ICSI 518 – Software Engineering

Assignment 4 - JWT Authentication & Dockerization

Due: Sunday Nov 17, 2024, 23:59

Objective: Enhance the existing e-commerce product page application by adding user registration, login, and profile management features, while using Docker for containerization and JWT for secure authentication. This assignment provides hands-on experience with user authentication, containerization, and managing user data.

Requirements:

1. **User Registration**: Create a registration page where new users can sign up.
   - Include fields for Name, email, and password.
   - Implement form validation for all fields and ensure the password meets security requirements.
   - On successful registration, store user information (in database).
2. **User Login**: Create a login page for users to authenticate.
   - Include fields for email and password.
   - Validate user credentials against the registered users.
   - On successful login, generate a JWT and store it in local storage to maintain the user's authentication state.
   - Redirect the user to home page/product list page upon successful login.
3. **JWT Implementation**: Use JWT for user authentication and session management.
   - On login, validate the credentials, generate a JWT that includes user information and a secret key and return the JWT.
4. **Profile Page**: Create a user profile page that displays user information.
   - Show the Name and email of the logged-in user.
   - Provide options to update the user’s profile information.
   - Restrict access to the profile page to authenticated users only. Display only the current user's data based on the JWT.
   - Implement a logout feature that clears the JWT from local storage and redirects to the login page.

5. **Modify Cart and Wishlist Functionality**
   - Ensure that only authenticated users can add items and view or manage their cart and wishlist.
   - If a non-authenticated user attempts to add items or access the cart or wishlist, redirect them to the login page with a relevant message.
   - Display the cart and wishlist contents specific to each authenticated user, so users see only their own saved items.
6. **Dockerization and Caddy Reverse Proxy**
   - Build the React application using the command ` npm run build` . This will create a production-ready build of your React app in the build folder.
   - Copy the contents of the build folder (generated by npm run build) into the public folder of your Express application. This setup enables the Express server to serve the static files of the React app, ensuring that both the Express API and the React application are accessible through the same port.
   - Create a Dockerfile for the application and use Docker Compose to orchestrate the backend, MongoDB, and Caddy services.
   - Set up the MongoDB container with a volume to persist data.
   - Configure Caddy to act as a reverse proxy, routing requests to the appropriate services. Caddy should handle SSL termination to enable secure HTTPS connections and manage request routing efficiently.
   - Ensure it installs dependencies and starts the server on the specified port (e.g: 5000).
   - Ensure that the configuration allows the backend to connect to the MongoDB container seamlessly.
7. **Database Setup (Local MongoDB with Docker)**
   - Use a local MongoDB instance set up within a Docker container instead of MongoDB Atlas for database management.
   - In the Docker Compose file, define a MongoDB service. Expose necessary ports (e.g., 27017) to allow the backend to connect.
   - Update the backend to connect to the local MongoDB instance by using the container's service name as the host (e.g: mongodb://mongodb:27017/db_name).
   - The application should connect to this local MongoDB instance for all data storage and retrieval operations, including products, cart, and wishlist data.
   - Use initialization scripts to configure predefined collections(Products) in MongoDB.
8. **AWS Deployment**

- Deploy the application using docker compose on the EC2 instance.
- Use PM2 to run the Node.js server and ensure the application runs continuously.

Docker and Caddy Reference: Link

Submission Instructions:

- Submit your project via GitHub, including a README file.
- In the README file, provide detailed instructions on starting the Docker container and accessing the application. Include the web address of the deployed site (with port number), the IP address of the EC2 instance, and a link to your GitHub repository.
- The client-side (React) and server-side (Express.js) source code should be organized into separate subdirectories within the same main directory in the Git repository.
- If you use any AI assistance to generate code or components, make sure to clearly indicate it in your project with appropriate comments.

What to Submit to Brightspace:

Download your GitHub repository as a ZIP file and upload it to Brightspace.

Grading Rubric:

Category	**Criteria**	**Max Marks**
User Registration	Form includes fields for username, email, and password, with client-side validation. User information is successfully saved to MongoDB upon registration.	10
User Authentication	Form includes fields for email and password with client-side validation.Successful authentication against MongoDB users and JWT generation. JWT stored securely in local storage; user redirected to homepage upon login.	10
JWT Implementation	JWT contains user information and is signed with a secret key.	5
Profile page	Displays username and email of logged-in user. Profile update functionality, with changes saved to MongoDB. Logout feature clears JWT and redirects to login page.	10
Cart and Wishlist Functionality	Only authenticated users can add/view/manage their cart and wishlist. Non-authenticated users redirected to login page with appropriate message. Cart/wishlist content is displayed specifically to the authenticated user.	10
Dockerization Setup	Use Docker Compose to orchestrate the backend, MongoDB, and Caddy services. Configured to expose necessary ports.	15
Caddy Reverse Proxy	Set up Caddy as a reverse proxy to enable HTTPS connections.	10
Database Setup with Local MongoDB	MongoDB container properly configured and running locally within Docker.	10
Deployment	Application deployed on AWS EC2.	10
At Least 5 Commits	At least 5 commits in the GitHub repository (more commits are better).	3
.gitignore File Exists	.gitignore file exists and includes node_modules.	1
ReadMe File	ReadMe file contains clear instructions.	6
Total		100