LAB 5

Name: Ayanika Paul

Batch:D1

Roll no. 22

Reg\_no : 220905128

SOLVED:

Q1) Build Your First Web Application in Django

myproject/settings.py

CODE:

from pathlib import Path

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

SECRET\_KEY = 'django-insecure-k%dwvv4t1\_2&2gm(1lhf-sm\*)d3q4ll7m%w2@\_7aa#j+&^47hf'

DEBUG = True

ALLOWED\_HOSTS = []

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

]

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 = 'q1.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        '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 = 'q1.wsgi.application'

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

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',

    },

]

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

STATIC\_URL = 'static/'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

myproject/urls.py

CODE:

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('webapp.urls')),  # Add this line to include webapp.urls for the root URL

]

webapp/urls.py

CODE:

from django.urls import path  # Import path from django.urls

from . import views  # Assuming you have views to handle the URLs

urlpatterns = [

    path('', views.index, name='index'),  # Replace `url()` with `path()`

]

webapp/views.py

CODE:

from django.shortcuts import render

from django.http import HttpResponse

def index(request):

    return HttpResponse("<h2>HEY! Welcome to Edureka!</h2>")

OUTPUT:

A screenshot of a video call

Description automatically generated

2. Displaying a Template

myproject/settings.py

CODE:

from pathlib import Path

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

SECRET\_KEY = 'django-insecure-k%dwvv4t1\_2&2gm(1lhf-sm\*)d3q4ll7m%w2@\_7aa#j+&^47hf'

DEBUG = True

ALLOWED\_HOSTS = []

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

]

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 = 'q1.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        '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 = 'q1.wsgi.application'

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

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',

    },

]

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

STATIC\_URL = 'static/'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

webapp/views.py

CODE:

from django.shortcuts import render

from django.http import HttpResponse

from datetime import date

import calendar

from calendar import HTMLCalendar

def index(request, year=None, month=None):

    if year is None or month is None:

        today = date.today()

        year = today.year

        month = today.month

    year = int(year)

    month = int(month)

    if year < 1900 or year > 2099:

        year = date.today().year

    month\_name = calendar.month\_name[month]

    title = f"MyClub Event Calendar - {month\_name} {year}"

    cal = HTMLCalendar().formatmonth(year, month)

    return HttpResponse(f"<h1>{title}</h1><p>{cal}</p>")

webapp/urls.py

CODE:

from django.urls import path, re\_path

from . import views

urlpatterns = [

    path('', views.index, name='index'),

    re\_path(r'^(?P<year>[0-9]{4})/(?P<month>(0?[1-9]|1[0-2]))/$', views.index, name='index'),

]

myproject/urls.py

CODE:

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('webapp.urls')),  # Add this line to include webapp.urls for the root URL

]

OUTPUT:



LAB Q

Q1) Create a web form which allows the teacher to enter the student details like name, date of birth, address, contact number, email id and marks of English, Physics and Chemistry. After filling all the information and on clicking submit button, details should be added to a textarea displayed on the same page. Display the total percentage of marks obtained in a label.

CODE:

q1/settings.py

from pathlib import Path

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

SECRET\_KEY = 'django-insecure-m9rls0zebd\*%5j+y)8xzf#st#%ks#vq!&sq5d74&y5@uim)zxg'

DEBUG = True

ALLOWED\_HOSTS = []

INSTALLED\_APPS = [

    'formapp',

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

]

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 = 'q1.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        '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 = 'q1.wsgi.application'

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

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',

    },

]

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

STATIC\_URL = 'static/'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

q1/urls.py

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('formapp.urls')),

]

formapp/urls.py

from django.urls import path

from . import views

urlpatterns = [

    path('', views.student\_form, name='student\_form'), ]

formapp/views.py

from django.shortcuts import render

from .forms import StudentForm

from .models import Student

def student\_form(request):

    students = Student.objects.all()

    form = StudentForm()

    if request.method == 'POST':

        form = StudentForm(request.POST)

        if form.is\_valid():

            form.save()

            form = StudentForm()  # Reset form after submission

    return render(request, 'student\_form.html', {'form': form, 'students': students})

formapp/views.py

CODE:

from django.shortcuts import render

from .forms import StudentForm

from .models import Student

def student\_form(request):

    students = Student.objects.all()

    form = StudentForm()

    if request.method == 'POST':

        form = StudentForm(request.POST)

        if form.is\_valid():

            form.save()

            form = StudentForm()  # Reset form after submission

    return render(request, 'student\_form.html', {'form': form, 'students': students})

formapp/templates/Student\_form.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Student Details Form</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

            padding: 20px;

            background-color: #f4f4f4;

