Full Stack Development

Assignment 2

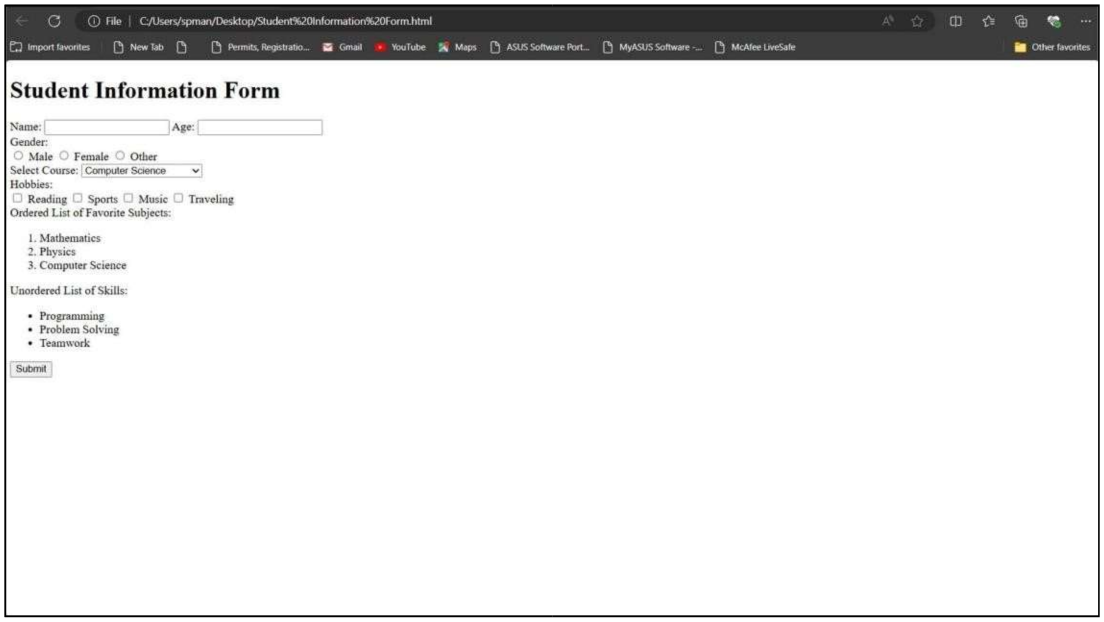
Submitted By:

PREETI CHAND H H (1DT21CS115)

Creation of Student Form

Form using HTML 5 and Django Framework With Comparision ,advantages and disadvantages

Using HTML5 Forms :



Student Information Form

Name: Age:
Gender:
○ Male ○ Female ○ Other
Select Course: Computer Science
Hobbies:
☐ Reading ☐ Sports ☐ Music ☐ Traveling
Ordered List of Favorite Subjects:

1. Mathematics
2. Physics
3. Computer Science

Unordered List of Skills:

- Programming
- Problem Solving
- Teamwork

Submit

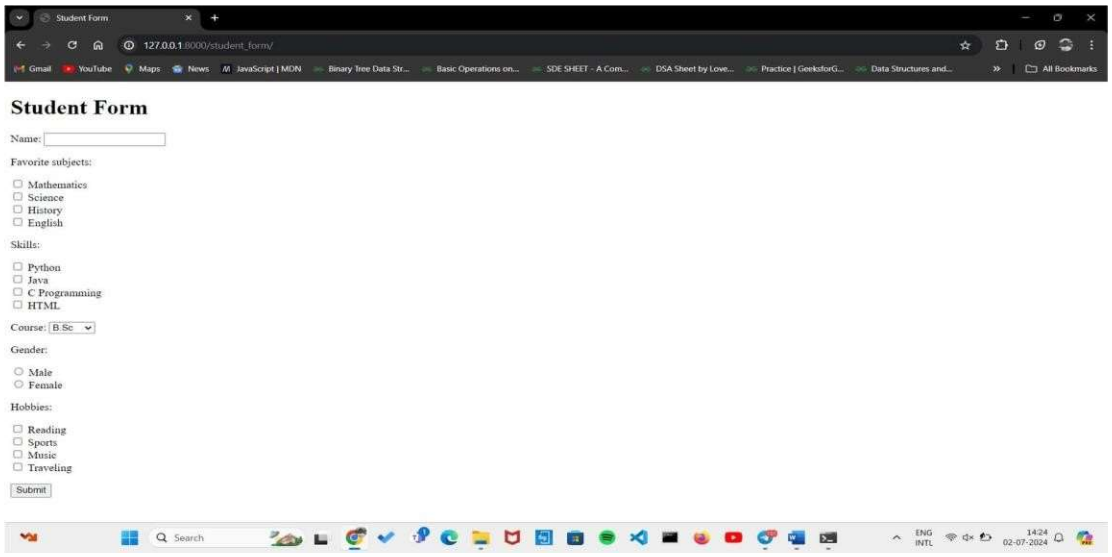
Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Student Form</title>
</head>
<body>
  <h1>Student Registration Form</h1>
  <form action="/submit_form" method="post">
    <label for="name">Name:</label><br>
    <input type="text" id="name" name="name"><br><br>

