MYSORE UNIVERSITY SCHOOL OF ENGINEERING

Manasagangotri campus, Mysuru-570006 (Approved by AICTE, New Delhi)

UNIVERSITY OF MYSORE

Full Stack Development(21CD71) Assessment Report On:

"Develop a Contact Form System with URL Configuration and Custom Validation"

Under the guidance:
Mr. Karthik M N
Assistant Professor,
Department of Computer Science & Design,
MUSE.

Submitted by:
Rohan TR
Reg No: 21SECD35

Introduction:

In modern web applications, a Contact Form System plays a crucial role in facilitating communication between users and businesses. This project is designed to provide a seamless way for users to submit inquiries, while ensuring proper validation and data management.

The system includes the following key features:

✅ A contact form where users can submit their name, email, phone number, and message.

✅ Custom validation to ensure phone numbers contain exactly 10 digits and emails belong to a corporate domain (@company.com).

✅ Database storage for submitted messages, allowing admins to manage inquiries efficiently.

✅ Multiple URL configurations to separate different functionalities, such as form submission and admin message viewing.

Project overview:

Contact_form_project/

├── manage.py

├── db.sqlite3

├── contact_form_project/

│ ├── __init__.py

│ ├── settings.py

│ ├── urls.py

│ ├── asgi.py

│ ├── wsgi.py

├── contact/

│ ├── __init__.py

│ ├── admin.py

│ ├── apps.py

│ ├──forms.py

│ ├── models.py

│ ├── tests.py

│ ├── views.py

│ ├── urls.py

```
│   ├── templates/
│   │   ├── contact/
│   │   │   ├── contact_messages.html
│   │   │   ├──contact_form.html
│   │   │   ├──contact_success.html
```

Detailed steps Implementation:

Step 1: Install Django and Create a Virtual Environment

 # Create a virtual environment

>>python -m venv fsdlab

 # Activate the virtual environment

>>fsdlab\Scripts\activate

 # Install Django

>>pip install Django

Step 2: Create a Django Project

Run the following command to create a Django project:

>>django-admin startproject contact_form_project

>>cd contact_form_project

Step 3: Create a Django App

>>python manage.py startapp contact

Step 4: Configure settings.py

Open contact/settings.py and add 'contact′ to *INSTALLED_APPS*

Step 5: Create the Model:

```
from django.db import models

class ContactMessage(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=10)
    message = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.name} - {self.email}"
```

Run migrations to apply the model:

>>python manage.py makemigrations

>>python manage.py migrate

Step 6: Register the Model in Django Admin:

In contact/admin.py:

```
from django.contrib import admin
from .models import ContactMessage

@admin.register(ContactMessage)
class ContactMessageAdmin(admin.ModelAdmin):
    list_display = ('name', 'email', 'phone')
    search_fields = ('name', 'email', 'phone')
```

Step 7: Create Views for ecommerce:

In **contact/views.py**

```
from django.shortcuts import render, redirect
from .forms import ContactForm
from .models import ContactMessage
```

```
def contact_view(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)
        if form.is_valid():
            form.save()  # Save the contact message to the database
            return redirect('contact_success')  # Redirect to a success page
    else:
        form = ContactForm()  # Create an empty form instance
    return render(request, 'contact/contact_form.html', {'form': form})

def contact_success(request):
    return render(request, 'contact/contact_success.html')

def contact_messages(request):
    messages = ContactMessage.objects.all()  # Retrieve all contact messages
    return render(request, 'contact/contact_messages.html', {'messages': messages})
```

step 8: forms:

```
from django import forms
from .models import ContactMessage
import re

class ContactForm(forms.ModelForm):
    class Meta:
        model = ContactMessage
        fields = ['name', 'email', 'phone', 'message']

    def clean_phone(self):
```

```
        phone = self.cleaned_data.get('phone')
        if not phone.isdigit() or len(phone) != 10:
            raise forms.ValidationError("Phone number must contain exactly 10 digits.")
        return phone

    def clean_email(self):
        email = self.cleaned_data.get('email')
        if not email.endswith('@company.com'):
            raise forms.ValidationError("Only company emails (@company.com) are allowed.")
        return email
```

Step 9: Configure URLs:

