Авторизација

Домашна 2, Информациска Безбедност

Оваа надграбда на претходниот проект додава едноставен систем на авторизација, односно систем на улоги со различни привилегии. Тоа се улогите ‘user’, ‘admin’ и ‘owner’. Улогата owner е издадена на посебен профил и е со највисока привилегија. Има целосна контрола врз другите корисници. Admin е издадена од owner на избрани корисници со улогата user. Корисниците со admin имаат контрола врз корисниците со user улогата. User е стандардната улога која ја добиваат сите новорегистрирани корисници и има најопшти привилегии (пристап до main страната).

HTML страни со EJS templating

main.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Main</title>
</head>
<body>
  <center>
    <br/><br/><br/><br/><br/>
    <h1>Welcome <%= user.username %></h1>
    <h3>blah blah blah</h3>
    <br/><br/>
    <a href="/login">Logout</a>
    <!-- ako user ima uloga admin ili owner ke mu prikaze link do admin page -->
    <% if (user && (user.role === "admin" || user.role === "owner")) { %>
      <br/><br/>
      <a href="/admin">Admin Page</a>
    <% } %>
  </center>
</body>
</html>
```

На тие што имаат улога admin или owner им се појавува екстра линк кој води до тајна страна на администратори. Обичните корисници немаат пристап до оваа страна.

adminPage.ejs

```
<br/><br/><br/><br/><br/>
<h1>Admin Page</h1>
<p>This page is accessible only to users with admin privileges.</p>
<br/>
<a href="/main">Back to Main Page</a>

<!-- del za brisenje users koi imaat 'user' uloga (i admin i owner imaat pristap) -->
<br/><br/>
<h3>Delete User Account</h3>
<form action="/delete-user" method="post" onsubmit="return confirm('Are you sure you want to delete this user account?');">
  <label for="deleteUser">Select user to delete:</label>
  <select name="deleteUser" id="deleteUser">
    <!-- Populate the select options with usernames -->
    <% userData.filter(user => user.role !== "admin" && user.role !== "owner").forEach(user => { %>
      <option value="<%= user.username %>"><%= user.username %></option>
    <% }); %>
  </select>
  <button type="submit">Delete User</button>
</form>

<!-- del za promocija i degradiranje ulogi (samo dostapno do owner) -->
<% if (userRole && userRole === "owner") { %>
  <br/></br>
  <h3>Change User Role</h3>
  <form action="/change-role" method="post" onsubmit="return confirm('Are you sure you want to promote this user to admin?');">
    <label for="changeToAdmin">Select user to promote: </label>
    <select name="changeToAdmin" id="changeToAdmin">
      <% userData.filter(user => user.role === "user").forEach(user => { %>
        <option value="<%= user.username %>"><%= user.username %></option>
      <% }); %>
    </select>
    <button type="submit">Promote to Admin</button>
  </form>
  <br/>
  <form action="/change-role" method="post" onsubmit="return confirm('Are you sure you want to demote this admin to user?');">
    <label for="changeToUser">Select admin to demote: </label>
    <select name="changeToUser" id="changeToUser">
      <% userData.filter(user => user.role === "admin").forEach(user => { %>
        <option value="<%= user.username %>"><%= user.username %></option>
      <% }); %>
    </select>
    <button type="submit">Demote to User</button>
  </form>
<% } %>
```

Оваа е страна до која само администратори можат да пристапат. Обичните администратори можат да бришат профили, додека сопственикот (owner) има екстра опции каде може да додава и одзема admin улогата.

server.js

Направени се промени во server.js скриптата со цел да овозможат функционирање на различните улоги. Подоле со прикачени скриншотови со делот од кодот кој е променет со коментари за неговата фукција што ја извршува.

```
const express = require('express');
const session = require('express-session');
const bcrypt = require('bcrypt');
const fs = require('fs');
const path = require('path');

// kreira express aplikacija
const app = express();
app.listen(3000);    // da moze da prima baranja (localhost, porta 3000)
app.use(express.urlencoded({ extended: true }))    // za parsiranje na baranjata
app.use(express.static('public'));   // pristap do html vo 'public' folderot
app.set('view engine', 'ejs');

// go cita json failot so korisnickite informacii
const dataPath = path.join(__dirname, 'databasesim.json');
let userData = [];
let validationCodes = {};

const roles = { owner: 'owner', admin: 'admin', user: 'user', };

try {
   const data = fs.readFileSync(dataPath, 'utf-8');
   userData = JSON.parse(data);
}
catch (err) { console.error(err); }

app.use(session({
   secret: 'mcxedonc43ffdcmef9paa',   // taen kluc
   resave: true,
   saveUninitialized: true
}));

app.get('/login', (req, res) => {res.sendFile(path.join(__dirname, 'public', 'login.html'));});
app.get('/register', (req, res) => {res.sendFile(path.join(__dirname, 'public', 'register.html'));});
app.get('/validate', (req, res) => {res.sendFile(path.join(__dirname, 'public', 'validate.html'));});
app.get('/main', (req, res) => {res.render('main', { user: req.session.user });});
app.get('/admin', (req, res) => {
   // dozvoluva pristap samo na users so admin ili owner uloga
   if (req.session.user && (req.session.user.role === roles.admin || req.session.user.role === roles.owner))
     res.render('adminPage', { userData: userData, userRole: req.session.user.role });
   else res.redirect('/main');
});
```

```
  // zacuvuva informaciite za noviot profil vo userData
  if(username === 'alex') userData.push({ username, email, role: 'owner', password: hash, salt });
  else userData.push({ username, email, role: 'user', password: hash, salt });        // mu dava uloga na noviot korisnik
```

```
req.session.user = { username: user.username, role: user.role };     // za prakanje informacii za userot do ejs datotekata
res.render('main', { user: req.session.user });    //ako e uspesna, zapisuva informaciite vo databazata i prenesuva korisnikot na main.html
```

```
// izbrisi user
app.post('/delete-user', (req, res) => {
   const { deleteUser } = req.body;
   const userIndex = userData.findIndex(user => user.username === deleteUser);

   if (userIndex !== -1) {
     userData.splice(userIndex, 1);
     fs.writeFileSync(dataPath, JSON.stringify(userData, null, 2), 'utf-8');
     res.redirect('/admin');
   }
   else res.send('User not found.');
});

// smeni uloga na user
app.post('/change-role', (req, res) => {
   const { changeToAdmin, changeToUser } = req.body;
   const userToChange = changeToAdmin || changeToUser;

   const userIndex = userData.findIndex(user => user.username === userToChange);

   if (userIndex !== -1) {
     if (changeToAdmin) userData[userIndex].role = 'admin';
     else if (changeToUser) userData[userIndex].role = 'user';

     fs.writeFileSync(dataPath, JSON.stringify(userData, null, 2), 'utf-8');
     res.redirect('/admin');
   }
   else res.send('User not found.');
});
```