Name : Kshirsagar Devanshi Kiran

Roll No : 14

304 – OSWD Practical Assignment 2

Q-1.Develop a user registration form and store its data in any database using Express. Form should also contain file upload (single, multiple) with validations.

➔

Server.js

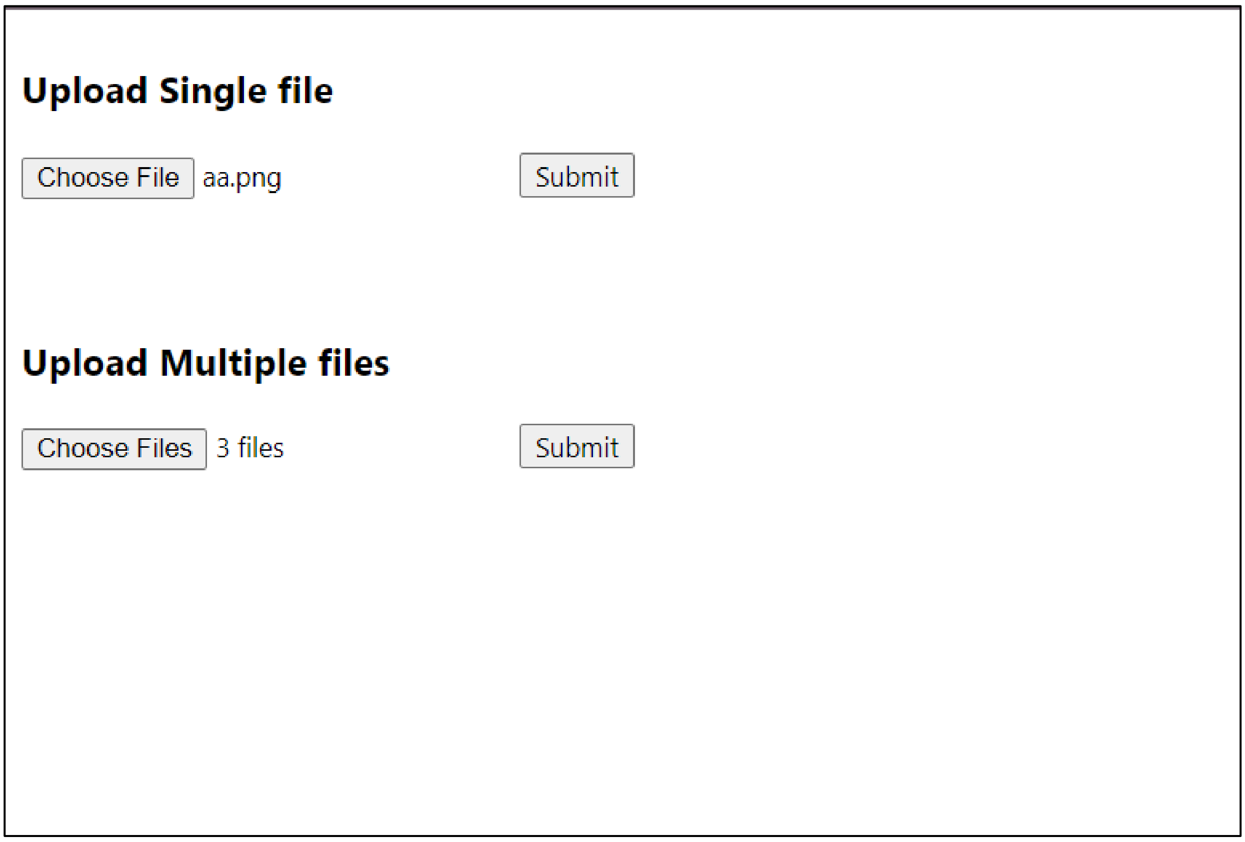
```
const express = require('express')
const app = express();
const multer = require('multer')
var fs = require('fs')
var path = require('path')

var Storage = multer.diskStorage({
    destination: (req, file, callback) => {
        const destFolder = './uploads'
        //check if folder exist or not
        if (fs.existsSync(destFolder)) {
            callback(null, destFolder)
        } else {
            fs.mkdir(destFolder, (err) => {
                err ? console.error(err.stack) : callback(null, destFolder)
            })
        }
    },
    filename: (req, file, callback) => {
        callback(null,
`${file.fieldname}_${Date.now()}${path.extname(file.originalname)}`)
    }
})
const acceptedTypes = ["image/jpeg", "image/jpg", "image/png", "image/svg",
"image/gif"]

var upload = multer({
    storage: Storage, fileFilter: (req, file, callback) => {
        if (acceptedTypes.includes(file.mimetype)) {
            callback(null, true)
        } else {
            callback(null, false)
            return callback(`only ${acceptedTypes.toString(',')} format allowed`)
        }
    }
})
app.post('/upload_single', upload.single('userFile'), (req, res) => {
    return res.send('file is uploaded');
})
```

```
    app.post('/upload_multiple', upload.array('userFiles', 4), (req, res) => {
        return res.send('files uploaded successfully.');
    })

    app.get('/', (req, res) => {
        return res.sendFile(__dirname + '/views/index.html');
    })

    app.use(function (err, req, res, next) {
        if (err instanceof multer.MulterError) {
            console.log("ERRRR");
            res.status(500).send("file upload  err " + err.message);
        }
        else
            next(err);
    });
    app.listen(8000, () => {
        console.log(`App listening`)
    })
```

Index.html

```
<!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>Upload single/multiple image</title>
    <style>
        * {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif
        }

        .container {
            display: flex;
            flex-direction: column;
            justify-content: space-between;
            height: 30vh;
        }
    </style>
</head>

<body>
    <div class="container">
        <div class="single-upload">
            <div class="header-single">
                <h3>Upload Single file</h3>
            </div>
            <form action="/upload_single" method="post" enctype="multipart/form-data">
                <input type="file" name="userFile" id="">
                <input type="submit" value="Submit">
            </form>
        </div>

        <div class="multi-upload">
            <div class="header-single">
                <h3>Upload Multiple files</h3>
            </div>
            <form action="/upload_multiple" method="post" enctype="multipart/form-data">
                <input type="file" name="userFiles" id="" multiple="multiple">
                <input type="submit" value="Submit">
            </form>
        </div>
    </div>
</body>
</html>
```

