Name: Vishwa V. Shah

Roll No.: 62

Subject: Open Source Web Development

Assignment: Practical Assignment – 2

Class: Msc.ICT – 3

Open Source Web Development

(Practical Assignment 2)

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.

(index.js)

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

const mongoose = require('mongoose');

const multer = require('multer');

const path = require('path');

mongoose.connect('mongodb://localhost:27017/assignment2quest1')

.then(()=>{

  console.log('Mongo connected successfully.');

})

.catch((err)=>{

  console.log(`Error : ${err}`);

})


const filesSchema = new mongoose.Schema({

  name: {

    type: String,

    required: true

  },

  email: {

    type: String,

    required: true

  },
```

```
    password: {
        type: String,
        required: true
    },
    aadhar: {
        type: String,
        required: true
    }
});

const documentSchema = new mongoose.Schema({
    fileId: {
        required: true,
        type : mongoose.Types.ObjectId,
        ref: 'files'
    },
    name: {
        required: true,
        type: String
    }
})

const File = new mongoose.model('files', filesSchema);
const Document = new mongoose.model('docs', documentSchema);

const storage = multer.diskStorage({
    destination: (req, file, cb)=>{
        try {
            if(file.mimetype !== 'application/pdf')
```

```
        {
            cb('Only pdf file is supported');
        }
        else
        {
            cb(null, './uploads');
        }
    } catch (error) {
        cb(error)
    }
  },
  filename: (req, file, cb)=>{
    try {
        cb(null, Date.now() + "-" + file.originalname);
    } catch (error) {
        cb(error)
    }
  }
});

const uploads = multer({storage})

const app = express();

app.get('/', (req, res)=>{
  try {
    return res.sendFile(path.join(__dirname, 'form.html'));
  } catch (error) {
    console.log(error);
```

```
        return res.status(500).json(error);
    }
});

app.get('/documents', (req, res)=>{
    try {
        return res.sendFile(path.join(__dirname, 'documents.html'));
    } catch (error) {
        return res.status(500).json({error});
    }
})

app.get('/docs', (req, res)=>{
    return res.sendFile(path.join(__dirname, 'download.html'));
})

app.get('/files', async(req, res)=>{
    try {
        const files = await File.aggregate([
            {
                $lookup: {
                    from: 'docs',
                    localField: '_id',
                    foreignField: 'fileId',
                    as: 'doc'
                }
            }
        ]);
```

```
        return res.status(200).json(files)
    } catch (error) {
        return res.status(500).json({error});
    }
});

app.post('/register', uploads.single('aadhar'), async (req, res)=>{
    const {name, email, password} = req.body;
    const aadhar = req.file.filename;
    const newUser = new File({name: name, email: email, password: password, aadhar: aadhar});
    await newUser.save();
    return res.redirect('/documents?id=' + newUser._id);
})

app.post('/document-uploads', uploads.array('docs'), async(req, res)=>{
    try {
        const id = req.body.id;
        const files = req.files;

        const docs = files.map(file => {
            return {
                fileId: new mongoose.Types.ObjectId(id),
                name: file.filename
            }
        });
        await Document.insertMany(docs);

        return res.redirect('/docs')
```

```
    } catch (error) {
        console.log(error);
        return res.status(500).json({error});
    }
});


app.listen(5000)
```

(documents.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Documents</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-12 mt-5">
                <form action="/document-uploads" enctype="multipart/form-data" method="post">
                    <div class="form-group">
                        <input type="hidden" name="id" id="id">
                        <label for="docs">Registration Documents</label>
                        <input type="file" multiple name="docs" id="docs" class="form-control"
accept="application/pdf">
                    </div>
                    <div class="form-group mt-5 text-center">
```

```
                <button type="submit" class="btn btn-primary">Register</button>
            </div>
          </form>
        </div>
      </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
    <script>
      const params = new URLSearchParams(window.location.search)
      document.getElementById('id').value = params.get('id');
    </script>
</body>
</html>
```

(download.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Downloads</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container table-responsive">
      <table class="table table-hover">
        <thead>
```

```
            <tr>
                <th>Name</th>
                <th>Email</th>
                <th>Aadhar</th>
                <th>Documents</th>
            </tr>
        </thead>
        <tbody id="tbody"></tbody>
    </table>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
<script>
    fetch('/files', {method: 'GET'})
    .then((response)=>{
        return response.json();
    })
    .then((response)=>{
        response.forEach(element => {
            const documentLinks = element.doc.map(e =>
                `<a download href='./uploads/${e.name}'>${e.name}</a>`
            ).join(', ');
            document.getElementById('tbody').innerHTML += `
                <tr>
                    <td>${element.name}</td>
                    <td>${element.email}</td>
                    <td><a download href='./uploads/${element.aadhar}'>${element.aadhar}</a></td>
                    <td>${documentLinks}</td>
```

