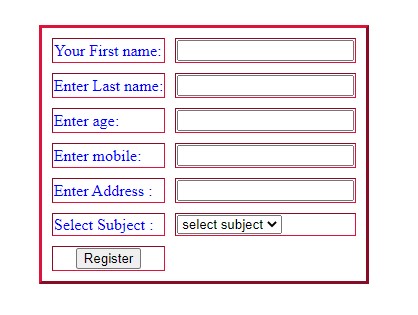
1. Create an HTML form that contain the Student Registration details and write a

JavaScript to validate Student first and last name as it should not contain other than

alphabets and age should be between 18 to 50.

Student-fom.htm

| <html>  <head>  <title style="color: blue;">Student Form</title>  </head>  <body>  <div id="error"></div>  <form id="form1" onsubmit="validation()">  <table align="center" border="3" cellspacing ="10" style="color: blue; bordercolor: crimson;">  <tr><td>Your First name: </td><td><input type="text" id="fname" name="Fname"></td></tr><br>  <tr><td>Enter Last name: </td><td><input type ="text" id="lname" name="lname"></td></tr><br>  <tr><td>Enter age: </td><td><input type ="text" id="age" name="age"></td></tr><br>  <tr><td>Enter mobile: </td><td><input type ="text" id="mobile" name="mobile"></td></tr><br>  <tr><td>Enter Address : </td><td><input type ="text" id="address" name="address"></td></tr><br>  <tr><td>Select Subject :</td><td><Select type="text" name="select" value="-1">  <option >select subject</option>  <option name ="BSC">BSC</option>  <option name ="BSC">BSC(CS)</option>  <option name ="BSC">BSC(CA)</option> </Select></td></tr>  <tr><td style="text-align: center;">  <input type = "submit" value="Register"></td></tr>  </table>  </form>  <script src="validateJS.js" type="text/javascript">  </script>  </body>  </html> |
| --- |



ValidateJS.js

| function validation(){ const fname=document.getElementById("fname") const lname=document.getElementById("lname") const form = document.getElementById("form1") const age = document.getElementById("age") const error = document.getElementById("error") const mobile = document.getElementById("mobile") const address = document.getElementById("address") const pattern = /^[A-Z a-z]+$/; const mpatrn = /^([9]{1})-([0-9]{9})$/ const addpatrn = /^[A-Z a-z 0-9]+$/ if(!pattern.test(fname.value))  { alert("first name should contain alphabate only!!") return false  }if(!pattern.test(lname.value))  { alert("Last name should contain alphabates only") return false  }if(age.value <= 18 || age.value > 50)  {  alert("Age should be between 18 to 50 ") return false  }if(mobile.value.length != 10)  {  alert("Mobile number should be of ten numbers") return false  }if(!addpatrn.test(address.value))  { alert("Address does not contains special character") return false  }} |
| --- |

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

2. Create an HTML form that contain the Employee Registration details and write a

JavaScript to validate DOB, Joining Date, and Salary.

Employee-form.htm

<html>

<head>

</head>

<center><h2>Employee Registration Form</h2></center>

<body>

<div id="error"></div>

<form id="form1" onsubmit="validation()">

<table align="center" border="3" cellspacing ="10">

<tr><td>Your First name: </td><td><input type="text" id="fname" name="Fname"></td></tr><br>

<tr><td>Enter Last name: </td><td><input type ="text" id="lname" name="lname"></td></tr><br>

<tr><td>Enter age: </td><td><input type ="text" id="age" name="age"></td></tr><br>

<tr><td>Enter mobile: </td><td><input type ="text" id="mobile" name="mobile"></td></tr><br>

<tr><td>Enter Address : </td><td><input type ="text" id="address" name="address"></td></tr><br>

<tr><td>Select Designation :</td><td><Select type="text" id="desig" name="designation" >

<option value="null" >select designation</option>

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

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

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

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

<option value ="Employee ">Technical support</option>

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

</Select></td></tr>

<tr><td>Date OF Birth(DOB) </td><td><input type ="text" id="dob" name="dob"></td></tr><br>

<tr><td>Date OF Joining </td><td><input type ="text" id="doj" name="doj"></td></tr><br>

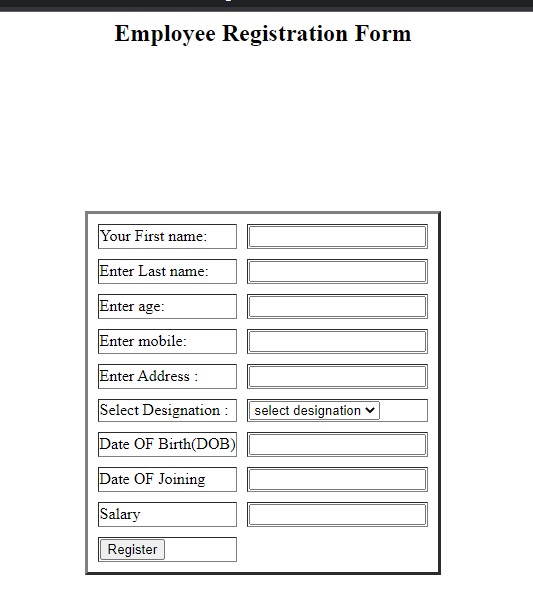
<tr><td>Salary </td><td><input type ="text" id="sal" name="sal"></td></tr><br>

<tr><td><input type = "submit" value="Register"></td></tr>

</table>

</form>

<script src="Assignment2.js" type="text/javascript"></script></body></html>



Assingnment2.js

| function validation(){  const fname=document.getElementById("fname") const lname=document.getElementById("lname") const form = document.getElementById("form1") const age = document.getElementById("age") const error = document.getElementById("error") const mobile = document.getElementById("mobile") const address = document.getElementById("address") const dob = document.getElementById("dob") const desig = document.getElementById("desig") const dojoin = document.getElementById("doj") const sal = document.getElementById("sal")    const salpattern = /^\d{1,6}(?:\.\d{0,2})?$/ const dobPattern=/(((0|1)[0-9]|2[0-9]|3[0-1])\/(0[1-9]|1[0-  2])\/((19|20)\d\d))$/; const pattern = /^[A-Z a-z]+$/; const mpatrn = /^([9]{1})-([0-9]{9})$/ const addpatrn = /^[A-Z a-z 0-9]+$/  if(!pattern.test(fname.value))  {  alert("first name should contain alphabate only or it can't be null") return false  }  if(!pattern.test(lname.value))  {  alert("Last name should contain alphabates only") return false |
| --- |
| } if(age.value <= 18 || age.value > 50)  { alert("Age should be between 18 to 50 ") return false  } if(mobile.value.length != 10)  { alert("Mobile number should be of ten numbers") return false    } if(!addpatrn.test(address.value))  { alert("Address does not contains special character") return false  } if(!dobPattern.test(dob.value))  { alert("Enter birth date in [dd/mm/yyyy] format") return false  } if(desig.value == "")  { alert("Select your designation") return false  } if(!dobPattern.test(dojoin.value))  { alert("Enter join date in [dd/mm/yyyy] format ") return false  } if(!salpattern.test(sal.value))  {  alert("Something is wrong while entering salary!") return false  }  } |

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

Regular Expression.

Login-Form.htm

<!DOCTYPE html>

<html>

<head>

</head>

<center><h2>Login Form</h2></center>

<body>

<div id="error1"></div>

<form id="form1" onsubmit="validation()">

<table align="center" border="3" cellspacing="10">

<tr><td>Your First name: </td><td><input type="text" id="fname" name="Fname"></td></tr>

<br>

<tr><td>Enter Last name: </td><td><input type="text" id="lname" name="lname"></td></tr>

<br>

<tr><td>Enter mobile: </td><td><input type="text" id="mobile" name="mobile"></td></tr>

<br>

<tr><td>Enter Address : </td><td><input type="text" id="address" name="address"></td></tr>

<br>

<tr><td>Enter email\_id : </td><td><input type="text" id="email" name="email" /></td></tr

<br />

<tr><td>Date OF Birth(DOB) </td><td><input type="text" id="dob" name="dob"></td></tr>

<br>

<tr><td><input type="submit" value="Login"></td></tr>

</table>

</form>

<script src="Valid.js" type="text/javascript">

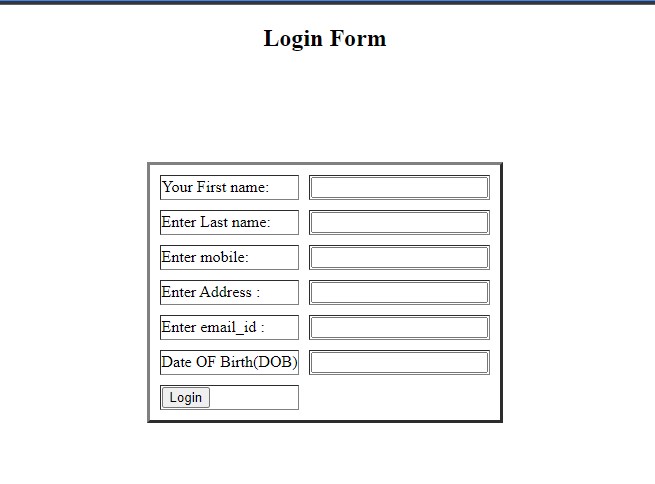
</script>

</body>

</html>

Valid.js

| function validation() { const fname = document.getElementById("fname") const lname = document.getElementById("lname") const form = document.getElementById("form1") const error1 = document.getElementById("error1") const mobile = document.getElementById("mobile") const address = document.getElementById("address") const email = document.getElementById("email") const pattern = /^[A-Z a-z]+$/; const mpatrn = /^([9]{1})-([0-9]{9})$/ const addpatrn = /^[A-Z a-z 0-9]+$/ const emailpattern = /^[a-zA-Z0-9.\_-]+@ [ a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/ if (!pattern.test(fname.value))  { alert("first name should contain alphabate only!!") return false  } if (!pattern.test(lname.value))  { alert("Last name should contain alphabates only") return false  } if (mobile.value.length != 10)  { alert("Mobile number should be of ten numbers") return false  } if (!addpatrn.test(address.value))  { alert("Address does not contains special character") return false  }  if (!emailpattern.test(email.value))  {  alert("Enter the correct email address") return false  }  } |
| --- |

