**1. 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.**

<html>

<head>

<script type="text/javascript">

function validateStudent(){

var regName = /^[a-zA-z]+[a-zA-Z]+$/;

var fname = document.getElementById('fsname').value;

var lname = document.getElementById('lsname').value;

var age = document.getElementById('age').value;

var phone = document.getElementById('phone').value;

if(age<18 || age>50)

alert("student age must be 18 to 50");

if(!regName.test(fname))

alert("invalid name");

if(!regName.test(lname))

alert("invalid name");

}

</script>

</head>

<body>

<form>

<h1>Student Form :</h1>

student first name :

<input type="text" name="fsname" id="fsname"><br>

student last name :

<input type="text" name="lname" id="lsname"><br>

student age :

<input type="text" name="age" id="age"><br>

student mobile :

<input type="text" name="phone" id="phone"><br>

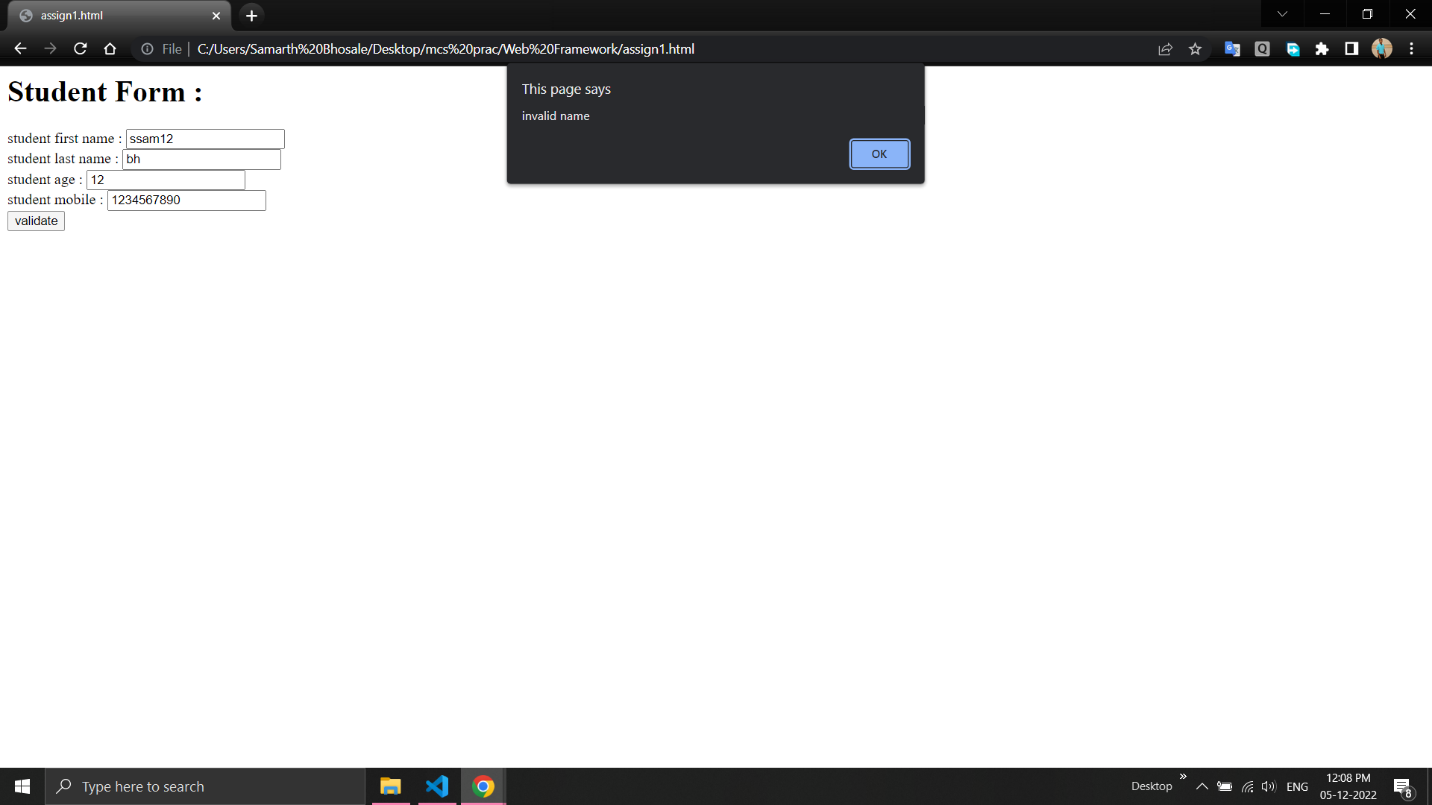
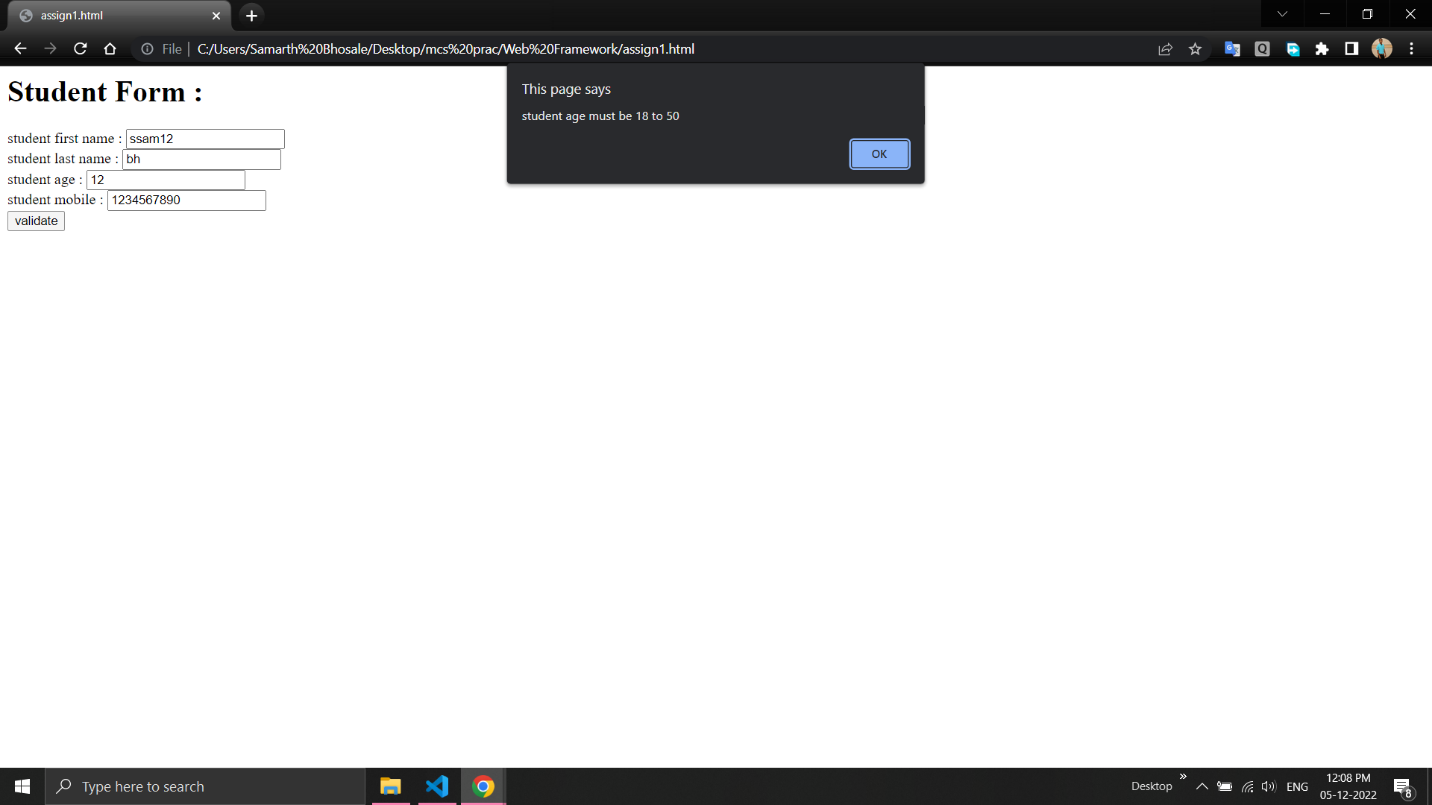
<input type="button" value="validate" onclick="validateStudent()">

</form>

</body>

</html>

**//Output:**



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

<html>

<head>

<script type="text/javascript">

function validate(){

var regName=/^[a-zA-z]+[a-zA-Z]+$/;

var dateformatdob = /^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/;

var dateformatjdate = /^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/;

var salaryformat=/^\d{1,6}(?:\.\d{0,2})?$/

var name=document.getElementById("name").value;

var dob=document.getElementById("dob").value;

var jdate=document.getElementById("jdate").value;

var salary=document.getElementById("salary").value;

if(!regName.test(name))

alert("invalid name is given");

if(!dateformatjdate.test(jdate))

alert("invalid joining date is given");

if(!dateformatdob.test(dob))

alert("invalid date of birth is given");

if(!salaryformat.test(salary))

alert("invalid salary");

}

</script>

</head>

<body>

<form>

<h1>Employee Rsgistration Details</h1>

employee name :

<input type="text" name="fname" id="name"><br>

date of birth :

<input type="text" name="dob" id="dob"><br>

joining date :

<input type="text" name="jdate" id="jdate"><br>

salary :

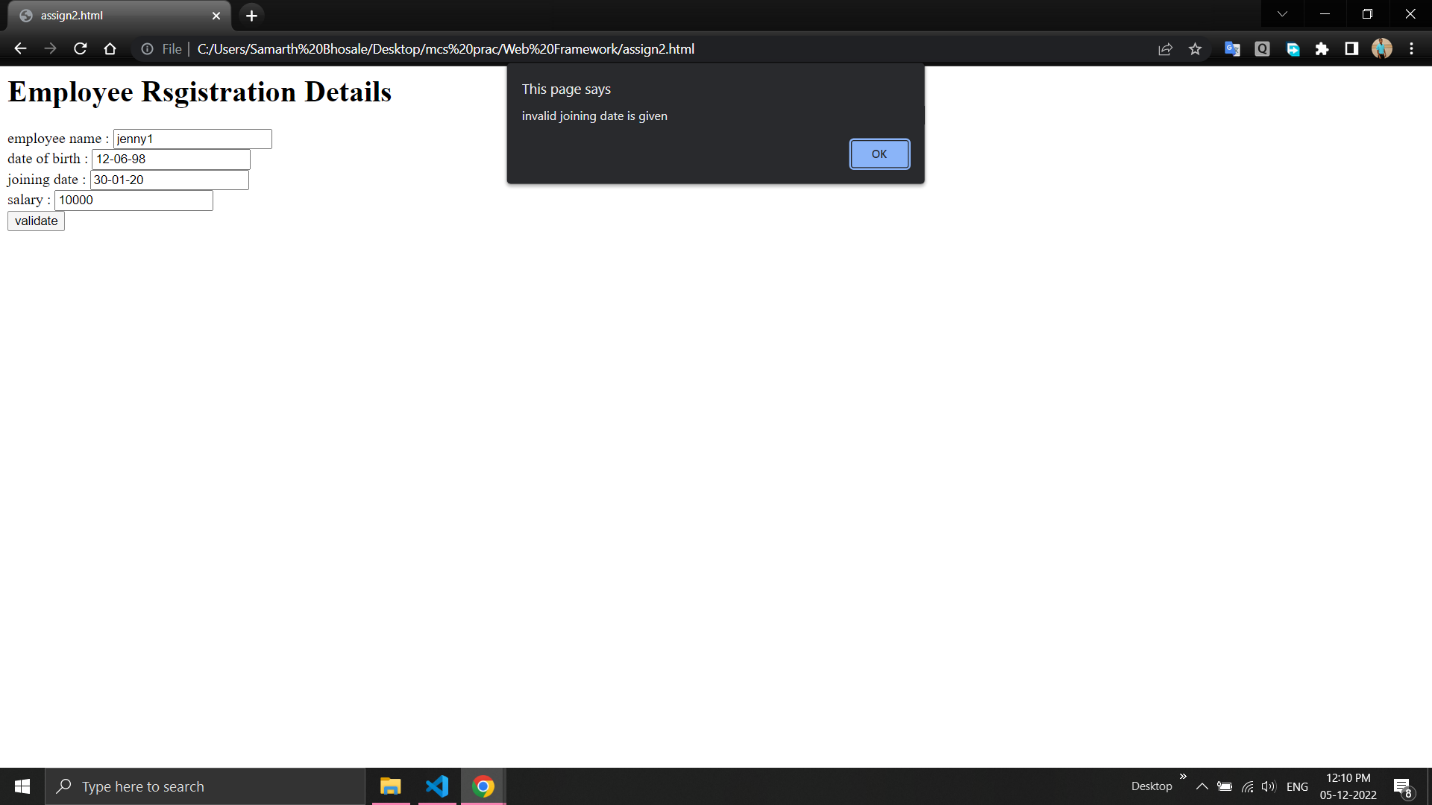
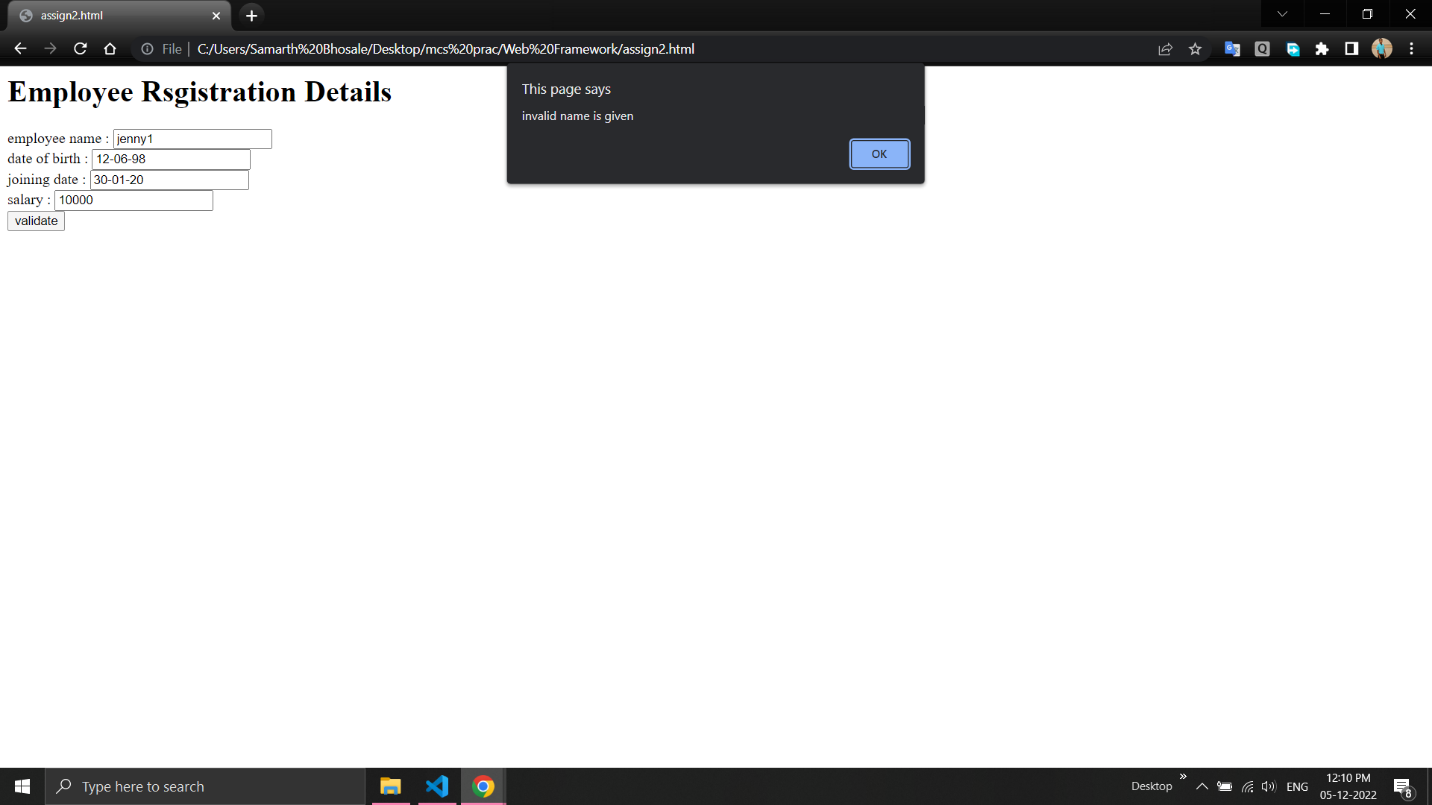
<input type="text" name="salary" id="salary"><br>

