

Name:-Parth Patel

Roll No:-48

Assignment:-2

Q1.

Upload.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>

```

Index.html

```

const express = require('express')
const app = express();
const multer = require('multer')
const PORT = 8000;
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/upload.html');
})

app.use(function (err, req, res, next) {
    if (err instanceof multer.MulterError) {
        console.log("ERRRRR");
        res.status(500).send("file upload err " + err.message);
    }
    else
        next(err);
});

```

```
app.listen(PORT, () => {  
  console.log(`app listening on http://localhost:${PORT}`)  
});
```

OutPut:-

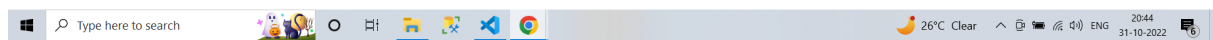


Upload Single file

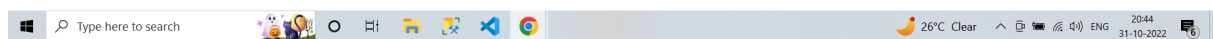
No file chosen

Upload Multiple files

No file chosen



file is uploaded



Q2.

Home.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>
  </div>
</body>

</html>
```

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">
        <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>

```

Session-Data:-

```

{"cookie":{"originalMaxAge":null,"expires":null,"httpOnly":true,"path":
"/"},"loggedIn":true,"username":"Dhananjay","__lastAccess":166702760156
9}

```

```

{"cookie":{"originalMaxAge":null,"expires":null,"httpOnly":true,"path":
"/"},"loggedIn":true,"username":"dhananjay","__lastAccess":166712108501
5}

```

Index.js

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

const PORT = 8000
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' })
}))

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

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

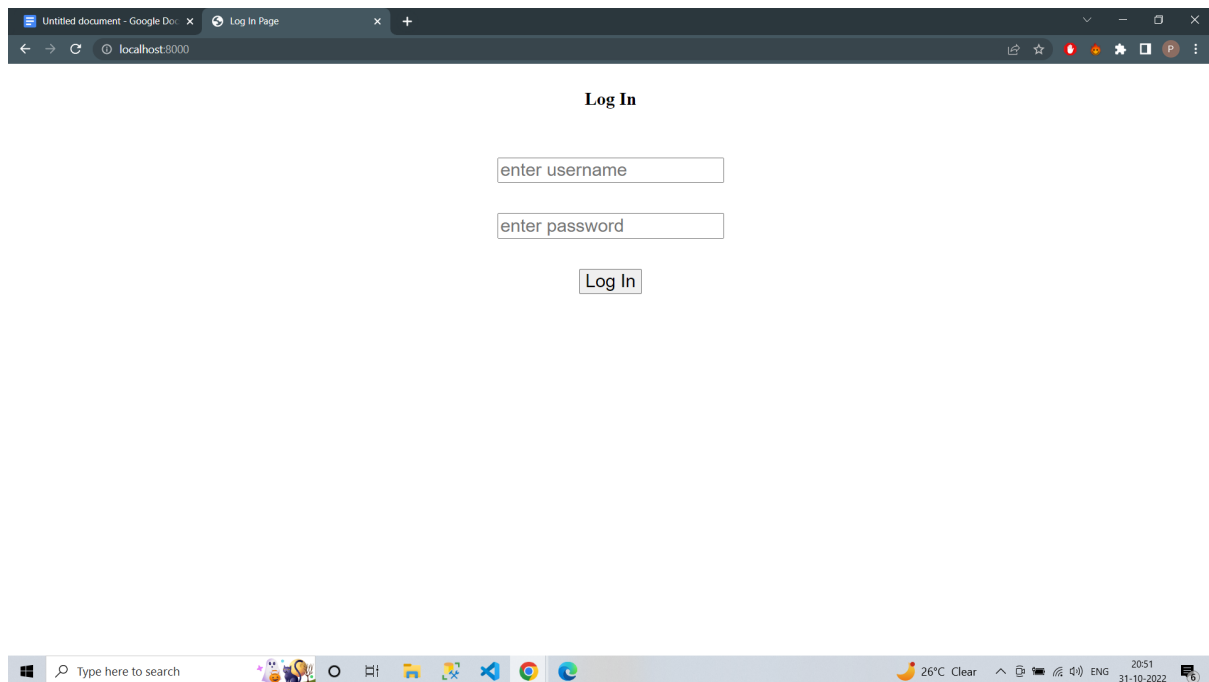
  if (username && password) {
    //credential validation logic here, if credential matches than
we store data in session
    req.session.loggedIn = true;
    req.session.username = username
    console.log(req.body)
    return res.redirect('/home')
  } else {
    return res.send('Please Enter Username And Password!')
  }
})

app.get('/home', (req, res) => {
  if (req.session.loggedIn) {
    console.log('logged in')
    res.sendFile(__dirname + '/public/home.html')
  } else {
```

```
        console.log('not logged in')
        res.sendFile(__dirname + '/public/login.html')
    }
})

app.listen(PORT, () => {
    console.log(`server listening on http://localhost:${PORT}`)
})
```

