Final Report

for

Cryptographically - Secure Password Manager

Version 1.0

Prepared by

Anushka Singh | 2021IMG-014
Avijeet Jain | 2021IMG-018
Yuvraj Kumar | 2021IMG-066

ABV-Indian Institute of Information Technology & Management, Gwalior

18th April 2022

Submitted in complete fulfillment

Of the requirements of

IMIT-2202 Software Engineering Lab

Table of Contents

1. Introduction

1.1 About Project

A cryptographically secured password manager provides a secure and convenient way for users to store and manage their passwords and other sensitive information. A password manager typically holds login credentials for various websites and applications and can also store sensitive information such as credit card numbers, social security numbers, and other personal information.

Passman is a password manager built with the MERN (MongoDB, Express, React, Node.js) stack that uses the RSA (Rivest–Shamir–Adleman) cryptographic algorithm for encryption and decryption of user passwords.

This application provides a secure and user-friendly interface for storing and managing passwords, ensuring that users can have access to their accounts' credentials with ease.

Features of PassMan

- Secure storage of user passwords in an encrypted database
- Use of RSA algorithm to encrypt and decrypt user passwords
- Automatic password generation for strong and unique passwords
- Simple and intuitive user interface for easy password management
- Public Private key encryction for maximum security
- Private key is not stored anywhere in our databases

1.2 Git Repository

Link for GitHub Repository: https://github.com/Spedrick/PassMan

Link to GitHub profiles of group members:

- Yuvraj Kumar (Spedrick) - https://github.com/Spedrick
- Avijeet Jain (AvijeetJain) - https://github.com/AvijeetJain
- Anushka Singh (singh-anushka) - https://github.com/singh-anushka

Yuvraj Kumar established the git repository (2020BCS-044). Our repository's name is PassMan, which is also the name of our project. Yuvraj Kumar is the owner of the PassMan repository, and both team members, Avijeet Jain (2021IMG-018) and Anushka Singh (2021IMG-014) have admin access.

2. Details of Implemented Modules

The client-server architecture is a common approach for building software systems, including password manager applications. In this architecture, the software is divided into two main components: the client and the server.

The client module of the password manager application runs on the user's device, such as a laptop, smartphone, or tablet. It provides a graphical user interface (GUI) that allows the user to interact with the password manager system. The client module is responsible for managing the user's passwords and associated information, such as website URLs, login credentials, and security questions. The client module communicates with the server module to store and retrieve password data securely.

The server module of the password manager application runs on a remote server hosted by the service provider. The server module is responsible for securely storing the user's password data and providing authentication and authorization services. The server module communicates with the client module to receive user requests and respond with the appropriate data. The server module also includes security measures to protect the user's password data from unauthorized access, such as encryption, firewalls, and intrusion detection systems.

2.1. Client Module

The client module is responsible for the presentation layer and user interface of the password manager application. It provides a graphical user interface (GUI) that allows

users to interact with the system. The client module communicates with the server module to manage and store password data securely.

The main functions of the client module are as follows:

1. User Authentication: The client module is responsible for authenticating the user to access their password data. This involves verifying the user's identity through a secure login process, such as a username and password or biometric authentication.
2. Password Management: The client module allows users to create, store, and manage their passwords securely. Users can generate strong passwords using the password generator feature and organize their passwords by categories or tags.
3. Password Retrieval: The client module enables users to retrieve their passwords quickly and easily when needed. Users can search for passwords by keywords, categories, or tags.
4. Password Sharing: The client module allows users to share their passwords securely with other users or groups. Users can control the level of access granted to each recipient.
5. Security Features: The client module includes several security features to protect the user's password data, such as encryption, two-factor authentication, and session management.

2.2. Server Module

The server module is responsible for the application's business logic, data storage, and security. It provides a secure backend for the password manager application and communicates with the client module to manage and store password data.

The main functions of the server module are as follows:

1. Data Storage: The server module stores password data securely in a database or file system. The password data is encrypted using strong cryptographic algorithms to prevent unauthorized access.
2. User Authentication: The server module authenticates the user's identity and grants access to their password data. This involves verifying the user's credentials and validating their permissions to access specific password data.
3. Password Management: The server module performs several password management tasks, such as password validation, expiration, and complexity checks. It also manages password policies and rules to enforce strong password standards.
4. Security Features: The server module includes several security features to protect the user's password data, such as encryption, access control, and intrusion detection. It also implements secure communication protocols, such as HTTPS, to ensure data privacy and integrity.
5. Backup and Recovery: The server module includes backup and recovery mechanisms to prevent data loss in case of system failures or disasters. It also performs regular backups and updates to ensure data availability and integrity.

The client-server architecture of the password manager application provides several benefits, such as centralized management of password data, improved security, and increased accessibility. By separating the client and server modules, the application can provide a consistent user experience across multiple devices, platforms, and locations. The server module can also provide additional services, such as backup and recovery, reporting, and analysis, to enhance the functionality of the password manager application.

Overall, the client and server modules of the project PassMan - A Cryptographically Secured Password Manager project work together to provide a secure and user-friendly password management system. The client module enables users to manage their passwords easily, while the server module ensures the security and reliability of the password data.

