CloudHaven

Grant Matejka, Justin Evans, Sarah Samora, Evan Rozhon, Kaitlin Clever

Biggest Pig

Jira

Repository

Created On: November 10, 2020

Last Updated: December 2, 2020

I. Introduction

Section A: Overview, Problem Description and Summary

Overview

Cloudhaven aims to provide a level playing field that standardizes application development by providing rapid UI development and an ability to integrate easily with other apps. Cloudhaven also aims to give users more control over their privacy and personal data.

Problem Description

Cloudhaven aims to solve the problem of differentiating UI and lack of privacy, which affects users and companies at all levels. The time it takes users to adjust and learn the UI of different applications is time lost to the individual and the company.

Summary

Cloudhaven is a "UI as a service" technology that aims to create a seamless user experience across multiple application scenarios.

Section B: Glossary

UI as a Service - Rather than from scratch UI's the front end is provided, application neutral, to middleware that can adapt to any use case. This makes it easy for users to adapt the ui to many of their needs using pre built components that can be accessed through a plug-in or api call.

Tokenization* - Takes meaningful data and encrypts it in such a way that no data will be breached if the 'token' is compromised. These tokens then serve as a reference to the original data but contain no traces of the original data themselves.

Single Sign On -** Gives users the opportunity to log into many independent services or applications through a single id and password. This means that a user only needs to know of a single accounts login information to access many applications.

Offloaded Development - Takes pressure off of other developers/implementers by providing resources through a separate source/team/service

Vendor-Neutral Solution - Ensures broad compatibility and support by being non proprietary and providing an open solution that can be adapted/utilized by a variety of users

Data Regulatory Compliance - With new laws being put in place compliance is of utmost importance. Sensitive and identifying data has the highest levels of protection. Posting this guide

as such an important topic:
https://cloud.netapp.com/blog/data-compliance-regulations-hipaa-gdpr-and-pci-dss

Notes:

**CloudHaven is a Single Login product rather than Single Sign On
*Tokenization is not necessary for all data, rather for personally identifiable data or upon request

Section C: Context

The problem was brought up to the team by Richard Vann. This problem of differentiating UI and lack of privacy affects users and companies at all levels. The time it takes users to adjust and learn different applications and their UI is time lost to the individual and the company. Also in this day in age many apps have issues with user privacy and agreements that are not transparent. With digital dignity becoming a topic of concern more and more people are becoming privacy conscious. While many apps have tried to tackle one of these issues no one app has tried to tackle it as a whole. Google plugins have tried to make UI more manageable and customizable while apps like lastpass and brave have tried to obfuscate personalized data they have done so more or less only in their own sphere.

Section D: Product Requirements and Technical Requirements

Product Requirements in the Form of User Stories

As an end user, I want to be able to log into Cloudhaven.

As an end user, I want to be able to access many of my important portals and information in one seamlessly integrated space.

As a portal provider, I want to be able to securely and consistently give my users the information they need.

As a portal provider, I want to be able to safely and consistently interface with the end user.

As a vendor, I want to serve my be able to easily implement the Cloudhaven API.

As a an Cloudhaven administrator for my company, I want to be able to control vendor permissions granully.

As an end user, I want to be seamlessly message relevant people in the context of the applications I am viewing.

Technical requirements

The system shall be compatible with the latest versions of Chrome, Safari, and Firefox.
The system shall be hosted on the cloud without needing installation on customer locally owned servers.

Section E: Out of Scope

Product and technical requirements that will be disregarded:

Native desktop application.
Supporting applications from a variety of vendors with different UI needs.

Section F: Future Goals

Product and technical requirements slated for a future time.

Tokenization of user Data

Multi factor authentication

Single UI for multiple applications

Allow for a customizable theme.

Mobile App as a stretch goal.

Section G: Assumptions

We are assuming that customer/implementer/user has internet access as well as a modern web browser. Continuing in these assumptions, we are assuming that the implementing party has general API experience, can understand basic documentation and understands their party's data needs. With these assumptions accounted for, CloudHaven will be able to be incorporated into and functional under any intended use case.

