

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №7**

з дисципліни “ Основи програмування ”

тема “Реєстрація та авторизація користувачів

”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) II курсу  групи КП-81  Кузюра Ольга Валентинівна  (*прізвище, ім’я, по батькові*)  варіант №\_\_\_\_\_\_\_ |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2018

**Мета роботи**

Реалізувати персоналізацію вмісту сайту за допомогою користувачів і авторизації із використанням сесій.

**Постановка завдання**

Знати:

1. Реєстрація, ідентифікація, аутентифікація та авторизація.
2. HTTP Cookies.

Потрібно:

1. Створити у навчальному репозиторії директорію проекту labs/lab7 і перейти до неї у редакторі.
2. Скопіювати у корінь проект попередньої лабораторної роботи для продовження роботи над ним.
3. Видалити з проекту приховану директорію .git.
4. Встановити пакети: passport, passport-local, cookie-parser, express-session.

### Попереднє завдання

1. Додати шаблони сторінок register.mst та login.mst.
2. Додати пусті обробники для запитів (винести у окремий Router для /auth):

GET /auth/register

POST /auth/register

GET /auth/login

POST /auth/login

POST /auth/logout

1. Налаштувати рендер сторінок register та login у відповідних обробниках.

### Завдання

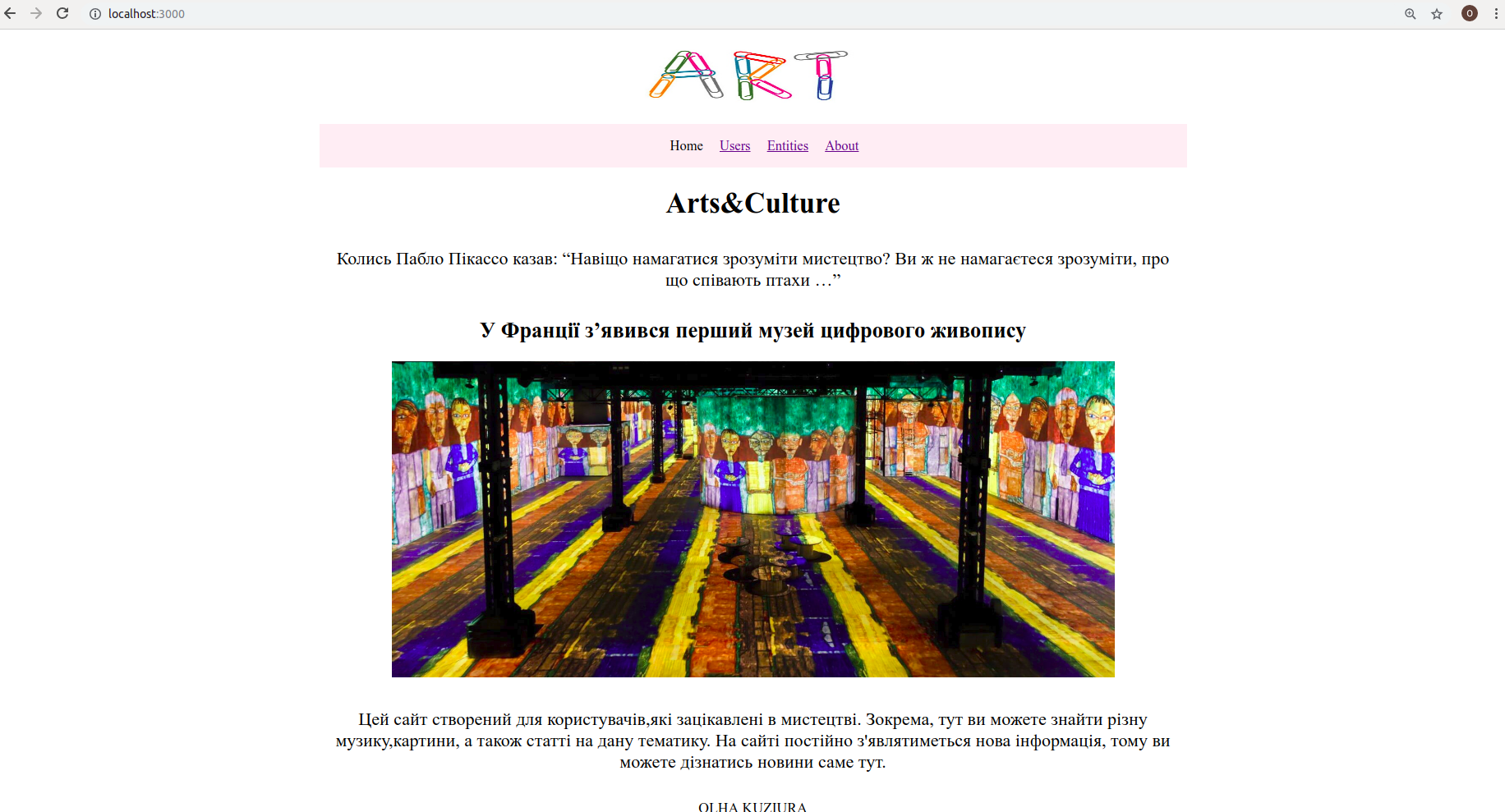
1. Додати у БД відмітки про ролі користувачів сайту:
   * Стандартний користувач
   * Адміністратор
2. Реалізувати реєстрацію користувачів на сайті по шляху /auth/register:
   * Реєстрація по унікальному логіну і паролю.
   * За допомогою HTML атрибутів задати мінімальний та максимальний розміри строк вводу та їх обов'зковість.
   * На форму реєстрації розмістити два поля для вводу паролю (ввід прихований) для перевірки співпадіння паролю (поки на сервері). Якщо реєстрація пройшла успішно - перевести користувача на сторінку логіну (/auth/login) у іншому випадку - знову на сторінку реєстрації, але з помилкою у аргументах URL (наприклад, /auth/register?error=Username+already+exists).
   * Перед формою виводити повідомлення про помилку із аргументів URL, якщо така існує.
   * Роль створеного користувача "Стандартний користувач".
3. За допомогою клієнтських сесій і HTTP Cookies реалізувати логін (/auth/login) та логаут (/auth/logout) користувачів на сайті. У шапці всіх сторінок відображати ім’я авторизованого користувача або повідомлення про те, що користувач не увійшов на сайт.
4. Дозволити гостям (неавторизованим користувачам) доступ лише до таких сторінок:
   * головної сторінки (/)
   * сторінки реєстрації (/auth/register)
   * входу (/auth/login)
   * та допоміжних сторінок (/about та ін.).
5. Обмежити доступ до сторінки перегляду списку зареєстрованих користувачів (/users).  
   Лише користувачі-адміністратори:
   * бачать посилання на цю сторінку у шапці сайту
   * можуть переглядати цю сторінку
   * можуть змінювати роль користувача на його сторінці (/users/:id)

