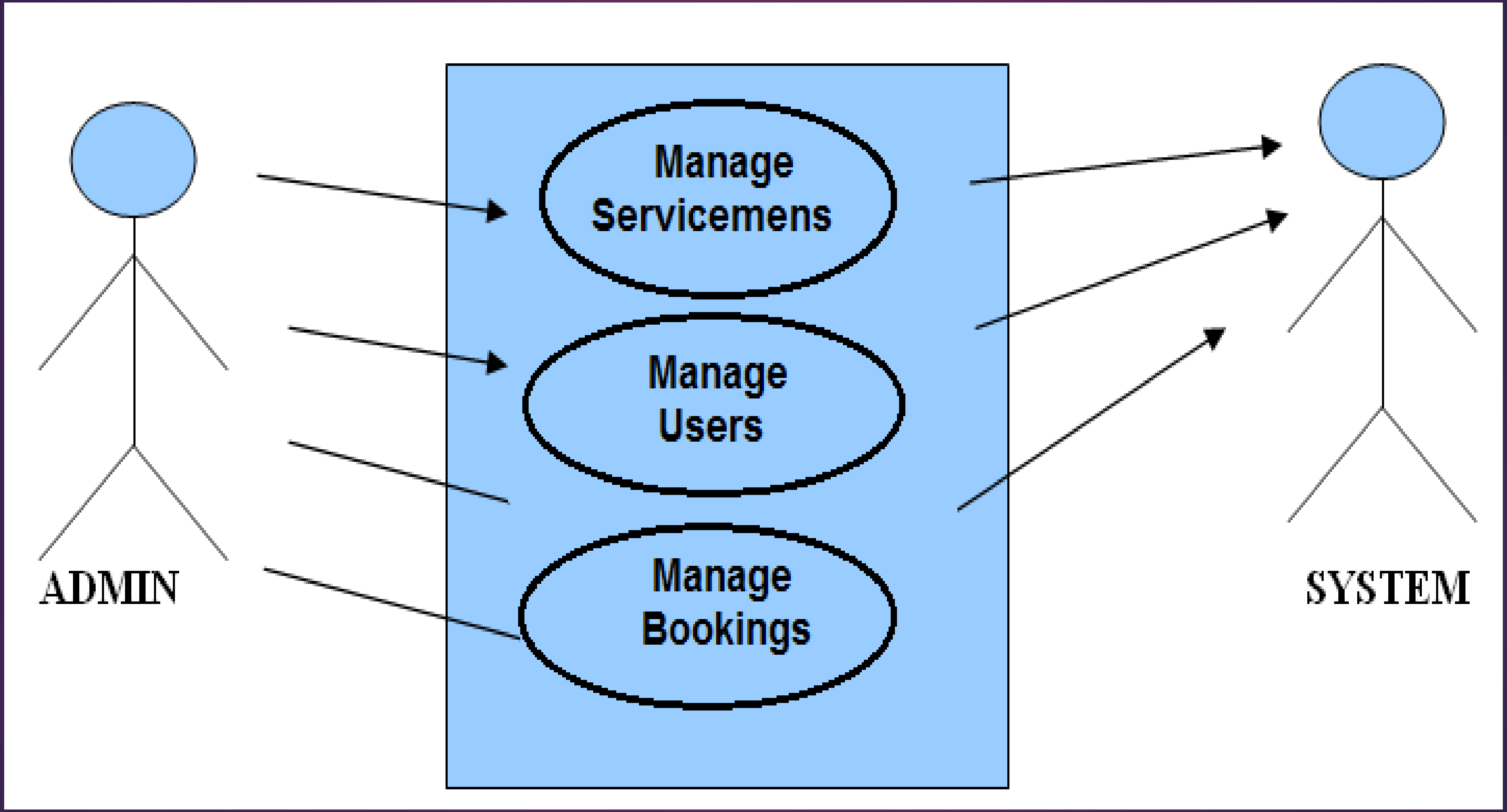


USE CASE DIAGRAM

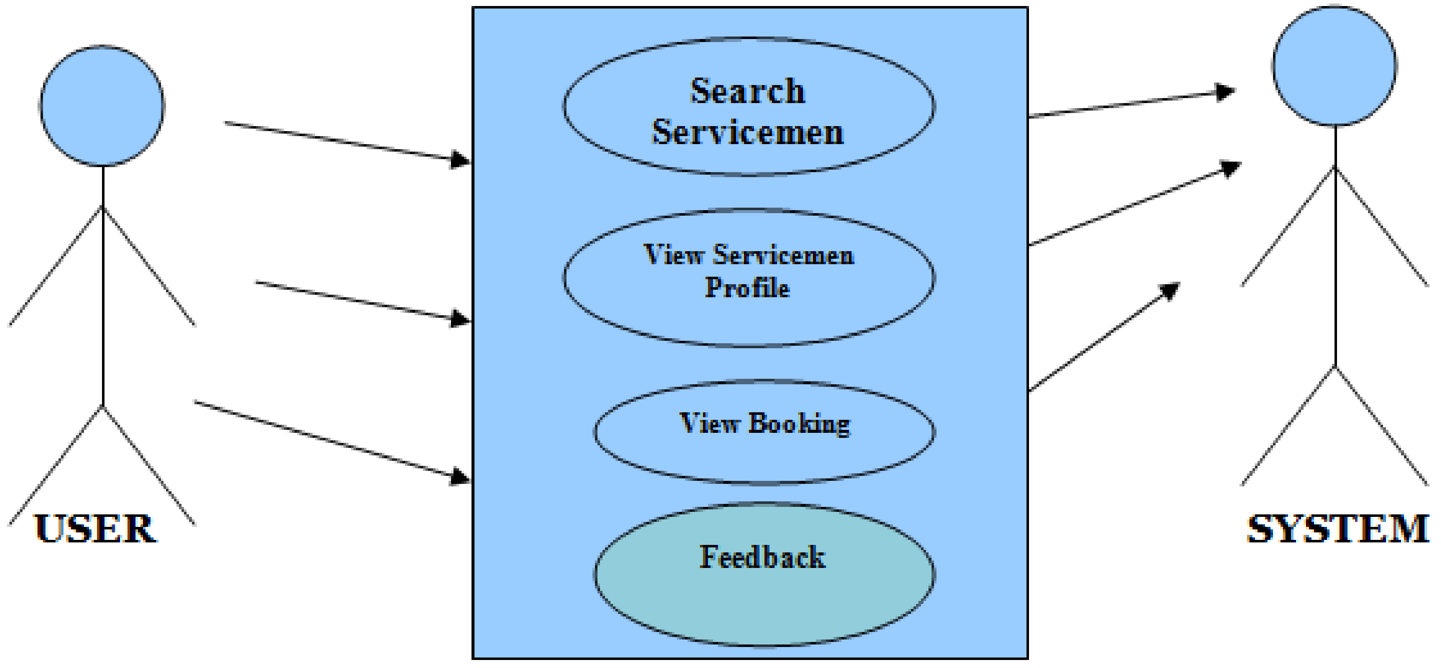
The key points are:

- The main purpose is to show the interaction between the use cases and the actor.
- To represent the system requirement from user’s perspective.
- The use cases are the functions that are to be performed in the module.
- An actor could be the end-user of the system or an external system.