Output



Q-2. Express Login application with file session store.
➔

Server.js

```
const express = require('express')
const app = express()
const session = require('express-session')
const path = require('path')
const FileStore = require('session-file-store')(session)

app.use(express.static('public'))
app.use(express.urlencoded({ extended: false }))

app.use(session({
    secret: 'secret',
    resave: false,
    saveUninitialized: false,
    store: new FileStore({ path: './session-data' })
}))

// Define a list of valid users with their passwords
const validUsers = [
    { username: 'Devanshi', password: 'dev1234' },
    { username: 'Henisha', password: '111' }
];

app.get('/', (req, res, next) => {
    res.sendFile(__dirname + '/views/login.html')
})

app.post('/login', (req, res) => {
    const username = req.body.username;
    const password = req.body.password;

    // Check if provided username and password match any valid user
    const validUser = validUsers.find(user => user.username === username &&
user.password === password);

    if (validUser) {
        // Valid credentials, store data in session
        req.session.loggedIn = true;
        req.session.username = username;
        return res.redirect('/home');
    } else {
        return res.send('Invalid Username or Password!');
    }
})

app.get('/home', (req, res) => {
    if (req.session.loggedIn) {
            console.log('logged in')
            res.sendFile(__dirname + '/views/home.html')
        } else {
            console.log('not logged in')
            res.sendFile(__dirname + '/views/login.html')
        }
    })

    app.get('/logout', (req, res) => {
        req.session.destroy();
        res.redirect('/');
    });
    app.listen(3000, () => {
        console.log(`server listening`)
    })
```

login.html

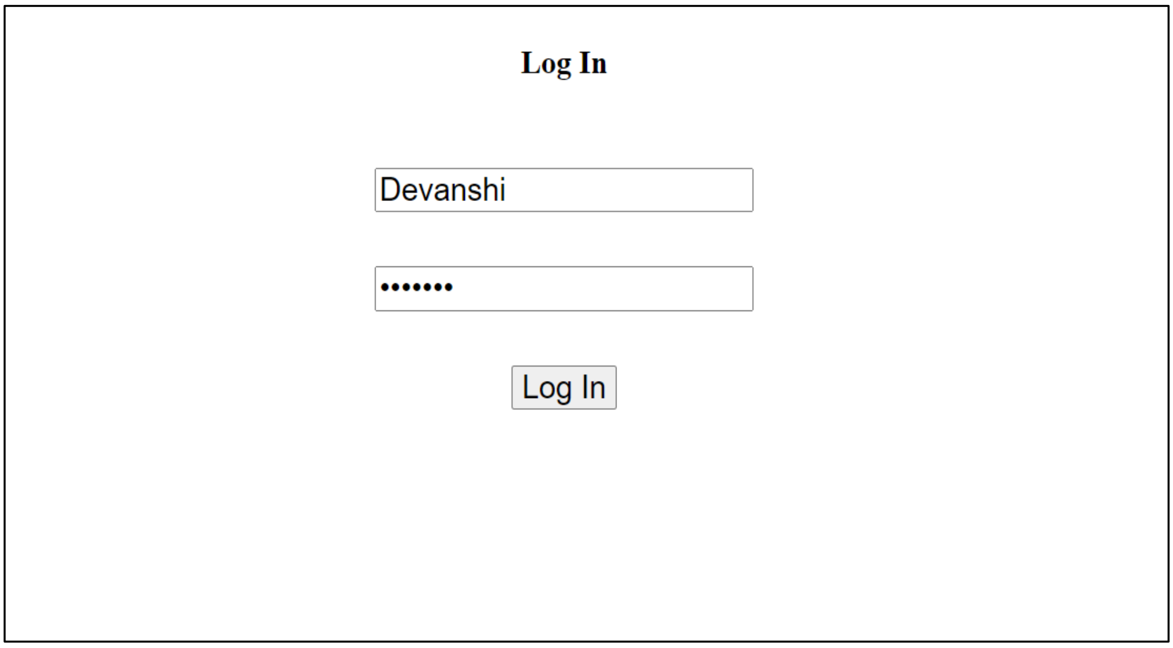
```
<!DOCTYPE html>
<html lang="en">

<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>Log In Page</title>
    <style>
        .mb-2 {
            margin-bottom: 2rem;
        }

        .mt-2 {
            margin-top: 2rem;
        }

        .flex-center {
            display: flex;
            justify-content: center;
            flex-direction: column;
            align-items: center;
        }

        * {
            font-size: 1.2rem;
        }
    </style>
</head>

<body>
    <div class="container flex-center">
```

```html
        <div class="header">
            <h3>Log In</h3>
        </div>

        <div class="form-container mt-2 ">
            <form class="flex-center" action="/login" method="post">
                <div class="username mb-2">
                    <input type="text" placeholder="enter username" name="username" required>
                </div>
                <div class="password mb-2">
                    <input type="password" name="password" placeholder="enter password" id="" required>
                </div>
                <div class="submit">
                    <input type="submit" value="Log In">
                </div>
            </form>
        </div>
    </div>
</body>

</html>
```

home.html

```html
<!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>Home Page</title>
</head>

<body>
    <div class="container">
        <h2>Home Page, You Are Authentic User</h2>
        <a href="/logout">Logout</a>
    </div>
</body>

</html>
```

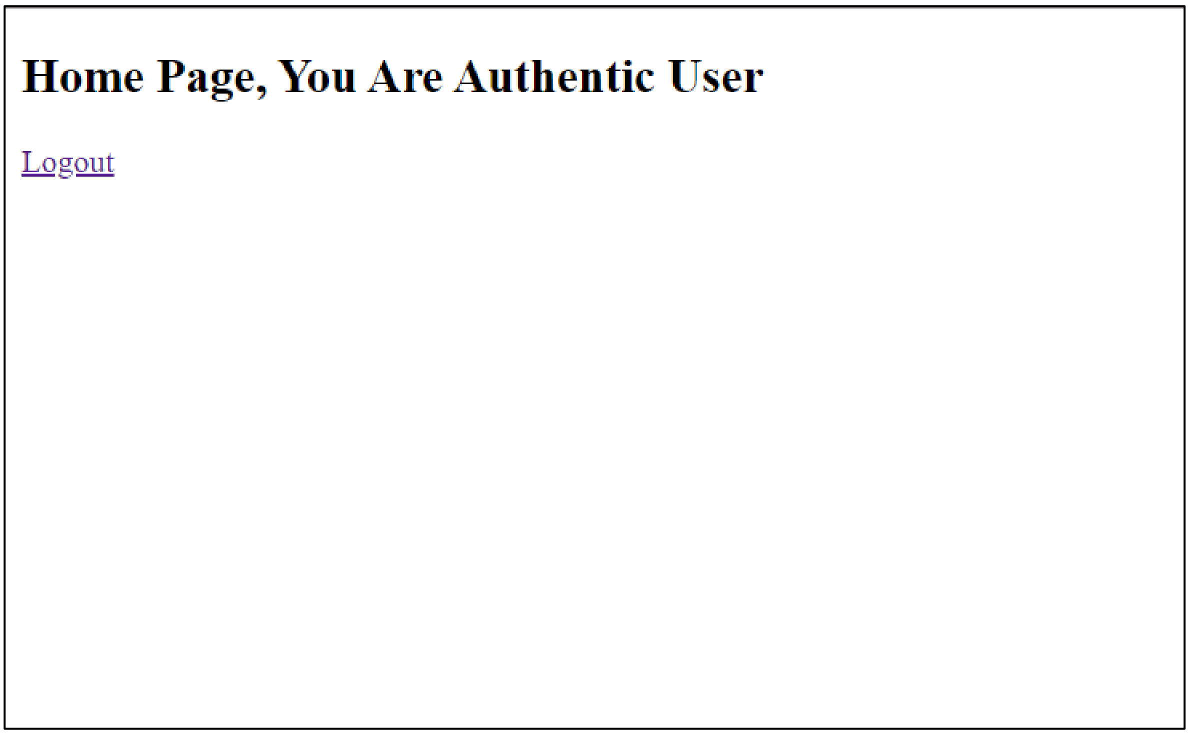
Output


Log In

Devanshi

•••••••

Log In


Home Page, You Are Authentic User

Logout

Q-4. Login with JWT, CRUD operations for students table with mongoose, express.