```
            </tr>
          `;
        });
      })
      .catch((err)=>{
        alert('Error occurred');
        console.log(err);
      })
  </script>
</body>
</html>
```

(form.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registration Form</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
</head>
<body>
  <div class="container">
    <div class="row">
      <div class="col-12 mt-5">
        <form action="/register" enctype="multipart/form-data" method="post">
          <div class="form-group">
            <label for="name">Name</label>
```

```
                <input type="text" name="name" id="name" class="form-control" placeholder="Name
of user">
            </div>
            <div class="form-group">
                <label for="email">Email</label>
                <input type="email" name="email" id="email" class="form-control"
placeholder="Email of user">
            </div>
            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" name="password" id="password" class="form-control"
placeholder="Password">
            </div>
            <div class="form-group">
                <label for="aadhar">Aadhar Card</label>
                <input type="file" name="aadhar" id="aadhar" class="form-control"
accept="application/pdf">
            </div>
            <div class="form-group mt-5 text-center">
                <button type="submit" class="btn btn-primary">Register</button>
            </div>
        </form>
    </div>
  </div>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

2) Express Login application with file session store.

(index.js)

```
const express = require('express');
const session = require('express-session');
const FileStore = require('session-file-store')(session);
const bodyParser = require('body-parser');
const path = require('path');
const { error } = require('console');

const app = express();

app.use(session({
   store: new FileStore({
      path: './sessions',
      ttl: '3600',
      retries: 5
   }),
   saveUninitialized: false,
   resave: false,
   secret: 'deepganatra'
}));

app.use(bodyParser.urlencoded({extended: false}))

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

app.use(express.static(path.join(__dirname, 'public')));
```

```
const users = [
   {name: 'Deep Ganatra', email: 'ganatradeep9@gmail.com', password: 'deep123'},
   {name: 'Vishwa Shah', email: 'shahvishwa212002@gmail.com', password: 'vishwa123'},
   {name: 'Harsh Adadhiyawala', email: 'harshadadhiyawala@gmail.com', password: 'harsh123'}
];

app.get('/', async(req, res)=>{
   if(req.session.name)
   {
      return res.render('dashboard', {name: req.session.name});
   }
   else
   {
      return res.redirect('/login');
   }
});

app.get('/login', (req, res)=>{
   return res.render('login', {error: null});
})
app.post("/login", (req, res)=>{
   const {email, password} = req.body;
   let flag = false;
   users.map(user=>{
      if(user.email === email && user.password === password)
      {
         req.session.name = user.name;
         flag = true
         return;
```

```
        }
    });

    if(flag)
        return res.redirect('/dashboard')
    else
        return res.render('login', {error: 'Invalid email or password'});
})
app.get('/dashboard', (req, res)=>{
    return res.render('dashboard', {name: req.session.name});
})
app.get('/logout', (req, res)=>{
    req.session.destroy();
    res.clearCookie('connect.sid');
    return res.redirect('/login');
})

app.listen(5000)
```

(login.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Page</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
```

```
    <h1>Login</h1>
    <%
      if(error)
      {%>
        <p class="error"><%= error %></p>
      <%
    }
    %>
    <form action="/login" method="post">
      <div>
        <label for="email">Email</label>
        <input type="email" name="email" id="email" required>
      </div>
      <div>
        <label for="password">Password</label>
        <input type="password" name="password" id="password" required>
      </div>
      <button type="submit">Login</button>
    </form>
  </div>
</body>
</html>
```

(dashboard.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dashboard</title>
```

```
</head>
<body>
    <div class="container">
        <p>
            Hello <%= name %>
        </p>
        <button onclick="window.location.replace('/logout')" type="button">Logout</button>
    </div>
</body>
</html>
```

3) Express Login application with redis session store.

(index.js)

```
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const RedisStore = require('connect-redis').default;
const redis = require('redis');
const path = require('path');

const redisClient = redis.createClient({
   host: 'localhost',
   port: 6379
});

redisClient.on('error', (err)=>{
   console.log(err);
})

const app = express();
app.use(session({
   store: new RedisStore({client: redisClient}),
   secret: 'deepganatra',
   resave: false,
   saveUninitialized: false,
   cookie: {maxAge: 36000000}
}));
app.use(bodyParser.urlencoded({extended: false}));
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
```

```
app.use(express.static(path.join(__dirname, 'public')));

const users = [
   {name: 'Deep Ganatra', email: 'ganatradeep9@gmail.com', password: 'deep123'},
   {name: 'Vishwa Shah', email: 'shahvishwa212002@gmail.com', password: 'vishwa123'},
   {name: 'Harsh Adadhiyawala', email: 'harshadadhiyawala@gmail.com', password: 'harsh123'}
];

app.get('/', (req, res)=>{
   if(req.session.name)
   {
      return res.redirect('/dashboard');
   }
   else
   {
      return res.redirect('/login');
   }
});
app.get('/login', (req, res)=>{
   return res.render('login', {error: null});
})
app.post('/login', (req, res)=>{
   const {email, password} = req.body;
   let flag = false

   users.map(user=>{
      if(user.email === email && password === user.password)
      {
         req.session.name = user.name;
```