USE CASE - ADMIN

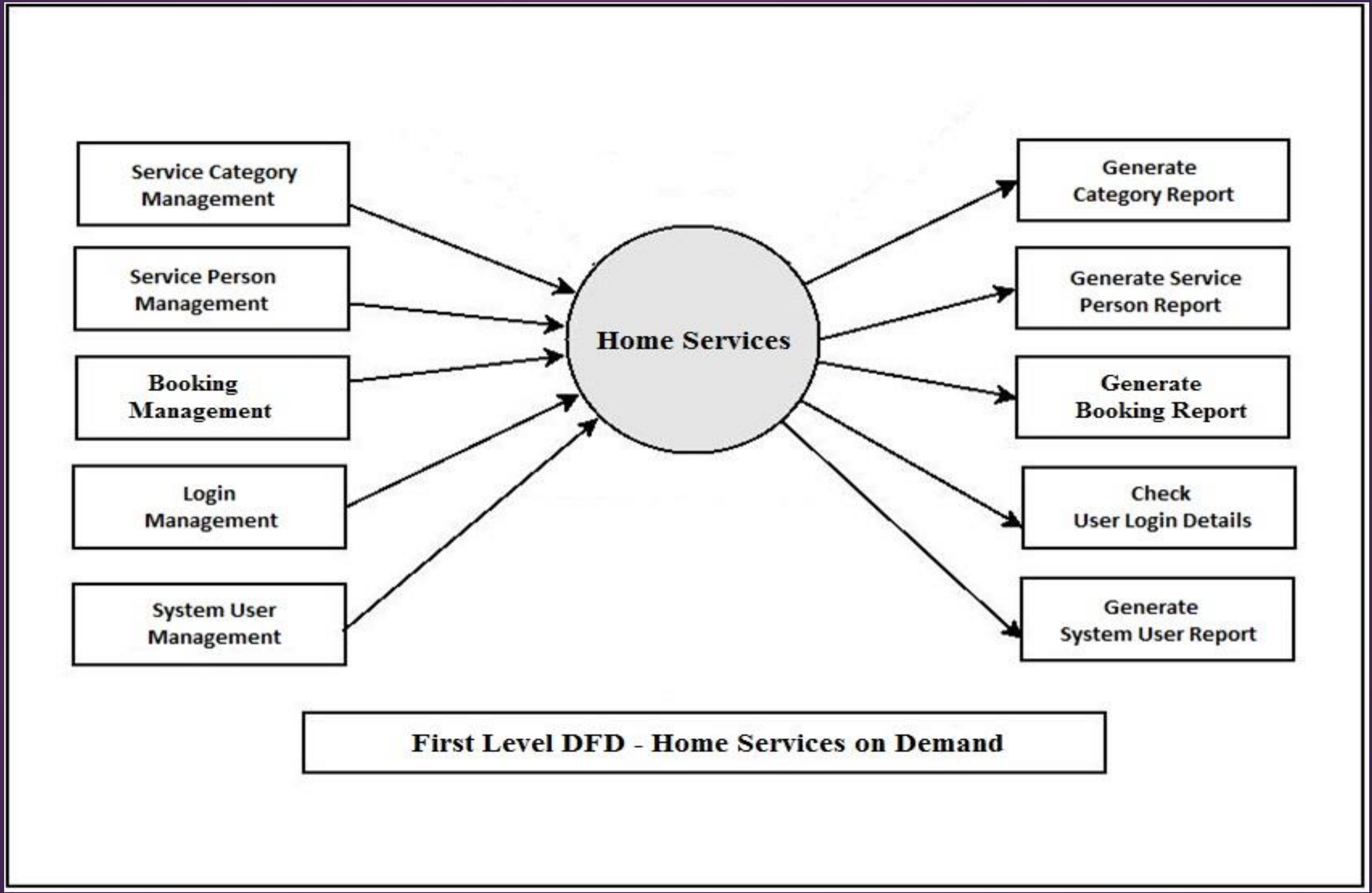


USE CASE - USER



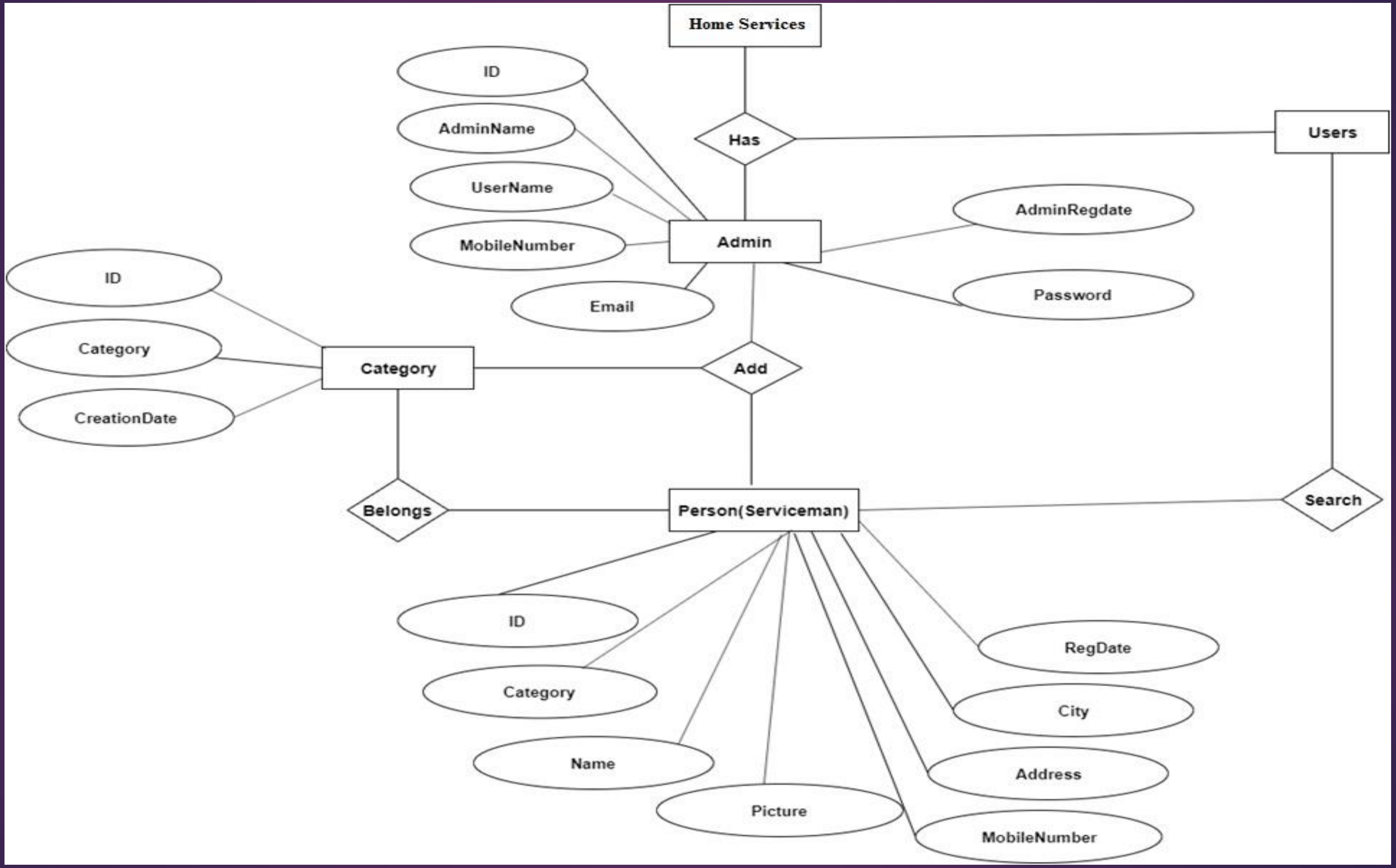
Use Case Diagram between USER and SYSTEM:

DFD (Data Flow Diagram)

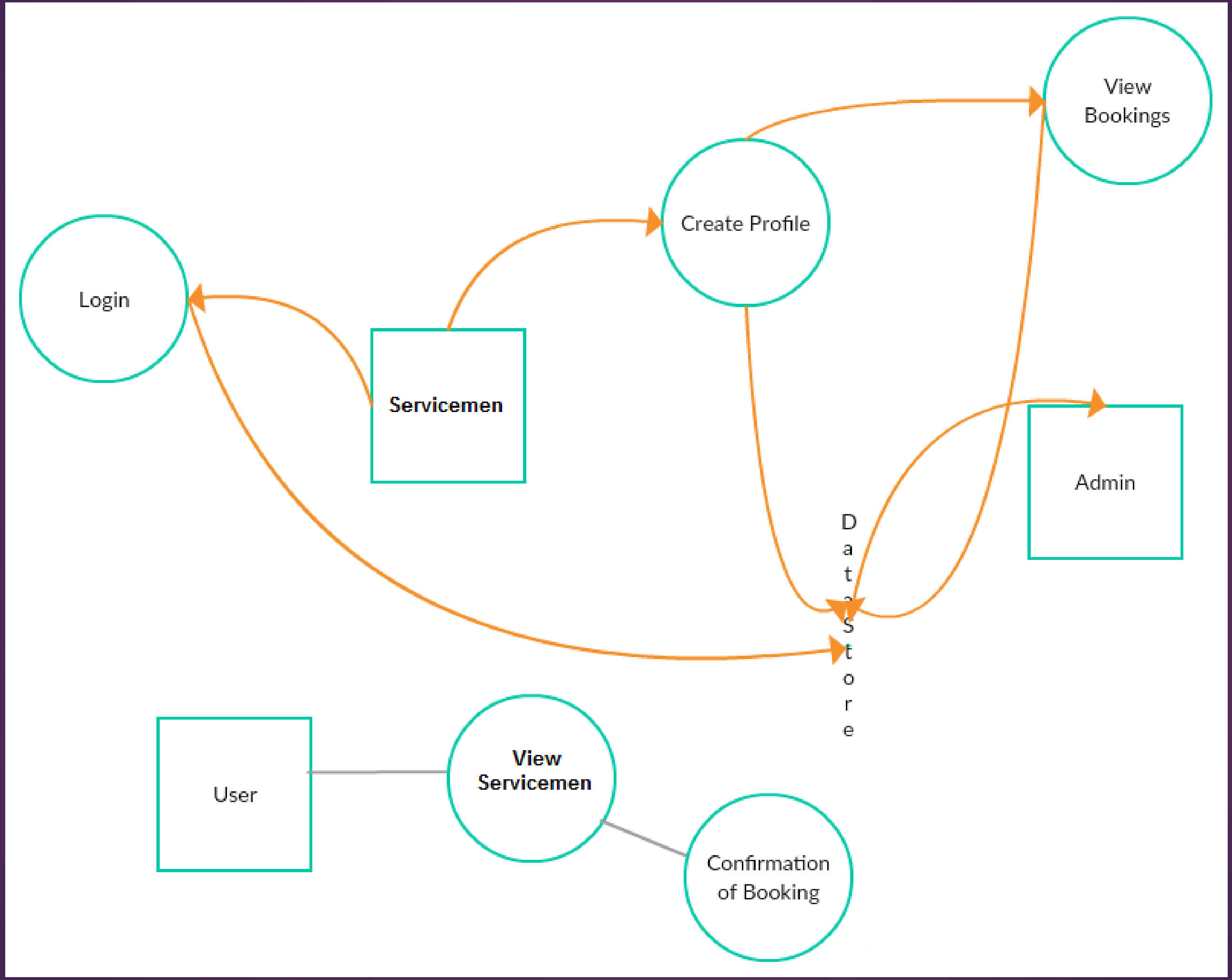


First Level DFD - Home Services on Demand

ER DIAGRAM


Home Services
ID
AdminName
UserName
MobileNumber
Email
Has
Users
Admin
AdminRegdate
Password
ID
Category
CreationDate
Category
Add
Belongs
Person(Serviceman)
Search
ID
Category
Name
Picture
RegDate
City
Address
MobileNumber

Working diagram



Brief about our framework

What is Django?

- Django is a Python framework that makes it easier to create web sites using Python.
- Django takes care of the difficult stuff so that you can concentrate on building your web applications.
- Django emphasizes reusability of components, also referred to as DRY (Don't Repeat Yourself), and comes with ready-to-use features like login system, database connection and CRUD operations (Create Read Update Delete).

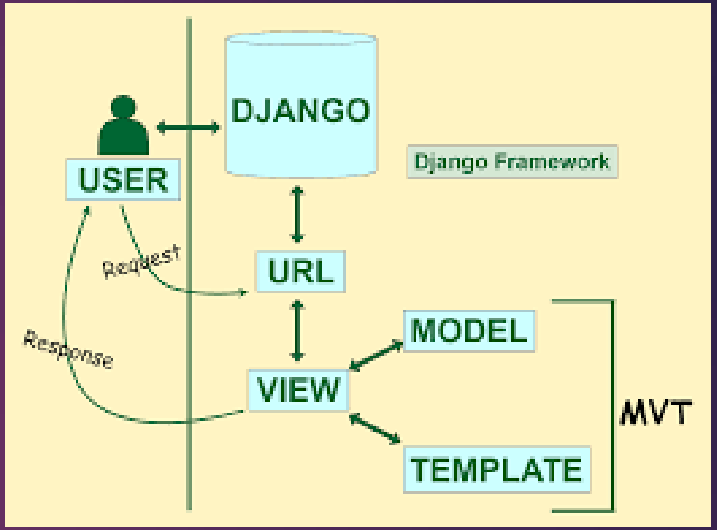
How does Django Work?

Django follows the MVT design pattern (Model View Template).

•Model - The data you want to present, usually data from a database.

•View - A request handler that returns the relevant template and content - based on the request from the user.

•Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.



Model

The model provides data from the database.

In Django, the data is delivered as an Object Relational Mapping (ORM), which is a technique designed to make it easier to work with databases.

The most common way to extract data from a database is SQL. One problem with SQL is that you have to have a pretty good understanding of the database structure to be able to work with it.

Django, with ORM, makes it easier to communicate with the database, without having to write complex SQL statements.

The models are usually located in a file called models.py.

View

A view is a function or method that takes http requests as arguments, imports the relevant model(s), and finds out what data to send to the template, and returns the final result.

The views are usually located in a file called views.py.

Template

A template is a file where you describe how the result should be represented.

Templates are often .html files, with HTML code describing the layout of a web page, but it can also be in other file formats to present other results, but we will concentrate on .html files.

Django uses standard HTML to describe the layout, but uses Django tags to add logic:

<h1>My Homepage</h1> <p>My name is {{ firstname }}.</p> The templates of an application is located in a folder named templates.

URLs

Django also provides a way to navigate around the different pages in a website.
When a user requests a URL, Django decides which *view* it will send it to.

This is done in a file called urls.py.

So, What is Going On?

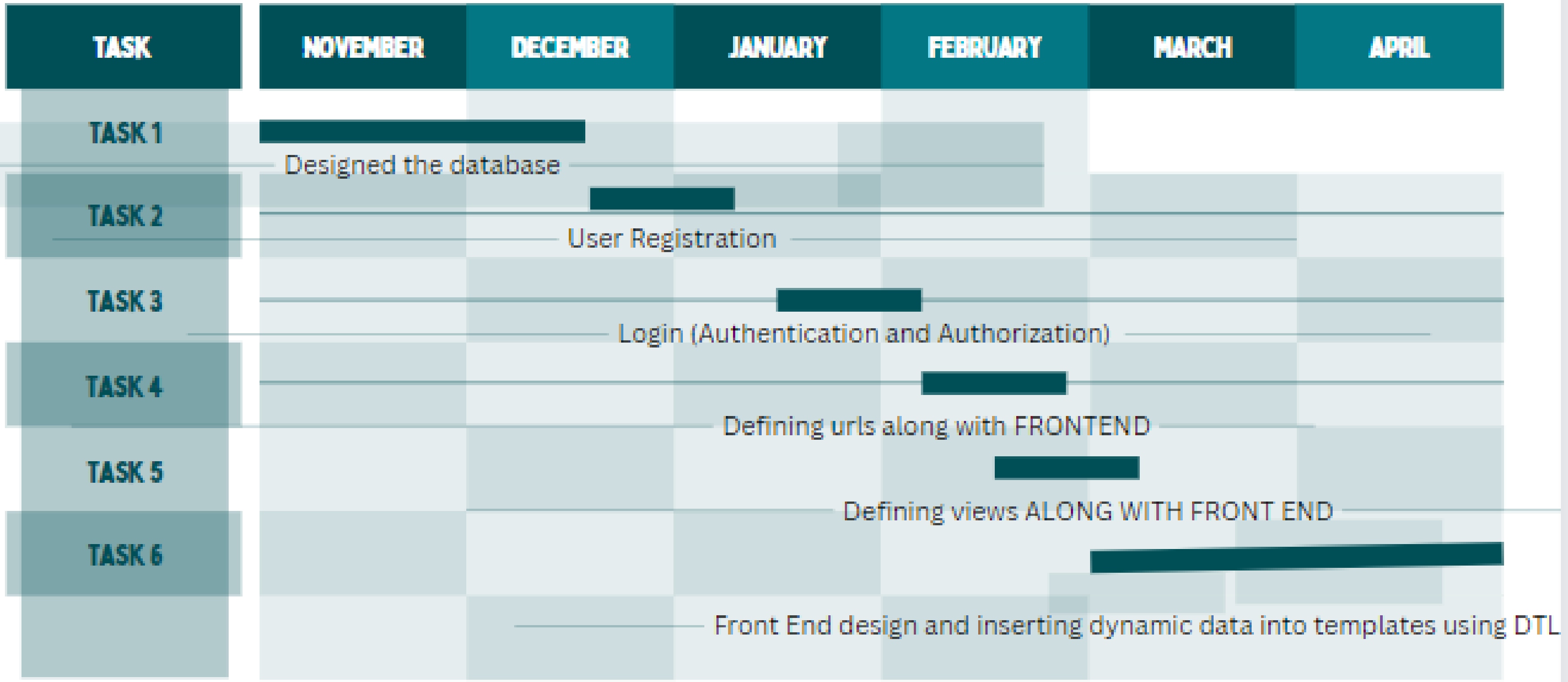
When we have installed Django and created our first Django web application, and the browser requests the URL, this is basically what happens:

1. Django receives the URL, checks the urls.py file, and calls the view that matches the URL.
2. The view, located in views.py, checks for relevant models.
3. The models are imported from the models.py file.
4. The view then sends the data to a specified template in the template folder.
5. The template contains HTML and Django tags, and with the data it returns finished HTML content back to the browser.

Online Home Services Portal

GANTT CHART

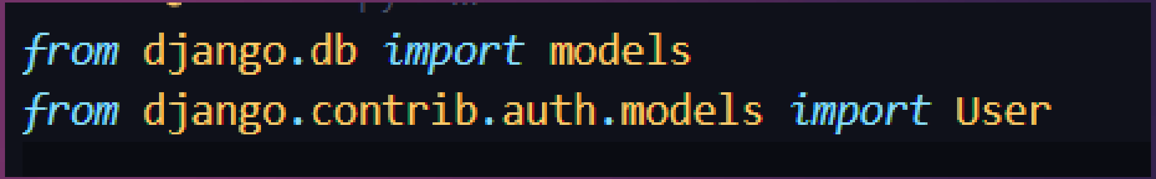
Stepwise workflow of our project



Stepwise Workflow (Main Points)

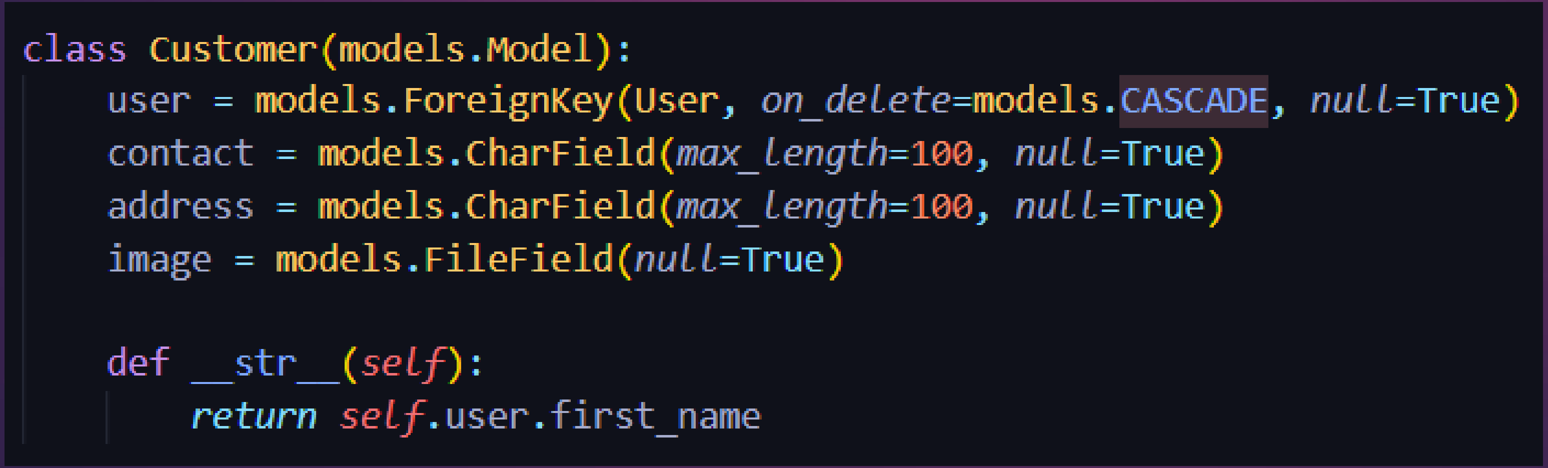
1. Design our database

A Django model is a table in database. So we have created multiple models according to our need in models.py file. It comes with a inbuilt user model which we can use as foreign key in our customer model or servicemen model or admin model.



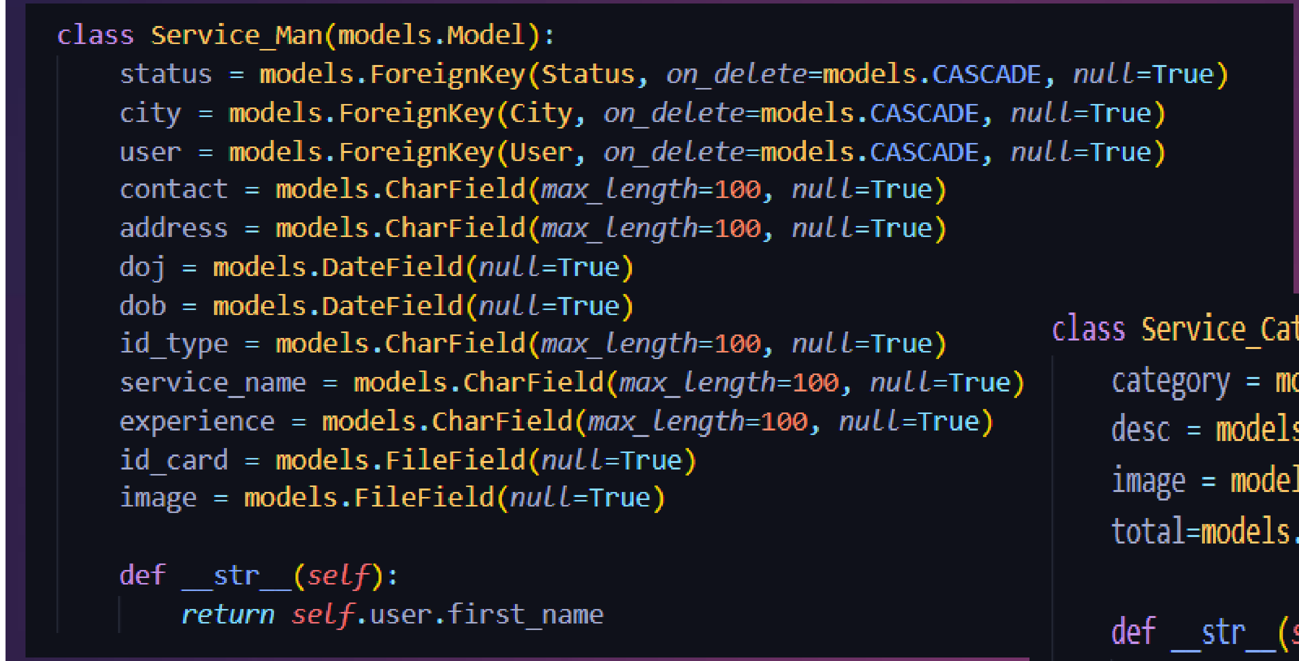
```
from django.db import models
from django.contrib.auth.models import User
```