3. Individual Contributions and Activities

3.1. Yuvraj Kumar | 2021IMG-066

- **RSA module for password encryption and decryption**
  Implemented RSA algorithm which is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and the Private key is kept private.

- **MongoDB as the database**

  Implemented MongoDB which is a NoSQL database where each record is a document comprising of key-value pairs that are similar to JSON (JavaScript Object Notation) objects. MongoDB is flexible and allows its users to create schema, databases, tables, etc. Documents that are identifiable by a primary key make up the basic unit of MongoDB.

- **Express as the web application framework**
  Express is a Node.js framework. Rather than writing the code using Node.js and creating loads of Node modules, Express makes it simpler and easier to write the back-end code. Express helps in designing great web applications and APIs. Express supports many middlewares which makes the code shorter and easier to write.

- **Axios for HTTP requests**

  Axios is a client HTTP API based on the XMLHttpRequest interface provided by browsers. The most common way for frontend programs to communicate with servers is through the HTTP protocol.

3.2. Avijeet Jain | 2021IMG-018

- **Initiated the GitHub Repository**
    - Created a basic structure for the project
    - Added fundamental code for client, server and executor

- **Integrating frontend with the backend**

  Integrated the backend with the frontend which is a crucial step in developing a successful software application. This process involves connecting the server-side logic of the application with the client-side interface, ensuring that the frontend can communicate with the backend seamlessly. The integration process involves defining APIs, connecting to databases, and ensuring that data flows securely between the two modules. By integrating the backend with the frontend, developers can create a robust and scalable software application that provides a seamless user experience.

- **Creating Routes**

  Created routes for the backend as it is an essential step in developing a web application.The routes in the project are the following:

    - Pkey.js - for generating the private key.
    - Auth.js - for the authentication of the users.
    - Password.js - for the encrypted passwords of the users stored in the manager.

- **JWT for token-based authentication**

  JWT (JSON Web Token) is a popular standard for token-based authentication. It is a compact and self-contained mechanism for securely transmitting information between parties as a JSON object. With JWT-based authentication, users can authenticate once and then receive a token that can be used for subsequent requests without the need for the user to re-authenticate every time. This makes JWTs a

convenient and efficient solution for token-based authentication in web applications.

3.3 Anushka Singh | 2021IMG014

- **React as the front-end framework**
  React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

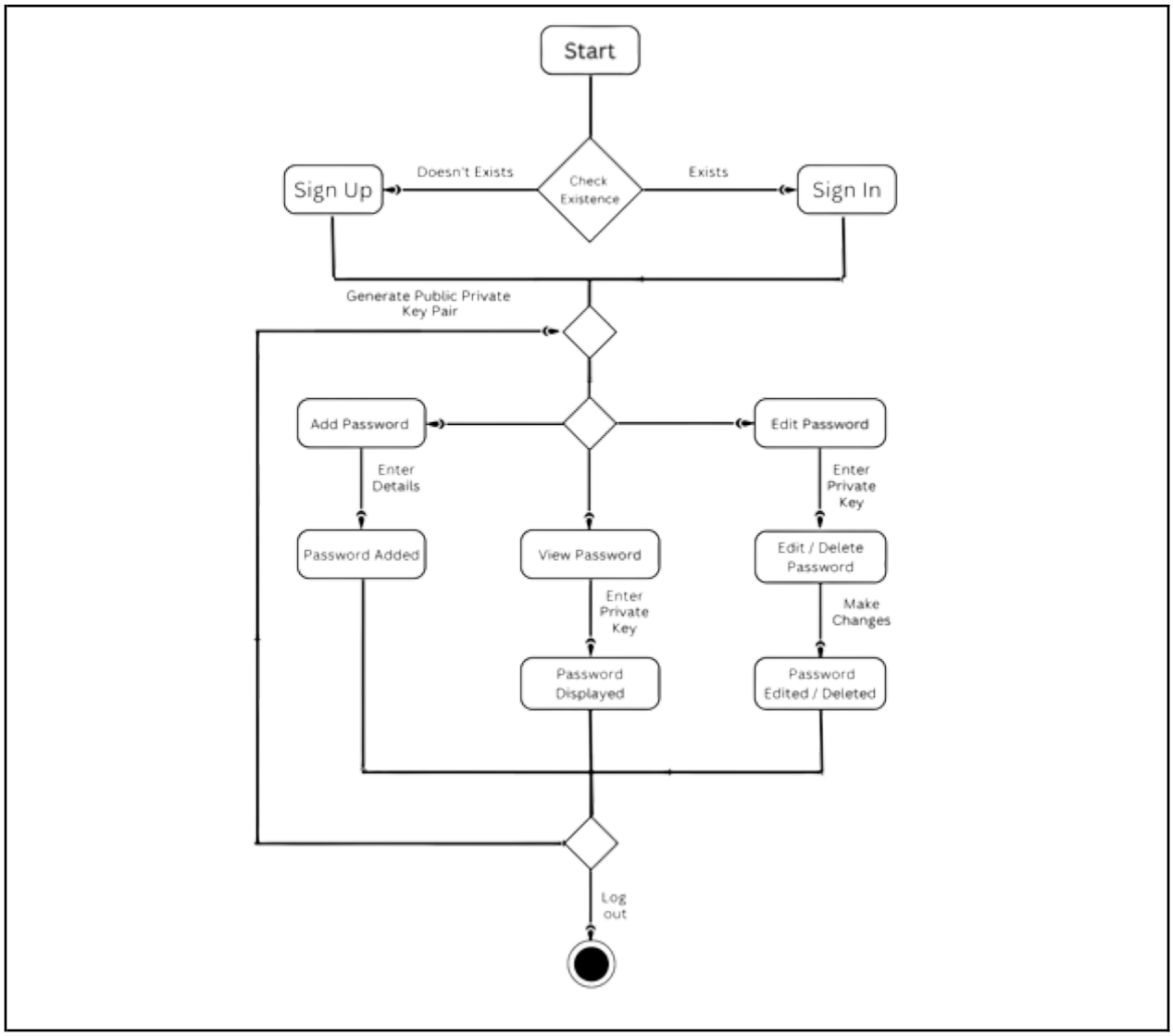
- **Material-UI for front-end design and styling**
  Material UI offers a wide variety of high quality components that have allowed us to ship features faster. MUI has been used by more than a hundred engineers in our organization. What's more, Material UI's well architected customization system has allowed us to differentiate ourselves in the marketplace.

