Q.1) Write a program to implement I/O decorator for converting uppercase letters to Lowercase letters.

```
1.Create Interface:
public interface ToLowerDecorator {
	public void lower(String ch);
}
```

2. LowerCase.java

```
package javaprograms;
import java.lang.*;
import java.io.*;
public class LowerCase implements ToLowerDecorator{
	public void lower(String ch)
	{
		ch=ch.toLowerCase();
		System.out.println("Lowercase:"+ch);
	}
}
3.Decorator.java
package javaprograms;
import java.io.*;
import java.util.Scanner;
public class Decorator {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		ToLowerDecorator l=new LowerCase();
		//l.lower("HeLLO");
		Scanner sc=new Scanner(System.in);
	 System.out.println("enter character:");
	 String   s=sc.nextLine();
	 System.out.println("entered character:"+s);
	   l.lower(s);
	}
}
```

Output –

```
enter character:
HELLO
entered character:HELLO
Lowercase:hello
```

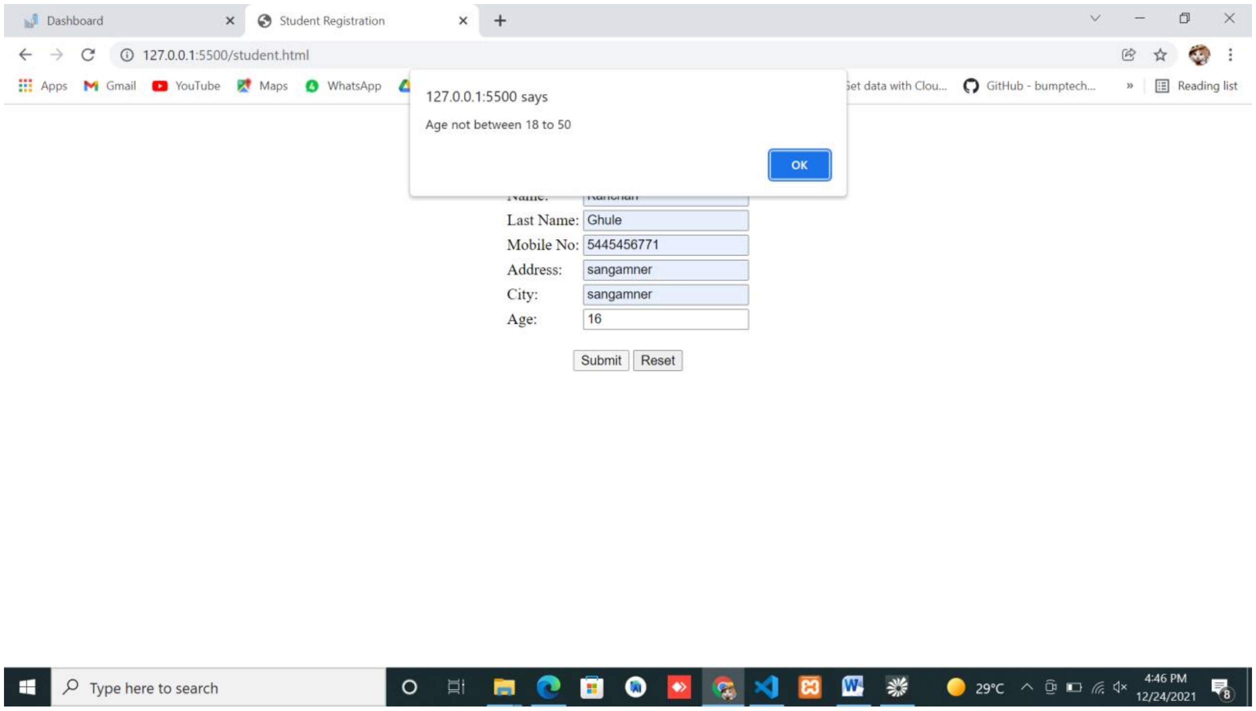
2) Write a python program to prepare scatter plot for Iris dataset.

```
import pandas as pd
import matplotlib.pyplot as plt
iris=pd.read_csv("Iris.csv")
iris.plot(kind="scatter",x='SepalLengthCm',y='PetalLengthCm')
```

```
plt.grid()
plt.show()
```

Q.3) Create an HTML form that contain the Student Registration details and write a JavaScript to validate Student first and last name as it should not contain other than alphabets and age should be between 18 to 50.

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Registration</title>
</head>
<body>
  <script type="text/javascript">
    function validation() {
      var name = document.frm.name.value;
      var lname = document.frm.lname.value;
      var mobile = document.frm.mob.value;
      var address = document.frm.add.value;
      var city = document.frm.city.value;
      var age = document.frm.age.value;
      if (name != "" && lname != "" && mobile != "" && address != "" && city != "" && age
!= "") {
        if (!name.match(/^[a-z A-Z]+$/)) {
          alert("Invalid name fields");
        } else if (!lname.match(/^[a-z A-Z]+$/)) {
          alert("Invalid last name fields");
        } else if (!mobile.match(/^\d{10}$/)) {
          alert("Invalid mobile no fields");
        }else if(age>=18 && age<=50) {
          alert("Submit succesfully......");
        } else {
            alert("Age not between 18 to 50")
        }
      }
      else {
        alert("Invalid fields");
      }
    }
  </script>
  <center>
    <h1>Student Registration Form</h1>
    <form name="frm">
      <table>
        <tr>
          <td>Name: </td>
          <td><input type="text" name="name"></td>
        </tr>
        <tr>
          <td>Last Name: </td>
          <td><input type="text" name="lname"></td>
        </tr>
        <tr>
          <td>Mobile No: </td>
          <td><input type="text" name="mob"></td>
        </tr>
        <tr>
          <td>Address: </td>
          <td><input type="text" name="add"></td>
        </tr>
        <tr>
          <td>City: </td>
          <td><input type="text" name="city"></td>
        </tr>
        <tr>
          <td>Age: </td>
          <td><input type="text" name="age"></td>
        </tr>
      </table><br>
      <input type="button" name="b1" onclick="validation()" value="Submit">
      <input type="reset" name="b2">
    </form>
</center>
</body>
</html>
```



```
127.0.0.1:5500 says
Age not between 18 to 50
```

Last Name:	Ghule
Mobile No:	5445456771
Address:	sangamner
City:	sangamner
Age:	16

Q.4) Write a java program to implement singleton pattern for multithreading.

```
package javaprograms;
public class SingletoneTest
{
    private static final int        PROCESSOR_COUNT =
Runtime.getRuntime().availableProcessors();
    private static final Thread[]   THREADS         = new
Thread[PROCESSOR_COUNT];
    private static int              instancesCount  = 0;
    private static SingletoneTest    instance        = null;

    /*** private constructor to prevent Creation of Object from
Outside of the * This class.
     */
    private SingletoneTest()
    {
    }
    /*** return the instance only if it does not exist */
    public static SingletoneTest getInstance()
    {
        if (instance == null)
        {
            instancesCount++;
            instance = new SingletoneTest();
        }
        return instance;
    }
    /*** reset instancesCount and instance.*/
    private static void reset()
    {
        instancesCount = 0;
        instance = null;
    }
    /*** validate system to run the test*/
    private static void validate()
    {
        if (SingletoneTest.PROCESSOR_COUNT < 2)
        {
            System.out.print("PROCESSOR_COUNT Must be >= 2 to Run the
test.");
            System.exit(0);
        }
    }
    public static void main(String... args)
    {
        validate();
        System.out.printf("Summary :: PROCESSOR_COUNT %s, Running Test
with %s of Threads. %n", PROCESSOR_COUNT, PROCESSOR_COUNT);
        long currentMili = System.currentTimeMillis();
        int testCount = 0;
        do
        {
            reset();
            for (int i = 0; i < PROCESSOR_COUNT; i++)
                THREADS[i] = new Thread(SingletoneTest::getInstance);

            for (int i = 0; i < PROCESSOR_COUNT; i++)
                THREADS[i].start();

            for (int i = 0; i < PROCESSOR_COUNT; i++)
                try
                {
                    THREADS[i].join();
                }
                catch (InterruptedException e)
                {
                    e.printStackTrace();
                    Thread.currentThread().interrupt();
                }
            testCount++;
        }
        while (instancesCount <= 1 && testCount < Integer.MAX_VALUE);

        System.out.printf("Singleton Pattern is broken after %d try.
%nNumber of instances count is %d. %nTest duration %dms", testCount,
instancesCount, System.currentTimeMillis() - currentMili);
    }
}
```

```
Output-
Summary :: PROCESSOR_COUNT 4, Running Test with 4 of Threads.
Singleton Pattern is broken after 144 try.
Number of instances count is 2.
Test duration 232ms
```

5) Write a python program to find all null values in a given dataset & remove them.