Customer model



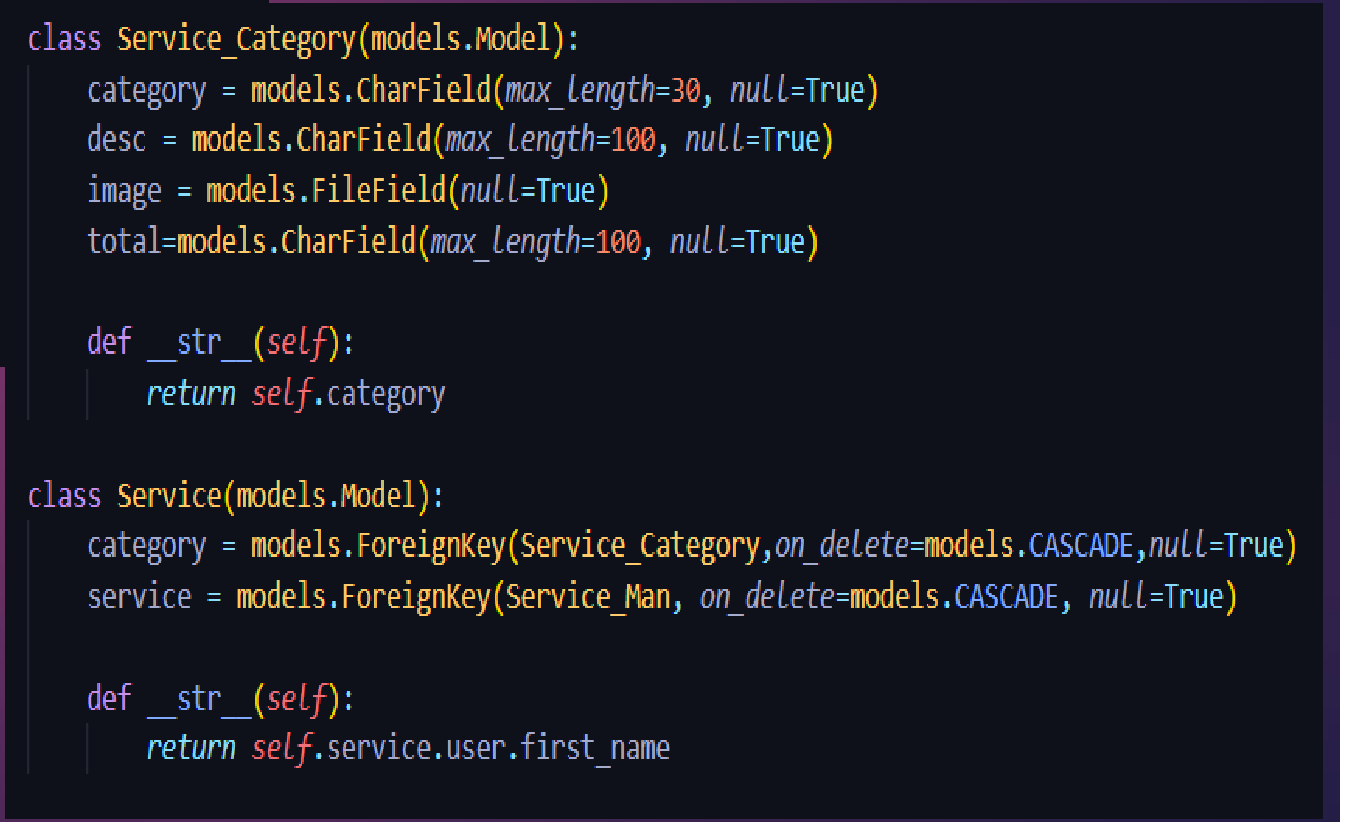
```
class Customer(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    contact = models.CharField(max_length=100, null=True)
    address = models.CharField(max_length=100, null=True)
    image = models.FileField(null=True)

    def __str__(self):
        return self.user.first_name
```

Servicemen model



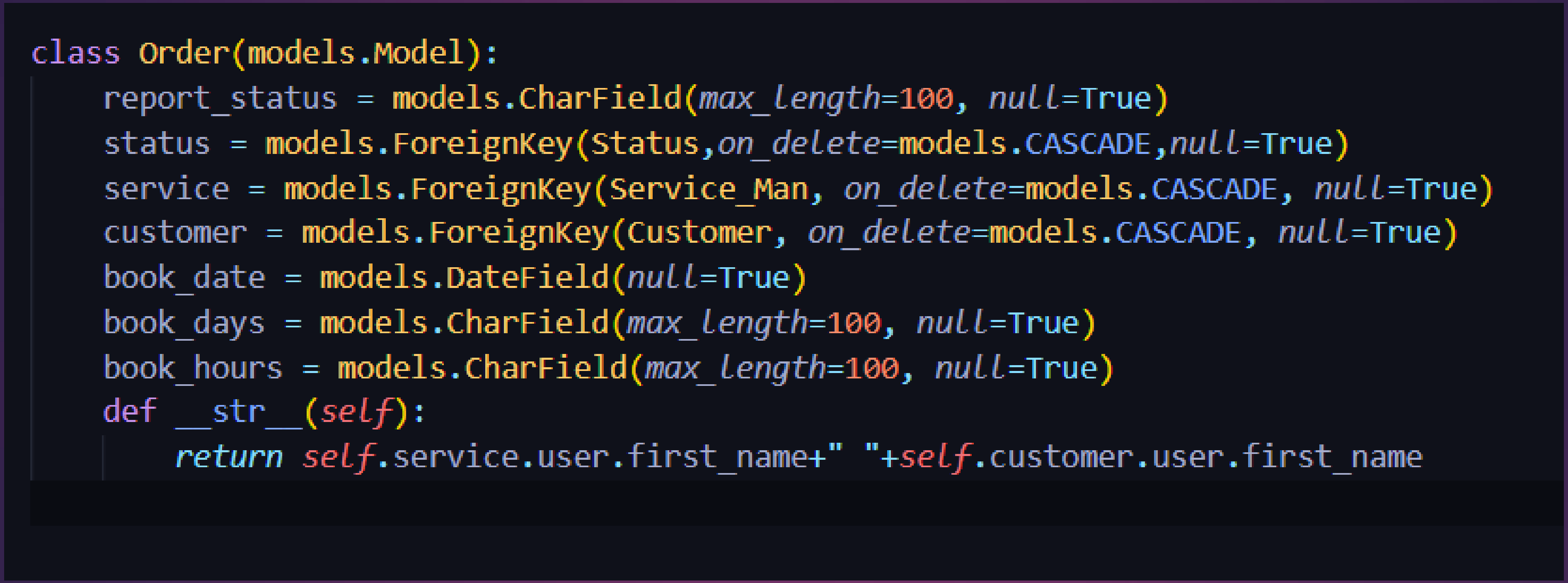
```python
class Service_Man(models.Model):
    status = models.ForeignKey(Status, on_delete=models.CASCADE, null=True)
    city = models.ForeignKey(City, on_delete=models.CASCADE, null=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    contact = models.CharField(max_length=100, null=True)
    address = models.CharField(max_length=100, null=True)
    doj = models.DateField(null=True)
    dob = models.DateField(null=True)
    id_type = models.CharField(max_length=100, null=True)
    service_name = models.CharField(max_length=100, null=True)
    experience = models.CharField(max_length=100, null=True)
    id_card = models.FileField(null=True)
    image = models.FileField(null=True)

    def __str__(self):
        return self.user.first_name
```

Service Category & Service model



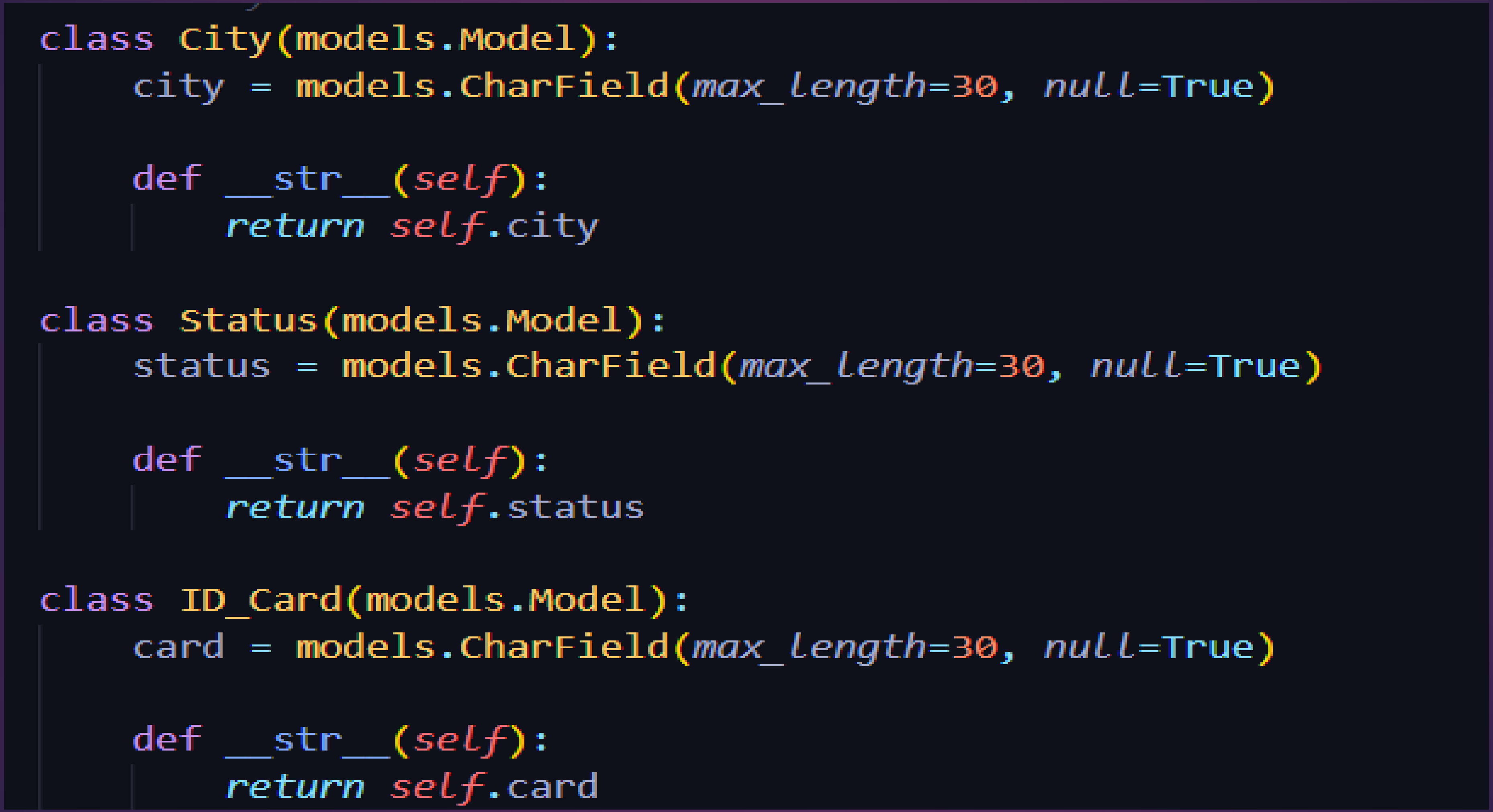
```python
class Service_Category(models.Model):
    category = models.CharField(max_length=30, null=True)
    desc = models.CharField(max_length=100, null=True)
    image = models.FileField(null=True)
    total=models.CharField(max_length=100, null=True)

    def __str__(self):
        return self.category

class Service(models.Model):
    category = models.ForeignKey(Service_Category,on_delete=models.CASCADE,null=True)
    service = models.ForeignKey(Service_Man, on_delete=models.CASCADE, null=True)

    def __str__(self):
        return self.service.user.first_name
```