- **Tailwind CSS and Bootstrap Styling**
  Tailwind CSS is basically a utility-first CSS framework for rapidly building custom user interfaces. It is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override.

  The beauty of this thing called tailwind is it doesn’t impose design specifications or how your site should look, you simply bring tiny components together to construct a user interface that is unique. What Tailwind simply does is take a ‘raw’ CSS file, process this CSS file over a configuration file, and produces an output.

4. Component Design and Overview

4.1 Activity diagram



4.2 Overview

In this final report, we will provide an overview of the project, including its scope, objectives, and key deliverables. We will also discuss the methodology used in the project, as well as its performance, limitations, and potential for future development.

4.2.1. Project Scope

The scope of the project was to create a secure and user-friendly password manager that would enable users to store, manage, and generate passwords. The project aimed to address the security concerns associated with traditional password management methods by using cryptographic algorithms to encrypt and protect user passwords.

The project also aimed to provide a user-friendly interface and features, such as two-factor authentication and automatic password generation, to make it easy and convenient for users to manage their passwords.

4.2.2. Objectives

The objectives of the project were to:

1. Create a secure password vault to store user passwords.
2. Implement cryptographic algorithms to encrypt and protect user passwords.
3. Provide two-factor authentication to enhance security.
4. Enable automatic password generation to simplify password management.
5. Create a user-friendly interface for easy password management.

4.2.3. Deliverables

The key deliverables of the project included:

- A fully functional cryptographically secured password manager with a secure password vault, two-factor authentication, and automatic password generation features.
- A user-friendly interface that allows users to easily manage their passwords.
- A detailed verification report to ensure the security and usability of the password manager.

4.2.4. Methodology

The project was developed using an iterative methodology, which involved continuous development, testing, and refinement of the password manager. The project team consisted of a project manager, a software developer, and a security expert.

The development process began with the creation of a detailed project plan, including timelines, milestones, and deliverables. The project plan also included a risk management plan to identify and address potential risks throughout the project.

The software developer worked on the design and development of the password manager, using a combination of programming languages, including Python and JavaScript. The security expert provided guidance on the implementation of cryptographic algorithms and security protocols.

The development process involved continuous testing and refinement of the password manager, including unit testing, integration testing, and system testing. The testing process was carried out using a range of testing tools, including automated testing tools and manual testing methods.

Performance:
The performance of the cryptographically secured password manager was evaluated based on its security, usability, and compatibility.

Security:
The password manager was evaluated for its security features, including encryption, two-factor authentication, and password strength. The system was also subjected to various security tests, including penetration testing, vulnerability scanning, and code review, to ensure that it was secure against potential attacks.

Usability:
The password manager was evaluated for its usability, including ease of use, user interface, and user experience. User feedback was collected throughout the development process, and the system was refined based on this feedback to ensure that it was user-friendly and easy to use.