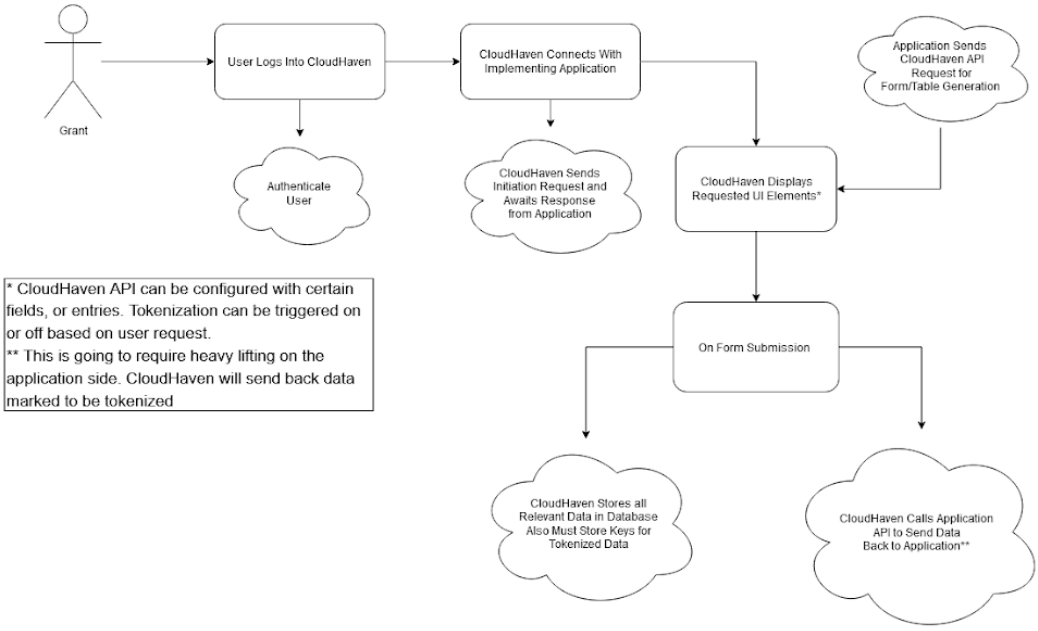
II. Solutions

Section A: Current or Existing Solution / Design

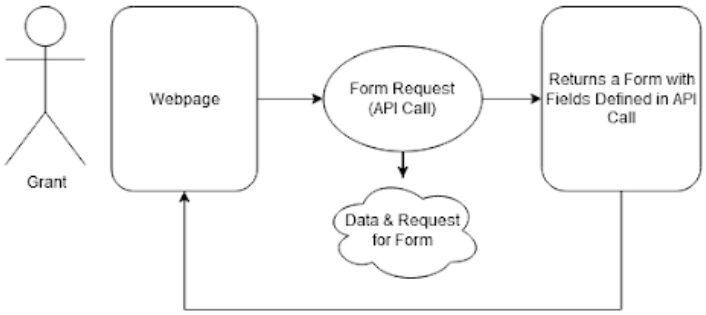
Current solution is designed to take in user information as well as the application's forms that they are planning to use and convert them to the CloudHaven UI design. It is still in the initial development phase and is evolving every week. In its current iteration it is still buggy and not ready for use but the basic groundwork has been put in place. The pros of this deployment are that it is easily customizable and features can be changed and direction pivoted without too much struggle. The cons are that it does not fill the role intended by the app yet.

Section B:Proposed Design

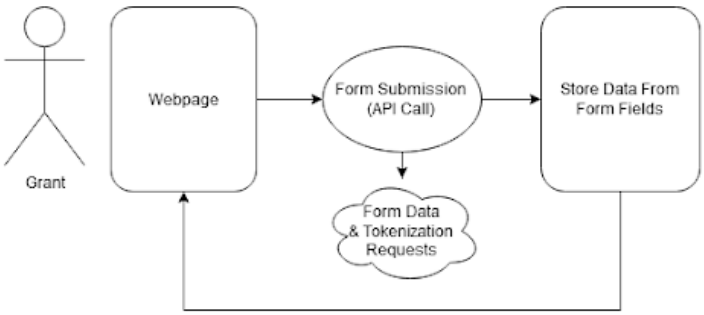
General User Interaction With Cloudhaven



