Password Management System: PassSafe

A Project Report

Submitted By

22BCE2819 Mohammed Omar Shaikh

22BCE3741 Ramkumar Arcot Dharmalingam

of

Bachelor of Technology

in

Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vellore Institute of Technology

Vellore

April – 2025

Table of Contents

Abstract

The Password Management System is a secure and user-friendly application designed to help users efficiently store, manage, and retrieve their passwords. The system enables users to add, edit, and save passwords for various websites while ensuring robust validation and real-time feedback via toast notifications. A search functionality allows users to quickly filter stored credentials by site name or username, enhancing accessibility.

To improve security and usability, the system includes a password visibility toggle, enabling users to view or hide passwords with a simple click. Additionally, users can add custom fields (e.g., email) and mark them as sensitive, ensuring flexibility in data storage. A password strength meter provides real-time feedback on the security of newly entered passwords, encouraging stronger credentials.

Passwords are securely fetched and stored on a backend server, utilizing Auth0 authentication for secure access. The system supports POST and PUT API requests to create and update entries, preventing duplicate passwords with appropriate warnings. To enhance user experience, it includes loading animations and feedback mechanisms during API interactions. Once retrieved, stored passwords are displayed in a structured table where users can view, update, or delete entries as needed.

By integrating strong security measures with an intuitive interface, the Password Management System simplifies password management while ensuring data safety and ease of use.

Objective

The objective of this project is to develop a secure and user-friendly Password Management System that simplifies storing and managing credentials while ensuring robust security.

Key Goals:

- Centralized Credential Management: Provide a secure repository for storing and organizing passwords and associated information.
- Enhanced User Experience: Implement an intuitive interface with search, password visibility toggle, and real-time feedback.
- Improved Security: Integrate a password strength meter to encourage stronger passwords.
- Customization Flexibility: Allow users to add, modify, and remove custom fields as needed.
- Secure Data Handling: Use Auth0 authentication and backend validation to prevent duplicate entries and ensure safe storage.
- Optimized API Interactions: Ensure smooth operations with real-time feedback, loading animations, and responsive API calls.

Specification

1. Functional Requirements

a. Credential Management:

i. Add Passwords:

- Users can add new password entries with fields including site name, username, password, and additional custom fields (e.g., email).
- The form will validate inputs in real-time, ensuring that mandatory fields are completed and that password requirements (if any) are met.

ii. Edit and Update Passwords:

- Users can edit existing password entries.
- The system supports both updating a record (via PUT requests) and warning the user if the password already exists.

iii. Delete Passwords:

- Users can remove password entries from the system with a confirmation prompt to avoid accidental deletions.

b. Search Functionality:

- A dynamic search bar allows filtering of stored credentials by keywords found in the site name or username fields.
- Instant search feedback provides real-time filtering as the user types.

c. Password Visibility Toggle:

- An eye icon enables users to toggle the visibility of the password field, switching between obscured (masked) and plain text views.

d. Custom Field Management:

- Users can add additional fields (such as email) to each password entry.
- Custom fields can be marked as sensitive, altering how data is displayed or handled.
- The system allows removal of these additional fields if needed.

e. Password Strength Meter:

- While the user types a password, a strength meter provides visual feedback (e.g., weak, moderate, strong) based on criteria such as length, character diversity, and complexity.

f. Data Fetching and Storage:

i. Backend Integration:

- Password entries are fetched from and stored on a backend server.
- Secure data handling is ensured through the use of Auth0 for authentication.

ii. API Operations:

- POST Requests: For creating new password entries.
- PUT Requests: For updating existing password entries.

iii. Duplicate Prevention:

- The system checks for duplicate entries and alerts users with appropriate warnings before saving.

g. Feedback and Loading Indicators:

i. Loading Animations:

- Displayed during API calls (fetching, saving, updating, or deleting) to inform the user of ongoing processes.

ii. Toast Notifications:

- Provide immediate feedback for successful operations, errors, and validation messages.

h. Display of Stored Passwords:

- Password entries are presented in a table format, displaying key details (e.g., site name, username) while allowing quick actions (edit, delete).
- Sensitive information remains protected until the user explicitly toggles its visibility.

2. Non-functional Requirements

a. Security:

i. Authentication:

- Integration with Auth0 ensures secure user authentication.

ii. Data Encryption:

- All sensitive data, including passwords and additional custom fields, is transmitted and stored securely.

iii. Access Control:

- Only authenticated users can access and manipulate stored credentials.

b. Performance and Scalability:
    i. Responsive UI:
        - The application provides immediate feedback and maintains performance even with a large number of entries.
    ii. Efficient Data Handling:
        - Optimized API calls and data caching where applicable to minimize load times and server strain.

c. Usability:
    i. Intuitive Design:
        - User interface follows modern design principles, ensuring ease of navigation and clear visibility of functions.
    ii. Accessibility:
        - The application adheres to accessibility standards, making it usable for a diverse range of users.
    iii. Responsive Design:
        - Adaptable UI for various screen sizes, ensuring usability across desktops, tablets, and mobile devices.

d. Reliability:
    i. Error Handling:
        - Robust error handling mechanisms and clear user notifications in cases of API failures, duplicate entries, or network issues.
    ii. Data Integrity:
        - Consistent state management between the client-side application and backend storage to avoid data loss or corruption.

e. Maintainability:
    i. Modular Code Structure:
        - The application is built with a modular architecture to ease future enhancements and maintenance.
    ii. Documentation:
        - Detailed documentation for both the API and the frontend functionalities ensures maintainability and ease of onboarding new developers.

3. User Interface & User Experience (UI/UX) Specifications

a. Layout and Navigation:

i. Main Dashboard:

- Displays the list of saved password entries in a table format with clear options for search, add, edit, and delete.

ii. Form Design:

- A clean, user-friendly form for entering and updating credentials, including inline validations and contextual help.

iii. Visual Cues:

- Icons (such as the eye icon for password visibility) and color-coded indicators for password strength provide intuitive visual cues.

b. Feedback Mechanisms:

i. Real-time Validation:

- Immediate inline feedback and toast notifications for successful operations or errors.

ii. Animations:

- Smooth loading animations during API calls to enhance user perception of speed and responsiveness.

4. Data Management & Security Specifications

a. Data Storage:

- Password entries and related data are stored in a secure backend database.
- Sensitive data is encrypted both in transit (via HTTPS) and at rest.

b. Authentication & Authorization:

- Use of Auth0 for managing secure login and access control.
- Role-based permissions to ensure only authorized users can perform specific actions (e.g., editing or deleting entries).

c. API Specifications:

- Endpoints:
  - Create Entry (POST): Accepts new password entries and performs duplicate checks.
  - Update Entry (PUT): Allows modifications to existing entries, ensuring that changes are saved securely.

- Fetch Entries (GET): Retrieves password entries from the backend, ensuring data is loaded efficiently.
- Delete Entry (DELETE): Removes a password entry from the database after user confirmation.

- Response Handling:
  - Each API call returns appropriate status codes and messages, with error handling for unexpected scenarios.

5. Testing and Quality Assurance

a. Unit Testing:
   - Each component (form validations, API calls, UI elements) will be tested individually.

b. Integration Testing:
   - End-to-end tests to ensure seamless integration between the frontend and backend.

c. Security Testing:
   - Regular security audits and penetration testing to verify the robustness of the authentication and encryption mechanisms.

d. Usability Testing:
   - User testing sessions to gather feedback on the interface and ensure intuitive navigation and interaction.

6. Deployment and Maintenance

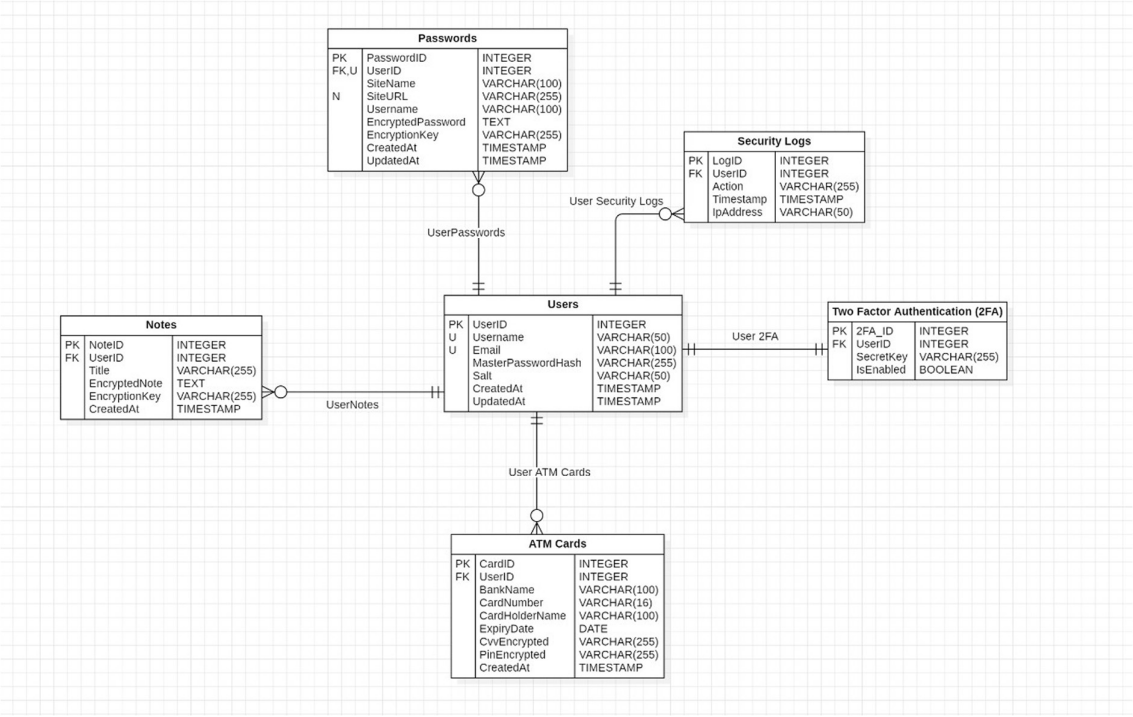
a. Deployment Strategy:
   - Use of continuous integration and deployment pipelines for seamless updates and feature rollouts.

b. Monitoring:
   - Integrated logging and monitoring to track system performance and quickly identify any issues.

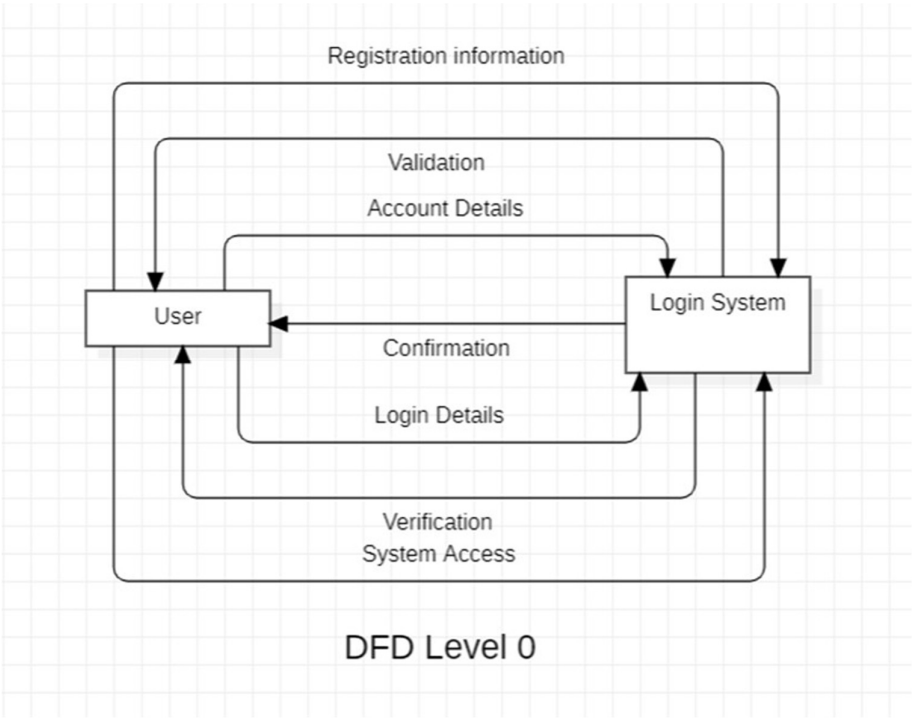
c. Maintenance:
   - Regular updates to address security vulnerabilities and performance enhancements.