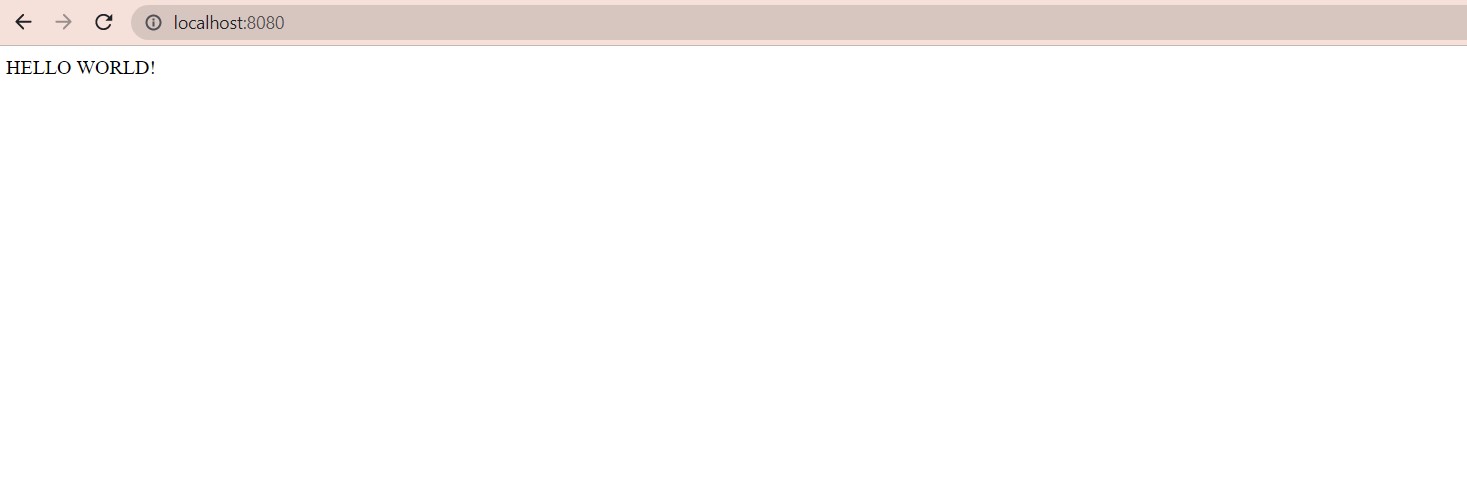


4. Create a Node.js file that will convert the output "Hello World!" into upper-case letters:

File.js

var http = require('http'); var uc = require('upper-case'); http.createServer(function (req, res) { res.writeHead(200, {'Content-Type': 'text/html'}); /\*Use our upper-case module to upper case a string:\*/ res.write(uc.upperCase("Hello World!")); res.end();

}).listen(8080);

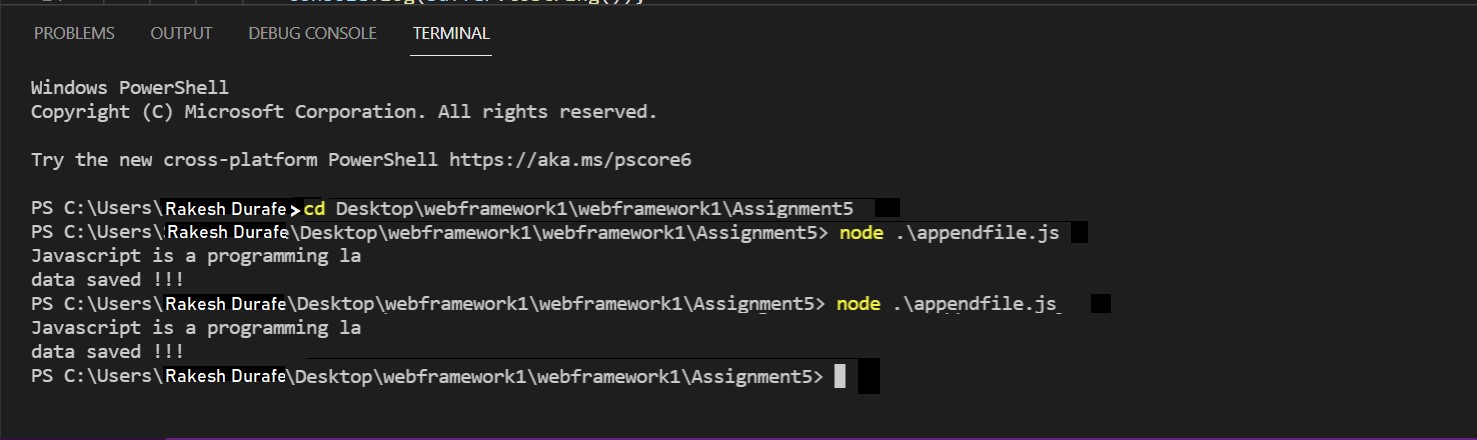


5. Using nodejs create a web page to read two file names from user and append contents

of first file into second file

| var fs = require('fs'); var file1 ='input.txt'; var file2 = 'output.txt';  function fileValidation()  { fs.open(file1,'r',function(err,fd){ if(err) { return console.error(err);  } var buffer = new Buffer.alloc(30); fs.read(fd,buffer,0,buffer.length,0,function(err,bytes){ if(err) throw err; console.log(buffer.toString());  }); fs.appendFile(file2,buffer,function(err){ if(err) throw err; console.log("data saved !!!"); fs.close(fd,function(err)  { if(err) throw err;});  });  });  }  fileValidation(); |
| --- |

Output:



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

| var http = require('http'); var url = require('url'); var fs = require('fs'); http.createServer(function (req, res) { var pathname = url.parse(req.url, true).pathname; console.log("Request for" + pathname + "received."); fs.readFile(pathname.substr(1), function (err, data) { if (err) { console.log(err); res.writeHead(404, { 'content-type': 'text/html' });  res.end('<html><body><h1>404 Not found</h1></body></html>');  } else { res.writeHead(200, { 'content-type': 'text/html' }); res.write(data);  res.end();  }  });    }).listen(9030); console.log('server is running on port 8080'); |
| --- |

index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Sample Page</title>

</head>

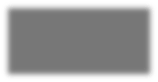
<body>

Hello World! Welcome to web module.

</body>

</html>

Out Put:



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

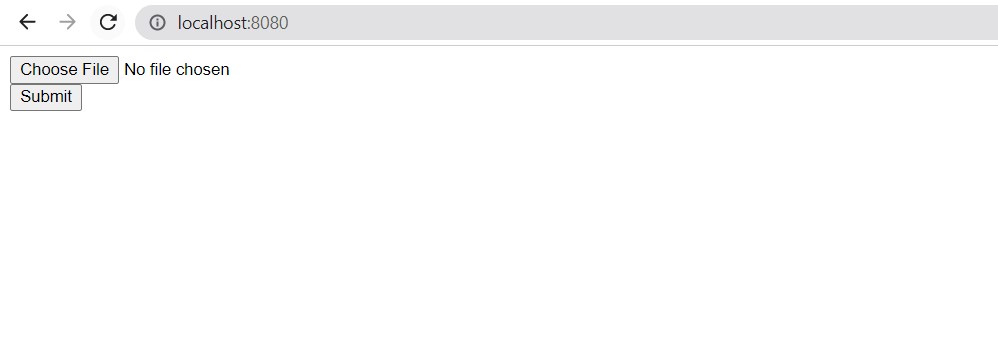
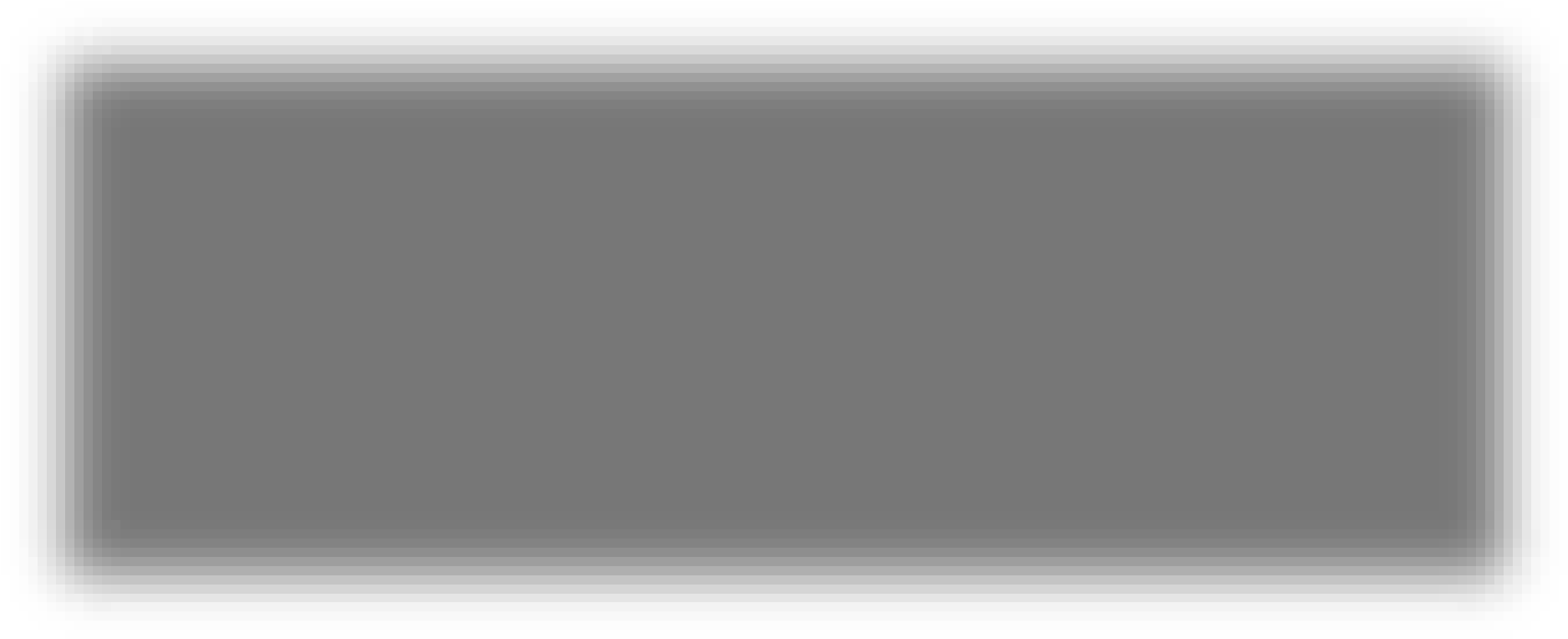
File.js