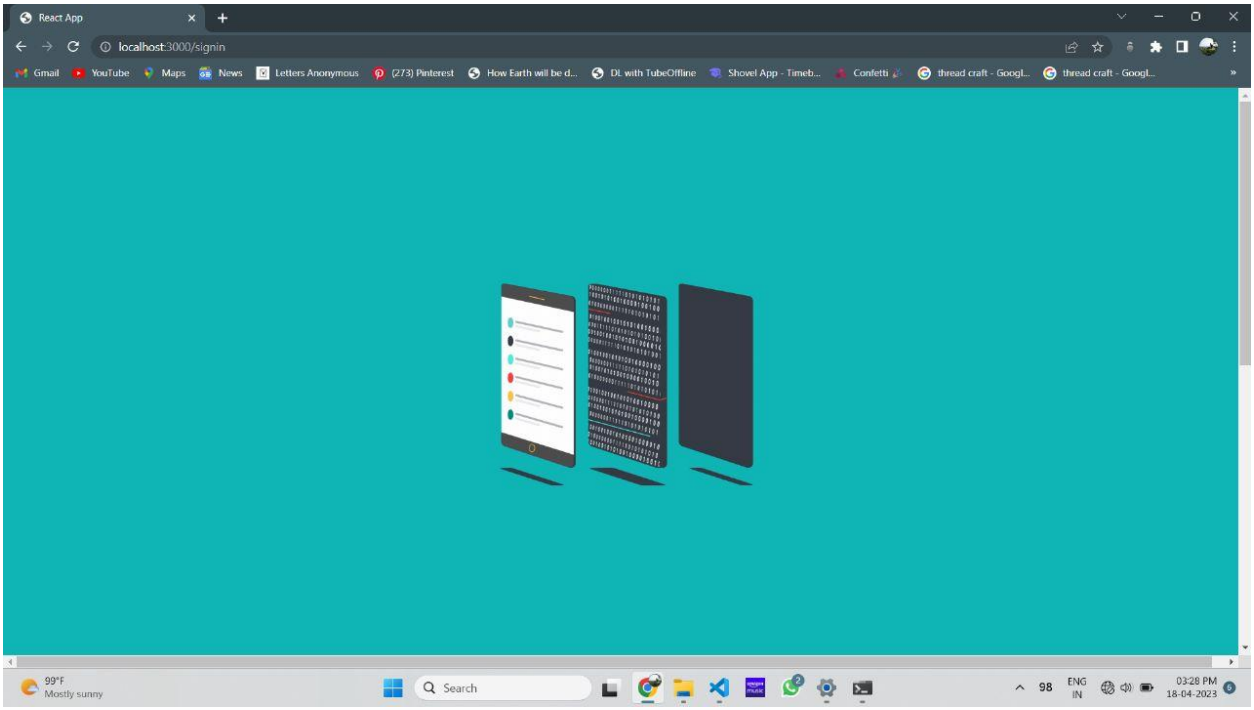
Compatibility:
The password manager was evaluated for its compatibility with different operating systems, web browsers, and hardware configurations. The system was tested on a range of platforms and devices to ensure that it worked correctly and reliably across different environments.