Order model



```python
class Order(models.Model):
    report_status = models.CharField(max_length=100, null=True)
    status = models.ForeignKey(Status,on_delete=models.CASCADE,null=True)
    service = models.ForeignKey(Service_Man, on_delete=models.CASCADE, null=True)
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE, null=True)
    book_date = models.DateField(null=True)
    book_days = models.CharField(max_length=100, null=True)
    book_hours = models.CharField(max_length=100, null=True)
    def __str__(self):
        return self.service.user.first_name+" "+self.customer.user.first_name
```

City & Status & ID_Card models

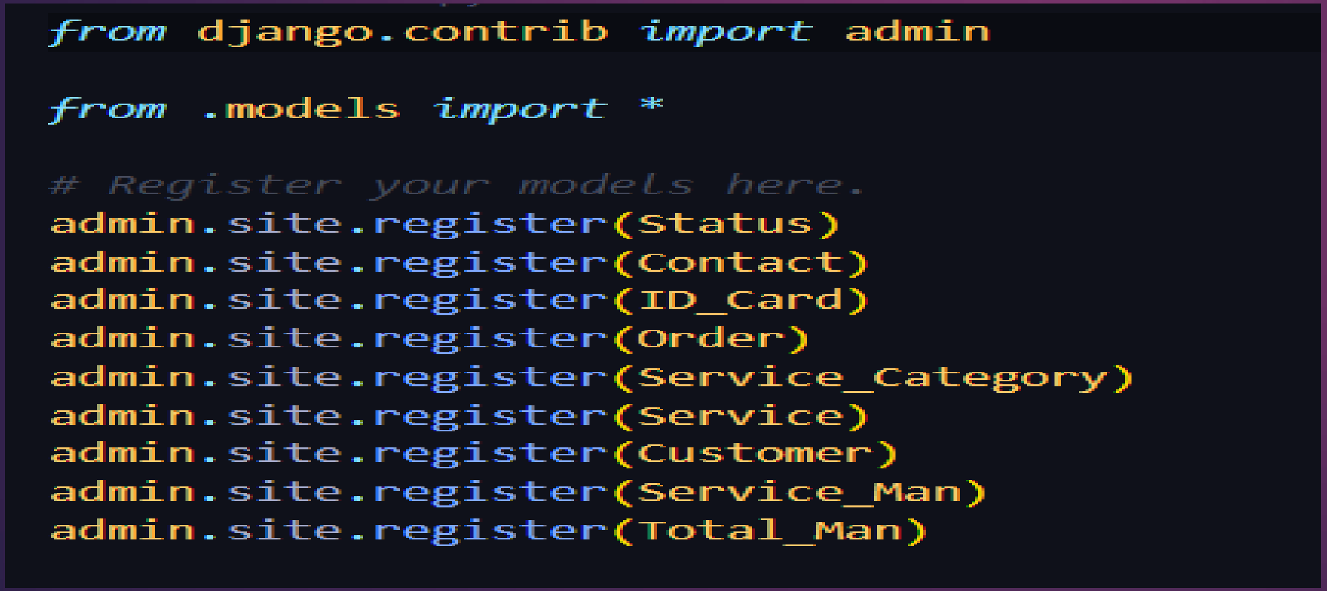


```
class City(models.Model):
    city = models.CharField(max_length=30, null=True)

    def __str__(self):
        return self.city

class Status(models.Model):
    status = models.CharField(max_length=30, null=True)

    def __str__(self):
        return self.status

class ID_Card(models.Model):
    card = models.CharField(max_length=30, null=True)

    def __str__(self):
        return self.card
```

2. Create an admin panel from where admin can perform all basic CRUD operations on database.

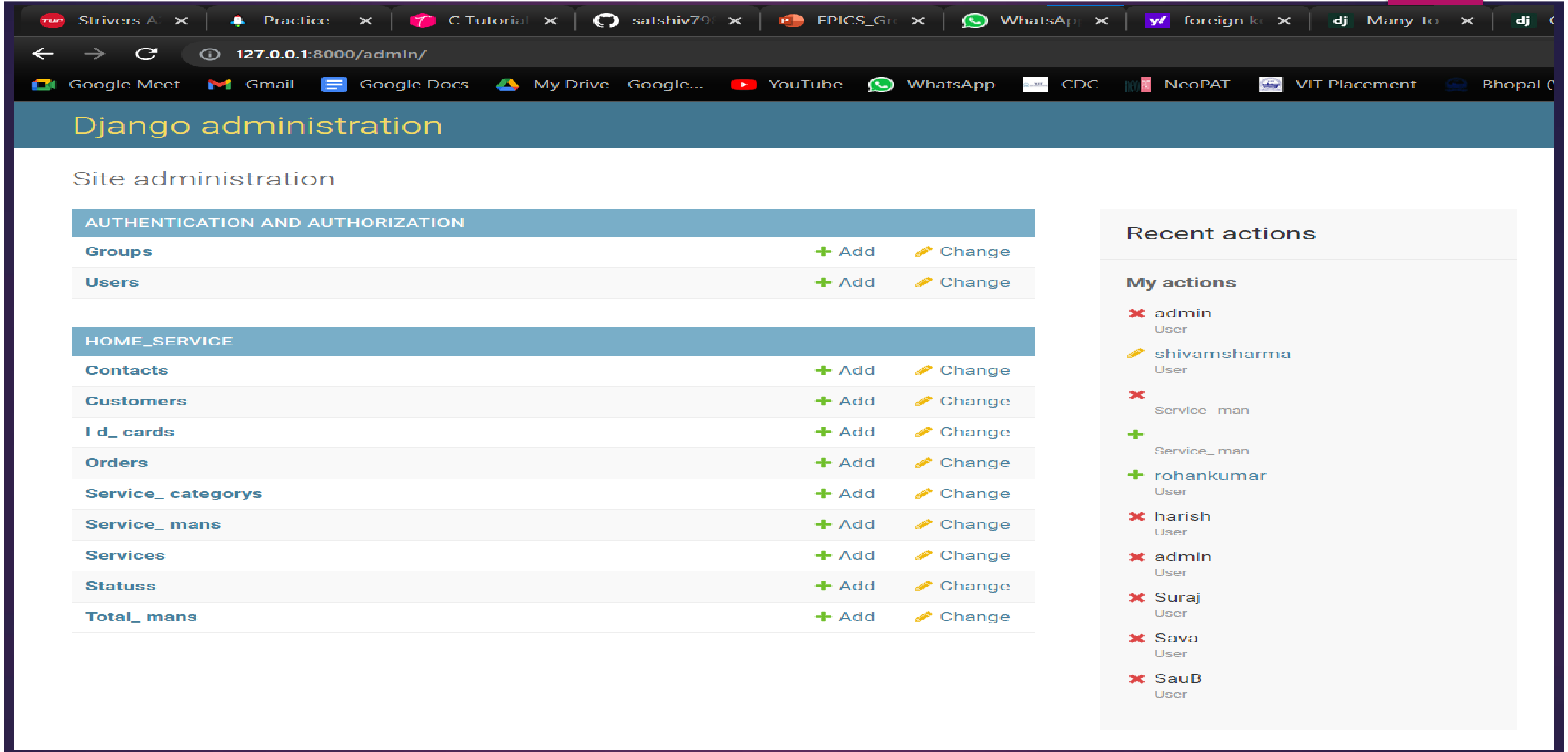
For example – If we have gathered servicemen data from nearby areas and we want to feed that data into our database, we should have a proper interface which allows admin to not have SQL knowledge to execute queries to put that data.

With the admin panel , all database CRUD operations becomes easy.



```
from django.contrib import admin

from .models import *

# Register your models here.
admin.site.register(Status)
admin.site.register(Contact)
admin.site.register(ID_Card)
admin.site.register(Order)
admin.site.register(Service_Category)
admin.site.register(Service)
admin.site.register(Customer)
admin.site.register(Service_Man)
admin.site.register(Total_Man)
```

We need to register our models in admin.py file.