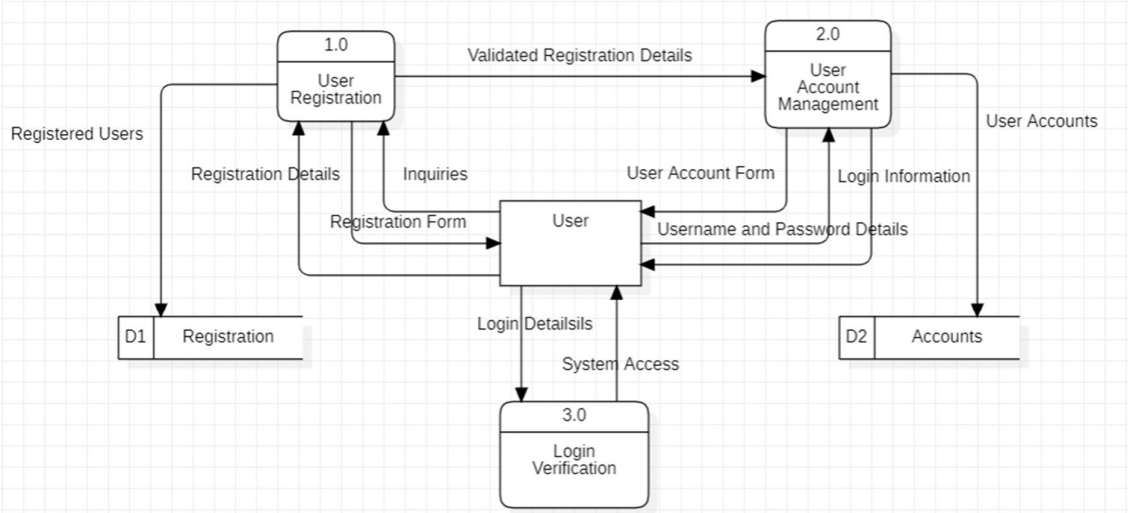
Diagrams

Entity Relationship Diagram

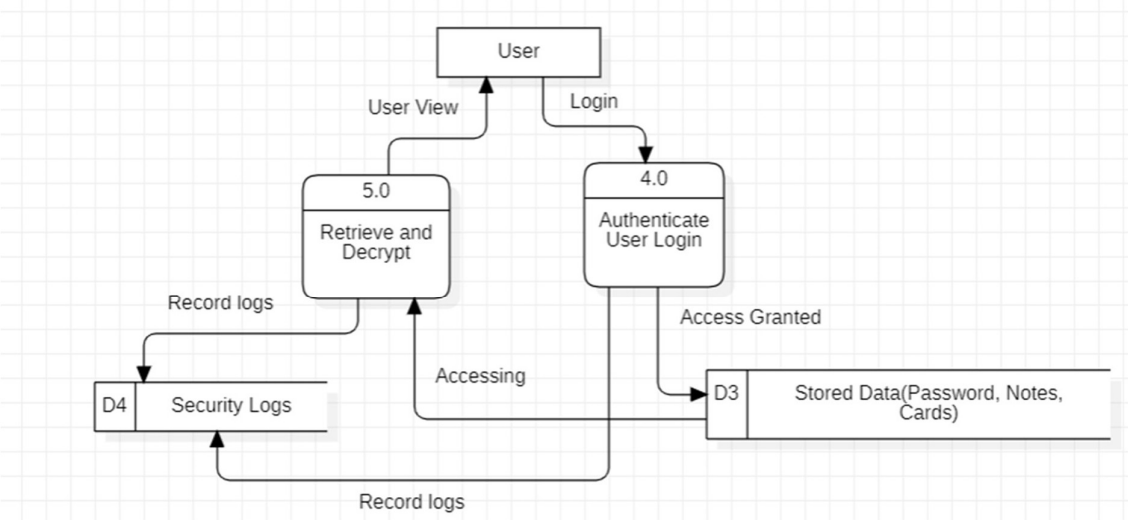


Data Flow Diagram



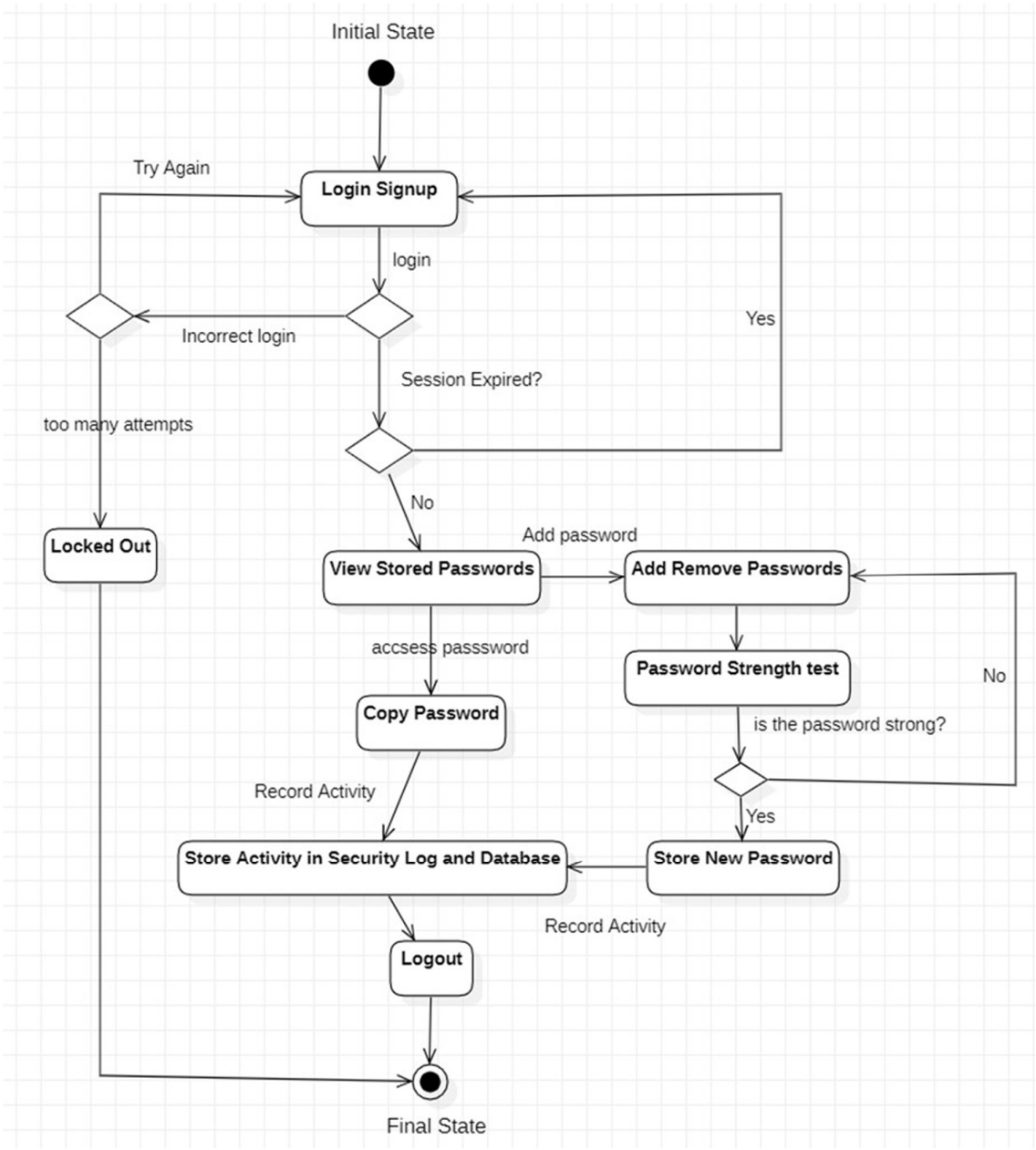


DFD Level 1

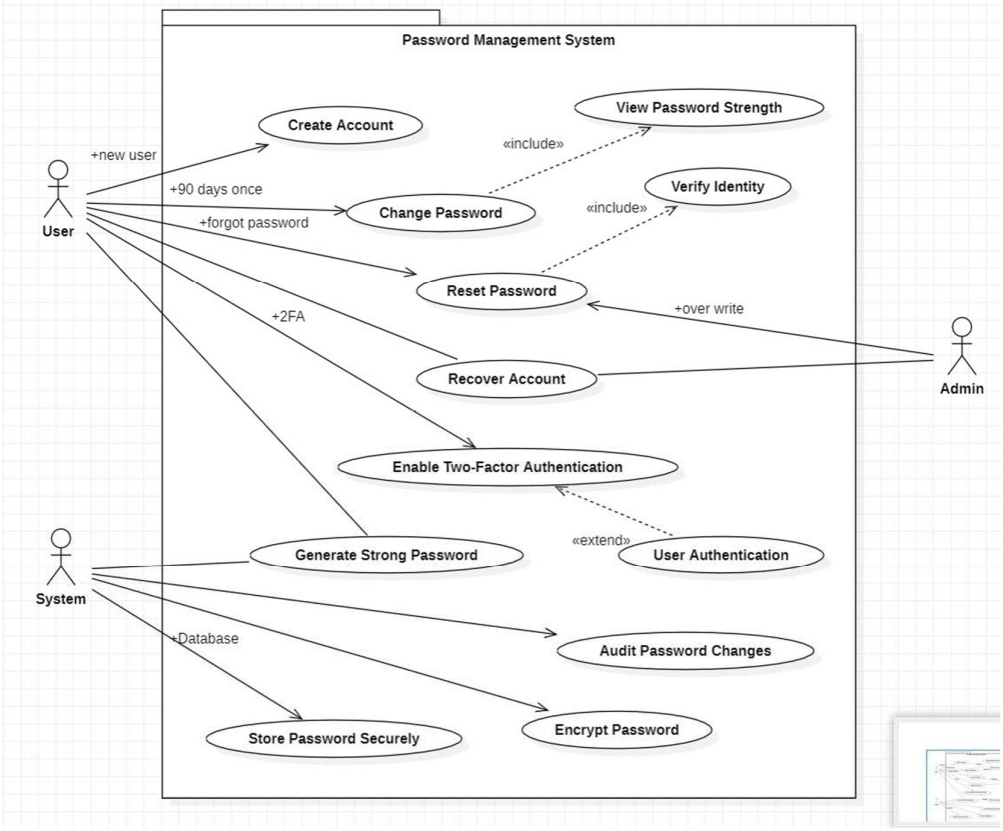


DFD Level 2

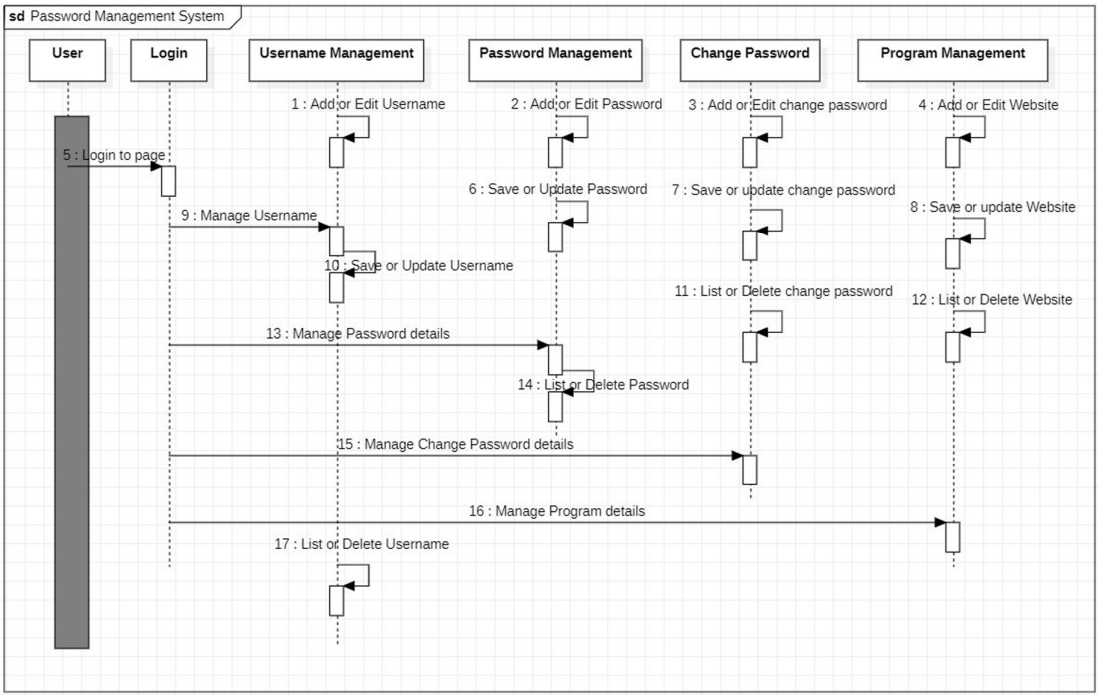
State Chart Diagram



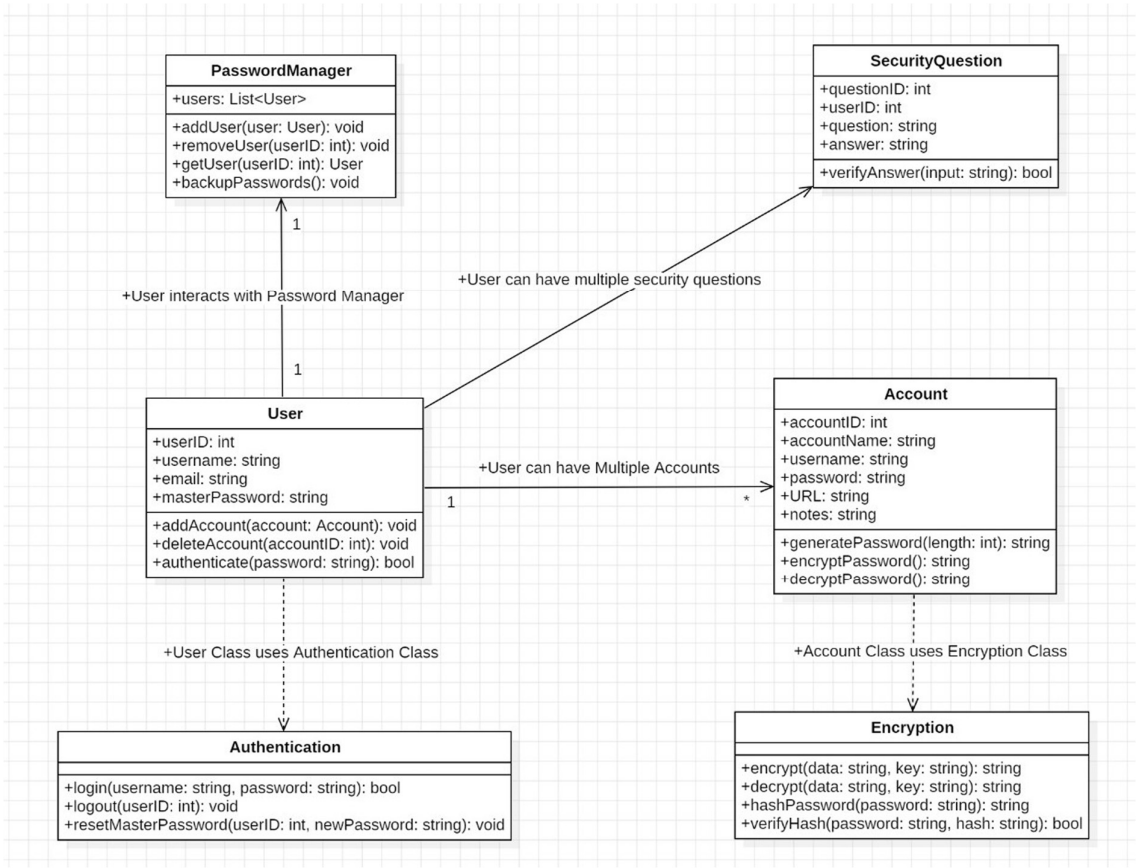
Use Case Diagram



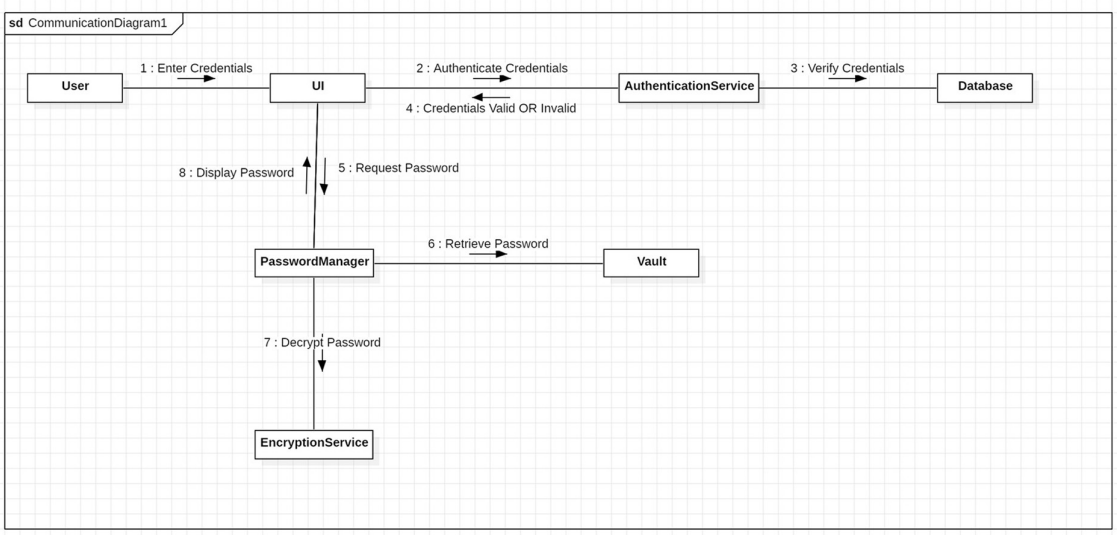
Sequence Diagram:



Class Diagram



Communication Diagram



Module Description

1. User Interface (UI) Module

Responsibilities:

- Provide an intuitive and responsive layout for all user interactions.
- Render key pages, including the dashboard, password entry form, and settings.
- Ensure consistency in design through reusable components like input fields, buttons, and icons.

Key Features:

a. Dashboard View:
   - Displays a table of saved passwords with options for editing or deletion.
   - Incorporates a dynamic search bar to filter credentials by site or username.

b. Password Entry Form:
   - Supports both adding new entries and editing existing ones.
   - Includes fields for site name, username, password, and custom fields (e.g., email).
   - Implements inline validation and a password strength meter.

c. Interactive Elements:
   - Toggle icons (e.g., an eye icon for showing/hiding passwords).
   - Toast notifications for immediate user feedback.

d. Responsive Design:
   - Adjusts layouts for desktops, tablets, and mobile devices.

2. Authentication & Authorization Module

Responsibilities:

- Securely manage user login and session handling.
- Ensure that only authenticated users can access and modify data.

Key Features:

a. Auth0 Integration:
   - Facilitates secure login and token management.
   - Provides role-based access control for different user privileges.

b. Session Management:

- Maintains secure user sessions and handles logout functionality.
- Protects sensitive routes and API endpoints from unauthorized access.

3. Credential Management Module

Responsibilities:

- Handle the creation, retrieval, updating, and deletion (CRUD) of password entries.
- Manage custom fields and ensure data consistency.

Key Features:

a. Add & Edit Passwords:
    - Processes form submissions with real-time validation.
    - Supports POST for new entries and PUT for updating existing ones.
    - Checks for duplicate entries and alerts the user when necessary.
b. Delete Passwords:
    - Implements safe deletion with confirmation prompts to prevent accidental loss of data.
c. Custom Field Handling:
    - Enables users to add or remove custom fields (e.g., email, notes) for each password entry.
    - Allows marking fields as sensitive to adjust their display or encryption settings.

4. API Integration Module

Responsibilities:

- Serve as the intermediary between the frontend application and the backend server.
- Handle asynchronous API calls for all CRUD operations related to credentials.

Key Features:

a. RESTful Endpoints:
    - GET: Retrieve a list of password entries from the backend.
    - POST: Create new password entries.
    - PUT: Update existing password records.

- DELETE: Remove a password entry from the database.

b. Error Handling & Loading States:

- Displays loading animations during API calls.
- Provides clear error messages and toast notifications in case of API failures.

c. Security:

- Transmits data securely over HTTPS.