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/formdata">'); res.write('<input type="file" name="filetoupload"><br>'); res.write('<input type="submit">'); res.write('</form>'); return res.end();

}).listen(8080);

Out Put:

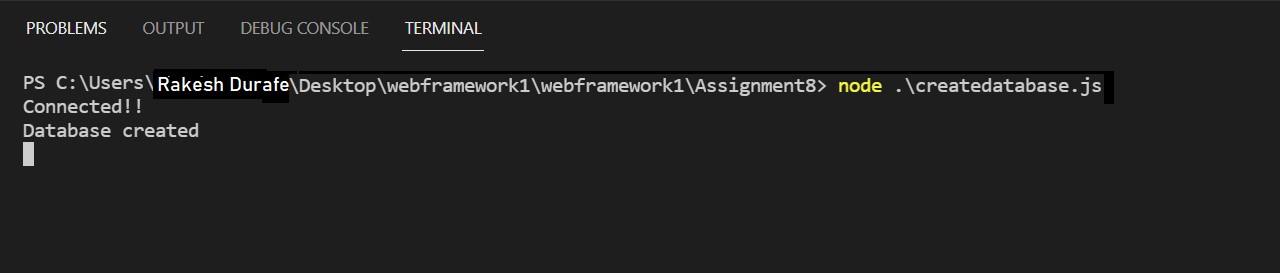


8. Create a Node.js file that demonstrate create database and table in MySQL

Createdatabase.js

| var mysql= require('mysql');  var con= mysql.createConnection({ host: "localhost",  user: "root",  password: "satara@123"  });  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");    });  }); |
| --- |
|  |

Out Put:



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

Db.js

| var mysql= require('mysql'); var con= mysql.createConnection({ host:"localhost", user:"root", password:"satara@123", database: "mydb"  }); con.connect(function(err){ if (err) throw err;  con.query("select name , address from Customer",function(err,result,fields){ if (err) throw err; console.log(result);  });  }); |
| --- |

Out Put:



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

Student.js

| var mysql=require('mysql');  var con=mysql.createConnection({ host:"localhost", user:"root", password:"satara@123", database:"mydb"  }); con.connect(function(err){ if (err) throw err; console.log("Connected!!!"); var sql="INSERT INTO Student (name,address) Values ?"; var values=[  ['Pritam','Highway 71'],  ['Sneha','Lowstreet 4'],  ['Sid','Apple st 652'],  ['Sam','Valley 345'],  ['Michael','Green Greass 1'],  ['Griss','One way 98'],  ['Richard','Sky st 331']  ]; con.query(sql,[values],function(err,result){ if (err) throw err;  console.log("Number of records inserted:" + result.affectedRows);  });  }); |
| --- |

Out Put:

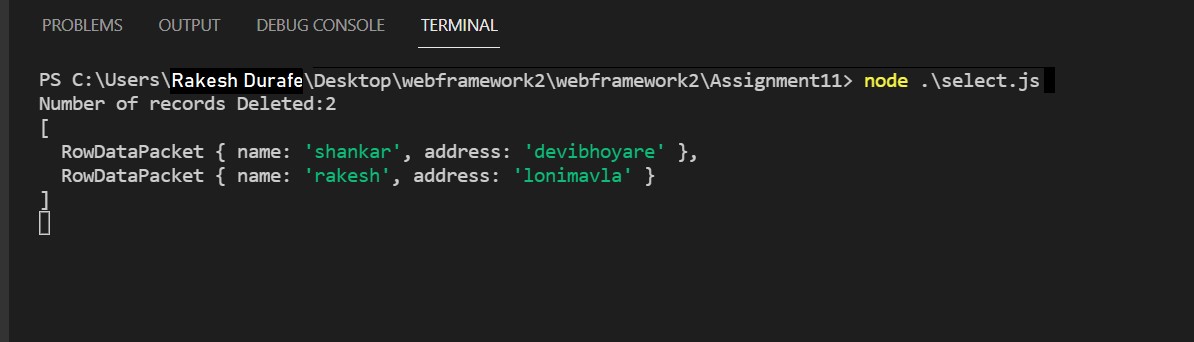


11. Create a node.js file that Select all records from the "customers" table, and delete the specified record.

DbConnect.js

| var mysql= require('mysql'); var con= mysql.createConnection({ host:"localhost", user:"root", password:"satara@123", database: "mydb"  }); con.connect(function(err){ if (err) throw err;  con.query("select name , address from Customer",function(err,result,fields){ if (err) throw err; console.log(result);  }); }); var sql = "DELETE FROM Customer WHERE address='Valley 345'"; con.query(sql,function(err,result){ if (err) throw err;  console.log("Number of records Deleted:" + result.affectedRows);  }); |
| --- |

Out Put:



12. Create a Simple Web Server using node js.

Webserver.js

| var mysql= require('mysql'); var con= mysql.createConnection({ host:"localhost", user:"root", password:"satara@123", database: "mydb"  }); con.connect(function(err){ if (err) throw err;  con.query("select name , address from Customer",function(err,result,fields){ if (err) throw err; console.log(result);  }); }); var sql = "DELETE FROM Customer WHERE address='Valley 345'"; con.query(sql,function(err,result){ if (err) throw err; console.log("Number of records Deleted:" + result.affectedRows); }); |
| --- |

Out Put:



13: Using node js create a User Login System

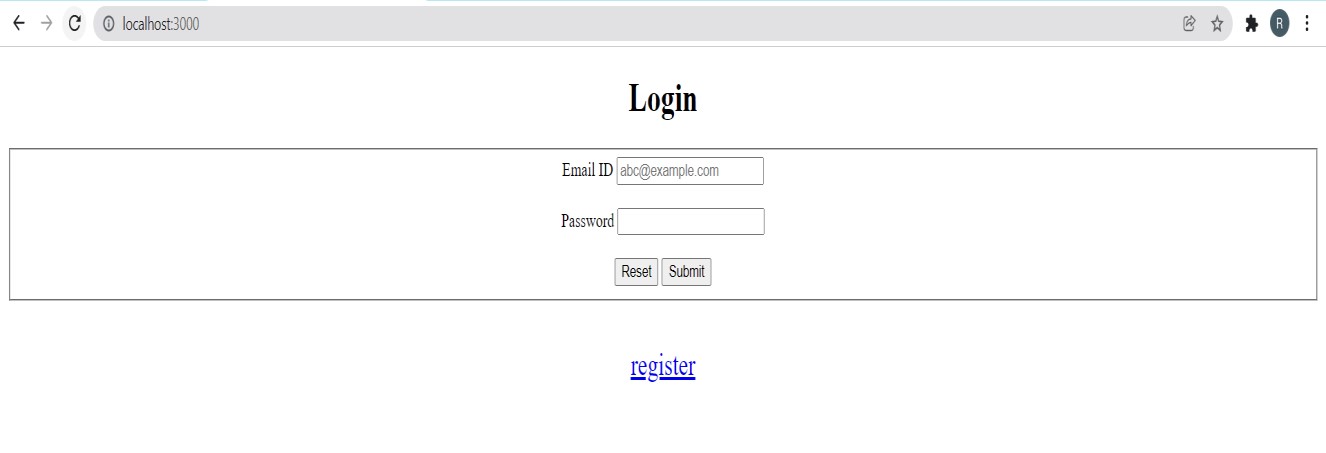
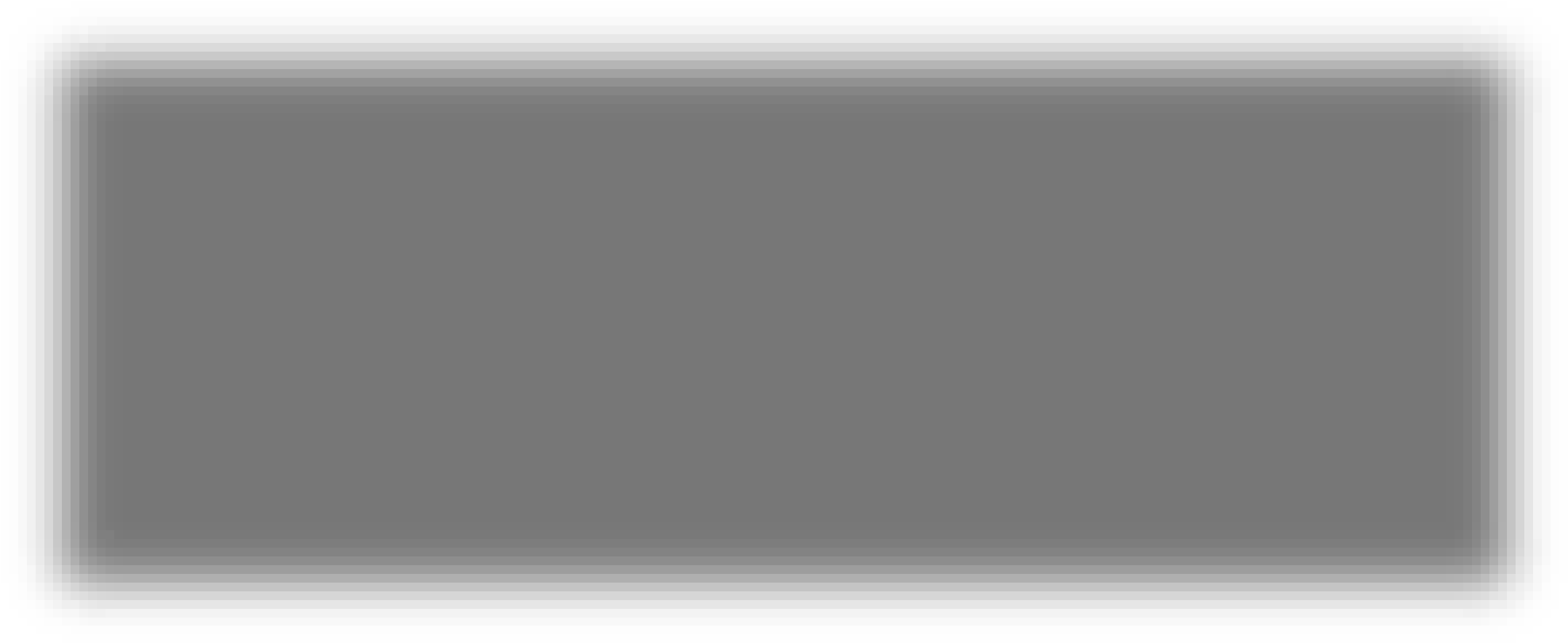
Index.html

| <!DOCTYPE html>  <html lang = "en">  <head>  <meta charset = "UTF-8">  <title> My Form </title>  <style> #mylink{ font-size: 25px;  }  </style>  </head>  <body align='center'>  <header>  <h1>Login</h1>  </header>    <form action="/login" method="POST">  <fieldset>    <label>Email ID</label>  <input type ="email" id = 'email' name="email" placeholder="abc@example.com" required>  <br><br>    <label>Password</label>  <input type="password" id = "password" name="password" required>  <br><br>    <button type ="reset">Reset</button>  <button type ="submit">Submit</button>  </fieldset>  </form>  <br><br>  <a id="mylink" href="./registration.html">register</a>    </body>  </html> |
| --- |

File.js

| const express = require('express'); const http = require('http'); const bcrypt = require('bcrypt'); const path = require("path"); const bodyParser = require('body-parser'); const users = require('./data').userDB;  const app = express(); const server = http.createServer(app);  app.use(bodyParser.urlencoded({extended: false})); app.use(express.static(path.join(\_\_dirname,'./')));    app.get('/',(req,res) => { res.sendFile(path.join(\_\_dirname,'./index.html')); });  app.post('/register', async (req, res) => { try{ let foundUser = users.find((data) => req.body.email === data.email); if (!foundUser) {  let hashPassword = await bcrypt.hash(req.body.password, 10);  let newUser = { id: Date.now(), username: req.body.username, email: req.body.email, password: hashPassword,  };  users.push(newUser);  console.log('User list', users);  res.send("<div align ='center'><h2>Registration successful</h2></div><br><br><div align='center'><a href='./login.html'>login</a></div><br><br><div align='center'><a href='./registration.html'>Register another user</a></div>");  } else {  res.send("<div align ='center'><h2>Email already used</h2></div><br><br><div align='center'><a  href='./registration.html'>Register again</a></div>");  } } catch{  res.send("Internal server error");  } |
| --- |
| }); app.post('/login', async (req, res) => { try{ let foundUser = users.find((data) => req.body.email === data.email); if (foundUser) {  let submittedPass = req.body.password; let storedPass = foundUser.password;  const passwordMatch = await bcrypt.compare(submittedPass, storedPass); if (passwordMatch) { let usrname = foundUser.username; res.send(`<div align ='center'><h2>login successful</h2></div><br><br><br><div align ='center'><h3>Hello  ${usrname}</h3></div><br><br><div align='center'><a href='./login.html'>logout</a></div>`);  } else { res.send("<div align ='center'><h2>Invalid email or password</h2></div><br><br><div align ='center'><a href='./login.html'>login again</a></div>");  } } else {  let fakePass = `$2b$$10$ifgfgfgfgfgfgfggfgfgfggggfgfgfga`; await bcrypt.compare(req.body.password, fakePass);  res.send("<div align ='center'><h2>Invalid email or password</h2></div><br><br><div align='center'><a href='./login.html'>login again<a><div>");  } } catch{  res.send("Internal server error");  }  });  server.listen(3000, function(){ console.log("server is listening on port: 3000");  }); |

Out Put:



14. Using node js create an eLearning System

App.js

| var express = require('express'); var path = require('path'); var favicon = require('serve-favicon'); var logger = require('morgan'); var cookieParser = require('cookie-parser'); var bodyParser = require('body-parser'); var mongoose = require('mongoose');    var flash = require('connect-flash'); var passport = require('passport'); var cookieParser = require('cookie-parser'); var session = require('express-session');  /\* var routes = require('./routes/index'); var users = require('./routes/users'); var login = require('./routes/login');  \*/ var app = express();  var CourseHandler = require('./app/controllers/courseController.server.js');    var configDB = require('./app/config/database');  mongoose.connect(configDB.url);    require('./app/config/passport')(passport);    // view engine setup  app.set('views', path.join(\_\_dirname, 'views')); app.set('view engine', 'ejs');    app.use(session({ secret: 'asdf', saveUninitialized: true, resave: true |
| --- |
| })); // session secret app.use(passport.initialize()); app.use(passport.session()); // persistent login sessions app.use(flash()); // use connect-flash for flash messages stored in session  // uncomment after placing your favicon in /public app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico'))); app.use(logger('dev')); app.use(bodyParser.json()); app.use(bodyParser.urlencoded({ extended: false  })); app.use(cookieParser()); app.use(express.static(path.join(\_\_dirname, 'public')));  require('./routes/route')(app, passport);  // catch 404 and forward to error handler app.use(function (req, res, next) { var err = new Error('Not Found'); err.status = 404; next(err);  }); if (app.get('env') === 'development') { app.use(function (err, req, res, next) { res.status(err.status || 500); res.render('error', { message: err.message, error: err });  });  }  app.use(function (err, req, res, next) { res.status(err.status || 500); res.render('error', { message: err.message, error: {}  });  });  module.exports = app; |

CourseController.server.js

| 'use strict';  var Users = require('../models/users.js'); var course = require('../models/course.js'); var Paper = require('../models/paper.js'); var Question = require('../models/question.js'); var Studentpaper = require('../models/studentpaper.js'); var path = process.cwd(); var paypalconfig = require('../config/ppconfig'); var Employees = require('../models/employee'); var Paymentinfos = require('../models/paymentinfo'); var Chartdatas = require('../models/chartdata'); var Enrolls = require('../models/enroll');  var paypal = require('paypal-rest-sdk');  var ch = new ClickHandler(); ch.initPaypal();  function ClickHandler() { this.popularCourse = function (req, res) { course.find({}, { "name": 1  , "coverurl": 1  })  .exec(function (err, result) {  if (err) throw err;  Chartdatas.find({})  .exec(function (err, chart\_data) { var label = []  , data = [];    for (var i = 0; i < chart\_data.length; i++) { label.push(chart\_data[i]['label']); data.push(chart\_data[i]['data']) }  res.render('index', { result: result  , session: req.user  , label  , data  });  }); |
| --- |

| });  }; this.courseInfo = function (req, res) {  var cid = req.params.id; course.findById(cid, function (err, result) { console.log("This is is is cid " + result) if (err) throw err;    Paper.find({  "course\_id": cid  })  .exec(function (err, test) { if (err) throw err;  Studentpaper.find({  "student\_name": req.user.name  }, {  "paper\_id": 1  , "status": 1  })  .exec(function (err, status) { if(status[0]==null) status="no";  Paymentinfos.find({  "name": req.user.name  , "course": result['name']  })  .exec(function (err, payment) { if (payment[0] ==null) payment = "no"  Employees.find({})  .exec(function (err, emp\_info) {  if (emp\_info[0] == null) emp\_info = "no" var enroll\_info;  Enrolls.findOne({  "username": req.user.name  , "course": {  $in: [cid]  }  })  .exec(function (err, enroll) { if (enroll == null) |
| --- |