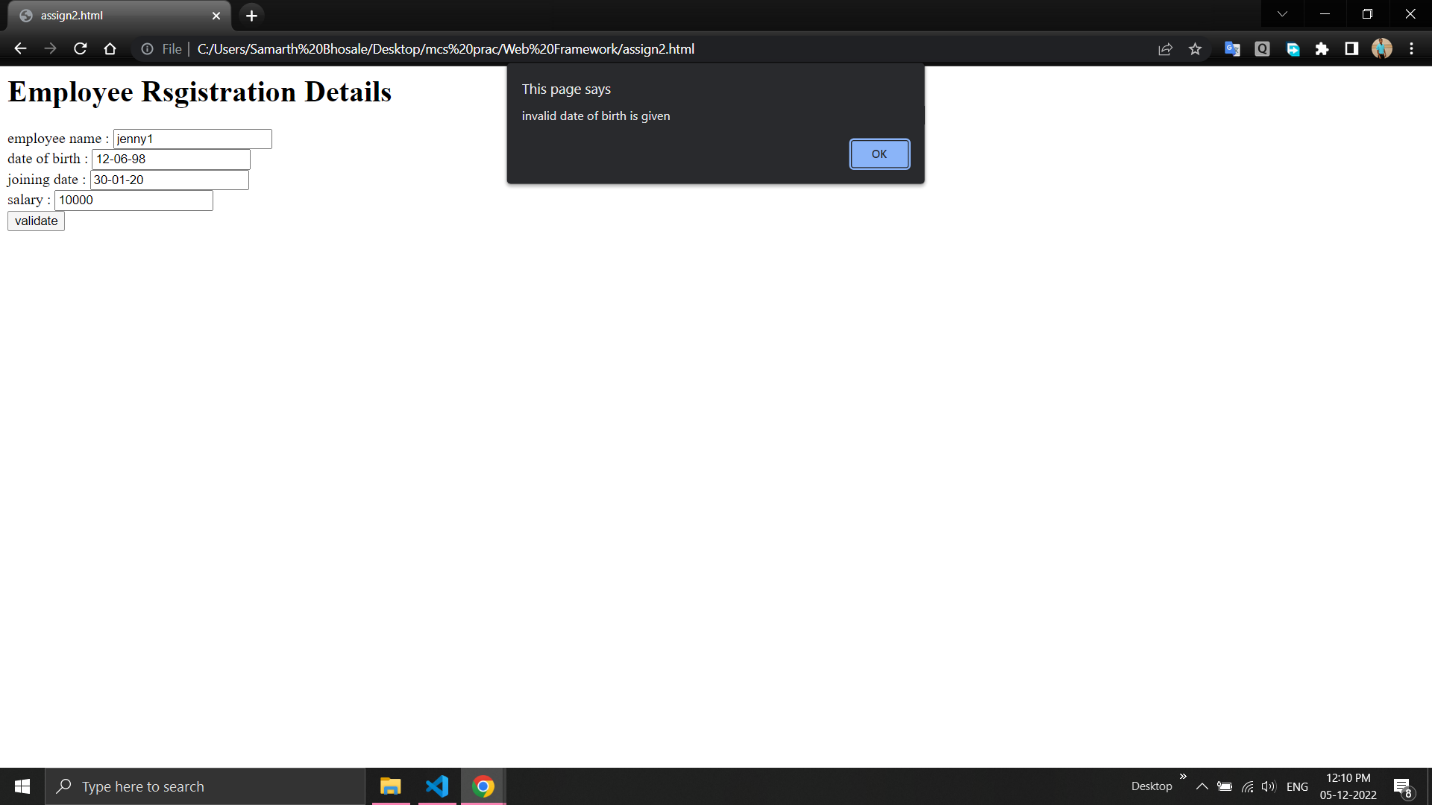
<input type="button" value="validate" onclick="validate()">

</form>

</body>

</html>

**//Output:** 



**3. Create an HTML form for Login and write a JavaScript to validate email ID using Regular Expression.**

<html>

<head>

<script>

function validateform(){

var email = document.getElementById("email").value;

var password = document.getElementById("psw").value;

if (!(/^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/.test(email)))

alert("You have entered an invalid email address!");

}

</script>

</head>

<body>

<form name="myform" onsubmit="return validateform()">

<h1>Please fill in this form to Login.</h1>

Email :

<input type="text" autocomplete="off" placeholder="Enter Email" name="email" id="email" required><br>

Password :

<input type="password" autocomplete="off" placeholder="Enter Password" name="psw" id="psw" required><br>

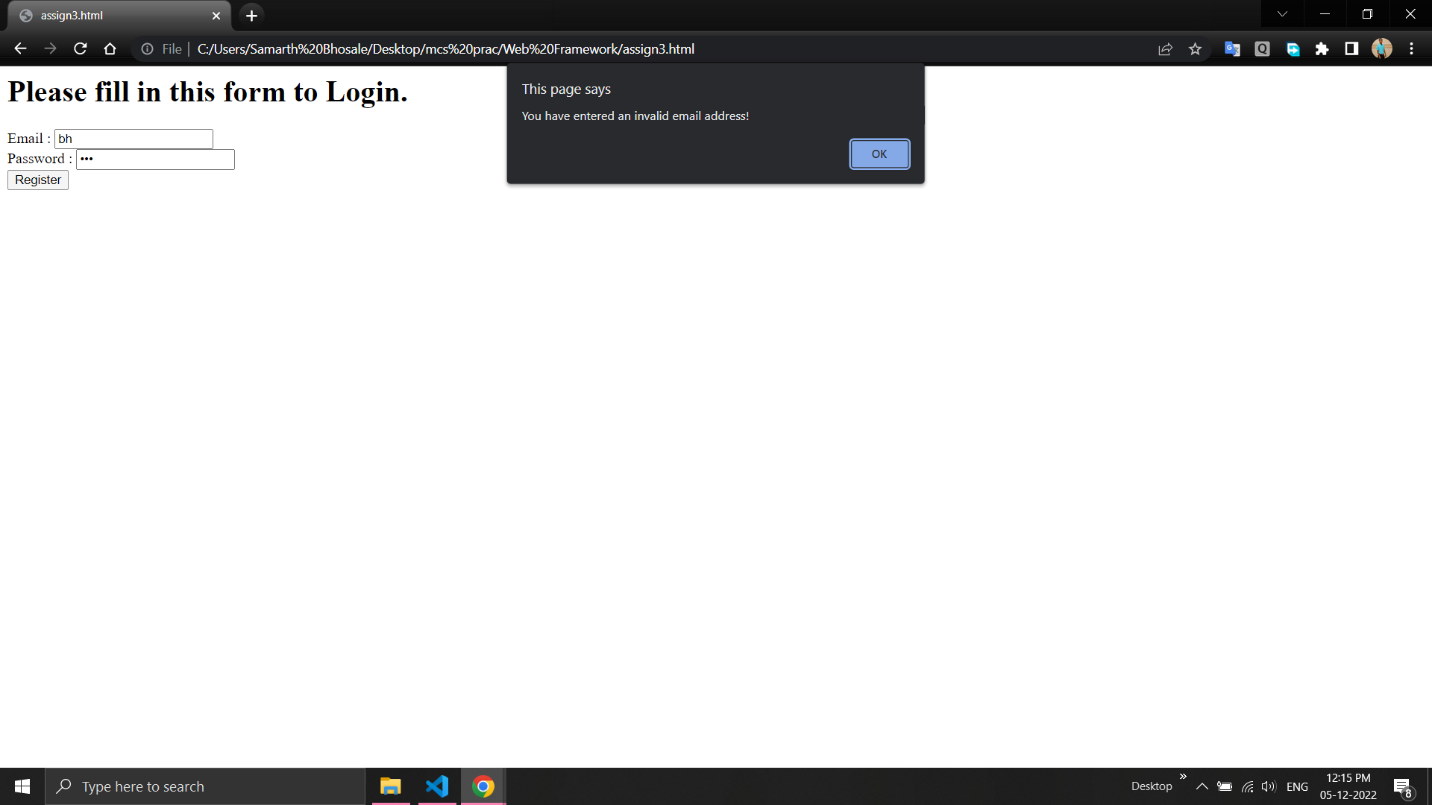
<button type="submit" class="registerbtn">Register</button>

</form>

</body>

</html>

**//Output:**



**4. Create a Node.js file that will convert the output "Hello World!" Into upper-case letters.**

var http = require('http');

var uc = require('upper-case');

http.createServer(function (req, res){

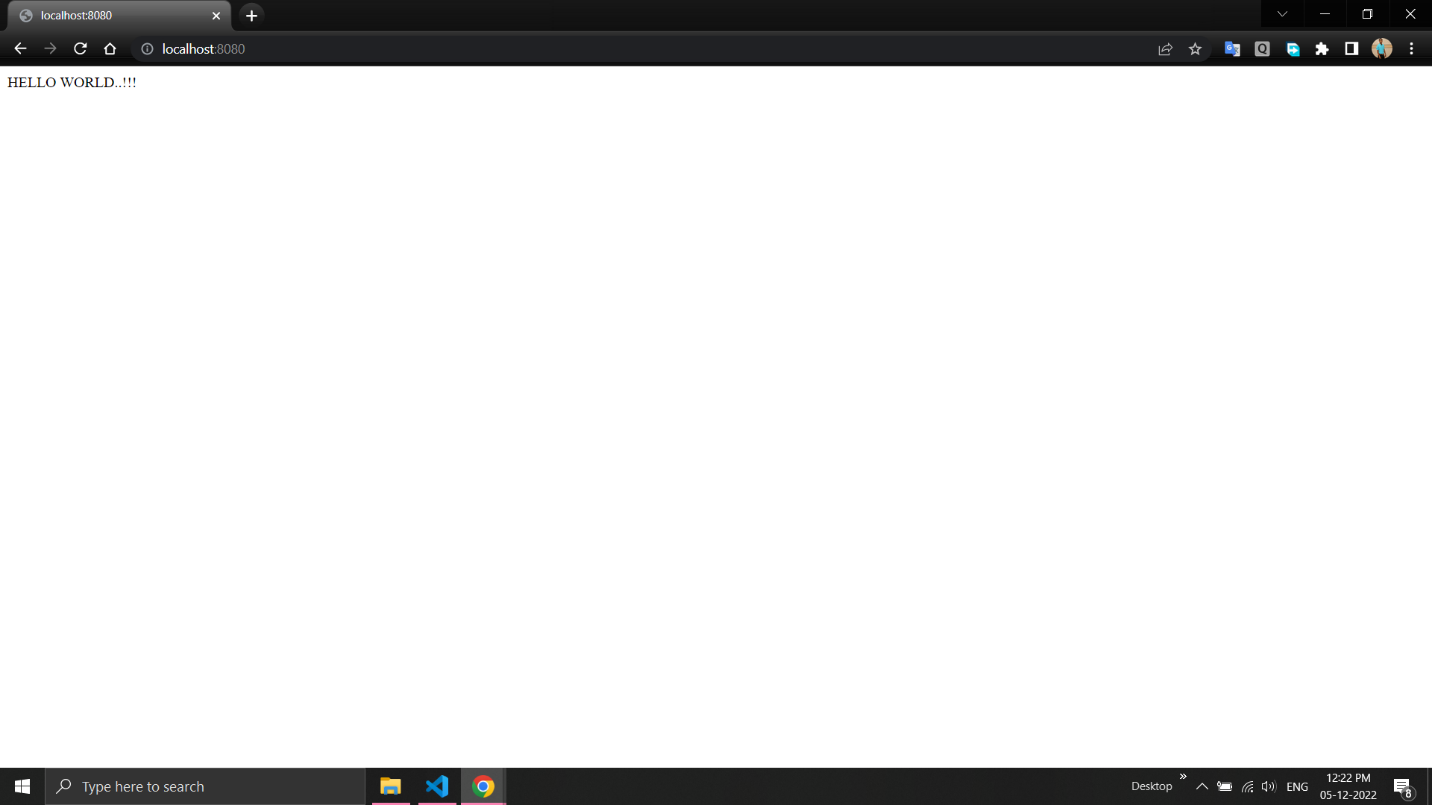
res.writeHead(200, {'Content-Type': 'text/html'});

res.write(uc.upperCase("hello world..!!!"));

res.end();

}).listen(8080);

**//Output:**



**5. Using Node.js create a web page to read two file names from user and append contents of first file into second file.**

var fs = require('fs');

console.log("\nFile Contents of file before append:");

var a = fs.readFileSync("file1.txt", "utf8");

fs.appendFile("file2.txt", a, function(err){

if (err)

console.log(err);

else {

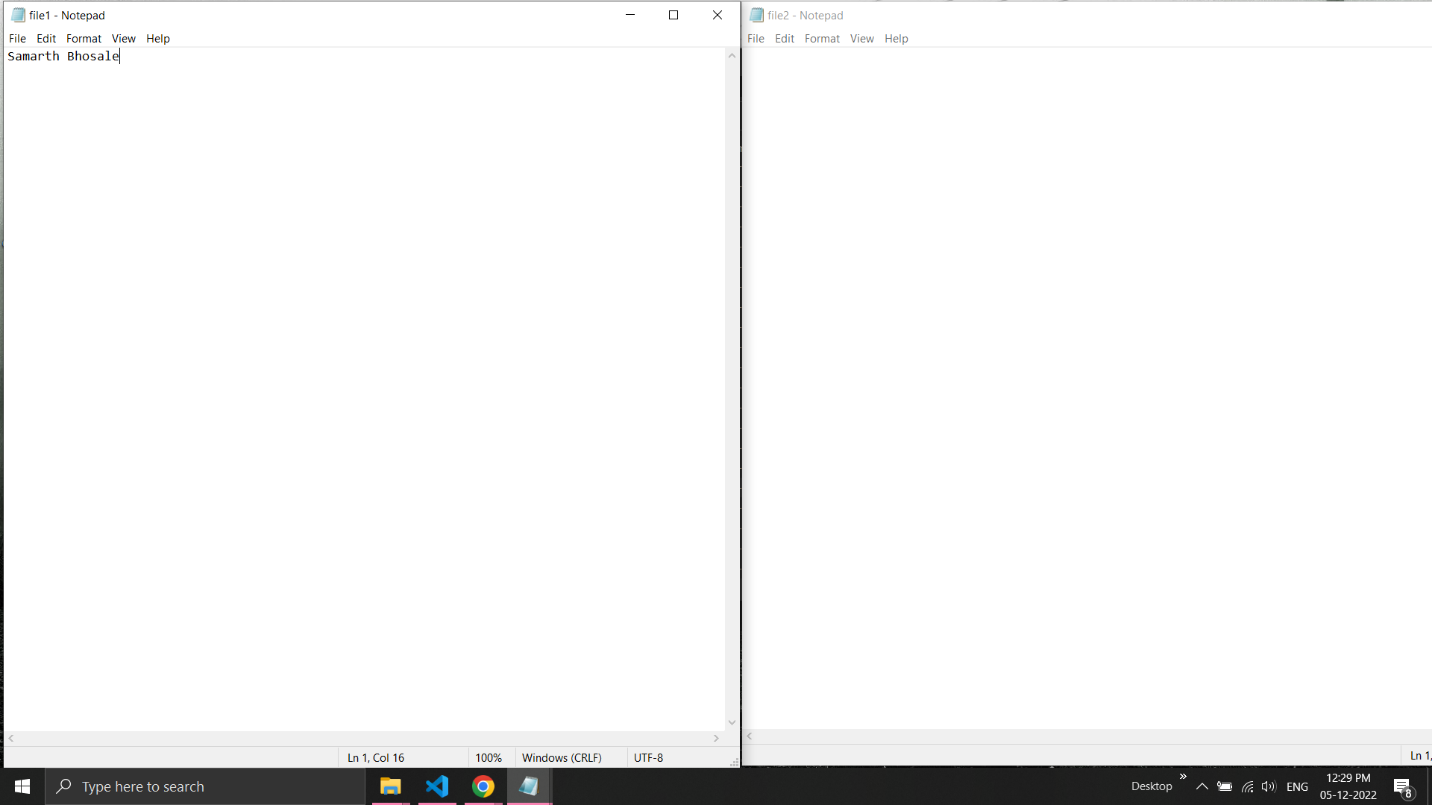
console.log("\nFile Contents of file after append:",

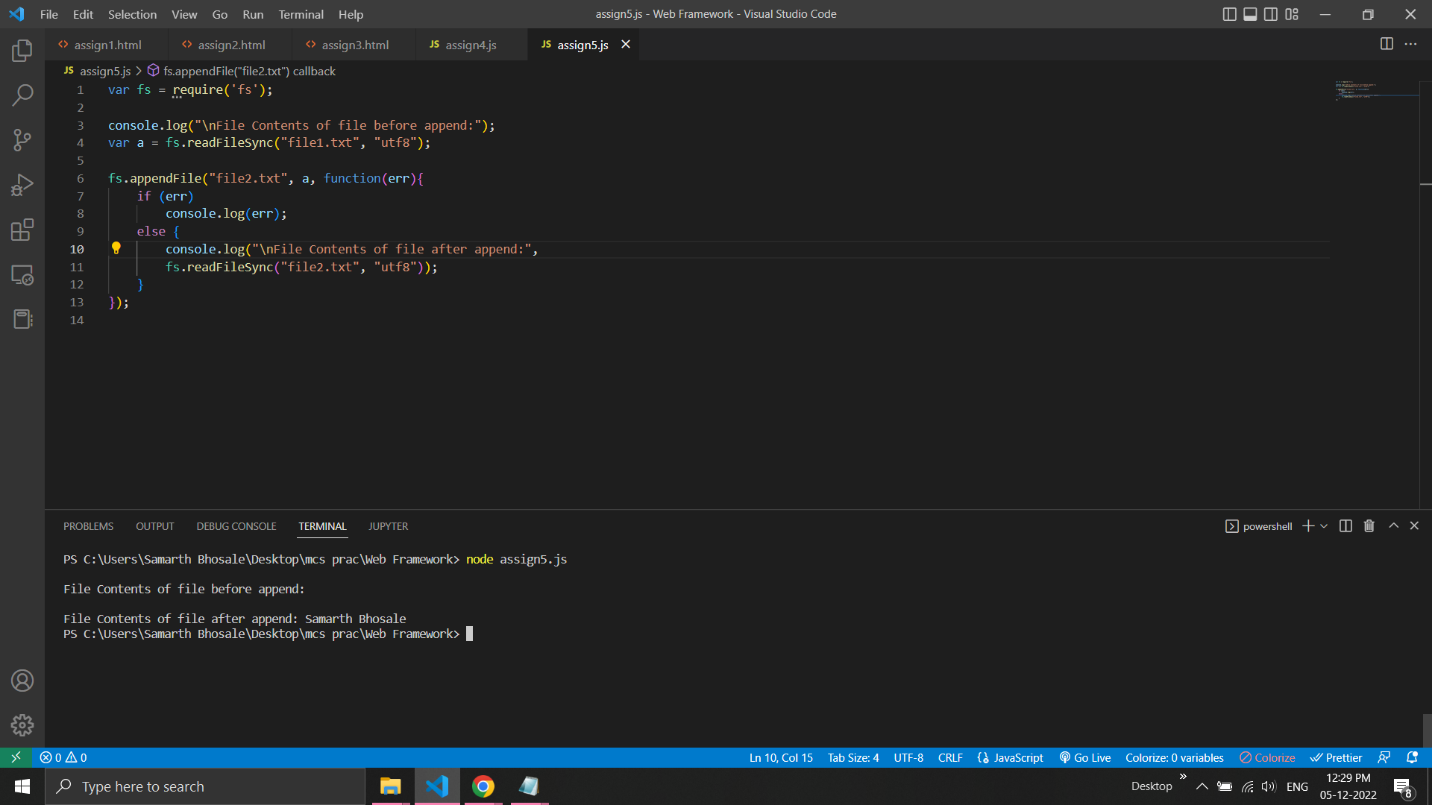
fs.readFileSync("file2.txt", "utf8"));