➔

Index.js

```
const express = require('express')
const app = express()
const PORT = 8000
const database = require('./db/database')
const student = require('./routes/student')
require('dotenv').config()
const jwt = require('jsonwebtoken')
const cookieParser = require('cookie-parser')
const accessTokenSecret = process.env.TOKEN_SECRET;
const studentModel = require('./models/student')
const verifyToken = require('./middleware/token')

app.use(cookieParser())
app.set('view engine', 'ejs')
app.use(express.json())
app.use(express.urlencoded({ extended: true }))
app.use(express.static(__dirname + '/public'));

app.use('/student', student)

app.get('/', (req, res) => {
    const token = req.cookies.token
    if (token) {
        //verify token
        if (jwt.verify(token, process.env.TOKEN_SECRET)) {
            return res.redirect('/student')
        } else {
            return res.status(401).end('Unauthorized access');
        }
    }
    else {
        //redirect to login
        return res.render('login')
    }
})

app.post('/login', async (req, res) => {
    let { username, password } = req.body
    if (username && password) {
        try {
            let result = await studentModel.find({ name: username, password:
password })
            if (result.length === 1) {
                //Generate Token
                const accessToken = jwt.sign({ name: username, password: password },
accessTokenSecret)
                res.cookie('token', accessToken, { httpOnly: true });
                return res.redirect('/student')
            } else {
                res.render('login', { errors: { validationError: 'Enter Valid
Username Or Password' } })
            }
        } catch (error) {
            res.render('login', { errors: { validationError: 'There Was Problem
While Log In.' } })
        }
    } else {
        res.render('login', { errors: { AllFieldsError: 'Please Enter Username And
Password' } })
    }
})
app.get('/logout', (req, res) => {
    res.clearCookie('token')
    res.render('login')
})
app.listen(PORT, () => {
    console.log(`app listening on http://localhost:${PORT}`)
})
```

Database.js

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

    //database config
    const server = "127.0.0.1:27017"
    const Database = "studentDB"

    class database {
        constructor() {
            this._connect()
        }
        async _connect() {
            try {
                await mongoose.connect(`mongodb://${server}/${Database}`)
                console.log(`database connection successfull`)
            } catch (error) {
                console.error('Database connection error: ' + error);
            }
        }
    }
    module.exports = new database();
```

Token.js

```
const jwt = require('jsonwebtoken')

const signToken = async (req, res) => {
    //TODO: implement sign token logic here
}

const verifyToken = async (req, res, next) => {
    const token = req?.cookies?.token
    if (token) {
        //verify token
        if (jwt.verify(token, process.env.TOKEN_SECRET)) {
            // return res.send('token found and verified');
            next()
        } else {
            return res.status(401).end('Unauthorized access');
        }
    }
    else {
        //redirect to login
        return res.status(401).send('Token Missing')
    }
}

module.exports = { signToken, verifyToken }
```

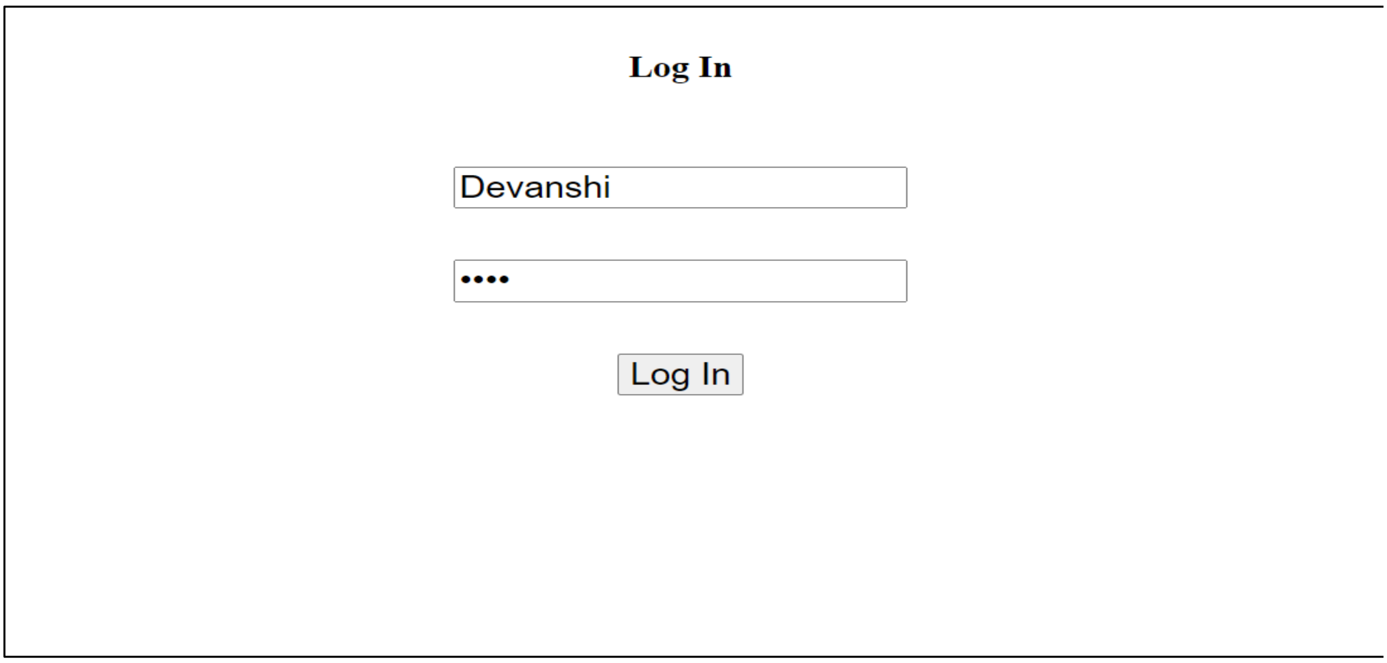
Student.js(Model)