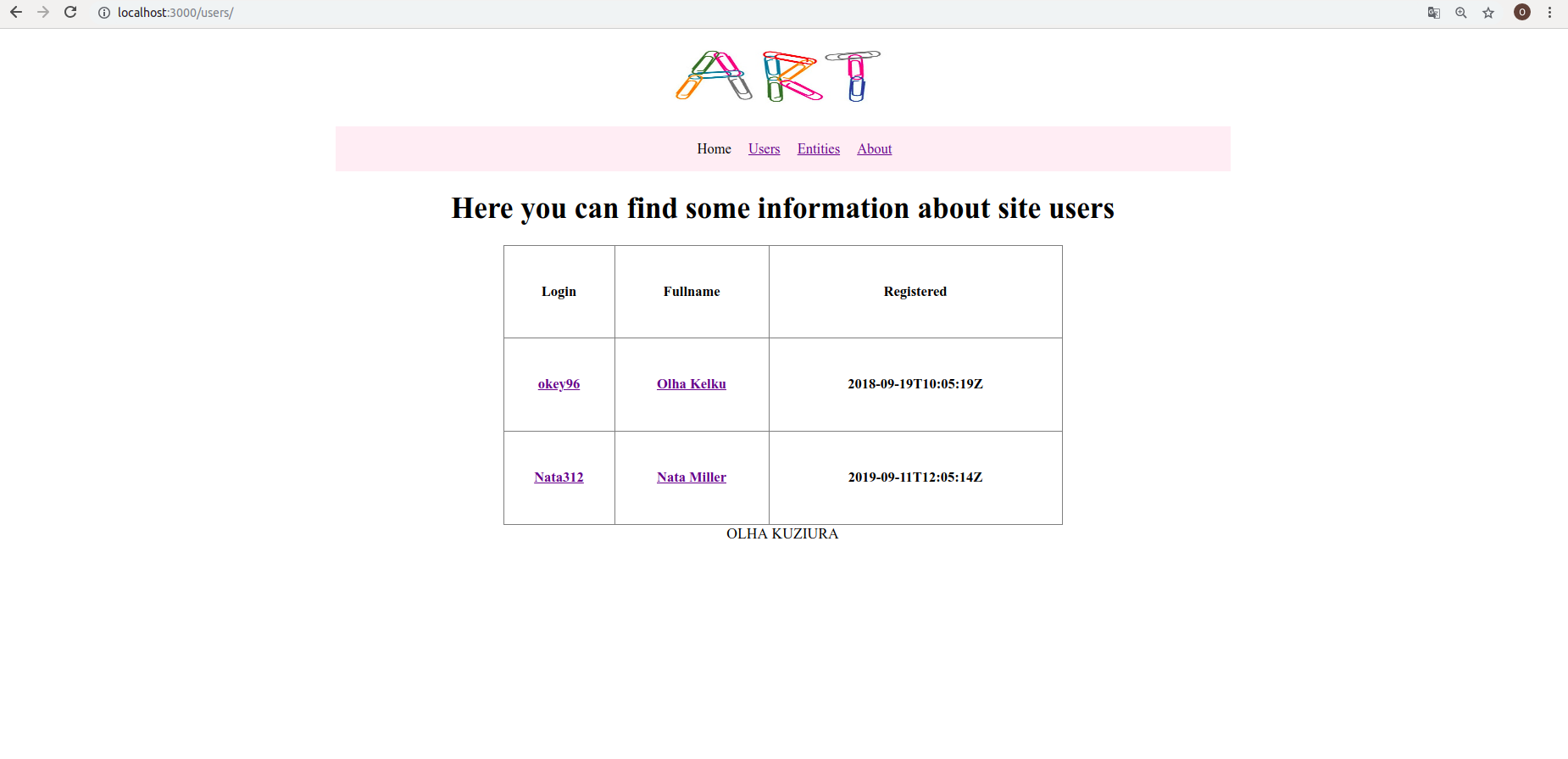
**Тексти коду програм**

|  |
| --- |
| **App.js** |
| const consolidate = require('consolidate');  const path = require('path');  const express = require('express');  const user = require('./models/user');  const article = require('./models/article');  const category = require('./models/category');  const app = express();  const mustache = require('mustache-express');  const busBoyBodyParser = require("busboy-body-parser");  const moment = require('moment');  const multer = require("multer");  const upload = require("./multer");  const mongoose = require('mongoose');  const config = require('./config');  const cloudinary = require("cloudinary");  const connectOptions = { useNewUrlParser: true };  const databaseUrl = config.DataBaseUrl;  const serverPort = config.ServerPort;  //console.log(databaseUrl);  //console.log(serverPort);  //app.listen(serverPort, function () { console.log('Server is ready'); });  mongoose.connect(databaseUrl, connectOptions)  .then(() => console.log(`Database connected : ${databaseUrl}`))  .then(() => app.listen(serverPort, function () { console.log('Server is ready'); }))  .catch(err => console.log(`Start error: ${err}`));  cloudinary.config({  cloud\_name: config.cloud\_name,  api\_key: config.api\_key,  api\_secret: config.api\_secret  });  function cloudinaryUploadPromise(fileBuffer, ...args) {  return new Promise((resolve, reject) => {  cloudinary.v2.uploader.upload\_stream(...args, (err, data) => {  if (err) return reject(err); resolve(data);  }).end(fileBuffer);  });  }  app.use(busBoyBodyParser());  const bodyParser = require('body-parser');  app.use(bodyParser.urlencoded({ extended: true }));  app.use(bodyParser.json());  // will open public/ directory files for http requests  app.use(express.static('public'));  const Path = path.join(\_\_dirname, 'views');  app.engine('mst', mustache(path.join(Path, 'partials')));  // set app options  app.set('views', Path);  app.set('view engine', 'mst');  let fileName;  // usage  app.get('/', function (req, res) {  res.render('index');  });  app.get('/new', (req, res) => {  res.render('new');  });  app.get('/users/new', (req, res) => {  res.render('new-user');  });  app.post('/new',/\*upload.single('fileurl'),async\*/(req, res) => {  // console.log(req);  // const result = await cloudinary.v2.uploader.upload(req.file.path);  // let filename = "/"+req.file.path;  //console.log(req.body);  let fileObject = req.files.fileurl;  let fileBuffer = fileObject.data;  cloudinaryUploadPromise(fileBuffer, { resource\_type: 