2023

WEB DEVELOPMENT (CS 346)

F E E D N I

Raseel Nasser Alkhamees
441021460
Hissah Sharekh Alsharekh
439018813
Najla bander alganawi
439022525

DR.BASMAH ALSOULY

TABLE OF CONTENTS

01 Table of Pages

02 INTRODUCTION

03 LOGIN-REGISTER PAGE

04 LOGIN-REGISTER PAGE STYLES

05 SIDEBAR MENUE

06 SIDEBAR MENUE STYLES

07 MY ACCOUNT

08 Coop,programming,.. HTML

09 coop...javascript

10 backend app.js

11 backend server.js

12 backend package .json

TABLE OF PAGES

01 LOGIN-REGISTER PAGE

02 SIDEBAR MENUE

03 MY ACCOUNT

04 HOMEPAGE

05 CATEGORY

06 PROGRAMMING

07 LANGUAGES

08 COOP TRAINING

INTRODUCTION

FEEDNI is an online platform that allows users to share their knowledge and experiences with others. The website's mission is to bring individuals together to share valuable insights and knowledge in various fields, including programming, languages, and COOP treating.

Upon visiting the website, users can easily navigate through the different sections and explore other users' experience blogs. They can also write and share their own experiences by creating their own blog posts. The website provides a space for individuals to learn from others and share their own experiences, whether they are professionals in their field or just starting out.

The principle of FEEDNI , "Where Experience Gathering" or "Where You Can Find The Experience," highlights the website's goal to gather individuals who are passionate about sharing their experiences and knowledge. The website aims to foster a strong community of individuals who are eager to learn and grow from each other.

Overall, FEEDNI provides a valuable platform for individuals to share their experiences and knowledge with others. The website's user-friendly interface and easy navigation make it simple for users to explore and contribute to the community. By joining FEEDNI, users can expand their knowledge and connect with like-minded individuals who share their passion for learning and sharing experiences.

FEEDNI LINK

by :Raseel
FEEDNI 2023

LOGIN-REGISTER PAGE

login-register page function

THE MOST IMPORTANT FUNCTIONS ARE:

- **hashPassword**() - This function hashes the user's password using SHA-256 encryption before saving it to the database. This is important for securely storing passwords.

This function hashes the user's password using the crypto module. It creates a SHA-256 hash, updates it with the password string, and returns the hashed password as a hexadecimal string.

- **saveUser**() - This function saves a new user object to the database (users array in this case).

It calls hashPassword() to get a hashed password, creates a new User object with the hashed password, and then pushes that user to the users array, saving the user.

- **validateUsername**() - This function validates a username using several checks.

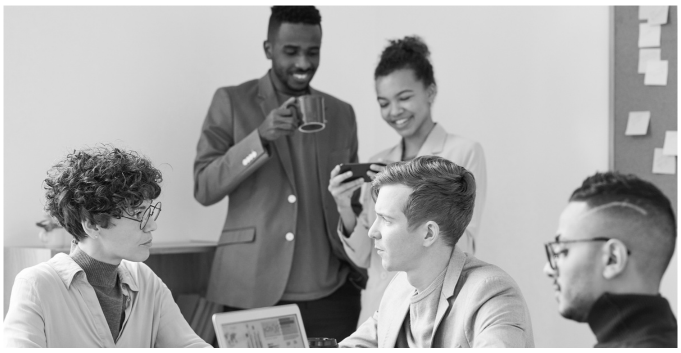
It calls checkUsername() to validate the minimum length and allowed characters. It also calls usernameExists() to check if the username is already taken. It returns a boolean indicating if the username is valid.

- **validatePassword**() - This function validates a user's password using several checks.

It calls checkPassword() to validate the minimum length. It also checks if the two password fields match. It returns a boolean indicating if the password is valid.

hashing the password, saving the user, and validating the username/password are the most important functions from a security and user experience perspective. They allow user to:

- Securely store user passwords
- Save new users to the database
- Ensure users choose valid, unique usernames
- Ensure users choose strong passwords



by :Raseel
FEEDNI 2023

LOGIN-REGISTER PAGE

login-register page function

REST LOGIN-REGISTER PAGE FUNCTION :USER.JS:

- User - The User class stores a user's details
- hashPassword() - Hashes the user's password using SHA-256
- saveUser() - Saves a new User object to the users array
- validateUsername() - Validates the username using several checks
- validatePassword() - Validates the password using several checks

auth.js:

- handleLoginFormSubmission() - Handles logging in a user
- handleRegisterFormSubmission() - Handles registering a new user
- isValid() - Checks if a username/password is valid
- showLoginForm() - Shows the login form
- showRegisterForm() - Shows the register form
- checkLoginState() - Checks if a user is logged in

menu.js:

- toggleNavMenu() - Toggles the navigation menu between normal and responsive mode

sidebar.js:

- toggleSidebar() - Toggles the sidebar open/closed

SO :

- THE USER CLASS STORES A USER'S DETAILS
- HASHPASSWORD() SECURELY HASHES THE PASSWORD
- SAVEUSER() SAVES A NEW USER TO THE DATABASE
- VALIDATEUSERNAME() AND VALIDATEPASSWORD() ENSURE STRONG CREDENTIALS
- THE AUTH.JS FUNCTIONS HANDLE LOG IN/REGISTRATION AND CHECK CREDENTIAL VALIDITY
- THE DOM FUNCTIONS TOGGLE THE MENU AND SIDEBAR

by :Raseel
FEEDNI 2023

LOGIN-REGISTER PAGE STYLES

login-register page styles include : styles for various elements on the page, such as the navigation menu, login and registration forms, input fields, buttons, and error messages.

No. 01 —

.submit:

styles the submit button for the login and registration forms.

No. 02 —

.form__message:

styles the success and error messages that appear on the form.

No. 03 —

.form__input--error:

styles the input fields when there is an error.

No. 04 —

.form__input-erroer-message:

styles the error messages that appear under the input fields.

No. 05 —

.nav-menu.responsive:

styles the navigation menu when it is in responsive mode for smaller screens.

No. 06 —

@media only screen and (max-width: 786px):

styles for smaller screens, such as hiding the navigation buttons and displaying the navigation menu in a different format.

No. 07 —

.input-field:

styles the input fields for the username, password, and other form fields.

No. 08 —

.top span:

styles the text at the top of the form.

No. 09 —

.login-container, .register-container:

styles the login and registration forms.

No. 10 —

.form-box:

styles the container for the login and registration forms.

No. 11 —

.nav-menu ul li .link:

styles the links in the navigation menu.

No. 12 —

.nav:

styles the navigation menu at the top of the page.

No. 12 —

AS:.wrapper:

centers the login and registration forms on the page and sets the background color.

No. 11 —

.nav-menu ul li .link:

styles the links in the navigation menu.

by :Raseel
FEEDNI 2023

SIDEBAR MENUE

sidebar menue function

to adds functionality to a sidebar and checks whether a user is logged in before allowing them to access certain pages. Here is a summary of the code:

- The first two lines of the code use the document.querySelector method to select the button and the sidebar elements from the DOM and store them in the 'btn' and 'sidebar' variables respectively.
- The next three lines of code add a click event listener to the 'btn' element, and when the button is clicked, it toggles the 'active' class on the 'sidebar' element, allowing the sidebar to be shown or hidden.
- The final lines of code check whether the user is logged in by calling the 'checkLoginState()' function. If the user is not logged in, the code saves the URL of the current page in local storage using the 'localStorage.setItem()' method. This allows the user to be redirected to the previous page after logging in.

sidebar menue styles

sidebar menu that provides a responsive and visually appealing interface.

- .user-img: styles the user image in the sidebar.
- .sidebar: styles the sidebar element, including its position, width, background color, padding, and transition effects.
- .sidebar.active ~ .main-content: styles the main content area when the sidebar is active by adjusting its left and width properties.
- .sidebar.active: styles the active state of the sidebar by adjusting its width.
- .sidebar #btn: styles the button that toggles the sidebar visibility.
- .sidebar.active #btn: styles the button when the sidebar is active, moving it to the right of the screen.
- .sidebar .top .logo: styles the logo in the sidebar, including its color, font size, height, and opacity.
- .sidebar.active .top .logo: styles the logo when the sidebar is active by adjusting its opacity.
- .user: styles the user information in the sidebar.
- .bold: styles the font weight of certain text.
- .sidebar p: styles the paragraph text in the sidebar, setting its opacity to zero.

SIDEBAR MENUE

sidebar menue styles

sidebar menu that provides a responsive and visually appealing interface.