OutPut:-

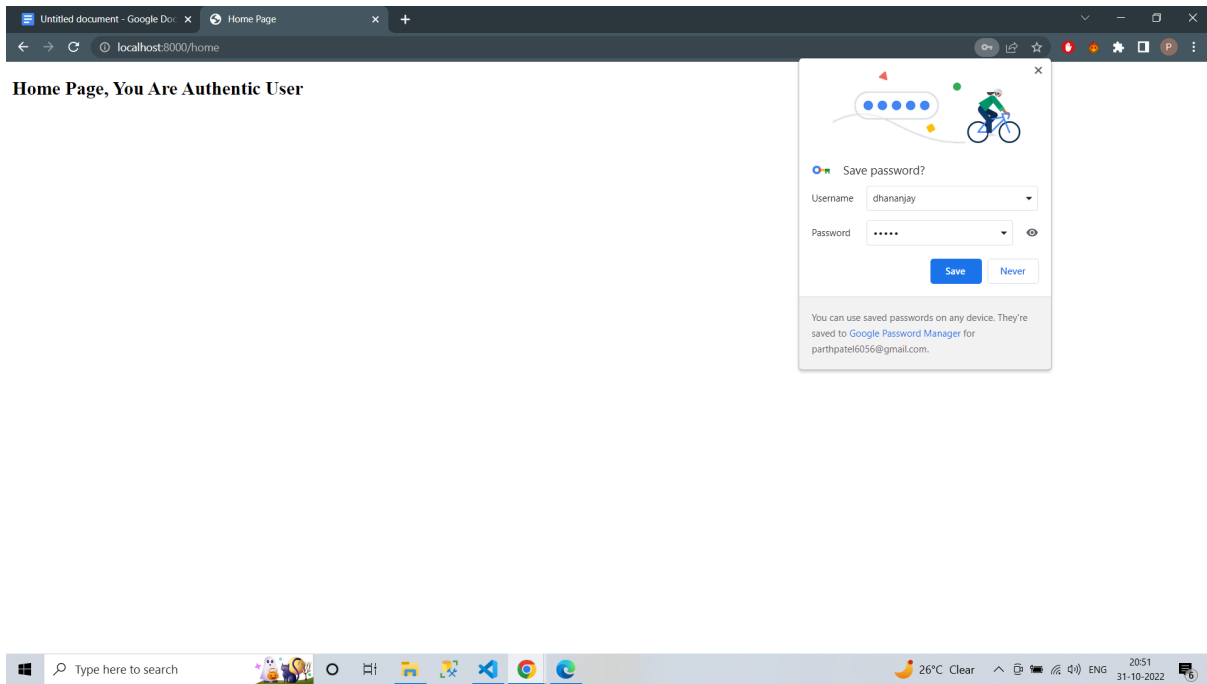


Log In

enter username

enter password

Log In



Q4.

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(models)

```
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

```

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')
    }
  })