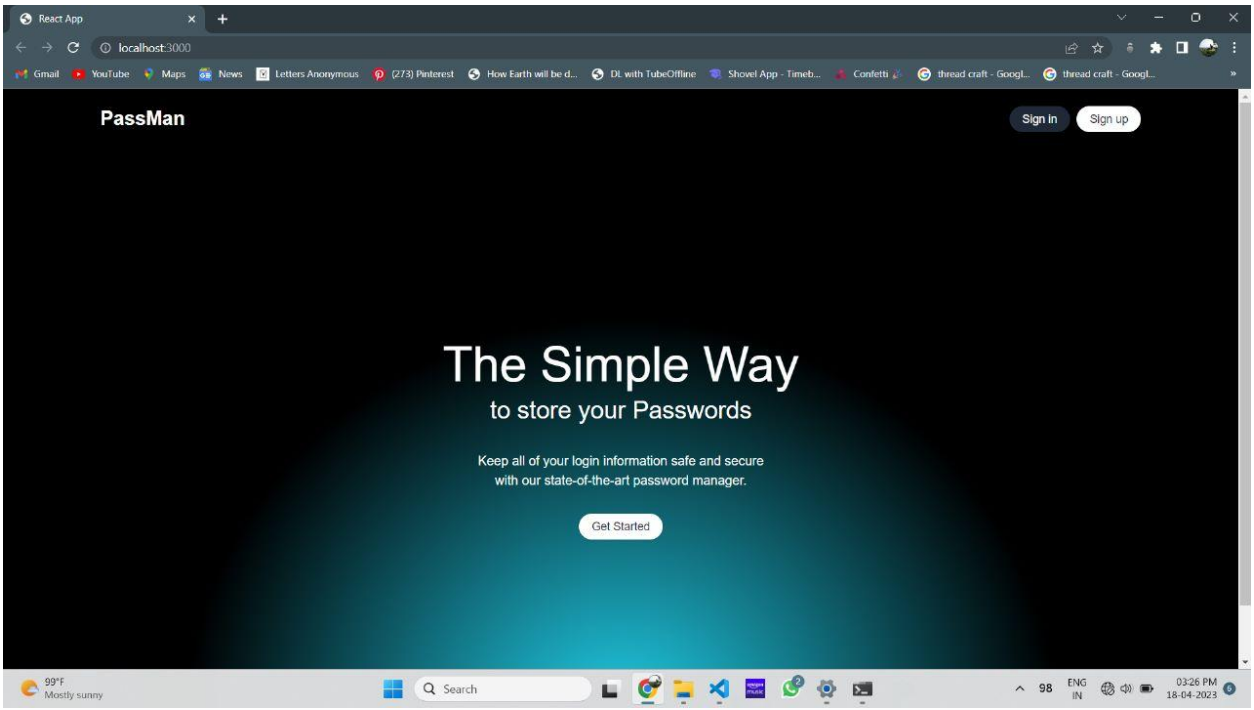
4.2.5 Limitations

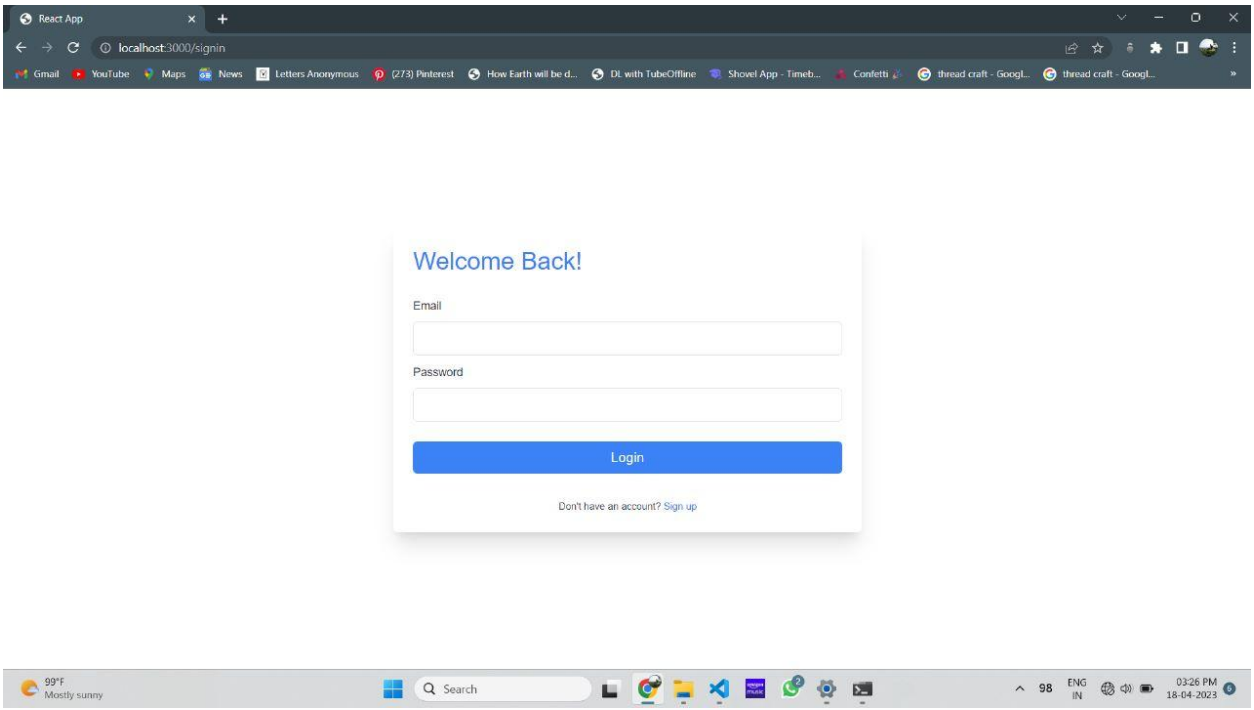
While the cryptographically secured password manager was developed to the highest standards of security and usability, there are some limitations to the system.

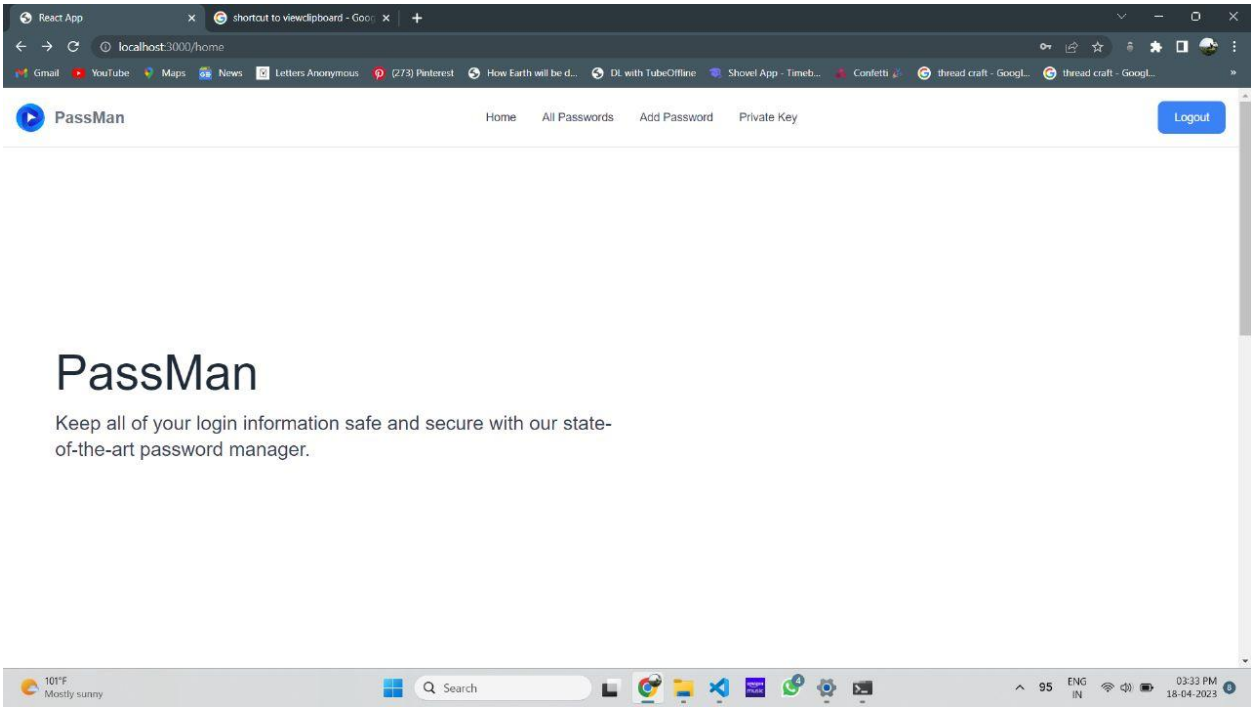
- Firstly, the system requires an internet connection to function, which may limit its usability in areas with poor connectivity.
- Secondly, while the two-factor authentication feature provides an added layer of security, it may be inconvenient for some