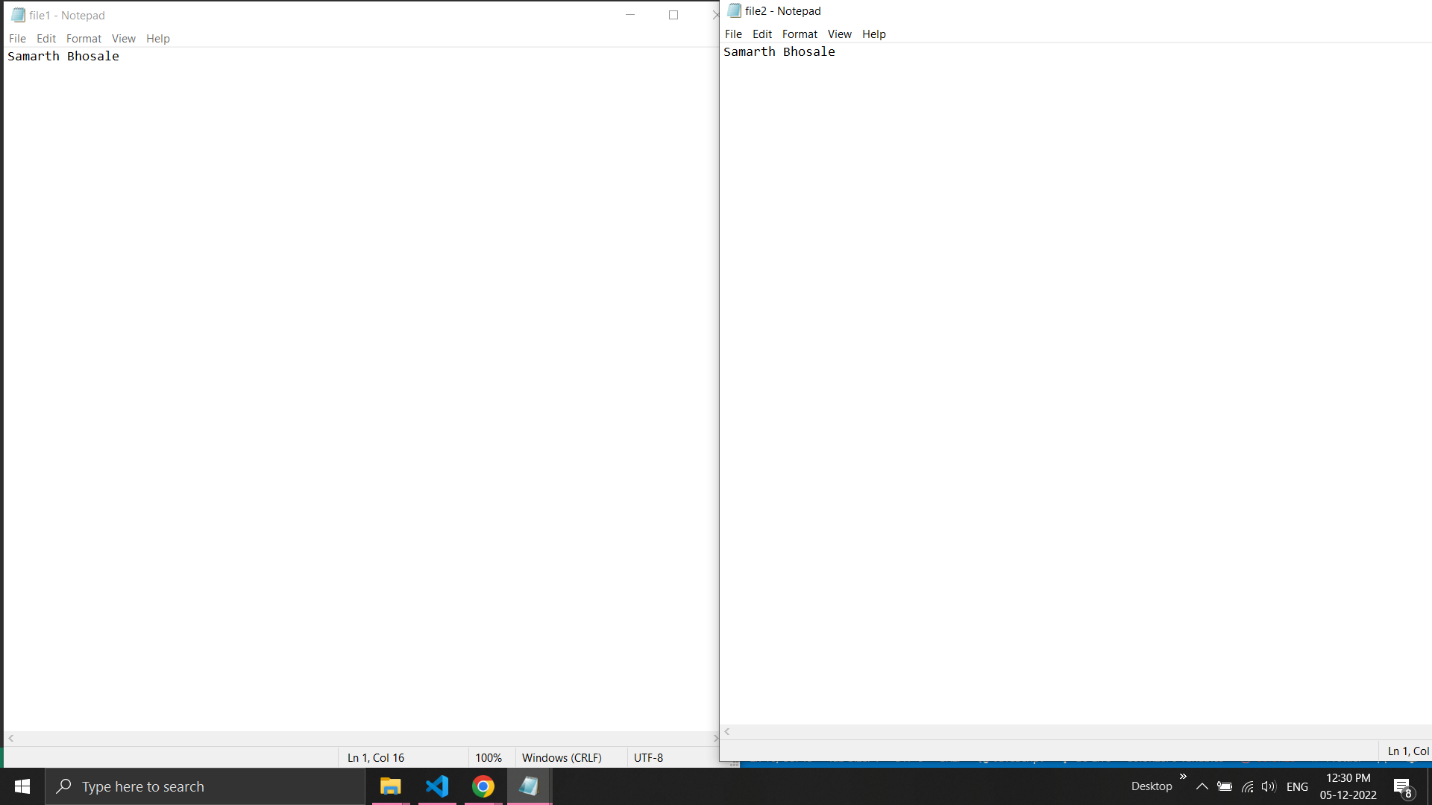
}

});

**//Output:**







**6. Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error.**

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'});

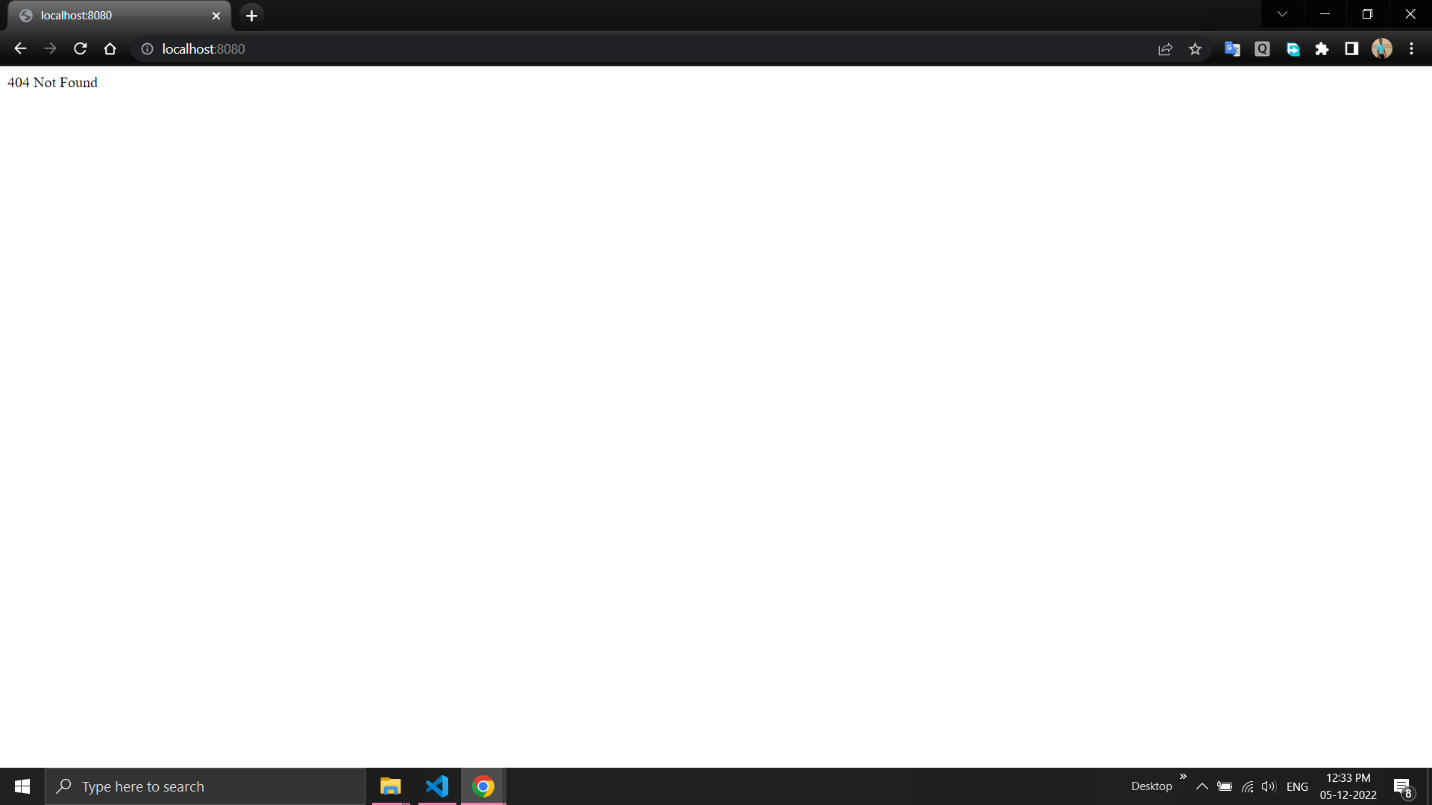
res.write(data);

return res.end();

});

}).listen(8080);

**//Output:**



**7. Create a Node.js file that writes an HTML form, with an upload field.**

var http = require('http');

http.createServer(function (req, res) {

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>');

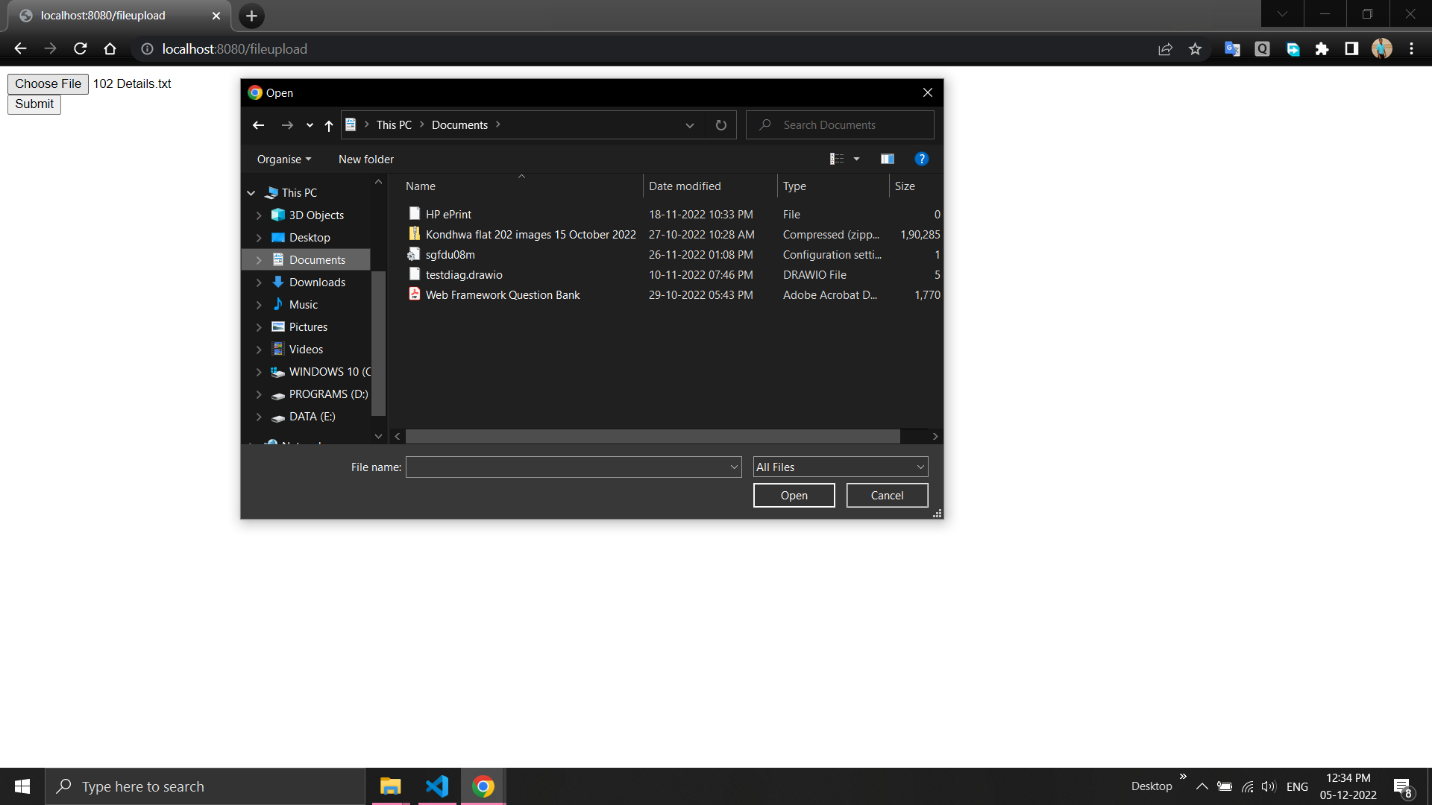
res.write('<input type="submit">');

res.write('</form>');

return res.end();

}).listen(8080);

**//Output:**



**8. 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: "",

database: "sam"

});

con.connect(function(err) {

if (err) throw err;

else

console.log("Connected!");

con.query("create database sam13", function (err, result) {

if (err)

console.log(err);

console.log("Database Created");

});

});

con.query("use sam13",function(err){

if(err)

console.log(err);

else

console.log("Database { "+"sam13 } selected!");

});

var sql = "create table customer1(name varchar(10), address varchar(10))";

con.query(sql, function (err, result) {

if (err)

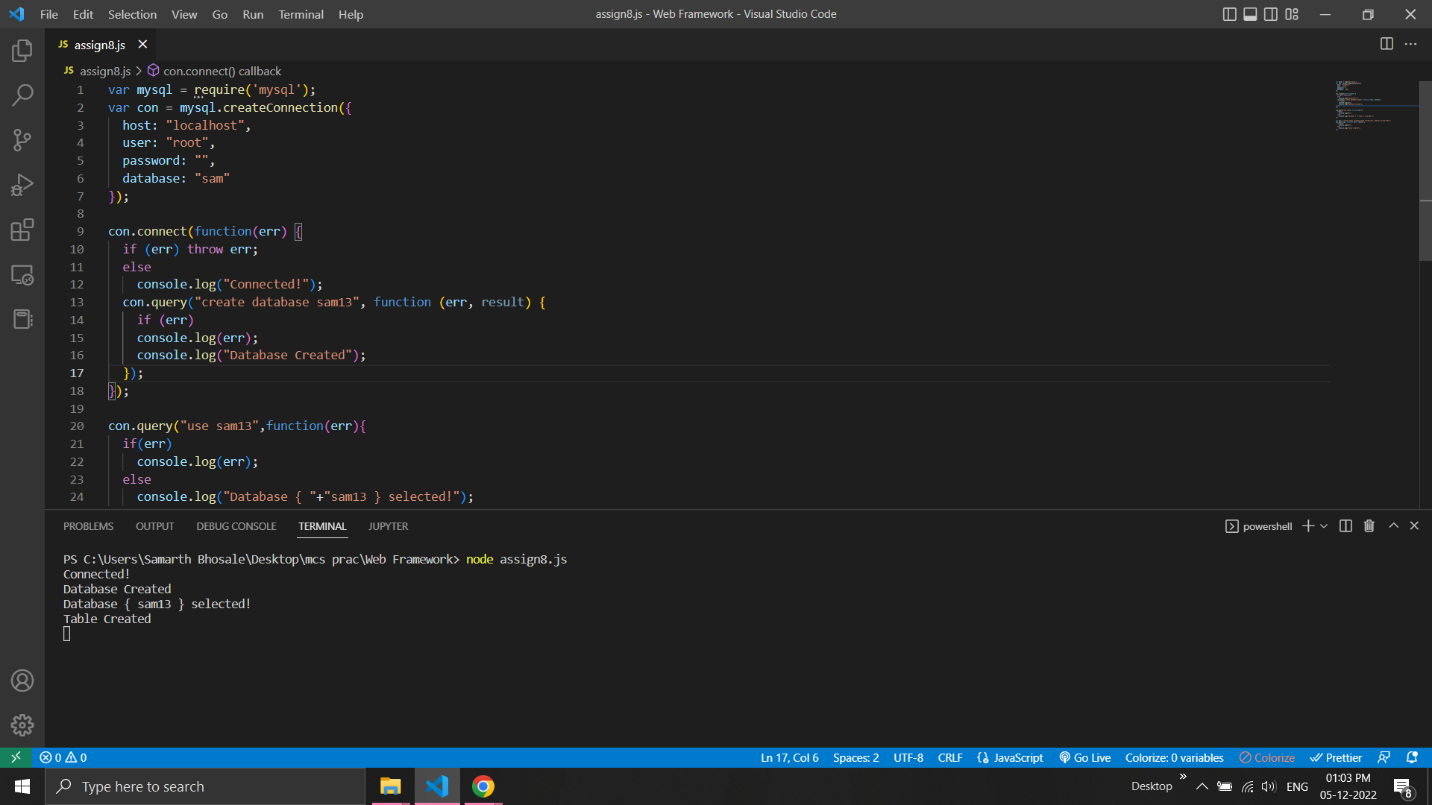
console.log(err);

else

console.log("Table Created");

});

**//Output:**



**9. 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:'sam'

});

con.connect(function(err) {

if (err) throw err;

else

console.log("Connected!");

});

con.query('select \* from customer1', function(err,rows){

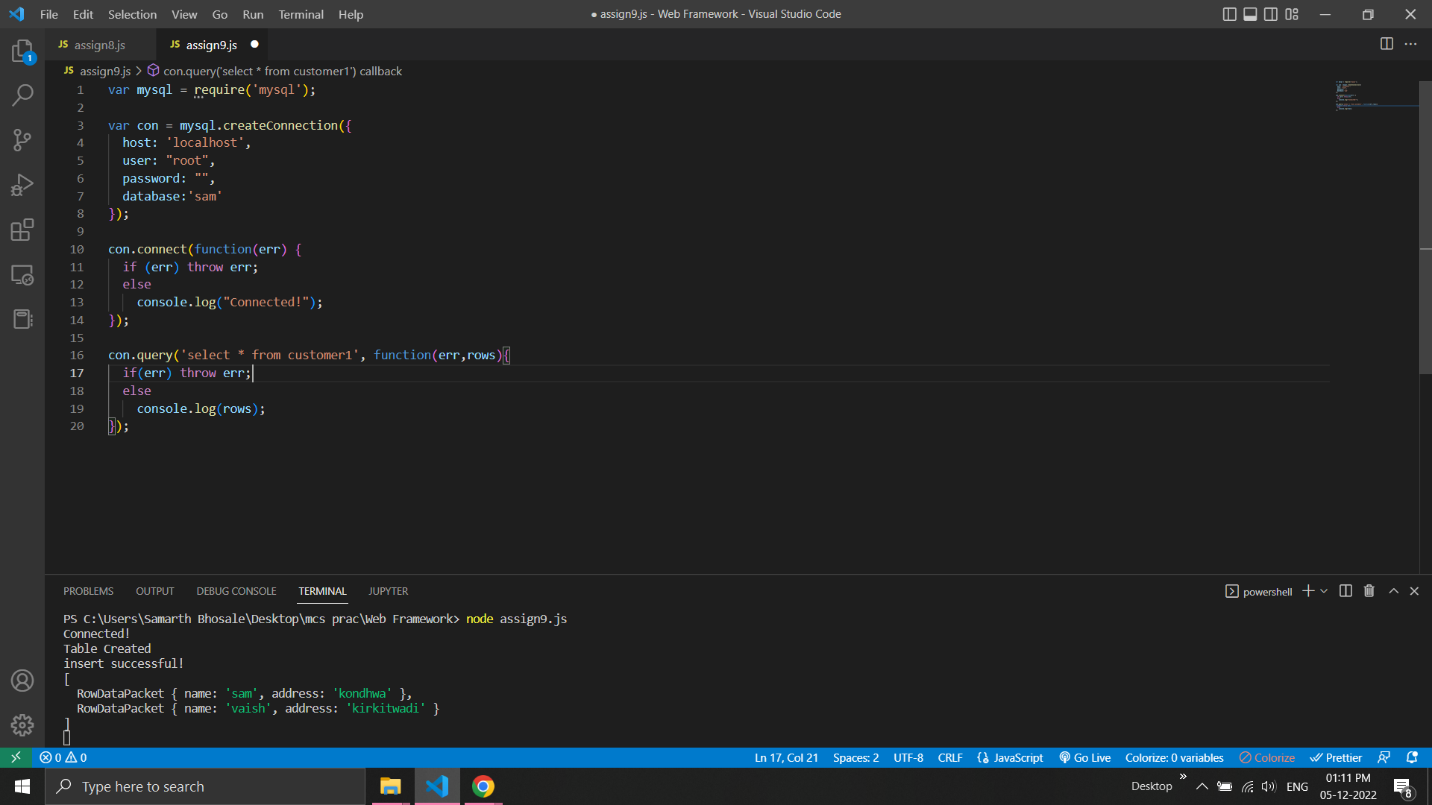
if(err) throw err;

else

console.log(rows);

