mark_registration_system > mark_registration_system

> settings.py

```python
"""
Django settings for mark_registration_system project.

Generated by 'django-admin startproject' using Django 5.1.3.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.1/ref/settings/
"""

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-7*6f(@uzvw#paz%1ujdjlx-uu#)%e+90v4=j2g$^*@)-qw3(on'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'marks',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'mark_registration_system.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            BASE_DIR / 'templates',  # If you want to store global templates in a folder at the project level
        ],
        'APP_DIRS': True,  # Make sure this is set to True to allow Django to search in app-specific templates
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]


WSGI_APPLICATION = 'mark_registration_system.wsgi.application'
```

```python
# Database
# https://docs.djangoproject.com/en/5.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/5.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True
```

```python
USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.1/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

# Define the directory where media files will be saved
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Define the URL path to access media files
MEDIA_URL = '/media/'
```

> urls.py

```python
"""
URL configuration for mark_registration_system project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
```

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

> templates > marks

> home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}Marks Registration System{% endblock %}</title>
    <!-- Link to Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <!-- Custom Styles -->
    <style>
        /* Navigation link hover effect */
        .nav-link:hover {
            color: #0056b3; /* Darker blue on hover */
            text-decoration: underline; /* Adds underline on hover */
        }

        /* Button hover effect */
        .btn:hover {
            background-color: #004085; /* Change background color on hover */
            color: #fff; /* Change text color on hover */
        }
    </style>
</head>
<body>
    <!-- Header Section -->
    <header class="bg-primary text-white text-center py-3">
        <h1>Welcome to the Marks Registration System</h1>
    </header>
    <!-- Navigation Bar -->
    <nav class="my-4">
        <ul class="nav justify-content-center">
            <li class="nav-item">
                <a class="nav-link text-primary" href="{% url
'home' %}">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-primary" href="{% url
'input_marks' %}">Input Marks</a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-primary" href="{% url
'update_marks' %}">Update Marks</a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-primary" href="{% url
'view_marks' %}">View Marks</a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-primary" href="{% url
'visualization' %}">Visualization</a>
            </li>
        </ul>
    </nav>

    <!-- Main Content Section -->
    <main class="container text-center">
        {% block content %}
        <p class="mb-4">Manage student marks efficiently and visualize their
performance.</p>

        <div class="row justify-content-center">
            <!-- Card for Number of Students -->
            <div class="col-md-4">
                <div class="card m-2">
                    <div class="card-body">
                        <h5 class="card-title">No. of Students</h5>
                        <p class="card-text">{{ student_count }}</p>
                    </div>
                </div>
            </div>
            <!-- Card for Number of Modules -->
            <div class="col-md-4">
                <div class="card m-2">
                    <div class="card-body">
                        <h5 class="card-title">No. of Modules</h5>
                        <p class="card-text">{{ module_count }}</p>
                    </div>
                </div>
            </div>
        </div>
        {% endblock %}
    </main>

    <!-- Link to Bootstrap JS -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.
js"></script>
</body>
</html>
```

> input_marks.html

```
{% extends "marks/home.html" %}

{% block content %}
<h2>Input Marks</h2>

<!-- Display Messages -->
{% if messages %}
    {% for message in messages %}
        <div class="alert alert-success" role="alert">
            {{ message }}
        </div>
    {% endfor %}
{% endif %}

<form method="post" class="mt-4" autocomplete="off">
    {% csrf_token %}
    <div class="mb-3">
        {{ form.module_code.label_tag }}                <!--  -->
        {{ form.module_code }}
        {% if form.module_code.errors %}
            <div class="text-danger">{{ form.module_code.errors }}</div>
        {% endif %}
    </div>
    <div class="mb-3">
        {{ form.module_name.label_tag }}
        {{ form.module_name }}
        {% if form.module_name.errors %}
            <div class="text-danger">{{ form.module_name.errors }}</div>
        {% endif %}
    </div>
    <div class="mb-3">
        {{ form.cw1_marks.label_tag }}
        {{ form.cw1_marks }}
        {% if form.cw1_marks.errors %}
            <div class="text-danger">{{ form.cw1_marks.errors }}</div>
        {% endif %}
    </div>
    <div class="mb-3">
        {{ form.cw2_marks.label_tag }}
        {{ form.cw2_marks }}
        {% if form.cw2_marks.errors %}
            <div class="text-danger">{{ form.cw2_marks.errors }}</div>
        {% endif %}
    </div>
    <div class="mb-3">
        {{ form.cw3_marks.label_tag }}
        {{ form.cw3_marks }}
        {% if form.cw3_marks.errors %}
            <div class="text-danger">{{ form.cw3_marks.errors }}</div>
        {% endif %}
    </div>
    <div class="mb-3">
        {{ form.student_id.label_tag }}
        {{ form.student_id }}
        {% if form.student_id.errors %}
            <div class="text-danger">{{ form.student_id.errors }}</div>
        {% endif %}
    </div>
    <div class="mb-3">
        {{ form.student_name.label_tag }}
        {{ form.student_name }}
        {% if form.student_name.errors %}
```

