Softwarica College of IT & E-commerce | in collaboration with Coventry University

Report on Design and implementation of secure e-commerce web application

Submitted By:
Name: Bhawana Shahi
Student Id: 210332

Submitted To:
Arya Pokharel

Contents

Abstract

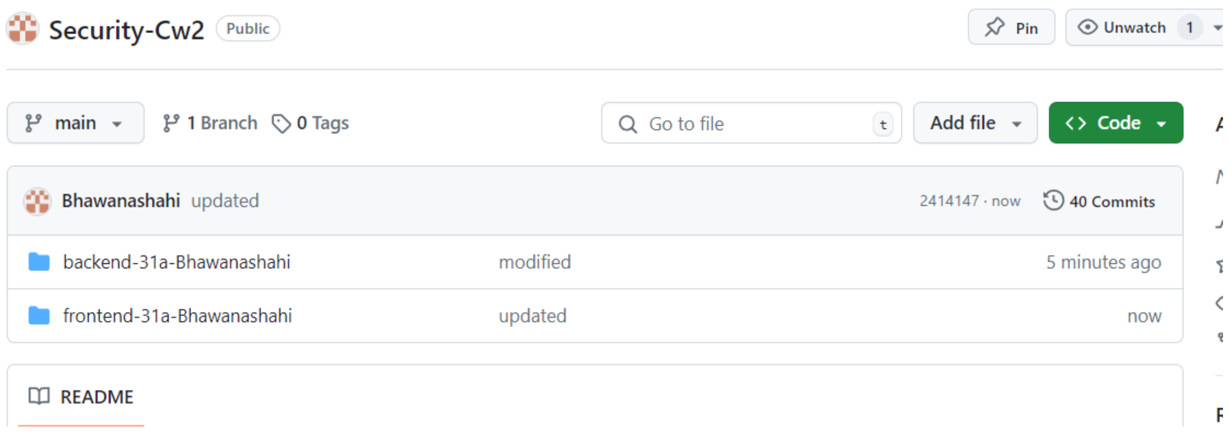
Security was a top priority during the development of EventEase. This choice of technology guarantees a strong and safe structure. Measures such as implementing password length and complexity requirements, as well as preventing password reuse, were put in place to bolster the security of user credentials. AES encryption provides robust protection for user data, while the user and admin dashboards are enhanced with additional security measures, allowing administrators to effectively monitor user logs and safely manage different sessions. The application utilizes input sanitization, MIME type filtering for picture uploads, and XSRF tokens to enhance its security measures against potential security risks.

Introduction

The objective of this study is to document the progression of creating a secure information technology system, specifically emphasizing software applications. This program places a high importance on safeguarding data and defending against any assaults, while also offering users a wide range of features. This part offers a concise overview of the software's characteristics and technical details, together with details about the security measures implemented throughout its development to ensure its safety. (anon, 2020)

Software Details

EventEase, an event management website for booking services including photography, beauty artists, and venues, uses React JSX, JavaScript XML, to build its UI. JSX empowers developers to construct a seamless and interactive user interface by integrating HTML-like syntax into JavaScript, enhancing the booking experience with dynamic and responsive features.



Bhawanashahi updated		2414147 · now · 40 Commits
backend-31a-Bhawanashahi	modified	5 minutes ago
frontend-31a-Bhawanashahi	updated	now

Figure 1: GitHub Repo

Framework

The MERN stack, a robust and secure set of web technologies, was employed in the process of software development. The Model-View-Controller (MVC) architectural model, which is adopted by the MERN stack, promotes code modularity and separation of concerns. JavaScript, a widely used and adaptable programming language, is employed for backend development through the utilization of Node.js and Express. It is managed by MongoDB and React for front-end components and databases, respectively.

.

Programming Language

A mix of programming languages well-known for their functions in web development and security is used to create the application.

ReactJsx

EventEase, an event management website for booking services including photography, beauty artists, and venues, uses React JSX, JavaScript XML, to build its UI. JSX empowers developers to create a seamless and interactive user interface by integrating HTML-like syntax into JavaScript, enhancing the booking experience with dynamic and responsive features.