- Includes necessary headers for Auth0 token validation.

5. Data Validation & Feedback Module

Responsibilities:

- Ensure that all user inputs meet predefined criteria before submission.
- Provide real-time validation feedback to the user.

Key Features:

a. Input Validation:

- Checks for empty fields, password complexity, and proper formatting of custom fields.
- Implements client-side validation with instant feedback.

b. Password Strength Meter:

- Analyses the strength of passwords based on length, complexity, and diversity of characters.
- Visually represents password strength (e.g., weak, moderate, strong).

c. Feedback Mechanisms:

- Utilizes toast notifications and inline messages to inform users about validation issues or successful operations.

6. Data Storage & Security Module

Responsibilities:

- Securely store user credentials and sensitive data in a backend database.
- Encrypt data both in transit and at rest to protect user information.

Key Features:

- Secure Storage:
    - Employs encryption protocols to safeguard stored passwords and custom fields.
    - Integrates with the backend database ensuring robust data integrity.
- Duplicate Prevention:
    - Checks for existing entries before saving new passwords.
    - Provides warnings to the user when duplicate credentials are detected.
- Data Integrity:
    - Ensures consistency between the client state and backend records through robust synchronization mechanisms.

7. Notification & User Feedback Module

Responsibilities:

- Enhance user interaction by providing timely feedback on actions and system states.
- Alert users to both successful operations and potential errors.

Key Features:

a. Toast Notifications:
    - Quick alerts for actions like saving, updating, or deleting entries.
    - Error messages for issues such as failed validations or API errors.

b. Loading Animations:
    - Visual indicators during data fetching and other asynchronous operations.
    - Helps maintain a smooth user experience during backend interactions.

8. Testing & Quality Assurance Module

Responsibilities:

- Ensure that each component and integration point works as expected.
- Maintain high-quality code through regular testing and updates.

Key Features:

a. Unit Testing:
    - Tests individual components (e.g., form validations, API methods).

b. Integration Testing:

- Validates the interactions between modules, particularly between the UI and backend API.

c. Security Testing:

- Conducts vulnerability assessments and penetration tests, especially for authentication and data encryption modules.

d. User Acceptance Testing (UAT):

- Gathers feedback from real users to fine-tune usability and ensure the system meets end-user needs.

Inter-module Interactions

- UI Module interacts with the Credential Management Module to display and update password entries.
- Authentication Module works across all modules to ensure secure access and protect sensitive operations.
- API Integration Module serves as a bridge between the frontend (UI, Credential Management) and the backend (Data Storage & Security).
- Data Validation & Feedback Module integrates with both the UI and Credential Management modules to ensure data integrity before reaching the backend.
- Notification Module collaborates with all modules to provide consistent feedback, ensuring users are always informed about the status of their actions.