'raw' })  .then(result => {  let image = result.url;  article.insert(image, req.body)  .then(article => res.redirect("/articles/" + article.id))  .catch(err => res.status(500).send(err));  })  .catch(err => res.status(500).send(err));  });  app.post('/users/new', (req, res) => {  // let filename = "/"+req.file.path;  //console.log(req.body);  let fileObject = req.files.fileurl;  let fileBuffer = fileObject.data;  cloudinaryUploadPromise(fileBuffer, { resource\_type: 'raw' })  .then(result => {  let image = result.url;  user.insert(image, req.body)  .then(user => res.redirect("/users/" + user.id))  .catch(err => res.status(500).send(err));  })  .catch(err => res.status(500).send(err));  });  app.post('/articles/:id', (req, res) => {  article.deleteById(req.params.id)  .then(() => res.redirect("/articles"))  .catch(err => res.status(500).send(err));  });  app.get('/users', function (req, res) {  user.getAll()  .then(users => res.render("users",  {  users: users  }))  .catch(err => res.status(500).send(err));  });  //file  app.get('/data/fs/:file', (req, res) => {  let curr = path.join(\_\_dirname, "data/fs/");  res.sendFile(curr + req.params.file);  });  app.get('/users/:id', function (req, res) {  const user\_id = req.params.id;  user.getById(user\_id)  .then(user => {  if (typeof user === "undefined") {  res.statusCode = 404;  res.send("User not found");  }  else {  res.render("user",  {  user: user  });  }  })  .catch(err => res.status(500).send(err));  });  app.get('/categories', (req, res) => {  category.getAll()  .then(categories => res.render("categories",  {  categories: categories  }))  .catch(err => res.status(500).send(err));  });  app.get('/categories/new', (req, res) => {  res.render('new-category');  });  app.post('/categories/new', (req, res) => {  console.log(req);  category.insert(req.body)  .then(category => res.redirect("/categories/" + category.id))  .catch(err => res.status(500).send(err));  });  app.post('/categories/:id', (req, res) => {  category.deleteById(req.params.id)  .then(() => res.redirect("/categories"))  .catch(err => res.status(500).send(err));  });  app.get('/categories/:id', function (req, res) {  category.getById(req.params.id)  .then(category => {  if (typeof category === "undefined") {  res.statusCode = 404;  res.send(" not found");  }  else {  res.render("category",  {  category: category  });  }  })  .catch(err => res.status(500).send(err));  });  app.get('/articles', function (req, res) {  let search = req.query.search;  console.log(search);  if (!search) {  search = "";  }  article.getAll(search)  .then(articles => {  if (!req.query.page) {  req.query.page = 0;  }  let next = +req.query.page + 1;  let prev = +req.query.page - 1;  let currPage = +req.query.page + 1;  let maxPage = Math.ceil(articles.length / 1);  let right = true;  let left = true;  if (currPage == maxPage || maxPage == 1) {  right = false;  }  if (currPage == 1) {  left = false;  }  let artcileOfArticles = articles.slice(+req.query.page, +req.query.page + 1);  const data1 = { artcileOfArticles, maxPage, next, prev, currPage, right, left, search };  res.render('articles', data1);  })  .catch(err => res.status(500).send(err));  });  app.get('/articles/update/:id', (req,res) =>{  res.render('article-update');  });  app.post('/articles/update/:id', (req,res) =>{  let fileObject = req.files.fileurl;  let fileBuffer = fileObject.data;  cloudinaryUploadPromise(fileBuffer, { resource\_type: 'raw' })  .then(result => {  let image = result.url;  article.update(image,req.params.id, req.body)  .then(article => res.redirect("/articles/" + article.id))  .catch(err => res.status(500).send(err));  })  .catch(err => res.status(500).send(err));  });  app.get('/articles/:id', function (req, res) {  article.getById(req.params.id)  .then(article => {  if (typeof article === "undefined") {  res.statusCode = 404;  res.send(" not found");  }  else {  res.render("article",  {  article: article  });  }  })  .catch(err => res.status(500).send(err));  });  app.get('/about', function (req, res) {  res.render('about');  });  app.get('/api/users', function (req, res) {  const users = user.getAll();  res.setHeader('Content-Type', 'application/json');  res.send(users);  });  app.get('/api/users/:id', function (req, res) {  res.setHeader('Content-Type', 'application/json');  let id = req.params.id;  const userById = user.getById(id);  if (userById == undefined) {  res.statusCode = 404;  res.end('Not Found');  }  else {  res.send(userById);  }    });    // const port = 3002;  // app.listen(port, (err) => {  // if (err) {  // return console.log('something bad happened', err);  // }  // console.log(`server is listening on ${port}`);  // }); |
| **new-user.mst** |
| <!doctype html>  <html lang="en">  <head>  {{>head}}  </head>  <body>  {{>header}}  <form method="POST" enctype="multipart/form-data">  <fieldset>  Login:<br>  <input type="text" name="login" ><br>  Fullname:<br>  <input type="text" name="fullname" ><br><br>  Avatar file:<br> <input type="file" name="avaUrl"><br>  <input type="submit" value="Submit">  </fieldset>  </form>  </body>  <footer>{{>footer}}</footer>  </html> |