```
# importing pandas as pd
import pandas as pd
# importing numpy as np
import numpy as np
# dictionary of lists
```

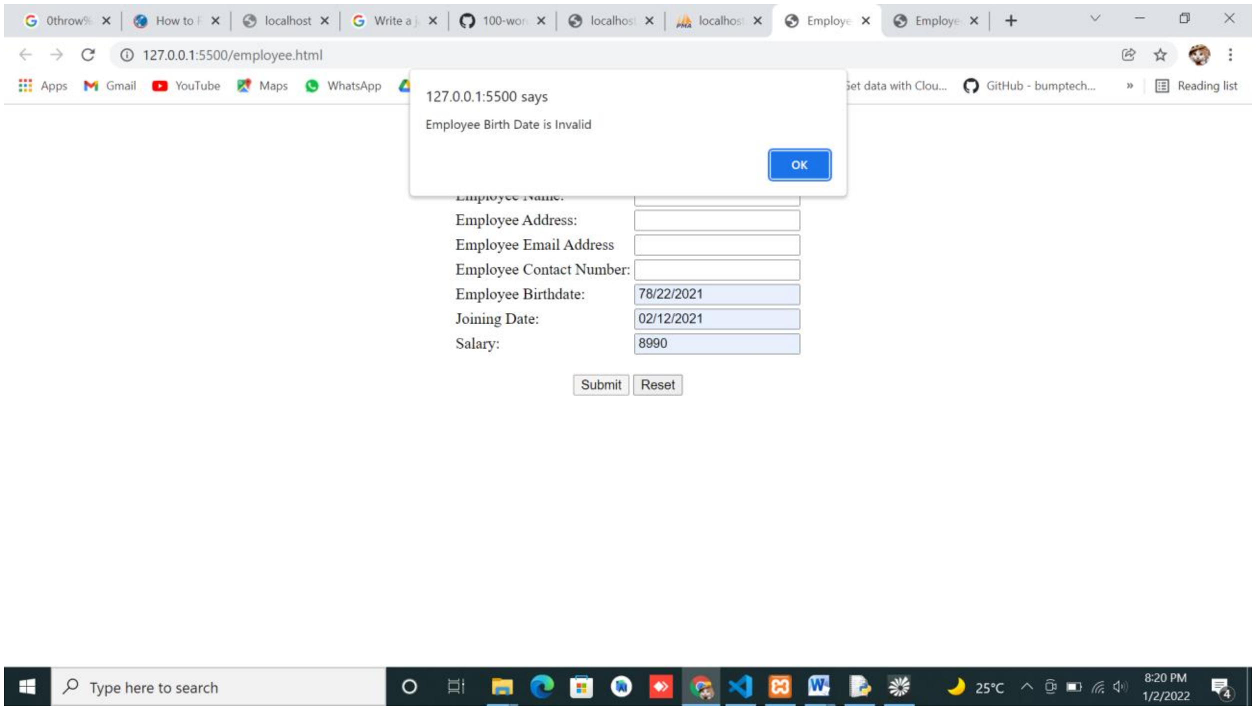
```
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}
# creating a dataframe from list
df = pd.DataFrame(dict)
# using isnull() function
print("\n isnull() function ");
print(df.isnull())
# filling missing value using fillna()
print("\n After filling null values with 0");
print(df.fillna(0))
```

Output-

```
 isnull() function
   First Score  Second Score  Third Score
0        False         False         True
1        False         False        False
2         True         False        False
3        False          True        False

 After filling null values with 0
   First Score  Second Score  Third Score
0        100.0          30.0          0.0
1         90.0          45.0         40.0
2          0.0          56.0         80.0
3         95.0           0.0         98.0
```

Q.6) Create an HTML form that contain the Employee Registration details and write a JavaScript to validate DOB, Joining Date, and Salary.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Employee Details</title>

</head>

<body>
    <center>
        <h1>Employee Registration Form</h1>
        <form name="frm">
            <table>
                <tr>
                    <td>Employee Name: </td>
                    <td><input type="text" name="name"></td>
                </tr>
                <tr>
                    <td>Employee Address: </td>
                    <td><input type="text" name="Address"></td>
                </tr>

                <tr>
                    <td>Employee Email Address</td>
                    <td><input type="text" name="Email"></td>
                </tr>
                <tr>
                    <td>Employee Contact Number: </td>
                    <td><input type="text" name="Telephone"></td>
                </tr>
                <tr>
                    <td>Employee Birthdate: </td>
                    <td><input type="text" name="bdate"
placeholder="dd/mm/yyyy"></td>
                </tr>
                <tr>
                    <td>Joining Date: </td>
                    <td><input type="text" name="jdate"
placeholder="dd/mm/yyyy"></td>
                </tr>
                <tr>
                    <td>Salary: </td>
                    <td><input type="text" name="salary" ></td>
                </tr>
            </table><br>
            <input type="button" name="b1" onclick="validate()" value="Submit">
            <input type="reset" name="b2">
        </form>
    </center>
    <script>
        function validate() {
    var bdate = document.frm.bdate.value;
    var jdate = document.frm.jdate.value;
    var salary = document.frm.salary.value;
    let date = /^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/;
    if (bdate != "" && jdate != "" && salary != "") {
      if (isNaN(salary)) {
        alert("Enter only digit please");
        } else         if(bdate.search(date) == -1){
            alert("Employee Birth Date is Invalid");
        }
        else if(jdate.search(date) == -1){
            alert("Employee Join Date is Invalid");
        }else
         {
            alert("Submit succesfully......");
        }
    }
    else {
        alert("Invalid fields");
    }
}
    </script>
</body>
</html>
```



127.0.0.1:5500 says

Employee Birth Date is Invalid

OK

Employee Address:	
Employee Email Address	
Employee Contact Number:	
Employee Birthdate:	78/22/2021
Joining Date:	02/12/2021
Salary:	8990

Submit Reset

7) Write a java program to implement built-in support (java.util.observable) Weather station with members temperature, humidity, pressure & methods measurementchanged(), setmeasurements(), gettemperature(), gethumidity(), getpressure().

1.Create Interface

Observer.java

```
package javaprograms;
public interface Observer {
      public void update(float temp,float humidity,float pressure);
}
```

```
Displayelement.java
package hello;

public interface DisplayElement {

	public void display();
}
Subject.java
package hello;

public interface Subject {

	public void registerObserver(Observer o);
	public void removeObserver(Observer o);
	public void notifyObservers();
}
```

2. create classes

```
CurrentConditionDispaly.java
package hello;

public class CurrentConditionDispaly implements
Observer,DisplayElement {

	private float temprature;
	private float humidity;
	private Subject  weatherData;

	public CurrentConditionDispaly(Subject weatherData)
{
		this.weatherData=weatherData;
		weatherData.registerObserver(this);


}

	public void update(float temprature,float humidity,float
pressure) {
		this.temprature=temprature;
		this.humidity=humidity;
		display();
	}
	public void display()
	{
		System.out.println("current conditions:"+temprature+"F
degree and "+humidity+"% humidity");
```

```
      }


}
```

ForecastDisplay.java

```
package hello;

public class ForecastDisplay implements Observer,DisplayElement {

      private float  currentpressure=29.92f;
      private float lastpressure;
      private WeatherData weatherData;

      public ForecastDisplay(WeatherData weaherdata) {
            this.weatherData=weatherData;
            weatherData.registerObserver(this);
      }
      public void update(float temp,float humidity,float pressure) {
            lastpressure=currentpressure;
            currentpressure=pressure;
            display();
      }

      public void display()
      {
            System.out.println("forecast:");
            if(currentpressure > lastpressure) {
                  System.out.println("improving weather on the way!.");
            }else if(currentpressure==lastpressure) {
                  System.out.println("more of the same");
            }else if(currentpressure < lastpressure) {
                  System.out.println("watch out for cooler,rainy 
weather");
            }

      }

}
```

