**Repository:** https://github.com/istan18/COMP6841-Something-Awesome

**URL** (takes minutes to boot up) https://password-manager-6841-something-awesome.onrender.com/

Project Name: **Password Manager**

Christian Tolentino, H12A, z5420628, COMP6841

Over the course, I have developed a strong understanding of the importance of handling data securely and verification/authentication techniques  but I've always wondered how are these used in a real world application?  Hence, for my Something Awesome project, I created a password manager in an attempt to learn about the technical side of authentication and verification as well as figuring out the specific use cases of master passwords, encryption, decryption, keys and secrets. I have attempted to add as many security features as I could to this application and in the future I would love to add many more. Ideas I have are account activity logging, OAUTH, security questions and authentication apps.

Tech Stack:

- **Backend:** Express.js
- **Frontend:** React.js
- **Database:** MongoDB (no SQL injections here!)
- **Notable libraries and services (in the context of security):**
    - **crypto.js** - for generating encryption and decryption keys for passwords (PBKDF2) stored within the password manager
    - **cors** - manages which routes are allowed to send requests to backend
    - **twilio** - for 2FA via SMS
    - **nodemailer -** for 2FA via email
    - **dotenv** - to store secrets and hidden keys
    - **jsonwebtoken** - to generate tokens for user authentication
    - **bcrypt** - for the hashing of passwords
    - **react-google-recaptcha -** Google's API for ReCaptcha
    - **secure-random-password** - for password generation
    - **render -** a cloud service that provides DDoS protection and TLS certificates

Features/Security features:

- Simple register, login and logout capabilities with tokens used to handle user sessions
- 2FA via SMS and Email (both are required)
- Ability to add, delete or edit passwords with name and optional image
- Limited password attempts until lock out
- Password generation creation via a secure algorithm (criteria is minimum 2 uppercase, 2 lowercase, 1 number, 1 special character, character length greater than 8)
- ReCAPTCHA (on hardest difficulty via Google's API)
- Pinpad authentication upon login - users must reinput their passcode upon minutes of inactivity
- All sensitive information (passwords/passcodes) are encrypted via bcrypt or PBKDF2
- Master password (used for login) is used to decrypt user's managed passwords
- DDoS protected web service secured by a TLS certificate

Refer to the GitHub repository for relevant frontend/backend routes.

**Repository:** https://github.com/istan18/COMP6841-Something-Awesome

**URL** (takes minutes to boot up) https://password-manager-6841-something-awesome.onrender.com/

## Results

I finished an MVP (with probably a lot of bugs) of a password manager. Along the way, I learned a lot about certain JavaScript libraries that provide a clear and easy abstraction of security tools so that developers can easily use them. Users should be able to use my application to securely store passwords and know well that it would be very hard for an attacker to gain access. Even if someone were to gain control of my MongoDB account (by hacking into my gmail account or by knowing the MongoDB URI, which is stored as a secret), they would not be able to gain access to any passwords without 1. Knowing the master password for each user (which is encrypted) and 2. Knowing the secrets. On a non-security note, I refreshed my Type/JavaScript knowledge in the context of Express, React as well as HTML and CSS.

## What I did

80% of my time spent on this project was spent researching, googling and reading documentation. I didn't really get heavily started on my project until Week 5 where I spent around 5-8 hours each week. During the period of Week 5-7, there were a lot of assignments and midterm exams and other commitments in my life that were taking up my time, so the workflow was quite inconsistent and sporadic. I spent Week 5, mostly setting up the application (MongoDB, backend, frontend). I also did research and planning on what features I want to do, what features seem easy to do, which features seem hard to do, which features that I could do if I had enough time and what features that I needed more knowledge in. Once those were decided, in Week 6/7, I started on some backend routes for my prioritised features, then tried sending a request to those backend routes via my frontend (that was initially in plain HTML, CSS). This was repeated about 3 or 4 times until I was finished. In week 8, to polish my work, I decided to make a quick design on a few pages via figma and then coded that up into my frontend. This was not a big priority since that was not the main focus of my project. I then cleaned up the code a little bit, added a few linters and deployed it to a reliable cloud service.

## How I was challenged

I think the hardest thing for me was understanding how to store the passwords within the database. Can I store it as plaintext? No, then it wouldn't be secure. Can't I encrypt them? But how would users see the passwords in the application? Can't I generate a key? Well, how is it generated? What algorithm do I use? AES, SHA256, MD5, something else? Well it must be a two way function since the passwords must be all hashed the same way. Also the key must be the same each time. Do we store the key in the database? No, then that's not secure. Do we store it in the client? LocalStorage? Within state variables? How are they passed to the backend? These were questions that I had to ask myself and the answers to them I still don't really know but with enough research I was able to make an educated guess on them. Other main challenges were understanding different documentation and other common coding problems (not understanding syntax, type errors, logic errors). For example, navigating the twilio library was confusing, I kept getting the same error codes and their library was so huge. The Google ReCaptcha dashboard was also difficult to navigate and set up. This was my first time making a relatively bigger project on my own from scratch, so I did not know how to dedicate my time accordingly and ended up cutting some features that I wanted to include. The only thing I would change is to start earlier so I am under less time pressure to implement things leading to higher code quality. There were also a lot of secrets and keys I had to store so managing them was quite difficult.