        }

        .container {

            width: 60;

            background: white;

            padding: 33px;

            border-radius: 8px;

            box-shadow: 0px 0px 10px gray;

        }

        label {

            font-weight: bold;

        }

        input, textarea {

            width: 100%;

            padding: 8px;

            margin: 5px 0 15px;

            border: 1px solid #ccc;

            border-radius: 4px;

        }

        button {

            background: #28a745;

            color: white;

            padding: 10px;

            border: none;

            cursor: pointer;

            width: 100%;

        }

        button:hover {

            background: #218838;

        }

        textarea {

            width: 100%;

            height: 150px;

            margin-top: 15px;

        }

        .result {

            font-size: 18px;

            font-weight: bold;

            color: #333;

        }

    </style>

</head>

<body>

<div class="container">

    <h2>Student Details Form</h2>

    <form method="POST">

        {% csrf\_token %}

        {{ form.as\_p }}

        <button type="submit">Submit</button>

    </form>

    <h3>Student Records:</h3>

    <textarea readonly>

{% for student in students %}

Name: {{ student.name }}

DOB: {{ student.dob }}

Address: {{ student.address }}

Contact: {{ student.contact }}

Email: {{ student.email }}

Marks - English: {{ student.english }}, Physics: {{ student.physics }}, Chemistry: {{ student.chemistry }}