});

**//Output:**



**10. 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: "sam"

});

con.connect(function(err) {

if (err) throw err;

else{

console.log("Connected!");

con.query("create table students1(roll int primary key, name text, address text)",function(err){

if(err) throw err;

else{

con.query("insert into students1 values(111,'sam','kondhwa'),(222,'vaish','kirkitwadi'),(333,'saarth','wanawadi')",function(err){

if(err) throw err;

else{

console.log("data inserted!");

con.query("select \* from students", function (err, result) {

if (err) throw err;

else

console.log(result);

});

}

});

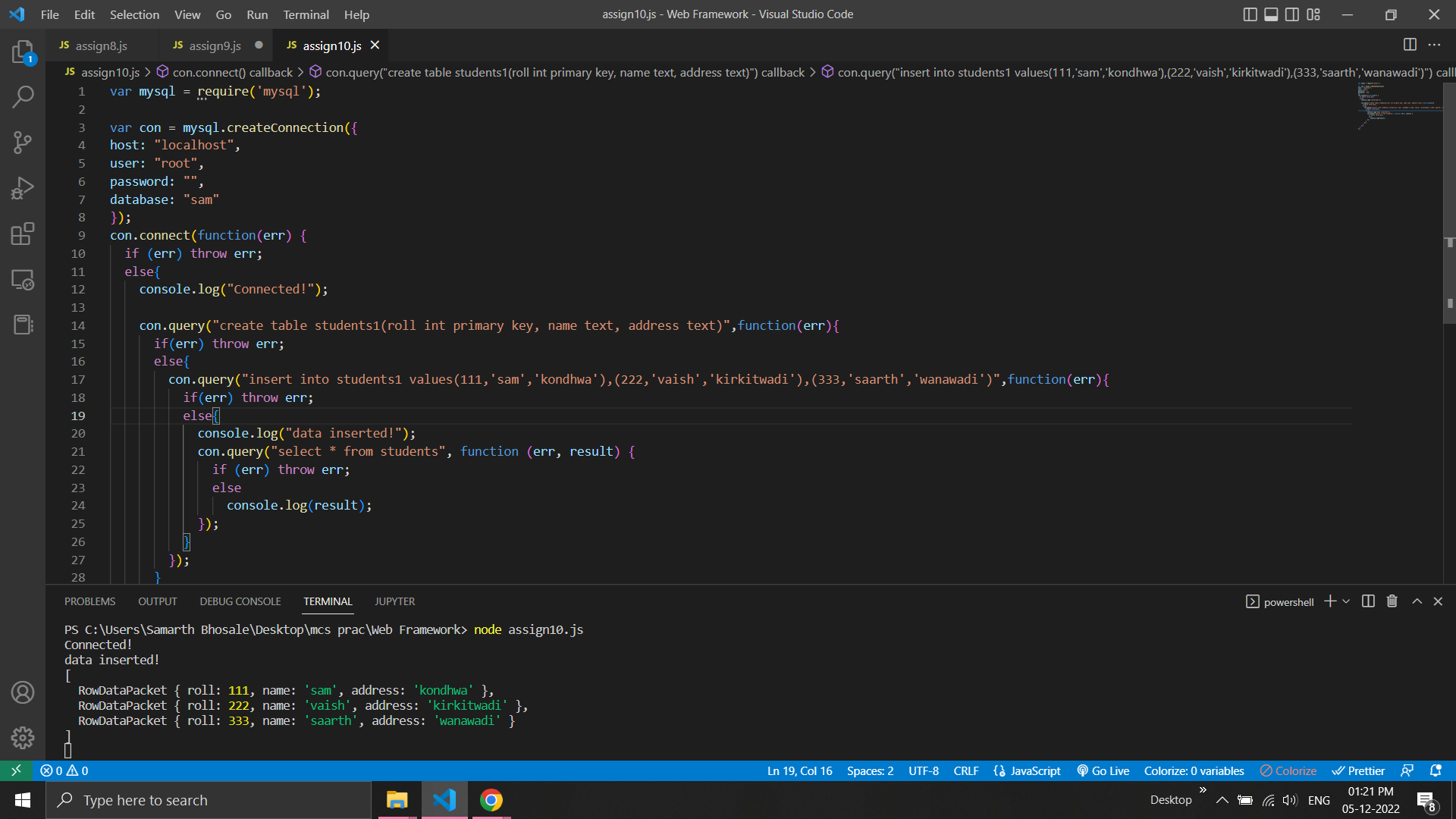
}

});

}

});

**//Output:**

****

**11. 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: "sam"

});

con.connect(function(err) {

if (err) throw err;

else{

console.log("Connected!");

con.query("select \* from customer1", function (err, result) {

if (err) throw err;

else{

console.log(result);

con.query("delete from customer1 where name = 'riya'", function (err, result){

if (err) throw err;

else{

console.log("Deleted Record : " + result.affectedRows);

con.query("select \* from students1", function (err, result) {

if (err) throw err;

else

console.log(result);

});

}

});

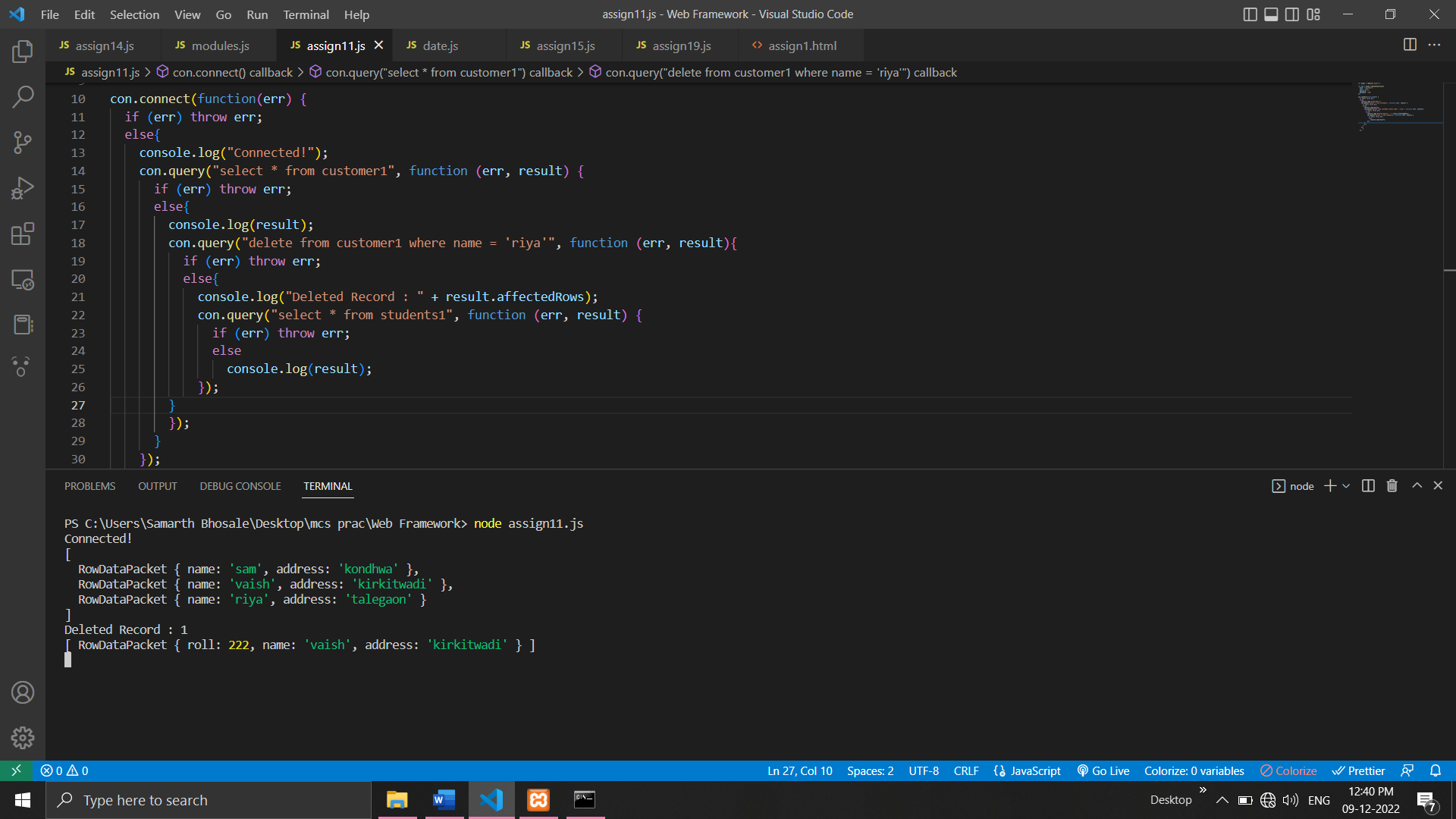
}

});

}

});

**//Output:**



**12. Create a Simple Web Server using node js.**

var http = require('http');

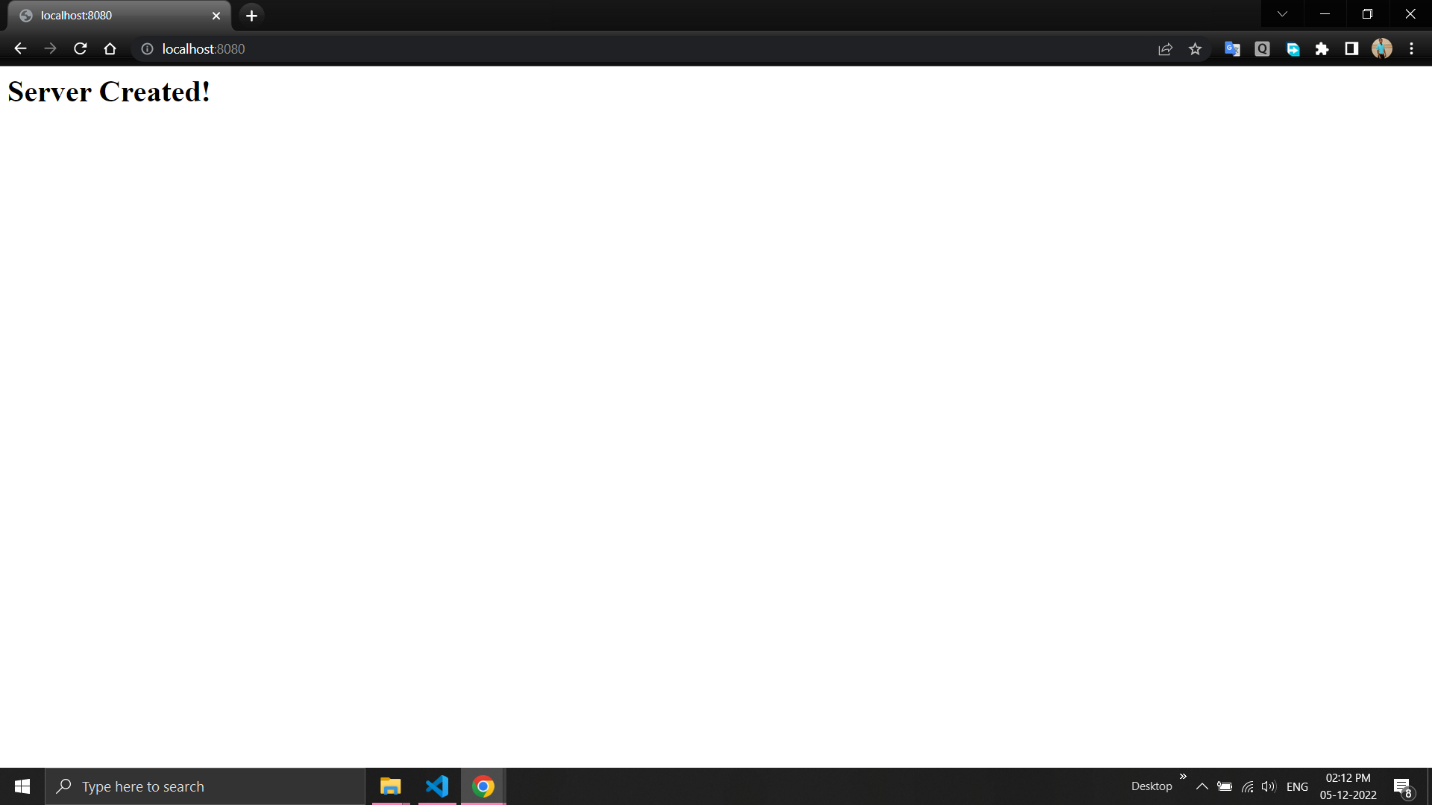
var server = http.createServer(function (req, res) {

res.write("<h1>Server Created!</h1>");

});

server.listen(8080);

**//Output:**



**13. Using node js create a User Login System.**

**html-**

<html>

<head>

<title>User login page</title>

<head>

<script>

function f(){

var validRegex =/^[a-zA-Z0-9.!#$%&'+/=?^\_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)$/;

var email=document.getElementById("email").value;

if(!validRegex.test(email))

alert("Invalid Email!");

else

alert("submitted succesfully");

return false;

}

</script>

</head>

<body>

<form name="login" onsubmit="f()">

<h1>Login here</h1>

email id:<input type="text" id="email"/><br><br>

password:<input type="password" id="password"><br><br>

<input type="submit" id="submit" value="submit">

</form>

</body>

</html>

**javascript-**

var http=require('http');

http.createServer(function(req,resp){

var fs=require("fs");

resp.writeHead(200,{'content-type':'text/html'});

var content= fs.readFileSync("login.html");

if(content){

resp.write(content);

}

else{

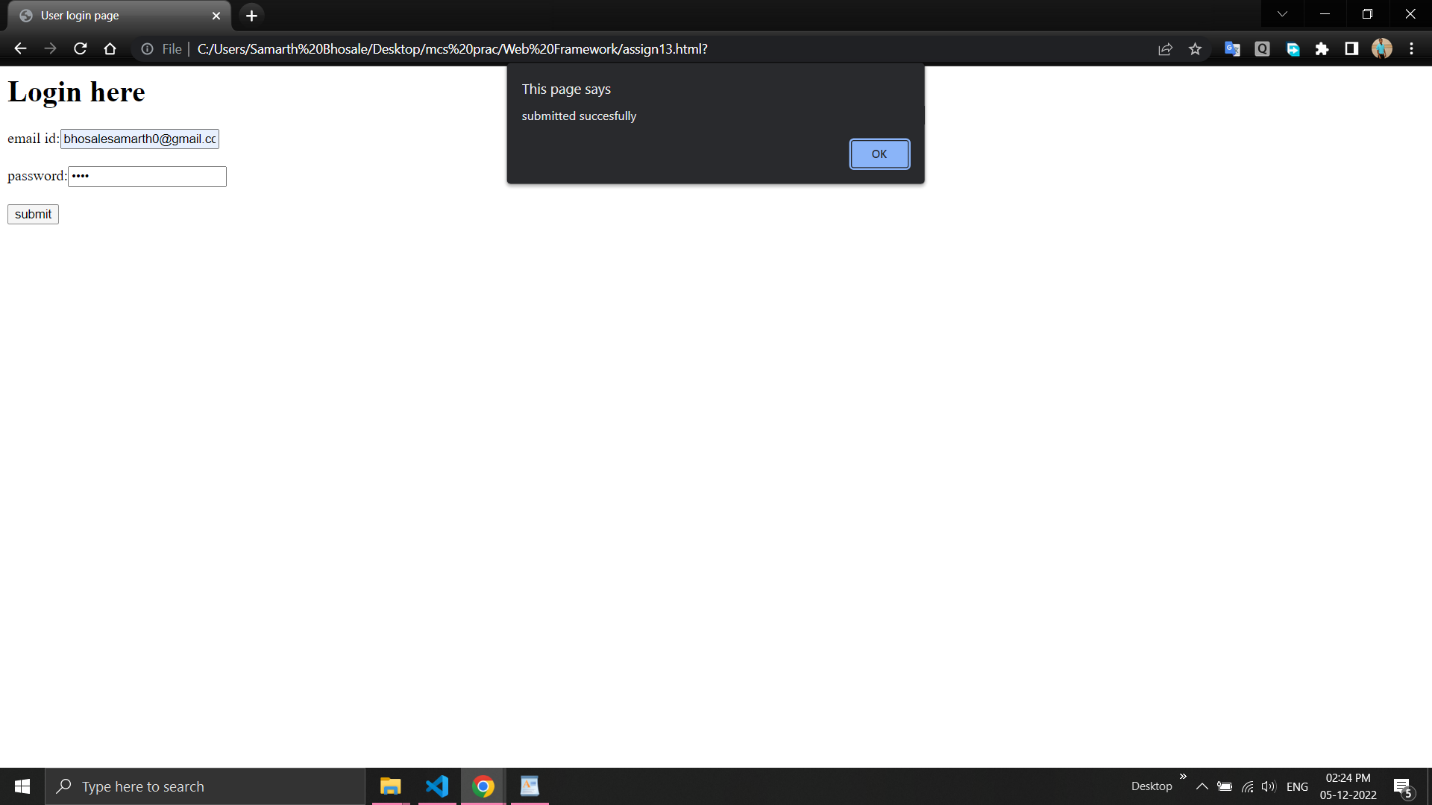
resp.write("404 error");

}

resp.end();

}).listen(6006);