Node Js

Node.js was employed to develop the infrastructure of EventEase, an event management platform that specializes in the reservation of services, including photography, makeup artists, and venues. This guarantees a user experience that is both responsive and seamless. (Javapoint, 2019)

Database Details

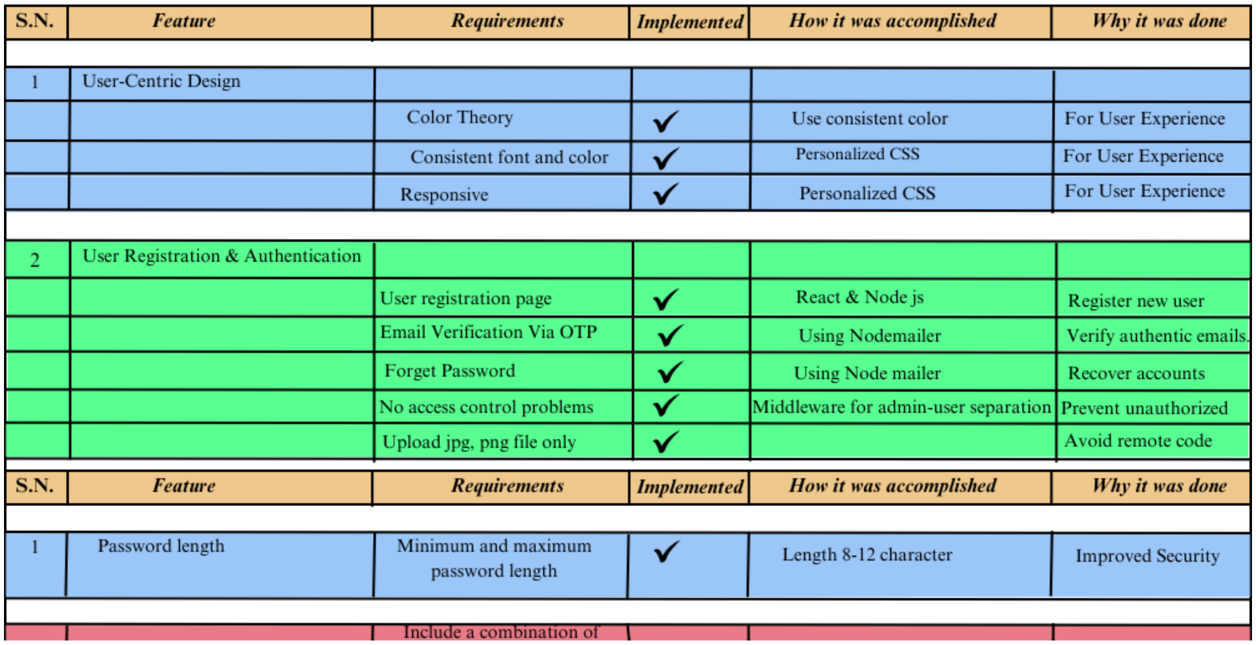
EventEase, an event management platform that facilitates the booking of services such as photography, makeup artists, and venues, employs MongoDB, a NoSQL database, to facilitate the administration of data. MongoDB's schema-less architecture enables the storing of data in a scalable and adaptive way, making it ideal for managing a wide range of data types that are diverse and constantly changing. Efficient data processing and robust data manipulation capabilities are ensured by the integration of this feature into the application's architecture.

Mongo DB

MongoDB is a freely available NoSQL database renowned for its ability to handle large amounts of data and its adaptability. The data is stored in a format that is oriented towards documents, which makes it well-suited for applications that include dynamic and unstructured data. In Eventease's MERN stack development, MongoDB functions as a dependable backend during both the development and production stages. It provides effective data management and the capacity to adapt to different project requirements, ensuring efficient handling and scalability.

The Checklist

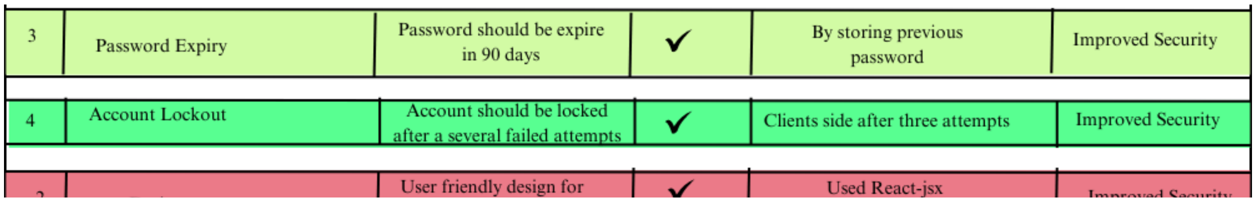
To ensure Eventease’s success, random security measures may cause misunderstanding about dependencies and their role. The checklist below outlines the security measures implemented to enhance Eventease's security.