HeatIndexDisplay.java

```
package hello;

public class HeatIndexDisplay implements Observer,DisplayElement {
      float heatIndex=0.0f;
      private WeatherData weatherData;
      public HeatIndexDisplay(WeatherData weatherData) {
```

```
		this.weatherData=weatherData;
		weatherData.registerObserver(this);
	  }
	public void update(float t,float rh,float pressure) {
		heatIndex=computeHeatIndex(t,rh);
		display();

	}
	private float computeHeatIndex(float t,float rh) {
		float index=(float)((16.923 + (0.185212 * t) + (5.37941 *
rh) - (0.100254 * t * rh) + (0.000345372 *(t * t * rh ))) +(0.00728898
* (rh * rh)) + (0.000345372 * (t * t * rh))  - (0.000814971 * (t * rh
* rh))+ (0.0000102102 * (t * t * rh * rh)) -(0.000038646 * (t * t *
t))  + (0.0000291683 * (rh * rh * rh)) + (0.00000142721 * (t * t * t *
rh )) + (0.000000197483 * (t * rh * rh * rh)) - (0.000000218429 * (t *
t * t * rh * rh )) + (0.000000000843296 * (t * t * rh * rh * rh)) -
(0.0000000000481975 * (t * t * t * rh * rh * rh)));
	return index;
	}
	public void display()
	{
		System.out.println("heat index"+heatIndex);
	}
}

StatisticDisplay.java
package hello;

public class StatisticDisplay  implements Observer,DisplayElement {
	 private float maxTemp=0.0f;
	 private float minTemp=200;
	 private float tempSum=0.0f;
	 private int numReadings;
	private WeatherData weatherData;
	public StatisticDisplay(WeatherData weatherData) {
	 this.weatherData=weatherData;
	 weatherData.registerObserver(this);
	}

	public void update(float temp,float humidity,float pressure)
	{
	 tempSum=temp;
	 numReadings++;

	 if(temp > maxTemp)
	 {
			maxTemp=temp;
		}
		 if(temp < minTemp) {
			minTemp=temp;
		 }
		 display();
		}

		public void display()
		{
		 System.out.println("AVG?MIN?MAX 
temprature="+(tempSum/numReadings )+"/"+maxTemp+"/"+minTemp);

		}
}
```

WeatherData.java

```
package hello;

import java.util.ArrayList;

public class WeatherData  implements Subject{
	private ArrayList<Observer> observers;
	private float temprature;
	private  float humidity;
	private float pressure;


	public WeatherData() {
		observers=new ArrayList<>();
	}

	public void registerObserver(Observer o) {
		observers.add(o);
	}

	public void removeObserver(Observer o) {
		int i=observers.indexOf(o);
		if(i>=0) {
			observers.remove(i);
		}
	}

public void notifyObservers() {
	for(int i=0;i<observers.size();i++) {
		Observer observer=(Observer)observers.get(i);
```

```
            observer.update(temprature, humidity, pressure);
      }

}
public void measurementChanged() {
      notifyObservers();
}
public void setMeasurement(float temprature,float humidity,float
pressure) {
this.temprature=temprature;
this.humidity=humidity;
this.pressure=pressure;
measurementChanged();
}
public float getTemprature()
{
  return temprature;
}
public float gethumidity()
{
return  humidity;
}
public float getpressure()
{
return pressure;
}
}
```

```
WeatherStation.java
package hello;
import  java.io.*;
public class WeatherStation {

      public static void main(String[] args) {
            // TODO Auto-generated method stub
            //try {
            WeatherData weatherData=new WeatherData();
            CurrentConditionDispaly currentDisplay=new
CurrentConditionDispaly(weatherData);
            StatisticDisplay statisticDisplay=new
StatisticDisplay(weatherData);
            weatherData.setMeasurement(80,65,30.4f);
            weatherData.setMeasurement(82, 70,29.2f);
            weatherData.setMeasurement(78,90,29.2f);
      }
}
```

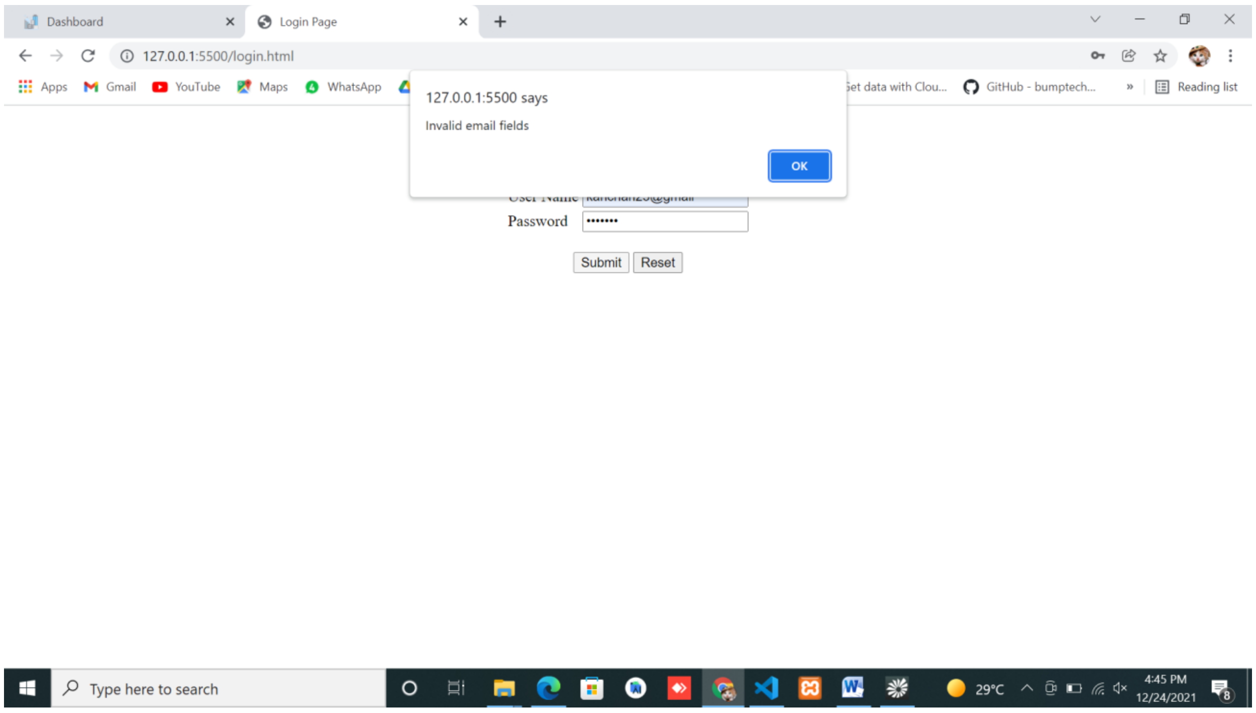
Output-

```
current conditions:80.0F degree and 65.0% humidity
AVG?MIN?MAX temprature=80.0/80.0/80.0
current conditions:82.0F degree and 70.0% humidity
AVG?MIN?MAX temprature=41.0/82.0/80.0
current conditions:78.0F degree and 90.0% humidity
AVG?MIN?MAX temprature=26.0/82.0/78.0
```

Q.9) Create an HTML form for Login and write a JavaScript to validate email ID using Regular Expression.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Login Page</title>
</head>
<body>
   <center>
      <h1>Login Form</h1>
      <form name="frm">
         <table>
            <tr>
               <td>User Name</td>
               <td><input type="text" name="Email"></td>
            </tr>
            <tr>
               <td>Password </td>
               <td><input type="text" name="password"></td>
            </tr>

         </table><br>
         <input type="button" name="b1" onclick="validate()" value="Submit">
         <input type="reset" name="b2">
      </form>
   </center>
   <script>
      function validate() {
         var email = document.frm.Email.value;
         var password = document.frm.password.value;
         var filter = /^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/;
         if (email != "" && password != "") {
            if (!email.match(filter)) {
               alert("Invalid email fields");
            } else if (password.length < 6 || password.length > 8) {
```

```
          alert("Password min and max length is 6.");
        }
        else {
          alert("Thank you for Login")
        }
      }
      else {
        alert("Invalid fields");
      }
    }
  </script>
</body>
</html>
```



127.0.0.1:5500 says

Invalid email fields

Q.10) Write a Java Program to implement Factory method for Pizza Store with createPizza(), orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizzas like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.

Create Class –

1)Pizza.class