127.0.0.1:8000/admin/

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

HOME_SERVICE		
Contacts	+ Add	Change
Customers	+ Add	Change
I d_ cards	+ Add	Change
Orders	+ Add	Change
Service_ categorys	+ Add	Change
Service_ mans	+ Add	Change
Services	+ Add	Change
Statuss	+ Add	Change
Total_ mans	+ Add	Change

Recent actions

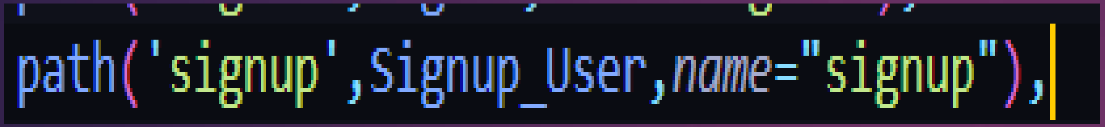
My actions

- admin
  User
- shivamsharma
  User
- Service_ man
- Service_ man
- rohankumar
  User
- harish
  User
- admin
  User
- Suraj
  User
- Sava
  User
- SauB
  User

http://127.0.0.1:8000/admin/

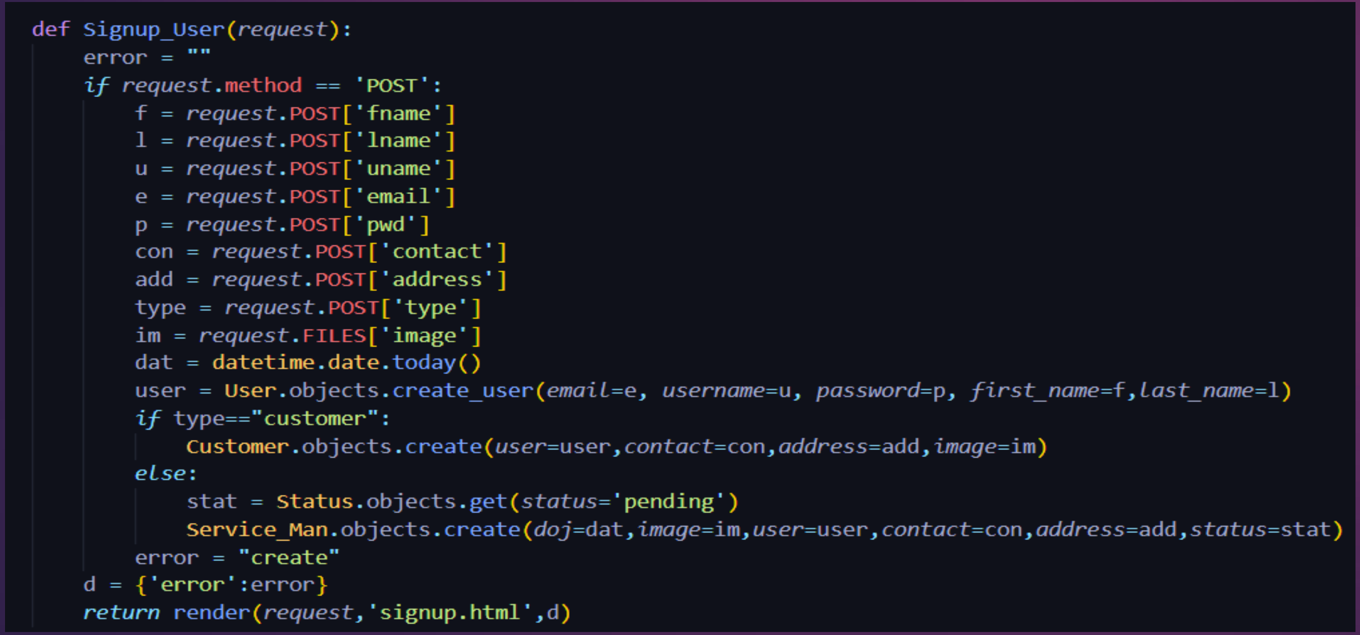
3. User Registration

Define URL

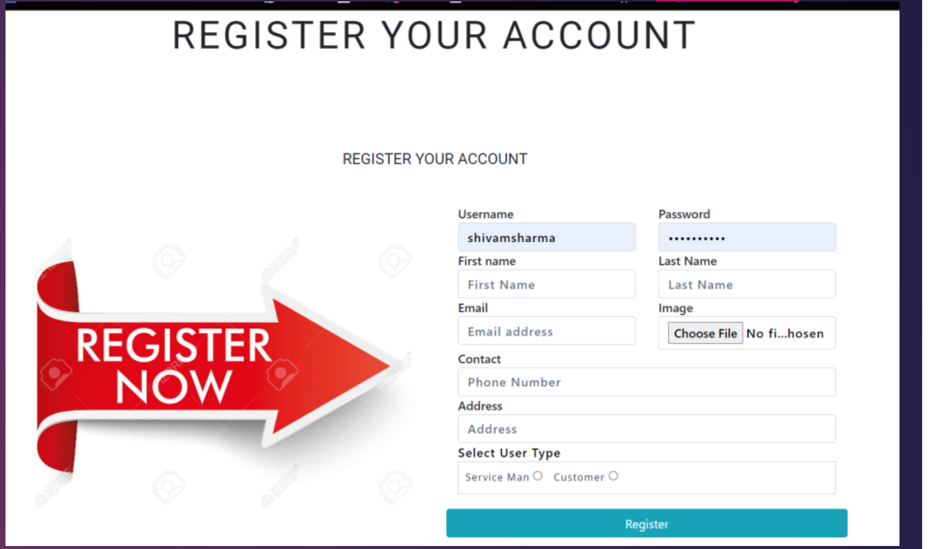


```
path('signup',Signup_User,name="signup"),
```

Define view



```
def Signup_User(request):
    error = ""
    if request.method == 'POST':
        f = request.POST['fname']
        l = request.POST['lname']
        u = request.POST['uname']
        e = request.POST['email']
        p = request.POST['pwd']
        con = request.POST['contact']
        add = request.POST['address']
        type = request.POST['type']
        im = request.FILES['image']
        dat = datetime.date.today()
        user = User.objects.create_user(email=e, username=u, password=p, first_name=f,last_name=l)
        if type=="customer":
            Customer.objects.create(user=user,contact=con,address=add,image=im)
        else:
            stat = Status.objects.get(status='pending')
            Service_Man.objects.create(doj=dat,image=im,user=user,contact=con,address=add,status=stat)
        error = "create"
    d = {'error':error}
    return render(request,'signup.html',d)
```



REGISTER YOUR ACCOUNT

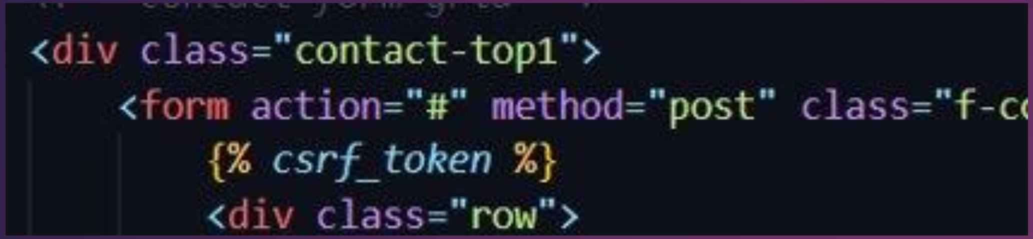
REGISTER YOUR ACCOUNT

Username: shivamsharma	Password: ••••••••••
First name: First Name	Last Name: Last Name
Email: Email address	Image: Choose File No fi...hosen
Contact: Phone Number	
Address: Address	
Select User Type: Service Man ○ Customer ○	

Register

SignUp.html

CSRF_token

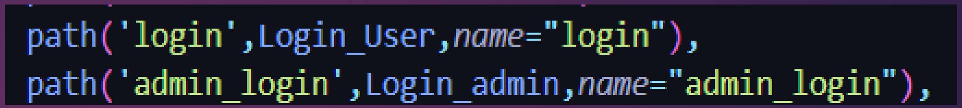


```
<div class="contact-top1">
    <form action="#" method="post" class="f-co
        {% csrf_token %}
        <div class="row">
```

- The CSRF middleware and template tag provides easy-to-use protection against Cross Site Request Forgeries.
- This type of attack occurs when a malicious website contains a link, a form button or some JavaScript that is intended to perform some action on your website, using the credentials of a logged-in user who visits the malicious site in their browser. A related type of attack, ‘login CSRF’, where an attacking site tricks a user’s browser into logging into a site with someone else’s credentials, is also covered.

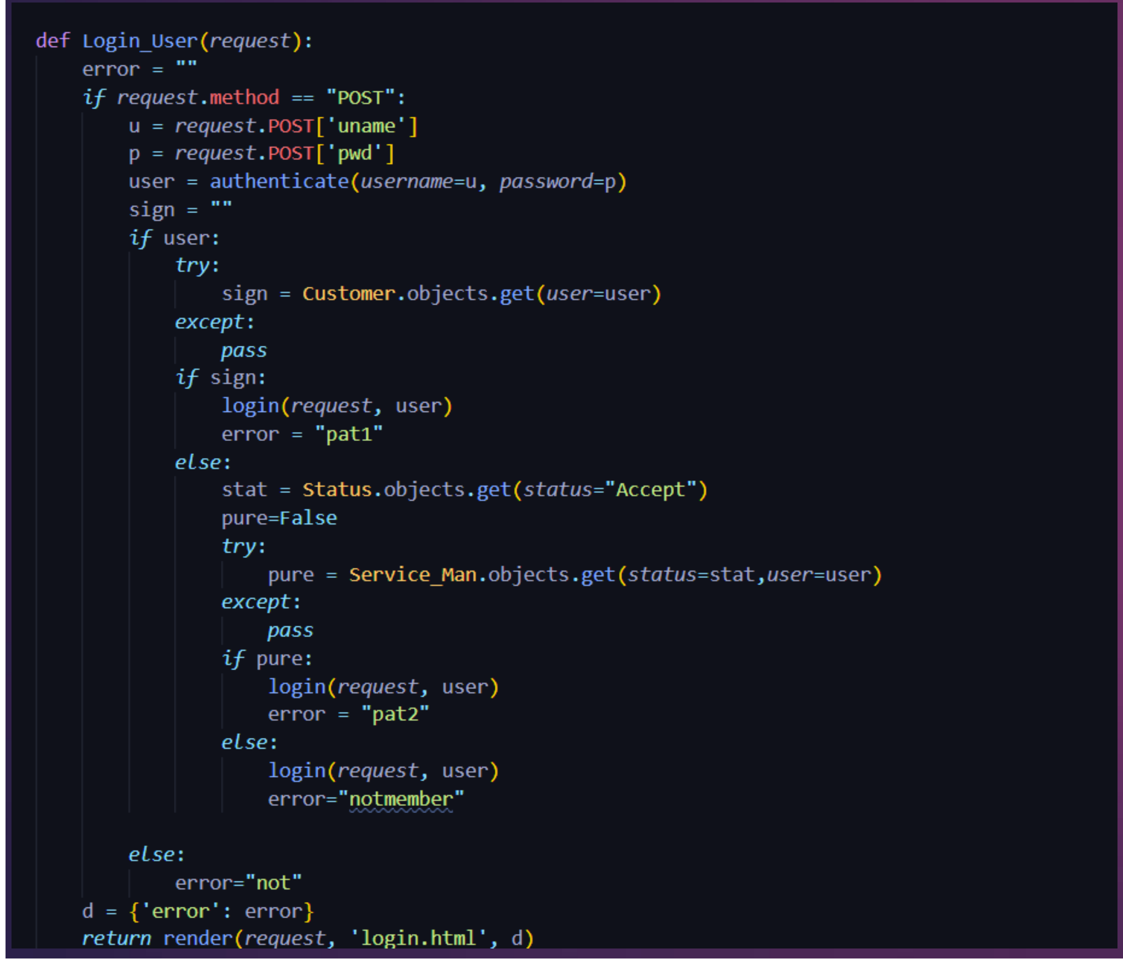
4. Login (Authentication)