```
            flag = true;
            return;
        }
    });

    if(flag)
    {
        return res.redirect('dashboard');
    }
    else
    {
        return res.render('login', {error: 'Invalid email or password'});
    }
})
app.get('/dashboard', (req, res)=>{
    return res.render('dashboard', {name: req.session.name})
})
app.get('/logout', (req, res)=>{
    req.session.destroy();
    res.clearCookie('connect.sid')
    return res.redirect('/login', {error: null});
})

app.listen(3000)
```

(login.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <%
            if(error)
            {%>
                <p><%= error %></p>
            <%}
        %>
        <form action="/login" method="post">
            <div>
                <label for="email">Email</label>
                <input type="email" name="email" id="email" required>
            </div>
            <div>
                <label for="password">Password</label>
                <input type="password" name="password" id="password" required>
            </div>
            <button type="submit">Login</button>
        </form>
    </div>
</body>
</html>
```

(dashboard.ejs)

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

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>
            Hello <%= name %>
        </h1>
        <button onclick="window.location.replace('/logout')">Logout</button>
    </div>
</body>
</html>
```

4) Login, CRUD operations for students table with mongoose, express and any one template engine, Logout.

(index.js)

```
const express = require('express');
const path = require('path');
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');
const cookieParser = require('cookie-parser');


mongoose.connect('mongodb://localhost:27017/assignment2quest1').then(()=>{
  console.log('MongoDB Connected Successfully.');
});


const userSchema = mongoose.Schema({
  name: {
    required: true,
    type: String
  },
  password: {
    required: true,
    type: String
  },
  email: {
    required: true,
    unique: true,
    type: String
  }
```

```
});
const User = mongoose.model('users', userSchema);
const sk = 'ganatradeep'

const app = express();

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
app.use(bodyParser.urlencoded({extended: false}));
app.use(cookieParser());

app.get('/', (req, res)=>{
   return res.render('register');
})
app.post('/', async(req, res)=>{
   const {name, password, email} = req.body;
   const hash = await bcrypt.hash(password, 10);

   const user = new User({
      name: name,
      password: hash,
      email: email
   });
   await user.save();
   return res.redirect('/login');
})
app.get('/login', (req, res)=>{
   return res.render('login', {error: null});
})
```

```
app.post('/login', async(req, res)=>{
    const {email, password} = req.body;
    const user = await User.findOne({email: email});

    if(!user)
        return res.render('login', {error: 'Credentials invalid'});
    if(!bcrypt.compareSync(password, user.password))
        return res.render('login', {error: 'Credentials invalid'});

    const token = jwt.sign(JSON.stringify(user), sk);
    res.cookie('token', 'Bearer ' + token)

    return res.redirect('dashboard')
});

app.get('/dashboard', (req, res)=>{
    const auth = req.cookies.token;

    let token = null;

    token = auth.split(' ')[1];
    if(!token)
        return redirect('/login');

    const payload = jwt.verify(token, sk);

    return res.render('dashboard', {token: payload})
})
app.get('/logout', (req, res)=>{
```

```
    res.clearCookie('token');
    return res.redirect('/');
})

app.listen(500);
```

(register.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register User</title>
</head>
<body>
    <form action="/" method="post">
        <label for="name">Name</label>
        <input type="text" name="name" id="name" required>
        <br/>
        <label for="email">Email</label>
        <input type="email" name="email" id="email" required>
        <br/>
        <label for="password">Password</label>
        <input type="password" name="password" id="password" required>
        <br/>
        <button type="submit">Register</button> | <a href="/login">Login</a>
    </form>
</body>
</html>
```

(login.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Login</title>
</head>
<body>
   <p style="color: red;font-size: small;">
      <%
         if(error)
         {%>
            <%= error %>
         <%
         }
      %>
   </p>
   <form action="/login" method="post">
      <label for="email">Email</label>
      <input type="email" name="email" id="email" required>
      <br/>
      <label for="password">Password</label>
      <input type="password" name="password" id="password" required>
      <br/>
      <button type="submit">Login</button> | <a href="/">Register</a>
   </form>
</body>
</html>
```

(dashboard.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Dashboard</title>
</head>
<body>
   <%= token.name %>
   <br>
   <button onclick="window.location.replace('/logout')">Logout</button>
</body>
</html>
```

5) Stateless auth with JWT, CRUD operations for students table with mongoose, express and frontend(html,css,javascript/jquery), Logout.

(index.js)

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

const mongoose = require('mongoose');

const path = require('path');

const bodyParser = require('body-parser');

const bcrypt = require('bcrypt');


mongoose.connect('mongodb://localhost:27017/assignment2quest1').then(()=>{console.log('MongoDB connected successfully');})


const app = express();


app.use(bodyParser.urlencoded({extended: false}));

app.use(express.static(path.join(__dirname, 'public')));


const userSchema = mongoose.Schema({
   name: {
      required: true,
      type: String
   },
   password: {
      required: true,
      type: String
   },
   email: {
      required: true,
      unique: true,
```

```
        type: String
    }
});
const User = mongoose.model('users', userSchema);

app.get('/students', async(req, res)=>{
    const users = await User.find();
    return res.status(200).json(users);
})
app.get('/students/:id', async(req, res)=>{
    const user = await User.findById(req.params.id);
    return res.status(200).json(user);
})
app.delete('/students/delete/:id', async(req, res)=>{
    await User.findByIdAndDelete(req.params.id);
    return res.status(200).json({});
})
app.post('/students/add', async(req, res)=>{
    const {name, email, password} = req.body;
    const hash = await bcrypt.hash(password, 10);

    const user = new User({
        name: name,
        email: email,
        password: hash
    });
    await user.save();
    return res.status(200).json({});
})
```

