

**Assignment 1:**

1. Create registration page in html with username, email and phone number and by using POST method display it in next html page.

Server used: Apache Tomcat

**Code:**

Registration.html

```
<html>
  <body>
    <form method="POST" action="serv1">
      <fieldset>
        <legend>User Details:</legend>
        <h3>Username: </h3>
        <input type="text" name="uname">
        <h3>Email: </h3>
        <input type="email" name="email">
        <h3>Phone Number: </h3>
        <input type="number" name="pno">
        <br><br><br>
        <input type="submit">
      </fieldset>
    </form>
  </body>
</html>
```

Display.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.http.HttpSession;
import java.util.*;
import java.io.*;

public class display extends HttpServlet{

    public void doPost(HttpServletRequest req,
HttpServletResponse res){
```

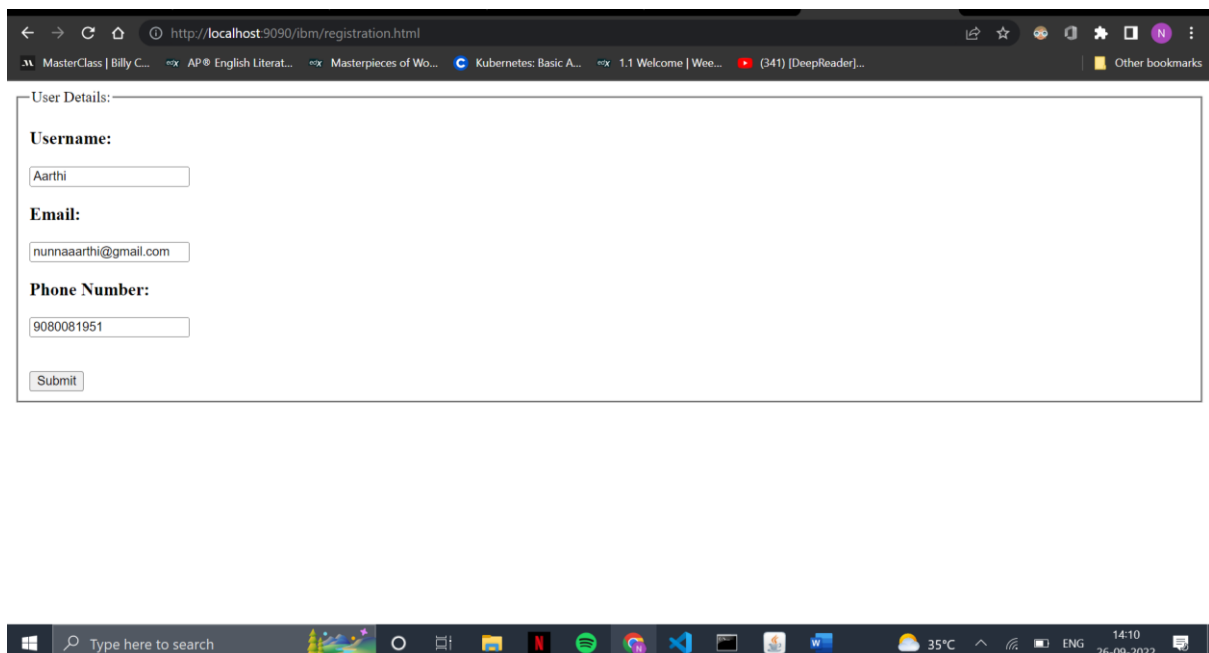
```
String uname = req.getParameter("uname");
String pno = req.getParameter("pno");
String email = req.getParameter("email");