```
package javaprograms;
import java.util.ArrayList;

abstract public class Pizza {
    String name;
    String dough;
    String sauce;
    ArrayList toppings = new ArrayList();

    public String getName() {
        return name;
    }

    public void prepare() {
        System.out.println("Preparing " + name);
    }

    public void bake() {
        System.out.println("Baking " + name);
    }

    public void cut() {
        System.out.println("Cutting " + name);
    }

    public void box() {
        System.out.println("Boxing " + name);
    }

    public String toString() {
        // code to display pizza name and ingredients
        StringBuffer display = new StringBuffer();
        display.append("---- " + name + " ----\n");
        display.append(dough + "\n");
        display.append(sauce + "\n");
        for (int i = 0; i < toppings.size(); i++) {
            display.append((String) toppings.get(i) + "\n");
        }
        return display.toString();
    }
}
```
2)PizzaStore.class
```
package javaprograms;

public class PizzaStore {
    SimplePizzaFactory factory;

    public PizzaStore(SimplePizzaFactory factory) {
        this.factory = factory;
    }

    public Pizza orderPizza(String type) {
        Pizza pizza;
        pizza = factory.createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();

        return pizza;
    }
}
3)SimplePizzaFactory.class
package javaprograms;

public class SimplePizzaFactory {

    public Pizza createPizza(String type) {
        Pizza pizza = null;

        if (type.equals("cheese")) {
            pizza = new NYCheesePizza();
        }  else if (type.equals("veggie")) {
            pizza = new ChicagoCheesePizza();
        }
        return pizza;
    }
}
4)NYCheesePizza.class
package javaprograms;
public class NYCheesePizza extends Pizza {
    public NYCheesePizza() {
        name = "NY Cheese Pizza";
        dough = "Regular Crust";
        sauce = "Marinara Pizza Sauce";
        toppings.add("Fresh Mozzarella");
        toppings.add("Parmesan");
    }
}
5)ChicagoCheesePizza.class
package javaprograms;
public class ChicagoCheesePizza extends Pizza {
    public ChicagoCheesePizza() {
        name = "Chicago Cheese Pizza";
        dough = "Crust";
        sauce = "Marinara sauce";
        toppings.add("Shredded mozzarella");
        toppings.add("Grated parmesan");
        toppings.add("Diced onion");
        toppings.add("Sliced mushrooms");
        toppings.add("Sliced red pepper");
        toppings.add("Sliced black olives");
    }
}
```

Output-

```
Preparing NY Cheese Pizza
Baking NY Cheese Pizza
Cutting NY Cheese Pizza
Boxing NY Cheese Pizza
We ordered a NY Cheese Pizza

Preparing Chicago Cheese Pizza
Baking Chicago Cheese Pizza
Cutting Chicago Cheese Pizza
Boxing Chicago Cheese Pizza
We ordered a Chicago Cheese Pizza
```

11) Write python program to implement simple linear regression for predicting house price.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#sns.set_style("whitegrid")
#plt.style.use("fivethirtyeight")

USAhousing = pd.read_csv('USA_Housing.csv')
USAhousing.head()

X = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area
Number of Rooms',
               'Avg. Area Number of Bedrooms', 'Area Population']]
y = USAhousing['Price']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.4, random_state=101)
from sklearn.linear_model import LinearRegression
```

```
lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)


pred = lin_reg.predict(X_test)
plt.scatter(y_test, pred)
plt.show()
```

12) Create nodejs file that will convert the output 'Hello World!' into uppercase letters.

```
var http= require('http');
var uc=require('upper-case');
http.createServer(function(req,res){
    res.writeHead(200,{'Content-type':'text/html'});
    res.write(uc.upperCase("Hello World"));
    res.end();
}).listen(8080);
```

Output-
HELLO WORLD

13) **Write a Java Program to implement Adapter pattern for Enumeration iterator.**

14) Write a python program to implement multiple Linear Regression for a given dataset.

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pylab as pl

df = pd.read_csv('Fuelconsumption.csv')
df.head()
cdf =
df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_CITY','FUELCONSUMPTION_H
WY','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
cdf.head()

plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS, color='blue')
plt.xlabel('Engine Size')
plt.ylabel('Emissions')
plt.show()

msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]
```

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
x =
np.asanyarray(train[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB']]
)
y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit(x,y)

print('Coefficients: ', regr.coef_)
```

15) Using nodejs create a web page to read two file names from user and append contents of first file into second file.

```
const fs = require('fs');

// open destination file for appending
var write = fs.createWriteStream("message.txt", {flags: 'a'});
// open source file for reading
var read = fs.createReadStream("input.txt");
write.on('close', function() {
    console.log("done writing");
});
read.pipe(write);
```

16) Write a Java Program to implement command pattern to test Remote Control.
1)Create Interface-
Command.java

```
package javaprograms;

interface Command
{
    public void execute();
}

2)Create Class
Light.java
package javaprograms;
public class Light
{
    public void on()
    {
        System.out.println("Light is on");
    }
    public void off()
    {
        System.out.println("Light is off");
    }
}
```

LightOnCommand.java

```java
package javaprograms;
class LightOnCommand implements Command
{
    Light light;

    // The constructor is passed the light it
    // is going to control.
    public LightOnCommand(Light light)
    {
       this.light = light;
    }
    public void execute()
    {
       light.on();
    }
}
```

LightOffCommand.java

```java
package javaprograms;
class LightOffCommand implements Command
{
    Light light;
    public LightOffCommand(Light light)
    {
        this.light = light;
    }
    public void execute()
    {
         light.off();
    }
}
```

Stereo.java

```java
package javaprograms;

public class Stereo
{
    public void on()
    {
        System.out.println("Stereo is on");
    }
    public void off()
    {
        System.out.println("Stereo is off");
    }
    public void setCD()
    {
        System.out.println("Stereo is set " +
                           "for CD input");
    }
    public void setDVD()
    {
        System.out.println("Stereo is set"+
                                 " for DVD input");
    }
    public void setRadio()
    {
        System.out.println("Stereo is set" +
                                   " for Radio");
    }
    public void setVolume(int volume)
    {
       // code to set the volume
       System.out.println("Stereo volume set"
                                  + " to " + volume);
    }
}
```

StereoOffCommand.java

```
package javaprograms;

class StereoOffCommand implements Command
{
    Stereo stereo;
    public StereoOffCommand(Stereo stereo)
    {
        this.stereo = stereo;
    }
    public void execute()
    {
       stereo.off();
    }
}
```

StereoOnWithCDCommand.java

```
package javaprograms;

class StereoOnWithCDCommand implements Command
{
     Stereo stereo;
     public StereoOnWithCDCommand(Stereo stereo)
     {
         this.stereo = stereo;
     }
     public void execute()
     {
         stereo.on();
         stereo.setCD();
         stereo.setVolume(11);
     }
}
```

SimpleRemoteControl.java

```
package javaprograms;
class SimpleRemoteControl
{
    Command slot;  // only one button

    public SimpleRemoteControl()
    {
    }

    public void setCommand(Command command)
    {
        // set the command the remote will
        // execute
        slot = command;
    }
    public void buttonWasPressed()
    {
        slot.execute();
    }
}
```

```
RemoteControlTest.java
package javaprograms;

class RemoteControlTest
{
    public static void main(String[] args)
    {
        SimpleRemoteControl remote =
                  new SimpleRemoteControl();
        Light light = new Light();
        Stereo stereo = new Stereo();

        // we can change command dynamically
        remote.setCommand(new
                    LightOnCommand(light));
        remote.buttonWasPressed();
        remote.setCommand(new
                StereoOnWithCDCommand(stereo));
        remote.buttonWasPressed();
        remote.setCommand(new
                   StereoOffCommand(stereo));
        remote.buttonWasPressed();
     }
  }
```

```
Output-
Light is on
Stereo is on
Stereo is set for CD input
Stereo volume set to 11
Stereo is off
```

17) Write a python program to implement Polynomial Regression for given dataset.

18) Create a Nodejs file that opens the requested file and returns the content to the client . If anything goes wrong throw 404 error.