```
app.put('/students/update/:id', async(req, res)=>{
    const {email, name} = req.body;
    const id = req.params.id;

    const user = await User.findById(id);

    if(!user)
        return res.status(404);

    user.name = name;
    user.email = email;

    if(req.body.password)
    {
        const password = await bcrypt.hash(req.body.password, 10);
        user.password = password;
    }

    await user.save();
    return res.status(200).json({});
})
app.listen(5000);
```

(/public/index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Students CRUD</title>
```

```
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdn.datatables.net/2.1.5/css/dataTables.dataTables.min.css">
</head>
<body>
    <div class="container">
        <form action="#" method="post">
            <input type="hidden" name="id">
            <label for="name" class="form-label">Name</label>
            <input type="text" name="name" id="name" class="form-control">
            <br>
            <label for="email" class="form-label">Email</label>
            <input type="email" name="email" id="email" class="form-control">
            <br>
            <label for="password" class="form-label">Password</label>
            <input type="password" name="password" id="password" class="form-control">
            <br>
            <button type="submit" class="btn btn-outline-success">Save</button>
        </form>
    </div>
    <table class="table table-hover">
        <thead>
            <tr>
                <th>Name</th>
                <th>Email</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody></tbody>
```

```
    </table>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.21.0/jquery.validate.min.js"
integrity="sha512-
KFHXdr2oObHKI9w4Hv1XPKc898mE4kgYx58oqsc/JqqdLMDI4YjOLzom+EMlW8HFUd0QfjfAvxSL6sEq/a42f
Q==" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
    <script src="https://cdn.datatables.net/2.1.5/js/dataTables.min.js"></script>
    <script>
        $(document).ready(function(){
            Swal.showLoading();


            $.ajax({
                type: 'GET',
                method: 'GET',
                dataType: 'JSON',
                url: '/students',
                success: function(response){
                    $.each(response, function(){
                        $("tbody").append(`
                        <tr>
                            <td>${this.name}</td>
                            <td>${this.email}</td>
                            <td>
                                <button data-id="${this._id}" class="edit btn btn-sm btn-outline-
primary">Edit</button>
                                <button data-id="${this._id}" class="delete btn btn-sm btn-outline-
danger">Delete</button>
```

```
            </td>
          </tr>
          `);
      });

      $('.table').DataTable();
      Swal.close();
    }
})
$("form").submit(function(e){
    e.preventDefault();

    if($('input[name="id"]').val())
    {
      // edit
      $("form").validate({
          rules:{
              name: "required",
              email: {
                  required: true,
                  email: true
              }
          },
          messages:{
              name: "<span class='text-danger'>*Please enter name</span>",
              email: {
                  required: "<span class='text-danger'>*Please enter email</span>",
                  email: "<span class='text-danger'>*Please enter email properly</span>"
              }
```

```
    }
});
if($("form").valid())
{
    Swal.showLoading();
    setTimeout(() => {
        const json = {};
        $.each($("form").serializeArray(), function(){
            json[this.name] = this.value
        });
        $.ajax({
            type: 'PUT',
            method: 'PUT',
            data: json,
            dataType: 'JSON',
            url: '/students/update/' + $('input[name="id"]').val(),
            success: function(response){
                Swal.close();
                Swal.fire({
                    icon: 'success',
                    title: 'Success',
                    text: 'Student added successfully.'
                }).then(()=>{
                    window.location.reload();
                })
            },
            error: function(err){
                Swal.close();
                Swal.fire({
```

```
                    icon: 'success',
                    title: 'Success',
                    text: 'Student added successfully.'
                }).then(()=>{
                    window.location.reload();
                })
            }
        })
    }, 500);
  }
}
else
{
  // insert
  $("form").validate({
    rules:{
      name: "required",
      password: "required",
      email: {
        required: true,
        email: true
      }
    },
    messages:{
      name: "<span class='text-danger'>*Please enter name</span>",
      password: "<span class='text-danger'>*Please enter password</span>",
      email: {
        required: "<span class='text-danger'>*Please enter email</span>",
        email: "<span class='text-danger'>*Please enter email properly</span>"
```

```
        }
    }
});
if($("form").valid())
{
    Swal.showLoading();
    setTimeout(() => {
        const json = {};
        $.each($("form").serializeArray(), function(){
            json[this.name] = this.value
        });
        $.ajax({
            type: 'POST',
            method: 'POST',
            data: json,
            dataType: 'JSON',
            url: '/students/add',
            success: function(response){
                Swal.close();
                Swal.fire({
                    icon: 'success',
                    title: 'Success',
                    text: 'Student added successfully.'
                }).then(()=>{
                    window.location.reload();
                })
            },
            error: function(err){
                Swal.close();
```

```
                    Swal.fire({
                        icon: 'success',
                        title: 'Success',
                        text: 'Student added successfully.'
                    }).then(()=>{
                        window.location.reload();
                    })
                }
            })
        }, 500);
    }
  }
})
$("tbody").on('click', '.edit', function(){
    const id = $(this).data('id');
    Swal.showLoading();
    setTimeout(() => {
        $.ajax({
            type: 'GET',
            method: 'GET',
            url: '/students/' + id,
            dataType: 'JSON',
            success: function(response){
                Swal.close();
                $("#name").val(response.name)
                $("#email").val(response.email)
                $("input[name='id']").val(response._id)
            }
        })
```

```
            }, 500);
        })
        $("tbody").on('click', '.delete', function(){
            const id = $(this).data('id');
            Swal.showLoading();
            setTimeout(() => {
                $.ajax({
                    type: 'DELETE',
                    method: 'DELETE',
                    url: '/students/delete/' + id,
                    dataType: 'JSON',
                    success: function(response){
                        Swal.close();
                        Swal.fire({
                            icon: 'success',
                            title: 'Success',
                            text: 'Student deleted successfully.'
                        }).then(()=>{
                            window.location.reload()
                        })
                    }
                })
            }, 500);
        })
    })
  </script>
</body>
</html>
```