```
            <div class="text-danger">{{ form.student_name.errors }}</div>
        {% endif %}
    </div>
    <div class="mb-3">
        {{ form.gender.label_tag }}
        {{ form.gender }}
        {% if form.gender.errors %}
            <div class="text-danger">{{ form.gender.errors }}</div>
        {% endif %}
    </div>
    <div class="mb-3">
        {{ form.date_of_entry.label_tag }}
        {{ form.date_of_entry }}
        {% if form.date_of_entry.errors %}
            <div class="text-danger">{{ form.date_of_entry.errors }}</div>
        {% endif %}
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
    <button type="reset" class="btn btn-secondary">Reset</button>
</form>
{% endblock %}
```

> update_marks.html

```
{% extends "marks/home.html" %}

{% block content %}
<h2>Update Marks</h2>

{% if messages %}
    {% for message in messages %}
        <div class="alert alert-success" role="alert">
            {{ message }}
        </div>
    {% endfor %}
{% endif %}

<!-- Search Form for Module Code -->
<form method="get" class="mb-4">
    <div class="mb-3">
        <label for="module_code" class="form-label">Module Code:</label>
        <input
            type="text"
            name="module_code"
            id="module_code"
            class="form-control"
            placeholder="Enter module code"
            value="{{ module_code|default:'' }}"
        >
    </div>
    <button type="submit" class="btn btn-primary">View</button>
</form>

{% if records %}
    <h3>Student Records for Module Code: {{ module_code }}</h3>
    <table class="table table-striped">
        <thead>
            <tr>
                <th>Student ID</th>
                <th>Student Name</th>
                <th>CW1</th>
                <th>CW2</th>
                <th>CW3</th>
                <th>Total</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            {% for record in records %}
                <tr>
                    <td>{{ record.student_id }}</td>
                    <td>{{ record.student_name }}</td>
                    <td>{{ record.cw1_marks }}</td>
                    <td>{{ record.cw2_marks }}</td>
                    <td>{{ record.cw3_marks }}</td>
                    <td>{{ record.total_marks }}</td>  <!-- Display the
dynamically calculated total marks -->
                    <td>
                        <a
href="?module_code={{ module_code }}&student_id={{ record.id }}" class="btn
btn-warning">Modify</a>
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>

    {% if student_id %}
        <h4>Edit Marks for Student ID: {{ student_id }}</h4>
        <form method="post" action="">
            {% csrf_token %}
            <div class="mb-3">
                <label for="cw1_marks" class="form-label">Coursework 1
Marks:</label>
                <input type="number" name="cw1_marks" class="form-control"
value="{{ records.first.cw1_marks }}">
            </div>
            <div class="mb-3">
                <label for="cw2_marks" class="form-label">Coursework 2
Marks:</label>
                <input type="number" name="cw2_marks" class="form-control"
value="{{ records.first.cw2_marks }}">
            </div>
            <div class="mb-3">
                <label for="cw3_marks" class="form-label">Coursework 3
Marks:</label>
                <input type="number" name="cw3_marks" class="form-control"
value="{{ records.first.cw3_marks }}">
            </div>
            <button type="submit" class="btn btn-primary">Update Marks</button>
        </form>
    {% endif %}
{% elif module_code %}
    <p>No records found for module code "{{ module_code }}".</p>
{% endif %}
{% endblock %}
```

> view_marks.html

```
{% extends "marks/home.html" %}

{% block content %}
<h2>View Marks</h2>

<form method="get" class="mb-4">
    <div class="mb-3">
        <label for="module_code" class="form-label">Module Code:</label>
        <input
            type="text"
            name="module_code"
            id="module_code"
            class="form-control"
            placeholder="Enter module code"
            value="{{ module_code|default:'' }}"
        >
    </div>
    <button type="submit" class="btn btn-primary">View</button>
</form>

{% if records %}
    <table class="table table-striped">
        <thead>
            <tr>
                <th>Student ID</th>
                <th>Student Name</th>
                <th>CW1</th>
                <th>CW2</th>
                <th>CW3</th>
                <th>Total</th>
            </tr>
        </thead>
        <tbody>
            {% for record in records %}
                <tr>
                    <td>{{ record.student_id }}</td>
                    <td>{{ record.student_name }}</td>
                    <td>{{ record.cw1_marks }}</td>
                    <td>{{ record.cw2_marks }}</td>
                    <td>{{ record.cw3_marks }}</td>

