Logbook Application - Milestone 2

Introduction

Cloud Log is a web application that will allow skydivers to log their jumps and view their jump history. This project was created to address the need for a simple, easy to use web app that allows skydivers to log their jumps and view their jump history. The core features of the api are completed, and in this milestone I focused on protecting the api by adding authentication and authorization. I onboarding Google OAuth 2.0 for authentication with the apis and added CORS policies to ensure that the api is only accessible from the frontend that I am hosting.

Status of the Project

The project underwent a lot of redesigns. The initial design was to use my own authentication system. Weighing the risks of managing user accounts, I realized how susceptible the system would be to attacks. I also did not see it as a smart idea to store sensitive user data like their email and password. Using a third party authentication service would also create higher trust from the user. The authentication system currently works for direct calls to the api, where you can authenticate in the browser. User information is stored with AWS DynamoDB with Google's unique user id. The database is secured with a user managed key. This key costs about 0.04 USD a day, so I have turned it off for now.

Additionally, CORS policies were added to the Logbook and Authentication services. This helps protect the api from unknown sources, and Google OAuth is also given allowed access. The only account that can currently authenticate is my own. In addition, I also ensured that any key that is used is stored in a secure location. The keys are stored in a file that is not tracked by git. The key files are read in in the `LogbookService.Settings.ProjectSettings`

To be Completed

The next step is to implement the frontend. I have already started working on the frontend to test the authentication system. I am experiencing a CORS policy issue when redirecting to the Google OAuth 2.0 page. Even though the frontend is also listed as an authorized domain, the issue persists. I believe the best approach to solving this is to make the backend handle the redirect with a /login endpoint, otherwise return a 403 error. Once the frontend can authenticate, I will be able to create a user interface for the user.

I have two design considerations for implementing the frontend authentication.

1. Have the frontend send a login request to the authentication service.
2. Have the frontend send a login request to the Logbook Service.

I think the second option is the best approach. This will make it more modular and easier to add new authentication providers in the future. If I can't solve the CORS Policy issue, I will have to use the first option.

Timeline to Completion

In its current state, my primary goal is to have the authentication system working in full by the time of the deadline of December 11th. If this is achieved sooner, I will build the functionality to list, create, edit, and delete jumps using the interface. After that, I will work on improving the styling of the page and try polishing up the application.