Create **contact/urls.py**

```
from django.urls import path
from .views import contact_view, contact_success, contact_messages

urlpatterns = [
    path('', contact_view, name='contact_form'),
    path('success/', contact_success, name='contact_success'),
    path('messages/', contact_messages, name='contact_messages'),
]
```

Link the **contact** app to the project's main **urls.py** in **contact_form_project/urls.py**

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('contact/', include('contact.urls')),  # Include the contact app URLs
```

]

Step 10: Create HTML Templates:

Inside **contact/templates/contact/** create files naming: **contact_form.html**

```
<!DOCTYPE html>
<html>
<head>
   <title>Contact Form</title>
</head>
<body>
   <h2>Contact Us</h2>
   {% if messages %}
      {% for message in messages %}
         <p style="color: green;">{{ message }}</p>
      {% endfor %}
   {% endif %}
   <form method="post">
      {% csrf_token %}
      {{ form.as_p }}
      <button type="submit">Submit</button>
   </form>
</body>
</html>
```

Create file :Contact/template/contect_message.html

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Contact Messages</title>
</head>
<body>
    <h2>Submitted Messages</h2>
    <ul>
        {% for message in messages %}
            <li><strong>{{ message.name }}</strong> ({{ message.email }}) - {{
message.message }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

Contact/templates contact_success.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Success</title>
</head>
<body>
    <h1>Thank You!</h1>
    <p>Your message has been sent successfully.</p>
    <a href="{% url 'contact_form' %}">Go back to the contact form</a>
</body>
</html>
```

Step 11: Create a Superuser for Admin Panel:

>>python manage.py createsuperuser

Step 12: Run the Django Development Server

>>python manage.py runserver

Visit **127.0.0.1:8000/admin** and add products

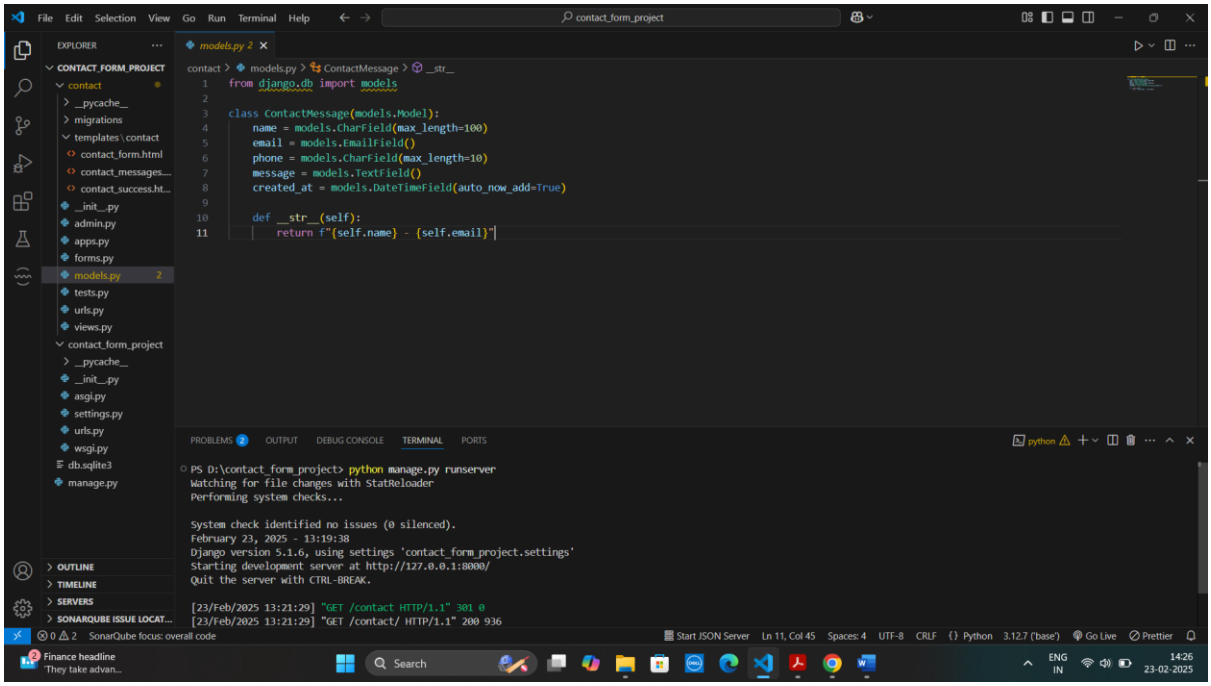
Visit **127.0.0.1:8000/products** to see the final output