S.N.	Feature	Requirements	Implemented	How it was accomplished	Why it was done
1	User-Centric Design				
		Color Theory	✓	Use consistent color	For User Experience
		Consistent font and color	✓	Personalized CSS	For User Experience
		Responsive	✓	Personalized CSS	For User Experience
2	User Registration & Authentication				
		User registration page	✓	React & Node js	Register new user
		Email Verification Via OTP	✓	Using Nodemailer	Verify authentic emails.
		Forget Password	✓	Using Node mailer	Recover accounts
		No access control problems	✓	Middleware for admin-user separation	Prevent unauthorized
		Upload jpg, png file only	✓		Avoid remote code

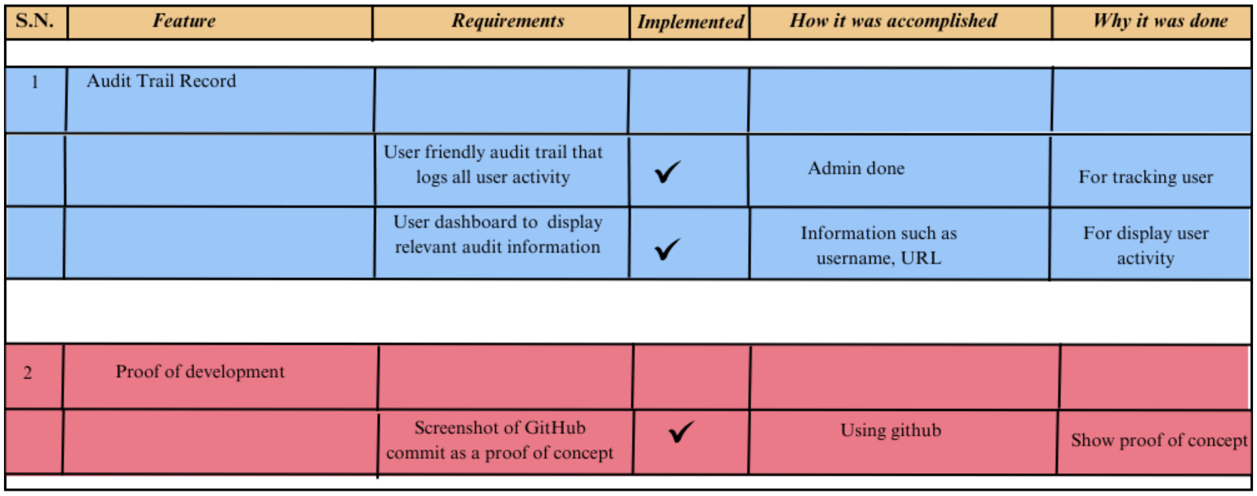
S.N.	Feature	Requirements	Implemented	How it was accomplished	Why it was done
1	Password length	Minimum and maximum password length	✓	Length 8-12 character	Improved Security
		Include a combination of			

Figure 2: Web app features



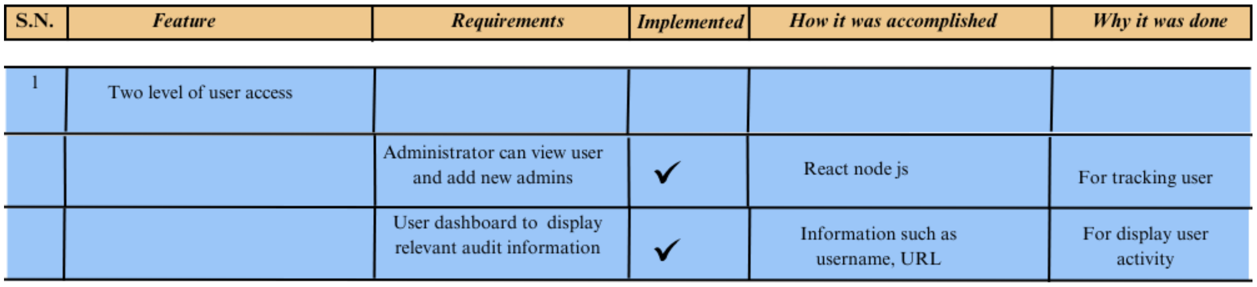
3	Password Expiry	Password should be expire in 90 days	✓	By storing previous password	Improved Security
4	Account Lockout	Account should be locked after a several failed attempts	✓	Clients side after three attempts	Improved Security
		User friendly design for	✓	Used React-jsx	Improved Security