Fileprogram.js
```
var http = require('http');
var url = require('url');
var fs = require('fs');

http.createServer(function (req, res) {
  var q = url.parse(req.url, true);
  var filename = "." + q.pathname;
  fs.readFile(filename, function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

summer.html
```
<!DOCTYPE html>
<body>
<h1>Summer</h1>
<p>I love the sun!</p>
</body>
</html>
```

winter.html
```
<!DOCTYPE html>
<body>
<h1>Winter</h1>
<p>I love the snow!</p>
</body>
</html>
```

19) Write a Java Program to implement undo command to test Ceiling fan.
20) Write a python program to Implement Naïve Bayes.
```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB

dataset = datasets.load_iris()

#Creating our Naive Bayes Model
model = GaussianNB()
```

```
model.fit(dataset.data, dataset.target)

#Making Predictions
expected = dataset.target
predicted = model.predict(dataset.data)

#Getting Accuracy and Statistics
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

21) Create Nodejs file that write an HTML form with an upload fields.

```
var http = require('http');
var formidable = require('formidable');

http.createServer(function (req, res) {
  if (req.url == '/fileupload') {
    var form = new formidable.IncomingForm();
    form.parse(req, function (err, fields, files) {
      res.write('File uploaded');
      res.end();
    });
  } else {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<form action="fileupload" method="post" enctype="multipart/form-
data">');
    res.write('<input type="file" name="filetoupload"><br>');
    res.write('<input type="submit">');
    res.write('</form>');
    return res.end();
  }
}).listen(8080);
```

22) Write a Java Program to implement State Pattern for Gumball Machine. Create instance variable that holds current state from there, we just need to handle all actions, behaviors and state transition that can happen. For actions we need to implement methods to insert a quarter, remove a quarter, turning the crank and display gumball.
1) Create Interface-
State.java

```
package javaprograms;


public interface State {

	public void insertQuarter();
	public void ejectQuarter();
```

```
        public void turnCrank();
        public void dispense();

        public void refill();
}
```

2) Create Class

SoldState.java

```
package javaprograms;

public class SoldState implements State {

    GumballMachine gumballMachine;

    public SoldState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

        public void insertQuarter() {
                System.out.println("Please wait, we're already giving you a gumball");
        }

        public void ejectQuarter() {
                System.out.println("Sorry, you already turned the crank");
        }

        public void turnCrank() {
                System.out.println("Turning twice doesn't get you another gumball!");
        }

        public void dispense() {
                gumballMachine.releaseBall();
                if (gumballMachine.getCount() > 0) {
                        gumballMachine.setState(gumballMachine.getNoQuarterState());
                } else {
                        System.out.println("Oops, out of gumballs!");
                        gumballMachine.setState(gumballMachine.getSoldOutState());
                }
        }

        public void refill() { }

        public String toString() {
                return "dispensing a gumball";
        }
}
```

SoldOutState.java

```
package javaprograms;


public class SoldOutState implements State {
    GumballMachine gumballMachine;
    public SoldOutState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
             System.out.println("You can't insert a quarter, the machine is sold
out");
      }

      public void ejectQuarter() {
             System.out.println("You can't eject, you haven't inserted a quarter
yet");
      }

      public void turnCrank() {
             System.out.println("You turned, but there are no gumballs");
      }

      public void dispense() {
             System.out.println("No gumball dispensed");
      }

      public void refill() {
             gumballMachine.setState(gumballMachine.getNoQuarterState());
      }

      public String toString() {
             return "sold out";
      }
}
NoQuarterState.java
package javaprograms;

public class NoQuarterState implements State {
    GumballMachine gumballMachine;

    public NoQuarterState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
             System.out.println("You inserted a quarter");
             gumballMachine.setState(gumballMachine.getHasQuarterState());
      }

      public void ejectQuarter() {
             System.out.println("You haven't inserted a quarter");
      }

      public void turnCrank() {
             System.out.println("You turned, but there's no quarter");
       }

      public void dispense() {
             System.out.println("You need to pay first");
```

```
	}

	public void refill() { }

	public String toString() {
		return "waiting for quarter";
	}
}
```

HasQuarterState.java

```
package javaprograms;


public class HasQuarterState implements State {
	GumballMachine gumballMachine;

	public HasQuarterState(GumballMachine gumballMachine) {
		this.gumballMachine = gumballMachine;
	}

	public void insertQuarter() {
		System.out.println("You can't insert another quarter");
	}

	public void ejectQuarter() {
		System.out.println("Quarter returned");
		gumballMachine.setState(gumballMachine.getNoQuarterState());
	}

	public void turnCrank() {
		System.out.println("You turned...");
		gumballMachine.setState(gumballMachine.getSoldState());
	}

    public void dispense() {
        System.out.println("No gumball dispensed");
    }

    public void refill() { }

	public String toString() {
		return "waiting for turn of crank";
	}
}
```

GumballMachine.java

```
package javaprograms;


public class GumballMachine {

	State soldOutState;
	State noQuarterState;
	State hasQuarterState;
	State soldState;
	State state;
	int count = 0;

	public GumballMachine(int numberGumballs) {
		soldOutState = new SoldOutState(this);
		noQuarterState = new NoQuarterState(this);
		hasQuarterState = new HasQuarterState(this);
		soldState = new SoldState(this);

		this.count = numberGumballs;
		if (numberGumballs > 0) {
			state = noQuarterState;
		} else {
			state = soldOutState;
		}
	}

	public void insertQuarter() {
		state.insertQuarter();
	}

	public void ejectQuarter() {
		state.ejectQuarter();
	}

	public void turnCrank() {
		state.turnCrank();
		state.dispense();
	}

	void releaseBall() {
		System.out.println("A gumball comes rolling out the slot...");
		if (count > 0) {
			count = count - 1;
		}
	}

	int getCount() {
		return count;
	}

	void refill(int count) {
		this.count += count;
		System.out.println("The gumball machine was just refilled; its new count 
is: " + this.count);
		state.refill();
	}

	void setState(State state) {
		this.state = state;
	}
     public State getState() {
         return state;
     }
```

```
    public State getSoldOutState() {
        return soldOutState;
    }

    public State getNoQuarterState() {
        return noQuarterState;
    }

    public State getHasQuarterState() {
        return hasQuarterState;
    }

    public State getSoldState() {
        return soldState;
    }

       public String toString() {
              StringBuffer result = new StringBuffer();
              result.append("\nMighty Gumball, Inc.");
              result.append("\nJava-enabled Standing Gumball Model #2004");
              result.append("\nInventory: " + count + " gumball");
              if (count != 1) {
                     result.append("s");
              }
              result.append("\n");
              result.append("Machine is " + state + "\n");
              return result.toString();
       }
}
```

GumballMachineTestDrive.java

```
package javaprograms;

public class GumballMachineTestDrive {

       public static void main(String[] args) {
              GumballMachine gumballMachine = new GumballMachine(2);

              System.out.println(gumballMachine);

              gumballMachine.insertQuarter();
              gumballMachine.turnCrank();

              System.out.println(gumballMachine);

              gumballMachine.insertQuarter();
              gumballMachine.turnCrank();
              gumballMachine.insertQuarter();
              gumballMachine.turnCrank();

              gumballMachine.refill(5);
              gumballMachine.insertQuarter();
              gumballMachine.turnCrank();
```

```
            System.out.println(gumballMachine);
      }
}

Output-
Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 2 gumballs
Machine is waiting for quarter

You inserted a quarter
You turned...
A gumball comes rolling out the slot...

Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 1 gumball
Machine is waiting for quarter

You inserted a quarter
You turned...
A gumball comes rolling out the slot...
Oops, out of gumballs!
You can't insert a quarter, the machine is sold out
You turned, but there are no gumballs
No gumball dispensed
The gumball machine was just refilled; its new count is: 5
You inserted a quarter
You turned...
A gumball comes rolling out the slot...

Mighty Gumball, Inc.
Java-enabled Standing Gumball Model #2004
Inventory: 4 gumballs
Machine is waiting for quarter
```

23) Write a python program to Implement Decision Tree whether or not to play tennis.

```
import numpy as np
import pandas as pd

PlayTennis = pd.read_csv("PlayTennis.csv")

#We can convert all the non numerical values into numerical values using
LabelEncoder

from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()

PlayTennis['outlook'] = Le.fit_transform(PlayTennis['outlook'])
PlayTennis['temp'] = Le.fit_transform(PlayTennis['temp'])
PlayTennis['humidity'] = Le.fit_transform(PlayTennis['humidity'])
```

```
PlayTennis['windy'] = Le.fit_transform(PlayTennis['windy'])
PlayTennis['play'] = Le.fit_transform(PlayTennis['play'])

#Lets split the training data and its coresponding prediction values.
#y - holds all the decisions.
#X - holds the training data.
y = PlayTennis['play']
X = PlayTennis.drop(['play'],axis=1)

# Fitting the model
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)

# We can visualize the tree using tree.plot_tree
tree.plot_tree(clf)

import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph
```

24) Create a Node.js file that demonstrate create database and table in MySQL.