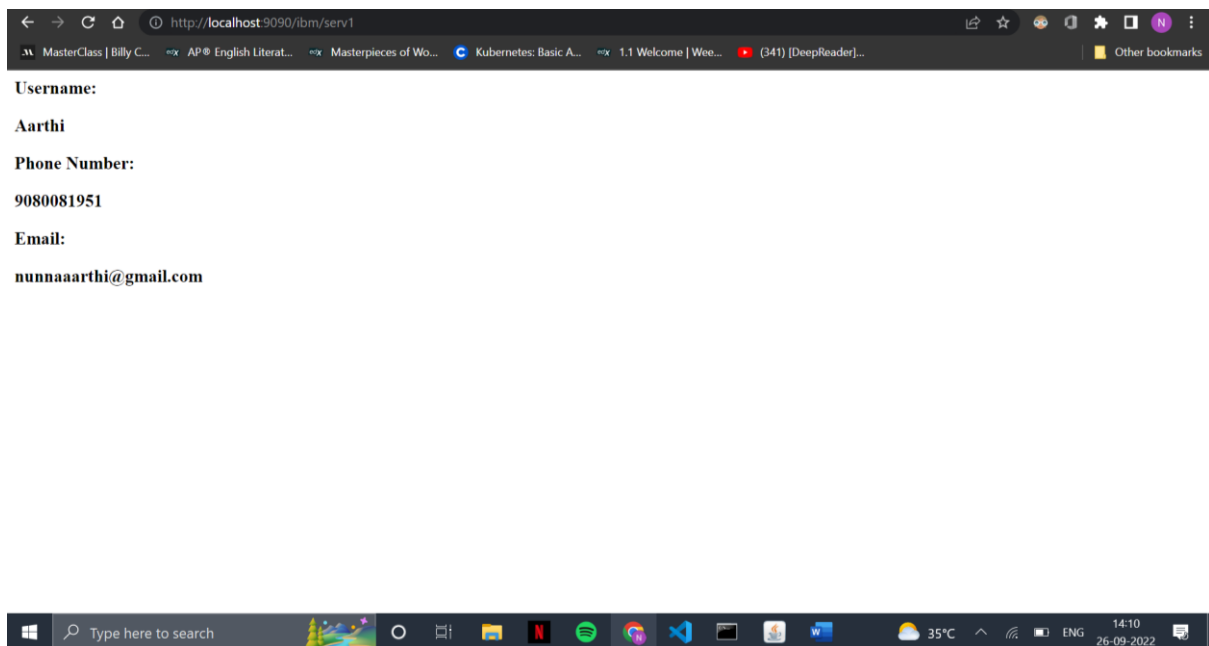
try{

    res.setContentType("text/html");
    PrintWriter pw = res.getWriter();

    pw.print("<h3>Username: <h3>" + uname);
    pw.print("<h3>Phone Number: <h3>" + pno);
    pw.print("<h3>Email: <h3>" + email);
}
catch(Exception e){
    System.out.println(e);
}
}
```

### Output Screenshots:





2. Develop a flask program which should contain atleast 5 packages used from pypi.org.

Packages used: NumPy, Pandas, Sklearn, Pendulum, Matplotlib

**Code:**

```
import io
from flask import Response, Flask, jsonify
from matplotlib.backends.backend_agg import FigureCanvasAgg
as FigureCanvas
from matplotlib.figure import Figure
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pendulum

app = Flask(__name__)

@app.route('/numpy')
def numpy_display():
    arr = np.array([1, 2, 3, 4, 5])
```

```
        return "I have " + str(arr[2]) + " apples with me"

@app.route('/matplotlib')
def matplotlib_display():
    fig = Figure()
    axis = fig.add_subplot(1, 1, 1)
    xs = [10,20,30]
    ys = [20,30,40]
    axis.plot(xs, ys)
    output = io.BytesIO()
    FigureCanvas(fig).print_png(output)
    return Response(output.getvalue(), mimetype='image/png')

@app.route('/panda')
def panda_display():
    data = {
        "Subject": ["Maths", "Physics", "Chemistry"],
        "Marks" : [100,87,98]
    }
    df = pd.DataFrame(data)
    return jsonify(df.to_dict(orient="records"))

@app.route('/pendulum')
def pendulum_display():
    utc_time = pendulum.now('UTC')
    kolkata_time = utc_time.in_timezone('Asia/Kolkata')
    str1 = 'Current Date Time in Kolkata =' +
str(kolkata_time)
    sydney_time = utc_time.in_timezone('Australia/Sydney')
    str2 = 'Current Date Time in Sydney =' + str(sydney_time)
    return str1 + "\n" + str2

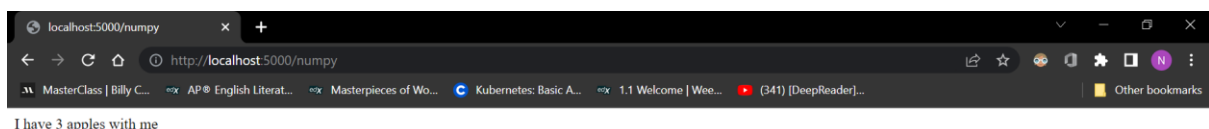
@app.route('/sklearn')
def sklearn_display():
    iris = load_iris()
    X = iris.data
    y = iris.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.4, random_state=1)