urls.py

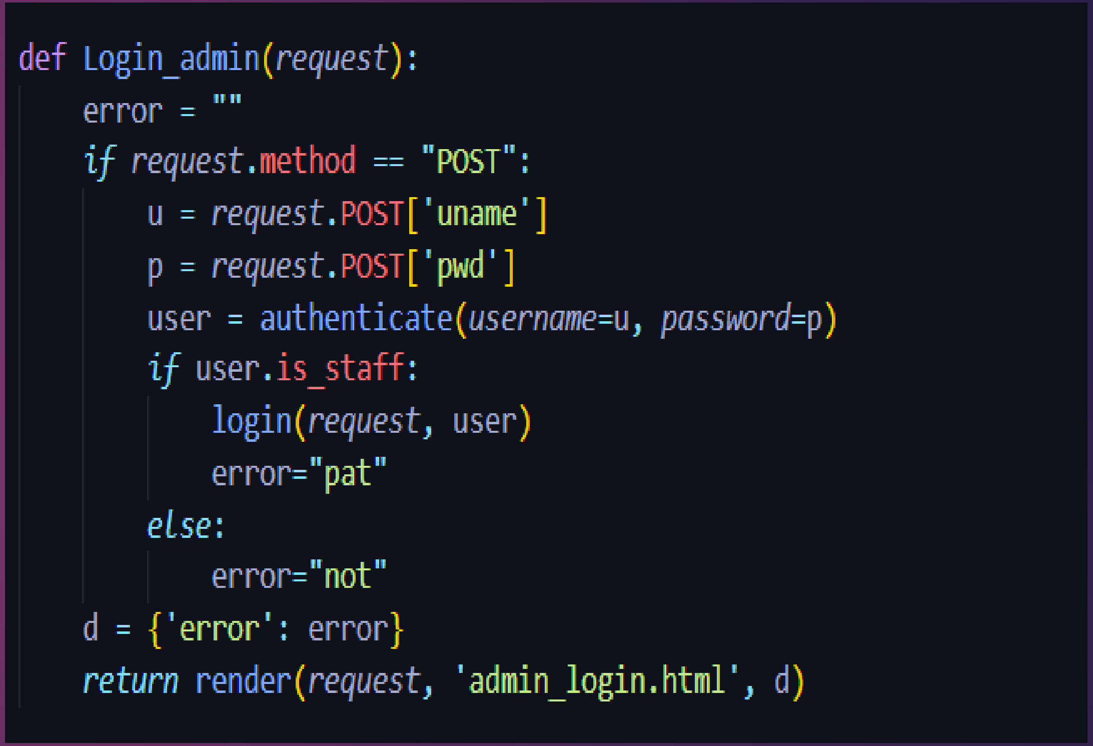


```python
path('login',Login_User,name="login"),
path('admin_login',Login_admin,name="admin_login"),
```

Views.py



```python
def Login_User(request):
    error = ""
    if request.method == "POST":
        u = request.POST['uname']
        p = request.POST['pwd']
        user = authenticate(username=u, password=p)
        sign = ""
        if user:
            try:
                sign = Customer.objects.get(user=user)
            except:
                pass
            if sign:
                login(request, user)
                error = "pat1"
            else:
                stat = Status.objects.get(status="Accept")
                pure=False
                try:
                    pure = Service_Man.objects.get(status=stat,user=user)
                except:
                    pass
                if pure:
                    login(request, user)
                    error = "pat2"
                else:
                    login(request, user)
                    error="notmember"

        else:
            error="not"
    d = {'error': error}
    return render(request, 'login.html', d)
```

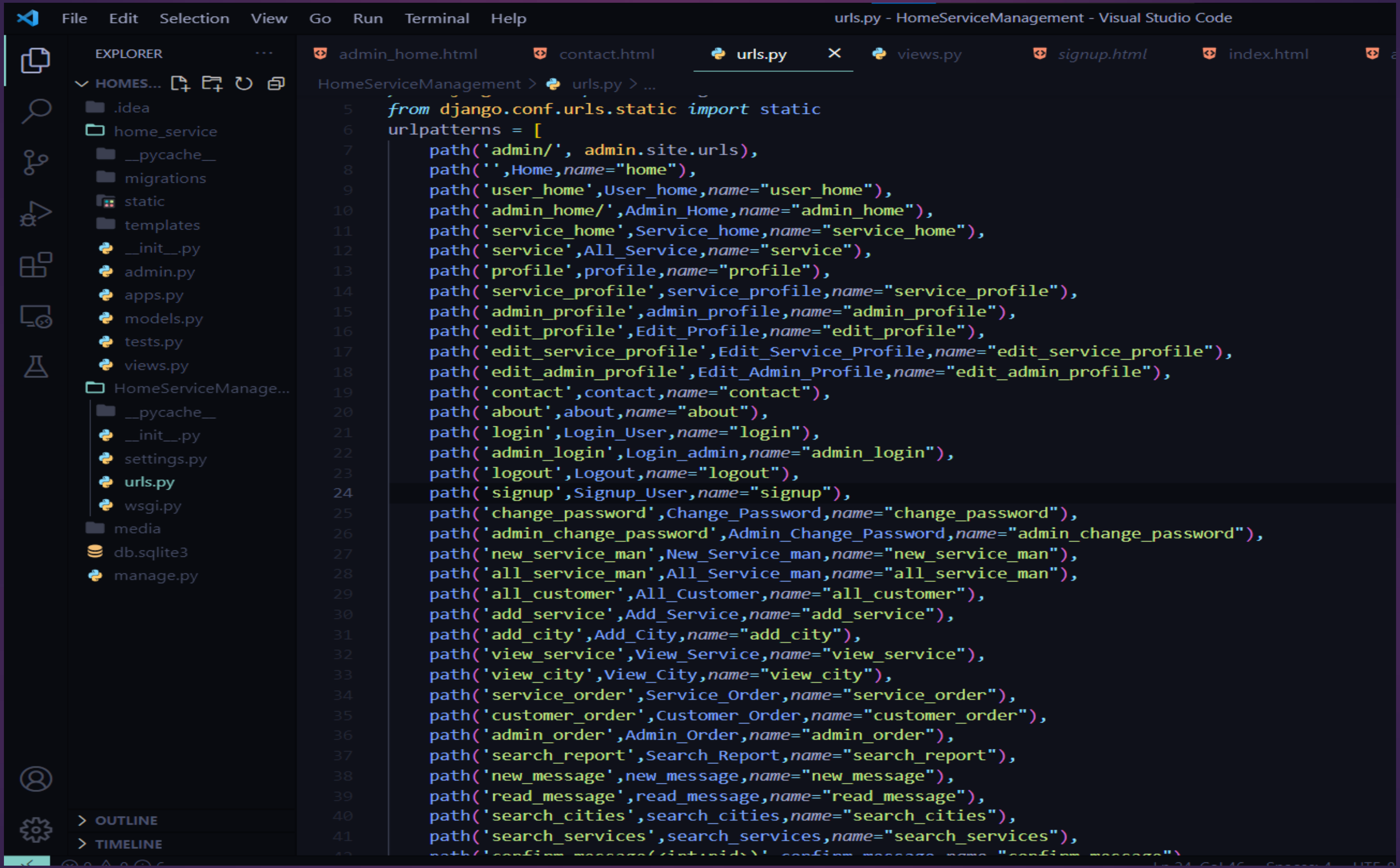


```python
def Login_admin(request):
    error = ""
    if request.method == "POST":
        u = request.POST['uname']
        p = request.POST['pwd']
        user = authenticate(username=u, password=p)
        if user.is_staff:
            login(request, user)
            error="pat"
        else:
            error="not"
    d = {'error': error}
    return render(request, 'admin_login.html', d)
```