6) Consider Category and Products collections. Develop modules of CRUD. Use Express, Mongoose, EJS. For a product, user can upload multiple pictures.

(index.js)

```
const express = require('express');
const path = require('path');
const bodyParser = require('body-parser');
const cors = require('cors');
const session = require('express-session');
const authRouter = require('./controllers/auth');
const catRouter = require('./controllers/category');
const productRouter = require('./controllers/product');
const authMiddleWare = require('./middleware/auth');

const app = express();

app.use(cors());
app.use(bodyParser.urlencoded({extended: false}));
app.use(session({secret: 'deepganatra', resave: false, saveUninitialized: false}));
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

app.use(express.static(path.join(__dirname, 'uploads')));

app.get('/', (req, res)=>{
   return res.render('login', {error: null});
});
app.get('/dashboard', authMiddleWare, (req, res)=>{
   return res.render('dashboard', {name: req.session.name});
});
```

```
app.use('/auth', authRouter);
app.use('/category', catRouter);
app.use('/product', productRouter);

app.listen(5000);
```

(db.js)

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

mongoose.connect('mongodb://localhost:27017/ass2question6')
.then(()=>{
   console.log('MongoDB Connected Successfully');
})
.catch((err)=>{
   console.log(`Error : ${err}`);
})

module.exports = mongoose;
```

(migrate.js)

```
const roleSeed = require('./seeders/role');
const userSeed = require('./seeders/user');

const seed = async () =>{
   console.log(`Seed DB started`);

   await roleSeed();
   await userSeed();

   console.log(`Seed DB completed`);
```

```
}

seed();
```

(/models/Role.js)

```
const mongoose = require('../db');

const roleSchema = mongoose.Schema({
   role: {
      required: true,
      type: String
   }
})

const Role = mongoose.model('Role', roleSchema);

module.exports = Role;
```

(/models/User.js)

```
const mongoose = require('../db');
const Role = require('./Role');

const userSchema = mongoose.Schema({
   name: {
      required: true,
      type: String
   },
   email: {
      required: true,
      unique: true,
      type: String
```

```
  },
  password: {
    required: true,
    type: String
  },
  role_id: {
    required: true,
    type: mongoose.Types.ObjectId,
    ref: Role
  }
});

const User = mongoose.model('User', userSchema);

module.exports = User;
```

(/models/Category.js)

```
const mongoose = require('../db');

const categorySchema = mongoose.Schema({
  name: {
    required: true,
    type: String
  },
  image: {
    required: true,
    type: String
  }
});
```

```
const Category = mongoose.model('Category', categorySchema);


module.exports = Category;
```

(/models/Product.js)

```
const mongoose = require('../db');
const Category = require('./Category');


const productSchema = mongoose.Schema({
   name: {
      type: String,
      required: true
   },
   price: {
      type: mongoose.Types.Decimal128,
      required: true,
      min: 0
   },
   quantity: {
      type: Number,
      required: true,
      min: 0
   },
   sku: {
      type: String,
      required: true,
      unqiue: true
   },
   category_id: {
      required: true,
```

```
        type: mongoose.Types.ObjectId,
        ref: Category
    }
});

const Product = mongoose.model('Product', productSchema);

module.exports = Product;
```

(/models/ProductImages.js)

```
const mongoose = require('../db');
const Product = require('./Product');

const productImageSchema = mongoose.Schema({
    file: {
        type: String,
        required: true
    },
    product_id: {
        type: mongoose.Types.ObjectId,
        required: true,
        ref: Product
    }
});

const ProductImages = mongoose.model('ProductImages', productImageSchema);

module.exports = ProductImages;
```

(/seeders/role.js)

```
const Role = require('../models/Role');
```

```
const seed = async () => {
   await Role.deleteMany();

   const roles = [
      {role: 'admin'},
      {role: 'user'},
   ];

   await Role.insertMany(roles);
   console.log(`Role Seeder Successfully Implemented`);
}

module.exports = seed;
```

(/seeders/user.js)