<td>{{ record.cw1_marks|add:record.cw2_marks|add:record.cw3_marks }}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% elif module_code %}
    <p>No records found for module code "{{ module_code }}".</p>
{% endif %}
{% endblock %}
```

> visualization.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Visualizations</title>

    <style>
        body {
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
            margin: 0;
            font-family: Arial, sans-serif;
            text-align: center; /* Center-align text */
        }
        h1, h2 {
            margin: 10px 0;
        }
        img {
            max-width: 80%; /* Make charts responsive */
            height: auto;
            margin: 20px 0;
        }
    </style>

</head>
<body>
    <h1>Data Visualizations</h1>

    {% if charts.pie_chart %}
        <h2>Pie Chart: Marks Distribution</h2>
        <img src="data:image/png;base64,{{ charts.pie_chart }}" alt="Pie
Chart">
    {% endif %}

    {% if charts.bar_chart %}
```

```
        <h2>Bar Chart: Average Marks by Module</h2>
        <img src="data:image/png;base64,{{ charts.bar_chart }}" alt="Bar
Chart">
    {% endif %}

    {% if charts.scatter_plot %}
        <h2>Scatter Plot: CW1 vs CW2 Marks</h2>
        <img src="data:image/png;base64,{{ charts.scatter_plot }}" alt="Scatter
Plot">
    {% endif %}
</body>
</html>
```

> apps.py

```
from django.apps import AppConfig


class MarksConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'marks'
```

```
from django import forms

class MarkEntryForm(forms.Form):
    module_code = forms.CharField(max_length=20, label="Module Code")
    module_name = forms.CharField(max_length=100, label="Module Name")
    cw1_marks = forms.IntegerField(label="Coursework 1 Marks")
    cw2_marks = forms.IntegerField(label="Coursework 2 Marks")
    cw3_marks = forms.IntegerField(label="Coursework 3 Marks")
    student_id = forms.CharField(max_length=20, label="Student ID")
    student_name = forms.CharField(max_length=100, label="Student Name")
    gender = forms.ChoiceField(choices=[('M', 'Male'), ('F', 'Female')],
label="Gender")
    date_of_entry = forms.DateField(widget=forms.DateInput(attrs={'type':
'date'}), label="Date of Entry")
```

```
#Ensure that coursework marks are within a valid range (e.g., 0-100).
    def clean_cw1_marks(self):
        cw1 = self.cleaned_data.get('cw1_marks')
        if cw1 < 0 or cw1 > 100:
            raise forms.ValidationError("Coursework 1 Marks must be between 0 
and 100.")
        return cw1

    def clean_cw2_marks(self):
        cw2 = self.cleaned_data.get('cw2_marks')
        if cw2 < 0 or cw2 > 100:
            raise forms.ValidationError("Coursework 2 Marks must be between 0 
and 100.")
        return cw2

    def clean_cw3_marks(self):
        cw3 = self.cleaned_data.get('cw3_marks')
        if cw3 < 0 or cw3 > 100:
            raise forms.ValidationError("Coursework 3 Marks must be between 0 
and 100.")
        return cw3
```

> models.py

```
from django.db import models

# Create your models here.

class Marks(models.Model):
    module_code = models.CharField(max_length=20)
    module_name = models.CharField(max_length=100)
    cw1_marks = models.IntegerField()
    cw2_marks = models.IntegerField()
    cw3_marks = models.IntegerField()
    student_id = models.CharField(max_length=20)
    student_name = models.CharField(max_length=100)
    gender = models.CharField(max_length=1, choices=[('M', 'Male'), ('F', 
'Female')])
    date_of_entry = models.DateField()
    def __str__(self):
        return f"{self.student_name} - {self.module_code}"
```

> urls.py

```
from django.conf import settings
from django.conf.urls.static import static
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('input_marks/', views.input_marks, name='input_marks'),
    path('update_marks/', views.update_marks, name='update_marks'),
    path('view_marks/', views.view_marks, name='view_marks'),
    path('visualization/', views.visualization, name='visualization'),
]

# Serve media files in development (not for production use)
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

> views.py

```
import csv
import os
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib import messages  # Import messages
from .forms import MarkEntryForm
from .models import Marks  # Import the Marks model
from django.conf import settings
from django.db.models import Sum
import matplotlib.pyplot as plt
from io import BytesIO
import base64
import pandas as pd

import matplotlib
```

```python
matplotlib.use('Agg')  # Use a non-GUI backend



# Helper function to save data to a CSV file
def save_to_csv():
    csv_file = os.path.join(settings.MEDIA_ROOT, 'marks_entries.csv')  # Save
in media folder

    # Open CSV to rewrite with the updated records
    with open(csv_file, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['Student ID', 'Student Name', 'Module Code', 'Module
Name', 'CW1 Marks', 'CW2 Marks', 'CW3 Marks', 'Total Marks'])

        # Write each record to the CSV (including updated records)
        for mark in Marks.objects.all():  # Fetch all records from the database
            total_marks = mark.cw1_marks + mark.cw2_marks + mark.cw3_marks  #
Calculate total marks
            writer.writerow([
                mark.student_id,
                mark.student_name,
                mark.module_code,
                mark.module_name,
                mark.cw1_marks,
                mark.cw2_marks,
                mark.cw3_marks,
                total_marks
            ])

# Helper function to generate charts as Base64 strings
def generate_chart(plt):
    """Convert a Matplotlib plot to a Base64 image."""
    buf = BytesIO()
    plt.savefig(buf, format='png')
    buf.seek(0)
    image_base64 = base64.b64encode(buf.read()).decode('utf-8')
    buf.close()
    plt.close()  # Close the plot to avoid overlapping visuals
    return image_base64