```
const mongoose = require('mongoose')

const studentSchema = new mongoose.Schema({
    name: String,
    email: String,
    password: String,
    age: Number,
    gender: String,
    city: String
})
module.exports = new mongoose.model('student', studentSchema, 'students')
```

Student.js(Routes)

```
const express = require('express');
const { verifyToken } = require('../middleware/token');
const student = express.Router();
const studentModel = require('../models/student')

student.use(verifyToken)

//create
student.get('/add', (req, res) => {
    res.render('add');
})
student.post('/', async (req, res) => {
    try {
        const newStudent = req.body;
        const student = new studentModel({
            ...newStudent
        })
        await student.save()
        res.redirect('/')
    } catch (error) {
        res.send('There was problem creating students')
    }
})

//read
student.get('/:id', async (req, res) => {
    try {
        const student = await studentModel.findById(req.params.id)
        student ? res.render('view', { data: student }) : res.render('index', {
noDataError: 'No students found' })
    } catch (error) {
        res.send('There was problem getting student')
    }
})

student.get('/', async (req, res) => {
    try {
        const students = await studentModel.find()
        students ? res.render('index', { data: students }) : res.render('index', {
noDataError: 'No students found' })
    } catch (error) {
        res.send('There was problem getting students')
    }
})

//update
student.get('/edit/:id', async (req, res) => {
    try {
        const student = await studentModel.findById(req.params.id)
        student ? res.render('update', { data: student }) : res.render('index', {
noDataError: 'student not found' })
    } catch (error) {
        res.send('There was problem getting student')
    }
})

student.post('/edit/:id', async (req, res) => {
    console.log('student post /:id')
    try {
        const updatedStudent = req.body;
        const id = req.params.id;
        console.log('updated student: ', updatedStudent)
        delete updatedStudent['_id']
        await studentModel.findOneAndUpdate({ _id: id }, updatedStudent)
        return res.redirect('/')
    } catch (error) {
        return res.send('There was problem updating students')
    }
})

//delete
student.get('/delete/:id', async (req, res) => {
    try {
        const student = await studentModel.findById(req.params.id)
        student ? res.render('delete', { data: student }) : res.render('index', {
noDataError: 'student not found' })
    } catch (error) {
        res.send('There was problem getting student')
    }
})
student.post('/delete', async (req, res) => {
    console.log('delete student request')
    try {
        const id = req.body._id
        console.log("id: ", id)
        await studentModel.findOneAndDelete({ _id: id })
        res.redirect('/')
    } catch (error) {
        console.log('Error: ', error)
        res.send('There was problem deleting students')
    }
})

module.exports = student
```

Output

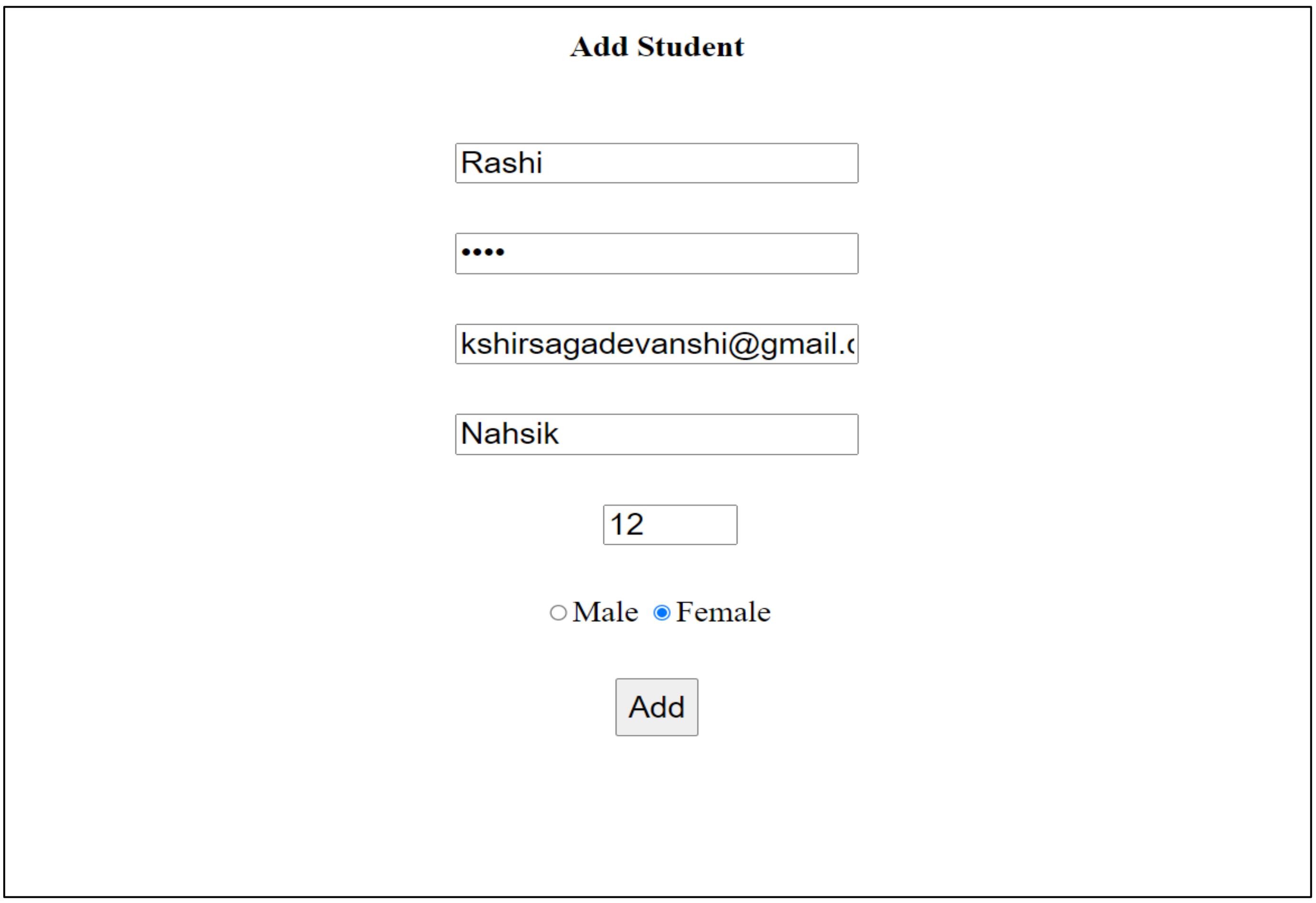


Log In

Devanshi

••••

Log In



Add Student

Rashi

••••

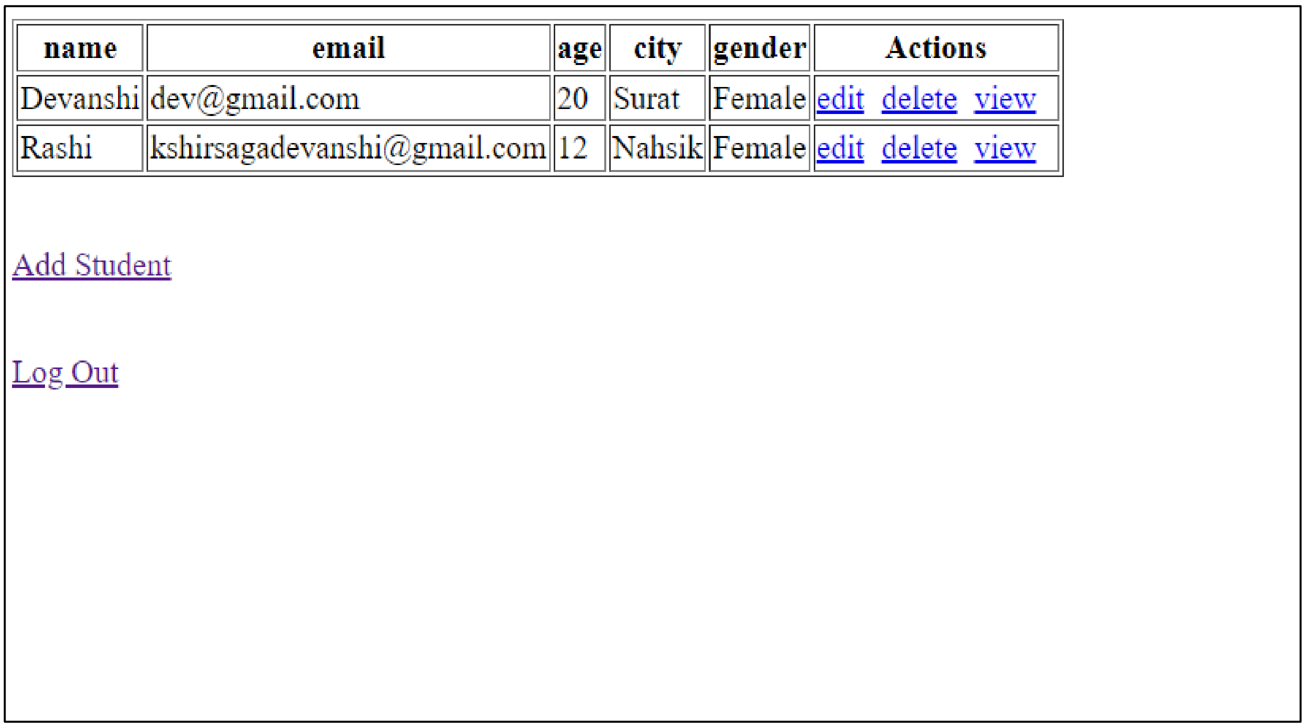
kshirsagadevanshi@gmail.c

Nahsik

12

Male [✓ Female]

Add



name	email	age	city	gender	Actions
Devanshi	dev@gmail.com	20	Surat	Female	edit delete view
Rashi	kshirsagadevanshi@gmail.com	12	Nahsik	Female	edit delete view

Add Student

Log Out

Q-3. Express Login application with redis session store

➔

Server.js