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: ""
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query("CREATE DATABASE mydb", function (err, result) {
    if (err) throw err;
    console.log("Database created");
  });
});
```

25) Design simple HR Application using Spring Framework.

26) Write a python program to implement linear SVM.

```
# Import the Libraries
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets

# Import some Data from the iris Data Set
iris = datasets.load_iris()

# Take only the first two features of Data.
# To avoid the slicing, Two-Dim Dataset can be used

X = iris.data[:, :2]
y = iris.target

# C is the SVM regularization parameter
C = 1.0

# Create an Instance of SVM and Fit out the data.
# Data is not scaled so as to be able to plot the support vectors
svc = svm.SVC(kernel ='linear', C = 1).fit(X, y) #Fit the SVM model according to
the given training data.
                                                  #SVC=Support Vector Classifier

# create a mesh to plot
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),     #Return coordinate matrices
from coordinate vectors.
        np.arange(y_min, y_max, h))

# Plot the data for Proper Visual Representation
plt.subplot(1, 1, 1) #Add a subplot to the current figure.subplot(nrows, ncols,
indexOfSubplot)

# Predict the result by giving Data to the model
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])   #ravel()-Return a contiguous
flattened array.
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap = plt.cm.Paired, alpha = 0.8) #contour and contourf
draw contour lines and filled contours.

plt.scatter(X[:, 0], X[:, 1], c = y, cmap = plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')

# Output the Plot
plt.show()
```

27) Create a node.js file that Select all records from the "customers" table, and display the result object on console.

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});
```

28) Write a java program to implement strategy pattern for duck behavior create instance variable that holds current state of duck from there we just need to handle all flying behavior and quack behavior.

1) Create Interface

```
QuackBehaviour.java
package javaprograms;
public interface QuackBehaviour {
        public default void quack() {
          System.out.println("Quack");
        }
}

FlyBehaviour.java
package javaprograms;

public interface FlyBehaviour {
        public void fly();
}
```

2)Create Class -

```
FlyWithWings.java
package javaprograms;
```

```
public class FlyWithWings implements FlyBehaviour {
        public void fly() {
          System.out.println("I'm flying!!");
        }
}

Quack.java
package javaprograms;

public class Quack implements QuackBehaviour {
        public void quack() {
          System.out.println("Quack");
        }
}

ModolDuck.java
package javaprograms;
public class ModolDuck extends Duck {
        public ModolDuck() {
          flyBehaviour = new FlyNoWay();
          quackBehaviour = new Quack();
        }

        public void display() {
          System.out.println("I'm a model duck");
        }
}

MallardDuck.java
package javaprograms;

public class MallardDuck extends Duck {

        public MallardDuck() {
          quackBehaviour = new Quack();
          flyBehaviour = new FlyWithWings();
        }

        public void display() {
          System.out.println("I'm a real Mallard duck");
        }
}

Duck.java
package javaprograms;

public class MallardDuck extends Duck {

        public MallardDuck() {
          quackBehaviour = new Quack();
          flyBehaviour = new FlyWithWings();
        }

        public void display() {
          System.out.println("I'm a real Mallard duck");
        }
}

FlyRocketPowered.java
package javaprograms;
public class FlyRocketPowered implements FlyBehaviour {
        public void fly() {
          System.out.println("I'm flying with a rocket!");
        }
}

FlyNoWay.java
package javaprograms;
public class FlyNoWay implements FlyBehaviour {
        public void fly() {
          System.out.println("I can't fly");
        }
}

MiniDuckSimulator.java
package javaprograms;

public class MiniDuckSimulator {
        public static void main(String[] args) {
          Duck mallard = new MallardDuck();
          mallard.performQuack();
          mallard.performFly();
          Duck model = new ModolDuck();

          model.performFly();

          model.setFlyBehaviour(new FlyRocketPowered());

          model.performFly();
        }
}

Output-
Quack
I'm flying!!
I can't fly
I'm flying with a rocket!
```

29) Create a node.js file that Insert Multiple Records in "student" table, and display the result object on console.

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "mydb"
});
```

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name, address) VALUES ?";
  var values = [
    ['John', 'Highway 71'],
    ['Peter', 'Lowstreet 4'],
    ['Amy', 'Apple st 652'],
    ['Hannah', 'Mountain 21'],
    ['Michael', 'Valley 345'],
    ['Sandy', 'Ocean blvd 2'],
    ['Betty', 'Green Grass 1'],
    ['Richard', 'Sky st 331'],
    ['Susan', 'One way 98'],
    ['Vicky', 'Yellow Garden 2'],
    ['Ben', 'Park Lane 38'],
    ['William', 'Central st 954'],
    ['Chuck', 'Main Road 989'],
    ['Viola', 'Sideway 1633']
  ];
  con.query(sql, [values], function (err, result) {
    if (err) throw err;
    console.log("Number of records inserted: " + result.affectedRows);
  });
});
```

30) Write a java program to implement Adapter pattern to design Heart Model to Beat Model.

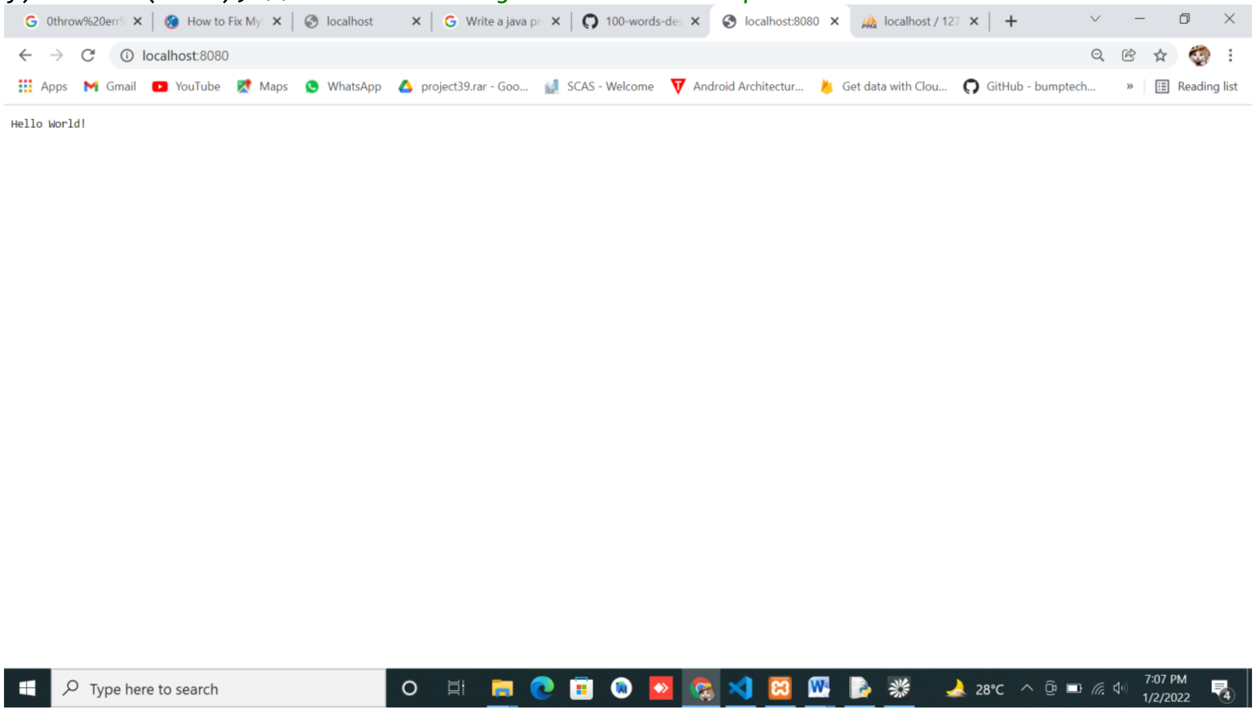
31) Create a node.js file that Select all records from the "customers" table, and delete the specified record.

