Team #4 Inception

GitHub Link: https://github.com/CardosoJavier/CSE_3311_Team_4

Project Concept

API Vault is a cross-platform desktop application designed for both professional developers and all student levels. It aims to speed up the development of small applications and quick prototypes by providing effortless access, retrieval, secure storage, and best practices of API keys. The application uses industry-accepted practices to secure sensitive user data. Additionally, API Vault provides micro-modules and other learning materials to educate users about API usage, the importance of encryption, and study cases implying API security.

Project Vision

API Vault's vision is to eliminate the complexities surrounding API key management offered by cloud services and big sensitive data storage applications. Our vision enables users to focus more on innovation, creativity, and accelerating the development process of their software projects or prototypes. All this while serving as an micro educational platform, imparting crucial knowledge in the domain of secure programming.

Features

- **Secure API key storage**
    - Core feature: Provide secure API storage using industry-accepted data storing including access control, data segregation, and secure configuration. Provide secure encryption using end-to-end encryption. Finally, offering data protection using data minimization and data tokenization.
    - Use demographic: Students and professional developers/engineers.
    - User case: Access, review, store, and retrieve securely API keys stored in API Vault application.
    - Implementation based on risk: Implement it after learning in detail the most used industry-accepted concepts to implement effectively data storing, encryption, and protection.

- **Cloud Storage Service**
    - Core feature: Allow users to access their API keys from different devices connected to the internet.
    - User demographic: Students and professional developers/engineers.
    - Use case: Access, review, store, and retrieve securely API keys stored in API Vault application from different devices or locations.
    - Implementation based on risk: Implement it after member's schedules are defined and identified team members with experience or practice in cloud storage.

- **Local data backup**
    - Core feature: Give to the user the option to create scheduled or manual local backups in the user machine.
    - User demographic: Students and professional developers/engineers.
    - Use case: Create emergency backups in user machines in case of no internet connectivity.
    - Implementation based on risk: Implement it after learning in detail the most used industry-accepted concepts to implement effectively data storing, encryption, and protection.

- **API key best practices build-in tools**
    - Core feature: Tools to help the user to keep good practices with their API keys such as key rotation, terminal environment variable loading, and among others.
    - User demographic: Students and professional developers/engineers.
    - Use case: Remind developers to change API key every few months to annually depending on users needs.
    - Implementation based on risk: Implement it after identifying team member with more expertise in both technologies used in project (C# and .Net MAUI) and other members catch up with the learning material about project stack.

- **API key grouping:**
    - Core feature: Allow users to create API groups for each project using multiple APIs. Enabling clean and efficient project organization.
    - User demographic: Students and professional developers/engineers
    - Use case: Group multiple API keys used in a single project into a single group to maximize project organization, API keys rotation, and key retrieval.
    - Implementation based on risk: Implement it after identifying team member with more expertise in both technologies used in project (C# and .Net MAUI) and other members catch up with the learning material about project stack.

- **Two factor authentication (2FA)**
    - Core feature: Give users another layer of security by enabling 2FA using their mobile devices.
    - User demographic: Students and professional developers/engineers.
    - Use case: Avoid undesired access to API keys from other team members, family members or strangers that have access to the developer machine.
    - Implementation based on risk: Implement it after identifying team member with more expertise in both technologies used in project (C# and .Net MAUI) and other members catch up with the learning material about project stack.

- **Cross-platform**
    - Core feature: Desktop application will be available for both Windows and MacOS operating systems.
    - User demographic: Students and professional developers/engineers.
    - Use case: Allow teams or developers with different/multiple OS machines to access their store APIs.
    - Implementation based on risk: Implement it after identifying team member with more expertise in both technologies used in project (C# and .Net MAUI) and other members catch up with the learning material about project stack. Additionally, learning about API or third-party technology such as Twilio.

- **Educational micro-modules**
    - Core feature: Visual material designed for quick learning and reinforcement of APIs and secure programming.
    - User demographic: Students learning SWE or programming.
    - Use case: Help beginning developer or student to learn about fundamentals of secure programming and encryption.
    - Implementation based on risk: Implement it after identifying team members with more expertise in encryption, APIs, and secure programming. Or, after all team members understand the basics concepts of the module's topics. Then, identifying members with free time in their schedules to record a short 5-10 minutes presentation about the module's topics.

- **Educational quizzes:**
    - Core feature: Test users' understanding of micro-modules with small quizzes. The users can take each quiz an infinite number of times.
    - User demographic: Students learning SWE or programming.
    - Use case: Students testing their understanding from micro-modules and figuring out if they need more information about the topic.
    - Implementation based on risk: Implement it after identifying team members with more expertise in encryption, APIs, and secure programming. Or, after all team members understand the basics concepts of the module's topics. Then, identifying members with free time in their schedules to create a short quiz (5-10 questions) about concepts covered in micro-modules.

Customers and Users

- **All levels students:** Students at any educational level who have an interest in developing small personal software projects or prototypes. These students possess a foundational

understanding of software engineering and programming concepts, enabling them to engage with the application to manage API keys effectively without the need for comprehensive cloud services or complex API key management systems. The primary demographic within this group consists of high school and college students who seek an intuitive and accessible tool to support their academic and personal projects. This user base is characterized by their educational or free time pursuits in technology-related fields and their need for a solution that aligns with their learning objectives and project requirements. To ensure the application meets their specific needs, a good communication channel and professional relationships with the representative students from this group is essential for quality feedback. For example, engaging with students through academic institutions, technology clubs, and online forums dedicated to programming and software development can provide valuable insights into their experiences and preferences.

