mark_registration_system > mark_registration_system >

> settings.py

```
"""
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',
```

```python
    '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'],
        'APP_DIRS': True,
        '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'



# 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

USE_TZ = True
```

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

STATIC_URL = '/static/'
# For development (ensure it's enabled)
STATICFILES_DIRS = [BASE_DIR / "static"]
# Set this to where you want to collect static files
STATIC_ROOT = BASE_DIR / 'staticfiles'  # or use an absolute path, e.g.,
'C:/path/to/staticfiles'

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

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

> urls.py

```
"""
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

> admin_login.html

```
{% load static %}

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

    <!-- Link to favicon -->
    <link rel="icon" href="{% static 'favicon.ico' %}">

    <!-- Directly embedding CSS -->
    <style>
        /* Global Styles */
        body, html {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            height: 100vh;
            width: 100vw;
            overflow: hidden;
        }

        body::before {
            content: "";
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            background-image: url("{% static 'image/R.jpg' %}");
            background-size: cover;
            background-position: center;
            filter: blur(2px);
            z-index: -1;
        }

        .login-container {
            background: white;
            padding: 40px;
            border-radius: 8px;
            box-shadow: 0 4px 8px rgba(0,0,0,0.1);
            width: 300px;
            position: absolute;
            top: 50%;
            left: 50%;
            transform: translate(-50%, -50%);
            filter: none; /* Ensures the login form is not blurred */
        }

        .input-group {
            margin-bottom: 20px;
        }

        input[type="text"],
        input[type="password"] {
            width: 100%;
            padding: 10px;
            margin-top: 8px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }

        button {
            width: 100%;
            padding: 10px;
            background-color: #830051;
            border: none;
            border-radius: 4px;
            color: white;
            cursor: pointer;
            font-size: 16px;
        }

        button:hover {
            background-color: #a50061;
        }
```

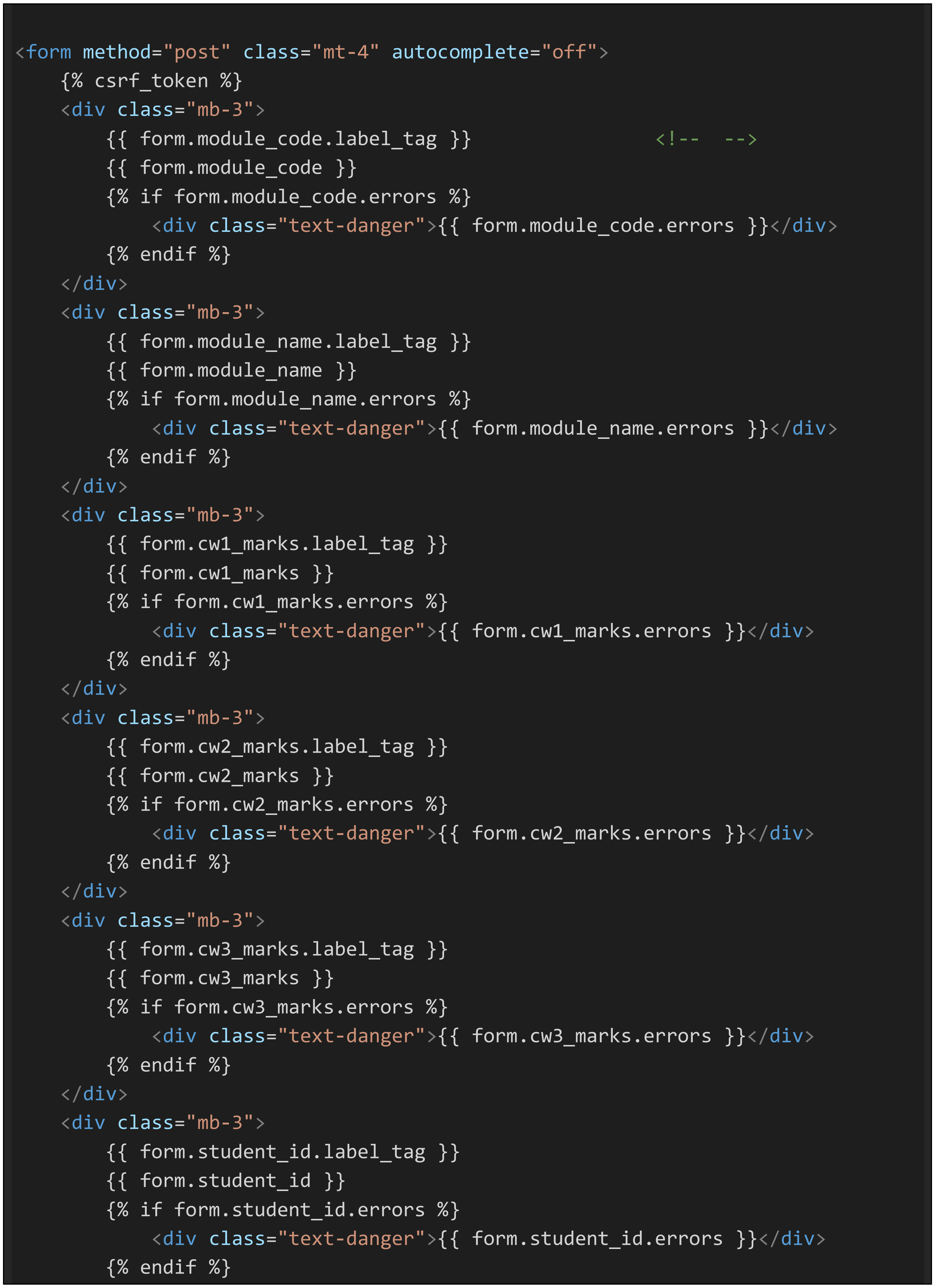
```
    </style>

