ENSF-381: Assignment 5

Winter 2024

Department of Electrical and Software Engineering

Schulich School of Engineering

Due: April 08, 2024, at 11:59 PM

Congratulations on reaching the final assignment of our Full Stack web development course! Throughout this journey, you have learned the essential tools and technologies needed to build modern web applications, from crafting interactive user interfaces with HTML, CSS, and JavaScript, to developing dynamic front-end functionality with React. For our final assignment, we will shift our focus to the backend of our E-commerce website. Here, you will have the opportunity to explore the foundational principles that underpin the server-side logic of web applications, gaining insights into data management, authentication, and business logic implementation.

Administrative Details

- The submission should be as a single compressed file (.zip) which includes a complete source code and resources (e.g., images) ready for execution without modifications. The name of the submitted file would have the following format: "Assignment#_ ENSF381_Section#_Group#". For example: "Assignment5_ENSF381_L01_Group01".
- When working in a group, submit ONLY ONE copy; i.e. do not submit 2 separate copies.
- Please mention the group member's name and UCID at the top of your code as source code comments in the index.js.
- Do not add any additional styling, functionality, or components if you haven't asked in the assignment. You must name the component as specified in the assignment.
- Create a text file named "readme.txt" in the project directory. List all the libraries you used/imported in both React and Python project. Once listed, provide the necessary commands to install these libraries using npm (e.g., npm install styled-project) or pip (e.g., pip install flask). This will ensure that all required dependencies are installed, and your project can run smoothly in the marker's environment.
- The solution of Assignment 4 will be available on March 27. You can find the solution in the GitHub repository.

Login page Requirements: 40 marks

- Create the login page layout with React components.
- The login page will initially display a login form with fields for **Username** and **Password**.
- There will be a signup form with fields for **Username**, **Password**, **Confirm Password**, and **Email**.

- Implement functionality to switch between the login form and a signup form.

Components and Structure:

- **Header:**
  - Reuse the Header component created earlier in the Assignment 04 and place it at the top of the login page for consistency.
- **LoginForm:**
  - LoginForm component include **Username** and **Password** input fields. Also, the component contains **Login** and **Switch to Signup** buttons.
  - Clicking the **Switch to Signup** button will display the **SignupForm** that you will create in the next step.
  - Ensure that both the username and password fields are not empty before submitting the form.
  - Add placeholder text for the **Username** and **Password** fields to provide user guidance.
- **SignupForm:**
  - SignupForm component includes input fields for **Username**, **Password**, **Confirm Password**, and **Email** along with **Submit** and **Switch to Login** buttons.
  - Clicking the **Switch to Login** button will display the **LoginForm.**
  - Ensure that none of fields are not empty before submitting the form. Also, make sure the entered values in the **Password** and **Confirm Password** are identical before submitting the form.
  - Add placeholder text for **Username**, **Password**, **Confirm Password**, and **Email** fields to provide user guidance.
- **Footer:**
  - Reuse the Footer created earlier in Assignment 04 and place it at the bottom of the login page.

Now, implement the **LoginForm** and **SignupForm** components.

Assembling Login Page Component:

- Inside the src/components directory of your React project, create a new file named LoginPage.js.
- In the LoginPage.js file, import the Header, LoginForm, SignupForm, and Footer components to compose the structure of the login page.
- Implement functionality so that clicking the **Switch to Signup** button displays the **SignupForm** and **Switch to Login** button displays the **LoginForm.** The page will display the **LoginForm** by default.

The structure of the LoginPage will have the following:

```
<div>
    <Header />
    <LoginForm />
    <Footer />
</div>
```

Integrating Login Page into the Project:

- In the App.js file, import the LoginPage component and render the LoginPage component within the App component.

Backend Setup and Data Storage: 10 marks

Create and initialize Flask App:

- Create a new directory named “Backend”.
- Navigate to the “Backend” directory and initialize a Flask app in the directory by create a Python file named “app.py”.