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "DELETE FROM customers WHERE address = 'Shivajinagar,Sangamner'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Number of records deleted: " + result.affectedRows);
  });
});
```

32) Write a java program to implement decorator pattern for interface car to define the assemble() method & then decorate it to sports car & luxury car.

33) Create a Simple Web Server using node js.

```
var http = require('http');
//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!'); //write a response to the client
  res.end(); //end the response
}).listen(8080); //the server object listens on port 8080
```



```
Hello World!
```

34)

35) Write a java program to implement an adapter design pattern in mobile charger. Define two classes Volt (to measure volts) and Socket (producing constant volts of 120v). Build an adapter that can produce 3 volts, 12 volts and default 120 volts. Implement adapter pattern using class adapter.

36) Using nodejs create a User Login System.

37) Write a java program to implement command design pattern for command interface with execute() use this to create verify of commands for Lighton command, Lightoff command, Doorup command, Storeon with CDCommon.

38) write node js script to interact with the filesystem, and serve a web page from a file.

39) Write java program to implement façade design pattern for Home Theater.

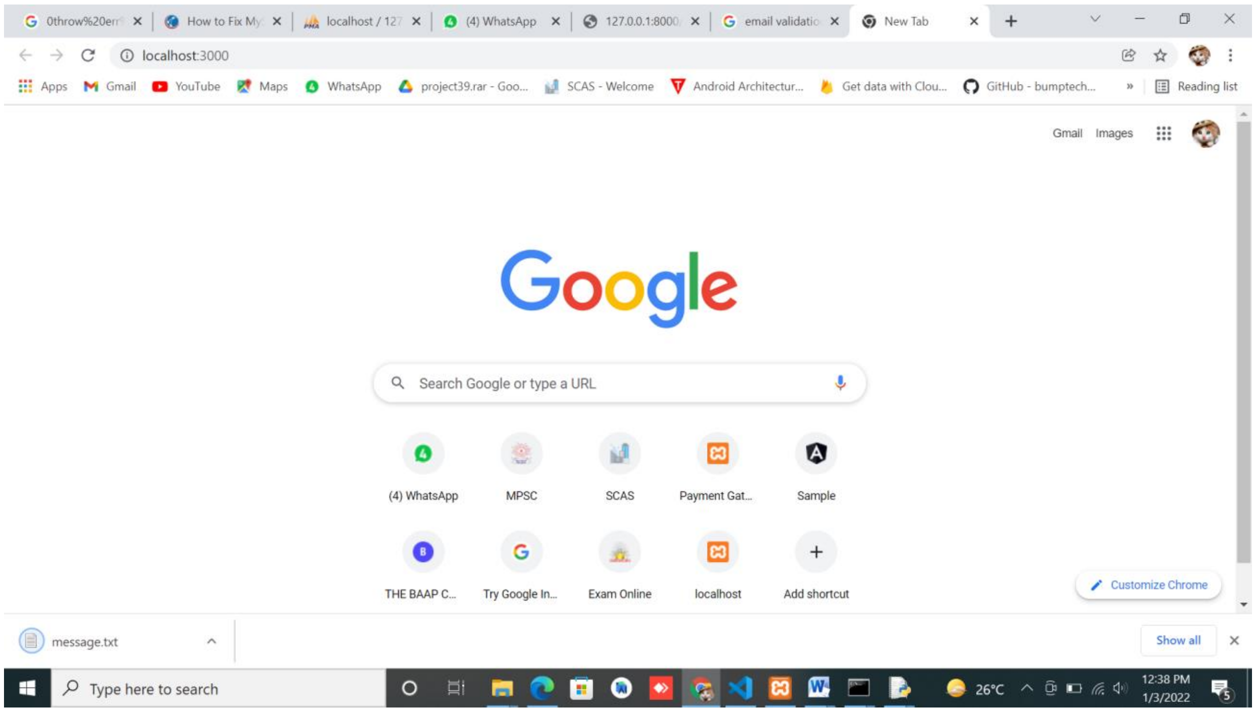
40) Write node js script to build Your Own Node.js Module. Use require (‘http’) module is a built-in Node module that invokes the functionality of the HTTP library to create a local server. Also use the export statement to make functions in your module available externally. Create a new text file to contain the functions in your module called, “modules.js” and add this function to return today’s date and time.

41) Write a python program to implement simple Linear Regression for predicting house price.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#sns.set_style("whitegrid")
#plt.style.use("fivethirtyeight")

USAhousing = pd.read_csv('USA_Housing.csv')
USAhousing.head()

X = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of 
Rooms',
               'Avg. Area Number of Bedrooms', 'Area Population']]
y = USAhousing['Price']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, 
random_state=101)



from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)


pred = lin_reg.predict(X_test)
plt.scatter(y_test, pred)
plt.show()
```

42) Create a js file named main.js for event-driven application. There should be a main loop that listens for events, and then triggers a callback function when one of those events is detected.
43) Write a java program to implement abstract factory pattern for shape interface.
44) Write node js application that transfer a file as an attachment on web and enables browser to prompt the user to download file using express js.

```
var express = require('express');
var app = express();
var PORT = 3000;
app.get('/', function(req, res){
    res.download('message.txt', function(error){
        console.log("Error : ", error)
    });
});
app.listen(PORT, function(err){
    if (err) console.log(err);
    console.log("Server listening on PORT", PORT);
});
```



45) Create your Django app in which after running the server, you should see on the browser, the text "Hello! I am learning Django", which you defined in the index view.
home app
views.py

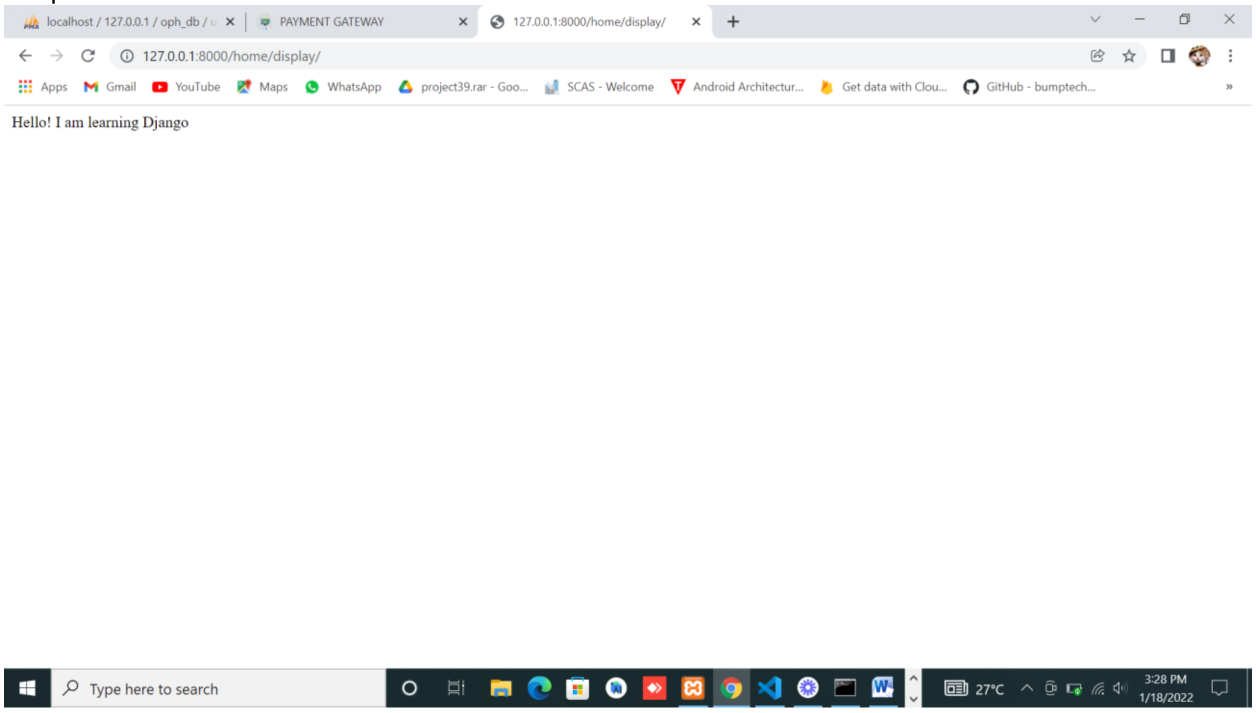
```
from django.shortcuts import render
# Create your views here.
```

```
from django.http import HttpResponse
def display(request):
    return HttpResponse("Hello! I am learning Django")
urls.py
from django.urls import path
from .views import display
urlpatterns = [
    path('display/',display, name="home")
]
```

test_project
urls.py

```
from django.contrib import admin
from django.urls import path
from django.urls import include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('home/', include("home.urls")),
]
```