Figure 3: Password Requirement



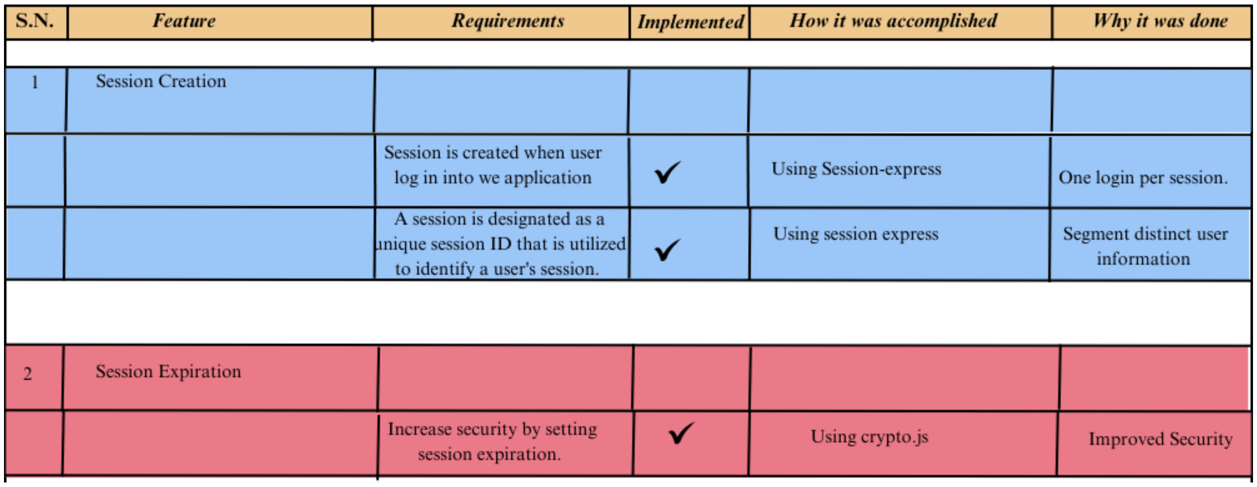
S.N.	Feature	Requirements	Implemented	How it was accomplished	Why it was done
1	Audit Trail Record				
		User friendly audit trail that logs all user activity	✓	Admin done	For tracking user
		User dashboard to display relevant audit information	✓	Information such as username, URL	For display user activity
2	Proof of development				
		Screenshot of GitHub commit as a proof of concept	✓	Using github	Show proof of concept

Figure 6: Audit Trail



S.N.	Feature	Requirements	Implemented	How it was accomplished	Why it was done
1	Two level of user access				
		Administrator can view user and add new admins	✓	React node js	For tracking user
		User dashboard to display relevant audit information	✓	Information such as username, URL	For display user activity

Figure 5: User Access level



S.N.	Feature	Requirements	Implemented	How it was accomplished	Why it was done
1	Session Creation				
		Session is created when user log in into we application	✓	Using Session-express	One login per session.
		A session is designated as a unique session ID that is utilized to identify a user's session.	✓	Using session express	Segment distinct user information
2	Session Expiration				
		Increase security by setting session expiration.	✓	Using crypto.js	Improved Security