Total Percentage: {{ student.percentage }}%

------------------------------

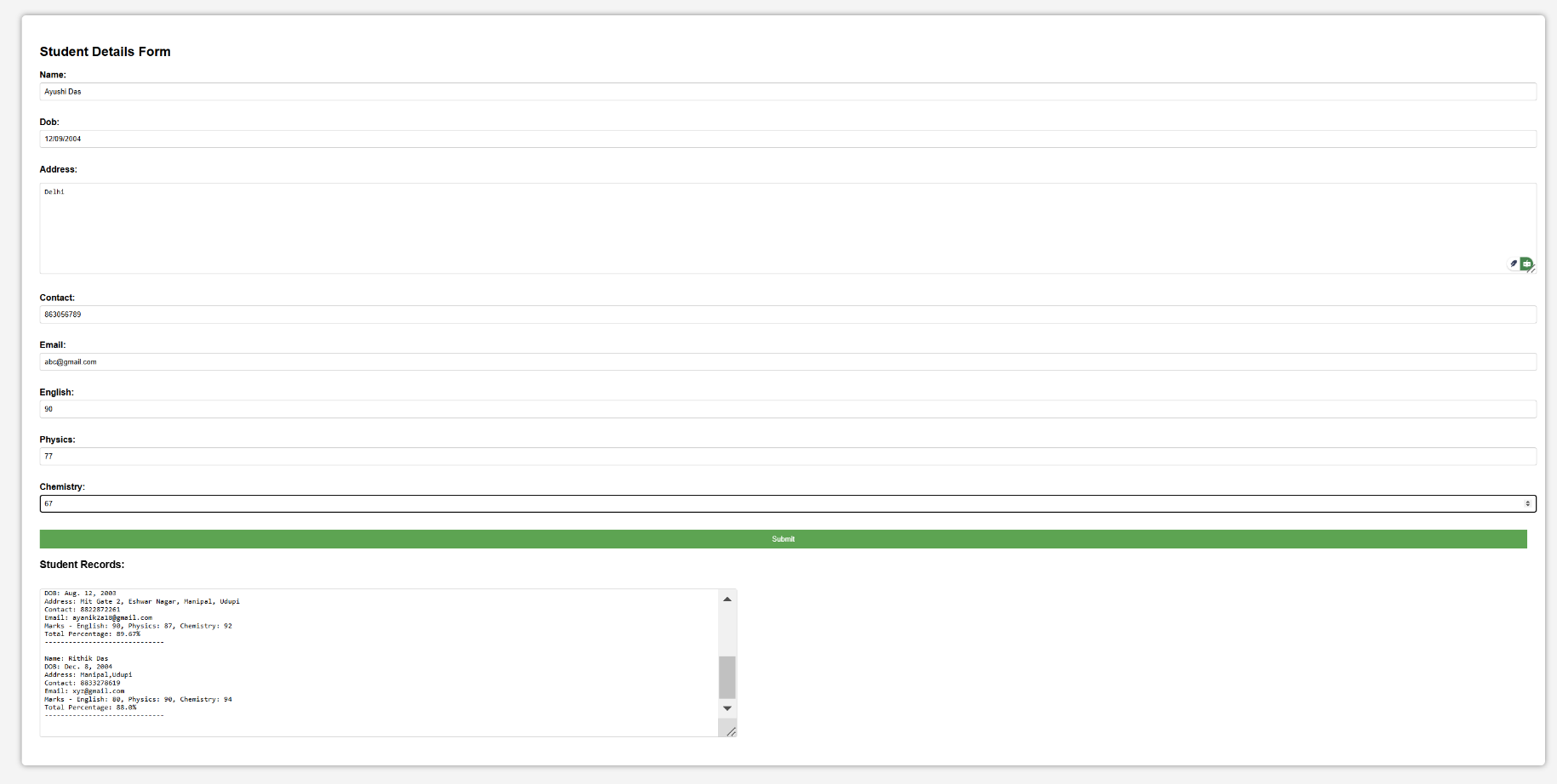
{% endfor %}

    </textarea>

</div>

</body>

</html>

OUTPUT:

A white rectangle with black lines

Description automatically generated

Q2) Create a web form with employee ids in dropdown list. Use a textbox for accepting date of joining. Add a button named “Am I Eligible for Promotion”. On clicking the button, if he has more than 5 years of experience, then display “YES” else “NO” in the label control.

q2/settings.py

CODE:

from pathlib import Path

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

SECRET\_KEY = 'django-insecure-(tcd%&4g!vc(^r9t=shhpwhfacf=287x2=gntd17f3lm)\_weov'

DEBUG = True

ALLOWED\_HOSTS = []

INSTALLED\_APPS = [

    'employeeapp',

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

]

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 = 'q2.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [

            BASE\_DIR / 'employeeapp/templates',  # Add this line to include the correct path

        ],

        '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 = 'q2.wsgi.application'

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

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',

    },

]

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

STATIC\_URL = 'static/'

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

q2/urls.py

CODE:

from django.contrib import admin

from django.urls import path, include

from employeeapp import views  # Import views from your app

urlpatterns = [

    path('admin/', admin.site.urls),

    path('employee/', include('employeeapp.urls')),

    path('', views.check\_promotion, name='home'),  # Serve the check\_promotion page at the root

]

employeeapp/urls.py

CODE:

from django.urls import path

from . import views

urlpatterns = [

    path('check\_promotion/', views.check\_promotion, name='check\_promotion'),

]

employeeapp/views.py

CODE:

from django.shortcuts import render

from datetime import datetime

def check\_promotion(request):

    error\_message = None  # Ensure the variable is always defined

    eligible = None  # Ensure eligible is always defined

    if request.method == 'POST':

employee\_id = request.POST.get('employee\_id')

        joining\_date\_str = request.POST.get('joining\_date')

        if not joining\_date\_str:  # If the joining date is missing

            error\_message = "Please provide the date of joining."

        else:

            try:

                joining\_date = datetime.strptime(joining\_date\_str, '%Y-%m-%d')

                experience = (datetime.now() - joining\_date).days // 365

                eligible = experience >= 5  # True if 5 or more years of experience

            except ValueError:

                error\_message = "Invalid date format. Please enter a valid date."

    return render(request, 'check\_promotion.html', {'eligible': eligible, 'error\_message': error\_message})

employeeapp/templates/check\_promotion.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Check Promotion Eligibility</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

            margin: 0;

        }

        .container {

            background: white;

            padding: 20px;

            border-radius: 8px;

            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

            width: 350px;

            text-align: center;

        }

        h2 {

            color: #333;

            margin-bottom: 20px;

        }

        form {

            display: flex;

            flex-direction: column;