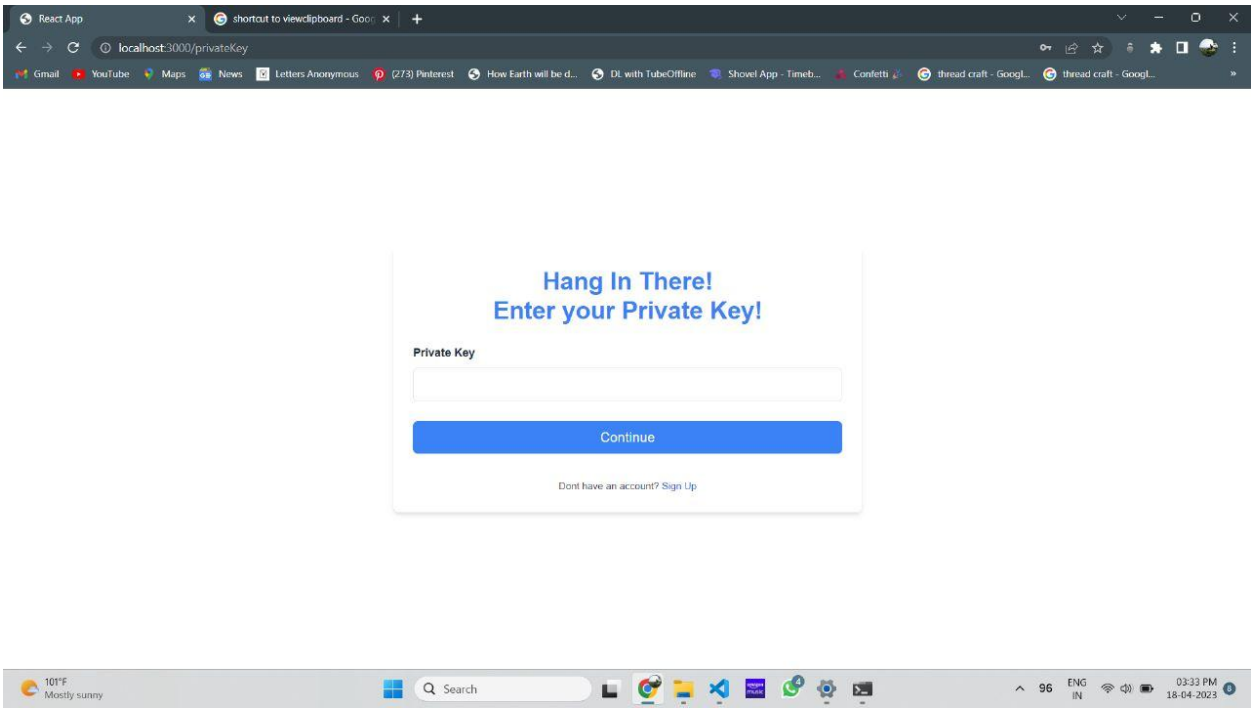
5. Screenshot of the final website application