```
const Role = require('../models/Role');
const User = require('../models/User');
const {encrypt} = require('../functions/hashing');

const seed = async () => {
   const admin = await Role.findOne({role: 'admin'});
   const users = [
      {name: 'Deep Ganatra', email: 'ganatradeep9@gmail.com', password: await encrypt('admin@123'),
role_id: admin._id}
   ];

   await User.deleteMany();
   await User.insertMany(users);
```

```
    console.log(`User Seeder Implemented Successfully`);
}

module.exports = seed;
```

(/middleware/auth.js)

```
function authMiddleWare(req, res, next)
{
    if(!req.session.name)
        return res.redirect('/');
    next();
}

module.exports = authMiddleWare;
```

(/controllers/auth.js)

```
const router = require('express').Router();
const Role = require('../models/Role');
const User = require('../models/User');
const {compare} = require('../functions/hashing');

router.post('/login', async(req, res)=>{
    try {
        const {email, password} = req.body;

        const user = await User.findOne({email: email});
        if(!user)
            return res.render('login', {error: 'Invalid credentials'});
        const com = await compare(password, user.password);
        if(!com)
            return res.render('login', {error: 'Invalid credentials'});
```

```
        req.session.userId = user._id;
        req.session.name = user.name;

        res.redirect('/dashboard');
    } catch (error) {
        return res.render('login', {error: error});
    }
});
router.get('/logout', (req, res)=>{
    req.session.destroy();
    res.clearCookie('connect.sid');
    return res.redirect('/');
})

module.exports = router;
```

(/controllers/category.js)

```
const router = require('express').Router();
const Category = require('../models/Category');
const authMiddleWare = require('../middleware/auth');
const catFun = require('../functions/category');
const multer = require('multer');
const path = require('path');
const fs = require('fs');

const storage = multer.diskStorage({
    destination: function(req, file, cb){
        cb(null, path.join(__dirname, '../uploads/category'));
    },
    filename: function(req, file, cb){
```

```
        cb(null, Date.now() + file.originalname);
    }
});

const upload = multer({storage});

router.get('/', authMiddleWare, async (req, res) => {
    const cats = await catFun.getAll();
    return res.render('category', {categories: cats, error: null});
})

router.post('/add', authMiddleWare, upload.single('image'), async (req, res)=>{
    try {
        const {name} = req.body;
        const image = req.file;

        const category = new Category();
        category.name = name;
        category.image = image.filename;
        await category.save();

        return res.redirect('/category');
    } catch (error) {
        const cats = await catFun.getAll();
        return res.render('category', {categories: cats, error: error});
    }
});
router.get('/get/:id',authMiddleWare, async(req, res)=>{
    try {
```

```
        const cat = await catFun.getById(req.params.id);
        return res.status(200).json(cat);
    } catch (error) {
        return res.status(500).json({error});
    }
})
router.post('/update', authMiddleWare,upload.single('image'), async(req, res)=>{
    try {
        const {name, id} = req.body;
        const category = await catFun.getById(id);
        category.name = name;
        if(req.file)
        {
            const image = req.file;
            fs.unlinkSync(path.join(__dirname, '../uploads/category/' + category.image));
            category.image = image.filename;
        }
        await category.save();
        return res.redirect('/category');
    } catch (error) {
        const cats = await catFun.getAll();
        return res.render('category', {categories: cats, error: error});
    }
})
router.get('/delete/:id', async(req, res)=>{
    try {
        const id = req.params.id;

        const category = await Category.findByIdAndDelete(id);
```

```
        fs.unlinkSync(path.join(__dirname, '../uploads/category/' + category.image));
        return res.status(200).json({})
    } catch (error) {
        return res.status(500).json({error});
    }
})

module.exports = router;
```

(/controllers/product.js)

```
const router = require('express').Router();
const Product = require('../models/Product');
const ProductImages = require('../models/ProductImages');
const authMiddleWare = require('../middleware/auth');
const catFun = require('../functions/category');
const prodFun = require('../functions/product');
const multer = require('multer');
const path = require('path');
const fs = require('fs');

const storage = multer.diskStorage({
    destination: function(req, file, cb){
        cb(null, path.join(__dirname, '../uploads/products'));
    },
    filename: function(req, file, cb){
        cb(null, Date.now() + file.originalname);
    }
});

const upload = multer({storage});
```

```
router.get('/', authMiddleWare, async(req, res)=>{
    const category = await catFun.getAll();
    const products = await prodFun.getAll();

    return res.render('product', {category: category, products: products, error: null});
});

router.post('/add', authMiddleWare, upload.array('image'), async(req, res)=>{
    try {
        const {name, price, quantity, sku, category_id} = req.body;

        const product = new Product();
        product.name = name;
        product.price = price;
        product.quantity = quantity;
        product.sku = sku;
        product.category_id = category_id;

        await product.save();

        req.files.forEach(element => {
            const prodImg = new ProductImages();
            prodImg.file = element.filename;
            prodImg.product_id = product._id;
            prodImg.save();
        });

        return res.redirect('/product');
```

```
    } catch (error) {
        const category = await catFun.getAll();
        const products = await prodFun.getAll();

        return res.render('product', {category: category, products: products, error: error});
    }
})

router.get('/get/:id', authMiddleWare, async(req, res)=>{
    try {
        const product = await prodFun.getById(req.params.id);
        return res.status(200).json(product);
    } catch (error) {
        console.log(error);
        return res.status(500).json({error});
    }
})

router.post('/update', authMiddleWare, upload.array('image'), async (req, res) => {
    try {
        const { name, price, quantity, sku, category_id, id } = req.body;
        const product = await Product.findById(id);
        product.name = name;
        product.price = price;
        product.quantity = quantity;
        product.sku = sku;
        product.category_id = category_id;
        await product.save();
        if (req.files.length > 0) {
```