- **Professional/experienced developers:** Software engineers and experienced developers who require a tool for rapidly prototyping software projects. These professionals are distinguished by their extensive knowledge and practical experience in the field, often working within the tech industry or on independent software development projects. Unlike the student group, these users bring a depth of technical expertise and a clear understanding of their needs in a prototyping tool, particularly valuing efficiency, reliability, and the capacity to manage API keys without integrating complex cloud services or advanced management systems initially. The application is positioned as an ideal resource for these users, offering the necessary functionalities to prototype projects efficiently before transitioning to more sophisticated cloud-based solutions for managing project secrets, including API keys. Establishing a direct channel for frequent feedback and collaboration with experienced developers is crucial. This can be achieved through professional networks, software development communities, tech meetups, and industry conferences. Their input is crucial to ensure API Vault serves as a powerful tool in any professional developer's toolkit, facilitating the initial stages of project development

Competitors

Our team had identified four applications and services with similar functionality regarding API key management. However, we provide strategic differences explained below:

- **Cloud Services (AWS, GCS, Azure)**: API key management services from big cloud services such as AWS secrets manager, GCS API manager, and Microsoft Azure secrets manager. They provide comprehensive and robust tools to manage all project sensitive data, including API keys. Their API managers are designed to handle all common uses and enforce good practices involving APIs. These services also include other tools to

manage most aspects of the software project in general. Also, they have a wide community and extensive documentation to set up their services in any project.

    - API Vault differentiator: Big cloud services are complex to set up, specially for beginning or inexperienced developers such as high school and college students. Even some experienced developers find the integration of these services in their projects frustrating. On the other hand, API Vault aims to serve as a small projects or quick prototypes tool to easily set up, retrieve, store, and provide other easy-to-use services regarding API keys. On the other hand, cloud services only offer limited time or resources for non-paying accounts, making it inaccessible for a large percentage of small teams, students, and beginning developers.

- **1Password:** Cross-platform password manager to create, store, and autofill a wide array of passwords in different platforms and services. They provide several tools for different users: Personal use, business, enterprise, and developers. As the last user suggested, they have tools to help developers handle application secrets including API keys. API tools include CLI to load environment variables and easy API key actions such as access, retrieval, storage, and among others.
    - API Vault differentiator: While 1Password also provides API key management services, they do not offer free-tier plans. Making it inaccessible to a large percentage of students aiming to learn software engineering and develop personal or small projects. Also, they offer courses focused on their product usage and general password managers. On the other hand, API Vault is aimed to help students to work on their projects. Additionally, our project offers micro-modules to learn about APIs, encryption, and secure programming.

- **LastPass**: Cross-platform application primarily focused on storing passwords and personal information securely. They provide high security by holding global security certifications, protecting against dark web by implementing dark web monitoring technologies, and full endpoint encryption.
    - API Vault differentiator: LastPass mainly focuses on daily use passwords management such as social media, email, streaming services, and among others passwords used by the general public. While it can store API keys, it is not optimized for developers, slowing down the developing process. Also, it does not provide any learning material design for secure programming concepts or any cybersecurity concept in general focused for developers.

- **LashLane**: Cross-platform application focused on securely storing and managing passwords, personal information, secure notes, and payment information for individuals and businesses. They also implement cloud-based data, enabling users to access their

passwords from different devices. Also, they support unlimited password sharing, one-click passwords autofill, password generator, and dark web monitoring.

- API Vault differentiator: Despite that LashLine services can be used to store API keys in the cloud, they do not have common developer tools to make an frictionless retrieval, store, and organization of API keys. This is exactly the opposite to the main goal of Vault API, which is to speed up the development of quick prototypes and small projects. Additionally, they do not offer any course or modules educating the user about the importance of securing passwords.

Risks

Listed below are the three main risk identified by our team that may cause an important delay or challenge in the completion of the project:

Insufficient Security and Encryption Practices

- Probability: High
- Impact: High
- Description: Given the varied levels of knowledge and experience within the team, particularly in areas critical to the application's core functionality such as security and encryption, there's a significant risk that the application may not implement best practices correctly or in time. This could lead to vulnerabilities where API keys data may be exposed to unauthorized access or breaches.
- Mitigation Plan: Engage with a mentor or advisor who has expertise in cybersecurity (CSEC officers and CSE 3311 professor) to review the design and implementation of security features. Additionally, use online resources to upskill team members in data security subjects.

Technical Challenges with C# and .NET MAUI

- Probability: High
- Impact: Medium
- Description: Some team members are not familiar with C# and .NET MAUI. So, the team might face technical challenges that could delay the project. These challenges could affect the application's cross-platform functionality, performance, and user experience.
- Mitigation Plan: Prioritize learning and development in .NET MAUI for all team members, utilizing available online tutorials, courses, and forums. Schedule regular knowledge-sharing sessions where team members can discuss challenges and solutions. Consider reaching out to the .NET MAUI community for advice and best practices. Establish a relationship with a technical mentor proficient in .NET MAUI who can provide guidance and support.

Limited Availability of Team Members Due to Scheduling Conflicts

- Probability: High
- Impact: Medium
- Description: Given that our team consists of college students, there's a significant risk associated with team members' availability due to other academic courses and job related time conflicts. Some team members have a totally different schedule compared to the average college student. These challenges may cause poor communication, project delays, and bad team organization, affecting the overall project in all aspects.
- Mitigation Plan: Gather detailed information about each team member's schedule, including classes, exams, and other commitments and use this information to create a comprehensive team calendar. Similarly, assign tasks considering team members' available time and strengths.

API Vault Prototype