**//Output:**



**14. Using node js create an eLearning System.**

**html –**

<html>

<head>

</head>

<body>

<center>

<h1>E-Learning</h1>

<h3>

<a href = "home.html">Home</a>

<a href = "team.html">Team</a>

<a href = "about.html">About</a>

<a href = "contact.html">Contact</a>

</h3>

</center>

</body>

</html>

**javascript –**

var fs=require("fs");

var http = require('http');

http.createServer(function(req,resp){

resp.writeHead(200,{"content-type":"text/html"});

var content=fs.readFileSync("elearningnew.html");

if(content)

resp.write(content);

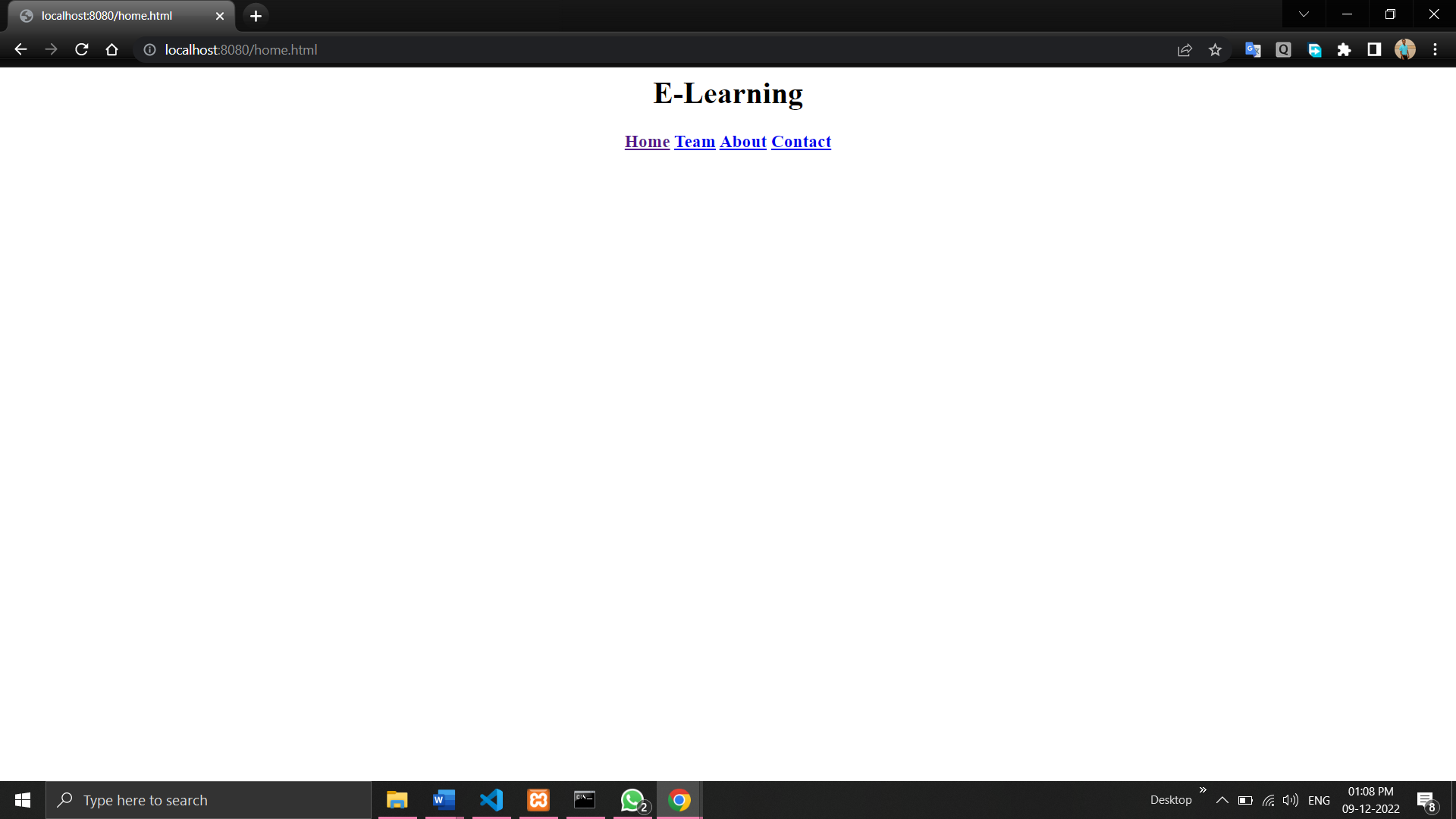
else

resp.write("404 error");

resp.end()

}).listen(8080);

**//Output:**



**15. Using node js create a Recipe Book.**

**html –**

<html>

<head>

<title>Recipe Book</title>

</head>

<body>

<center>

<h1>Recipe Book - Pasta</h1><br>

</center>

<h3>Ingredients</h3>

<p>Serves 2 people</p>

<ul>

<li>Veggies</li>

<li>Raw Pasta</li>

<li>Sauces</li>

<li>Oil</li>

<li>Herbs</li>

</ul>

<br>

<h3>Preparation</h3>

<ul>

<li>Take 2 cups water.</li>

<li>Boil the pasta half.</li>

<li>Fry the veggies.</li>

<li>Pasta is ready to serve.</li>

</ul>

</body>

</html>

**Javascript –**

var fs=require("fs");

var http = require('http');

http.createServer(function(req,resp){

resp.writeHead(200,{"content-type":"text/html"});

var content=fs.readFileSync("recipenew.html");

if(content)

resp.write(content);

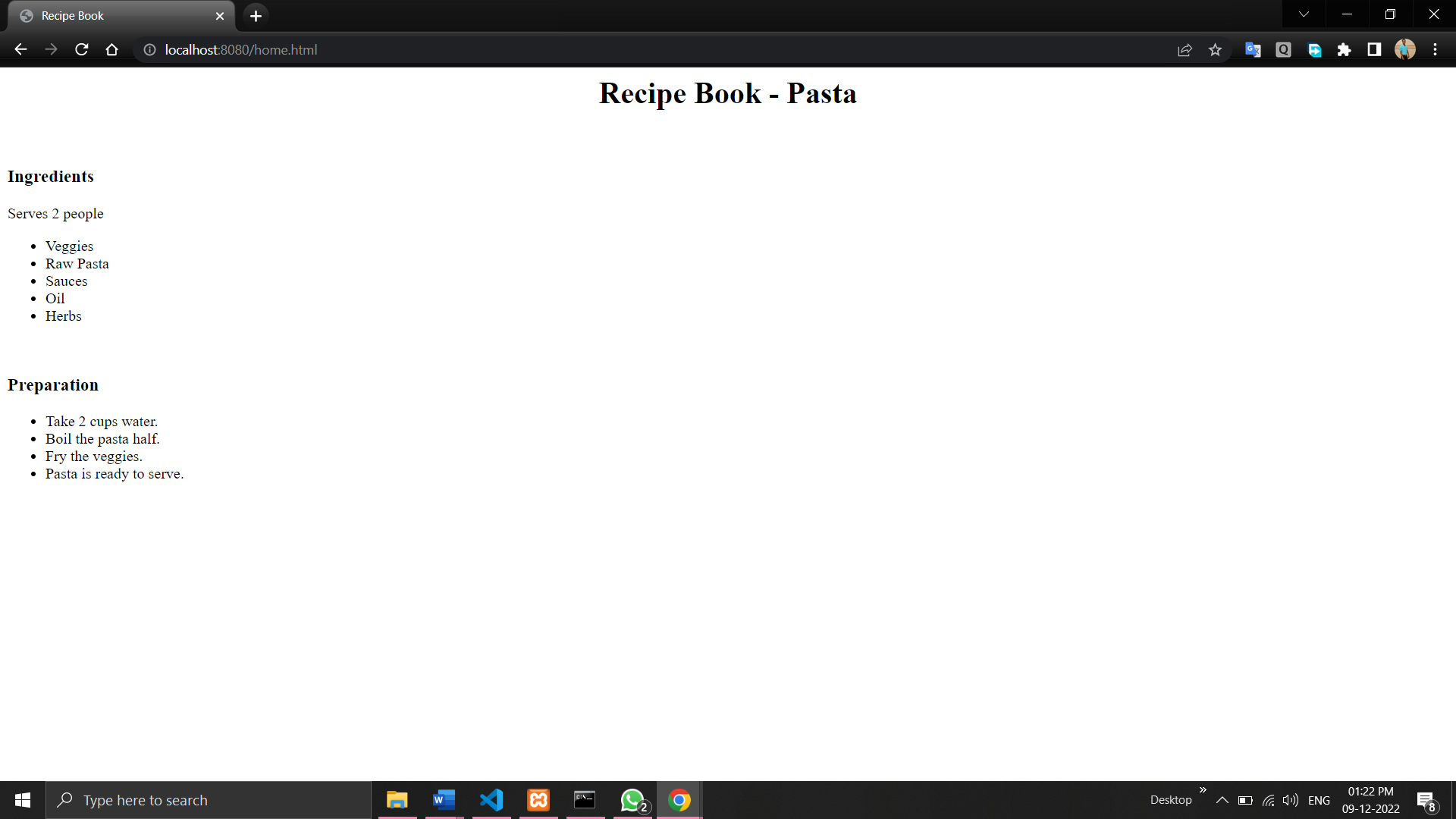
else

resp.write("404 error");

resp.end()

}).listen(8080);

**//Output:**



**16. Write node js script to interact with the filesystem, and serve a web page from a file.**

var express = require('express');

var app = express();

var PORT = 3000;

app.get('/', function(req, res){

    res.download('hello.txt');

});

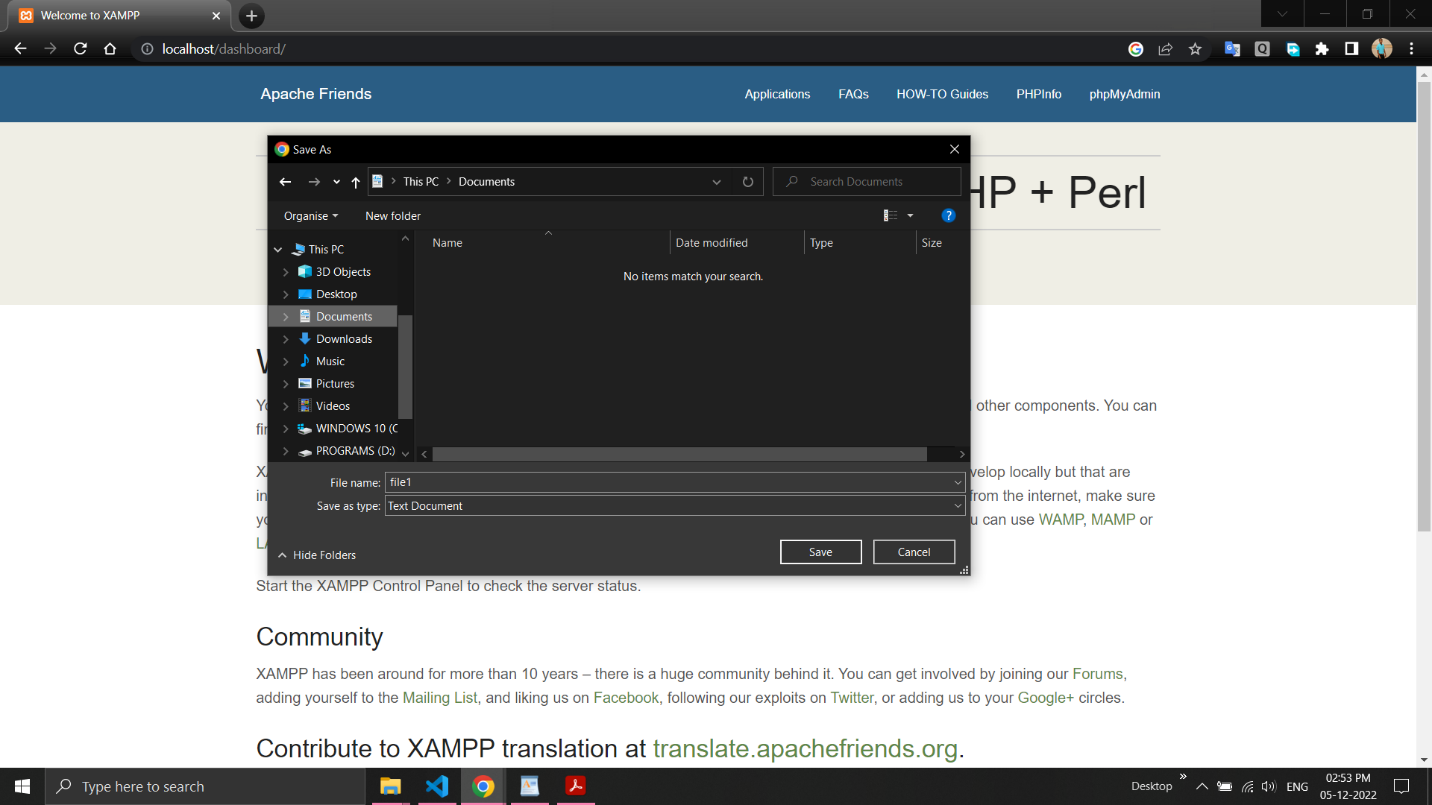
app.listen(PORT, function(err){

    if (err) console.log(err);

    console.log("Server listening on PORT", PORT);

});

**//Output:**



**17. 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.**

**Modules.js –**

module.exports.dt= new Date();

**date.js -**

var http = require('http');

var dt = require('./Modules');

http.createServer(function (req, res) {

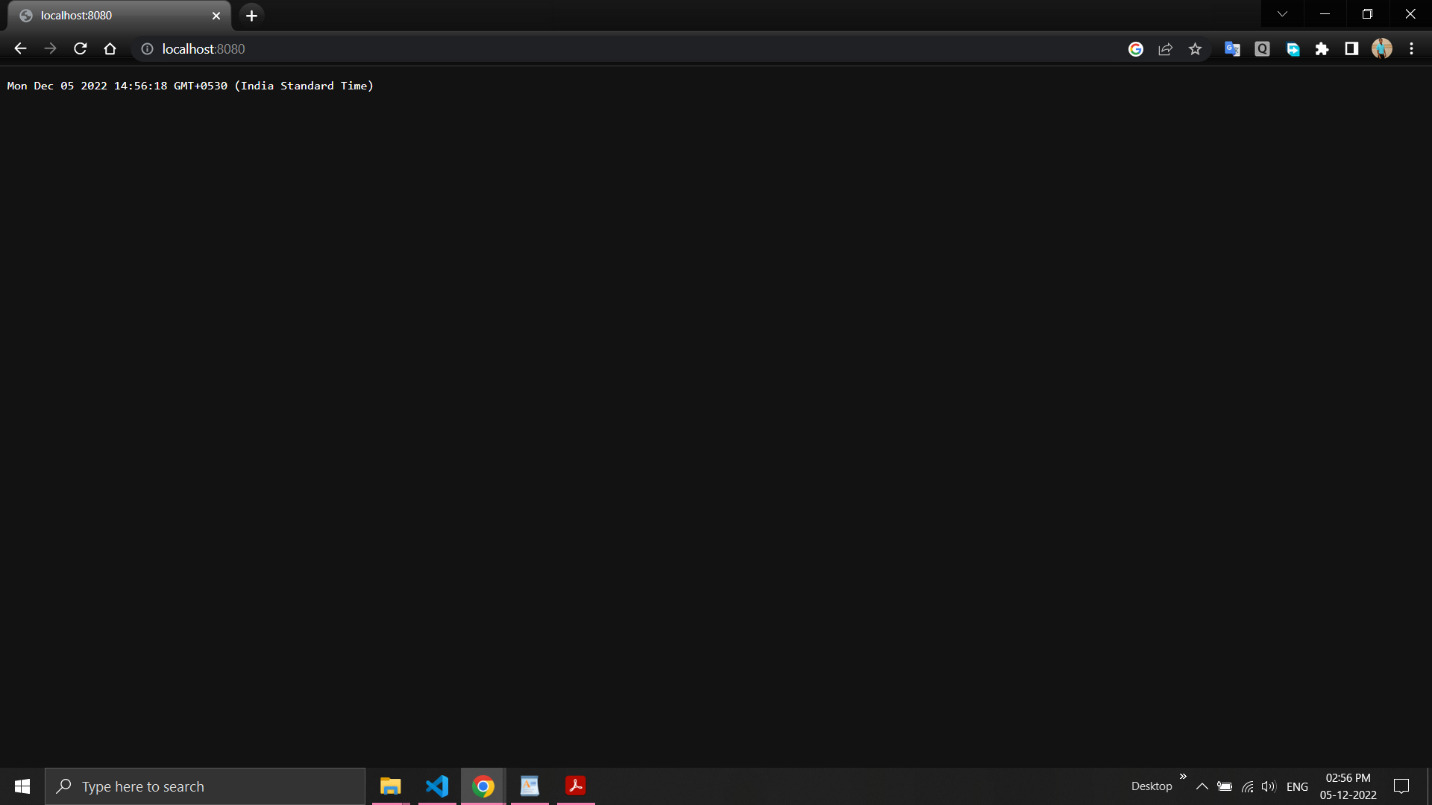
  res.writeHead(200, {'Content-Type': 'text/html'});

  res.write("The date and time are currently: " + dt.myDateTime());

  res.end();

}).listen(8080);

**//Output:**



**18. 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.**

var events = require('events');

var myeventEmitter = new events.EventEmitter();

myeventEmitter.on('myevent', function Listener1(){

     console.log('listener1 executed!');