</head>
<body>
    <div class="login-container">
        <form id="login-form" method="post" action="{% url 'admin_login' %}">
            {% csrf_token %}
            <h2>Admin Login</h2>
            <div class="input-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username" required>
            </div>
            <div class="input-group">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password" required>
            </div>
            <button type="submit">Login</button>
            {% if messages %}
                {% for message in messages %}
                    <p><strong>{{ message }}</strong></p>
                {% endfor %}
            {% endif %}
        </form>
    </div>

    <!-- Directly embedding JavaScript -->
    <script>
        document.getElementById('login-form').addEventListener('submit',
function(event) {
            event.preventDefault();
            const formData = new FormData(event.target);

            const csrftoken =
document.querySelector('[name=csrfmiddlewaretoken]').value;

            fetch('/admin_login/', {  // Ensure this matches the URL pattern
defined in urls.py
                method: 'POST',
                headers: {
                    'X-CSRFToken': csrftoken,  // CSRF token is sent with the
request
                },
                body: new URLSearchParams(formData)  // The form data is sent in the body of the request
            })
                .then(response => response.json())  // Expecting a JSON response
                .then(data => {
                    if (data.success) {
                        window.location.href = '/home';  // Redirect to home if login is successful
                    } else {
                        alert('Invalid credentials');
                    }
                })
                .catch(error => {
                    console.error('Error logging in:', error);
                });
        });

    </script>
</body>
</html>
```

> 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>
        /* Global Styles */
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f9; /* Light background color */
            color: #333; /* Text color */
        }

        /* Header Section */
        header {
            background-color: #700041; /* Deep maroon color */
            color: #fff; /* White text */
            text-align: center;
            padding: 1rem 0;
        }

        /* Navigation Bar */
        .nav-link {
            color: #830051; /* Dark pink for links */
            font-weight: bold;
        }

        .nav-link:hover {
            color: #700041 !important; /* Deep maroon on hover */
            text-decoration: underline; /* Add underline on hover */
        }

        /* Card Styles */
        .card {
            border: none; /* Remove card border */
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /* Subtle shadow for
cards */
        }

        /* Main Content Section */
        main {
            text-align: center;
            padding: 2rem;
        }

        /* Button and Placeholder Styles */
        .btn-primary {
            background-color: #830051; /* Set to deep pink color */
            border-color: black;
            box-shadow: 0px 4px 8px rgba(131, 0, 81, 0.5);
        }

        .btn-primary:hover {
            background-color: #a50061;
            box-shadow: 0px 6px 12px rgba(131, 0, 81, 0.7);
        }

        input::placeholder {
            color: #830051; /* Set placeholder color to match the theme */
            opacity: 1; /* Ensure placeholder is not faded */
        }

        /* Input Field Styles */
        input, select, textarea {
            border-color: #830051;
        }
    </style>
</head>
<body>
    <!-- Header Section -->
    <header>
        <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" href="{% url 'home' %}">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'input_marks' %}">Input
Marks</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'update_marks' %}">Update
Marks</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'view_marks' %}">View
Marks</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url
'visualization' %}">Visualization</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{% url 'logout' %}">Logout</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 %}

<style>
    /* Custom Deep Pink Button */
    .btn-deep-pink {
        color: white;
        background-color: #830051; /* Deep Pink Color */
        border-color: #000000; /* black border color */
        box-shadow: 0 4px 6px #a50061(0, 0, 0, 0.3);
    }

    /* Hover effect for Deep Pink Button */
    .btn-deep-pink:hover {
        color:white;
        background-color: #a50061; /* Slightly lighter deep pink */
        transform: scale(1.05); /* Slightly enlarges the button */
        border-color: #000000; /* Match the border with hover */
        box-shadow: 0 6px 8px #830051(0, 0, 0, 0.4); /* Shadow effect */
    }

    /* Active state for Deep Pink Button */
    .btn-deep-pink:active {
        background-color: #700041; /* Darker shade of deep pink */
        border-color: #700041;
    }
</style>
```



```
<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-deep-pink" id="myButton">
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 %}

<style>

    .btn-deep-pink {
        color:white;
        background-color: #830051;
        border-color: black;
        box-shadow: 1cm;
    }

    /* Hover state */
    .btn-deep-pink:hover {
        color:white;
        background-color: #a50061; /* Changes background color slightly */
        transform: scale(1.05); /* Slightly enlarges the button */
        box-shadow: 0 6px 8px rgba(0, 0, 0, 0.4); /* Enhances shadow on hover
*/
        border-color: black;
    }

       /* Active state for Deep Pink Button */
       .btn-deep-pink:active {
        background-color: #700041; /* Darker shade of deep pink */
        border-color: #700041;
    }

</style>


<!-- 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-deep-pink">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-deep-pink">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>

<style>

    .btn-deep-pink {
        color:white;
        background-color: #830051;
        border-color: black;
        box-shadow: 1cm;
    }