5. Defining urls & views for each functionality

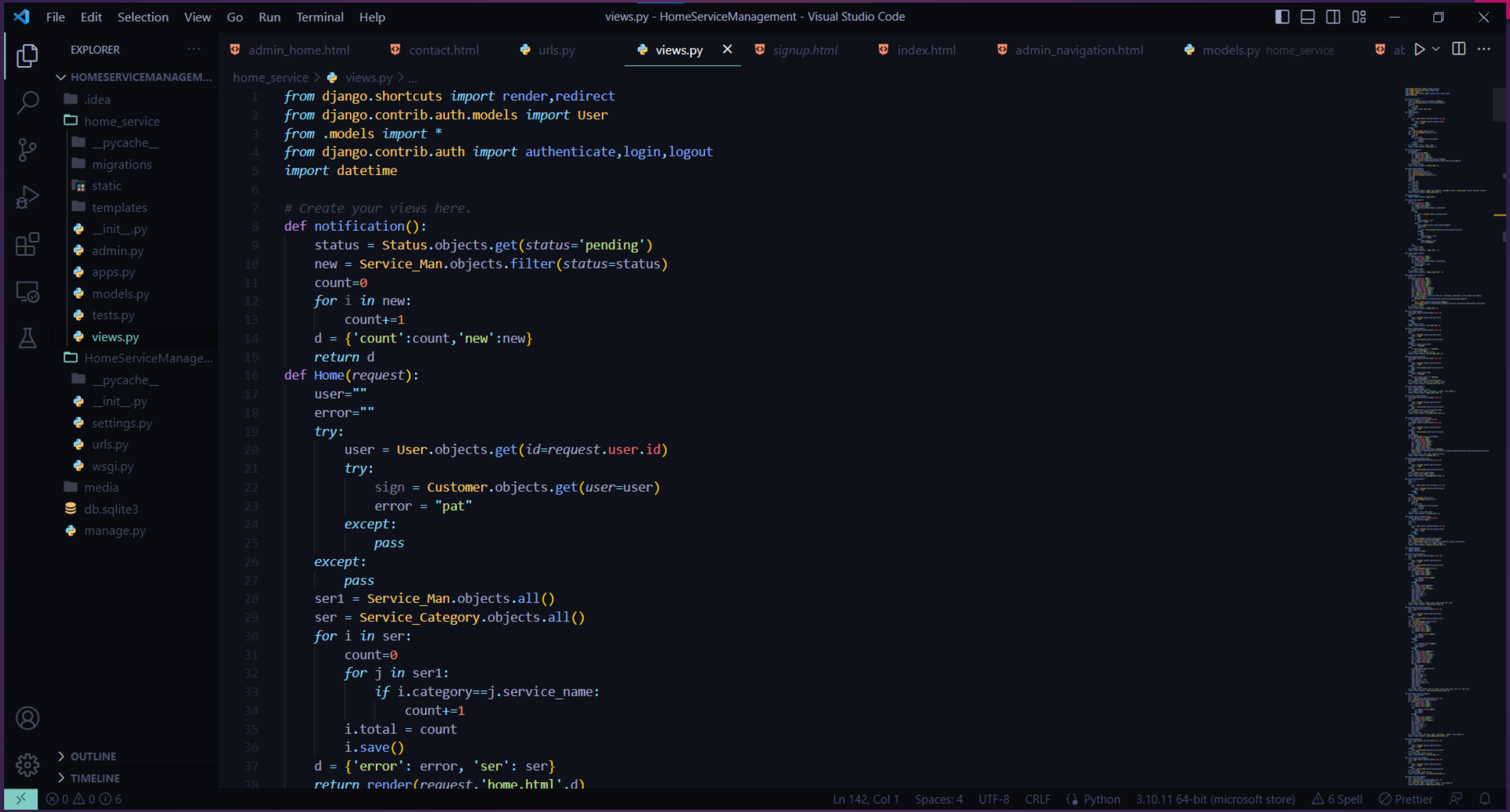
So basically, for every functionality like (login, signup, logout, booking, viewing services, profile etc. , we have first defined the URL which (on requested) will pass the request to corresponding view. That view will confirm the authentication of the user (whether the credentials are correct or not. If correct whether it is admin or customer or servicemen , accordingly it will fetch data from database and pass it to the template file (which is our front end).

Urls.py



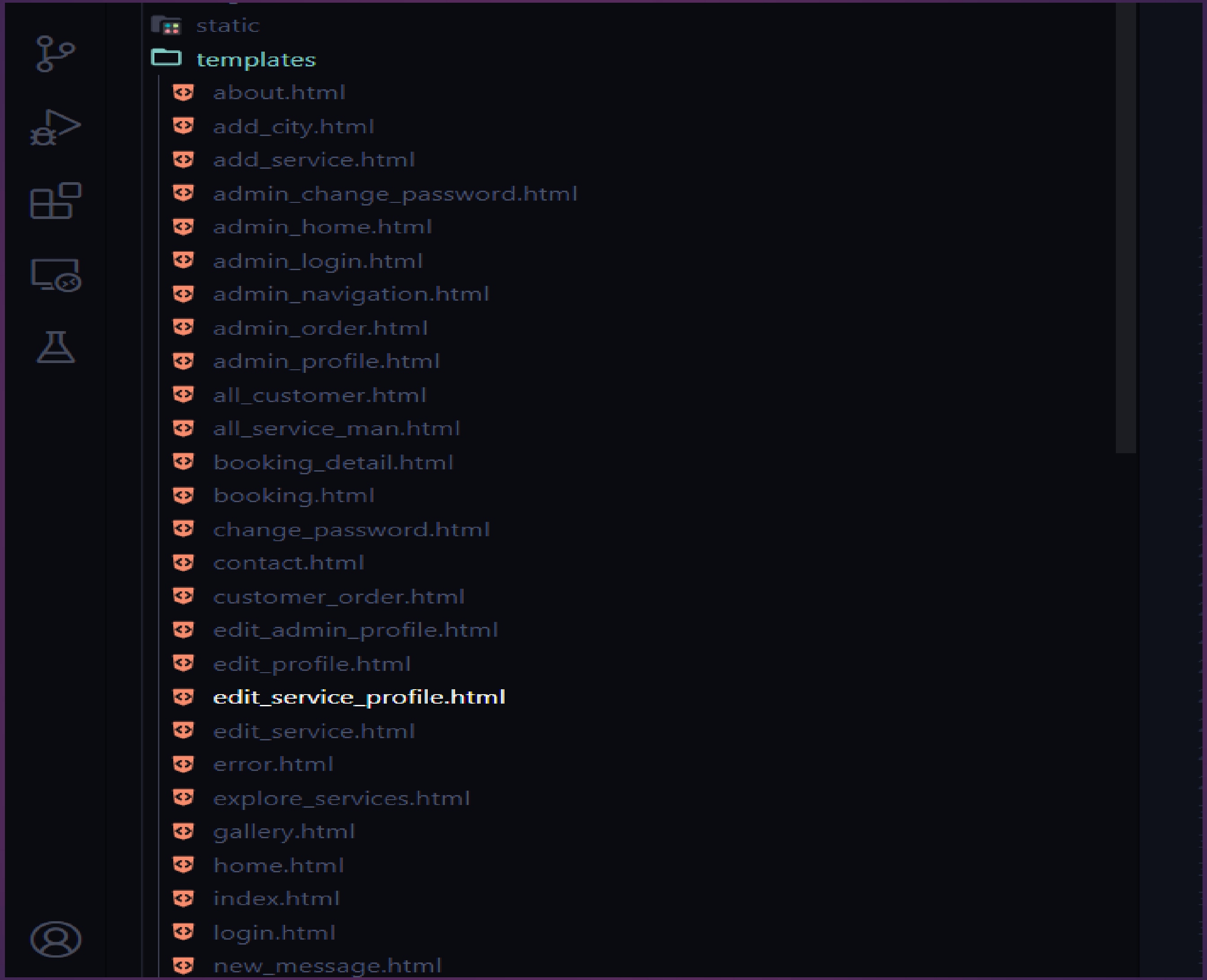
```python
from django.conf.urls.static import static
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',Home,name="home"),
    path('user_home',User_home,name="user_home"),
    path('admin_home/',Admin_Home,name="admin_home"),
    path('service_home',Service_home,name="service_home"),
    path('service',All_Service,name="service"),
    path('profile',profile,name="profile"),
    path('service_profile',service_profile,name="service_profile"),
    path('admin_profile',admin_profile,name="admin_profile"),
    path('edit_profile',Edit_Profile,name="edit_profile"),
    path('edit_service_profile',Edit_Service_Profile,name="edit_service_profile"),
    path('edit_admin_profile',Edit_Admin_Profile,name="edit_admin_profile"),
    path('contact',contact,name="contact"),
    path('about',about,name="about"),
    path('login',Login_User,name="login"),
    path('admin_login',Login_admin,name="admin_login"),
    path('logout',Logout,name="logout"),
    path('signup',Signup_User,name="signup"),
    path('change_password',Change_Password,name="change_password"),
    path('admin_change_password',Admin_Change_Password,name="admin_change_password"),
    path('new_service_man',New_Service_man,name="new_service_man"),
    path('all_service_man',All_Service_man,name="all_service_man"),
    path('all_customer',All_Customer,name="all_customer"),
    path('add_service',Add_Service,name="add_service"),
    path('add_city',Add_City,name="add_city"),
    path('view_service',View_Service,name="view_service"),
    path('view_city',View_City,name="view_city"),
    path('service_order',Service_Order,name="service_order"),
    path('customer_order',Customer_Order,name="customer_order"),
    path('admin_order',Admin_Order,name="admin_order"),
    path('search_report',Search_Report,name="search_report"),
    path('new_message',new_message,name="new_message"),
    path('read_message',read_message,name="read_message"),
    path('search_cities',search_cities,name="search_cities"),
    path('search_services',search_services,name="search_services"),
```

Views.py



```python
from django.shortcuts import render,redirect
from django.contrib.auth.models import User
from .models import *
from django.contrib.auth import authenticate,login,logout
import datetime

# Create your views here.
def notification():
    status = Status.objects.get(status='pending')
    new = Service_Man.objects.filter(status=status)
    count=0
    for i in new:
        count+=1
    d = {'count':count,'new':new}
    return d
def Home(request):
    user=""
    error=""
    try:
        user = User.objects.get(id=request.user.id)
        try:
            sign = Customer.objects.get(user=user)
            error = "pat"
        except:
            pass
    except:
        pass
    ser1 = Service_Man.objects.all()
    ser = Service_Category.objects.all()
    for i in ser:
        count=0
        for j in ser1:
            if i.category==j.service_name:
                count+=1
        i.total = count
        i.save()
    d = {'error': error, 'ser': ser}
    return render(request,'home.html',d)
```

Template.py


```
static
templates
  about.html
  add_city.html
  add_service.html
  admin_change_password.html
  admin_home.html
  admin_login.html
  admin_navigation.html
  admin_order.html
  admin_profile.html
  all_customer.html
  all_service_man.html
  booking_detail.html
  booking.html
  change_password.html
  contact.html
  customer_order.html
  edit_admin_profile.html
  edit_profile.html
  edit_service_profile.html
  edit_service.html
  error.html
  explore_services.html
  gallery.html
  home.html
  index.html
  login.html
  new_message.html
```

Live demonstration

http://127.0.0.1:8000/

Roles & Responsibilities

Register Number	Names	Roles
20BHI10033 20BCE10498 20BCE10845	Shivam Sharma Drishtavya Gupta Akshat Dadich	Backend, database model creation , handling views (Django)
20BCE10892 20BAI10327 20BCY10197	Deepanshu Pandey Yashaswi Patel Ayushi Srivastava	Frontend (html , CSS , JavaScript, bootstrap)
20BAI10299 20BAI10085	Rushil Bhatnagar Ayush Porwal	UI Design

FUTURE SCOPE

- Payment method can be made online
- WhatsApp integration with our application to receive booking details to both customers and servicemen.
- Servicemen feedback interface on portal

Thank you for your patience listening!