Backend-frontend Integration: 50 marks

API Endpoints:

- Define API endpoints to handle user registration and authentication.
- **User Registration API:**
    - When a **SignupForm** is submitted, make a POST request to this API endpoint. The body of the request will contain the entered values in the **Username**, **Password**, and **Email** fields of the **SignupForm.**
    - Create a Python list of JSON named **“users”** in your “app.py” file, where each element in the list represents a user’s “username”, “password”, and “email”. **“users”** list is empty at the beginning, once the the users sign up, the list should be updated.
    - Validate if the **Username** already exists in the **“users”** list. If it exists, display an error message in the frontend, otherwise store the **Username**, **Password**, and **Email** in the list.
- **User Authentication API:**
    - When the user clicks the **“Login”** button in the **LoginForm,** make a POST request to the API endpoint. The body of the request will contain the entered values in the **Username** and **Password** fields of the **LoginForm.**
    - Validate the **Username** and **Password** sent from the Frontend against the stored “username” and “password” in the **“users”** list in the Backend.

- If the entered **Username** and **Password** are correct, redirect the user to the Product page. Otherwise, display an error message stating that the entered username and password are incorrect.

- **Product API:**
  - Create a list named **"products"** in the "app.py" file. Copy the following contents into the "**products**" list:

```
products = [
  {
    "id": 1,
    "name": "Product 1",
    "description": "Description for Product 1",
    "price": 10.99,
    "image": 'images/product1.png'
  },
  {
    "id": 2,
    "name": "Product 2",
    "description": "Description for Product 2",
    "price": 20.99,
    "image": 'images/product2.jpg'
  },
  {
    "id": 3,
    "name": "Product 3",
    "description": "Description for Product 3",
    "price": 10.99,
    "image": 'images/product3.jpg'
  },
  {
    "id": 4,
    "name": "Product 4",
    "description": "Description for Product 4",
    "price": 10.99,
```

```
    "image": 'images/product4.jpg'
},
{
    "id": 5,
    "name": "Product 5",
    "description": "Description for Product 5",
    "price": 10.99,
    "image": 'images/product5.jpg'
},
{
    "id": 6,
    "name": "Product 6",
    "description": "Description for Product 6",
    "price": 10.99,
    "image": 'images/product6.jpg'
},
{
    "id": 7,
    "name": "Product 7",
    "description": "Description for Product 7",
    "price": 10.99,
    "image": 'images/product7.jpg'
},
{
    "id": 8,
    "name": "Product 8",
    "description": "Description for Product 8",
    "price": 10.99,
    "image": 'images/product8.jpg'
},
{
    "id": 9,
```

```
    "name": "Product 9",
    "description": "Description for Product 9",
    "price": 10.99,
    "image": 'images/product9.jpg'
  },
  {
    "id": 10,
    "name": "Product 10",
    "description": "Description for Product 10",
    "price": 10.99,
    "image": 'images/product10.jpg'
  }
]
```

- Define product API endpoint to get the product information. Make a GET request to the API endpoint. The response will contain the product information stored in the **"products"** list.
- Now, we will make the following changes to the **"ProductList"** component in the frontend that we have implemented in Assignment 04. **So, the product data will be provided from the backend ONLY.**
  - In **"ProductList"** in frontend**,** we imported the product information from the "products.js" file stored in the "src/data" directory.
  - Delete the "products.js" file stored in the "src/data" directory.
  - Make necessary adjustment to the **"ProductList"** component so that the product information will be retrieved from the response of the GET request to the product API endpoint.

Also, you need to make the following changes to the routing functionality of the frontend:

- If the user attempts to access the product page directly via the URL (e.g., "localhost:3000/products") or by clicking on navigation links without entering the correct username and password, the application should redirect the user to the login page.
- If the user is already successfully logged in, clicking on 'Products' in the navigation section will redirect them to the **Productpage**.

See **"loginpage1.gif"** and **"loginpage2.gif" (**attached with the assignment) for a visual representation of the interactive Login Page.