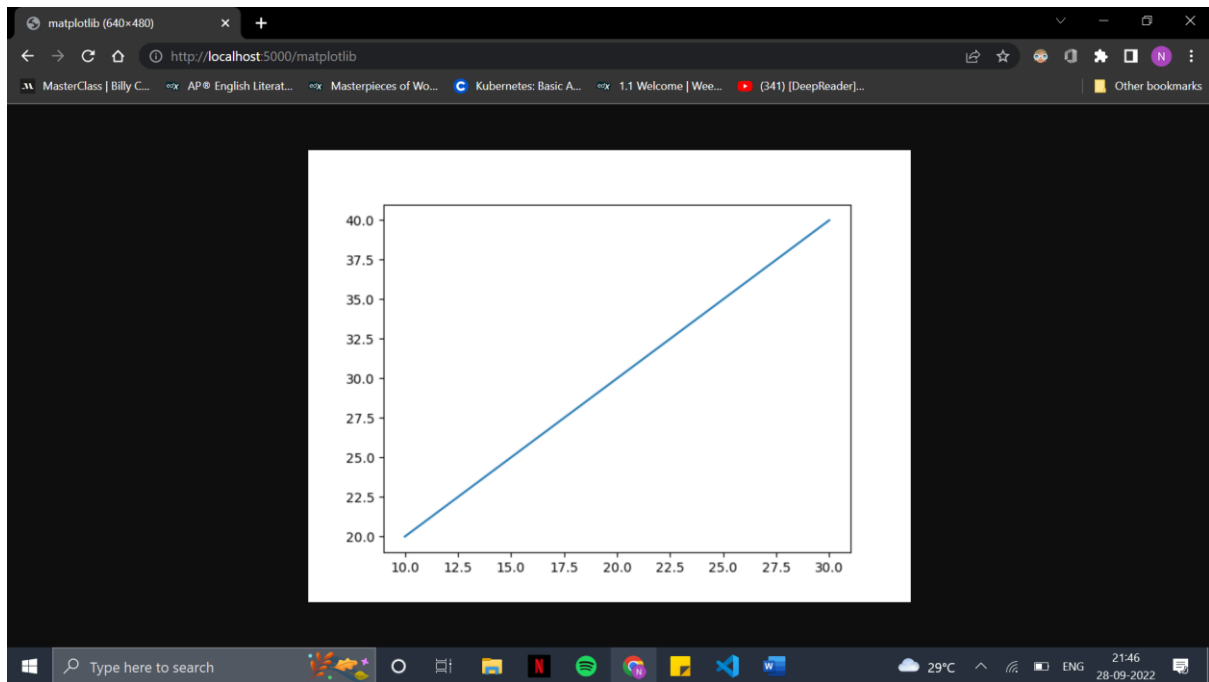
classifier_knn = KNeighborsClassifier(n_neighbors = 3)
classifier_knn.fit(X_train, y_train)
y_pred = classifier_knn.predict(X_test)
str1 = "<h3> Iris Data Set ML Predictions using KNN
</h3>"
str1 += "<h2> Training Data Input </h2>"
str1 += str(X_train)
str1 += "<h2> Training Data Output </h2>"
str1 += str(y_train)
str1 += "<h2> Y Predictions: </h2>"
str1 += "<p>" + str(y_pred) + "</p>"
str1 += "<h2> Y Test: </h2>"
str1 += "<p>" + str(y_test) + "</p>"
str1 += "<h2>" + "Accuracy" + "</h2>" +
str(metrics.accuracy_score(y_test,y_pred))
return str1
```

Output Screenshots:

Numpy



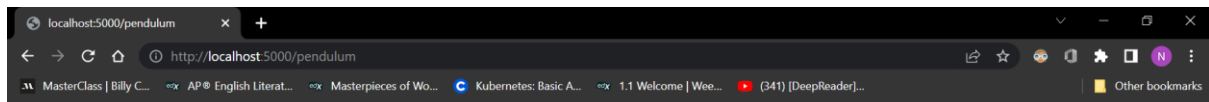
## Matplotlib



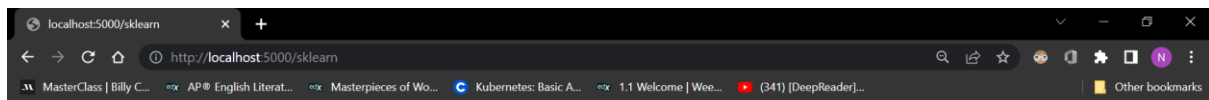
## Pandas

```
localhost:5000/panda x +
http://localhost:5000/panda
MasterClass | Billy C... AP® English Literat... Masterpieces of Wo... Kubernetes: Basic A... 1.1 Welcome | Wee... (341) [DeepReader]... Other bookmarks
[{"Marks":100,"Subject":"Maths"}, {"Marks":87,"Subject":"Physics"}, {"Marks":98,"Subject":"Chemistry"}]
```

## Pendulum



## Sklearn



### Iris Data Set ML Predictions using KNN

#### Training Data Input