Algorithm

Frontend:

Manager.jsx

```
import React from 'react';
import { useRef, useState, useEffect } from "react";
import PasswordTable from "./PasswordTable";
import { v4 as uuidv4 } from 'uuid';
import { toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Navbar } from "./Navbar";
import { useAuth0 } from "@auth0/auth0-react";
import { StrengthMeter } from './StrengthMeter';

const audience = import.meta.env.VITE_AUTH0_AUDIENCE;


export function Manager() {
    const [iconState, setIconState] = useState("hover-lashes"); //eye closed
useState hook
    const [typeInput, setTypeInput] = useState("password");  // show password
useState hook
    const ref = useRef();
    const [form, setForm] = useState({
        site: "",
        username: "",
        password: ""
    });
    const [inputFields, setInputFields] = useState([]); // store additional
Inputs
    const [passwordArray, setPasswordArray] = useState([]);
    const [editMode, setEditMode] = useState(null);   // Track if we are
editing
    const { getAccessTokenSilently } = useAuth0();
    const [isLoading, setIsLoading] = useState(false);  // Button loading
animation
    const [loadingPasswords, setLoadingPasswords] = useState(false);  //
Password fetching animation
    const [search, setSearch] = useState(""); // Search passwords
    const searchKeys = ["site", "username"];

    // fetch passwords from database
    const getPasswords = async () => {
        setLoadingPasswords(true);
        try {
            const token = await getAccessTokenSilently();
            const response = await fetch("https://lockcraft-
backend.onrender.com", {
                method: "GET",
                headers: {
                    Authorization: `Bearer ${token}`
                }
            });
```

```
            if (!response.ok) {
                setPasswordArray([]);
                return;
            }

            // Retrieve and set passwords
            let passwords = await response.json();
            if (Array.isArray(passwords) && passwords.length > 0) {
                setPasswordArray(passwords);
            } else {
                console.log('No passwords found');
                // Set empty array if no passwords were found
                setPasswordArray([]);
            }
            // setPasswordArray(response.data);
        } catch (error) {
            console.log('Error fetching passwords!', error);
        } finally {
            setLoadingPasswords(false);
        }

    }

    // fetch passwords on component mount
    useEffect(() => {
        getPasswords();
    }, [])


    // Search Passwords
    const handleSearch = (data) => {
        return data.filter((item) => {
            // Search site and username
            const matchSiteOrUsername = searchKeys.some(key =>
                item[key]?.toLowerCase().includes(search.toLowerCase())
            );

            // Search additonal fields (type and value)
            const matchAdditionalFields = item.additionalFields.some(field =>
                field.type?.toLowerCase().includes(search.toLowerCase()) ||
                field.value?.toLowerCase().includes(search.toLowerCase())
            );

            return matchSiteOrUsername || matchAdditionalFields;
        });
    }

    // Toggle show password
    const handleShowPassword = () => {
        setIconState(iconState === "morph-lashes-close" ? "morph-lashes" :
"morph-lashes-close");
        setTypeInput(typeInput === "password" ? "text" : "password");
    }
```

```
    // function to add new inputs
    const handleInputAdd = () => {
        // add a new set of input fields
        setInputFields([...inputFields, { type: "", value: "", isSensitive:
false }]);
    }

    // function to remove a particular input fields
    const handleRemoveField = (index) => {
        if (confirm('Are you sure you want to remove this field? This action
cannot be undone.')) {
            const removeField = inputFields.filter((_, i) => i !== index);
            setInputFields(removeField);
        }
    }

    const handleInputChange = (index, e) => {
        const updatedFields = [...inputFields];
        updatedFields[index] = { ...updatedFields[index], [e.target.name]:
e.target.value };
        setInputFields(updatedFields);
    }

    // handle sensitive checkbox
    const handleSensitiveChange = (index) => {
        const updatedFields = [...inputFields];
        updatedFields[index].isSensitive = !updatedFields[index].isSensitive;
        setInputFields(updatedFields);
    }

    // Save or Update password
    const handleSavePassword = async () => {
        // check input length
        if (form.site.length > 0 && form.username.length > 0 &&
form.password.length > 0) {

            // Start loading animation
            setIsLoading(true);

            try {
                const token = await getAccessTokenSilently();

                // store duplicate values
                const existingPassword = passwordArray.find((item) =>
                    item.site === form.site &&
                    item.username === form.username &&
                    item.password === form.password
                )

                // check if values already exist and is not Edit mode
                if (existingPassword && !editMode) {
                    // toast
                    toast.error('Password already exists!', {
                        position: "top-center",
                        pauseOnFocusLoss: false,
                        autoClose: 1000,
```

```
                        hideProgressBar: true,
                        closeOnClick: true,
                        theme: "light"
                    });
                } else if (editMode) {
                    // Update existing password
                    const updatedPassword = { ...form, additionalFields:
inputFields, _id: editMode };
                    await fetch(`https://lockcraft-backend.onrender.com`, {
                        method: "PUT",
                        headers: {
                            "Content-Type": "application/json",
                            Authorization: `Bearer ${token}`
                        },
                        body: JSON.stringify(updatedPassword)
                    })
                    // Toast
                    toast.success('Password updated!', {
                        position: "top-center",
                        pauseOnFocusLoss: false,
                        autoClose: 1000,
                        hideProgressBar: true,
                        closeOnClick: true,
                        theme: "light"
                    });

                    setPasswordArray((prev) =>
                        prev.map((password) =>
                            password._id === editMode ? updatedPassword :
password
                        ))
                    setEditMode(null) // Reset edit mode after saving
                    // Reset the input elements
                    setForm({ site: "", username: "", password: "" });
                    // Reset additional fields to one empty set
                    setInputFields([]);
                } else {
                    const newPassword = { ...form, additionalFields:
inputFields, id: uuidv4() };

                    await fetch("https://lockcraft-backend.onrender.com", {
                        method: "POST",
                        headers: {
                            "Content-Type": "application/json",
                            "Authorization": `Bearer ${token}`
                        },
                        body: JSON.stringify(newPassword)
                    })
                    setPasswordArray([...passwordArray, newPassword]);

                    // Toast
                    toast.success('Password saved!', {
                        position: "top-center",
                        autoClose: 1000,
                        hideProgressBar: true,
                        closeOnClick: true,
```

```
                        pauseOnHover: false,
                        draggable: true,
                        theme: "colored"
                    });

                    // Reset the input elements
                    setForm({ site: "", username: "", password: "" });
                    // Reset additional fields to one empty set
                    setInputFields([]);

                }
            } catch (error) {
                console.error("Error Saving password", error);
            } finally {
                // Stop loading animation
                setIsLoading(false);
            }

        } else {
            // toast
            toast.warning("Please fill in all fields!", {
                position: "top-center",
                pauseOnFocusLoss: false,
                autoClose: 3000,
                hideProgressBar: true,
                closeOnClick: true,
                draggable: true,
                theme: "light"
            });
        }

    }


    // input fields
    const handleChange = (e) => {
        setForm({ ...form, [e.target.name]: e.target.value })
    }

    return (
        <>
            {/* Background-color with radial gradient */}
            <div className="absolute top-0 z-[-2] h-screen w-screen bg-white
bg-[radial-gradient(ellipse_80%_80%_at_50%_-
20%,rgba(120,119,198,0.3),rgba(255,255,255,0))]"></div>

            <Navbar />

            <div className="container mx-auto max-w-4xl text-center py-5
overflow-y-auto scrollbar-hide">
                <h1 className="text-4xl font-bold hover:scale-110">
                    Pass<span className="text-green-600">Safe</span>
                </h1>
                <p className="text-green-900 text-lg text-center hover:scale-
105 md:pb-3">
                    Crafting Your Security
```

```
                </p>

                {/* Input Form */}
                <div className="flex flex-col p-4 text-black md:gap-8 gap-3
items-center relative mx-2">

                    <span className='md:w-1/3 px-2 py-1 hover:scale-y-110 h-
auto bg-white flex justify-between rounded-full items-center'>
                        <lord-icon
                            src="https://cdn.lordicon.com/wjyqkiew.json"
                            trigger="click"
                            colors="primary:#121331,secondary:#a39cf4"
                            style={{ "width": "25px", "height": "25px" }}
                        >
                        </lord-icon>

                        <input
                            placeholder="Search Password..."
                            onChange={(e) => setSearch(e.target.value)}
                            name="search"
                            className="rounded-full w-full px-1"
                            type="text"
                        />

                    </span>

                    <input
                        placeholder="Enter URL"
                        value={form.site}
                        onChange={handleChange}
                        name="site"
                        className="rounded-xl border border-slate-500 w-full
px-4 py-1 hover:scale-y-110 h-8"
                        type="text"
                    />

                    <div className="md:flex w-full justify-between gap-7">
                        <input
                            placeholder="Username"
                            value={form.username}
                            onChange={handleChange}
                            name="username"
                            className="rounded-xl border border-slate-500 w-
full px-4 py-1 hover:scale-y-110 h-8"
                            type="text"
                        />

                        <div className="relative md:w-2/4 md:py-0 py-2 mb-2">
                            <input
                                placeholder="Password"
                                value={form.password}
                                onChange={handleChange}
                                name="password"
                                className="rounded-xl border border-slate-500
w-full px-4 py-1 hover:scale-y-110 h-8"
                                type={typeInput}
```

```
                            />
                            <span
                                className="absolute right-2 cursor-pointer"
                                onClick={handleShowPassword}
                                ref={ref}
                            >
                                <lord-icon
                                    className="show"
                                    src="https://cdn.lordicon.com/vfczflna.jso
n"
                                    trigger="click"
                                    state={iconState}
                                ></lord-icon>
                            </span>

                            {/* Strength meter component */}
                            <StrengthMeter
                                password={form.password}
                            />
                        </div>
                    </div>

                    {/* Additional Input Fields */}
                    {inputFields.map((field, index) => (
                        <div key={index} className="md:flex w-full justify-
between gap-2">
                            <label className='block'>Type</label>
                            <input
                                placeholder="e.g. Email"
                                value={field.type || ""}
                                onChange={(e) => handleInputChange(index, e)}
                                name="type"
                                className="rounded-xl border border-slate-500
md:w-1/2 w-full px-4 py-1 hover:scale-y-110 h-8"
                                type="text"
                                autoFocus
                            />

                            <label className='block'>Value</label>
                            <input
                                placeholder="yourmail@mail.com"
                                value={field.value || ""}
                                onChange={(e) => handleInputChange(index, e)}
                                name="value"
                                className="rounded-xl border border-slate-500
md:w-1/2 w-full px-4 py-1 hover:scale-y-110 h-8"
                                type={!field.isSensitive ? "text" :
"password"}
                            />

                            <label htmlFor={`hide-${index}`}
className='opacity-50 mx-1'>Hide</label>
                            <input id={`hide-${index}`} type="checkbox"
checked={field.isSensitive} onClick={() => handleSensitiveChange(index)}
className='hide cursor-pointer' />
                            <span className='mx-4 md:mx-1'>
```

```
                                <lord-icon
                                    className="deleteIcon"
                                    onClick={() => handleRemoveField(index)}
                                    src="https://cdn.lordicon.com/skkahier.jso
n"
                                    trigger="hover"
                                    style={{ width: "20px", height: "20px",
paddingTop: "4px" }}>
                                </lord-icon>
                            </span>
                        </div>
                    ))}

                    {/* Add More Button */}
                    <div className="flex justify-center items-center bg-zinc-
400 rounded-full px-3 w-fit hover:bg-zinc-700 text-white hover:scale-y-110
gap-1 border border-slate-900 absolute left-5 bottom-12 my-3">
                        <button onClick={handleInputAdd}>Add More...</button>
                    </div>

                    {/* Save Password Button */}
                    <button
                        className="flex justify-center items-center bg-slate-
500 rounded-full px-6 py-1 w-fit hover:bg-slate-700 text-white hover:scale-y-
110 gap-1 border border-slate-900"
                        onClick={handleSavePassword}
                        disabled={isLoading}
                    >
                        {isLoading ? (
                            <span>
                                <lord-icon
                                    src="https://cdn.lordicon.com/ktsahwvc.jso
n"
                                    trigger="loop"
                                    delay="200"
                                    style={{ "width": "20px", "height":
"20px", "paddingTop": "4px" }}
                                >
                                </lord-icon>
                            </span>
                        ) : (
                            <>
                                <lord-icon
                                    src="https://cdn.lordicon.com/jgnvfzqg.jso
n"
                                    trigger="hover"
                                    colors="primary:#ffffff"
                                ></lord-icon>
                                Save
                            </>
                        )}
                    </button>
                </div>

                {/* Show Passwords */}
```

```
                <div className="showPasswords flex flex-col justify-center
items-center mx-4">
                    {loadingPasswords ?
                        <div className='md:mt-7 mt-3'>
                            <lord-icon
                                src="https://cdn.lordicon.com/ktsahwvc.json"
                                trigger="loop"
                                delay="200"
                                style={{ "width": "80px", "height": "80px",
"paddingTop": "4px" }}
                            >
                            </lord-icon>
                        </div>
                        : passwordArray.length === 0 ? (
                            <div>No Passwords to show. Add Some!</div>
                        ) : (
                            <PasswordTable
                                passwordArray={handleSearch(passwordArray)}  /
/ Pass filtered passwords
                                setPasswordArray={setPasswordArray}
                                setForm={setForm}
                                form={form}
                                setInputFields={setInputFields}
                                inputFields={inputFields}
                                getPasswords={getPasswords}
                                setEditMode={setEditMode}
                            // handleSearch={handleSearch(passwordArray)}
                            />
                        )}
                </div>
            </div>
        </>
    )
}
```