| enroll\_info = "no" else enroll\_info = enroll;  res.render('courseInfo', { result, test, status, session: req.user, payment, emp\_info, enroll\_info  });  })  });  })    })    });    });    };  this.getCourse = function (req, res) {  course.find({}, { "name": 1  })  .exec(function (err, result) { res.render("admin", { result  });  });    }; this.createPaper = function (req, res) { var courseId = req.params.id; var no = req.body.no; var paper = new Paper(); paper.course\_id = courseId; paper.paperNo = no; paper.save(function (err) { if (err) throw err; res.redirect("back");  }); |
| --- |

| };  this.courseMod = function (req, res) { var courseId = req.params.id; Paper.find({  "course\_id": courseId  })  .exec(function (err, result) { if (err) throw err; res.render("paper", { result  });  });    };  this.newQuestion = function (req, res) { res.render("question");    } this.addQuestion = function (req, res) { var paperId = req.params.pid;  var ques = new Question(); ques.paper\_id = paperId; ques.question = req.body.question; ques.options.a = req.body.opt1; ques.options.b = req.body.opt2; ques.options.c = req.body.opt3; ques.options.d = req.body.opt4; ques.answer = req.body.answer; ques.save(function (err) { if (err) throw err; res.redirect("back");  });    } this.showPaper = function (req, res) { var paperId = req.params.pid;    Question.find({  "paper\_id": paperId  }) |
| --- |

| .exec(function (err, result) { if (err) throw err;  res.render("testpaper", { result, session: req.user  });  });  }      this.processPaper = function (req, res) {  var pid = req.params.pid; var ans = req.body; var ansLen = Object.keys(ans).length; var q = []; var qp, marks = 0; console.log("asdasdasdasd") for (var i = 0; i < ansLen; i++) { var o = {}; o['question\_id'] = Object.keys(ans)[i]; o['answer'] = ans[Object.keys(ans)[i]];  q.push(o);  }    Question.find({  "paper\_id": pid  }, {  "answer": 1  })  .exec(function (err, result) { console.log(result); console.log(ans); qp = result; verifyAnswers(); checkexist(); res.redirect("back")  });    function verifyAnswers() {  for (var ab = 0; ab < Object.keys(ans).length; ab++) { if (ans[qp[ab].\_id] == qp[ab].answer) marks++;    }  } |
| --- |

| function checkexist() {    Studentpaper.count({  "student\_name": req.user.name  , "paper\_id": pid }).exec(function (err, result) { console.log("this is aaaa " + result); if (parseInt(result) == 0) saveDb(); else { var stats; if (marks < ansLen / 2) stats = "remove" else  stats = "ok"  Studentpaper.update({  "student\_name": req.user.name  , "paper\_id": pid  }, {  "status": stats  })  .exec(function (err, result) {  })  }  });    }  function saveDb() {  var sp = new Studentpaper(); sp.student\_name = req.user.name; sp.paper\_id = pid; sp.answers = q; if (marks < ansLen / 2) sp.status = "remove"; else  sp.status = "ok";  sp.save();  console.log(marks);  }    }; this.initPaypal = function () { paypal.configure(paypalconfig.api); |
| --- |

| console.log("adasdjaslkjdklasjdklasjdlkasjldkjasldjaslkdjaskldjaklsjdkalsjdlad asdjaslkjdklasjdklasjdlkasjldkjasldjaslkdjaskldjaklsjdkalsjdl"); }; this.pay = function (req, res) {  console.log("paypaypaypaypaypaypay"); var course\_name = req.body.course var payment = {  "intent": "sale"  , "payer": {  "payment\_method": "paypal"  }  , "redirect\_urls": {  "return\_url": "http://localhost:3000/a/execute"  , "cancel\_url": "http://localhost:3000/cancel"  }  , "transactions": [{  "amount": {  "total": "10.00"  , "currency": "USD"  }  , "description": "My awesome payment"  , "item\_list": {  "items": [{  "quantity": "1"  , "name": course\_name  , "price": "10.00"  , "sku": "product12345"  , "currency": "USD"  }]  }  }]  };    paypal.payment.create(payment, function (error, payment) { if (error) { console.log(error);  } else {  console.log(payment); if (payment.payer.payment\_method === 'paypal') { req.session.paymentId = payment.id; var redirectUrl; for (var i = 0; i < payment.links.length; i++) { var link = payment.links[i]; if (link.method === 'REDIRECT') { |
| --- |

| redirectUrl = link.href;  } } res.redirect(redirectUrl);  }  }  });  }; this.executePaypal = function (req, res) { var paymentId = req.session.paymentId; var payerId = req.query.PayerID; console.log(req.url); console.log(payerId); var details = {  "payer\_id": payerId  }; paypal.payment.execute(paymentId, details, function (error, payment) { if (error) { console.log(error);  } else { var course\_nme = payment.transactions[0].item\_list.items[0].name; var pamnt = new Paymentinfos(); pamnt.name = req.user.name; pamnt.course = course\_nme; pamnt.save(function (err) { res.redirect('/')  })    }  });  };  this.employeeRegister = function (req, res) {  res.render('jobEmployeeRegister', { session: req.user  }); };  this.postemployeeRegister = function (req, res) { Paymentinfos.find({  name: req.user.name  })  .exec(function (err, certificate) { var cert = []; for (var i = 0; i < certificate.length; i++) { cert.push(certificate[0]['course'])  } |
| --- |

| var employee = new Employees(); console.log(req.file) console.log(req.body) employee.Name = req.user.name employee.email = req.body.email employee.password = employee.generateHash(req.body.password); employee.Skills = req.body.Skills employee.contact = req.body.contactnumber employee.ResumeFilename = req.file.filename employee.ResumeFileOriginalname = req.file.originalname employee.Certificate = cert; employee.save(function (err) { res.redirect('/')  })  })  }; this.addVideo = function (req, res) {  var cid = req.params.id; var week = req.body.week;  course.findById(cid, function (err, course\_info) { if (course\_info == null) course\_info = "no" res.render("addVideo", { course\_info  })  }); }; this.saveVideo = function (req, res) {  //console.log(req.file) //console.log(req.body) res.redirect("back")  }; this.delVideo = function (req, res) { var cname = req.body.course; var videoName = req.body.name; console.log(req.body)    course.update({  "name": cname  }, {  $pull: {  "material.vids": {  "name": videoName  } |
| --- |

| }  })  .exec(function (err, reslt) { console.log("delete this") res.redirect("back")  })  }; this.addDocs = function (req, res) {  var cid = req.params.id; var week = req.body.week;  course.findById(cid, function (err, course\_info) { if (course\_info == null) course\_info = "no" res.render("addDocs", { course\_info  })  }); }; this.delDocument = function (req, res) { var cname = req.body.course; var videoName = req.body.name; console.log(req.body)    course.update({  "name": cname  }, {  $pull: {  "material.docs": {  "name": videoName  }  }  })  .exec(function (err, reslt) { console.log("delete this") res.redirect("back")  })  };  this.addCover = function (req, res) { res.render("addCover")  } this.saveCover = function (req, res) { res.redirect("back")  } |
| --- |

| this.addCourse = function (req, res) { var crs = new course(); console.log(req.body); crs.name = req.body.course; crs.material.vids = []; crs.material.docs = []; crs.about = req.body.about; crs.prerequisite = req.body.prerequisite; crs.length = req.body.length; crs.effort = req.body.effort; crs.subject = req.body.subject; crs.level = req.body.level; crs.language = req.body.language; crs.coverurl = "t" crs.save(function (err) { res.redirect("back");  });  }; this.showgraph = function (req, res) {  Chartdatas.find({})  .exec(function (err, chart\_data) { res.render("graph", { chart\_data  })  })  }  this.addgraphdata = function (req, res) {  var chartdata = new Chartdatas(); console.log(req.body)  chartdata.label = req.body.label chartdata.data = req.body.data; chartdata.save(function (err) { res.redirect("back")  }); }  this.deletegraph = function (req, res) {  var label\_id = req.body.id;    Chartdatas.findById(label\_id).remove().exec(function (err) { |
| --- |

| res.redirect("back")  });  }; this.enroll = function (req, res) {  var cid = req.body.course; var cname = req.body.coursename;  var enroll = new Enrolls(); var chart = new Chartdatas();  Enrolls.findOne({  "username": req.user.name  })  .exec(function (err, enroll\_info) { if (enroll\_info == null) { save(); addtograph();  } else {  Enrolls.update({  "username": req.user.name  }, {  $push: {  "course": cid  }  })  .exec(function (err, result) { addtograph(); res.redirect('/');  })  }  });  function save() {  enroll.username = req.user.name; enroll.course = cid; enroll.save(function (err) { res.redirect('/')  }) }  function addtograph() {    Chartdatas.findOne({  "label": cname  })  .exec(function (err, result) { if (result == null) { chart.label = cname; |
| --- |
| chart.data = 1; chart.save();  } else {  Chartdatas.update({  "label": cname  }, {  $inc: {  "data": 1  }  })  .exec();  }  })  }  };  }; module.exports = ClickHandler; |