```
const express = require("express");
const session = require("express-session");
const { createClient } = require("redis");
const hbs = require("hbs")
const path = require("path")

const app = express();

// Initialize client.

const redisClient = createClient();
redisClient.connect().then(()=>{
   console.log("redis client is connected");
}).catch((error)=>{
    console.log("not connected ",error)
});

// Initialize store.
const RedisStore = require("connect-redis").default
const redisStore = new RedisStore({
  client: redisClient,
  prefix: "Expressredis:",
});
```

```
// Initialize session storage.
app.use(
  session({
    store: redisStore,
    resave: false,
    saveUninitialized: false,
    secret: "keyboard cat",
  })
);

app.use(express.json())
app.use(express.urlencoded({extended:true}))
app.set("views",path.join(__dirname,"/src/views"))
app.set("view engine",hbs)



const routes = require("./src/Routes/routes")
app.use("/",routes)

// Start the server
app.listen(3000, () => {
  console.log("Server started on port 3000");
});
```

Routes.js

```
const router = require("express").Router()
 router.get("/", (req, res) => {
    res.render("Login.hbs")
  });

  const isLoggedIn = (req, res, next) => {
    if (req.session.userId) {
      next();
    } else {
      res.redirect("/login");
    }
  };
  router.post("/login", (req, res) => {
    try {
        const password = req.body.password;

        req.session.userId = Date.now();

        return res.render("dashboard.hbs");
    } catch (error) {
        console.log(error);
    }
});
```

```
router.get("/logout", (req, res) => {
// Destroy session and redirect to login page
req.session.destroy();
res.redirect("/");
});
module.exports = router
```

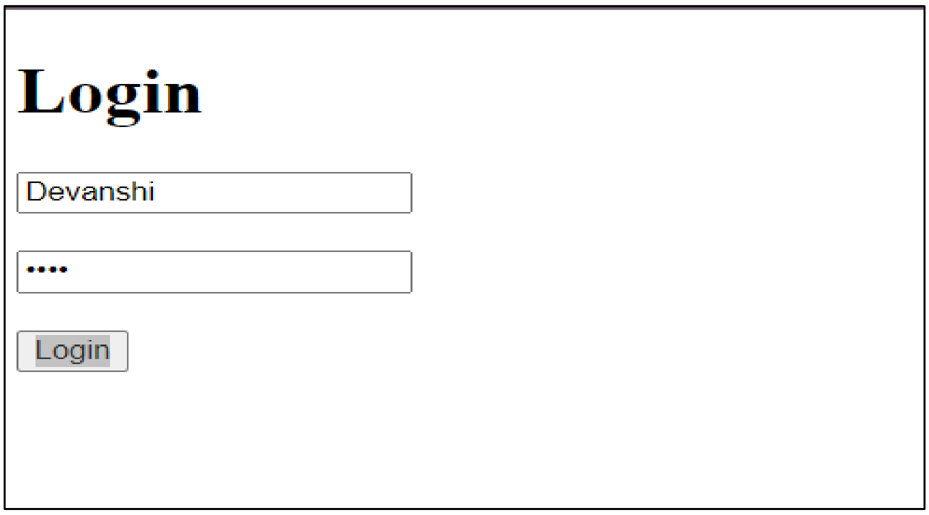
Dashboard.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Dashboard</h1>
      <p>Welcome, User!</p>
      <a href="/logout">Logout</a>
</body>
</html>
```