```
[[4.8 3.4 1.6 0.2] [5.7 2.5 5.2.] [6.3 2.7 4.9 1.8] [4.8 3.1 4.0 1.] [4.7 3.2 1.3 0.2] [6.5 3.5 8.2 2.] [4.6 3.4 1.4 0.3] [6.1 3.4 9.1 8.] [6.5 3.2 5.1 2.] [6.7 3.1 4.4 1.4] [5.7 2.8 4.5 1.3] [6.7 3.3 5.7 2.5] [6.3 4.8 1.8] [5.1 3.8 1.6 0.2] [6.2 2.4 1.] [6.4 2.9 4.3 1.3] [6.5 3.5 5.1 8.] [5.2 3.3 1.] [6.3 3.6 2.5] [5.5 2.5 4.1 3] [5.4 3.7 1.5 0.2] [4.9 3.1 1.5 0.2] [5.2 4.1 1.5 0.1] [6.7 3.3 5.7 2.1] [4.4 3.1 3.0 2] [6.2 7.5 1.6] [6.4 2.7 5.3 1.9] [5.9 3.5 1.8] [5.2 3.5 1.5 0.2] [5.1 3.3 1.7 0.5] [5.8 2.7 4.1 1.] [4.9 3.1 1.5 0.1] [7.4 2.8 6.1 1.9] [6.2 2.9 4.3 1.3] [7.6 3.6 6.2 1.] [6.7 3.5 2.2 3] [6.3 2.3 4.4 1.3] [6.2 3.4 5.4 2.3] [7.2 3.6 6.1 2.5] [5.6 2.9 3.6 1.3] [5.7 4.4 1.5 0.4] [5.8 2.7 3.9 1.2] [4.5 2.3 1.3 0.3] [5.5 2.4 3.8 1.1] [6.9 3.1 4.9 1.5] [5.3 4.1 6.0 4] [6.8 2.8 4.8 1.4] [5.3 5.1 6.0 6] [4.8 3.4 1.9 0.2] [6.3 3.4 5.6 2.4] [5.6 2.8 4.9 2.] [6.8 3.2 5.9 2.3] [5.3 3.1 4.0 2] [5.1 3.7 1.5 0.4] [5.9 3.2 4.8 1.8] [4.6 3.1 1.5 0.2] [5.8 2.7 5.1 1.9] [4.8 3.1 1.6 0.2] [6.5 3.5 2.2.] [4.9 2.5 4.5 1.7] [4.6 3.2 1.4 0.2] [6.4 3.2 5.3 2.3] [4.3 3.1 1.0 1] [5.6 3.4 1.1 3] [4.4 2.9 1.4 0.2] [5.5 2.4 3.7 1.] [5.2 3.5 1.] [5.1 3.5 1.4 0.2] [4.9 3.1 4.0 2] [4.9 2.4 3.3 1.] [4.6 3.6 1.0 2] [5.9 3.4 2.1 5] [6.1 2.9 4.7 1.4] [5.3 4.1 5.0 2] [6.7 3.1 4.7 1.5] [5.7 2.9 4.2 1.3] [6.2 2.2 4.5 1.5] [7.3 2.4 7.1 4] [5.8 2.7 5.1 1.9] [5.4 3.4 1.7 0.2] [5.3 1.6 0.2] [6.1 2.6 5.6 1.4] [6.1 2.8 4.1 3] [7.2 3.5 8.1 6] [5.7 2.6 3.5 1.] [6.3 2.8 5.1 1.5] [6.4 3.1 5.5 1.8] [6.3 2.5 4.9 1.5] [6.7 3.1 5.6 2.4] [4.9 3.6 1.4 0.1]]
```

#### Training Data Output

```
[0 2 2 0 0 2 0 2 2 1 1 2 2 0 1 1 2 1 2 1 0 0 0 2 0 1 2 2 0 0 1 0 2 1 2 2 1 0 1 0 1 1 0 1 0 0 2 2 2 0 0 1 0 2 0 2 2 0 2 0 1 0 1 1 0 0 1 0 1 1 1 1 2 0 0 2 1 2 1 2 2 1 2 0]
```

#### Y Predictions:

```
[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 2 0 1 0 1 2 2 0 1 2 1 2 0 0 0 1 0 0 2 2 2 2 2 1 2 1]
```

#### Y Test:

```
[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 2 0 1 0 1 2 2 0 2 2 1 2 0 0 0 1 0 0 2 2 2 2 2 1 2 1]
```

#### Accuracy

```
0.9833333333333333
```