```

```
/* Hover state */
    .btn-deep-pink:hover {
        color:white;
        background-color: #a50061; /* Changes background color slightly */
        transform: scale(1.05); /* Slightly enlarges the button */
        box-shadow: 0 6px 8px rgba(0, 0, 0, 0.4); /* Enhances shadow on hover
*/
        border-color: black;
    }

    .btn-deep-pink:focus {
        outline: none; /* Ensures no outline when focused */
        box-shadow: none;
    }

    /* Active state (clicking the button) */
    .btn-deep-pink:active {
        background-color: #90005a; /* Darker shade when clicked */
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.4); /* Reduces shadow depth */
        outline: none; /* Removes the blue outline */
    }
</style>



<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-deep-pink ">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

```
<!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>
```

> admin.py

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

# Register your models here.
admin.site.register(AdminUser)

```

> forms.py

```
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
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager,
PermissionsMixin

class AdminUserManager(BaseUserManager):
    def create_user(self, username, password=None):
        if not username:
            raise ValueError('Users must have a username')
        user = self.model(username=username)
        user.set_password(password)  # This handles password hashing
        user.save(using=self._db)
        return user

    def create_superuser(self, username, password):
        user = self.create_user(
            username=username,
            password=password,
        )
        user.is_admin = True
        user.is_staff = True
        user.is_superuser = True
        user.save(using=self._db)
        return user

class AdminUser(AbstractBaseUser, PermissionsMixin):
    username = models.CharField(max_length=255, unique=True)
    is_active = models.BooleanField(default=True)
    is_admin = models.BooleanField(default=False)
    is_staff = models.BooleanField(default=True)  # Necessary for admin access

    objects = AdminUserManager()

    USERNAME_FIELD = 'username'
    REQUIRED_FIELDS = []
```

```python
    def __str__(self):
        return self.username

    # Specify unique related names
    groups = models.ManyToManyField(
        'auth.Group',
        verbose_name='groups',
        blank=True,
        related_name='custom_adminuser_groups',  # Unique related_name
        help_text='The groups this user belongs to. A group is a collection of
permissions.'
    )
    user_permissions = models.ManyToManyField(
        'auth.Permission',
        verbose_name='user permissions',
        blank=True,
        related_name='custom_adminuser_permissions',  # Unique related_name
        help_text='Specific permissions for this user.'
    )

    def has_perm(self, perm, obj=None):
        return True

    def has_module_perms(self, app_label):
        return True

# Existing Marks model
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.login_view, name='admin_login'),  # Make login the home page
    path('admin_login/', views.login_view, name='admin_login'),
    path('home/', 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'),
    path('logout/', views.logout_view, name='logout'),
]

if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
```

>views.py

```
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
```

```python
from io import BytesIO
import base64
import pandas as pd
import matplotlib
from django.contrib.auth import authenticate, login
from django.http import JsonResponse
from django.contrib.auth.decorators import login_required
from django.contrib.auth import logout

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




# 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
@login_required
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
@login_required
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'],
            )

            # 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
@login_required
def update_marks(request):
    records = None
    module_code = request.GET.get('module_code')  # Get module_code from the
query parameter
    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

        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
@login_required
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 = ['mediumvioletred', 'hotpink', 'deeppink', 'palevioletred', ]

# Visualization page view
@login_required
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))
```

```python
        plt.pie([total_cw1, total_cw2, total_cw3], labels=['CW1', 'CW2', 'CW3'], autopct='%1.1f%%', startangle=90, colors=custom_colors[0:3])
        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[0:3])
        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[0])
        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})

def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)

        if user is not None and user.is_active:
            login(request, user)
            return JsonResponse({'success': True})  # Redirecting via JSON
        else:
            messages.error(request, 'Invalid username or password.')
            return JsonResponse({'success': False})  # Login failed

    return render(request, 'marks/admin_login.html')  # GET request
```

```
def logout_view(request):
    logout(request)
    return redirect('admin_login')  # Redirect to the login page after logout
```