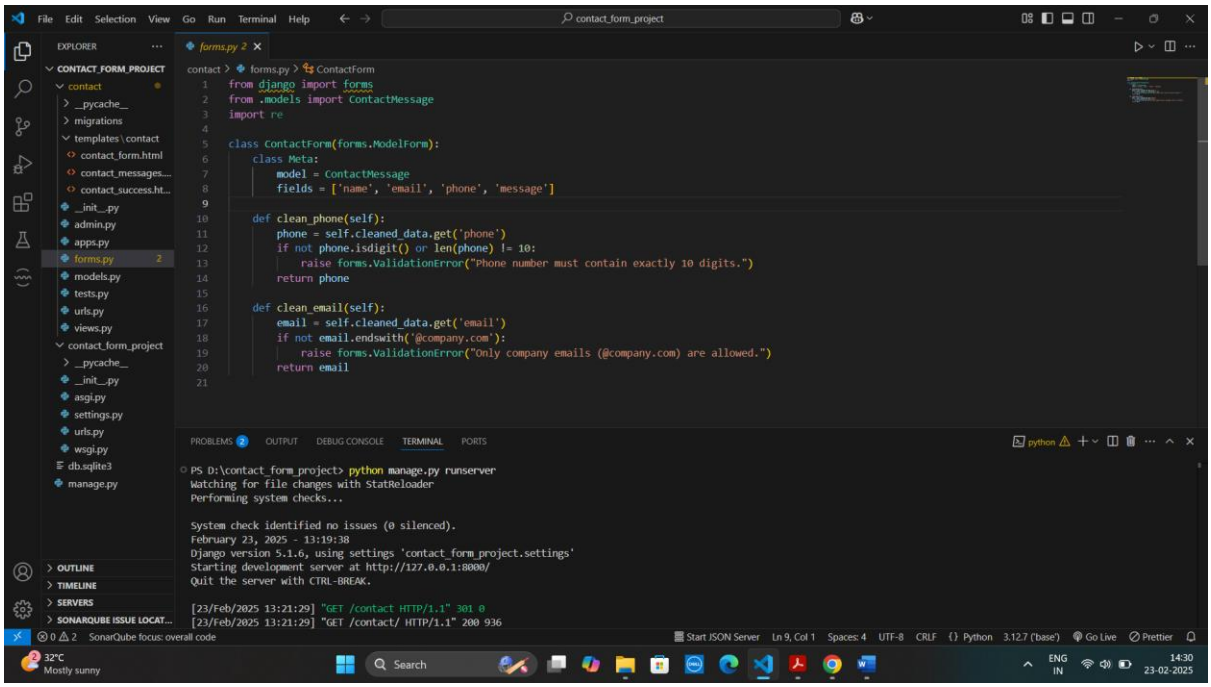
Conclusion

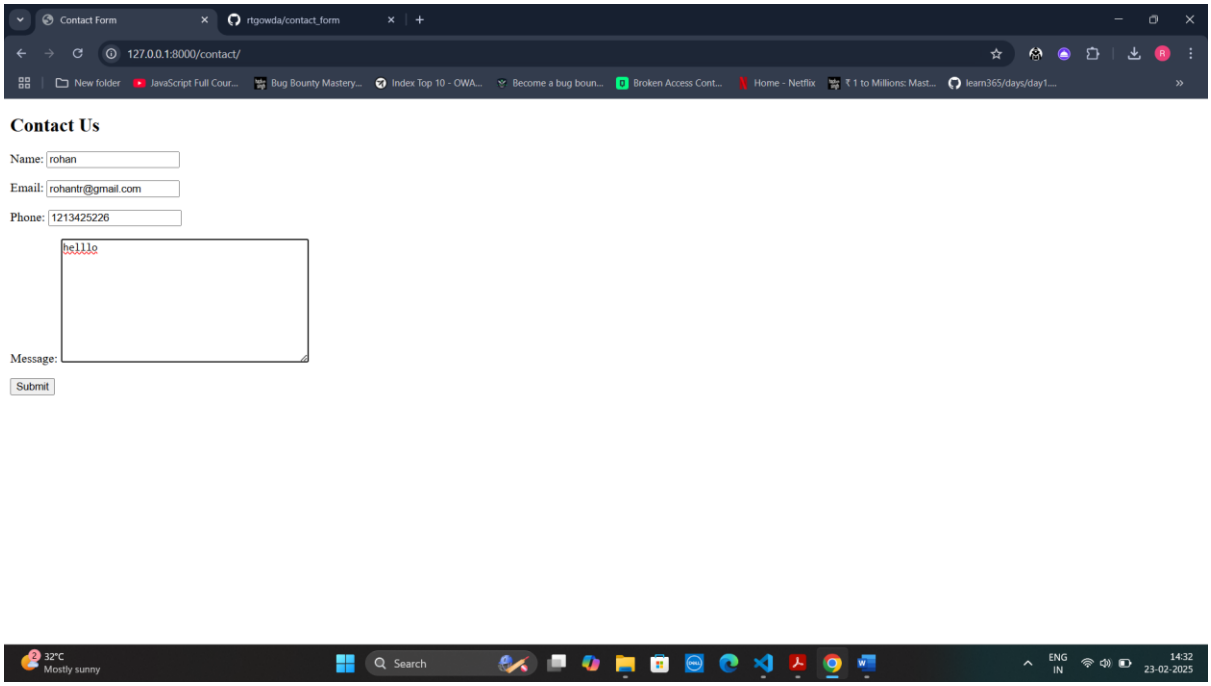
The Contact Form System provides a structured and efficient way to handle user inquiries while ensuring data validation and security. By implementing custom validation for phone numbers and corporate email domains, the system enhances data accuracy and prevents invalid submissions. Additionally, with database storage and an admin message viewing interface, businesses can efficiently manage and respond to customer inquiries.

Built using Django, this system is scalable, secure, and easily customizable to meet specific business needs. Whether integrated into a corporate website or a customer support portal, it streamlines communication and improves user engagement

Screenshots:


File Edit Selection View Go Run Terminal Help
contact_form_project
models.py 2
contact > models.py > ContactMessage > __str__
from django.db import models
class ContactMessage(models.Model):
name = models.CharField(max_length=100)
email = models.EmailField()
phone = models.CharField(max_length=10)
message = models.TextField()
created_at = models.DateTimeField(auto_now_add=True)
def __str__(self):
return f"{self.name} - {self.email}"
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\contact_form_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
February 23, 2025 - 13:19:38
Django version 5.1.6, using settings 'contact_form_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[23/Feb/2025 13:21:29] "GET /contact HTTP/1.1" 301 0