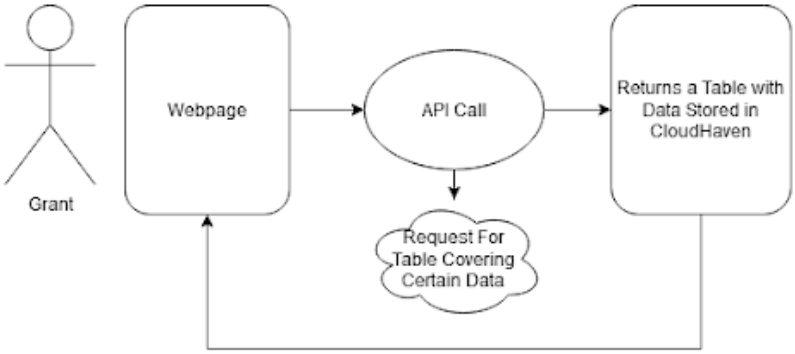
User Requests a Form



User Submits Form Data



User Requests a Table



API Endpoints

- Generate Form With Provided Fields
    - Each Specified Field has Option to be Tokenized
- Submit Data from Completed Form
    - Each Specified Field has Option to be Tokenized
- Generate Table with Requested Data as Columns

External components that the solution will interact with and that it will alter
Dependencies of the current solution
(if applicable)
Pros and cons of the proposed solution
Structure and behavior diagrams to represent the design.

Data Model

The data for Cloudhaven is being stored in a PostgreSQL database, which is a relational database. As of now, the schema only consists of the user table, which consists of columns for the user id, user email, and the hashed password.

There are plans to grow the database to include features like vendor permissions, client permissions, and messaging, but at this stage development is centered on achieving a working prototype for the front-end before moving on to the database. As the front-end becomes more robust, the database will grow with it to allow for more features and data.

Data Validation methods

Currently, all the data on the front-end is validated cursorily by the client side application code and then paired with an authentication token before it is passed into the database. All data sent from the client side is also sent with prepared statements to prevent SQL injections.

Business Logic

The Cloudhaven will consist of a specified format the vendor applications will have to implement to be displayed on Cloudhaven. The basic format of the request will include authorization (for the verification of access), target (the user who asked for the components to be displayed), and a list of components to display.

```
{
  "authorization": "6fa691d2c07bea6ae78b7022c80e9d20",
  "components": [
    {
      "columns": [
        {
          "children": [
            {
              "component": "textFormField",
              "label": "Full Name",
              "required": "true",
              "validation": "[a-fA-F0-9]"
            },
            {
              "component": "textFormField",
              "label": "Address",
              "required": "true",
              "validation": "[a-fA-F0-9]"
            }
          ]
        },
        {
          "children": [
            {
              "component": "textFormField",
              "label": "Phone Number",
              "required": "true",
              "validation": "[a-fA-F0-9]"
            },
            {
              "action": "Submit",
              "component": "button",
              "target": "78b83774d3a907fbea42783d58a95204"
            }
          ]
        }
      ],
      "component": "form",
      "formID": "78b83774d3a907fbea42783d58a95204"
    }
  ],
  "target": "42aefbae01d2dfd981f7da7d823d689e"
}
```


The above picture is a sample api request from the vendor to Cloudhaven that is requesting to display a form with two columns and in those two columns, three text entry fields and a button.

Failure states include incomplete or unauthorized API requests. Clear and readable error messages and strict adherence to error code standards will be practiced to make error states as

easy as possible to resolve.

Limitations are largely based on the implementation of the API on the incorporating party's end and Cloudhaven's implemented components. This means that CloudHaven will have to have some API endpoint to send necessary data 'back' to the incorporating party. Similarly, limitations arise when factoring in that interested users will have to learn and incorporate the CloudHaven API as well as develop their own.

Presentation Layer

CloudHaven is an UI as a Service product and thus finds the general UI/UX to be of great importance. We will be utilizing the material-ui CSS framework in our react based web app. Alongside this framework we will be utilizing general Material design ideologies throughout CloudHaven.

Here is a link to our prototype.

Web concerns include the incorporation and distribution of responsive and dynamically driven ui elements that can be served to our customers, through our API, and be aesthetically pleasing in modern browsers and different screen sizes.