- .sidebar ul li a:hover: styles the links in the sidebar when they are hovered over, changing their background color and color.
- .sidebar ul li a i: styles the icon in the sidebar links, including its minimum width, height, and line height.
- .sidebar .nav-item: styles the navigation items in the sidebar, setting their opacity to zero.
- .sidebar.active .nav-item: styles the navigation items in the sidebar when it is active by setting their opacity to one.
- .sidebar ul li .tooltip: styles the tooltips in the sidebar, including their position, color, background, padding, line height, and opacity.
- .sidebar ul li:hover .tooltip: styles the tooltips in the sidebar when they are hovered over, changing their opacity.
- .sidebar.active ul li .tooltip: hides the tooltips in the sidebar when it is active.
- .main-content: styles the main content area, including its position, background color, color, left and width properties, transition effects, and padding.
- .container: styles the container for the main content area, setting its display and justify-content properties.
- .sidebar.active p: styles the paragraph text in the sidebar when it is active by setting its opacity to one.
- .sidebar.active a: styles the links in the sidebar when it is active by setting their opacity to one and adding padding.
- .sidebar.active a:hover: styles the links in the sidebar when they are hovered over, changing their color.
- .sidebar ul li: styles the list items in the sidebar, including their height, width, margin, and line height.
- .sidebar ul li a: styles the links in the sidebar, including their color, display, text decoration, and border radius.

MY ACCOUNT

JAVASCRIPT

IsValideInputs : checks if user inter correct emil, user name and password.
User (First Name, Last Name, User Name , Email, password , Verify Password)
Submit Update User Account
Edit picture change User picture
Naw linked to category page with list of experience contents

STYLE MY ACCOUNT

Box style box design and contents

inputs

Label : style the edit picture button

STYLE HOME PAGE

imegShear : style imge inside box

layout

btn8 a : style the naw button

STYLE MY CATEGORY

btn a : styles for language,

programing, cooptrining.

LANGUAGES

COOP TRAINING

PROGRAMMING

HTML+CSS+JS

THIS PAGE WAS DESINGED BY HISSAH
AND JAVASCRIPT .

by :Hissah
FEEDNI 2023

LANGUAGES

COOP TRAINING **JAVASCRIPT**

PROGRAMMING

No. 01 – shareExperienceBtn.addEventListener("click", () => {...}):

This method adds an event listener to the "Share Experience NOW!" button. When the button is clicked, it checks if the user is logged in by checking the value of the "isLoggedIn" key in local storage. If the user is not logged in, it displays an alert message and asks if the user wants to go to the login page. If the user is logged in, it sends a POST request to a "/share-experience" endpoint with the experience data in the request body. If the request is successful, it displays an alert message saying the experience was shared and clears the experience input field.

No. 02 – form.addEventListener('submit', e => {...}):

This method adds an event listener to the search form. When the form is submitted, it prevents the default form submission behavior and gets the values of the search input, language select, and city select elements. It then sends a GET request to a "/api/experiences" endpoint with the search options as query parameters. If the request is successful, it renders the results by calling the renderCommentCard() function for each result.

No. 03 – function renderCommentCard(result) {...}:

This function takes a result object as a parameter and creates a new comment card element with the title and text properties of the result object. It then appends the card to the comment container.

by :Raseel
FEEDNI 2023

BACKEND:

WE HAVE THESE FILES :

app.js

server.js

package.json file

app.js

- **function route(page, res){...}:** This function takes a page (a string representing a file path) and a response object as parameters. It reads the content of the file at the given path and sends it as the response to the client.
- **const server = http.createServer((req,res)=>{...}):** This function creates an HTTP server that listens for incoming requests. When a request is received, it uses a switch statement to route the request to the appropriate page by calling the route()function. It also includes a route for a POST request to "/share-experience", which currently only responds with a 200 status code.
- **fetch('http://localhost:3000/share-experience', {...}):** This code sends a POST request to the "/share-experience" endpoint of the server running on localhost:3000. The request includes a JSON body with an "experience" property.
- **fetch(/api/experiences?search=${searchTerm}&lang=${lang}&city=${city}):** This code sends a GET request to the "/api/experiences" endpoint with query parameters for search term, language, and city. If the request is successful, it renders the results by calling the renderCommentCard() function for each result.
- **app.get('/',function(req,res){...}):** These functions define routes for various pages using the Express.js framework. Each route responds with the contents of a specific file.
- **function register(username, password) {...}** and function login(username, password) {...}: These functions are not defined in the code provided. However, they appear to be related to user authentication and registration. The register() function sends a POST request to a "/register" endpoint with the username and password in the request body. The login()function sends a POST request to a "/login" endpoint with the username and password in the request body, and returns a Promise that resolves with the response body as JSON.