# Home page view
def home(request):
    # Fetch counts for records display
    student_count = Marks.objects.values('student_id').distinct().count()
    module_count = Marks.objects.values('module_code').distinct().count()
    context = {
        'student_count': student_count,
        'module_count': module_count,
    }
    return render(request, 'marks/home.html', context)

# Input marks page view
def input_marks(request):
    if request.method == 'POST':
        form = MarkEntryForm(request.POST)
        if form.is_valid():
            # Save form data to the database
            new_mark = Marks.objects.create(
                module_code=form.cleaned_data['module_code'],
                module_name=form.cleaned_data['module_name'],
                cw1_marks=form.cleaned_data['cw1_marks'],
                cw2_marks=form.cleaned_data['cw2_marks'],
                cw3_marks=form.cleaned_data['cw3_marks'],
                student_id=form.cleaned_data['student_id'],
                student_name=form.cleaned_data['student_name'],
                gender=form.cleaned_data['gender'],
                date_of_entry=form.cleaned_data['date_of_entry'],
            )

            # Save the new entry to CSV
            save_to_csv()  # Rewrite the CSV with the latest data

            # Add a success message
            messages.success(request, 'Marks have been successfully recorded.')
            # Redirect to the same page to display the message and reset the
form
            return redirect('input_marks')
    else:
        form = MarkEntryForm()
    return render(request, 'marks/input_marks.html', {'form': form})

# Update marks page view
def update_marks(request):
    records = None
    module_code = request.GET.get('module_code')  # Get module_code from the
query parameter
```

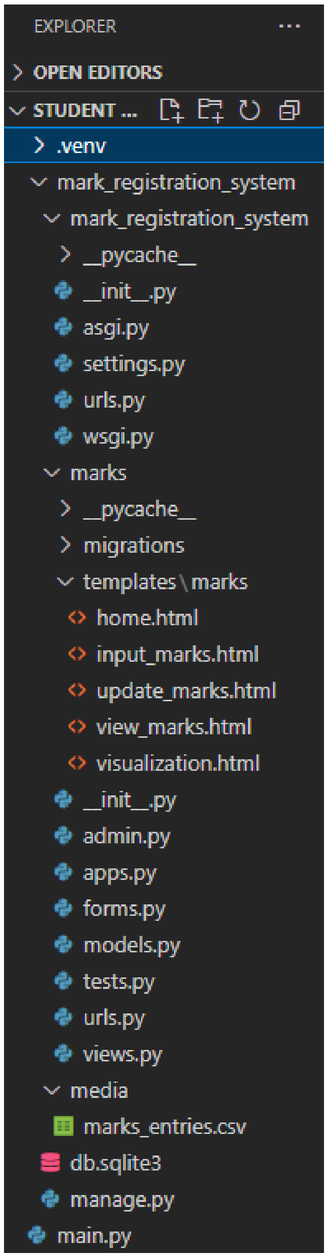
```python
    student_id = request.GET.get('student_id')  # Get student_id if modifying a
specific student's marks

    if request.method == 'POST' and student_id:
        # Fetch the student record to update
        student = get_object_or_404(Marks, id=student_id)

        # Update the marks based on the form input
        student.cw1_marks = request.POST.get('cw1_marks')
        student.cw2_marks = request.POST.get('cw2_marks')
        student.cw3_marks = request.POST.get('cw3_marks')
        student.save()  # Save the updated record

        # After updating the marks, save the updated data to CSV
        save_to_csv()  # Rewrite the entire CSV file with updated records

        messages.success(request, 'Marks updated successfully.')
        return redirect('update_marks')  # Redirect back to the same page

    if module_code:
        # Fetch all records with the given module_code
        records = Marks.objects.filter(module_code=module_code)

        # Calculate the total marks for each record and add it to the record
object
        for record in records:
            record.total_marks = record.cw1_marks + record.cw2_marks +
record.cw3_marks

    return render(request, 'marks/update_marks.html', {'records': records,
'module_code': module_code, 'student_id': student_id})

# View marks page view
def view_marks(request):
    module_code = request.GET.get('module_code')
    records = None
    if module_code:
        records = Marks.objects.filter(module_code=module_code)
    return render(request, 'marks/view_marks.html', {'records': records,
'module_code': module_code})
```

```
custom_colors = [
    #'red', 'coral', 'orange', 'gold', 'yellow', 'greenyellow', 'lime',
    'aquamarine', 'turquoise', 'cyan', 'deepskyblue', 'royalblue',
    'blue'#, 'navy', 'blueviolet', 'violet', 'fuchsia', 'deeppink', 'pink'
]

# Visualization page view
def visualization(request):
    # Fetch data
    marks = Marks.objects.all().values('module_name', 'cw1_marks', 'cw2_marks',
'cw3_marks')
    df = pd.DataFrame(marks)

    charts = {}

    if not df.empty:
        # 1. Pie Chart: Distribution of CW1, CW2, CW3 Marks
        total_cw1 = df['cw1_marks'].sum()
        total_cw2 = df['cw2_marks'].sum()
        total_cw3 = df['cw3_marks'].sum()
        plt.figure(figsize=(6, 6))
        plt.pie([total_cw1, total_cw2, total_cw3], labels=['CW1', 'CW2',
'CW3'], autopct='%1.1f%%', startangle=90, colors=custom_colors[1:4])  # Limit
to 3 colors(3CW))
        plt.title('Marks Distribution')
        charts['pie_chart'] = generate_chart(plt)

