Alli Garcia (Individual Project)

Student Information

- garciaa12@mymail.nku.edu

Planning the Individual Project

Summary of the Project

Imagine a secure vault where you keep your valuable information, like keys to your house. This product is like that vault, but for your online accounts. You know how you have to remember different passwords for your email, social media, and other accounts? This product takes all those passwords and keeps them safe in a digital vault.

When you want to access your accounts, you just need to remember one master password to unlock the vault. This way, you don't have to worry about forgetting passwords or using weak ones. Plus, we use super-strong locks to protect your information from anyone trying to sneak in.

So, if you want peace of mind knowing your online accounts are safe and secure, our product is the way to go!

Goals

I am on a mission to simplify and secure the way people manage their online accounts and passwords. We want to make it effortless for users to access their accounts while ensuring top-notch security against unauthorized access.

By developing innovative software solutions, we aim to create a seamless experience where users can store their passwords securely in an encrypted format. This way, they only need to remember one master password to unlock access to all their accounts.

The goal is to provide a user-friendly and reliable platform that removes the hassle of remembering multiple passwords and enhances overall digital security. I will strive to empower individuals and businesses with the tools they need to stay safe online without sacrificing convenience.

Why?

Accomplishing our goal of simplifying password management and enhancing security is crucial for several reasons:

1. Security: Weak or reused passwords are a major vulnerability that hackers exploit to gain unauthorized access to accounts. By providing a secure password management solution, this product helps users protect their sensitive information and prevent identity theft and data breaches.

2. Convenience: Remembering multiple complex passwords for various accounts can be challenging and frustrating. My product's solution streamlines this process by allowing users to store and access their passwords easily, reducing the cognitive load and time spent on managing passwords.

3. Productivity: With efficient password management, users can focus on their tasks without interruptions caused by forgotten passwords or account lockouts. This leads to increased productivity and smoother workflow experiences, especially in professional settings.

4. Trust and Reputation: Businesses and organizations that prioritize cybersecurity and offer robust password management solutions build trust with their customers and stakeholders. It demonstrates a commitment to data protection and can enhance their reputation in the market.

5. Compliance: In many industries, compliance with data protection regulations are mandatory. This solution helps businesses adhere to these regulations by implementing strong security measures for handling sensitive information.

Overall, accomplishing this goal is important because it contributes to a safer, more convenient, and productive digital environment for individuals and businesses alike. It addresses critical cybersecurity challenges and promotes responsible data management practices.

Features

1. **Input Management:**
   - Import user email addresses and passwords from a file (password.txt).
   - Validate the format of the input file to ensure it follows the email:password structure.
2. **Encryption:**
   - Utilize a secure hashing algorithm (e.g., SHA-256) to encrypt passwords.
   - Save the encrypted passwords along with email addresses in the format email:hash to a new file (password.enc.txt).
3. **Database Integration:**
   - Integrate with MongoDB using Mongoose to establish a connection with the database.
   - Upload the encrypted email-password pairs to the MongoDB database, ensuring secure storage.
4. **Authentication Functionality:**
   - Develop an authentication function that takes an email address and password as input parameters.
   - Verify if the provided email and password match any entry in the encrypted file (password.enc.txt).
   - Return true if the email-password pair is valid, indicating successful authentication; return false otherwise.
5. **Error Handling:**
   - Implement robust error handling to manage invalid inputs effectively.
   - Return false for scenarios such as missing email or password fields, incorrect email format, or an empty password string.

Project Test Plan

1. Unit Tests

- **Component:** Input Management Module
  - **Test Case 1:** Verify that the input file (password.txt) is read successfully.
  - **Test Case 2:** Validate the format of the input file to ensure it follows the email:password structure.
- **Component:** Encryption Module
  - **Test Case 3:** Test the encryption process using a known input password and verify the output hash.
  - **Test Case 4:** Ensure that the encrypted passwords are saved correctly in the output file (password.enc.txt).
- **Component:** Authentication Module
  - **Test Case 5:** Test the authentication function with valid email and password inputs, expecting a true result.
  - **Test Case 6:** Test the authentication function with invalid email and password inputs, expecting a false result.

2. Integration Tests

- **Component Integration:** Input Management Module and Encryption Module
  - **Test Case 7:** Verify that the input file is processed, and encrypted passwords are saved correctly in the output file.
- **Component Integration:** Encryption Module and Database Integration Module
  - **Test Case 8:** Test the upload of encrypted email-password pairs to MongoDB using Mongoose.
- **Component Integration:** Authentication Module and Database Integration Module
  - **Test Case 9:** Verify that the authentication function correctly retrieves encrypted data from the database and validates user credentials.

3. Regression Tests

- **Scenario:** Update to Encryption Algorithm
  - **Test Case 10:** After updating the encryption algorithm, ensure that existing encrypted passwords are still valid for authentication.
- **Scenario:** Database Connection Changes
  - **Test Case 11:** Simulate changes in database connection settings and verify that the system still uploads and retrieves data correctly.

4. Acceptance Tests

- **User Story:** As a user, I want to securely manage my passwords.
  - **Test Case 12:** User imports passwords from an input file, and the system encrypts and stores them securely.
- **User Story:** As a user, I want to be able to log in using my email and password.
  - **Test Case 13:** User attempts to log in with correct credentials and verifies successful authentication.
  - **Test Case 14:** User attempts to log in with incorrect credentials and verifies failed authentication.

Test Execution Strategy

1. **Unit Tests:** Execute unit tests during development to ensure individual components work correctly.
2. **Integration Tests:** Performs integration tests after development to check the interaction between modules.
3. **Regression Tests:** Conducted after code changes or updates to ensure existing functionality remains intact.
4. **Acceptance Tests:** Performed to validate system behavior against user requirements.