Navbar.jsx:

```
import React, { useState } from 'react';
import { Link, Route, Routes, useNavigate } from "react-router-dom";
import Generate from "./Generate";
import Profile from "./Profile";
import { useAuth0 } from "@auth0/auth0-react";

export function Navbar() {
    const { isAuthenticated, user } = useAuth0();
    const navigate = useNavigate();
    const [isMenuOpen, setIsMenuOpen] = useState(false);

    const handleShowProfile = () => {
        navigate('/profile');
    }
```

```
    const toggleMenu = () => {
        setIsMenuOpen(!isMenuOpen);
    };

    return (
        <>
            <nav className="bg-slate-300 flex justify-between items-center px-
4 h-14 ">
                <div className="logo-container hover:scale-110 flex gap-2
transition-transform">
                    <div className="logo font-bold md:text-2xl">
                        Pass
                        <span className="text-green-600">Safe</span>
                    </div>
                </div>

                {/* Hamburger Icon */}
                <button onClick={toggleMenu} className="block md:hidden">
                    <svg xmlns="http://www.w3.org/2000/svg" fill="none"
viewBox="0 0 24 24" strokeWidth={2} stroke="currentColor" className="w-6 h-6">
                        <path strokeLinecap="round" strokeLinejoin="round"
d="M4 6h16M4 12h16m-7 6h7" />
                    </svg>
                </button>

                {/* Navigation Links (visible on medium+ screens) */}
                <ul className="hidden md:flex md:gap-4 items-center">
                    <li className="flex gap-4 items-center">
                        <div className="flex gap-1 bg-green-400 rounded-2xl
ring-white p-1 border-2 border-opacity-75 hover:scale-110">


                        </div>
                        <Link className="hover:font-bold transition-transform
bg-slate-200 p-2 rounded-full border-slate-400 border" to="/generate">
Generate </Link>

                        {
                            isAuthenticated && (
                                <article className="profile cursor-pointer w-
auto hover:scale-110 hover:font-bold transition-transform" >
                                    <img
                                        src={user.picture ||
require(`./assets/noprofile.jpg`)}
                                        alt='Profile'
                                        className="rounded-full w-12 h-12
object-cover"
                                        onClick={handleShowProfile}
                                    />
                                </article>
                            )
                        }
                    </li>
                </ul>

                {/* Mobile Menu (visible when toggled) */}
```

```
                {isMenuOpen && (
                    <ul className="absolute top-14 left-0 w-full bg-slate-200
flex justify-around items-center p-2 md:hidden">
                        <li className="gap-4 items-center">
                            <Link className="hover:font-bold transition-
transform bg-slate-200 p-2 rounded-full border-slate-400 border"
to="/generate"> Generate </Link>
                        </li>
                        <li>
                            {
                                isAuthenticated && (
                                    <article className="profile cursor-pointer
w-auto hover:scale-110 hover:font-bold transition-transform" >
                                        <img
                                            src={user.picture ||
require(`./assets/noprofile.jpg`)}
                                            alt='Profile'
                                            className="rounded-full w-12 h-12
object-cover"
                                            onClick={handleShowProfile}
                                        />
                                    </article>
                                )
                            }
                        </li>
                        <li>
                            <div className="flex gap-1 bg-green-400 rounded-
2xl ring-white p-1 border-2 border-opacity-75 hover:scale-110">

                            </div>
                        </li>
                    </ul>
                )}
            </nav>

            <Routes>
                <Route path="/generate" element={<Generate />} />
                <Route path="/profile" element={<Profile />} />
            </Routes>
        </>
    )
}
```

PasswordTable.jsx

```
import React, { useRef, useState } from 'react';
import { toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Menu, MenuButton, MenuItem, MenuItems } from '@headlessui/react'
import { ChevronDownIcon } from '@heroicons/react/20/solid'
import { useAuth0 } from '@auth0/auth0-react';

export default function PasswordTable({ passwordArray, setPasswordArray,
setForm, setInputFields, setEditMode, getPasswords }) {

    const [iconState, setIconState] = useState("hover-lashes");
    const [showPassword, setShowPassword] = useState(false);
    const { getAccessTokenSilently } = useAuth0();
    const [visibleFields, setVisibleFields] = useState({});


    // Color according password strength
    function getStrengthBarColor(strength) {
        switch (strength) {
            case 1: return 'red';
            case 2: return 'orange';
            case 3: return 'yellow';
            case 4: return 'lightblue';
            case 5: return 'lightgreen';
            default: return 'gray';
        }
    }

    // Calculate strength of password
    const calculatePasswordStrength = (password) => {
        let strength = 0;
        if (password.length > 9) strength += 1;
        if (password.match(/[a-z]+/)) strength += 1;
        if (password.match(/[A-Z]+/)) strength += 1;
        if (password.match(/[0-9]+/)) strength += 1;
        if (password.match(/[^a-zA-Z0-9]+/)) strength += 1;
        return strength;
    }

    // Toggle show password
    const handleShowPassword = () => {
        setIconState(iconState === "morph-lashes-close" ? "morph-lashes" :
"morph-lashes-close");
        setShowPassword(!showPassword);
    }

    // Toggle show Field Values for each field
    const handleShowFieldValues = (fieldIndex) => {
        setVisibleFields((prev) => ({
            ...prev,
            [fieldIndex]: !prev[fieldIndex]  // Toggle visibility for specific
field
        }))
    }
```

```
    // handle copy function
    const handleCopyText = (text) => {
        navigator.clipboard.writeText(text);

        // show toast
        toast('Copied to Clipboard!', {
            position: "top-center",
            autoClose: 5000,
            hideProgressBar: false,
            closeOnClick: true,
            pauseOnHover: false,
            draggable: true,
            progress: undefined,
            theme: "light"
        });
    }


    //delete password
    const handleDeletePassword = async (id) => {
        let confirmation = confirm('Are you sure you want to remove this
password? This action cannot be undone.')
        if (confirmation) {
            const token = await getAccessTokenSilently();
            try {
                const response = await fetch("https://lockcraft-
backend.onrender.com", {
                    method: "DELETE",
                    headers: {
                        "Content-Type": "application/json",
                        Authorization: `Bearer ${token}`
                    },
                    body: JSON.stringify({ _id: id })
                });

                if (response.ok) {
                    setPasswordArray(passwordArray.filter(item => item._id !==
id));
                    toast.success('Password deleted!', {
                        position: "top-center",
                        autoClose: 5000,
                        hideProgressBar: false,
                        closeOnClick: true,
                        pauseOnHover: false,
                        draggable: true,
                        progress: undefined,
                        theme: "light"
                    });
                    // Refresh the list after deletion
                    getPasswords();
                } else {
                    toast.error('Failed to delete password', {
                        position: "top-center",
                        autoClose: 5000,
```

```
                        hideProgressBar: false,
                        closeOnClick: true,
                        pauseOnHover: false,
                        draggable: true,
                        progress: undefined,
                        theme: "light"
                    });
                }
            } catch (error) {
                toast.error('An error occurred!', {
                    position: "top-center",
                    autoClose: 5000,
                    hideProgressBar: false,
                    closeOnClick: true,
                    pauseOnHover: false,
                    draggable: true,
                    progress: undefined,
                    theme: "light"
                });
            }
        }
    }

    // Handle edit password
    const handleEditPassword = (password) => {
        setForm({
            site: password.site,
            username: password.username,
            password: password.password
        });
        setInputFields(password.additionalFields || []);
        // Set the current password being edited
        setEditMode(password._id);
    }

    return (
        <>
            <h2 className="font-bold text-xl py-4 hover:scale-110">Your
Passwords</h2>
            {Array.isArray(passwordArray) && passwordArray.length > 0 && (
                passwordArray.map((item, index) => {
                    const strength = calculatePasswordStrength(item.password);
                    const menuColor = getStrengthBarColor(strength);

                    return (
                        <Menu as="div" className="relative inline-block text-
left w-full max-w-3xl py-1 mx-2" key={item._id}>
                            <div>
                                <MenuButton className="flex justify-between w-
full gap-x-1.5 rounded-xl bg-slate-200 px-3 py-2 text-sm text-gray-900 shadow-
sm ring-1 ring-inset ring-gray-300 hover:bg-slate-300 hover:scale-105"
                                    style={{ borderColor: menuColor,
borderWidth: '1px', filter: `drop-shadow( 0 0 0 ${menuColor})` }}
                                >
                                    <div className="dropdown-Menu">
```

```
                                        <span className="font-semibold text-
base">{item.site}</span>
                                        <div>{item.username}</div>
                                    </div>

                                    <div className="dropdown-Logo my-auto">
                                        <ChevronDownIcon aria-hidden="true"
className="-mr-1 h-5 w-5 text-gray-400" />
                                    </div>
                                </MenuButton>
                            </div>

                            <MenuItems
                                transition
                                className="absolute z-10 mt-2 w-full origin-
top-right divide-y divide-slate-300 rounded-md bg-slate-100 shadow-lg ring-1
ring-black ring-opacity-5 transition focus:outline-none data-[closed]:scale-95
data-[closed]:transform data-[closed]:opacity-0 data-[enter]:duration-100
data-[leave]:duration-75 data-[enter]:ease-out data-[leave]:ease-in"
                            >

                                {/* Site */}
                                <div className="flex justify-between items-
center p-2 hover:bg-slate-200">
                                    <span className="font-semibold text-
base">Site:</span>
                                    <div className="flex items-center gap-2">
                                        <span className="text-gray-
700">{item.site}</span>
                                        <div className="copyIcon cursor-
pointer" onClick={() => { handleCopyText(item.site) }}>
                                            <lord-icon
                                                src="https://cdn.lordicon.com/
depeqmsz.json"
                                                trigger="hover"
                                                style={{ width: "20px",
height: "20px", paddingTop: "4px" }}>
                                            </lord-icon>
                                        </div>
                                    </div>
                                </div>

                                {/* Username */}
                                <div className="flex justify-between items-
center p-2 hover:bg-slate-200">
                                    <span className="font-semibold text-
base">Username:</span>
                                    <div className="flex items-center gap-2">
                                        <span className="text-gray-
700">{item.username}</span>
                                        <div className="copyIcon cursor-
pointer" onClick={() => { handleCopyText(item.username) }}>
                                            <lord-icon
                                                src="https://cdn.lordicon.com/
depeqmsz.json"
                                                trigger="hover"
```

```
                                                style={{ width: "20px",
height: "20px", paddingTop: "4px" }}>
                                            </lord-icon>
                                        </div>
                                    </div>
                                </div>

                                {/* Password */}
                                <div className="flex justify-between items-
center p-2 hover:bg-slate-200">
                                    <span className="font-semibold text-
base">Password:</span>
                                    <div className="flex items-center gap-2">
                                        <span className="text-gray-700"
                                            style={{ textDecoration:
"underline", textDecorationColor: menuColor}}
                                        >
                                            {showPassword ? item.password :
"*".repeat(item.password.length)}
                                        </span>

                                        <div className="copyIcon cursor-
pointer" onClick={() => { handleCopyText(item.password) }}>
                                            <lord-icon
                                                src="https://cdn.lordicon.com/
depeqmsz.json"

                                                trigger="hover"
                                                style={{ width: "20px",
height: "20px", paddingTop: "4px" }}>
                                            </lord-icon>
                                        </div>

                                        <span
                                            className='showIcon cursor-
pointer'

                                            onClick={handleShowPassword}
                                        // ref={refPassword}
                                        >
                                            <lord-icon
                                                className="show"
                                                src="https://cdn.lordicon.com/
vfczflna.json"

                                                trigger="click"
                                                state={iconState}
                                            ></lord-icon>
                                        </span>
                                    </div>
                                </div>

                                {/* Additional Fields */}
                                {item.additionalFields.map((field, idx) => (
                                    <div className="flex justify-between
items-center p-2 hover:bg-slate-200" key={idx}>
                                        <span className="font-semibold text-
base">{field.type}:</span>
```

```
                                    <div className="flex items-center gap-
2">

                                        <span className="text-gray-700">
                                            {field.isSensitive &&
!visibleFields[idx] ? "*".repeat(field.value.length) : field.value}
                                        </span>
                                        <div className="copyIcon cursor-
pointer" onClick={() => { handleCopyText(field.value) }}>
                                            <lord-icon
                                                src="https://cdn.lordicon.
com/depeqmsz.json"

                                                trigger="hover"
                                                style={{ width: "20px",
height: "20px", paddingTop: "4px" }}>
                                            </lord-icon>
                                        </div>

                                        {/* If isSensitive is true than
show icon */}
                                        {field.isSensitive && (
                                            <span
                                                className='showIcon
cursor-pointer'

                                                onClick={() =>
handleShowFieldValues(idx)}
                                            // ref={refField}
                                            >
                                                <lord-icon
                                                    className="show"
                                                    src="https://cdn.lordi
con.com/vfczflna.json"

                                                    trigger="click"
                                                    state={visibleFields[i
dx] ? "morph-lashes-close" : "morph-lashes"}
                                                ></lord-icon>
                                            </span>
                                        )}
                                    </div>
                                </div>
                            ))}

                            {/* Actions */}
                            <div className="actions items-center justify-
center flex p-2">
                                <div className="placeholder text-xs">
                                    <span className="opacity-
50">Actions:</span>
                                    <div className="py-1 menu-password
flex">
                                        <span className="px-2 cursor-
pointer">
                                            <lord-icon
                                                className="editIcon"
                                                onClick={() => {
handleEditPassword(item) }}
```

```
                                                    src="https://cdn.lordicon.
com/zfzufhzk.json"
                                                    trigger="hover"
                                                    colors="primary:#121131,se
condary:#242424,tertiary:#ebe6ef,quaternary:#f9c9c0,quinary:#3a3347"
                                                    style={{ width: "20px",
height: "20px", paddingTop: "4px" }}>
                                                </lord-icon>
                                            </span>

                                            <span className="px-2 cursor-
pointer">
                                                <lord-icon
                                                    className="deleteIcon"
                                                    onClick={() => {
handleDeletePassword(item._id) }}
                                                    src="https://cdn.lordicon.
com/skkahier.json"
                                                    trigger="hover"
                                                    style={{ width: "20px",
height: "20px", paddingTop: "4px" }}>
                                                </lord-icon>
                                            </span>
                                        </div>
                                    </div>
                                </div>
                            </MenuItems>
                        </Menu>
                    )
                })
            )}
        </>
    );

}
```