15: Using node is create a Recipe Book

Recipes.js

| var db = require('../utilities/SQL'); var Authentication = require('../utilities/Authentication');  module.exports = function(app) {    // GET /api/recipes app.get('/api/recipes', Authentication.BasicAuthentication, function(request, response, next){  db.query('SELECT \* FROM `recipes`', function (error, results, fields)  { if(error) { response.status(500).send({ error: 'Error getting data' });  } else { var data = []; results.forEach(function(item, index) { data.push({  'id': item['id'],  'name': item['name']  }) }); response.json(data);  }  });    });    // GET /api/recipes app.get('/api/user/recipes/:id', Authentication.BasicAuthentication, function(request, response){  db.query('SELECT \* FROM `recipes` WHERE `user\_id` = ?', [request.params.id], function (error, results, fields) { if(error) {  response.status(500).send({ error: 'Error getting data' });  } else { var data = []; results.forEach(function(item, index) { data.push({ |
| --- |

| 'id': item['id'],  'name': item['name']  })  }); response.json(data);  }  });    });    // GET /api/recipes/:id app.get('/api/recipes/:id', function(request, response){ db.query('SELECT \* FROM `recipes` WHERE `id` = ?', [request.params.id], function (error, results, fields) { if(error) { response.status(500).send({ error: 'Error getting data' });  } else { response.json({ 'id': results[0]['id'], 'name':  results[0]['name'] });  }  });  });    // POST /api/recipes/:id app.post('/api/recipes/:id', function(request, response){ db.query('INSERT INTO `recipes` SET ?', { 'user\_id':  request.params.id, 'name': request.body.name }, function (error, result, fields) { if(error) { response.status(500).send({ error: 'Error adding data' });  } else { response.json({  'id': result.insertId,  'name': request.body.name  })  }    });  });    // PUT /api/recipes/:id  app.put('/api/recipes/:id', function(request, response){ db.query('UPDATE `recipes` SET name = ? WHERE id = ?',  [request.body.name, request.params.id], function (error, result, fields) { if(error) { response.status(500).send({ error: 'Error updating data' });  } else { response.json({ |
| --- |
| 'id': request.params.id,  'name': request.body.name  });  }  });  });    // DELETE /api/recipes/:id app.delete('/api/recipes/:id', function(request, response){ db.query('DELETE FROM `recipes` WHERE `id` = ?; DELETE FROM `ingredients` WHERE `recipe\_id` = ?; DELETE FROM `directions` WHERE  `recipe\_id` = ?', [request.params.id, request.params.id, request.params.id], function (error, results, fields) { if(error) { response.status(500).send({ error: 'Error deleting data' });  } else { response.json({});  }  });  });  } |

Server.js

| var express = require('express'); var hbs = require('hbs'); var bodyParser = require('body-parser'); var cookieParser = require('cookie-parser'); var methodOverride = require('method-override'); var errorHandler = require('errorhandler'); var http = require('http'); var path = require('path'); var Middleware = require('./utilities/Middleware'); var app = express(); app.set('port', 8080);    app.set('view engine', 'html'); app.engine('html', hbs.\_\_express);    /\* cookie-parser - https://github.com/expressjs/cookie-parser  Parse Cookie header and populate req.cookies with an object keyed by the cookie names. \*/ app.use(cookieParser('SECRETCOOKIEKEY123'));    /\* body-parser - https://github.com/expressjs/body-parser  Node.js body parsing middleware. \*/ |
| --- |
| app.use(bodyParser.json()); app.use(bodyParser.urlencoded({ extended: true }));  /\* method-override - https://github.com/expressjs/method-override  Lets you use HTTP verbs such as PUT or DELETE in places where the client doesn't support it. \*/ app.use(methodOverride());    /\* errorhandler - https://github.com/expressjs/errorhandler Show errors in development. \*/ app.use(errorHandler({ dumpExceptions: true, showStack: true }));  app.use(express.static(path.join(\_\_dirname, '')));  app.use(Middleware.AppendPageInfo);    // send app to router require('./router')(app);  http.createServer(app).listen(app.get('port'), function(){ console.log('Express server listening on port ' + app.get('port')); }); |

Router.js

| var recipes = require('./api/recipes'); var users = require('./api/users'); var ingredients = require('./api/ingredients'); var directions = require('./api/directions');  module.exports = function(app){    // index.html app.get('/', function(request, response){ response.render('index', { });  }); users(app); recipes(app); ingredients(app); directions(app);    }; |
| --- |

16: write node is script to interact with the filesystem, and serve a web page from a file

| var http = require('http'); var url = require('url'); var fs = require('fs'); http.createServer(function (req, res) { var pathname = url.parse(req.url, true).pathname; console.log("Request for" + pathname + "received."); fs.readFile(pathname.substr(1), function (err, data) { if (err) { console.log(err); res.writeHead(404, { 'content-type': 'text/html' });  res.end('<html><body><h1>404 Not found</h1></body></html>');  } else { res.writeHead(200, { 'content-type': 'text/html' }); res.write(data);  res.end();  }  });    }).listen(9030); console.log('server is running on port 8080'); |
| --- |

index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Sample Page</title>

</head>

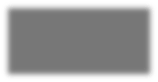
<body>

Hello World! Welcome to web module.

</body>

</html>

Out Put:



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.

var http = require('http'); var dateTime = require('node-datetime'); var dt = dateTime.create(); var formatted = dt.format('Y-m-d H:M:S'); http.createServer(function (req, res) { res.writeHead(200, {'Content-Type': 'text/html'}); res.write("The date and time are currently: " +formatted); res.end();

}).listen(8080);

Out Put:

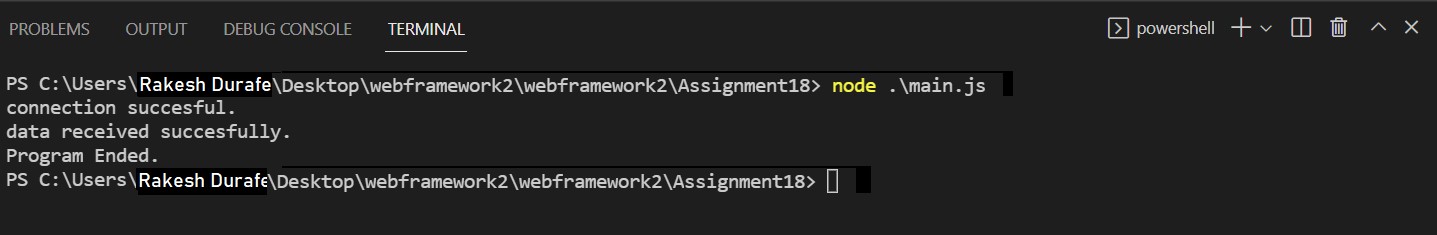


18. Create a node js file named main.js for event-driven application. There should be a main loop that listens for events, and then triggers a call back function when one of those events is detected.

Main.js

| // Import events module  var events = require('events');    // Create an eventEmitter object var eventEmitter = new events.EventEmitter();    // Create an event handler as follows var connectHandler = function connected() { console.log('connection succesful.');    // Fire the data\_received event eventEmitter.emit('data\_received');  }    // Bind the connection event with the handler eventEmitter.on('connection', connectHandler);    // Bind the data\_received event with the anonymous function eventEmitter.on('data\_received', function() { console.log('data received succesfully.');  });    // Fire the connection event eventEmitter.emit('connection');  console.log("Program Ended."); |
| --- |

Out Put:



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(); var PORT = 3000; window = {}; 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);  }); |
| --- |

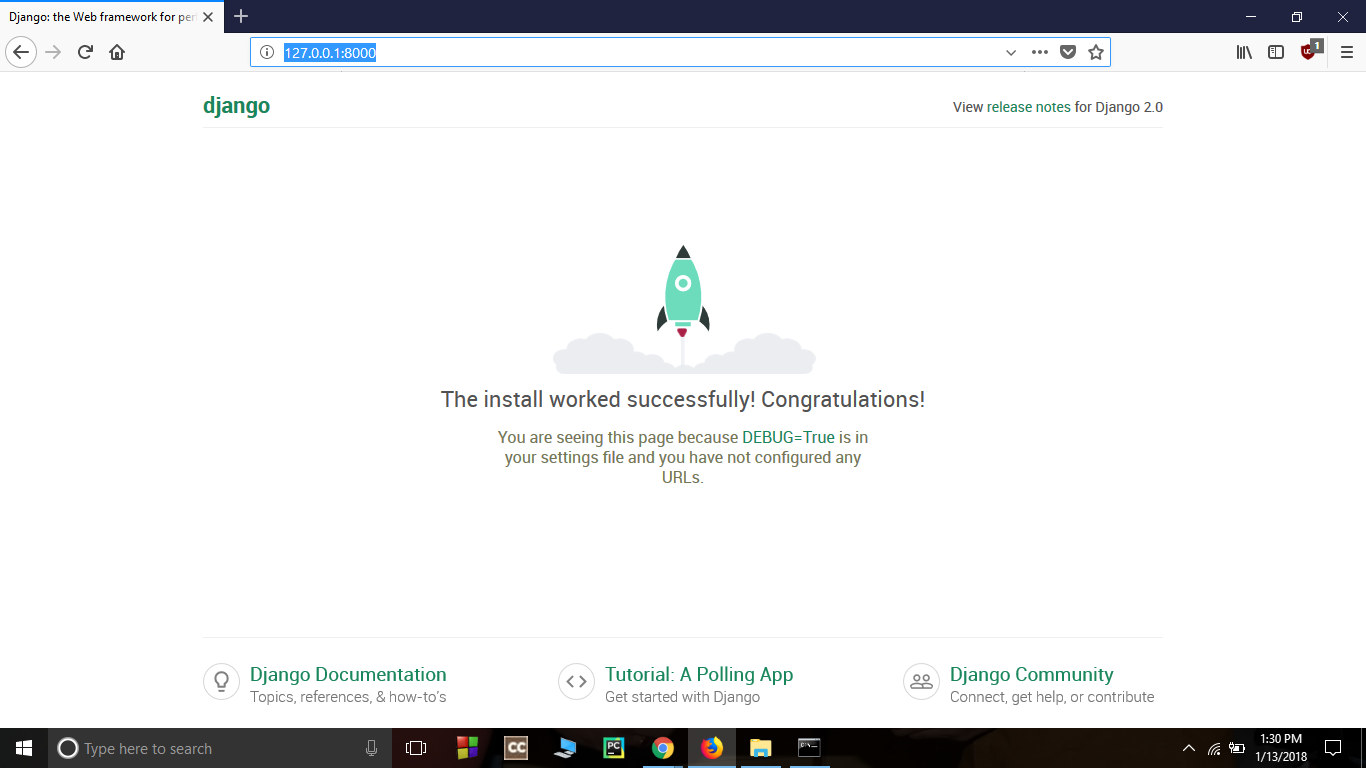
20. Create your Django app in which after running the server, you should see on the

Browser, the text “Hello! I am learning Django”, which you defined in the index view.