>static> image

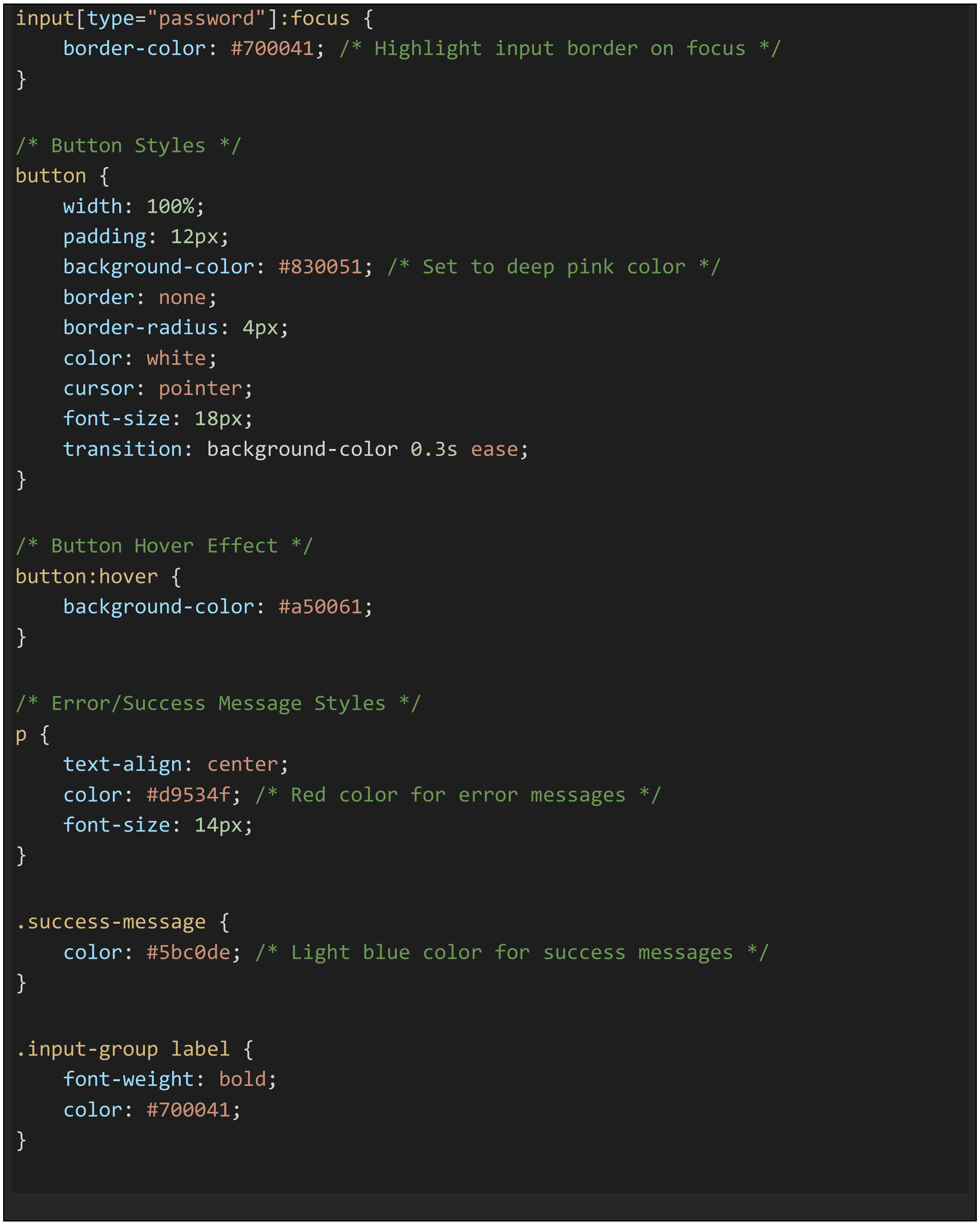
>admin_scripts.js

```
document.getElementById('login-form').addEventListener('submit',
function(event) {
    event.preventDefault();
    const formData = new FormData(event.target);

    const csrftoken =
document.querySelector('[name=csrfmiddlewaretoken]').value;

    fetch('/admin_login/', {  // Ensure this URL is correct
        method: 'POST',
        headers: {
            'X-CSRFToken': csrftoken,
        },
        body: new URLSearchParams(formData)
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            window.location.href = '/admin_dashboard'; // Adjust as needed
        } else {
            alert('Invalid credentials');
        }
    })
    .catch(error => {
        console.error('Error logging in:', error);
    });
});
```

>admin_styles.css

```
/* Global Styles */
body, html {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    height: 100vh;
    width: 100vw;
    overflow: hidden;
    background-color: #f4f4f9; /* Light background color */
}

/* Background Image */
body::before {
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-image:
url('https://th.bing.com/th/id/OIP.GKUDpmL2DuxN8jYXhNZnZQHaEK?rs=1&pid=ImgDetMa
in.jpg'); /* Make sure to provide a valid URL for background image */
    background-size: cover;
    background-position: center;
    filter: blur(8px);
    z-index: -1;
```