    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email"><br><br>
```

```
<label for="gender">Gender:</label><br>
<input type="radio" id="male" name="gender" value="male">
<label for="male">Male</label><br>
<input type="radio" id="female" name="gender" value="female">
<label for="female">Female</label><br><br>

<label for="course">Course:</label><br>
<select id="course" name="course">
   <option value="math">Math</option>
   <option value="science">Science</option>
   <option value="history">History</option>
</select><br><br>

<label for="hobbies">Hobbies:</label><br>
<input type="checkbox" id="reading" name="hobbies" value="reading">
<label for="reading">Reading</label><br>
<input type="checkbox" id="sports" name="hobbies" value="sports">
<label for="sports">Sports</label><br>
<input type="checkbox" id="music" name="hobbies" value="music">
<label for="music">Music</label><br><br>

<label for="skills">Skills:</label><br>
<ul>
   <li>HTML</li>
   <li>CSS</li>
   <li>JavaScript</li>
</ul>
<br>

<label for="languages">Languages:</label><br>
<ol>
   <li>English</li>
   <li>Spanish</li>
```

```
            <li>French</li>
        </ol>
        <br>


        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Using Django Framework:



127.0.0.1:8000/student_form/

Student Form

Name:

Favorite subjects:

- Mathematics
- Science
- History
- English

Skills:

- Python
- Java
- C Programming
- HTML

Course: B.Sc

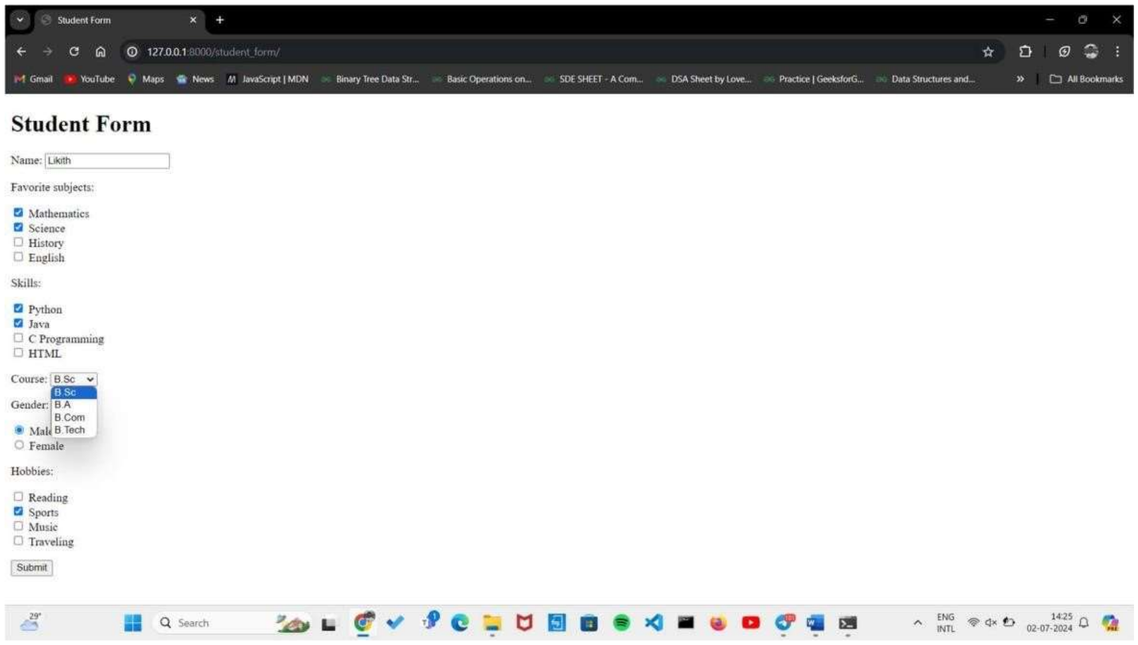
Gender:

- Male
- Female

Hobbies:

- Reading
- Sports
- Music
- Traveling

Submit



127.0.0.1:8000/student_form/

Student Form

Name: Likith

Favorite subjects:

- [✓ Mathematics]
- [✓ Science]
- History
- English

Skills:

- [✓ Python]
- [✓ Java]
- C Programming
- HTML

Course: B.Sc
[✓ B.Sc] B.A B.Com B.Tech

Gender:

- [✓ Male]
- Female

Hobbies:

- Reading
- [✓ Sports]
- Music
- Traveling

Submit

Code:

1. **Create a Django project named ACT**

```
Django-admin startproject studentForm

Cd studentForm

Python manage.py startapp mem
```

2. **Inside mem\form.py** # forms_app/forms.py from django
import forms from .models import Student

```
class StudentForm(forms.ModelForm):
    HOBBIES_CHOICES = [
        ('reading', 'Reading'),
        ('sports', 'Sports'),
        ('music', 'Music'),
    ]
    hobbies = forms.MultipleChoiceField(choices=HOBBIES_CHOICES,
widget=forms.CheckboxSelectMultiple)

    class Meta:
        model = Student         fields = ['name', 'email',
'gender', 'course', 'hobbies']
```

3. **Inside formapp\views.py**

```
# forms_app/views.py from
django.shortcuts import render
from .forms import StudentForm

def student_registration(request):
if request.method == 'POST':
      form = StudentForm(request.POST)
if form.is_valid():
```

```
            form.save()             return
render(request, 'success.html')     else:
        form = StudentForm()
    return render(request, 'registration.html', {'form': form})
```

4. **Inside fromapp\urls.py**

```
 forms_app/urls.py from django.urls
import path from .views import
student_registration

urlpatterns = [     path('register/', student_registration,
name='student_registration'),
]
```

5. **Inside Project urls.py**

```
# myproject/urls.py from
django.contrib import admin from
django.urls import include, path

urlpatterns = [     path('admin/',
admin.site.urls),     path('forms_app/',
include('forms_app.urls')),
]
```

6. **Inside application folder create html file i.e formapp\templates\formapp\registration.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">     <meta
name="viewport"
content="width=device-width,
initialscale=1.0">
    <title>Student Registration Form</title>
</head>
```

```
<body>
    <h1>Student Registration Form</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

❖ Comparative Study

Aspect	Traditional HTML Form	Django Form
Implementation	Direct HTML coding for the form.	Uses Django's form class for form definition and rendering.
Validation	Manual JavaScript/PHP/Backend validation required.	Built-in form validation provided by Django forms.
Security	CSRF protection must be manually added.	Automatic CSRF protection provided by Django.
Reusability	Limited, requires duplication for multiple forms.	Highly reusable, forms can be defined once and used across views.
Maintainability	Changes require updating HTML directly.	Changes can be made in forms.py, reducing maintenance effort.
Customization	Full control over HTML and CSS customization.	Customization possible but requires understanding Django templates.
Error Handling	Requires manual error handling in HTML or backend.	Built-in error handling and display with Django forms.
Integration	Needs manual integration with backend scripts and databases.	Seamless integration with Django's ORM and view functions.
Time Efficiency	Quicker for simple forms, but complex forms become cumbersome.	More efficient for complex forms due to Django's abstractions.

❖ Merits and Demerits **Traditional**

HTML Form:

- **Merits:** ○ Simplicity: Easy to start with and understand for beginners.
  - Flexibility: Full control over the HTML and styling.
- **Demerits:** ○ Validation: Requires additional scripting for validation. ○ Security: Must handle CSRF protection manually. ○ Maintainability: Changes need to be manually updated in the HTML.

Django Form:

- **Merits:** ○ Built-in Validation: Simplifies form validation. ○ Security: Automatic CSRF protection. ○ Reusability: Forms can be reused across different views.
  - Integration: Seamlessly integrates with Django's ORM and view functions.
- **Demerits:**
  - Learning Curve: Requires knowledge of Django framework. ○ Customization: Customizing HTML output can be more complex.

❖ Limitations

- **Traditional HTML Form:** ○ Limited to static form generation without backend support. ○ Requires more code for validation and security features.
- **Django Form:** ○ Requires a Django project setup, which can be overhead for simple forms. ○ Customization might require diving deeper into Django's templating system.