        # 2. Bar Chart: Average Marks by Module
        avg_marks = df.groupby('module_name')[['cw1_marks', 'cw2_marks',
'cw3_marks']].mean()
        avg_marks.plot(kind='bar', figsize=(9, 7),
color=custom_colors[:len(avg_marks)]) # Limit to the number of bars
        plt.title('Average Marks per Module')
        plt.xlabel('Modules')
        plt.ylabel('Average Marks')
        plt.xticks(rotation=45)  # Rotate labels
        plt.tight_layout()  # Adjust layout to prevent overlap
        charts['bar_chart'] = generate_chart(plt)

        # 3. Scatter Plot: CW1 vs. CW2 Marks
        plt.figure(figsize=(8, 6))
        plt.scatter(df['cw1_marks'], df['cw2_marks'], alpha=0.7,
color=custom_colors[5])  # Use the first color in the list
```

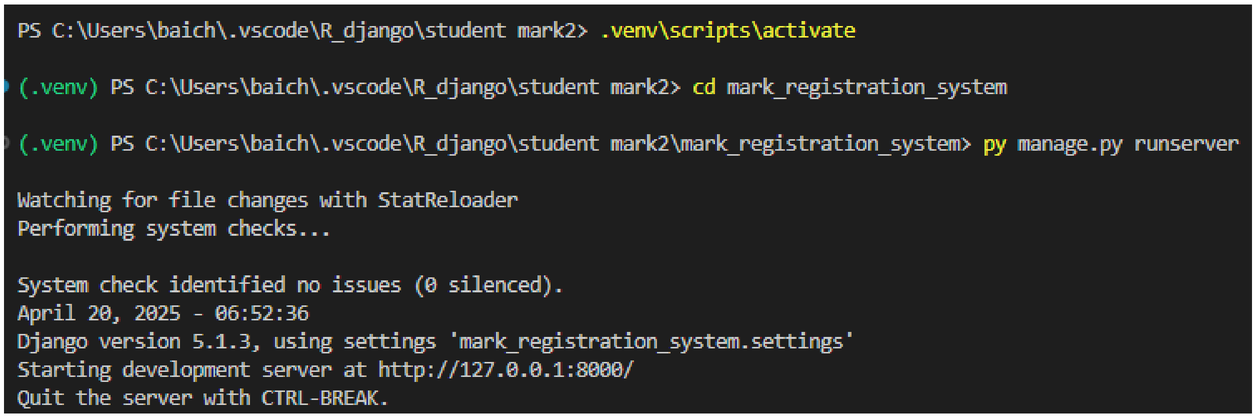
```
        plt.title('CW1 vs. CW2 Marks')
        plt.xlabel('CW1 Marks')
        plt.ylabel('CW2 Marks')
        charts['scatter_plot'] = generate_chart(plt)

    return render(request, 'marks/visualization.html', {'charts': charts})
```

marks_entries.csv

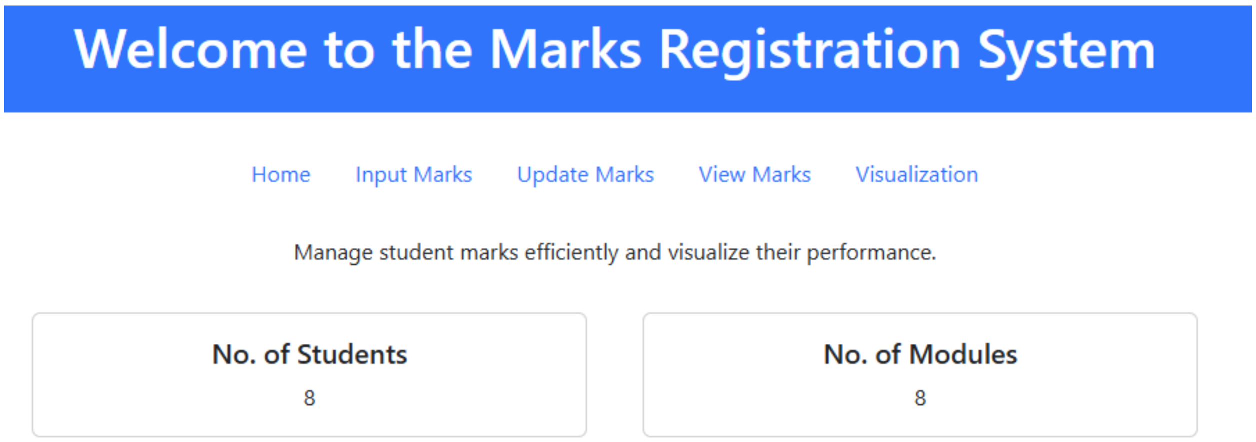
```
Student ID,Student Name,Module Code,Module Name,CW1 Marks,CW2 Marks,CW3
Marks,Total Marks
X1234,Fara Baichoo,A101,ICT,11,27,10,48
X1235,Ajmhul Baichoo,A102,BDA,14,25,35,74
X1236,Haroon Rasheed,A103,DCY,10,28,35,73
X1237,Kavina Narrainen,A104,Python,15,10,20,45
X1238,Seif Al Din,A105,IOT,15,10,25,50
X1239,Shaheen Sobhun,A106,Networking,20,36,15,71
X1244,Aureli Vencatachellum,A107,Soft Skills,15,30,25,70
X1224,Widaad Googoolee,A108,ACCA,20,30,10,60
```