StrengthMeter.jsx

```
import React from 'react'

export const StrengthMeter = ({ password }) => {


    const calculatePasswordStrength = () => {
        let strength = 0;

        if (password.length > 9) strength += 1;
        if (password.match(/[a-z]+/)) strength += 1;
        if (password.match(/[A-Z]+/)) strength += 1;
        if (password.match(/[0-9]+/)) strength += 1;
        if (password.match(/[^a-zA-Z0-9]+/)) strength += 1;
        return strength;
    }
```

```
    function getStrengthBarColor(strength) {
        switch (strength) {
            case 1: return 'red';
            case 2: return 'orange';
            case 3: return 'yellow';
            case 4: return 'lightblue';
            case 5: return 'lightgreen';
            default: return 'gray';
        }
    }

    const strength = calculatePasswordStrength();
    const strengthBarColor = getStrengthBarColor(strength);
    const strengthBarStyle = {
        width: `${(strength / 5) * 100}%`,
        backgroundColor: strengthBarColor,
        filter: strength > 0 ? `drop-shadow( 0 0 5px ${strengthBarColor})` :
'none',
        transition: `width 0.5s ease-in-out,
                background-color 0.5s ease-in-out,
                filter 0.5s ease-in-out`
    }

    return (
        <div className='strengthBar rounded-xl my-1 h-1'
            style={strengthBarStyle}
        >
        </div>
    )
}
```

App.jsx

```
import React from 'react';
import { BrowserRouter, Navigate, Route, Routes } from 'react-router-dom'
import './App.css'
import { Manager } from './components/Manager'
import { Navbar } from './components/Navbar'
import Generate from './components/Generate'
import { useAuth0 } from '@auth0/auth0-react'
import Profile from './components/Profile'
import { ToastContainer } from 'react-toastify'


function App() {
  const { isAuthenticated, isLoading, error } = useAuth0();

  if (isLoading) {
    return <p className='absolute top-1/3'
      style={{ "left": "45%" }}
    > <lord-icon
      src="https://cdn.lordicon.com/ktsahwvc.json"
      trigger="loop"
```

```
      delay="200"
      style={{ "width": "80px", "height": "80px", "paddingTop": "4px" }}
    >
      </lord-icon> </p>
  }

  if (error) {
    return <div className='absolute top-1/3 p-5'> Error: {error.message}
</div>
  }

  return (
    <main>
      {/* Toast */}
      <ToastContainer
        position="top-center"
        autoClose={5000}
        hideProgressBar={false}
        newestOnTop={false}
        closeOnClick
        rtl={false}
        pauseOnFocusLoss
        draggable
        pauseOnHover
        theme="light"
        transition="colored"
      />
      <ToastContainer />

      <Routes>

        // Authentication
        {isAuthenticated ? (
          <>
            {/* {console.log(isAuthenticated, "isit")} */}
            <Route path="" element={<Manager />} />
            <Route path="/profile" element={<Profile />} />
            <Route path="/generate" element={<Generate />} />
            <Route path='*' element={<Navigate to='/' />} />
          </>
        ) : (
          <>
            <Route path="/generate" element={<Generate />} />
            <Route path="*" element={<Navigate to="/generate" />} />
          </>
        )}
        {/* </Switch> */}
      </Routes>
    </main>
  );
}

export default App
```

Backend:

Server.js

```
const express = require('express');
const mongoose = require('mongoose');
const { auth } = require('express-oauth2-jwt-bearer');
const dotenv = require('dotenv');
dotenv.config();
const bodyparser = require('body-parser');
const cors = require('cors');
const crypto = require('crypto');
const { type } = require('os');
const domain = process.env.AUTH0_DOMAIN;
const audience = process.env.AUTH0_AUDIENCE;
const mongoUri = process.env.MONGO_URI;

const app = express();
const port = process.env.PORT || 5000;

const allowedOrigins = ['https://lockcraft.onrender.com',
'http://localhost:5173'];


// Auth0 configuration
const checkJwt = auth({
    audience: audience,
    issuerBaseURL: `https://${domain}/`,
    tokenSigningAlg: 'RS256'
});

// CORS Middleware
app.use(cors({
    origin: allowedOrigins,
    methods: ['GET', 'POST', 'PUT', 'DELETE'],
    credentials: true,
}));

app.set('trust proxy', true);

// Handle pre-flight requests
app.options('*', cors());

// Middleware
app.use(bodyparser.json());



// MongoDB connection
mongoose.connect(mongoUri)
    .then(() => console.log('MongoDB connected'))
    .catch((err) => console.log('MongoDB connection error ', err))

    mongoose.connection.on('connected', () => {
        console.log('Mongoose connected to mongoDB');
```

```
    })

    mongoose.connection.on('error', (err) => {
        console.log('Mongoose connection error', err);
    })

    mongoose.connection.on('disconnected', () => {
        console.log('Mongoose disconnected from mongoDB');
    })


// Define Encryption settings
const algorithm = 'aes-256-cbc';  // Encryption algorithm
const secretKey = process.env.ENCRYPTION_KEY;  // Key for encryption
const iv = crypto.randomBytes(16);   // Initialization vector

// Encrypt function
function handleEncrypt (text) {
    const cipher = crypto.createCipheriv(algorithm, Buffer.from(secretKey,
'hex'), iv);
    let encrypted = cipher.update(text, 'utf8', 'hex');
    encrypted += cipher.final('hex');
    return `${iv.toString('hex')}:${encrypted}`;  // Include IV with the
encrypted data
}

// Decrypt function
function handleDecrypt (text) {
    const [ivText, encryptedText] = text.split(':');
    const decipher = crypto.createDecipheriv(algorithm, Buffer.from(secretKey,
'hex'), Buffer.from(ivText, 'hex'));
    let decrypted = decipher.update(encryptedText, 'hex', 'utf8');
    decrypted += decipher.final('utf8');
    return decrypted;
}

// Password Schema
const PasswordSchema = new mongoose.Schema({
    site: {type: String, required: true},
    username: {type: String, required: true},
    password: {type: String, required: true},
    additionalFields: [{
        type: { type: String },
        value: { type: String },
        isSensitive: { type: Boolean, default: false }
    }],
    userId: {type: String, require: true}
});

const Password = mongoose.model('Password', PasswordSchema);


// Get password
app.get('/', checkJwt, async (req, res) => {
    try{
```

```
        const userId = req.auth.payload.sub  // Auth0 user ID( Get User ID
from the token)
        const passwords = await Password.find({ userId });

        if(!passwords || passwords.length === 0) {
            return res.status(404).json({ message: 'No passwords found' });
        }

        // Decrypt each password and additional field before sending it
        const decryptedPasswords = passwords.map(p => ({
            ...p._doc, // Copy other fields
            site: handleDecrypt(p.site),
            username: handleDecrypt(p.username),
            password: handleDecrypt(p.password),  // Decrypt the main password
            additionalFields: p.additionalFields.map(field => ({
                type: handleDecrypt(field.type),
                // value: field.isSensitive ? handleDecrypt(field.value) :
field.value,  // Decrypt if sensitive
                value: handleDecrypt(field.value),
                isSensitive: field.isSensitive    // Keep the isSensitive flag
            }))
        }))

        res.json(decryptedPasswords);
    } catch (error){
        console.log('Error fetching passwords', error);
        res.status(500).json({ message: `Error fetching passwords`, error});
    }
})

// Save password
app.post('/', checkJwt, async (req, res) => {
    const { site, username, password, additionalFields } = req.body;

    try {
        const encryptedSite = handleEncrypt(site);
        const encryptedUsername = handleEncrypt(username);
        const encryptedPassword = handleEncrypt(password);
        const userId = req.auth.payload.sub;
        // console.log('post request body:-- ', req.body);
        // console.log('User authentication:-- ', req.auth.payload.sub);
        const processedAdditionalFields = additionalFields.map(field => ({
            type: handleEncrypt(field.type),
            // value: field.isSensitive ? handleEncrypt(field.value) :
field.value,      // Encrypt only if sensitive,
            value: handleEncrypt(field.value),
            isSensitive: field.isSensitive  // Keep the isSensitive flag
        }))

        const newPassword = new Password({
            site: encryptedSite,
            username: encryptedUsername,
            password: encryptedPassword,
            additionalFields: processedAdditionalFields,
            userId
        });
```

```
        await newPassword.save();
        res.status(201).json(newPassword);
    } catch (error) {
        res.status(500).json({message: 'Error saving passwords', error});
    }
})


// Update password
app.put('/', checkJwt, async (req, res) => {
    const { _id, site, username, password, additionalFields } = req.body;
    try {
        const userId = req.auth.payload.sub;
        const encryptedSite = handleEncrypt(site);
        const encryptedUsername = handleEncrypt(username);
        const encryptedPassword = handleEncrypt(password);
        const processedAdditionalFields = additionalFields.map(field => ({
            type: handleEncrypt(field.type),
            // value: field.isSensitive ? handleEncrypt(field.value) :
field.value,      // Encrypt only if sensitive,
            value: handleEncrypt(field.value),
            isSensitive: field.isSensitive
        }))

        const updatedPassword = await Password.findOneAndUpdate(
            { _id, userId },   // Only update if the user owns the password
            { site: encryptedSite, username: encryptedUsername, password:
encryptedPassword, additionalFields: processedAdditionalFields },
            { new: true }
        );

        if(!updatedPassword) {
            return res.status(404).json({ message: 'Password not found'});
        }

        res.json(updatedPassword);
    } catch (error) {
        res.status(500).json({ message: 'Error updating password', error });
    }
})

// Delete password by ID
app.delete('/', checkJwt, async (req, res) => {
    const { _id } = req.body; // Take the ID from the request body

    try {
        const userId = req.auth.payload.sub  // Auth0 user ID
        const deletedPassword = await Password.findOneAndDelete({ _id, userId
});

        if (!deletedPassword) {
            return res.status(404).json({ message: `Password not found` });
        }

        res.json({ message: `Password deleted successfully` });
    } catch (error) {
```