BACKEND:

server.js :
This is a Node.js server that uses the Express.js framework and the Mongoose library to handle HTTP requests and interact with a MongoDB database.

server.js functions :

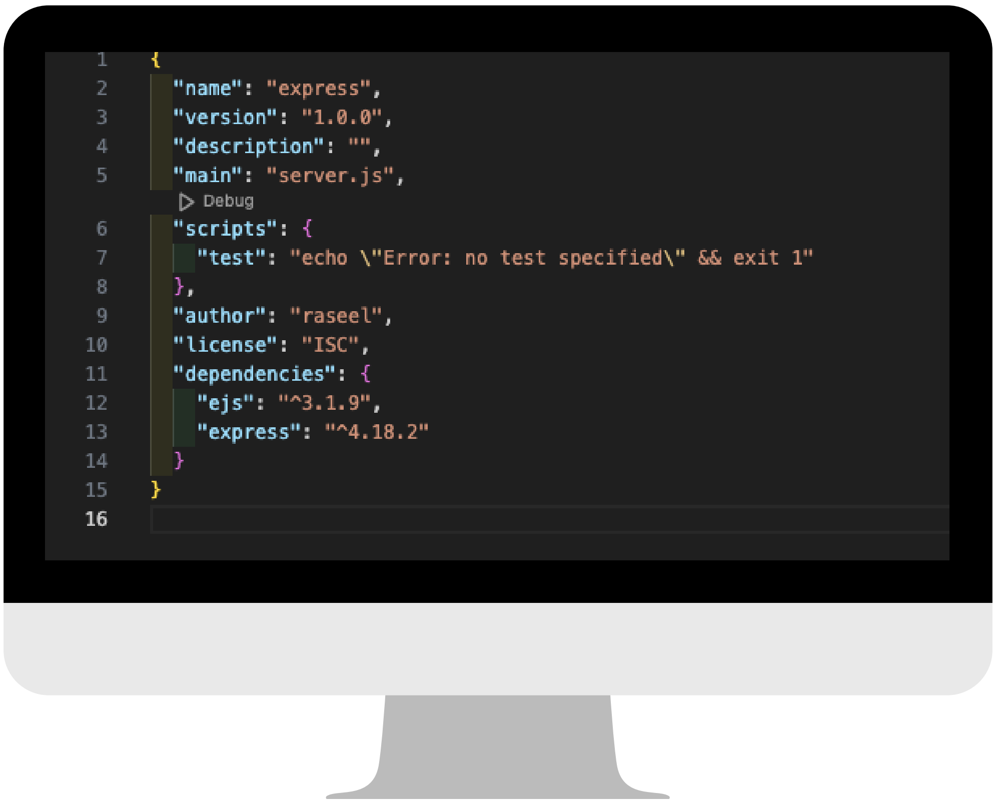
- **const express = require('express') and const app = express():** These lines import the Express.js library and create an instance of an Express app.
- **const mongoose = require('mongoose') and mongoose.connect('mongodb://localhost:27017/experiences'):** These lines import the Mongoose library and connect to a local MongoDB database named "experiences".
- **const experienceSchema = new mongoose.Schema({...}):** This code defines a Mongoose schema for an "Experience" object, which has two properties: "content", which is a string that holds the content of an experience, and "createdAt", which is a Date object that holds the creation date of the experience.
- **const Experience = mongoose.model('Experience', experienceSchema):** This code creates a Mongoose model for the "Experience" schema, which allows us to interact with the database as if we were working with objects.
- **app.post('/share-experience', (req, res) => {...}):** This code defines a route for handling POST requests to the "/share-experience" endpoint. When a request is received, it extracts the experience content from the request body, creates a new "Experience" object using the Mongoose model, and saves it to the database. If the save operation is successful, it sends a 200 status code to the client. If there's an error, it sends a 500 status code.
- **app.listen(3000, () => {...}):** This code starts the server and listens for incoming requests on port 3000. When the server is started, it logs a message to the console.

BACKEND:

package.json file :
is used for managing dependencies and other metadata for a Node.js project.

package.json file information

- **name**: The name of the project.
- **version**: The version of the project.
- **description**: A brief description of the project.
- **main**: The entry point for the project. In this case, it's server.js.
- **scripts**: A set of commands that can be executed with npm run <scriptname>. In this case, there's only one script called "test" that echoes an error message.
- **author**: The name of the person who created the project.
- **license**: The license under which the project is distributed. In this case, it's the ISC license.
- **dependencies**: A list of dependencies that the project requires to run.



```json
{
  "name": "express",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "raseel",
  "license": "ISC",
  "dependencies": {
    "ejs": "^3.1.9",
    "express": "^4.18.2"
  }
}
```

by :Raseel
FEEDNI 2023