Andrew Brown

October 6, 2024

Revised: October 16, 2024

CS 499

Databases: Milestone Four

The artifact for category three is from the CS465 course called the Travlr Admin Panel. The artifact is a full-stack application created in August of 2024 to manage the traveling packages on their website. It contains a back-end server to handle API calls for the user to login as well as add and edit trip data that is displayed for a consumer. For the front-end side, an administration section built using Angular allows a user to log in and add additional trips or make changes to the existing trips.

The artifact was selected since it can demonstrate two of the course outcomes related to security and allowing multiple users in an organization to collaborate with the trips that are displayed to a consumer. Before the enhancements were made to the artifact, a user could register an account without being logged in first, which is a security risk, and this would give them unrestricted access to modify or add new trips to the website. Another issue with the current state of the artifact is a user has no way to update their password or profile information. The artifact was enhanced by adding a profile page for the user to update their account information and password. Another page was added to accept new registrations, which requires the user to be logged in first. In the code files, a changelog file was added to show each change that was made in the project.

I employed strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science by adding a profile section for the user to make updates to their password or account information. The user can also register new accounts in case another user needs access to edit or add trips to the website.

Angular uses components for both page displays and separate sections within the page. To create the profile section, a new component was created in the project using the built-in Angular command: "ng generate component account" and then Angular-cli will add the boiler-plate files for the component. The component is added to the route list so it can be accessed in the browser. The component has a new form added to it, along with all methods to handle the actions within the form, such as submitting and displaying form errors.

The front-end portion for updating the account information is mostly complete at this stage, but the back-end would need to be modified to accept the API requests from the front-end. In the controller for authentication, an additional method was added to accept and validate the profile form data and update the database accordingly. Once the API requests are created, the front-end application can be modified to send the form data to the back-end server. The standard practice for sending data to an API in Angular is to use a service that is injected into the component. The project already had a service setup, so the only change needed was to add the additional code for sending the request to the back-end API, which completes both the front-end and back-end portions.

The process for creating the page to handle new user registration was almost the same as creating the page to update the profile. A new component was created and added to the route list, and an identical form code was placed in the new component. The back-end already had code written to handle new user registration, and the service in Angular had unused code for sending the form data to the API, leaving the only change to inject the service in the component to complete the process.

I developed a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources by implementing security controls to prevent other users or guests from modifying another user's profile information. The same security control is added to prevent a guest user from creating new accounts. For both cases, the user is now required to be logged in with a security token that is generated from the server. If the user is not logged in, they will now get an error if they try to register a new account or update their profile information.

When adding the enhancements to the artifact, there were a few discoveries. The first one is figuring out how the authentication service handles reading the token so it can keep track of which user is making the request. Once that is figured out, the enhancement of updating the profile information can be added. The second discovery was making sure the mongoose driver was updating the correct record. At first, the record would appear to be updated, but it wasn't taking the correct input for the user that was logged in, and this would create a security issue. After revising the code to only use the ID from the user's token, the security bug was fixed, and the correct record was updated.