});

myeventEmitter.on('myevent' ,function Listener2(a,b){

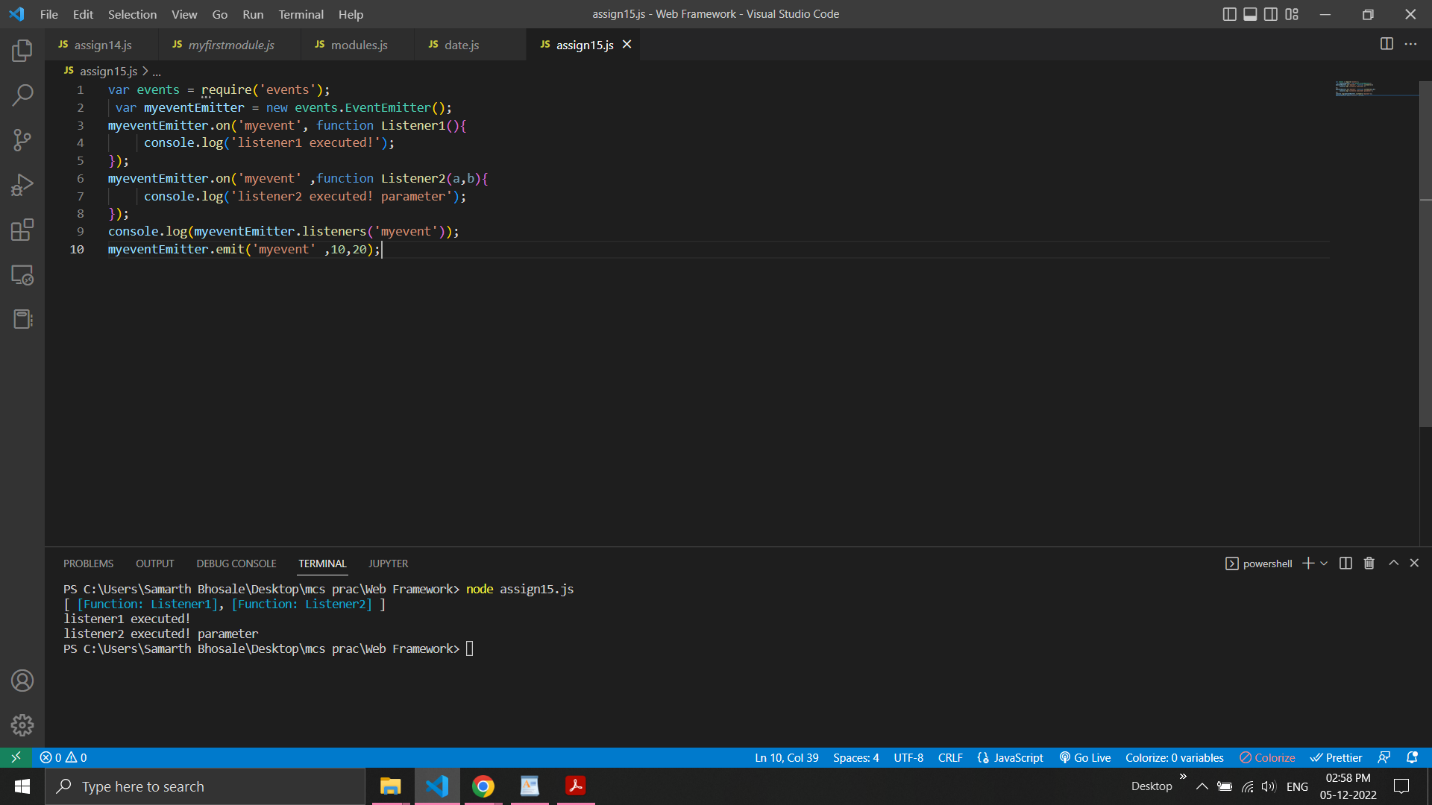
     console.log('listener2 executed! parameter');

});

console.log(myeventEmitter.listeners('myevent'));

myeventEmitter.emit('myevent' ,10,20);

**//Output:**



**19. 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();

app.get('/a/',function(req,res){

    res.download('file2.txt');

});

app.listen(8051,function(err){

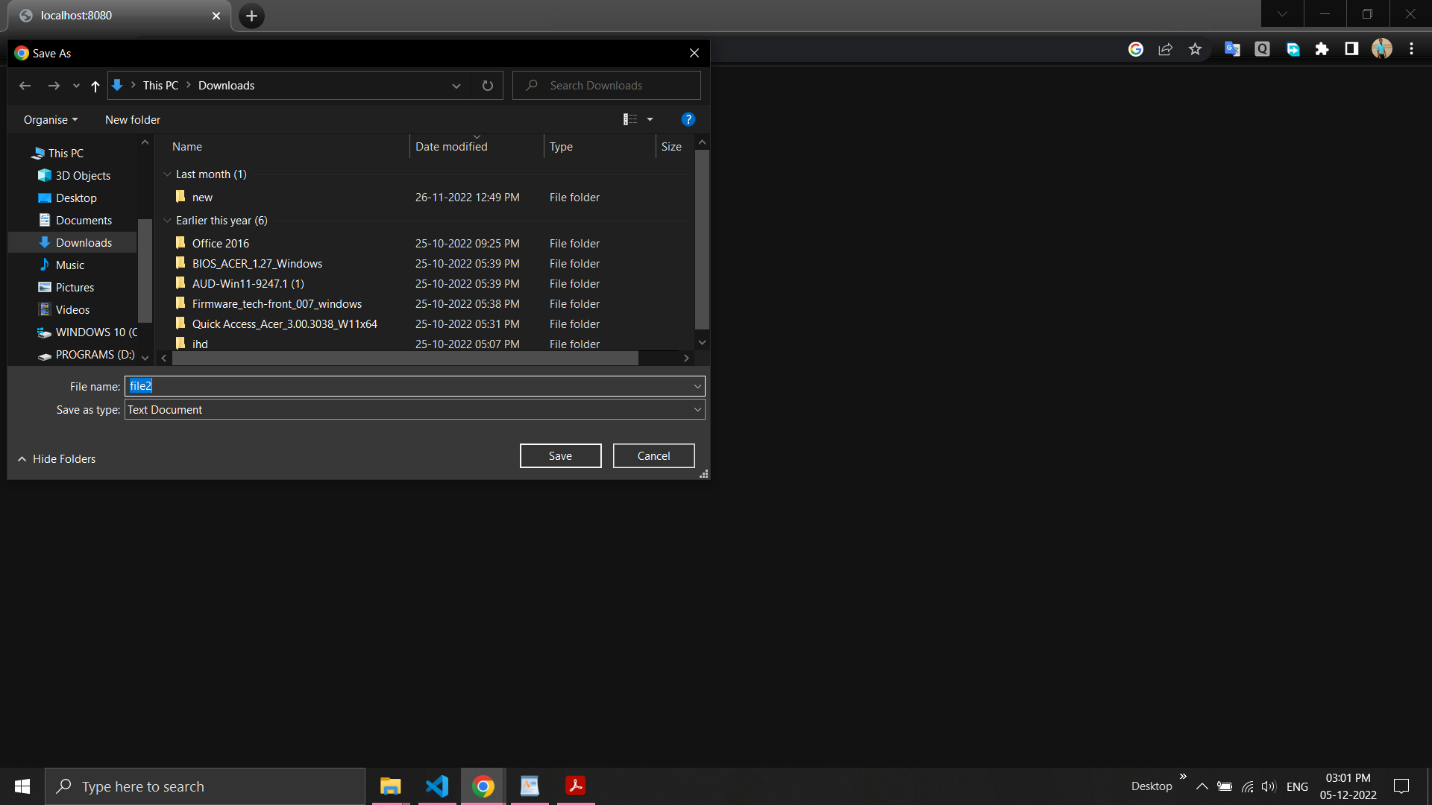
    if(err)

    console.log(err);

    console.log("Server is listening..");

});

**//Output:**



**20.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.**

**Views.py -**

# -\*- coding: utf-8 -\*-

from \_\_future\_\_ import unicode\_literals

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def index(req):

return render(req,"m.html")

**urls.py -**

"""myproj URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/1.11/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: url(r'^$', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: url(r'^$', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.conf.urls import url, include

2. Add a URL to urlpatterns: url(r'^blog/', include('blog.urls'))

"""

from django.conf.urls import url

from django.contrib import admin

from myapp1.views import index

urlpatterns = [

url(r'^admin/', admin.site.urls),

url("/",index),

]

**m.html -**

<html>

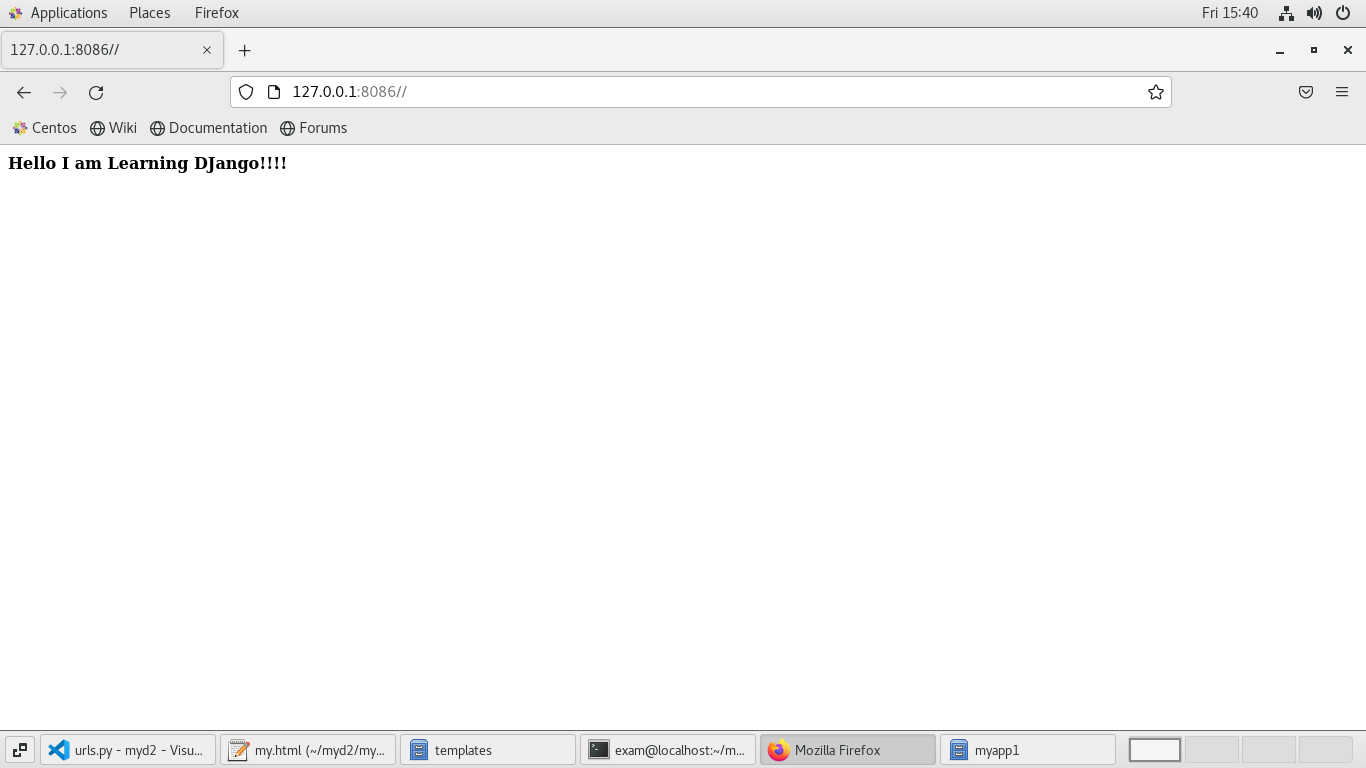
<body>

<b> Hello I am Learning Django</b>

</body>

</html>

**//Output –**

****

**21.Design a Django application that adds web pages with views and templates.**

**views.py -**

from django.shortcuts import render

from django.http import HttpResponse

def f1(req):

return render(req,"a.html")

def f2(req):

return render(req,"b.html")

def f3(req):

return render(req,"c.html")

**urls.py -**

from django.contrib import admin

from django.urls import path

from views.views import f1

from views.views import f2

from views.views import f3

urlpatterns = [

path('admin/', admin.site.urls),

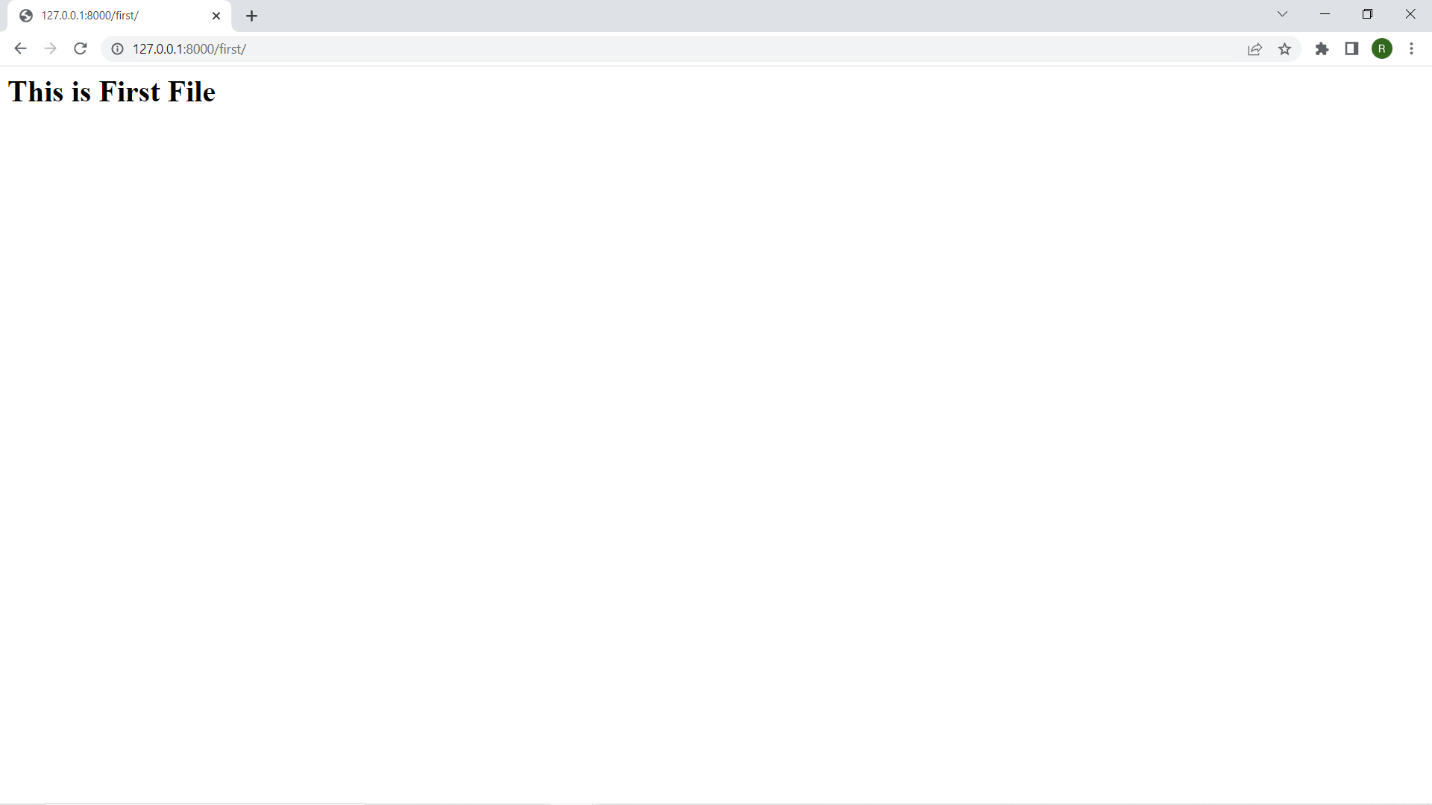
path('first/',f1),

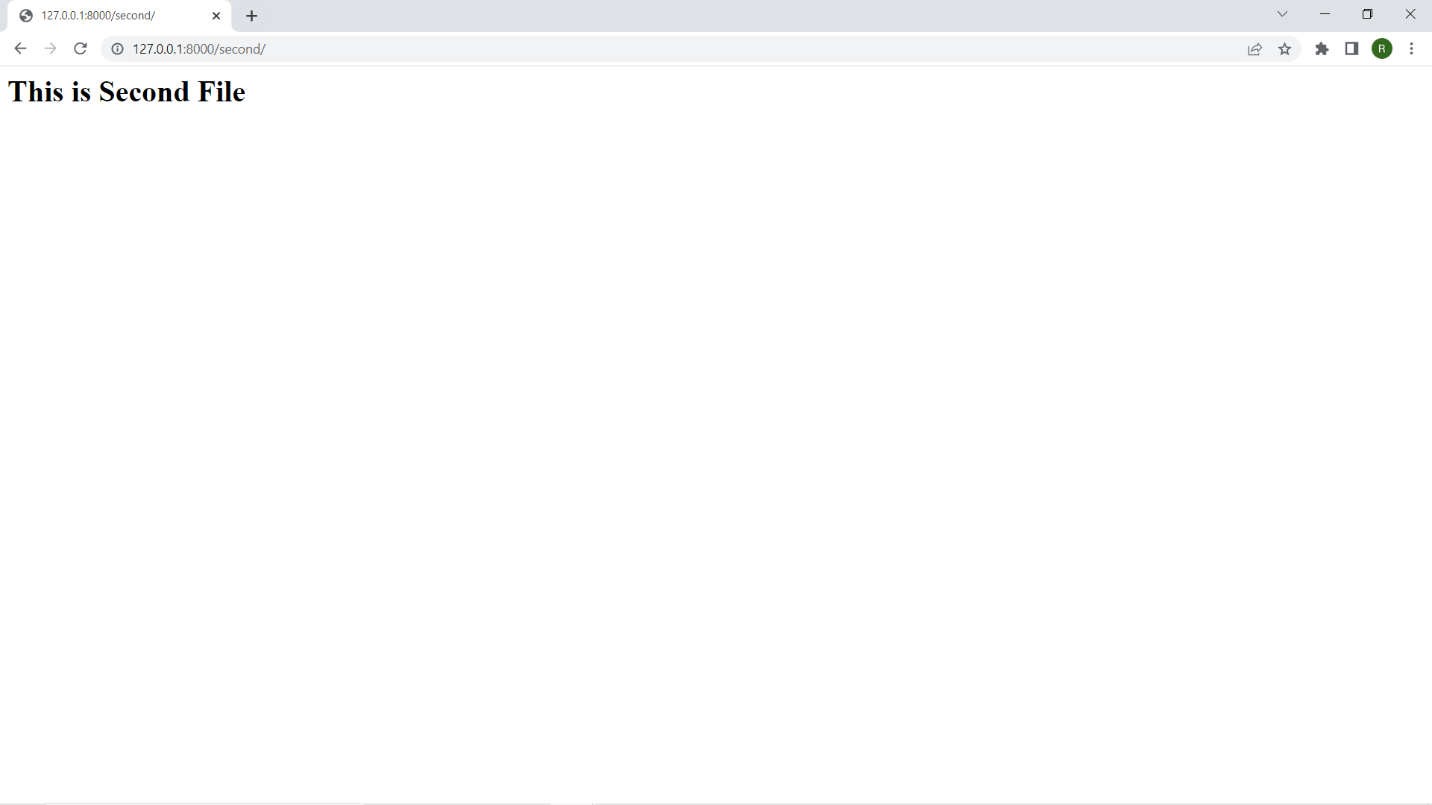
path('second/',f2),

path('third/',f3),

]