        }

        label {

            text-align: left;

            font-weight: bold;

            margin-top: 10px;

        }

        select, input {

            padding: 8px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 5px;

            width: 97%;

        }

        button {

            background-color: #28a745;

            color: white;

            padding: 10px;

            border: none;

            border-radius: 5px;

            font-size: 16px;

            cursor: pointer;

            margin-top: 15px;

        }

        button:hover {

            background-color: #218838;

        }

        .result {

            font-size: 18px;

            font-weight: bold;

            margin-top: 15px;

            color: #333;

        }

        .error {

            color: red;

            font-size: 14px;

            margin-top: 10px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h2>Check Promotion Eligibility</h2>

        <form method="POST">

            {% csrf\_token %}

            <label for="employee\_id">Employee ID:</label>

            <select id="employee\_id" name="employee\_id" required>

                <option value="">Select Employee</option>

                <option value="1">Employee 1</option>

                <option value="2">Employee 2</option>

                <option value="3">Employee 3</option>

            </select>

            <label for="joining\_date">Date of Joining:</label>

            <input type="date" id="joining\_date" name="joining\_date" required>

         <button type="submit">Am I Eligible for Promotion?</button>

        </form>

        {% if error\_message %}

            <p class="error">{{ error\_message }}</p>

        {% endif %}

        {% if eligible is not none %}

            <p class="result">Eligibility for promotion:

                <span style="color: {% if eligible %}green{% else %}red{% endif %};">

                    {% if eligible %}YES{% else %}NO{% endif %}

                </span>

            </p>

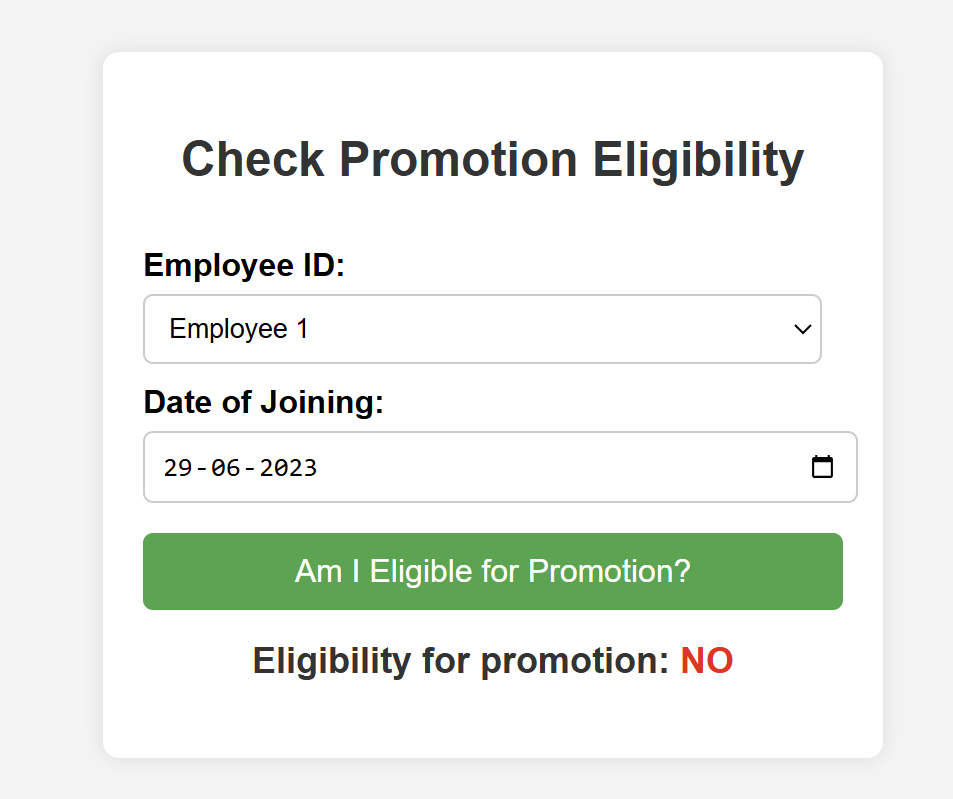
        {% endif %}

    </div>

</body>

</html>

OUTPUT:



A screenshot of a computer screen

Description automatically generated