**1. Introduction**

This report presents the plan for my COMP1004 module project, which focuses on creating a single-page application (SPA) aimed at securely managing passwords for different applications. This project is intended to help users store, organize, and access their passwords for various services in a secure and efficient manner.

The report will outline the Software Development Lifecycle (SDLC) and its application to the project. Additionally, I will present the design document, including the project vision, system architecture, and UML diagrams. The document also discusses the challenges faced, the planning process, and how I approached the development of the application.

**2. Project Vision**

**Project Overview:** The PassVault application is designed to help users store and manage their passwords for various online accounts and services in a secure manner. The app will utilize encryption for storing passwords and will provide features like password generation and organization by categories (e.g., social media, email, banking). The goal is to make it easier for users to keep track of their credentials without compromising security.

**Target Users:**

* Individuals who struggle to remember multiple passwords across various online services.
* Users seeking a secure, user-friendly method for managing passwords and other sensitive information.

**Core Features:**

* Secure password storage with encryption.
* Automatic password generation with customizable complexity.
* Categorization of passwords by service type.
* Search functionality to quickly find passwords.
* Option to copy passwords to the clipboard for easy use.
* Password strength analysis and recommendations.

**Feedback:** The project vision will be presented and refined during meetings with module staff.

**3. Software Development Lifecycle (SDLC)**

This section describes the various phases of the SDLC as applied to this project.

**Planning Phase:**

* Research on best practices for password security and existing password managers.
* Task estimation for each sprint, with a rough timeline for feature delivery.

**Requirements Phase:**

* Development of user stories to define the app’s functionality.
* Each feature identified in the user stories is mapped to specific tasks in the sprint plan.

**Design Phase:**

* Initial UI/UX design: A prototype was developed early on to visualize the layout and interaction flow of the web application.
* Architecture design: High-level architecture was designed, including interactions between components such as the front end, back end, and encryption services.

**Development and Implementation Phase:**

* Feature development in multiple sprints, focusing on building and testing core functionalities such as password storage and password generation.

**Testing Phase:**

* Extensive testing of features to ensure robustness, security, and usability.
* User testing to gather feedback on the interface and functionality.

**Deployment Phase:**

* Final deployment of the project, along with security audits and deployment on a secure server.

**Review Phase:**

* Post-deployment review and evaluation of feedback to iterate on future improvements.

**4. Project Plan**

The project is broken down into weekly sprints, where each sprint focuses on delivering specific features or improvements.

|  |  |  |  |
| --- | --- | --- | --- |
| Sprints | Task To Complete | Timeline | Status |
| Sprint 1 | **Research Project Background, Create project vision** | Week 1  01/12/2024-06/12/2024 | Completed |
| Sprint 2 | **Create Login and Register Page (Front and Back End)** | Week 2  06/12/2024-14/12/2024 | Completed |
| Sprint 3 | **Finalize Design Document** | Week 3  07/01/2025-Current | Uncompleted |
| Sprint 4 | **Create Password Management App page** | Week 4  …-… | Uncompleted |
| Sprint 5 | **Implement back End (data Storage)** | …-… | Uncompleted |
| Sprint 6 | **Improve CSS and Design Front-end Refinement (UI/UX)** | …-… | Uncompleted |
| Sprint 7 | **Testing, Debugging, finalize features, based on feedback, final presentation.** | …-… | Uncompleted |

A screenshot of a computer

Description automatically generated

**5. Research and Challenges**

**Potential Issues:**

* **Encryption Implementation:** Ensuring that encryption algorithms are secure and efficient may pose challenges, especially for handling large numbers of passwords.
* **User Security:** Proper implementation of security features like password hashing, two-factor authentication (2FA), and secure communication protocols (HTTPS) will be critical.
* **Cross-browser compatibility:** Ensuring that the app works seamlessly across different browsers and devices.
* **User Interface Design:** Ensuring a clean and user-friendly UI while maintaining a high level of security.

**6. Prototype (UI/UX)**

In the early weeks, I will prototype the look and feel of the application. This will include basic designs for:

* **Homepage:** Display login interface and a list of saved passwords (with masked values) for the user.
* **Password Entry:** Form to securely add new passwords, including password generation and categorization options.
* **Search Interface:** To allow users to find stored passwords quickly.

**Tools Used:** Figma for designing wireframes, Adobe XD for creating interactive prototypes, and HTML/CSS for early web implementation.

**7. Design Document**

**Project Vision:** (Provide the project vision you defined earlier.)

**Background:** (Provide the background and research details related to password management, including security best practices and existing tools.)

**Architecture:** The application will be structured using a Model-View-Controller (MVC) architecture:

* **Model:** Handles all data, including password storage and user data.
* **View:** The front-end part (UI for managing passwords and interacting with the app).
* **Controller:** Handles interaction between the Model and View, such as saving encrypted passwords, retrieving passwords, and managing categories.