```
EXPLORER
> OPEN EDITORS
v STUDENT ...
  > .venv
  v mark_registration_system
    v mark_registration_system
      > __pycache__
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
    v marks
      > __pycache__
      > migrations
      v templates \ marks
        home.html
        input_marks.html
        update_marks.html
        view_marks.html
        visualization.html
      __init__.py
      admin.py
      apps.py
      forms.py
      models.py
      tests.py
      urls.py
      views.py
    v media
      marks_entries.csv
    db.sqlite3
    manage.py
  main.py
```



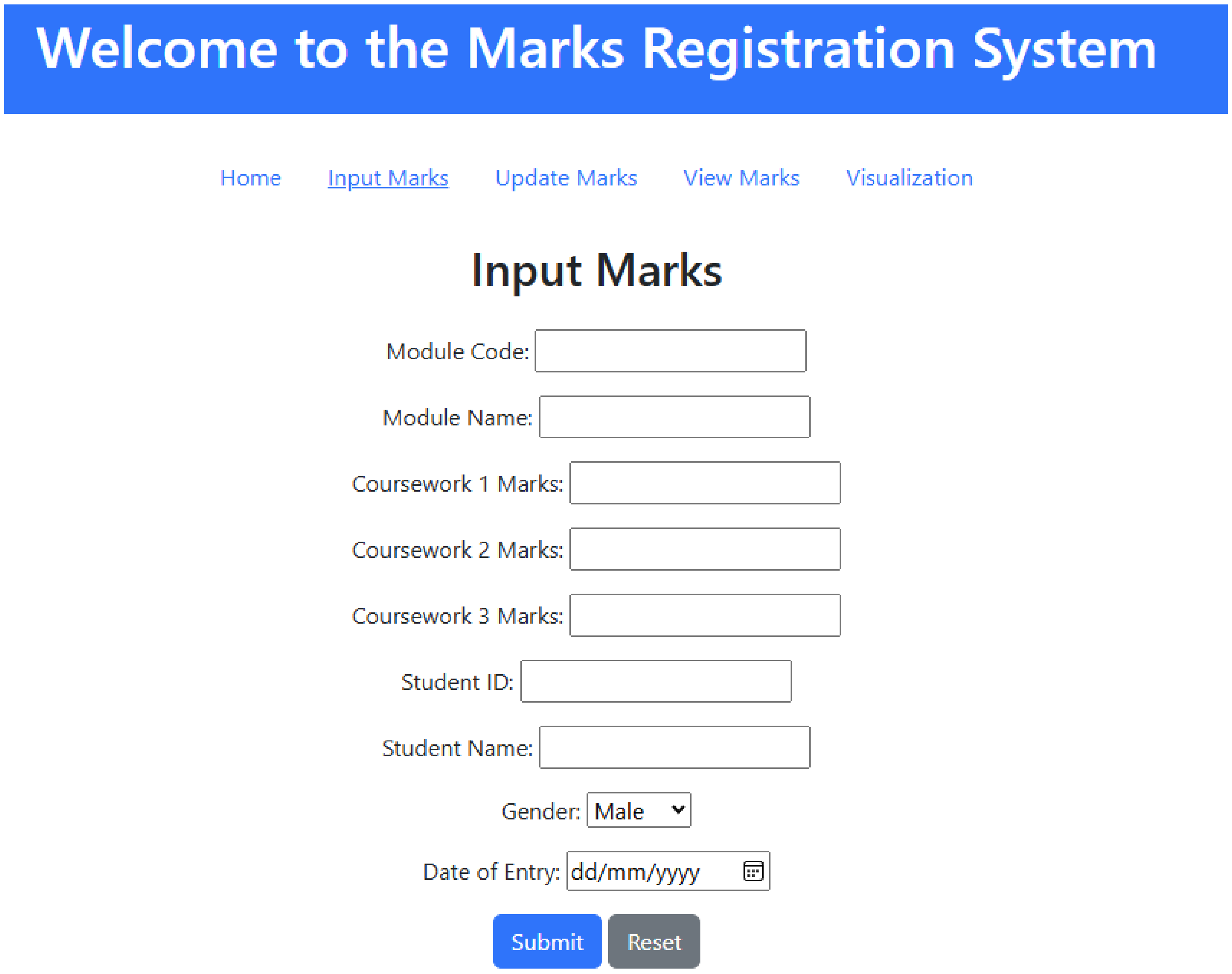
```
PS C:\Users\baich\.vscode\R_django\student mark2> .venv\scripts\activate

(.venv) PS C:\Users\baich\.vscode\R_django\student mark2> cd mark_registration_system

(.venv) PS C:\Users\baich\.vscode\R_django\student mark2\mark_registration_system> py manage.py runserver

Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 20, 2025 - 06:52:36
Django version 5.1.3, using settings 'mark_registration_system.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```


Welcome to the Marks Registration System

Home Input Marks Update Marks View Marks Visualization

Manage student marks efficiently and visualize their performance.