```
            const images = await ProductImages.find({ product_id: product.id });
            for (const element of images) {
                fs.unlinkSync(path.join(__dirname, '../uploads/products/' + element.file));
                await ProductImages.findByIdAndDelete(element._id);
            }
            for (const file of req.files) {
                const prodImg = new ProductImages({
                    file: file.filename,
                    product_id: product._id
                });
                await prodImg.save();
            }
        }

        return res.redirect('/product');
    } catch (error) {
        const category = await catFun.getAll();
        const products = await prodFun.getAll();

        return res.render('product', { category: category, products: products, error: error });
    }
});

router.get('/delete/:id', authMiddleWare, async(req, res)=>{
    try {
        await ProductImages.deleteMany({product_id: req.params.id});
        await Product.findByIdAndDelete(req.params.id);

        return res.status(200).json({});
```

```
    } catch (error) {
        return res.status(500).json(error);
    }
})

module.exports = router;
```

(/functions/category.js)

```
const Category = require('../models/Category');

const getAll = async () => {
    const category = await Category.find();
    return category;
}

const getById = async (id) => {
    const category = await Category.findById(id);
    return category;
}

module.exports = {getAll, getById};
```

(/functions/product.js)

```
const Category = require('../models/Category');
const Product = require('../models/Product');
const ProductImages = require('../models/ProductImages');

const getAll = async() => {
    const products = await Product.find().populate('category_id');
    for (let product of products) {
        const images = await ProductImages.find({ product_id: product._id });
```

```
        product.images = images;
    }
    return products;
}

const getById = async (id) => {
    const products = await Product.findById(id).populate('category_id');
    return products;
}

module.exports = {getAll, getById};
```

(/functions/hashing.js)

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

async function encrypt(str)
{
    const hash = await bcrypt.hash(str, 10);
    return hash;
}

async function compare(str, hash)
{
    const result = await bcrypt.compare(str, hash);
    return result;
}

module.exports = {encrypt, compare};
```

(/views/includes/head.ejs)

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
```

```
  <a class="navbar-brand" href="#">ECommerce</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" href="/dashboard">Dashboard</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/category">Category</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/product">Products</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/auth/logout">Logout</a>
    </li>
    </ul>
  </div>
</nav>
```

(/views/includes/scripts.ejs)

```
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.7/dist/umd/popper.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"></script>
```

(/views/includes/styles.ejs)

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">
```

(/views/login.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>

  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">
</head>
<body>
  <div class="container mt-5">
    <%
      if(error)
      {%>
        <p class="text-danger"><%= error %></p>
      <%}
    %>
    <h1 class="text-center">Login to ECommerce</h1>
    <form action="/auth/login" method="post">
      <div class="form-group">
        <label for="email" class="form-label">Email</label>
        <input type="email" name="email" id="email" class="form-control">
      </div>
      <div class="form-group">
        <label for="password" class="form-label">Password</label>
        <input type="password" name="password" id="password" class="form-control">
      </div>
```

```
        <button type="submit" class="btn btn-outline-primary text-center">Login</button>
      </form>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

(/views/dashboard.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard</title>

    <%- include('includes/styles') %>
</head>
<body>
    <%- include("includes/head") %>
    <div class="container mt-5">
        <h1>Howdy <%= name %></h1>,
        <p>This is your admin panel where you can manage category and products.</p>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

(/views/category.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Category</title>
   <%- include('includes/styles') %>
</head>
<body>
   <%- include('includes/head') %>
   <div class="container">
      <%
         if(error)
         {%>
            <p class="text-danger"><%= error %></p>
         <%}
      %>
      <form action="/category/add" method="post" enctype="multipart/form-data">
         <input type="hidden" name="id">
         <div class="form-group">
            <label for="name" class="form-label">Name</label>
            <input type="text" name="name" id="name" class="form-control">
         </div>
         <div class="form-group">
            <label for="image" class="form-label">Image</label>
            <input type="file" name="image" id="image" class="form-control" accept="image/*">
         </div>
         <div class="form-group text-center mt-5">
```

```
            <button type="submit" class="btn btn-outline-primary">Save</button>
        </div>
    </form>
    <div class="row">
        <%
            categories.forEach(element => {%>
                <div class="col-4">
                    <div class="card">
                        <div class="card-header">
                            <h5 class="card-title">
                                <%= element.name %>
                            </h5>
                        </div>
                        <div class="card-body">
                            <img src="/category/<%= element.image %>" class="w-100" alt="">
                        </div>
                        <div class="card-footer">
                            <button data-id="<%= element._id %>" class="edit btn btn-outline-
success">Edit</button>
                            <button data-id="<%= element._id %>" class="delete btn btn-outline-
danger">Delete</button>
                        </div>
                    </div>
                </div>
            <%
            });
        %>
    </div>
</div>
```