**//Output-**



****

**22.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.**

**models.py -**

from django.db import models

class Question(models.Model):

question\_text = models.CharField(max\_length = 200)

pub\_date = models.DateTimeField('date published')

def \_\_str\_\_(self):

return self.question\_text

class Choice(models.Model):

question = models.ForeignKey(Question, on\_delete = models.CASCADE)

choice\_text = models.CharField(max\_length = 200)

votes = models.IntegerField(default = 0)

def \_\_str\_\_(self):

return self.choice\_text

**views.py -**

from django.template import loader

from django.http import HttpResponse, HttpResponseRedirect

from django.shortcuts import get\_object\_or\_404, render,Http404

from django.urls import reverse

from .models import Question, Choice

def index(request):

latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

context = {'latest\_question\_list': latest\_question\_list}

return render(request, 'polls/index.html', context)

# Show specific question and choices

def detail(request, question\_id):

try:

question = Question.objects.get(pk = question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

return render(request, 'polls/detail.html', {'question': question})

# Get question and display results

def results(request, question\_id):

question = get\_object\_or\_404(Question, pk = question\_id)

return render(request, 'polls/results.html', {'question': question})

# Vote for a question choice

def vote(request, question\_id):

# print(request.POST['choice'])

question = get\_object\_or\_404(Question, pk = question\_id)

try:

selected\_choice = question.choice\_set.get(pk = request.POST['choice'])

except (KeyError, Choice.DoesNotExist):

# Redisplay the question voting form.

return render(request, 'polls/detail.html', {

'question': question,

'error\_message': "You didn't select a choice.",

})

else:

selected\_choice.votes += 1

selected\_choice.save()

# Always return an HttpResponseRedirect after successfully dealing

# with POST data. This prevents data from being posted twice if a

# user hits the Back button.

return HttpResponseRedirect(reverse('polls:results', args =(question.id, )))

**urls.py -**

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('polls/', include('polls.urls')),

path('admin/', admin.site.urls),

]

**admin.py -**

from django.contrib import admin

# Register your models here.

from .models import Question, Choice

# admin.site.register(Question)

# admin.site.register(Choice)

admin.site.site\_header = "Pollster Admin"

admin.site.site\_title = "Pollster Admin Area"

admin.site.index\_title = "Welcome to the Pollster Admin Area"

class ChoiceInLine(admin.TabularInline):

model = Choice

extra = 3

class QuestionAdmin(admin.ModelAdmin):

fieldsets = [(None, {'fields': ['question\_text']}), ('Date Information', {

'fields': ['pub\_date'], 'classes': ['collapse']}), ]

inlines = [ChoiceInLine]

admin.site.register(Question, QuestionAdmin)

**index.html -**

{% block content %}

<h1 class ="text-center mb-3">Poll Questions</h1>

{% if latest\_question\_list %}

{% for question in latest\_question\_list %}

<div class ="card-mb-3">

<div class ="card-body">

<p class ="lead">{{ question.question\_text }}</p>

<a href ="{% url 'polls:detail' question.id %}" class ="btn btn-primary btn-sm">Vote Now</a>

</div>

</div>

{% endfor %}

{% else %}

<p>No polls available</p>

{% endif %}

{% endblock %}

**Detail.html -**

{% block content %}

<h1 class ="text-center mb-3">{{ question.question\_text }}</h1>

{% if error\_message %}

<p class ="alert alert-danger">

<strong>{{ error\_message }}</strong>

</p>

{% endif %}

<form action ="{% url 'polls:vote' question.id %}" method ="post">

{% csrf\_token %}

{% for choice in question.choice\_set.all %}

<div class ="form-check">

<input type ="radio" name ="choice" class ="form-check-input" id ="choice{{ forloop.counter }}"

value ="{{ choice.id }}" />

<label for ="choice{{ forloop.counter }}">{{ choice.choice\_text }}</label>

</div>

{% endfor %}

<input type ="submit" value ="Vote" class ="btn btn-success btn-lg btn-block mt-4" />

</form>

{% endblock %}

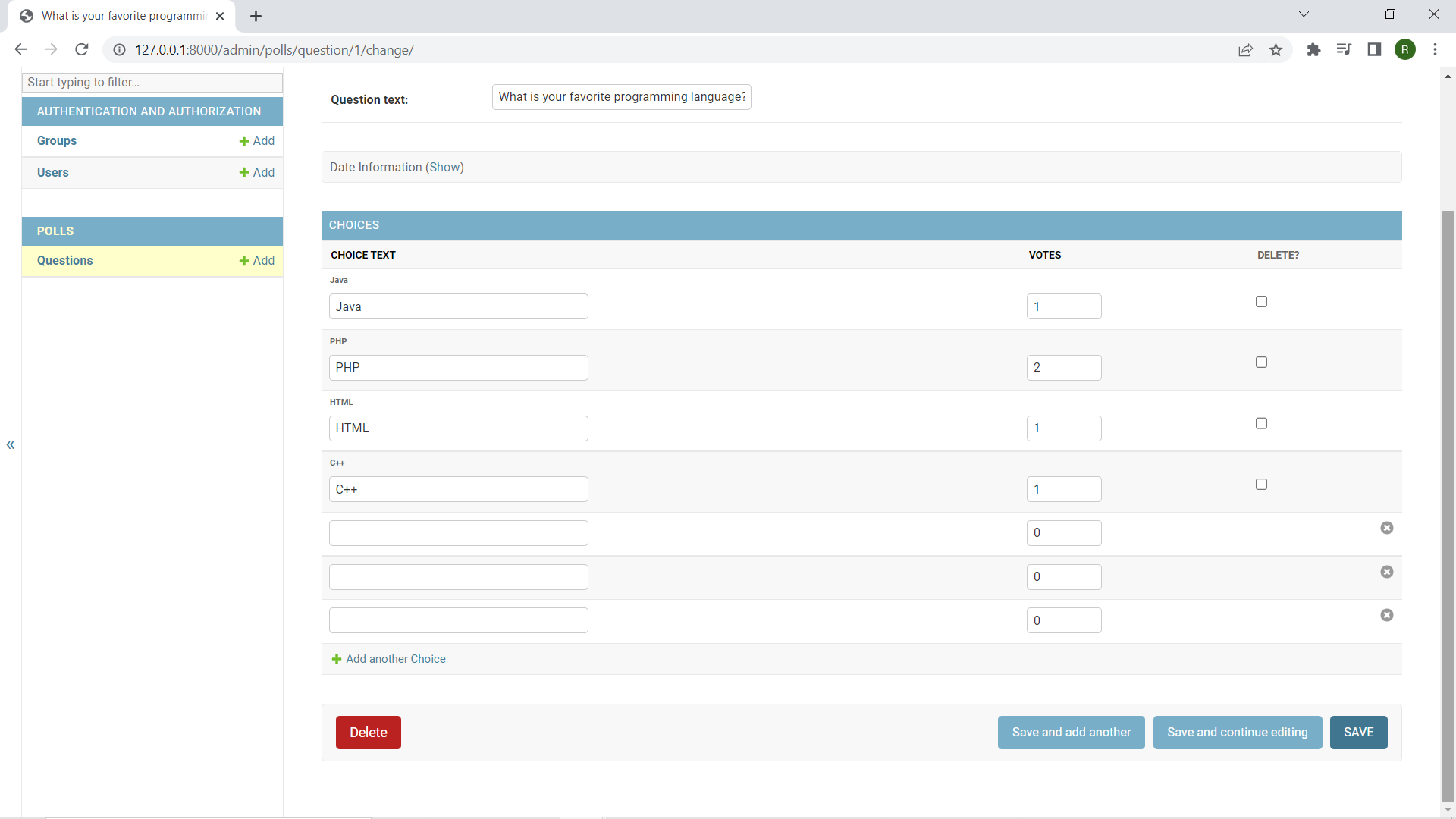
**Results.html -**

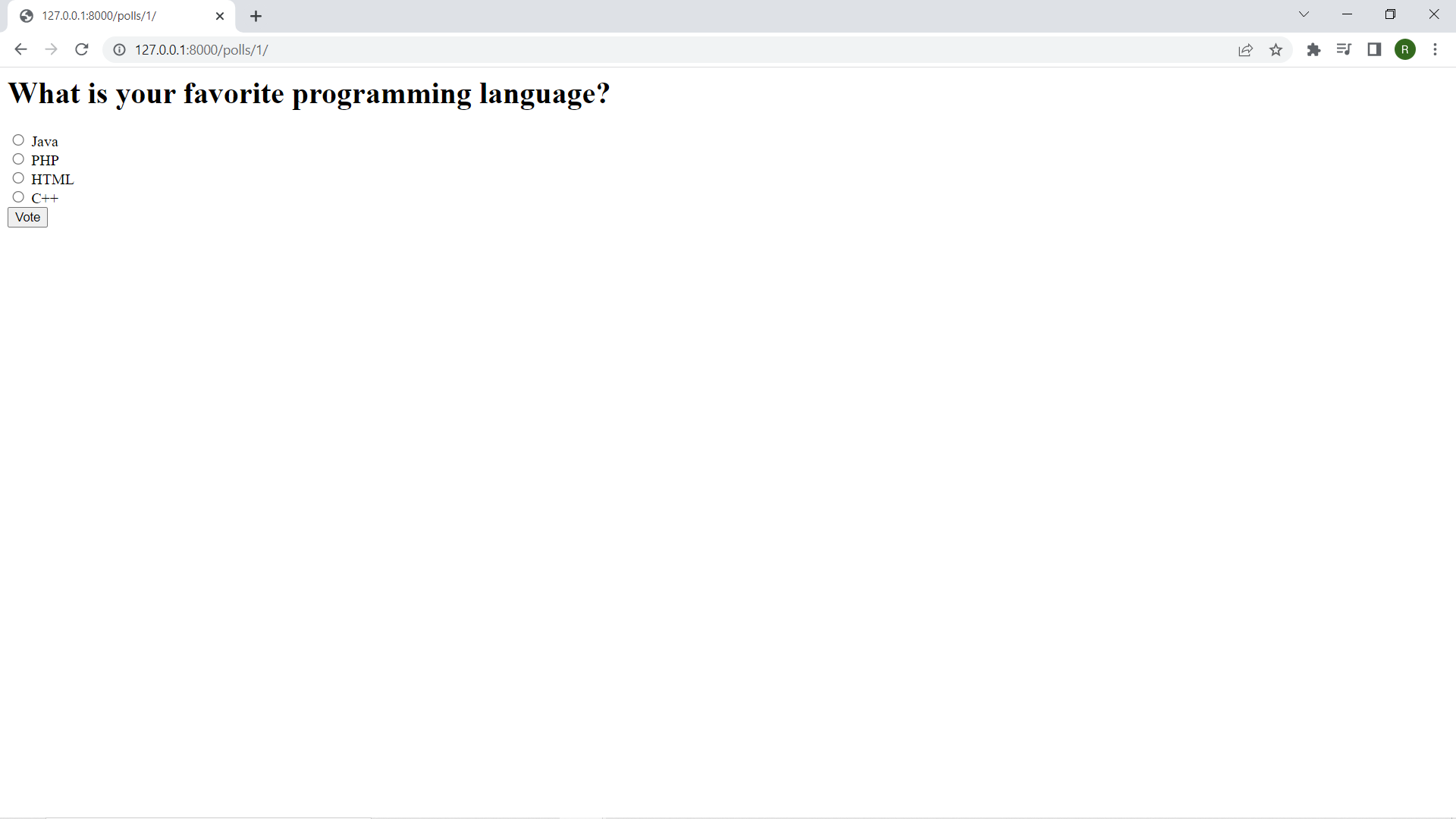
{% block content %}

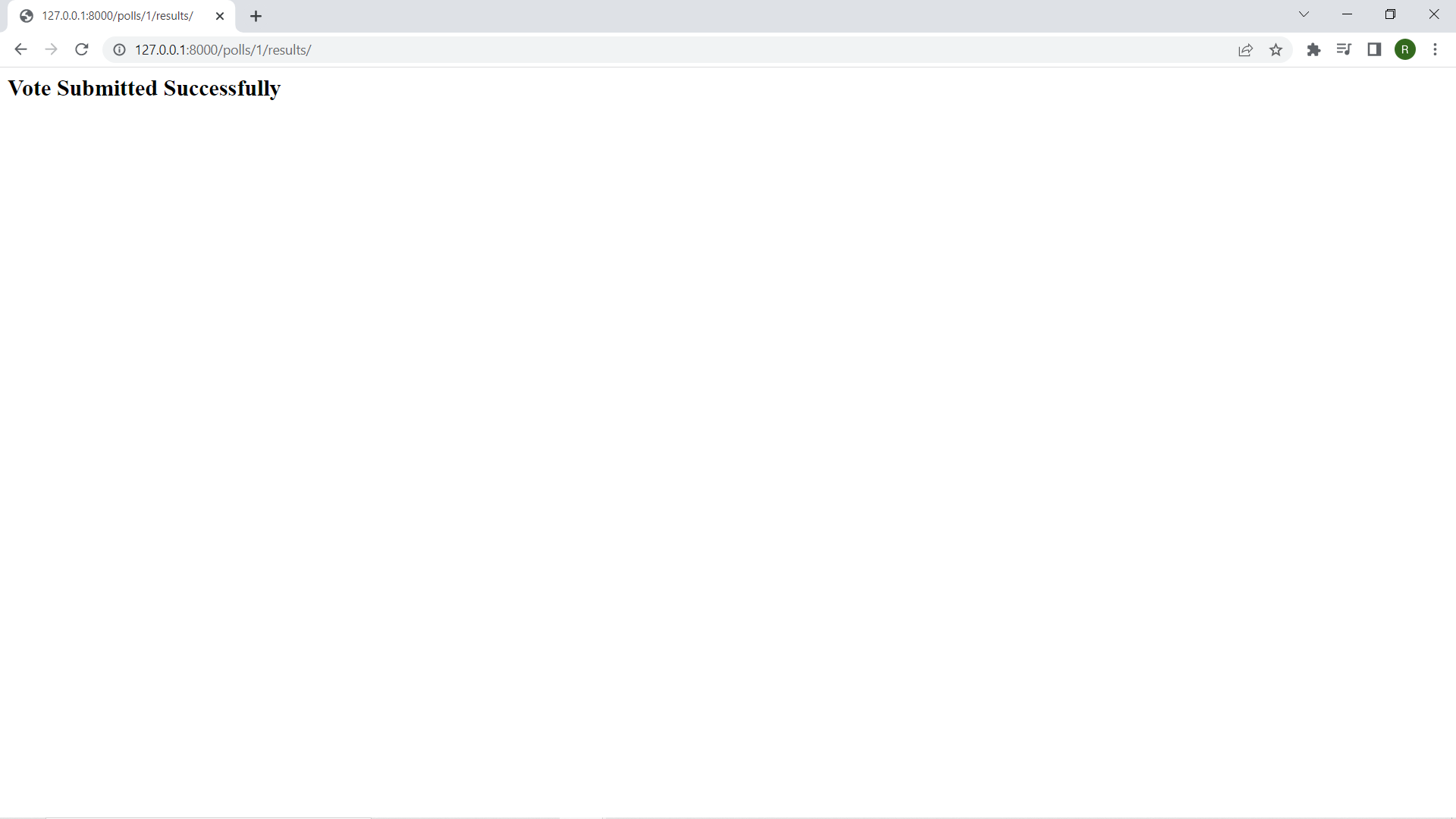
<h2> Vote Submitted Successfully</h2>

{% endblock %}

**//Output-**







**23. A public site in which user can pick their favourite programming language and vote.**

**views.py -**

from django.template import loader

from django.http import HttpResponse, HttpResponseRedirect

from django.shortcuts import get\_object\_or\_404, render,Http404

from django.urls import reverse

from .models import Question, Choice

def index(request):

latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

context = {'latest\_question\_list': latest\_question\_list}

return render(request, 'polls/index.html', context)

# Show specific question and choices

def detail(request, question\_id):

try:

question = Question.objects.get(pk = question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

return render(request, 'polls/detail.html', {'question': question})

# Get question and display results

def results(request, question\_id):

question = get\_object\_or\_404(Question, pk = question\_id)

return render(request, 'polls/results.html', {'question': question})

# Vote for a question choice

def vote(request, question\_id):

# print(request.POST['choice'])

question = get\_object\_or\_404(Question, pk = question\_id)

try:

selected\_choice = question.choice\_set.get(pk = request.POST['choice'])

except (KeyError, Choice.DoesNotExist):

# Redisplay the question voting form.

return render(request, 'polls/detail.html', {

'question': question,

'error\_message': "You didn't select a choice.",

})

else:

selected\_choice.votes += 1

selected\_choice.save()

# Always return an HttpResponseRedirect after successfully dealing

# with POST data. This prevents data from being posted twice if a

# user hits the Back button.

return HttpResponseRedirect(reverse('polls:results', args =(question.id, )))

**urls.py -**

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('polls/', include('polls.urls')),

path('admin/', admin.site.urls),

]

**index.html -**

{% block content %}

<h1 class ="text-center mb-3">Poll Questions</h1>

{% if latest\_question\_list %}

{% for question in latest\_question\_list %}

<div class ="card-mb-3">

<div class ="card-body">

<p class ="lead">{{ question.question\_text }}</p>

<a href ="{% url 'polls:detail' question.id %}" class ="btn btn-primary btn-sm">Vote Now</a>

</div>

</div>

{% endfor %}

{% else %}

<p>No polls available</p>

{% endif %}

{% endblock %}

**Detail.html -**

{% block content %}

<h1 class ="text-center mb-3">{{ question.question\_text }}</h1>

{% if error\_message %}

<p class ="alert alert-danger">

<strong>{{ error\_message }}</strong>

</p>

{% endif %}

<form action ="{% url 'polls:vote' question.id %}" method ="post">

{% csrf\_token %}

{% for choice in question.choice\_set.all %}

<div class ="form-check">

<input type ="radio" name ="choice" class ="form-check-input" id ="choice{{ forloop.counter }}"

value ="{{ choice.id }}" />

<label for ="choice{{ forloop.counter }}">{{ choice.choice\_text }}</label>

</div>

{% endfor %}

<input type ="submit" value ="Vote" class ="btn btn-success btn-lg btn-block mt-4" />