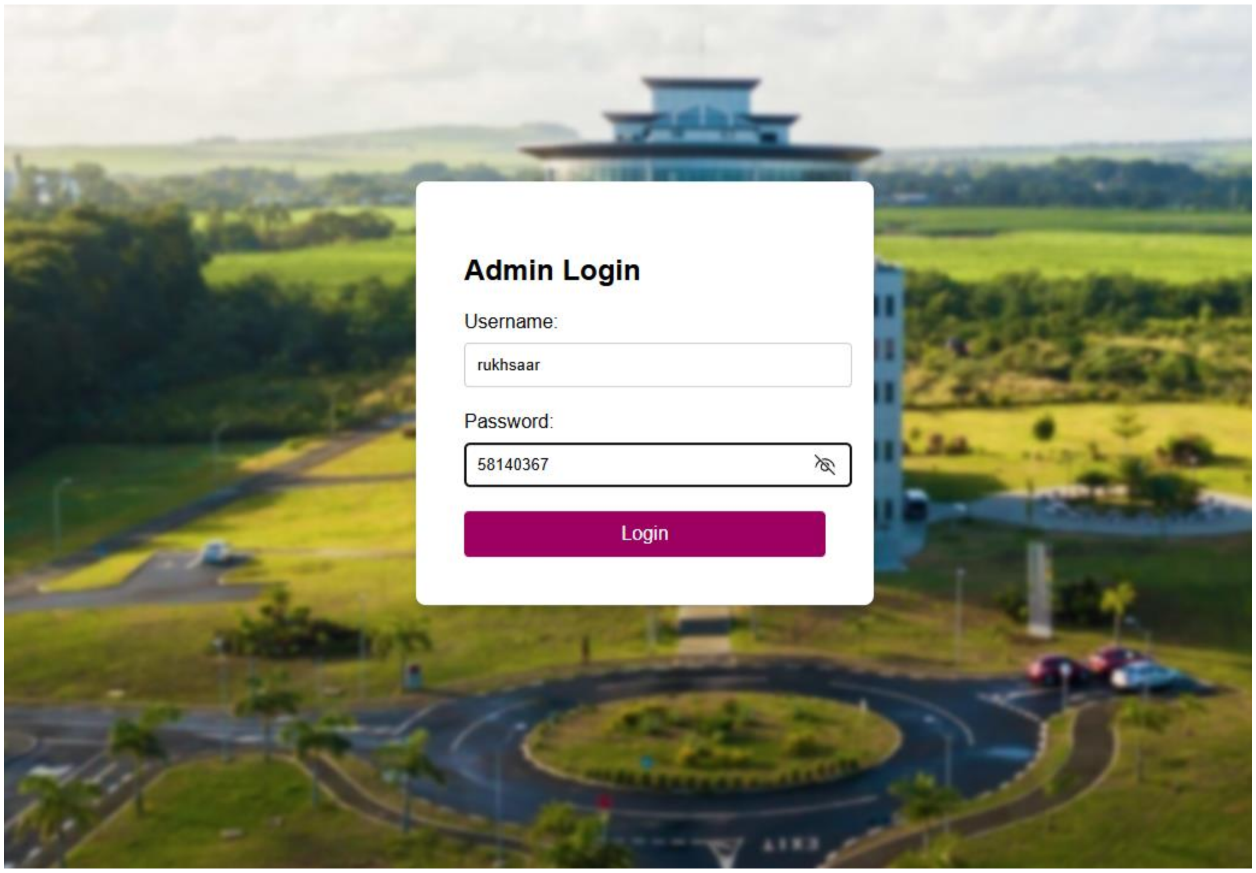
```css
}

/* Login Form Container */
.login-container {
    background: white;
    padding: 40px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 100%;
    max-width: 400px; /* Limit form width */
    margin: auto; /* Center form */
    position: relative;
    top: 50%;
    transform: translateY(-50%);
    filter: none; /* Ensures the login form is not blurred */
}

/* Heading */
h2 {
    text-align: center;
    margin-bottom: 20px;
    color: #700041; /* Deep maroon color */
}

/* Input Group Styles */
.input-group {
    margin-bottom: 20px;
}

input[type="text"],
input[type="password"] {
    width: 100%;
    padding: 12px;
    margin-top: 8px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 16px;
    background-color: #fafafa; /* Light background for inputs */
    transition: border-color 0.3s ease; /* Smooth transition on hover/focus */
}

/* Input Focus Effect */
input[type="text"]:focus,
```



```css
input[type="password"]:focus {
    border-color: #700041; /* Highlight input border on focus */
}

/* Button Styles */
button {
    width: 100%;
    padding: 12px;
    background-color: #830051; /* Set to deep pink color */
    border: none;
    border-radius: 4px;
    color: white;
    cursor: pointer;
    font-size: 18px;
    transition: background-color 0.3s ease;
}

/* Button Hover Effect */
button:hover {
    background-color: #a50061;
}

/* Error/Success Message Styles */
p {
    text-align: center;
    color: #d9534f; /* Red color for error messages */
    font-size: 14px;
}

.success-message {
    color: #5bc0de; /* Light blue color for success messages */
}

.input-group label {
    font-weight: bold;
    color: #700041;
}
```

Adding user and password to admin .

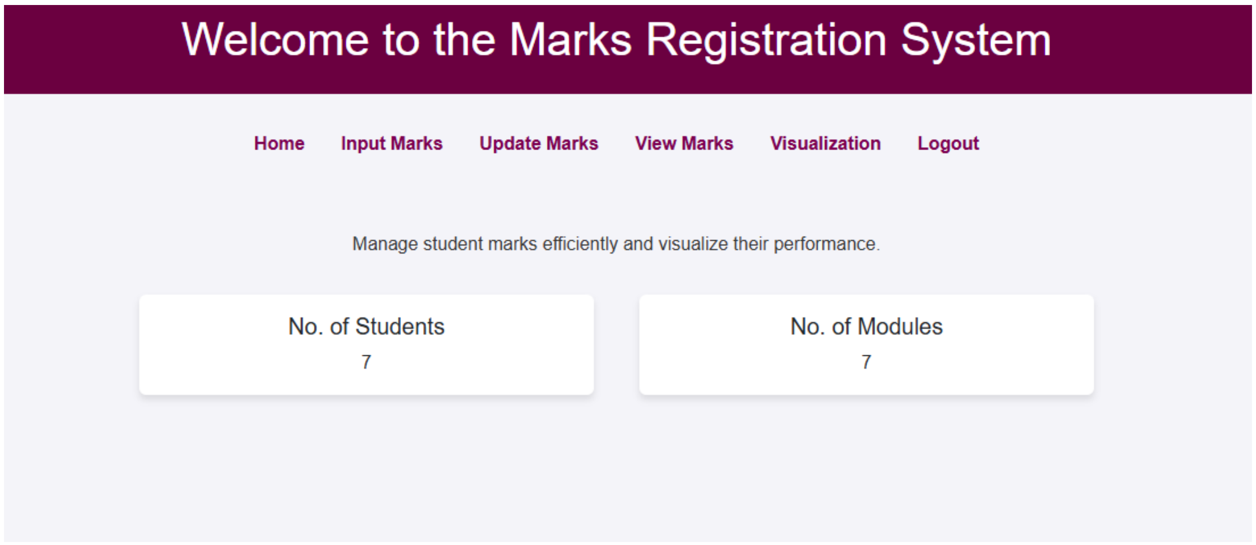
```
'''
admin
python manage.py createsuperuser

python manage.py shell

from django.contrib.auth.models import User
user = User.objects.get(username='yourusername')  #rukhsaar
user.email = 'yournewemail@example.com' #58140367
user.set_password('newpassword')  # If you want to update the password
user.save()

exit()

'''
```