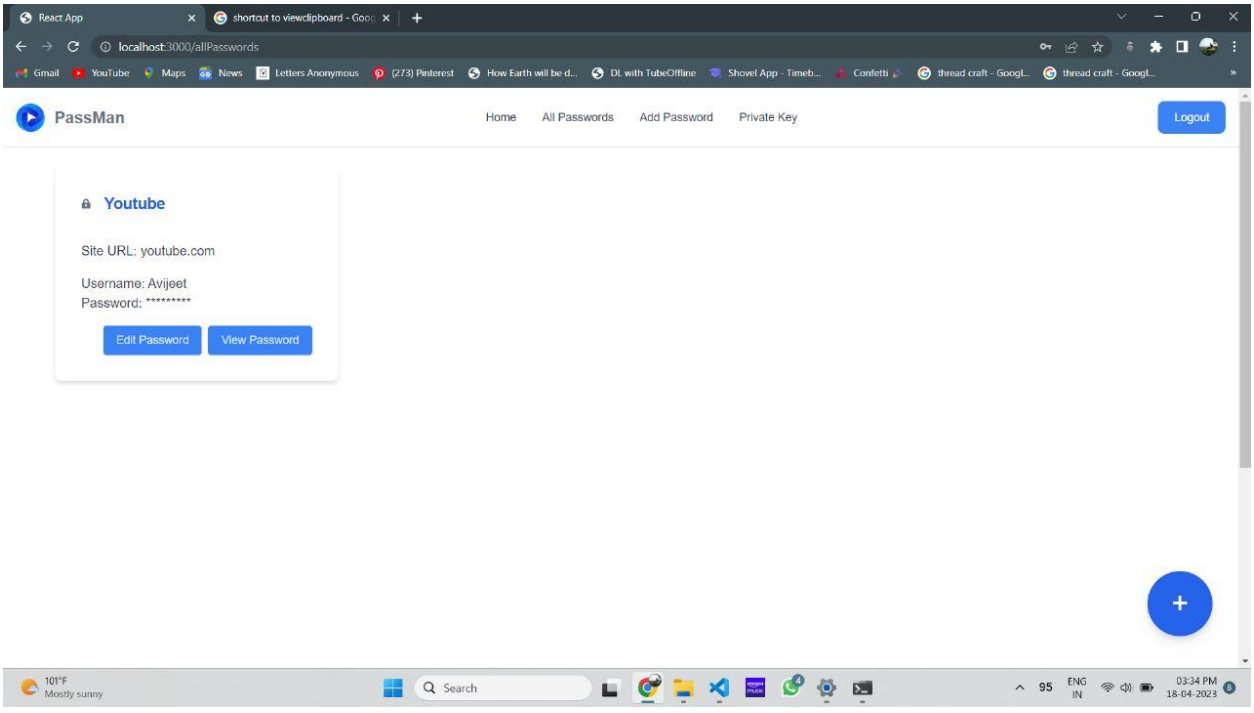


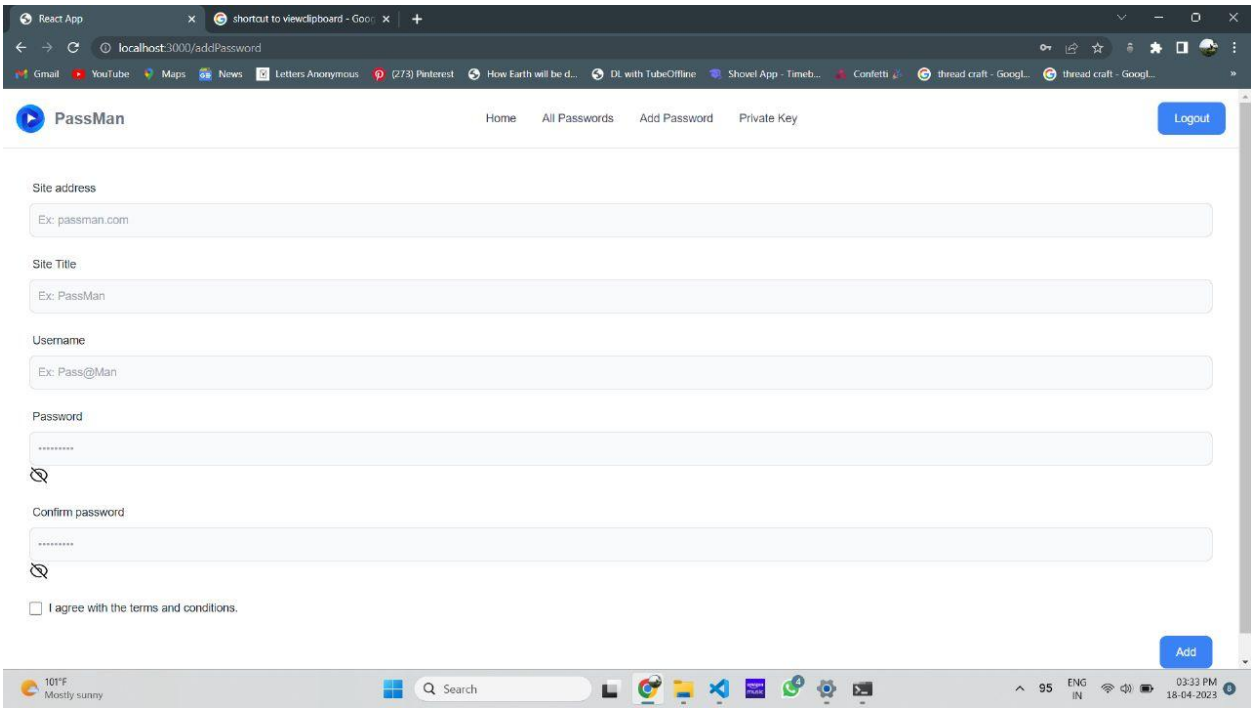


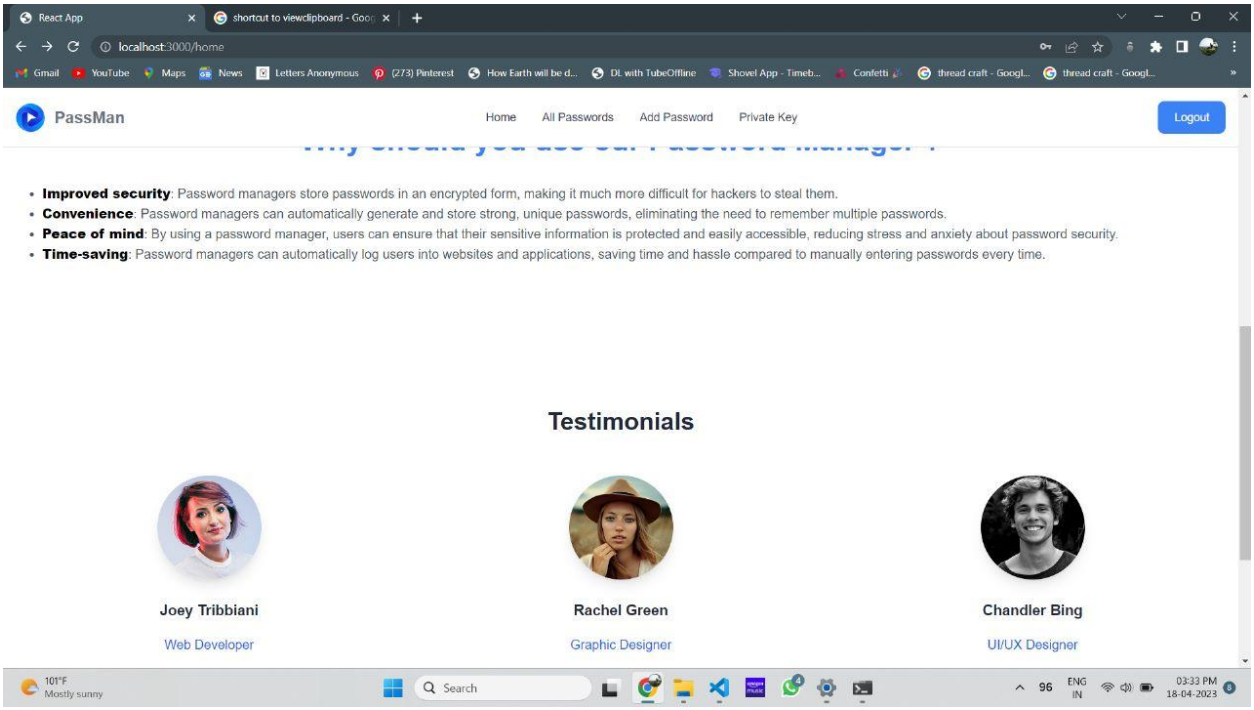

React App
localhost:3000/signin
Welcome Back!
Email
Password
Login
Don't have an account? Sign up


React App
shortcut to viewclipboard - Goo
localhost:3000/home
PassMan
Home
All Passwords
Add Password
Private Key
Logout
PassMan
Keep all of your login information safe and secure with our state-of-the-art password manager.


React App
shortcut to viewclipboard - Goo
localhost:3000/privateKey
Hang In There!
Enter your Private Key!
Private Key
Continue
Dont have an account? Sign Up


React App
shortcut to viewclipboard - Goo
localhost:3000/allPasswords
PassMan
Home
All Passwords
Add Password
Private Key
Logout
Youtube
Site URL: youtube.com
Username: Avijeet
Password: *********
Edit Password
View Password


React App
shortcut to viewclipboard - Goo
localhost:3000/addPassword
PassMan
Home
All Passwords
Add Password
Private Key
Logout
Site address
Ex: passman.com
Site Title
Ex: PassMan
Username
Ex: Pass@Man
Password
Confirm password
I agree with the terms and conditions.
Add
Search


React App
shortcut to viewclipboard - Goo
localhost:3000/home
PassMan
Home
All Passwords
Add Password
Private Key
Logout
Improved security: Password managers store passwords in an encrypted form, making it much more difficult for hackers to steal them.
Convenience: Password managers can automatically generate and store strong, unique passwords, eliminating the need to remember multiple passwords.
Peace of mind: By using a password manager, users can ensure that their sensitive information is protected and easily accessible, reducing stress and anxiety about password security.
Time-saving: Password managers can automatically log users into websites and applications, saving time and hassle compared to manually entering passwords every time.
Testimonials
Joey Tribbiani
Web Developer
Rachel Green
Graphic Designer
Chandler Bing
UI/UX Designer
Search

Conclusion

PassMan - A Cryptographically Secured Password Manager project is a secure and user-friendly password management system that provides a range of features for password storage, retrieval, sharing, and security. The project's main objective was to develop a password manager application that meets the needs of users who want to manage their passwords securely and efficiently.

Throughout the project, we conducted various analyses, such as feasibility, risk, and estimation, to ensure the project's viability and success. We also implemented a rigorous verification process to ensure that the project meets the quality standards and requirements of the client.

The client and server modules of the project work together to provide a secure and reliable backend and frontend system that offers password management functionalities to the end-user. The client module provides a user-friendly interface, while the server module ensures the security and reliability of the password data.

In conclusion, the Cryptographically Secured Password Manager project is a valuable contribution to the field of password management, offering a secure and user-friendly solution to managing passwords. We hope that this project will prove useful to users who seek a reliable and secure way to manage their passwords.