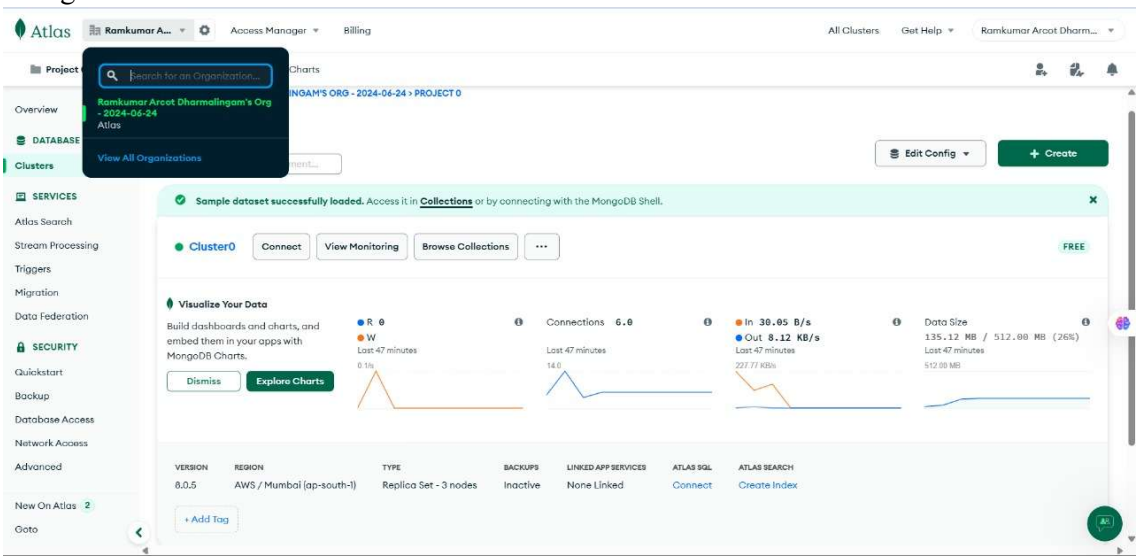
```
        // console.log(error);
        res.status(500).json({ message: `Error deleting password`, error });
    }
})

// Start the server
app.listen(port, () => {
    console.log(`Server running on ${port}`);
})
```

.env

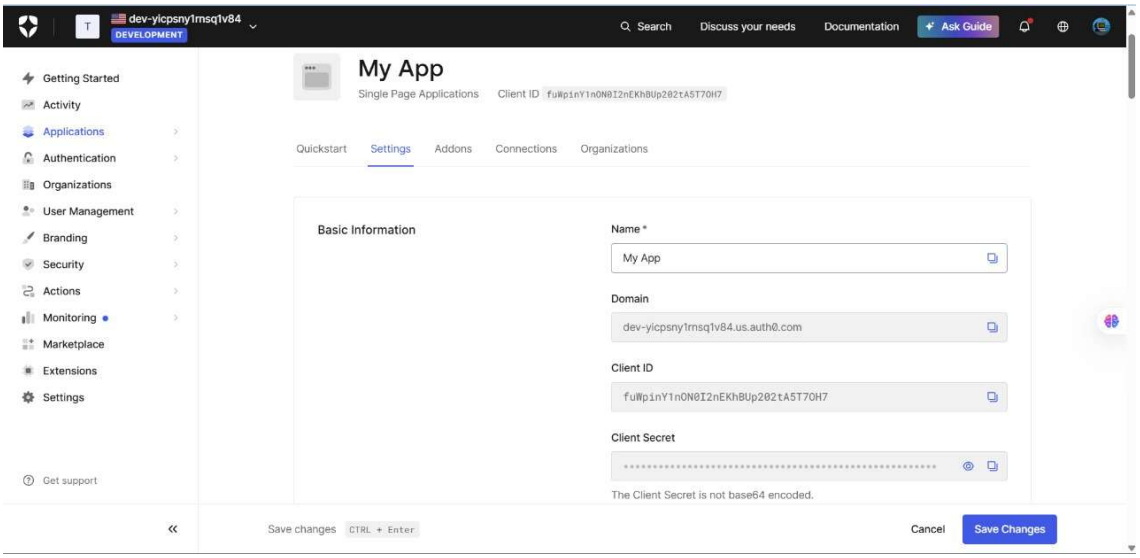
```
AUTH0_DOMAIN=dev-yicpsny1rnsq1v84.us.auth0.com
AUTH0_AUDIENCE=https://dev-yicpsny1rnsq1v84.us.auth0.com/api/v2/
VITE_AUTH0_DOMAIN=dev-yicpsny1rnsq1v84.us.auth0.com
VITE_AUTH0_CLIENT_ID=your-auth0-client-id
VITE_AUTH0_AUDIENCE=https://dev-yicpsny1rnsq1v84.us.auth0.com/api/v2/
MONGO_URI=mongodb+srv://ramkumararcot2022:shiva2004@cluster0.zaxbykl.mongodb.n
et/?retryWrites=true&w=majority&appName=Cluster0
ENCRYPTION_KEY=your-secure-encryption-key
PORT=5000
```