</form>

{% endblock %}

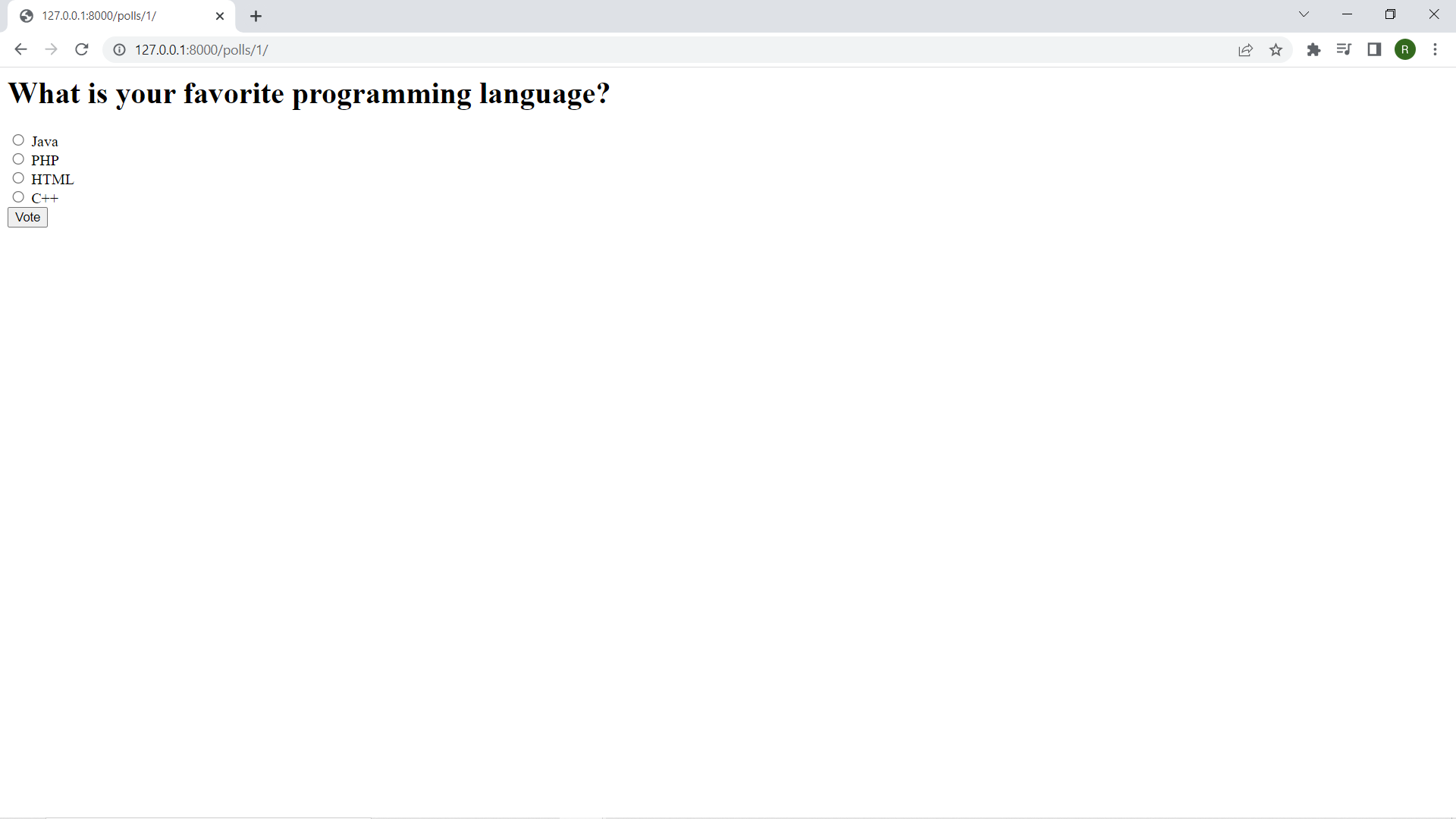
**Results.html-**

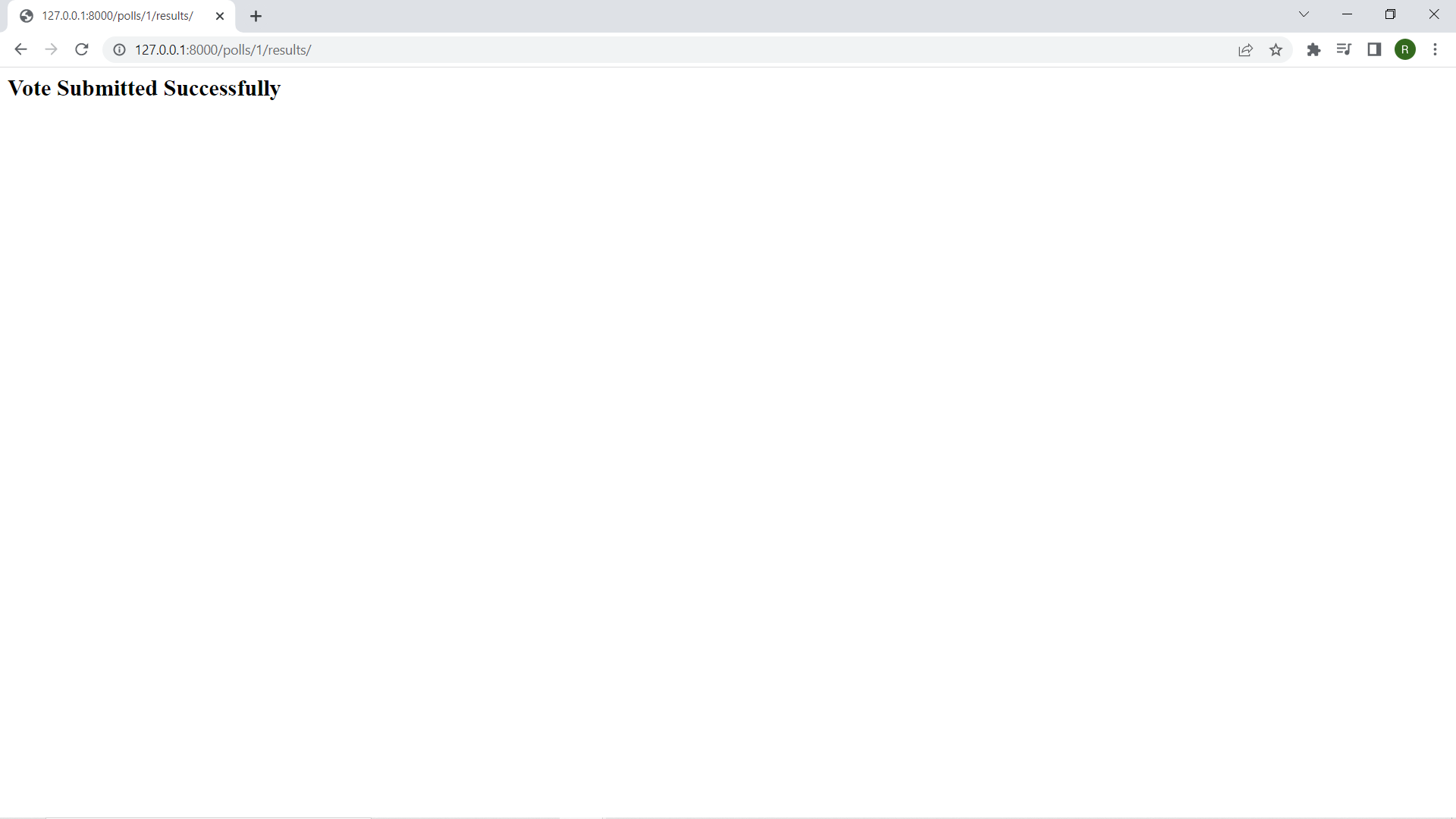
{% block content %}

<h2> Vote Submitted Successfully</h2>

{% endblock %}

**//Output-**





**24.An admin site that lets you add, change and delete programming languages.**

**models.py -**

from django.db import models

class Question(models.Model):

question\_text = models.CharField(max\_length = 200)

pub\_date = models.DateTimeField('date published')

def \_\_str\_\_(self):

return self.question\_text

class Choice(models.Model):

question = models.ForeignKey(Question, on\_delete = models.CASCADE)

choice\_text = models.CharField(max\_length = 200)

votes = models.IntegerField(default = 0)

def \_\_str\_\_(self):

return self.choice\_text

**admin.py -**

from django.contrib import admin

# Register your models here.

from .models import Question, Choice

# admin.site.register(Question)

# admin.site.register(Choice)

admin.site.site\_header = "Pollster Admin"

admin.site.site\_title = "Pollster Admin Area"

admin.site.index\_title = "Welcome to the Pollster Admin Area"

class ChoiceInLine(admin.TabularInline):

model = Choice

extra = 3

class QuestionAdmin(admin.ModelAdmin):

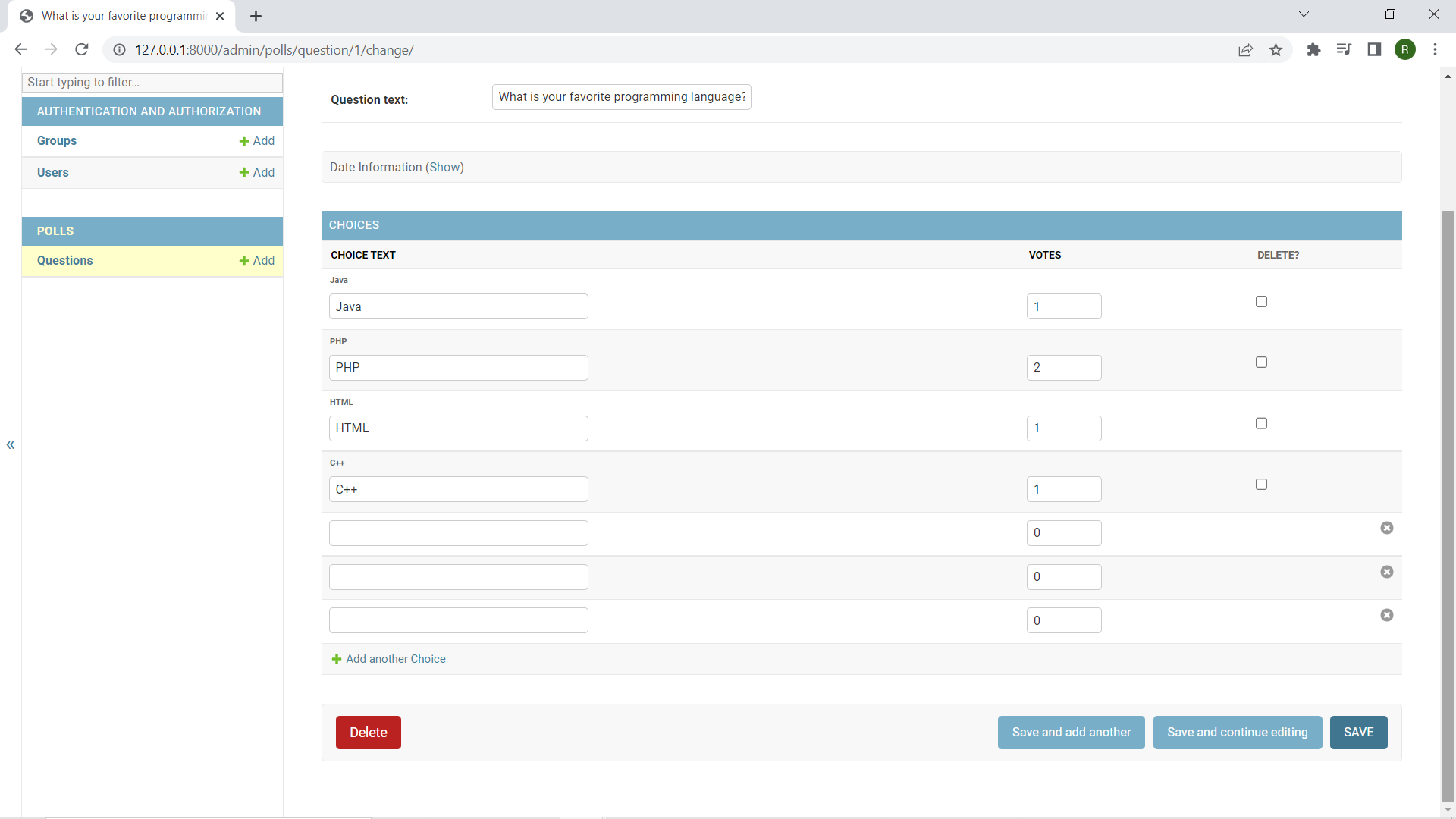
fieldsets = [(None, {'fields': ['question\_text']}), ('Date Information', {

'fields': ['pub\_date'], 'classes': ['collapse']}), ]

inlines = [ChoiceInLine]

admin.site.register(Question, QuestionAdmin)

**//Output-**



**25.Create your own blog using Django.**

**models.py -**

from django.db import models

from django.contrib.auth.models import User

class myblog(models.Model):

title = models.CharField(max\_length=200, unique=True)

slug = models.CharField(max\_length=200, unique=True)

**views.py -**

from django.shortcuts import render

from blog.models import myblog

def myf(req):

model = myblog

context = {"reg":myblog.objects.all()}

return render(req,'blog.html',context)

**urls.py -**

from django.contrib import admin

from django.urls import path

from blog.views import myf

urlpatterns = [

path('admin/', admin.site.urls),

path('blog/',myf)

]

**blog.html -**

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<div class="header">

<h2>Welcome to my Blog</h2>

</div>

<div class="row">

<div class="leftcolumn">

<div class="card">

<h2>Django</h2>

<h5>MSc. Computer Science Part 2</h5>

</div>

</div>

<div class="rightcolumn">

<div class="card">

<h2>Subjects :</h3>

<h4>SADP</h6>

<h4>Web Frameworks</h6>

<h4>Machine Learning</h6>

</div>

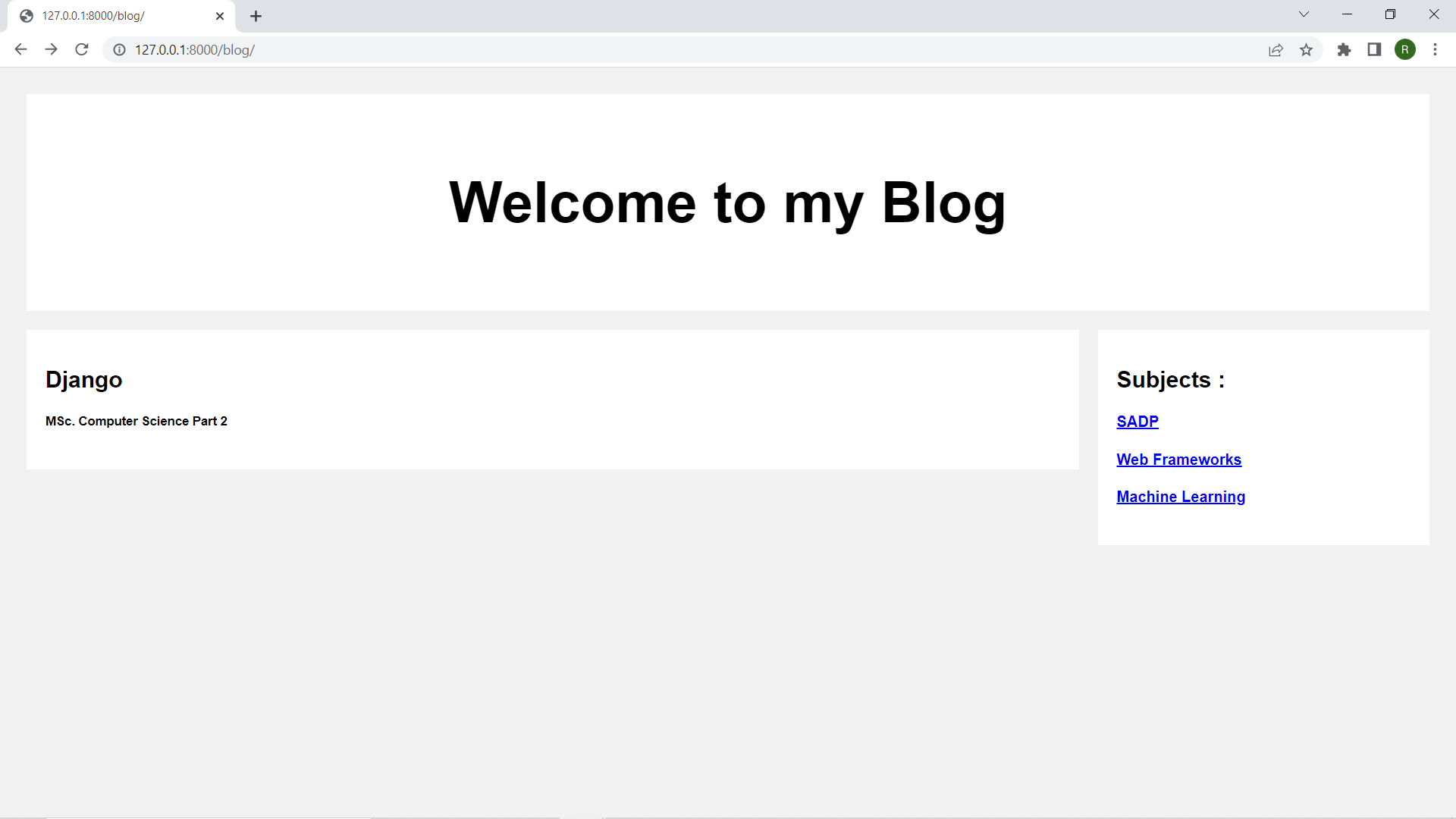
</div>

</div>

</body>

</html>

**//Output-**



**26. Create a clone of the “Hacker News” website.**

**urls.py –**

from django.contrib import admin

from django.urls import path

from newclone.views import newclone

urlpatterns = [

path('admin/', admin.site.urls),

path('home/', newclone),

]

**views.py –**

from django.shortcuts import render

def newclone(req):

return render(req,'index.html')

**index.html –**

as per design.

**//Output –**