Login.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
     <h1>Login</h1>
      <form method="POST" action="/login">
        <input type="text" name="username" placeholder="Username" required> <br><br>
        <input type="password" name="password" placeholder="password"
required><br><br>
        <button type="submit">Login</button>
      </form>
</body>
</html>
```

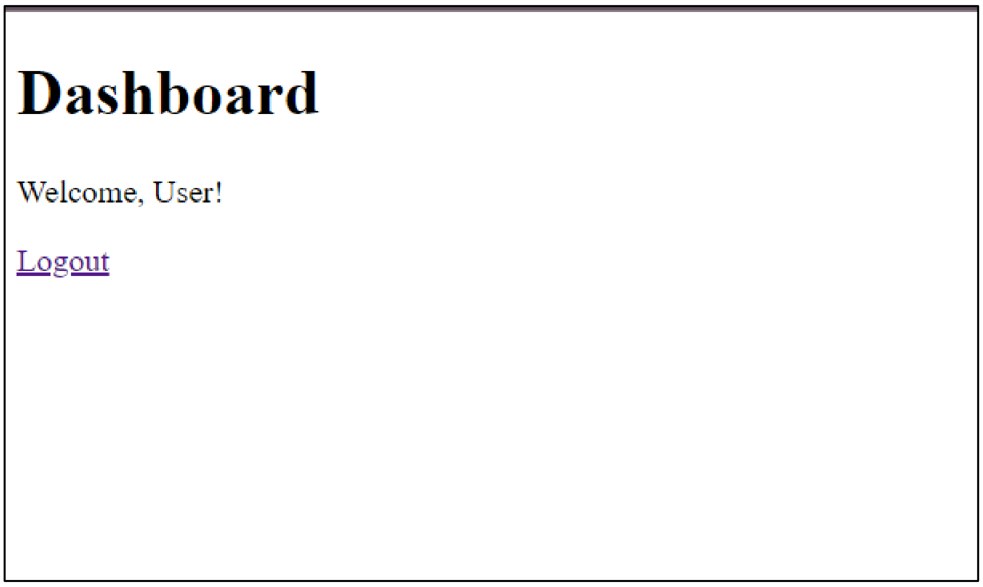
Output



Login

Devanshi

••••



Dashboard

Welcome, User!

Q-5. Login with JWT, CRUD operations for students table with mongoose, express and frontend(html,css,javascript/jquery/angularjs), Logout.

Server.js

```
require("dotenv").config();
const express = require("express");
const session = require("express-session");
const mongoose = require("mongoose");
const app = express();
const FileStore = require('session-file-store')(session)
const PORT = process.env.PORT || 8000;

//Database connection code
mongoose.connect(process.env.DB_URL,{
    useNewUrlParser : true,
    useUnifiedTopology : true

});
const db = mongoose.connection;

db.on("error", (error) =>{console.log(error)});
db.once("open",() =>{console.log("Connecte to databse")});

//middleware
app.use(express.urlencoded({extended:true}));
app.use(express.json());

app.use(session({
    secret : "My secret",
    saveUninitialized : true,
    resave : false,
    store: new FileStore({ path: './session-data' })
}))

app.use((req,res,next) =>{
    res.locals.message = req.session.message;
    delete req.session.message;
    next();
})

app.post("/logout",(req,res)=>{
    res.render("login");
})

app.post("/logout",(req,res)=>{
    res.cookie("token",null,{
        httpOnly: true,
            expires: new Date(Date.now()),
        });
        res.redirect("/login");
    })


    //tamplate engine
    app.set("view engine","ejs");

    //prefix routes
    app.use("",require("./routes/loginres"));

    app.listen(PORT,() =>{
        console.log('Server start at http://localhost:${PORT}')
    });
```

.env

```
PORT = 9000
DB_URl = mongodb://127.0.0.1:27017/studentDB
```

Student.js

```
const mongoose = require("mongoose");
const newUserSchema = new mongoose.Schema({
    name :{
        type : String,
        required : true
    },
    dept :{
        type : String,
        required : true
    },
    roll_no :{
        type : String,
        required : true
    },
    address :{
        type : String,
        required : true
    }
});

module.exports = mongoose.model("user",newUserSchema);
```

Register.js

```
const mongoose = require("mongoose");
const jwt = require("jsonwebtoken");
const newRegisterSchema = new mongoose.Schema({
    name :{
        type : String,
        required : true
    },
    username :{
        type : String,
        required : true
    },
    password :{
        type : String,
        required : true
    },
    address :{
        type : String,
        required : true
    }
});

module.exports = mongoose.model("register",newRegisterSchema);
```

Router.js

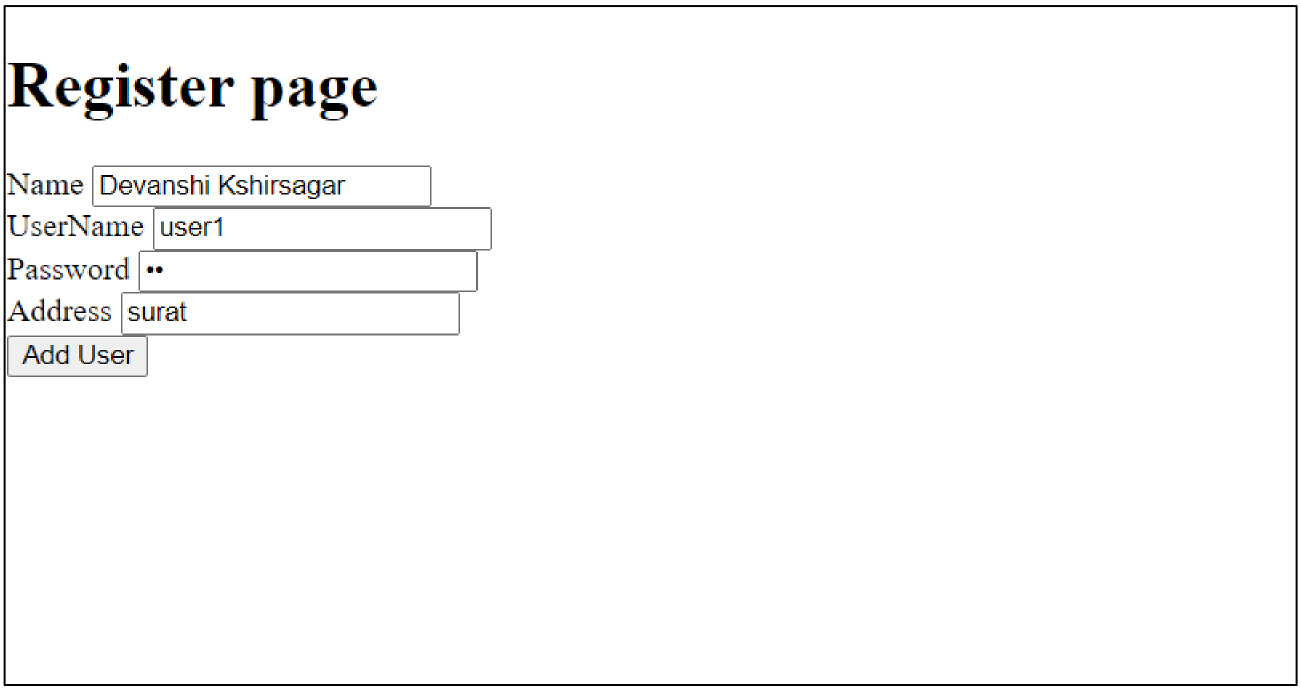
```
const express = require("express");
const router = express.Router();
const User = require("../models/student");