Figure 4:Session Management

Features

Features 1: Password Requirements

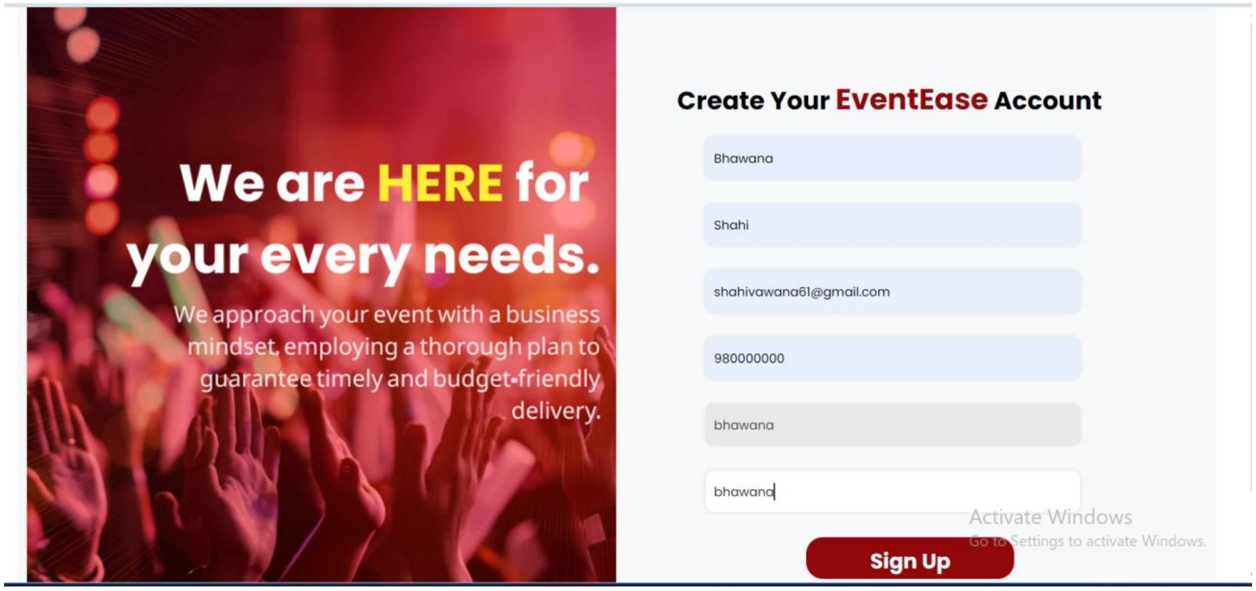
Password Length:

Background

One of the most important security measures to safeguard user accounts from potential password threats is to require a minimum password length. Frequently, these dangers entail a methodical attempt to guess passwords with fewer characters. The program seeks to strengthen its resistance against these kinds of attacks by mandating a password length of at least 8–12 characters.

Design

A password's permitted character count is specified within a specific range by the password length requirement. To enhance password security, users are informed that their password must have a length ranging from 8 to 12 characters when they establish or modify their password.


We are HERE for your every needs.
We approach your event with a business mindset, employing a thorough plan to guarantee timely and budget-friendly delivery.
Activate Windows
Go to Settings to activate Windows.
Sign Up