**Use Cases:**

* **Use Case 1:** Adding a new password.
  + **Actors:** User
  + **Precondition:** User is logged in.
  + **Post-condition:** Password is securely saved with encryption.
* **Use Case 2:** Searching for a password.
  + **Actors:** User
  + **Precondition:** User is logged in.
  + **Post-condition:** User can view stored passwords with option to copy.

**UML Diagrams:** Provide visual representations of your system’s architecture, including:

* **Use Case Diagram:** Shows user interactions with the system.
* **Class Diagram:** Illustrates the structure of the data model, including password data, categories, and user information.
* **Sequence Diagram:** Shows the interaction between components (e.g., user input triggering password storage, encryption/decryption, etc.).

**8. UML Diagrams**

1. **Use Case Diagram**  
   A diagram of a company

   Description automatically generated
2. **Class Diagram**

A screenshot of a computer

Description automatically generated

**Class Diagram Overview:**

**User** class: Contains user-specific data such as email, username, and password.

**PasswordEntry** class: Contains details about individual passwords, including title, password, and description.

**EncryptionService** class: Handles encrypting and decrypting passwords to ensure sensitive data is secure.

**UserManager** class: Manages user-related operations like registration, authentication, and password entry management.

**PasswordManager** class: Manages password entries, storing and retrieving them for a particular user.

**ValidatorService** class: Ensures the correctness and security of inputs such as usernames, emails, and passwords.

**App** class: Manages the overall application workflow (e.g., logging in a user, rendering pages, etc.).

**Breakdown of Interactions:**

**User → PasswordEntry**:

A **User** can have multiple **PasswordEntry** objects. This relationship is represented as a one-to-many relationship (one user, many password entries).

**UserManager**:

**UserManager** is responsible for handling **User** objects (e.g., creating, authenticating, and logging out users).

It interacts with the **User** class to authenticate and register users.

**PasswordManager**:

**PasswordManager** is responsible for managing the password entries for each user. It stores multiple **PasswordEntry** objects and provides functions like adding, removing, and retrieving entries.

**PasswordManager** interacts with **UserManager** to ensure that password entries are associated with the correct user.

**EncryptionService**:

**EncryptionService** provides security for the password entries by encrypting the password before storing it and decrypting it when retrieving it.

The **PasswordEntry** objects call the **EncryptionService** when a password is added or retrieved, ensuring passwords are never stored in plaintext.

**ValidatorService**:

**ValidatorService** is responsible for validating user input, such as checking the validity of a **username**, **email**, and **password** (e.g., ensuring strong password rules or proper email format).

**UserManager** uses **ValidatorService** to ensure inputs are valid when users register or update their profile.

**App**:

**App** is the central controller of the application and interacts with the **UserManager**, **PasswordManager**, and **EncryptionService** to manage the overall application flow.

The **App** class can log in users, register new users, add new password entries, and render various pages based on user interactions.

1. **Sequence Diagram**  
   A diagram of a software process

   Description automatically generated

**9. Sitemap**

This section includes the general navigation flow for the single-page application:

* **Homepage:** Displays login interface and password list.
* **Password Management:** Allows users to add new passwords, categorize, and manage saved credentials.
* **Settings:** User profile, password policies, and security settings (e.g., enabling 2FA).
* **Help/Documentation:** Provides a user guide and FAQs for secure password management.

**10. Wireframes**

Provide wireframes to show how key pages will appear in your application:

1. **Homepage Wireframe**  
   Includes login form and list of saved passwords.
2. **Password Entry Wireframe**  
   Includes form for adding new passwords and generating secure passwords.
3. **Search Wireframe**  
   Displays a search bar to find stored passwords and manage saved entries.

**11. Issues and Constraints**

Discuss any issues you encountered during development. These could include:

* **Encryption Implementation:** Challenges in ensuring that passwords are securely encrypted and stored.
* **User Interface Security:** Designing the UI while ensuring that sensitive data is never exposed, even temporarily.
* **Third-party Integrations:** Potential issues in integrating any third-party libraries or services for password generation or verification.

**12. GitHub Repository**

Make sure your GitHub repository is regularly updated, and the readme.md file contains the following:

* Project Title
* Project Vision
* Installation Instructions
* Key Features
* Link to the project repository: [GitHub Repository URL]

**13. References**

List any resources you've referred to while working on the project:

* **Books, articles, research papers:**
  + *"The Art of Cryptography,"* by John Doe
  + *"Secure Software Design,"* by Jane Smith
* **Websites, API documentation:**
  + CryptoJS Documentation: https://crypto-js.googlecode.com
  + OWASP Password Storage Cheat Sheet: https://cheatsheetseries.owasp.org

This should now align more closely with your project focus on password management!