No. of Students	No. of Modules
8	8

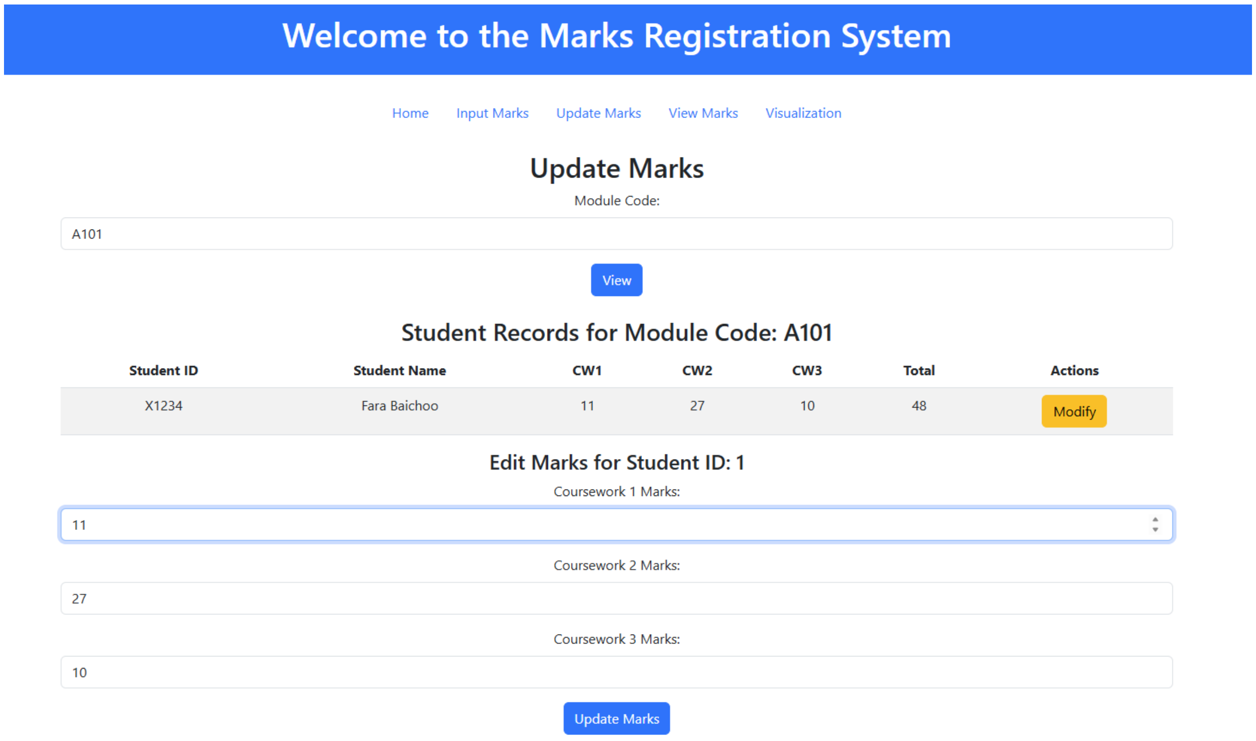


Welcome to the Marks Registration System

Home Input Marks Update Marks View Marks Visualization

Input Marks

Module Code:	
Module Name:	
Coursework 1 Marks:	
Coursework 2 Marks:	
Coursework 3 Marks:	
Student ID:	
Student Name:	
Gender:	Male
Date of Entry:	dd/mm/yyyy



Welcome to the Marks Registration System

Home Input Marks Update Marks View Marks Visualization

Update Marks

Module Code:

A101

View

Student Records for Module Code: A101

Student ID	Student Name	CW1	CW2	CW3	Total	Actions
X1234	Fara Baichoo	11	27	10	48	Modify

Edit Marks for Student ID: 1

Coursework 1 Marks:

11

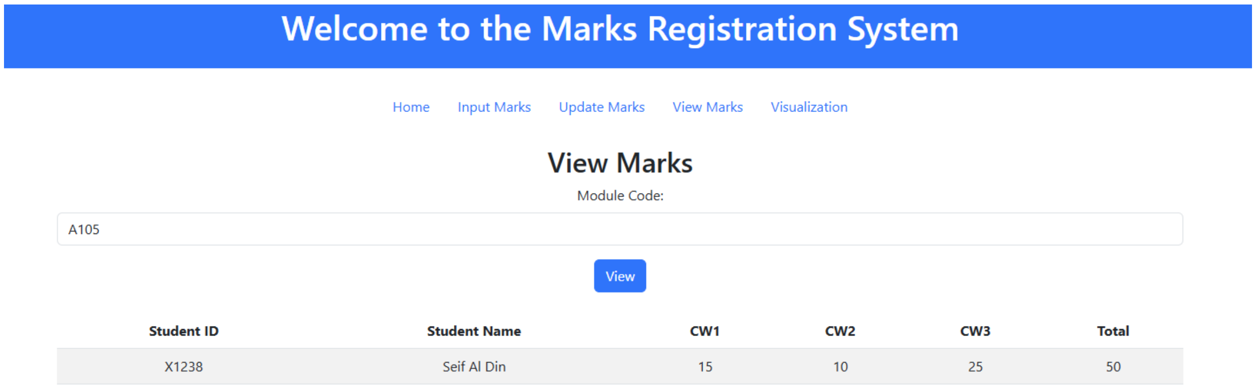
Coursework 2 Marks:

27

Coursework 3 Marks:

10

Update Marks



Welcome to the Marks Registration System

Home Input Marks Update Marks View Marks Visualization

View Marks

Module Code:

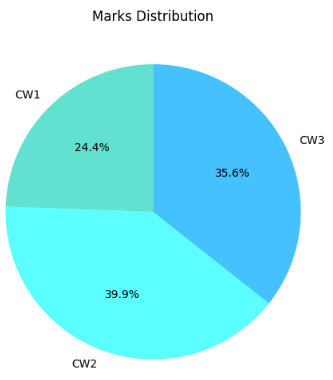
A105

View

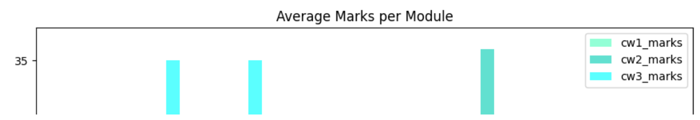
Student ID	Student Name	CW1	CW2	CW3	Total
X1238	Seif Al Din	15	10	25	50

Data Visualizations

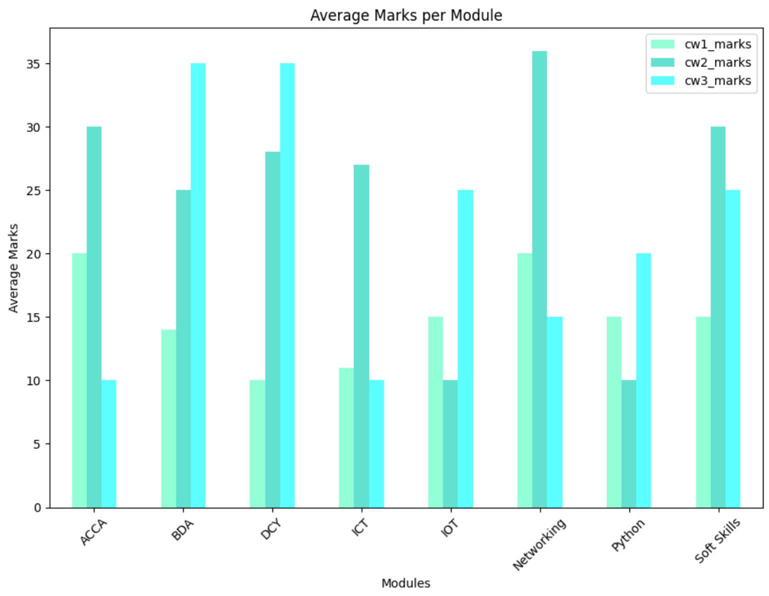
Pie Chart: Marks Distribution



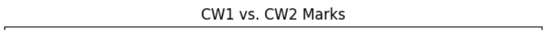
Bar Chart: Average Marks by Module



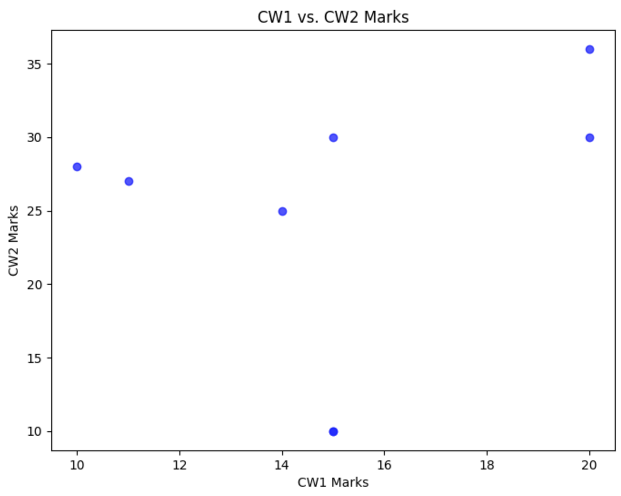
Bar Chart: Average Marks by Module



Scatter Plot: CW1 vs CW2 Marks



Scatter Plot: CW1 vs CW2 Marks


CW1 vs. CW2 Marks
CW2 Marks
CW1 Marks