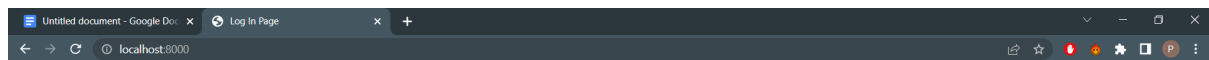
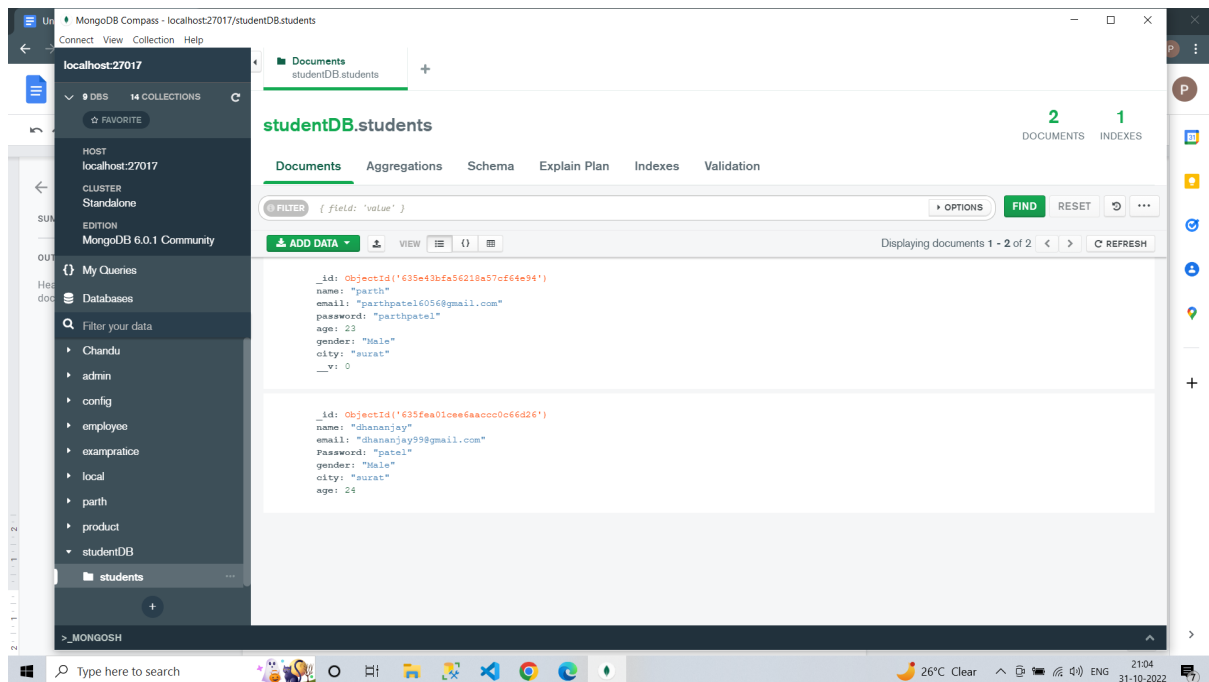
app.post('/login', async (req, res) => {
  let { username, password } = req.body
  if (username && password) {
    try {
      let result = await studentModel.find({ name: username,
password: password })
      if (result.length === 1) {
        //Generate Token
        const accessToken = jwt.sign({ name: username,
password: password }, accessTokenSecret)
        res.cookie('token', accessToken, { httpOnly: true });
        return res.redirect('/student')
        // return res.render('index', { session: result[0],
accessToken: accessToken })
      } 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}`)
})

```

Output:-



Log In



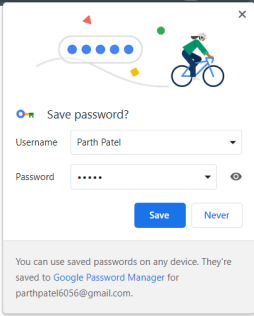
Google Docs: Online Document | x | Untitled document - Google Do... | x | NZ vs SL Live Streaming & L... | x | (22) WhatsApp | x | Home Page | x | +

localhost:8000/student

name	email	age	city	gender	Actions
Parth Patel	parthpatel6056@gmail.com	21	surat	Male	edit delete view

[Add Student](#)

[Log Out](#)



Save password?

Username: Parth Patel

Password:

[Save](#) [Never](#)

You can use saved passwords on any device. They're saved to Google Password Manager for parthpatel6056@gmail.com.

17_Anjali_Patel_PA....pdf | Show all | x

Type here to search

34°C Sunny 14:18 29-10-2022

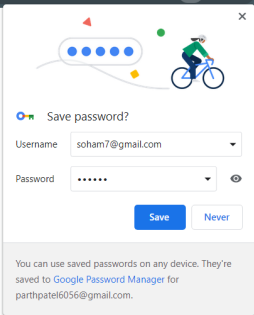
Google Docs: Online Document | x | Untitled document - Google Do... | x | NZ vs SL Live Streaming & L... | x | (22) WhatsApp | x | Home Page | x | +

localhost:8000/student

name	email	age	city	gender	Actions
Parth Patel	parthpatel6056@gmail.com	21	surat	Male	edit delete view
soham	soham7@gmail.com	22	surat	Male	edit delete view

[Add Student](#)

[Log Out](#)



Save password?

Username: soham7@gmail.com

Password:

[Save](#) [Never](#)

You can use saved passwords on any device. They're saved to Google Password Manager for parthpatel6056@gmail.com.

17_Anjali_Patel_PA....pdf | Show all | x

Type here to search

34°C Sunny 14:19 29-10-2022

Q5.

Index.js

```
const express = require('express')
const app = express()
const cors = require('cors')
const { sign, verify } = require('./auth/authenticate')
require('dotenv').config()
//database
```



```
require('./db/database')

//router
const studentRouter = require('./routes/studentRoute')

//model
const student = require('./models/studentModel')

//config
const PORT = process.env.PORT

//cors
app.use(cors())

//body-parser and json for sending json and reading body
app.use(express.json())
app.use(express.urlencoded({ extended: true }))

app.get('/', (req, res) => {
  console.log('default response')
  res.send('DEFAULT RESPONSE FROM SERVER')
})

//login request
app.post('/login', async (req, res) => {
  let { username, password } = req.body
  username = username
  password = password
  if (username && password) {
    let response = await student.find({ name: username, password:
password });
    if (response.length === 1) {
      return res.json(sign({ username }))
    } else {
      return res.status(401).json('Invalid username or password')
    }
  } else {
    return res.status(401).json('Please provide username and
password')
  }
})

app.use('/get-token', sign)
```

```
app.use('/student', [verify], studentRouter)

app.listen(PORT, () => {
  console.log(`app listening on http://localhost:${PORT}`);
})
```

Output:-