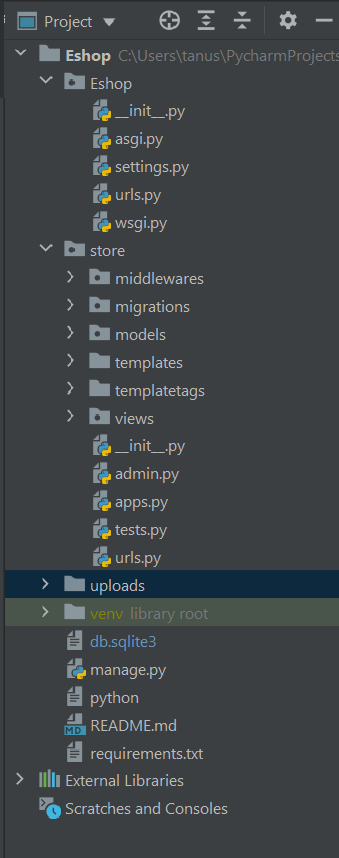
<https://www.youtube.com/watch?v=EOuOVYSJ4J0>

31. Implement your E-commerce Website using Django

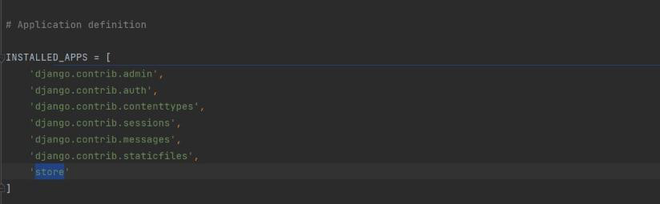
1. Create Normal Project: Open the IDE and create a normal project by selecting File -> New Project.
2. Install Django: Next, we will install the Django module from the terminal. We will use PyCharm integrated terminal to do this task. One can also use cmd on windows to install
3. the module by running python -m pip install django command
4. Check Installed Django version: To check the installed Django version, you can run the python -m django -version command as shown below.
5. Create Django Project: When we execute django-admin startproject command, then it will create a Django project inside the normal project which we already have created here. django-admin startproject ProjectName.
6. Check Python3 version: python3 –version
7. Run Default Django webserver: - Django internally provides a default webserver where we can launch our applications. Python manage.py runserver command in terminal. By default, the server runs on port 8000. Access the webserver at the highlighted URL.



Open the project folder using a text editor. The directory structure should look like this:



Now add store app in E-commerce website in **settings.py**.



urls.py

from django.contrib import admin

from django.urls import path, include

from django.conf.urls.static import static

from . import settings

urlpatterns = [

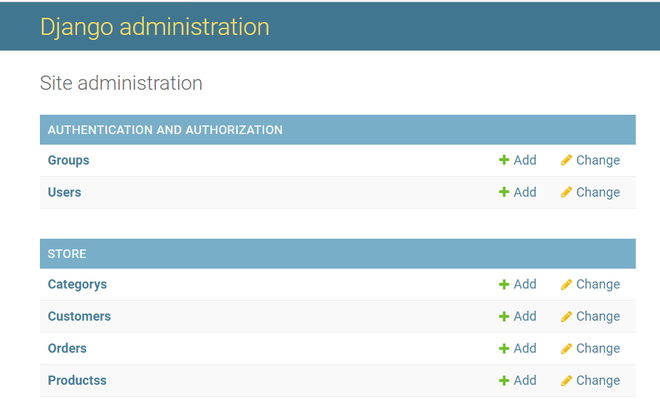
path('admin/', admin.site.urls),

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

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

[Models](https://www.geeksforgeeks.org/django-models/)

The below screenshot shows the required models that we will need to create. These models are tables that will be stored in the SQLite database.



Let’s see each model and the fields required by each model.

category.py

from django.db import models

class Category(models.Model):

name = models.CharField(max\_length=50)

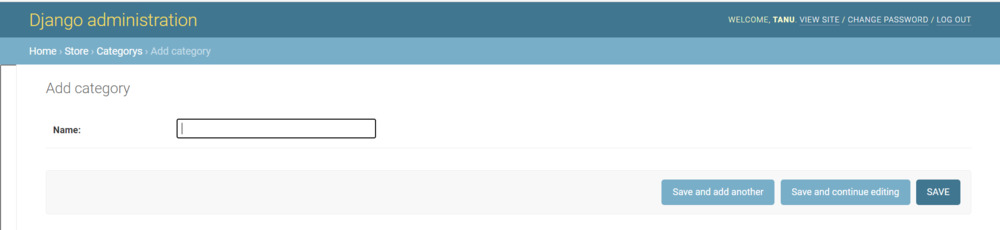
@staticmethod

def get\_all\_categories():

return Category.objects.all()

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

return self.name



customer.py

from django.db import models

class Customer(models.Model):

first\_name = models.CharField(max\_length=50)

last\_name = models.CharField(max\_length=50)

phone = models.CharField(max\_length=10)

email = models.EmailField()

password = models.CharField(max\_length=100)

# to save the data

def register(self):

self.save()

@staticmethod

def get\_customer\_by\_email(email):

try:

return Customer.objects.get(email=email)

except:

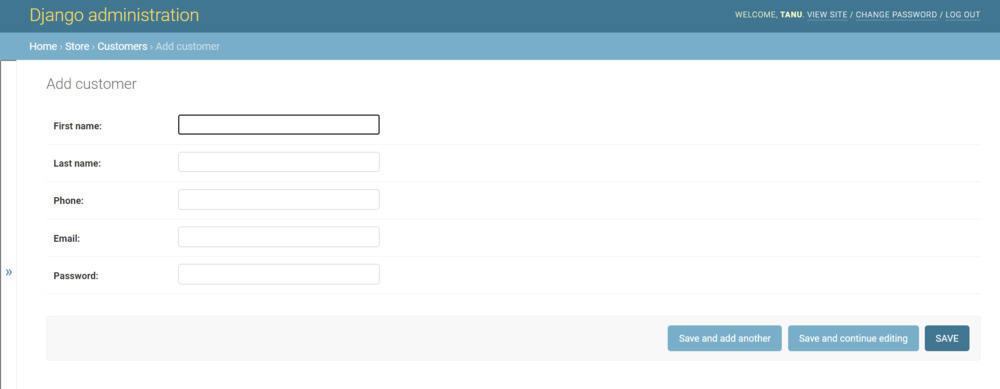
return False

def isExists(self):

if Customer.objects.filter(email=self.email):

return True

return False



products.py

from django.db import models

from .category import Category

class Products(models.Model):

name = models.CharField(max\_length=60)

price = models.IntegerField(default=0)

category = models.ForeignKey(Category, on\_delete=models.CASCADE, default=1)

description = models.CharField(

max\_length=250, default='', blank=True, null=True)

image = models.ImageField(upload\_to='uploads/products/')

@staticmethod

def get\_products\_by\_id(ids):

return Products.objects.filter(id\_\_in=ids)

@staticmethod

def get\_all\_products():

return Products.objects.all()

@staticmethod

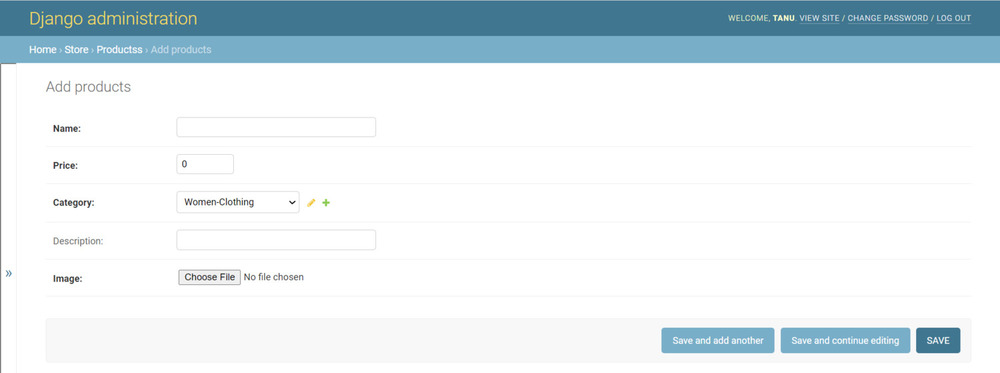
def get\_all\_products\_by\_categoryid(category\_id):

if category\_id:

return Products.objects.filter(category=category\_id)

else:

return Products.get\_all\_products()