Create Your EventEase Account

Bhawana

Shahi

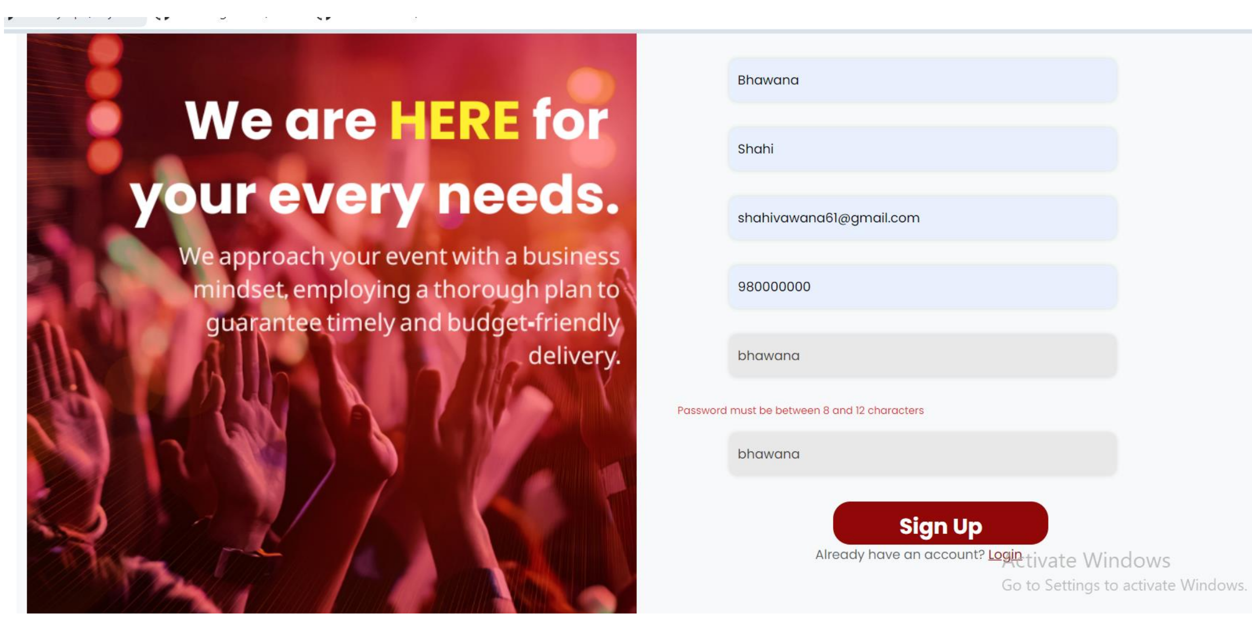
shahivawana61@gmail.com

980000000

bhawana

bhawana

Figure 7:Password less than 8 character


We are HERE for your every needs.
We approach your event with a business mindset, employing a thorough plan to guarantee timely and budget-friendly delivery.

Bhawana

Shahi

shahivawana61@gmail.com

980000000

bhawana

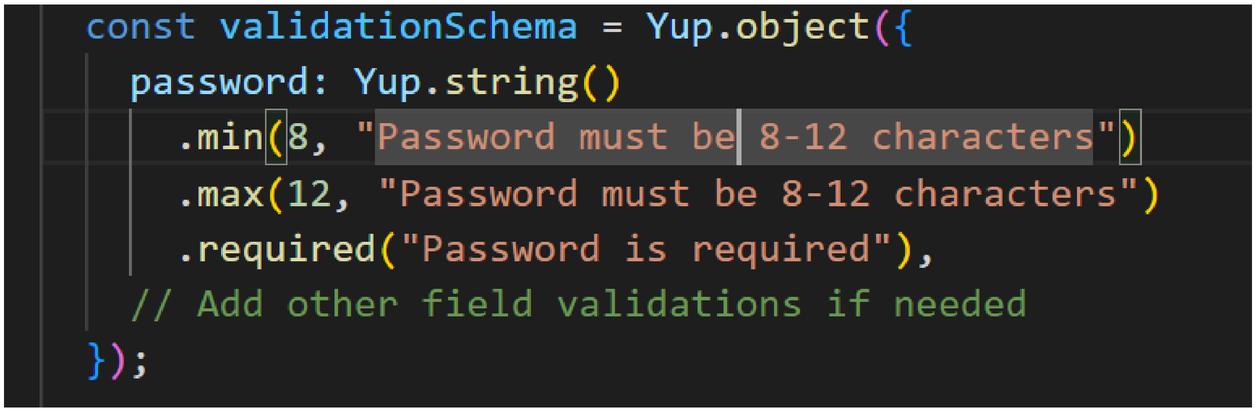
Password must be between 8 and 12 characters

bhawana

Sign Up
Already have an account? Login
Activate Windows
Go to Settings to activate Windows.

Figure 8: Password validation for fewer than 8 characters

Implementation



```
const validationSchema = Yup.object({
  password: Yup.string()
    .min(8, "Password must be 8-12 characters")
    .max(12, "Password must be 8-12 characters")
    .required("Password is required"),
  // Add other field validations if needed
});
```

Figure 9: code

In the implementation, the system checks that passwords entered by users are between the allotted 8 and 12 characters in length. The password is approved and continues to be processed if it satisfies this condition. A Node.js function in the application's backend code can be used to count the characters in the password. The system proceeds with the password creation procedure if the length of the password is within the permitted range.

