**Title**:

Secure and Efficient Access Management

**Problem:**

Username and password authentication continues to be an integral part of the day-to-day operations of many businesses. There is a need for an efficient and scalable way to store individual users and the access levels they should have.

The data storage method needs to be secure, as an easily compromised password will compromise the use case of this data storage method, and in extreme cases, lead to severe cybersecurity incidents.

At the same time, the data lookup needs to be efficient. Users will likely need to authenticate on at least a daily basis, and poor performance in this workflow will quickly decrease user satisfaction. In cases where the business grows, this also needs to be scalable to handle large increases in user volume.

**Primary Stakeholder**:

There are likely many stakeholders at a leadership/administrative level. Those with most riding on the business (example: C-level executives) will want to ensure their own documentation and operations remain secure. Those working in the technical operations of the organization will also be important to consider (example: sysadmins, Security leadership) as they will be responsible for keeping the underlying infrastructure secure and running.

However, every individual at a business/organization utilizing this functionality will ultimately have a say in this process. Their best practices will ensure the process remains secure, and that their individual accounts do not get compromised.

**Graphical Interface:**

The two core user interfaces would be a login page and a registration page. The login page would authenticate against existing data and the registration would create new entries. There is room for flexibility here on how many data elements we are collecting and the overall look and feel. Below are some sample designs, with the account creation also storing an email address.

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, Teams

Description automatically generated

**Data:**

A hash data structure will allow both efficient lookup and secure storage of certain values and using a node to store those in combination with a lookup key will likely achieve our core workflow. This would be taking a bucket approach to storing index positions.

A base-level mockup would store this information in the following way:

* Key: Username (hashed + indexed for lookup purposes)
* Value: Password (private variable in the node and hashed from user input)
  + Helper methods should only compare a user input to the value, but not allow the raw hash value to be pulled

Our method for handling collisions can be flexible, but we will likely achieve the best performance results by storing as a LinkedList.

A hash data structure will also allow flexibility with resizing. An easy way we can do this is by setting a load factor threshold to know when we need to expand our number of indices available.