Output



```
Hello! I am learning Django
```

46) Design a Django application that adds web pages with views and templates.
templates
home.html

```
<h2>Hello {{name}}</h2>
<form action="add/">
```

```
    Enter 1st number<input type="text" name="num1"><br>
    Enter 2nd number<input type="text" name="num2"><br>
    <input type="submit">
</form>
```
result.html
```
result:{{result}}
```

views.py
```
from django.http import HttpResponse
def name(request):
    return render(request,"home.html",{"name":"Akshada"})
def add(request):
    val1=int(request.GET['num1'])
    val2=int(request.GET['num2'])
    res=val1+val2
    return render(request,"result.html",{'result':res})
```

urls.py
```
from django.urls import path
from .views import add,name
urlpatterns = [
    path("", name,name="home"),
    path('add/',add,name="home"),
]
```

test_project
```
from django.contrib import admin
from django.urls import path
from django.urls import include
urlpatterns = [
    path('home/', include("home.urls")),
]
```

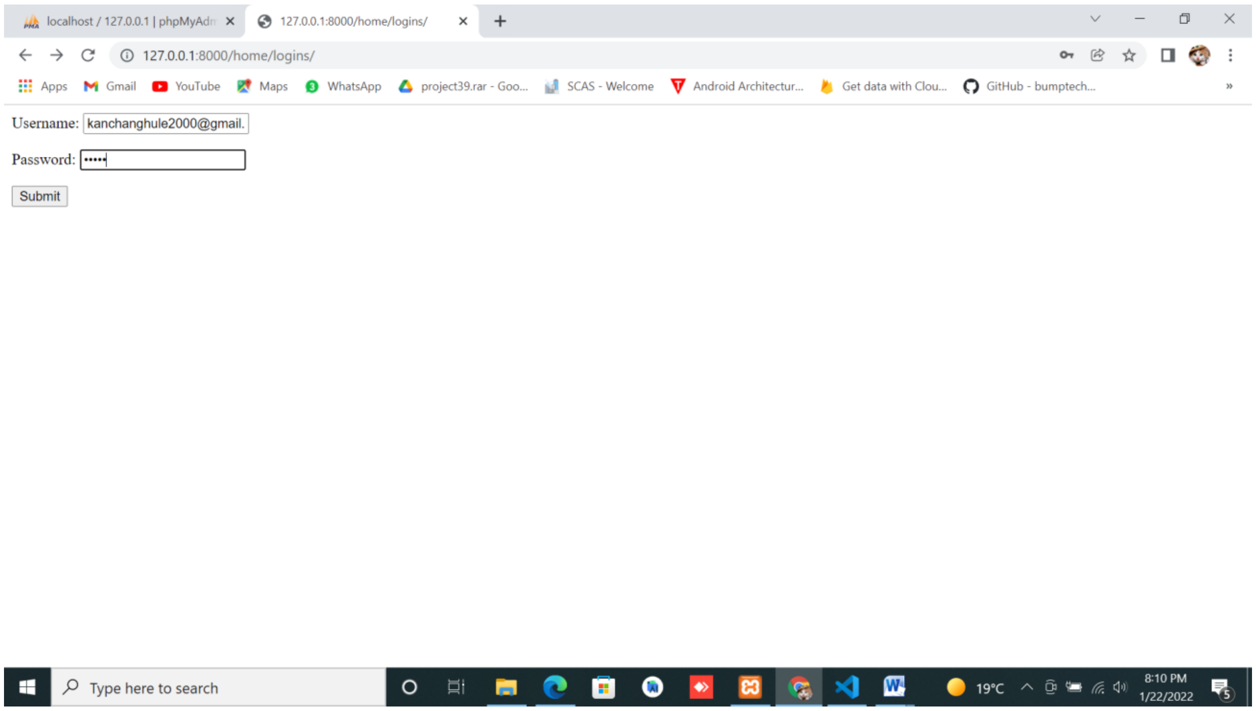
47) Develop a basic poll application (app).It should consist of two parts:
a) A public site in which user can pick their favourite programming language and vote.
b) An admin site that lets you add, change and delete programming languages.
48) Create your own blog using Django.
49) Implement Login System using Django.
home app
forms.py
```
from django import forms

class LoginForm(forms.Form):
    username=forms.CharField(max_length=63)
    password=forms.CharField(max_length=63,widget=forms.PasswordInput)
```

login.html

```
<form method="post">
    {{ form.as_p }}
    {% csrf_token %}
    <button type="submit">Submit</button>
</form>
```

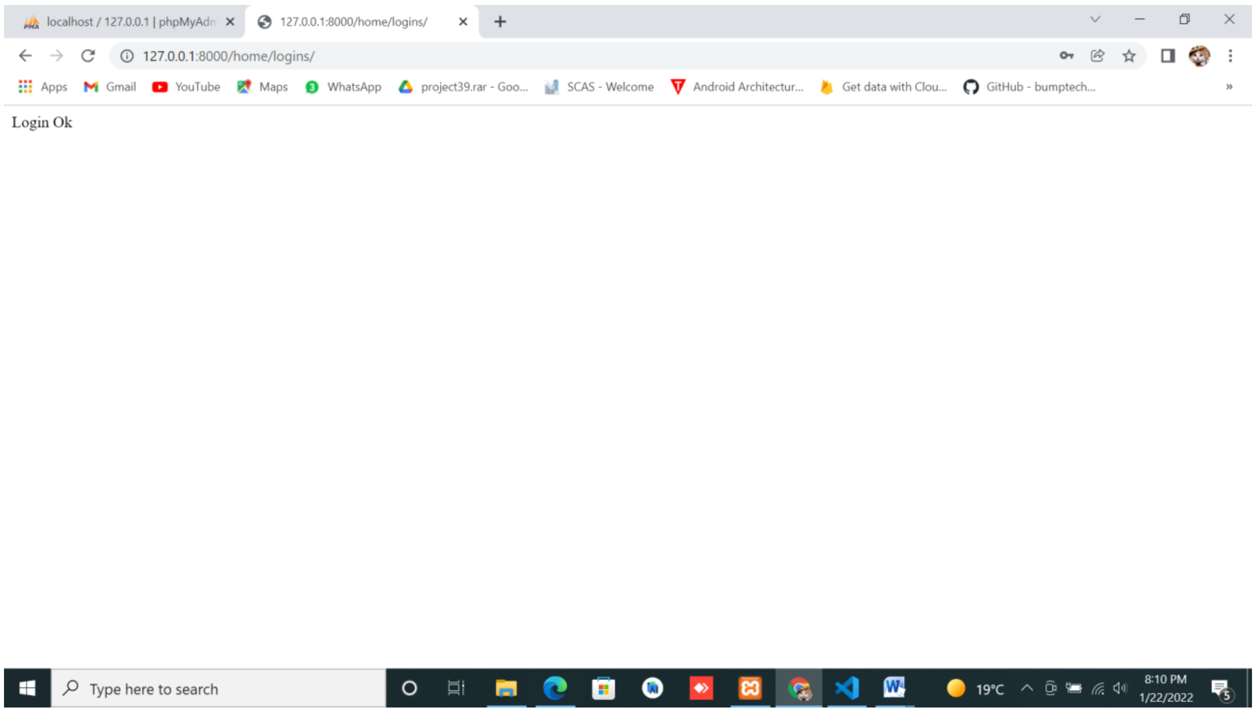
views.py

```
from django.shortcuts import render
from . import forms

def login(request):
    form=forms.LoginForm()
    if request.method=='POST':
        form=forms.LoginForm(request.POST)
        if form.is_valid():
            return HttpResponse("Login Ok")
    return render(request,'login.html',context={'form':form})
urls.py

from django.urls import path
from .views import login
urlpatterns = [
    path('logins/',login, name="home"),
]
test_project
from django.contrib import admin
from django.urls import path
from django.urls import include
urlpatterns = [
    path('home/', include("home.urls")),
]
```


Username: kanchanghule2000@gmail.

Password: •••••

Submit


Login Ok

50) Create a clone of the “Hacker News” website.