Admin Login

Username: rukhsaar

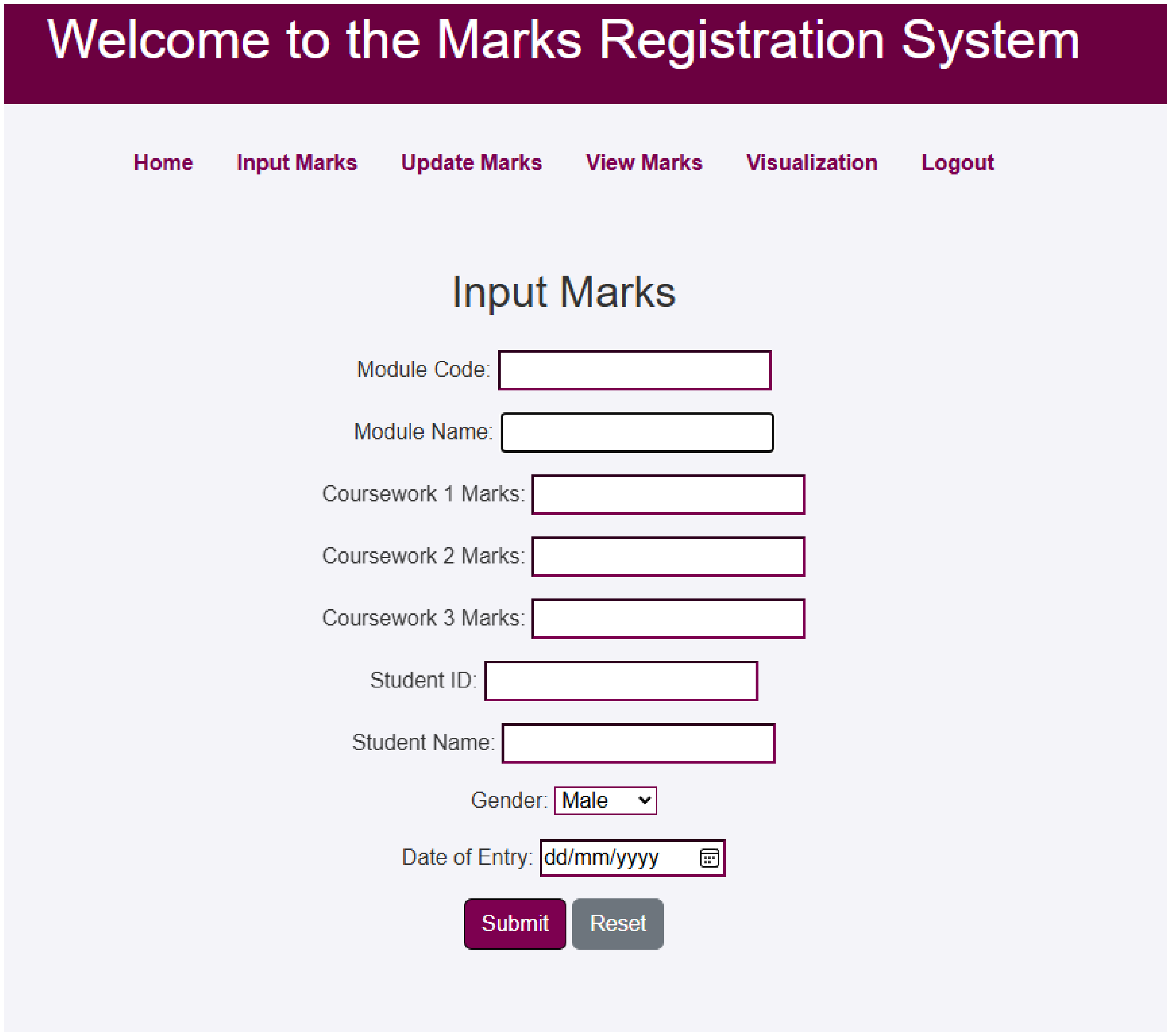
Password: 58140367



Welcome to the Marks Registration System

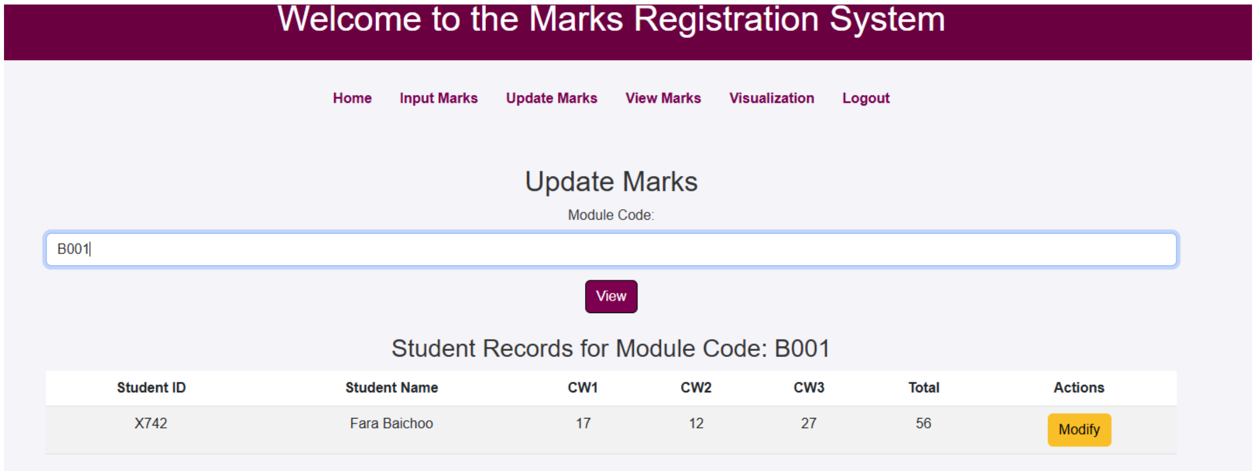
Manage student marks efficiently and visualize their performance.

