Wilson Quilli

Professor Nalubandhu

SWENG 861: Software Construction

February 6th, 2026

Wilson's Week 4 Assignment Frontend Report

My project uses the React framework for frontend. I chose React, because it makes building interactive websites easier by breaking the user interface into components. Instead of rewriting the same code again, React allows developers to reuse it like navigation bars and footers. This makes the application easier to manage and update. I've also had experience using React in the past, so I'm familiar with it.

All of the frontend code is organized in a folder structure for each section of the frontend. A page folder that represents full screens of the application, while reusable UI elements like navigation bars, and footers are stored in a components folder. API code, like functions that send requests to the backend server, is kept separate so that communication logic does not mix with visual layout code. This structure helps keep the project clean and makes it easier to understand where each part of the application belongs.

Each route in the application maps a specific URL path to a page component. For example, one route displays the login page, while another route displays the main dashboard or home page, once a user is logged in. Public routes, such as the login page, are accessible to anyone visiting the site. Protected routes, such as pages that display user data, require the user to be authenticated. If a user tries to access a protected route without being logged in, they are redirected back to the login page. This helps prevent unauthorized access to sensitive information. Just like in my backend report, users will have to Login in with their Google Accounts into Digital Doggy.

Authentication in the application follows a simple login flow. When a user enters their username and password and submits the login form, the frontend sends information to the backend API. The backend checks the credentials and, if they are valid, returns an authentication token representing the user's logged-in session. The frontend stores this token and uses it to confirm the user's identity for future requests. Once logged in, the user is allowed to access protected parts of the application.

API integration is handled through a centralized API client. This client automatically attaches the authentication token to every request sent to the backend. By doing this, the server can verify that the request is coming from an authorized user. If the server responds with a 401 (Unauthorized) or 403 (Forbidden) error, it usually means the token is missing, invalid, or expired. When this happens, the application handles the error by logging the user out or redirecting them to the login page so they can authenticate again. This keeps the app secure and prevents broken sessions.

The application also includes proper handling for asynchronous actions such as loading data or submitting forms. When the app is waiting for a response from the server, it shows a loading state so the user knows something is happening. If the request is successful, the app displays the returned data or confirmation message. If an error occurs, the app shows a clear error message instead of failing silently. These loading, success, and error states improve the user experience and make the application feel more responsive and reliable.

Sensitive information such as API keys and secrets are not stored directly in the frontend code. Instead, they are kept in environment variables (.env file) that are not uploaded to GitHub. This prevents private information from being exposed publicly. Authentication tokens are handled carefully and only sent over secure connections. The app also avoids common security risks like cross-site scripting by not directly rendering unsafe user input into the page.

I also implemented some of the bonus features such as accessibility and internationalization, allowing the app to support both English and Spanish, with the change of a button. Accessibility improvements, such as readable text, clear navigation, and proper labeling, help make the app usable for a wider range of users. Overall, this project demonstrates a well-organized frontend application that uses React to build reusable components, client-side routing for smooth navigation, secure authentication with API integration, and proper handling of asynchronous actions and errors.