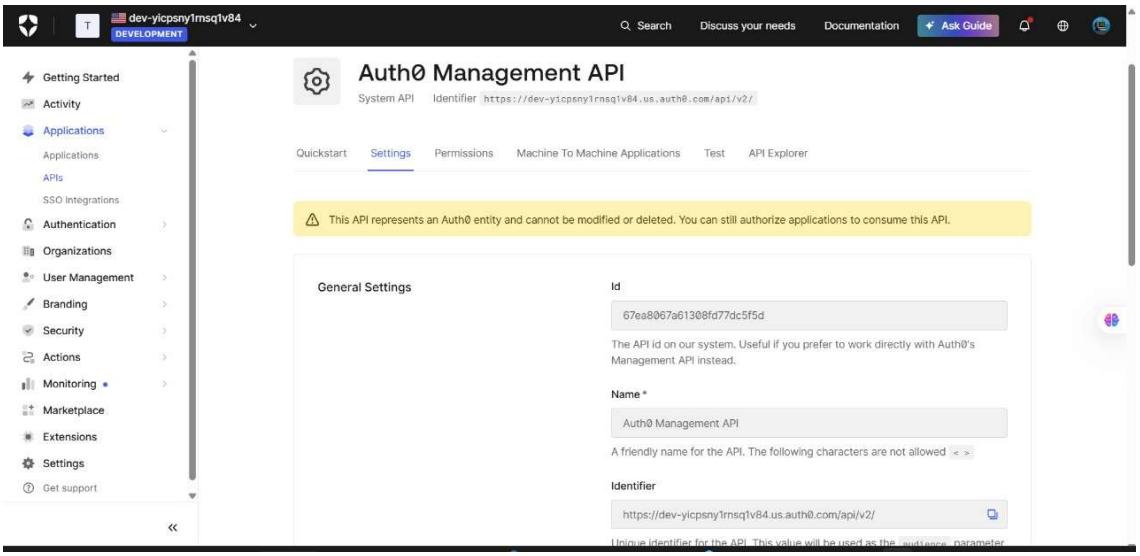
Input and Output

MongoDB

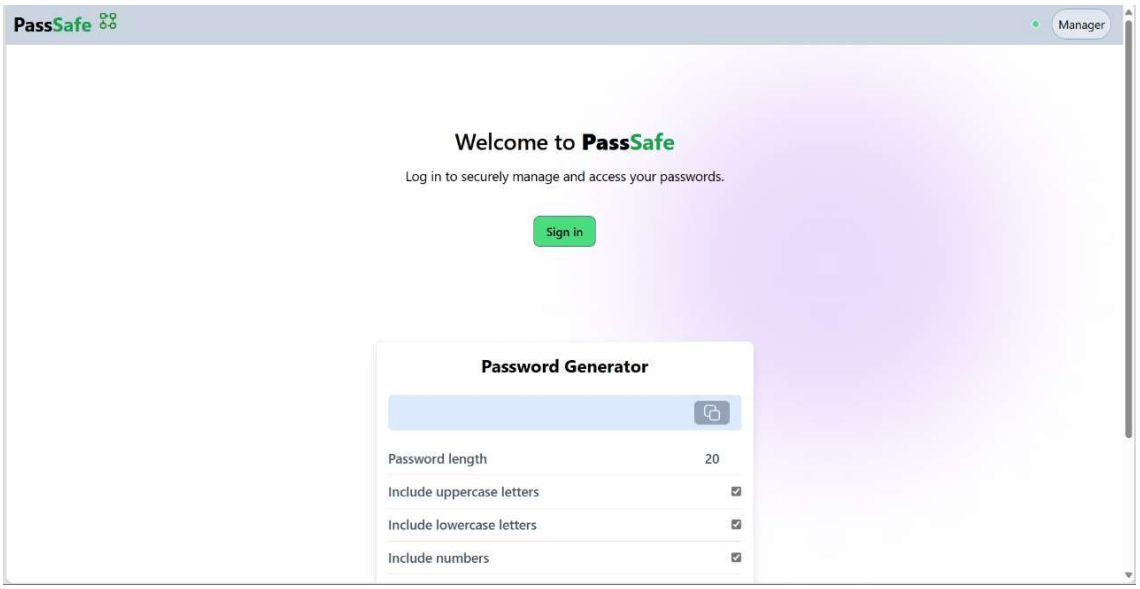


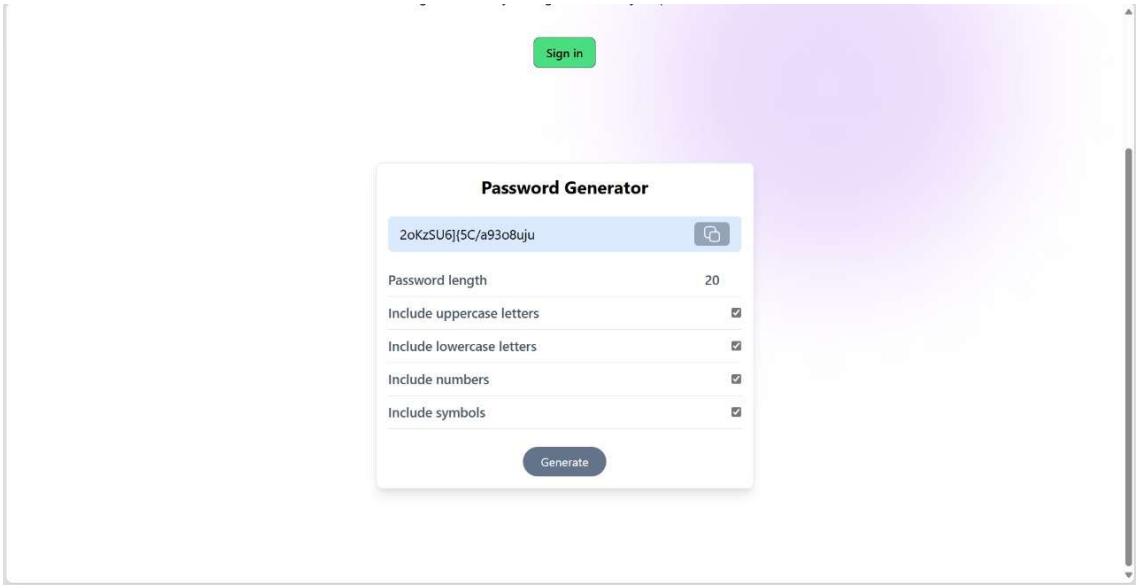
Auth0 API

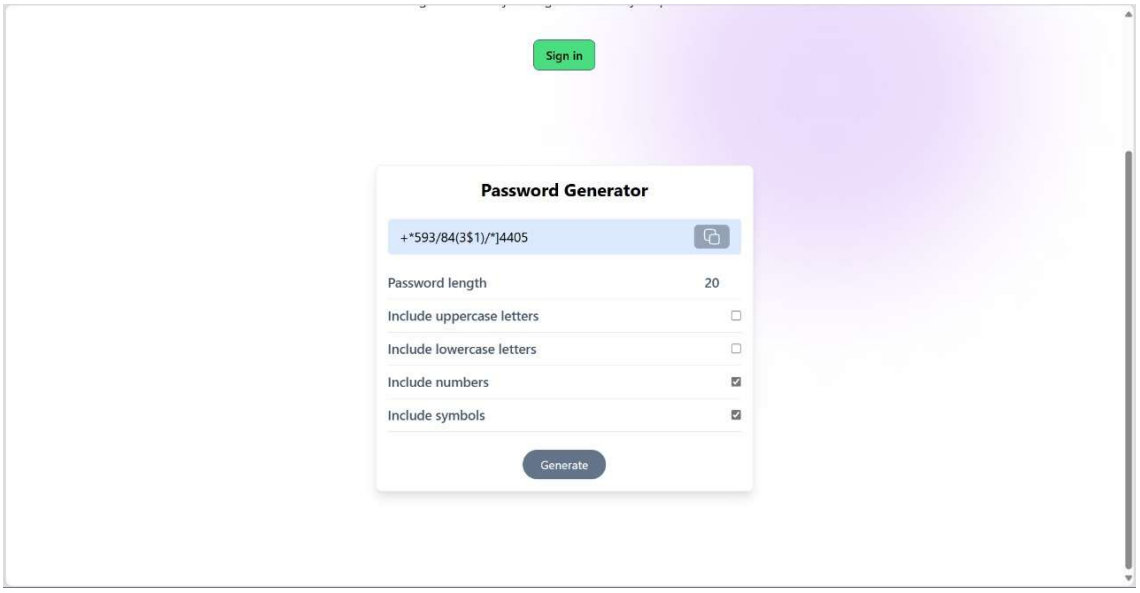




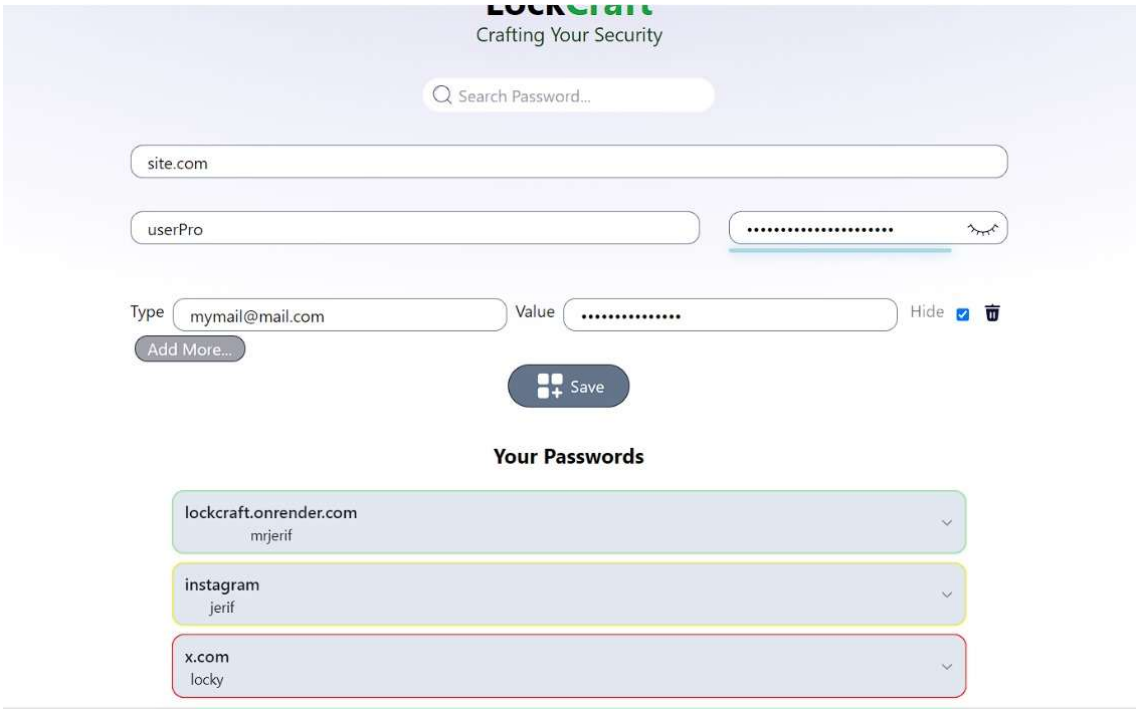
Password Generator

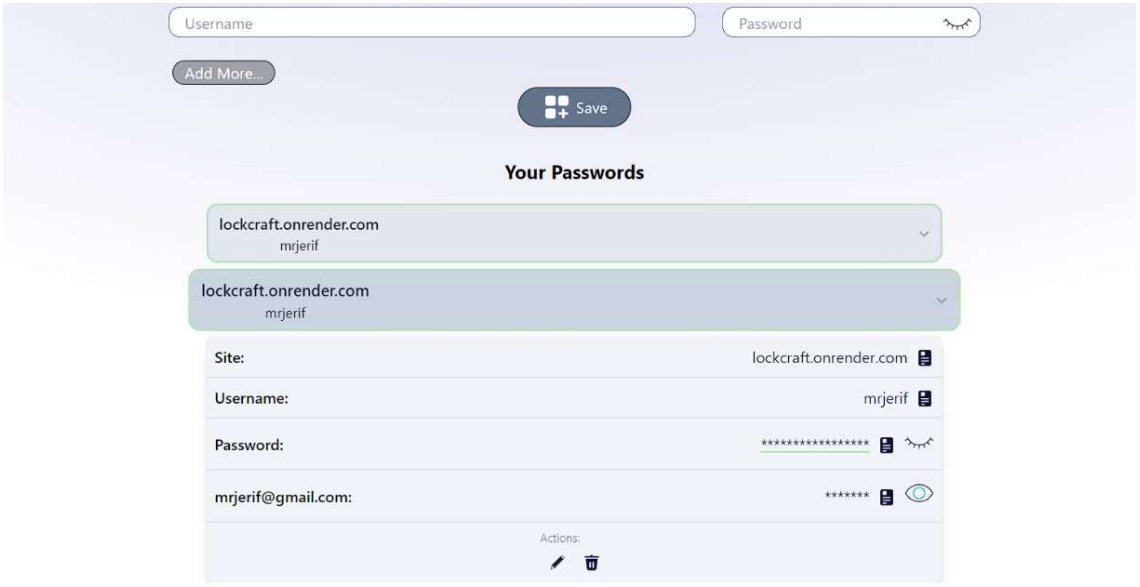



Sign in
Password Generator
2oKzSU6]{5C/a93o8uju
Password length 20
Include uppercase letters
Include lowercase letters
Include numbers
Include symbols
Generate

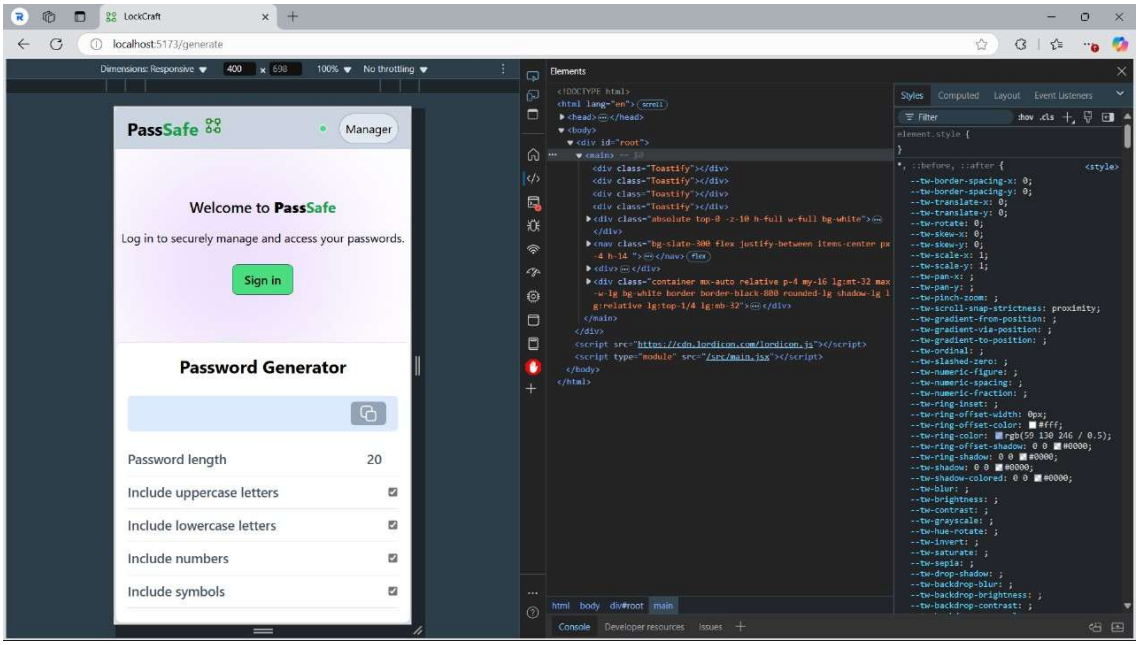

Sign in
Password Generator
+*593/84(3$1)/*]4405
Password length 20
Include uppercase letters
Include lowercase letters
Include numbers
Include symbols
Generate

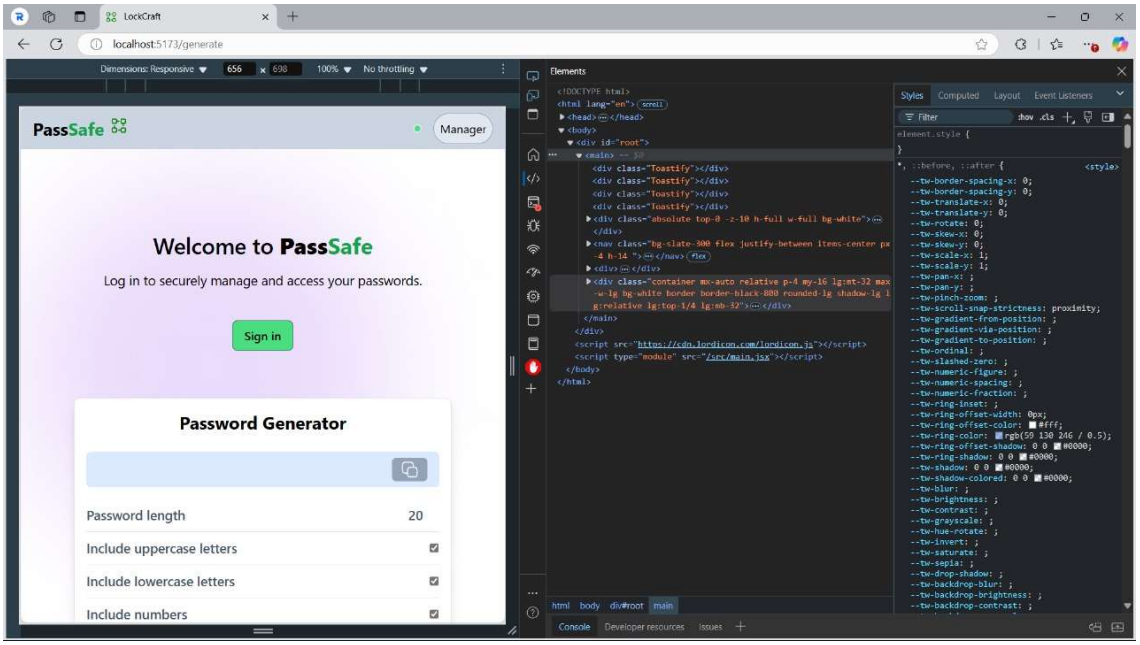
Password Manager





Responsive Design





Conclusion

In conclusion, the Password Management System project successfully delivers a secure, intuitive, and efficient solution for managing digital credentials. The platform not only centralizes password storage but also integrates advanced security features that address the multifaceted challenges faced by users in today's digital landscape. By incorporating an easy-to-use interface with essential functionalities—such as search capabilities, password visibility toggles, and real-time feedback through toast notifications—the system ensures that users can quickly locate and manage their credentials with minimal effort.

Furthermore, the integration of a robust password strength meter encourages the creation of strong, secure passwords, significantly reducing the risk of breaches due to weak credentials. The flexibility to add, edit, and remove custom fields further enhances user control over their data, making the system adaptable to varied user needs. Behind the scenes, secure data handling through Auth0 authentication and efficient backend processes ensures that all interactions are safe and reliable, preventing issues like duplicate entries and unauthorized access.

The project's design emphasizes both security and usability, ensuring that users experience seamless interactions without compromising on safety. With clear visual cues, responsive design, and smooth API interactions, the Password Management System stands out as a comprehensive solution that meets modern cybersecurity challenges while maintaining a user-centric approach. Ultimately, this project not only simplifies the complexities of password management but also builds a foundation for more secure digital practices in an increasingly interconnected world.