No. of Students	No. of Modules
7	7



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

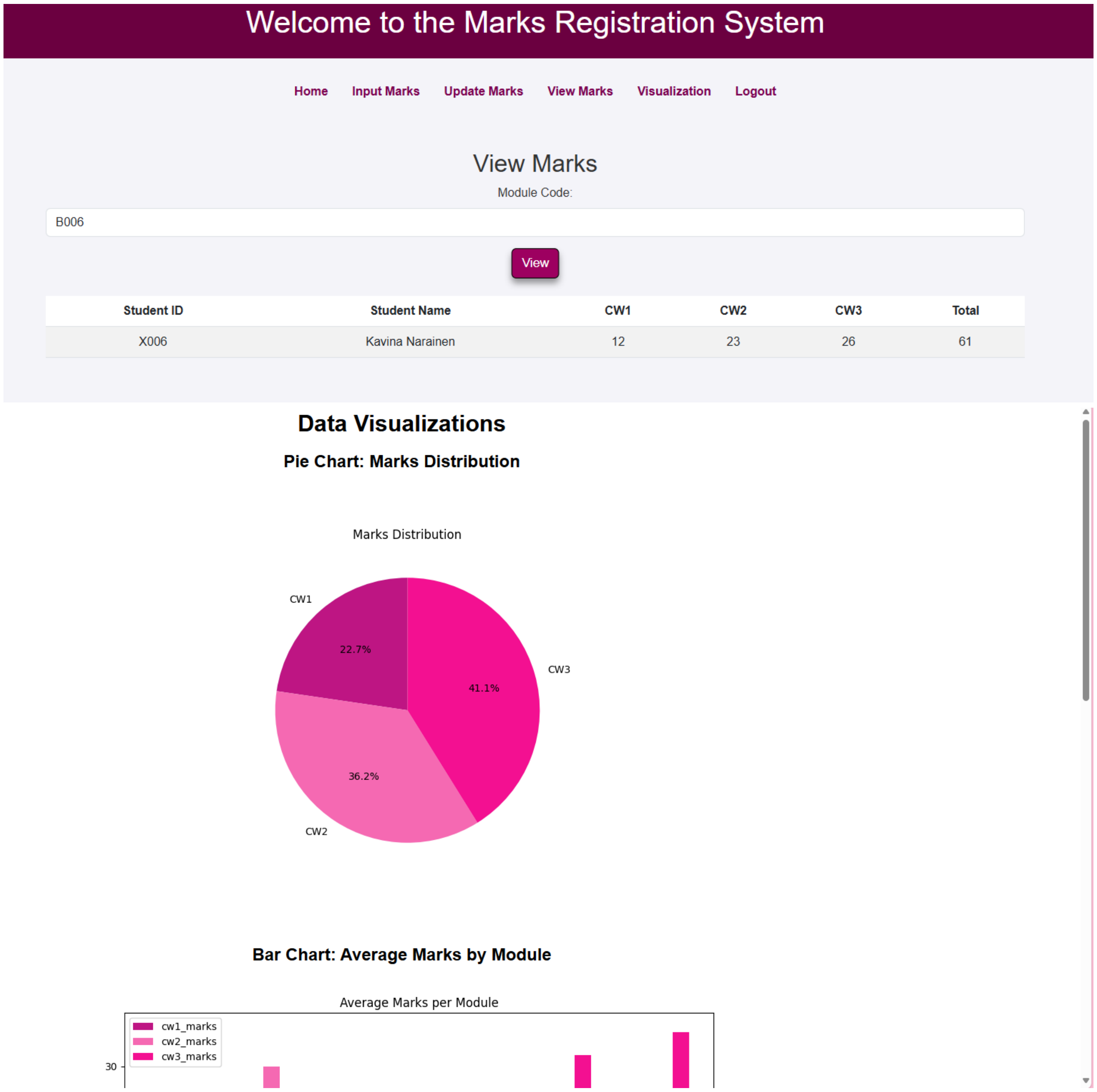


Update Marks

Module Code: B001

Student Records for Module Code: B001

Student ID	Student Name	CW1	CW2	CW3	Total	Actions
X742	Fara Baichoo	17	12	27	56	Modify



Welcome to the Marks Registration System

Home Input Marks Update Marks View Marks Visualization Logout

View Marks

Module Code:

B006

View

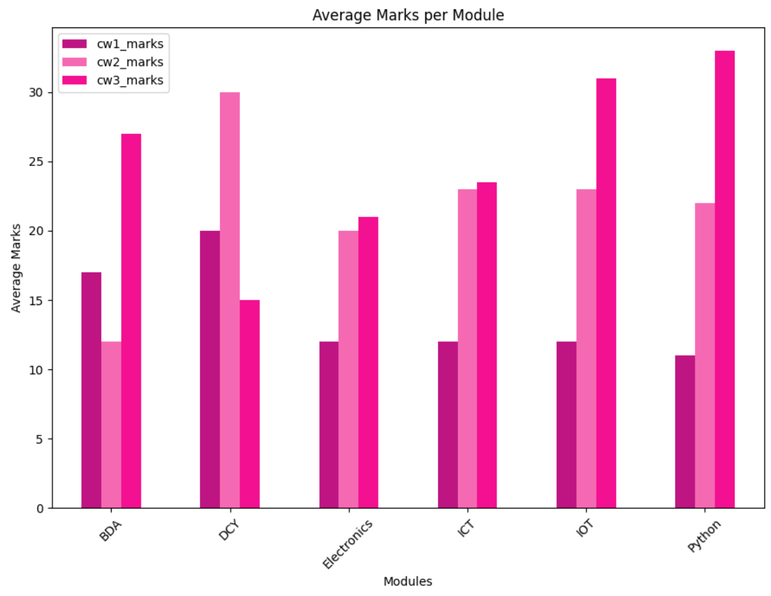
Student ID	Student Name	CW1	CW2	CW3	Total
X006	Kavina Narainen	12	23	26	61

Data Visualizations

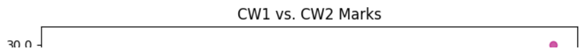
Pie Chart: Marks Distribution

Bar Chart: Average Marks by Module

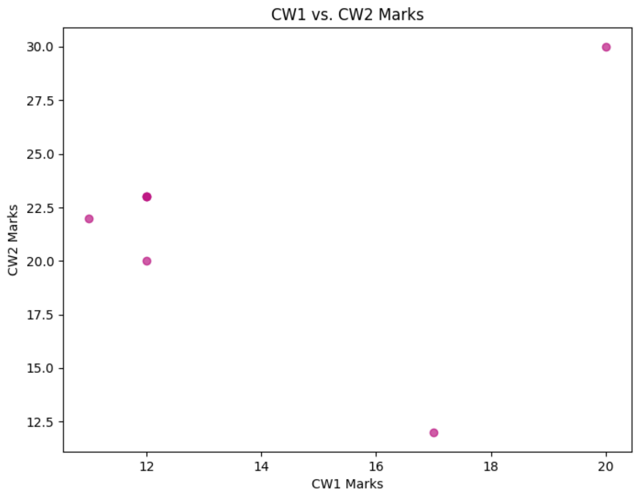
Bar Chart: Average Marks by Module

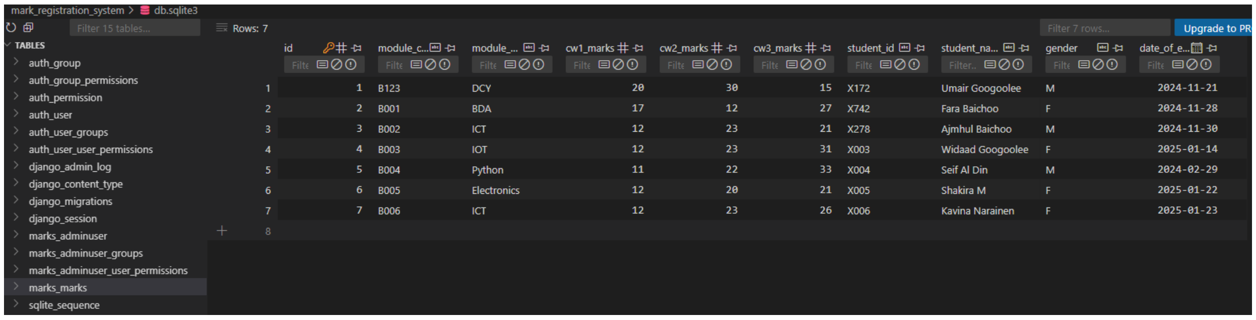


Scatter Plot: CW1 vs CW2 Marks



Scatter Plot: CW1 vs CW2 Marks





TABLES

- auth_group
- auth_group_permissions
- auth_permission
- auth_user
- auth_user_groups
- auth_user_user_permissions
- django_admin_log
- django_content_type
- django_migrations
- django_session
- marks_adminuser
- marks_adminuser_groups
- marks_adminuser_user_permissions
- marks_marks
- sqlite_sequence

	id	module_c...	module_...	cw1_marks	cw2_marks	cw3_marks	student_id	student_na...	gender	date_of_e...
1	1	B123	DCY	20	30	15	X172	Umair Googoolee	M	2024-11-21
2	2	B001	BDA	17	12	27	X742	Fara Baichoo	F	2024-11-28
3	3	B002	ICT	12	23	21	X278	Ajmhul Baichoo	M	2024-11-30
4	4	B003	IOT	12	23	31	X003	Widaad Googoolee	F	2025-01-14
5	5	B004	Python	11	22	33	X004	Seif Al Din	M	2024-02-29
6	6	B005	Electronics	12	20	21	X005	Shakira M	F	2025-01-22
7	7	B006	ICT	12	23	26	X006	Kavina Narainen	F	2025-01-23
8										