router.get('/index',(req,res) =>{
    User.find().exec((err,user) => {
        if(err){
            console.log("error");
        }else{
            res.render('index',{
               user : user
            })
        }
    })
 })

 router.get("/add",(req,res) =>{
    res.render("add_user");
 });

 router.post("/add",(req,res) => {
    const user = new User({
        name: req.body.name,
        dept : req.body.dept,
        roll_no : req.body.roll_no,
        address : req.body.address
    });
    user.save((err) =>{
        if(err){
            console.log("Error");
        }else{
            console.log("Sucsess");
            res.redirect("index");
        }
    })
 });
 router.get("/edit/:id",(req,res) => {
    let id = req.params.id;
    User.findById(id,(err,user) =>{
        if(err){
            console.log("Erroer");
        }else{
            if(user == null){
                res.redirect("/");
            }else{
                res.render("edit_user",{
                    user : user
                })
            }
        }
    })
 });

 router.post("/update/:id",(req,res) => {
    let id = req.params.id;
    User.findByIdAndUpdate(id,{
        name : req.body.name,
        dept : req.body.dept,
        roll_no : req.body.roll_no,
        address : req.body.address
    }, (err,result) =>{
        if(err){
            console.log("error");
        }else{
            console.log("success");
            res.redirect("index");
        }

    })
 });

 router.get("/delete/:id",(req,res) =>{
    let id = req.params.id;
    User.findByIdAndRemove(id,(err,result) =>{
        if(err){
            console.log("Error");
        }
```

```
else{
            console.log("Success");
            res.redirect("index");
        }
    })
 })
module.exports = router;
```

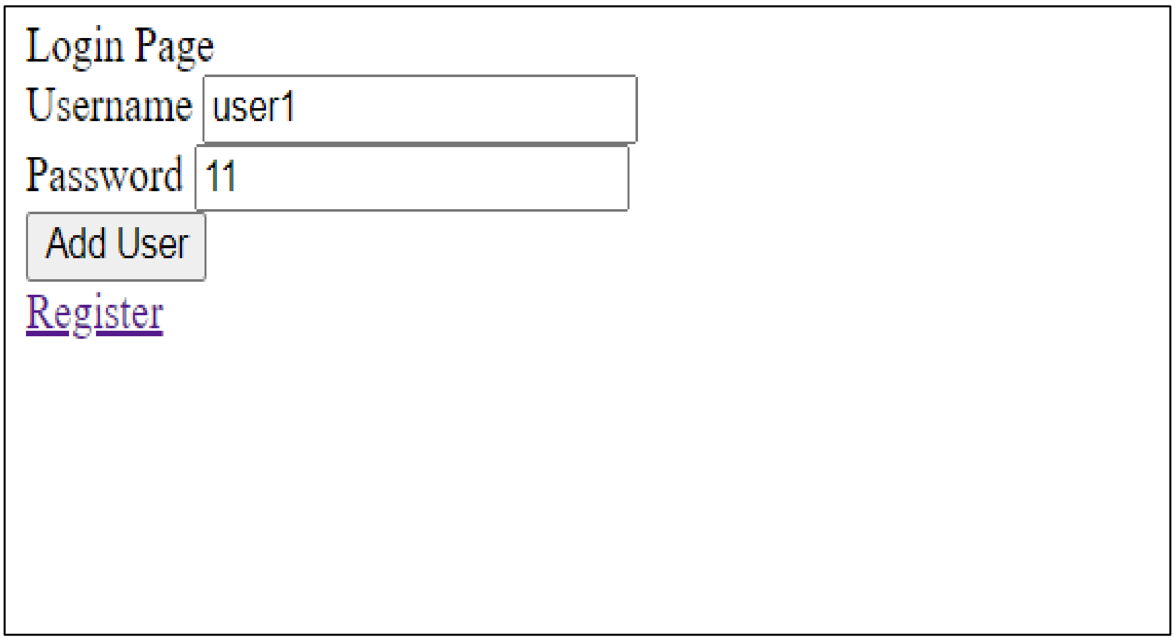
Loginres.js

```
const express = require("express");
const router = express.Router();
const users = require("../models/register");
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
router.get('/',(req,res) =>{
    res.render("login");
 })

// Rendering of buttons Done Now handling post requests
router.post("/login",async(req,res)=>{
    const {username,password}=req.body;
    let user= await users.findOne({username});
    if(!user)
    return res.redirect("/register");
    // Make sure this compare function is await else it will go on the remaining
procedure without actually comparing whether we have the right user or not and it
will login regardless.
    const isUser=await bcrypt.compare(password,user.password);
    if(!isUser){
        // We need to do return res.render to make sure further statements do not
get executed and we redirect from here itself.
       return res.render("login",{message:"Invalid Password"});
    }
    else{
        const token=jwt.sign({_id:user._id},"jsiufbdiufbuibfIU");
        res.cookie("token",token,{
            httpOnly: true,
            expires: new Date(Date.now()+60*1000),
        })
        res.redirect("index");
    }
})


router.get("/register",(req,res) =>{
    res.render("register");
 });
 router.post("/register",async(req,res)=>{
    const {name,username,password,address}=req.body;
    let user= await users.findOne({username});// Inside findOne u need to send an
object
if(user){
        res.redirect("/login");
    }
    console.log(name,username,password,address);
    const hashedPassword= await bcrypt.hash(password,10); // 10 is a salt that we
need to mention
    user=await users.create({
        name,
        username,
        address,
        password: hashedPassword,