Orders.py

from django.db import models

from .product import Products

from .customer import Customer

import datetime

class Order(models.Model):

product = models.ForeignKey(Products,

on\_delete=models.CASCADE)

customer = models.ForeignKey(Customer,

on\_delete=models.CASCADE)

quantity = models.IntegerField(default=1)

price = models.IntegerField()

address = models.CharField(max\_length=50, default='', blank=True)

phone = models.CharField(max\_length=50, default='', blank=True)

date = models.DateField(default=datetime.datetime.today)

status = models.BooleanField(default=False)

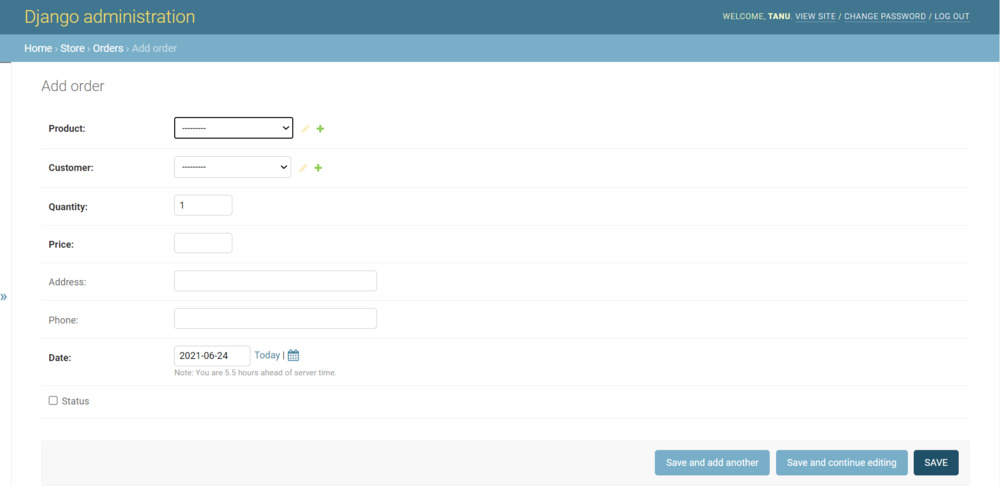
def placeOrder(self):

self.save()

@staticmethod

def get\_orders\_by\_customer(customer\_id):

return Order.objects.filter(customer=customer\_id).order\_by('-date')



[Views](https://www.geeksforgeeks.org/views-in-django-python/) :

In views, we create a view named home.py, login.py, signup.py, cart.py, checkout.py, orders.py  which takes a request and renders an HTML as a response. Create an home.html, login.html, signup.html, cart.html, checkout.html, orders.html in the templates. And map the views to the store\urls.py folder.

Urls.pyfrom django.contrib import admin

from django.urls import path

from .views.home import Index, store

from .views.signup import Signup

from .views.login import Login, logout

from .views.cart import Cart

from .views.checkout import CheckOut

from .views.orders import OrderView

from .middlewares.auth import auth\_middleware

urlpatterns = [

path('', Index.as\_view(), name='homepage'),

path('store', store, name='store'),

path('signup', Signup.as\_view(), name='signup'),

path('login', Login.as\_view(), name='login'),

path('logout', logout, name='logout'),

path('cart', auth\_middleware(Cart.as\_view()), name='cart'),

path('check-out', CheckOut.as\_view(), name='checkout'),

path('orders', auth\_middleware(OrderView.as\_view()), name='orders'),

]

home.py

from django.shortcuts import render, redirect, HttpResponseRedirect

from store.models.product import Products

from store.models.category import Category

from django.views import View

# Create your views here.

class Index(View):

def post(self, request):

product = request.POST.get('product')

remove = request.POST.get('remove')

cart = request.session.get('cart')

if cart:

quantity = cart.get(product)

if quantity:

if remove:

if quantity <= 1:

cart.pop(product)

else:

cart[product] = quantity-1

else:

cart[product] = quantity+1

else:

cart[product] = 1

else:

cart = {}

cart[product] = 1

request.session['cart'] = cart

print('cart', request.session['cart'])

return redirect('homepage')

def get(self, request):

# print()

return HttpResponseRedirect(f'/store{request.get\_full\_path()[1:]}')

def store(request):

cart = request.session.get('cart')

if not cart:

request.session['cart'] = {}

products = None

categories = Category.get\_all\_categories()

categoryID = request.GET.get('category')

if categoryID:

products = Products.get\_all\_products\_by\_categoryid(categoryID)

else:

products = Products.get\_all\_products()

data = {}

data['products'] = products

data['categories'] = categories

print('you are : ', request.session.get('email'))

return render(request, 'index.html', data)

login.py

from django.shortcuts import render, redirect, HttpResponseRedirect

from django.contrib.auth.hashers import check\_password

from store.models.customer import Customer

from django.views import View

class Login(View):

return\_url = None

def get(self, request):

Login.return\_url = request.GET.get('return\_url')

return render(request, 'login.html')

def post(self, request):

email = request.POST.get('email')

password = request.POST.get('password')

customer = Customer.get\_customer\_by\_email(email)

error\_message = None

if customer:

flag = check\_password(password, customer.password)

if flag:

request.session['customer'] = customer.id

if Login.return\_url:

return HttpResponseRedirect(Login.return\_url)

else:

Login.return\_url = None

return redirect('homepage')

else:

error\_message = 'Invalid !!'

else:

error\_message = 'Invalid !!'

print(email, password)

return render(request, 'login.html', {'error': error\_message})

def logout(request):

request.session.clear()

return redirect('login')

signup.py

from django.shortcuts import render, redirect

from django.contrib.auth.hashers import make\_password

from store.models.customer import Customer

from django.views import View

class Signup (View):

def get(self, request):

return render(request, 'signup.html')

def post(self, request):

postData = request.POST

first\_name = postData.get('firstname')

last\_name = postData.get('lastname')

phone = postData.get('phone')

email = postData.get('email')

password = postData.get('password')

# validation

value = {

'first\_name': first\_name,

'last\_name': last\_name,

'phone': phone,

'email': email

}

error\_message = None

customer = Customer(first\_name=first\_name,

last\_name=last\_name,

phone=phone,

email=email,

password=password)

error\_message = self.validateCustomer(customer)

if not error\_message:

print(first\_name, last\_name, phone, email, password)

customer.password = make\_password(customer.password)

customer.register()

return redirect('homepage')

else:

data = {

'error': error\_message,

'values': value

}

return render(request, 'signup.html', data)

def validateCustomer(self, customer):

error\_message = None

if (not customer.first\_name):

error\_message = "Please Enter your First Name !!"

elif len(customer.first\_name) < 3:

error\_message = 'First Name must be 3 char long or more'

elif not customer.last\_name:

error\_message = 'Please Enter your Last Name'

elif len(customer.last\_name) < 3:

error\_message = 'Last Name must be 3 char long or more'

elif not customer.phone:

error\_message = 'Enter your Phone Number'

elif len(customer.phone) < 10:

error\_message = 'Phone Number must be 10 char Long'

elif len(customer.password) < 5:

error\_message = 'Password must be 5 char long'

elif len(customer.email) < 5:

error\_message = 'Email must be 5 char long'

elif customer.isExists():

error\_message = 'Email Address Already Registered..'

# saving

return error\_message

cart.py

from django.db import models

from .product import Products

from .customer import Customer

import datetime

class Order(models.Model):

product = models.ForeignKey(Products,

on\_delete=models.CASCADE)

customer = models.ForeignKey(Customer,

on\_delete=models.CASCADE)

quantity = models.IntegerField(default=1)

price = models.IntegerField()

address = models.CharField(max\_length=50, default='', blank=True)

phone = models.CharField(max\_length=50, default='', blank=True)

date = models.DateField(default=datetime.datetime.today)

status = models.BooleanField(default=False)

def placeOrder(self):

self.save()

@staticmethod

def get\_orders\_by\_customer(customer\_id):

return Order.objects.filter(customer=customer\_id).order\_by('-date')

checkout.py

from django.shortcuts import render, redirect

from django.contrib.auth.hashers import check\_password

from store.models.customer import Customer

from django.views import View

from store.models.product import Products

from store.models.orders import Order

class CheckOut(View):

def post(self, request):

address = request.POST.get('address')

phone = request.POST.get('phone')

customer = request.session.get('customer')

cart = request.session.get('cart')

products = Products.get\_products\_by\_id(list(cart.keys()))

print(address, phone, customer, cart, products)

for product in products:

print(cart.get(str(product.id)))

order = Order(customer=Customer(id=customer),

product=product,

price=product.price,

address=address,

phone=phone,

quantity=cart.get(str(product.id)))

order.save()

request.session['cart'] = {}

return redirect('cart')

orders.py

from django.shortcuts import render, redirect

from django.contrib.auth.hashers import check\_password

from store.models.customer import Customer

from django.views import View

from store.models.product import Products

from store.models.orders import Order

from store.middlewares.auth import auth\_middleware

class OrderView(View):

def get(self, request):

customer = request.session.get('customer')

orders = Order.get\_orders\_by\_customer(customer)

print(orders)

return render(request, 'orders.html', {'orders': orders})

27. Create your own blog using Django

<https://djangocentral.com/building-a-blog-application-with-django/>

32. Implement Login System using Django

<https://ordinarycoders.com/blog/article/django-user-register-login-logout>