Logbook Application - Milestone 1

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 application that allows skydivers to log their jumps and view their jump history. As I started this project at the start of this semester, there was a lot of building blocks that needed to be put in place. In this milestone, I worked on setting up the project, creating the database, creating the API for logging, editing, listing, and deleting jumps, and implementing the backend for handling the API calls. For the milestone, this is on track with the project plan. I also currently have a proof of concept symmetric encryption algorithm that will encrypt the public fields of a jump object to be stored in the database.

Status of the Project

The project has a working API that can be used to log, edit, list, and delete jumps. The API is currently not protected by any authentication, so anyone can use the API to log jumps, even if the jumps do not belong to them. The structure of the database is also finalized. I used DynamoDB as the database for this project. DynamoDB is a NoSQL database that is designed to be scalable and fast. I have only worked with it once during my internship, and never worked on it in C#. Setting up the database was the most time consuming part in this milestone. I had to learn how to create tables, connect to the database in a Docker container, and how to use the DynamoDB SDK to interact with the database.

In this milestone, I met my goal of creating a working API that can be used to log jumps and also create an encryption service that will encrypt the public fields of a jump object to be stored in the database. This will help protect the privacy of the skydivers jumps and potentially revealing information in the event someone is able to retrieve a table from the database. I also discovered a new way to protect the API from revealing information about the structure and implementation of the codebase. In the beginning of the project, I was returning Exceptions to the client when an error occurred. Reflecting on this, I realized if a malicious agent wanted to find out how the API was implemented to exploit it, they could just send requests to the API and record the responses. The solution that I implemented to protect the API from revealing this information was to catch the exceptions and return a generic error message to the client. If there was a specific error that I wanted to return to the client, I would create a custom exception and return a pre-built error message giving better detail on what the client did wrong. For any other exceptions that are not custom, I would return a generic error message. By returning a custom error message for any exception I did not account for, I can further help protect the API from malicious agents.

Milestones yet to be Completed

- Milestone 2
    - Incorporate authentication into the API
    - Create new API Controller that can handle the authentication
    - Ensure passwords for the account are strong (and encrypted)
    - Ensure user can only access information that belongs to them (and they are authenticated to receive)
- Milestone 3
    - Create a user interface to access the API
        - Login / Sign Up Page
        - Log Jump Page
        - View Info Page
    - Develop multi-factor authentication (stretch goal)

Proposed Timelines to Accomplish the Remaining Milestones

Milestone 2 I am allotting 3 weeks to complete. For milestone 3, since I am very unfamiliar with frontend frameworks and I have a desire to attempt to implement multi-factor authentication, I am allotting 3-4 weeks to complete.