|  |
| --- |
| Auth.js |
| const express = require("express");  const router = express.Router();  const config = require("../config");  const upload = require("../multer");  const cloudinary = require('cloudinary');  const busboyBodyParser = require('busboy-body-parser');  const bodyParser = require('body-parser');  const user = require('../models/user');  const crypto = require('crypto');  const passport = require('passport');  const LocalStrategy = require('passport-local').Strategy;  const cookieParser = require('cookie-parser');  const session = require('express-session');  const serverSalt = config.salt;  function sha512(password, salt) {  const hash = crypto.createHmac('sha512', salt);  hash.update(password);  const value = hash.digest('hex');  return {  salt: salt,  passwordHash: value  };  };  let isadmin;  router.use(busboyBodyParser());  //router.set('view engine', 'ejs');  router.use(bodyParser.urlencoded({ extended: true }));  router.use(bodyParser.json());  router.use(busboyBodyParser({ limit: '5mb' }));  // new middleware  router.use(cookieParser());  router.use(session({  secret: "Some\_secret^string",  resave: false,  saveUninitialized: true  }));  router.use(passport.initialize());  router.use(passport.session());  passport.serializeUser(function (user, done) {  // наприклад, зберегти у Cookie сесії id користувача  done(null, user.\_id);  });  // отримує інформацію (id) із Cookie сесії і шукає користувача, що їй відповідає  passport.deserializeUser(function (id, done) {  user.getById(id)  .then(user => done(null, user))  .catch(err => done(err, null));  // отримати користувача по id і викликати done(null, user);  // при помилці викликати done(err, null)  });  // налаштування стратегії для визначення користувача, що виконує логін  // на основі його username та password  passport.use(new LocalStrategy((username, password, done) => {  user.getAll()  .then(users => {  let user1;  for (let i = 0; i < users.length; i++) {  if (username == users[i].username && users[i].password == sha512(password, serverSalt).passwordHash) {  user1 = users[i];  }  }  done(null, user1);  })  .catch(err => done(err, null));  }));  router.get('/auth/login', function (req, res) {  res.render('login');  });  router.post('/auth/login', passport.authenticate('local', { failureRedirect: '/auth/login' }), function (req, res) {  if (req.user.role == "admin") isadmin = true;  else isadmin = false; res.render('index', { user: req.user, isadmin: isadmin });  });  router.get('/auth/logout', function (req, res) {  req.logout();  res.redirect('/');  });  router.get('/auth/register', function (req, res) {  if(req.user)  {if (req.user.role == "admin") isadmin = true;  else isadmin = false;}  res.render('register',{isadmin:isadmin});  });  router.post('/auth/register', function (req, res) {  let error = false;  let avaUrl = 'https://res-console.cloudinary.com/dr6f32lfo/thumbnails/transform/v1/image/upload//v1575150703/c2FtcGxlcy9hbmltYWxzL2tpdHRlbi1wbGF5aW5n/drilldown';  if (req.body.password != req.body.confirmPassword) {  res.redirect('/auth/register?error=Passwords+do+not+match');  }  user.getAll()  .then(users => {  for (let i = 0; i < users.length; i++) {  if (req.body.username == users[i].username) {  error = true;  }  }  if (error) {  res.redirect('/auth/register?error=Username+already+exists');  }  else {  const hash\_pass = sha512(req.body.password, serverSalt).passwordHash;  user.insert(avaUrl, req.body, hash\_pass)  .then(user => {  res.redirect('/auth/login');  })  .catch(err => res.status(500).send(err));  }  })  .catch(err => res.status(500).send(err));  });  module.exports = router; |

**Приклади результатів**





**Висновки**

Я Реалізувала персоналізацію вмісту сайту за допомогою користувачів і авторизації із використанням сесій.