    }); // It is an async function

    // This is done to hide the exact id of the document that will be created inside
the users collection everytime a new user registers. We pass an object having key of
_id with value of that particular document's id and we encrypt it using jwt and an
algorithn which is a string in this case.
    const token=jwt.sign({_id:user._id},"jsiufbdiufbuibfIU");
    res.cookie("token",token,{
        httpOnly: true,
        expires: new Date(Date.now()+60*1000),
    })
    res.redirect("/");
})
// Authentication function
const authentication=async(req,res)=>{
    const {token}=req.cookies;
    if(token){
        // Extracting the original ID form the cookie, we decode using the token
        const iD=jwt.verify(token,"jsiufbdiufbuibfIU");
        const user= await users.findById(iD);
        if(user)
           return res.render("logout",{name:user.name});
    }
    return res.redirect("/login");
}

router.post("/logout",(req,res)=>{
    res.cookie("token",null,{
        httpOnly: true,
        expires: new Date(Date.now()),
    });
    res.redirect("/login");
})


router.use("",require("./router"));
module.exports = router;
```

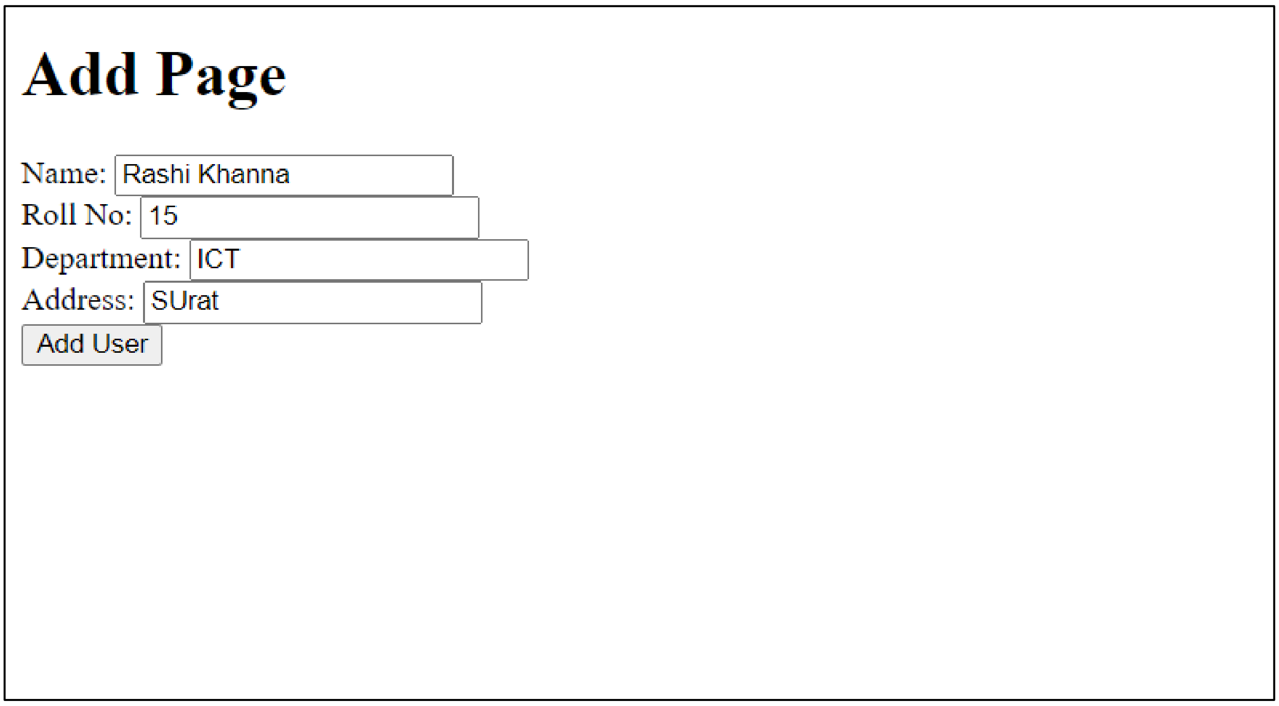
Output


Register page

Name: Devanshi Kshirsagar
UserName: user1
Password: ••
Address: surat

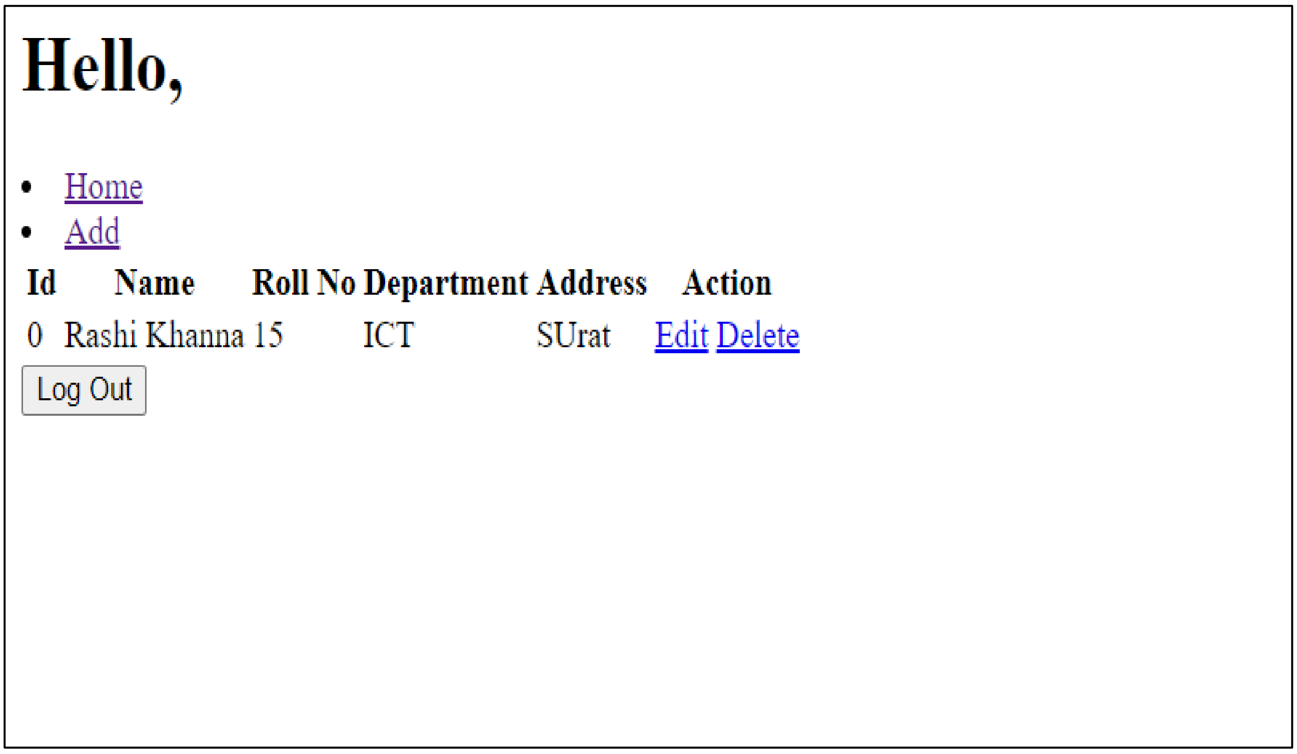

Login Page

Username: user1
Password: 11



Add Page

Name: Rashi Khanna
Roll No: 15
Department: ICT
Address: SUrat



Hello,

- Home
- Add

Id	Name	Roll No	Department	Address	Action
0	Rashi Khanna	15	ICT	SUrat	Edit Delete