```
<%- include('includes/scripts') %>
<script>
    $(".edit").click(function(){
        const id = $(this).data('id');
        fetch('/category/get/' + id)
        .then((response)=>{
            return response.json();
        })
        .then((data)=>{
            $("form").prop('action', '/category/update');
            $("#name").val(data.name)
            $("input[name='id']").val(data._id)
        })
        .catch((err)=>{
            console.log(err);
        })
    })

    $(".delete").click(function(){
        const id = $(this).data('id');
        fetch('/category/delete/' + id)
        .then((response)=>{
            window.location.reload();
        })
        .catch((err)=>{
            console.log(err);
        })
    })
</script>
```

```
</body>
</html>
```

(/views/product.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product</title>
    <%- include('includes/styles') %>
</head>
<body>
    <%- include('includes/head') %>
    <div class="container">
        <%
            if(error)
            {%>
                <p class="text-danger"><%= error %></p>
            <%}
        %>
        <form action="/product/add" method="post" enctype="multipart/form-data">
            <input type="hidden" name="id">
            <div class="form-group">
                <label for="name" class="form-label">Name</label>
                <input type="text" name="name" id="name" class="form-control">
            </div>
            <div class="form-group">
                <label for="price" class="form-label">Price</label>
                <input type="number" name="price" id="price" min="0" class="form-control">
```

```
</div>
<div class="form-group">
  <label for="quantity" class="form-label">Quantity</label>
  <input type="number" name="quantity" id="quantity" min="0" class="form-control">
</div>
<div class="form-group">
  <label for="sku" class="form-label">SKU</label>
  <input type="text" name="sku" id="sku" class="form-control">
</div>
<div class="form-group">
  <label for="image" class="form-label">Image</label>
  <input type="file" multiple name="image" id="image" class="form-control" accept="image/*">
</div>
<div class="form-group">
  <label for="category_id" class="form-label">Category</label>
  <select name="category_id" id="category_id" class="form-control">
    <%
      if(category)
      {
        category.forEach(element => {%>
          <option value="<%= element._id %>"><%= element.name %></option>
        <%});
      }
    %>
  </select>
</div>
<div class="form-group text-center mt-5">
  <button type="submit" class="btn btn-outline-primary">Save</button>
</div>
```

```
    </form>

    <div class="row">
      <%
        if(products)
        {
          products.forEach(element => {%>
            <div class="col-4">
              <div class="card">
                <div class="card-header">
                  <h5 class="card-title"><%= element.name %></h5>
                </div>
                <div class="card-body">
                  <div id="carouselExample<%= element._id %>" class="carousel slide">
                    <div class="carousel-inner">
                      <%
                      element.images.forEach((ele, index) => {%>
                        <div class="carousel-item <%= index === 0 ? 'active' : '' %>">
                          <img src="/products/<%= ele.file %>" class="d-block w-100" alt="...">
                        </div>
                        <%});
                      %>
                    </div>
                    <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExample<%= element._id %>" data-bs-slide="prev">
                      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
                      <span class="visually-hidden">Previous</span>
                    </button>
```

```
                        <button class="carousel-control-next" type="button" data-bs-
target="#carouselExample<%= element._id %>" data-bs-slide="next">
                            <span class="carousel-control-next-icon" aria-hidden="true"></span>
                            <span class="visually-hidden">Next</span>
                        </button>
                    </div>
                </div>
                <div class="card-footer">
                    <p><strong>Price : </strong><%= element.price %> $</p>
                    <p><strong>SKU : </strong><%= element.sku %></p>

                    <div class="mt-2">
                        <button data-id="<%= element._id %>" class="edit btn btn-outline-
success">Edit</button>
                        <button data-id="<%= element._id %>" class="delete btn btn-outline-
danger">Delete</button>
                    </div>
                </div>
            </div>
        <%});
        }
    %>
  </div>
</div>
<%- include('includes/scripts') %>

<script>
  $('.edit').click(function(){
    const id = $(this).data('id');
```

```
    fetch('/product/get/' + id)
    .then((response)=>{
        return response.json();
    })
    .then((data)=>{
        $("input[name='id']").val(data._id)
        $("#name").val(data.name)
        $("#price").val(data.price.$numberDecimal)
        $("#quantity").val(data.quantity)
        $("#sku").val(data.sku)
        $("#category_id").find('option[value="'+ data.category_id._id +'"]').prop('selected', true)


        $("form").attr('action', '/product/update');
    })
    .catch((err)=>{
        console.log(err);
    })
})

$('.delete').click(function(){
    const id = $(this).data('id');
    fetch('/product/delete/' + id)
    .then((response)=>{
        window.location.reload();
    })
    .catch((err)=>{
        console.log(err);
    })
});
```

```
    </script>
</body>
</html>
```