At this time CloudHaven will not be targeting mobile UI.

Error handling

CloudHaven will scale accordingly due to the API nature of the app and additional server based resources can be allocated upon adoption. Limitation includes difficulty in load bearing api requests, but as stated above can be resolved with flexible hosting plans that can allocate more resources as needed.

For our web application, we should never show users a full-on error page, we should land on an error page we made to redirect them to a safe page.

Future requirements can be implemented with full backwards compatibility of past provided features.

Section C: Test Plan

Tests will be implemented throughout development of the project. Due to the nature of the application, CloudHaven will stress the importance of integration, end-to-end and functional tests over unit tests. Unit tests will be implemented as cases are feasible. When data flow and tokenization is implemented we will conduct security stressing testing and penetration testing. Tests will automatically be run with all pushes to main and builds will not be deployed if the builds do not pass the tests.

Section D: Alternate Solutions and Designs

Modern web browsers and plugins provide a lot of collaborative tools already and password managers provide easy access to logging into multiple services. Companies like Google and Facebook also provide single sign on services and many more companies provide their own solution to collaboration difficulties.

These solutions have many benefits and have large user bases that are comfortable and familiar with each application. These products are also very fast, flexible, expandable and easy to use.

Our product is a better alternative to these applications and services as CloudHaven will capture all data and collaboration needs in one place with one signon. All data can be captured, displayed and protected in a single web application, CouldHaven. As well as data needs, future goals of CloudHaven include bringing together collaborative tools to make a central location for a team's communication. CloudHaven is also highly customizable with an easy to use/implement API. The centrality and flexibility of CloudHaven is its greatest advantage over all current alternatives.

III. Further Considerations

Section A: Security considerations

Potential threats to Cloudhaven are that handling user data while avoiding malicious input in the form of SQL injections. Malicious or otherwise compromised applications are a concern. Successfully securing the consumer data is also a concern, because Cloudhaven will be storing a wealth of consumer data.

With regard to SQL injections on the consumer side, Cloudhaven will be using prepared statements. Integrated applications will be put through a rigorous, continuous screening process. The database will secured by a length, unique password combined with another method of authentication (i.e. Yubikeys, Duo, or Google Authenticator).

Since applications will be put through a screening process, there will be a slight, but necessary, barrier to integrating an application into Cloudhaven. There are no concerns with the rest of the product with regards to using prepared statements and multi-factor authentication.

Section B: Privacy considerations

Cloudhaven does not currently follow local laws and legal policies on data privacy, but it is essential that CloudHaven follows the strictest legal policies on data privacy. Consumer privacy will be baked into Cloudhaven from the very beginning to ensure that privacy and following privacy laws are fundamental tenets of Cloudhaven.

Cloudhaven will give users total control over their privacy through tokenization and encryption of all protected user data. Users will have the final say over whether or not an application has access to their data, and they will be able to control that granularly through tokenization. Key pieces of user information will be distributed to the applications as a tokenized version that users will be able to approve the de-tokenization of in each individual case.

Section C: Risks

Cloudhaven risks that not enough companies will integrate themselves with Cloudhaven's API, either because of cost or desire to be different. There is a risk that making concessions to companies in terms of API design or "look" of the UI Cloudhaven generates for them cannot be walked back, because it is hard to take features away from companies and/or users, especially if they have come to rely on those features. However, it is necessary to take the risk of implementing Cloudhaven as originally intended, because of the enormous potential of the seamless UI platform Cloudhaven could be.

IV. Deliberation

Section A: Discussion

At this time, there are no elements of the solution that members of the team do not agree on. We are all happy with the direction that our project is going in.

Section B: Open Questions

Exact structure of the API requests/responses have not been decided upon at this point. In the same way, database structure has not been completely decided upon yet. Concrete decisions will be decided when implementation and utilization of said API begins.

V. References and Acknowledgements

Special acknowledgements to Richard Vann in his pitch, elaboration and consistent support of this product.

We also acknowledge Dr. Bruno Da Silva in his continued support and guidance in the development of this product and document.

Data Compliance Reference:
https://cloud.netapp.com/blog/data-compliance-regulations-hipaa-gdpr-and-pci-dss

Vann's CloudHaven Website: http://cloudhaven.net/