[23/Feb/2025 13:21:29] "GET /contact/ HTTP/1.1" 200 936


File Edit Selection View Go Run Terminal Help
contact_form_project
forms.py 2
contact > forms.py > ContactForm
from django import forms
from .models import ContactMessage
import re
class ContactForm(forms.ModelForm):
class Meta:
model = ContactMessage
fields = ['name', 'email', 'phone', 'message']
def clean_phone(self):
phone = self.cleaned_data.get('phone')
if not phone.isdigit() or len(phone) != 10:
raise forms.ValidationError("Phone number must contain exactly 10 digits.")
return phone
def clean_email(self):
email = self.cleaned_data.get('email')
if not email.endswith('@company.com'):
raise forms.ValidationError("Only company emails (@company.com) are allowed.")
return email
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\contact_form_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
February 23, 2025 - 13:19:38
Django version 5.1.6, using settings 'contact_form_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[23/Feb/2025 13:21:29] "GET /contact HTTP/1.1" 301 0
[23/Feb/2025 13:21:29] "GET /contact/ HTTP/1.1" 200 936


Contact Form
rtgowda/contact_form
127.0.0.1:8000/contact/
New folder
JavaScript Full Cour...
Bug Bounty Mastery...
Index Top 10 - OWA...
Become a bug boun...
Broken Access Cont...
Home - Netflix
₹ 1 to Millions: Mast...
learn365/days/day1....
Contact Us
Name: rohan
Email: rohantr@gmail.com
Phone: 1213425226
Message: hellllo
Submit
32°C
Mostly sunny
Search
ENG
IN
14:32
23-02-2025