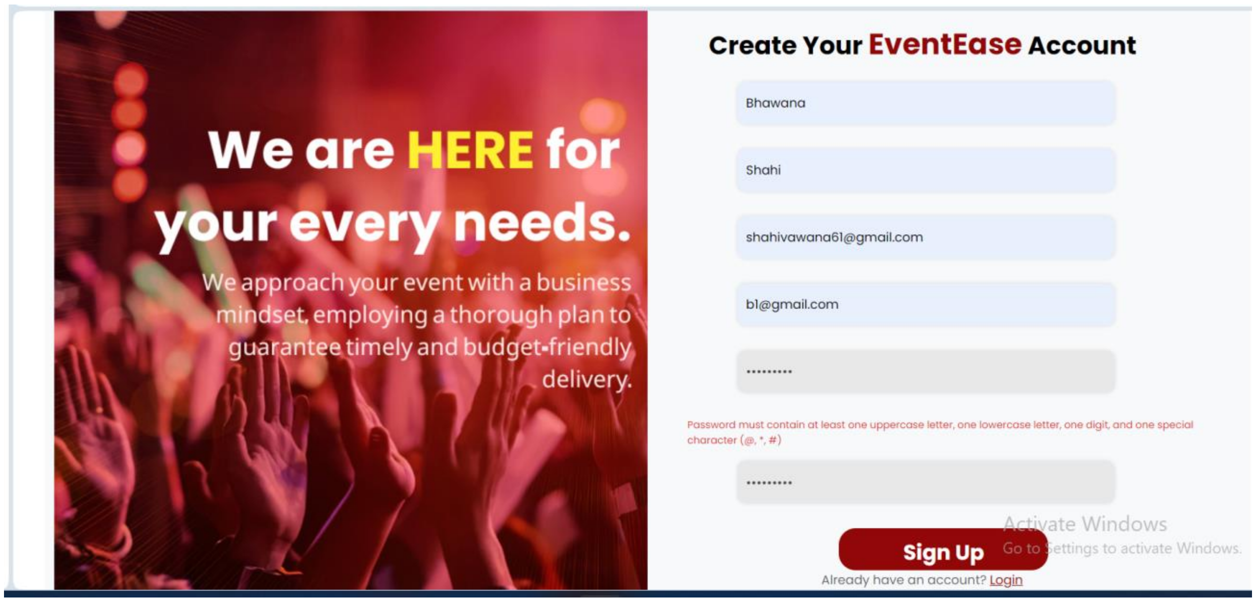
Password Complexity

Background

A key security feature that protects user accounts from numerous threats and unauthorized access is password complexity.

Design

Password difficulty in this application is configured to demand a combination of capital and lowercase letters, digits, and special characters like "*", "#", and "@". Users are given explicit guidelines when creating a password, stating that it must have a minimum of one capital letter, one lowercase letter, one number, and one special character.


We are HERE for your every needs.
We approach your event with a business mindset, employing a thorough plan to guarantee timely and budget-friendly delivery.
Sign Up
Activate Windows
Go to Settings to activate Windows.

Create Your EventEase Account

Bhawana

Shahi

shahivawana61@gmail.com

b1@gmail.com

•••••••••

Password must contain at least one uppercase letter, one lowercase letter, one digit, and one special character (@, *, #)

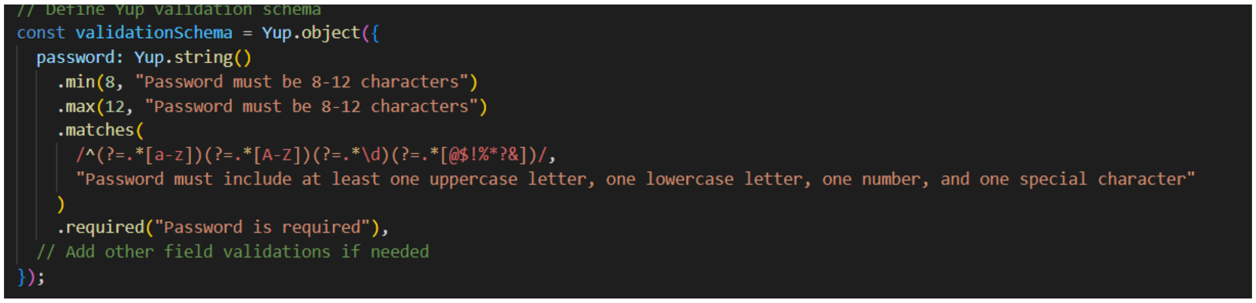
•••••••••

Already have an account? Login

Figure 10: For special character

Implementation

Strong validation logic should be incorporated into the backend programming that handles credentials in this application to ensure password complexity. The isupper(), islower(), and isdigit() functions in Node may be used by the backend of this Mern-based application to look over the password's constituent parts. Additionally, special characters can be detected using regular expressions.



```
// Define Yup validation schema
const validationSchema = Yup.object({
  password: Yup.string()
    .min(8, "Password must be 8-12 characters")
    .max(12, "Password must be 8-12 characters")
    .matches(
      /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])/,
      "Password must include at least one uppercase letter, one lowercase letter, one number, and one special character"
    )
    .required("Password is required"),
  // Add other field validations if needed
});
```

Figure 11: Code of Password complexity

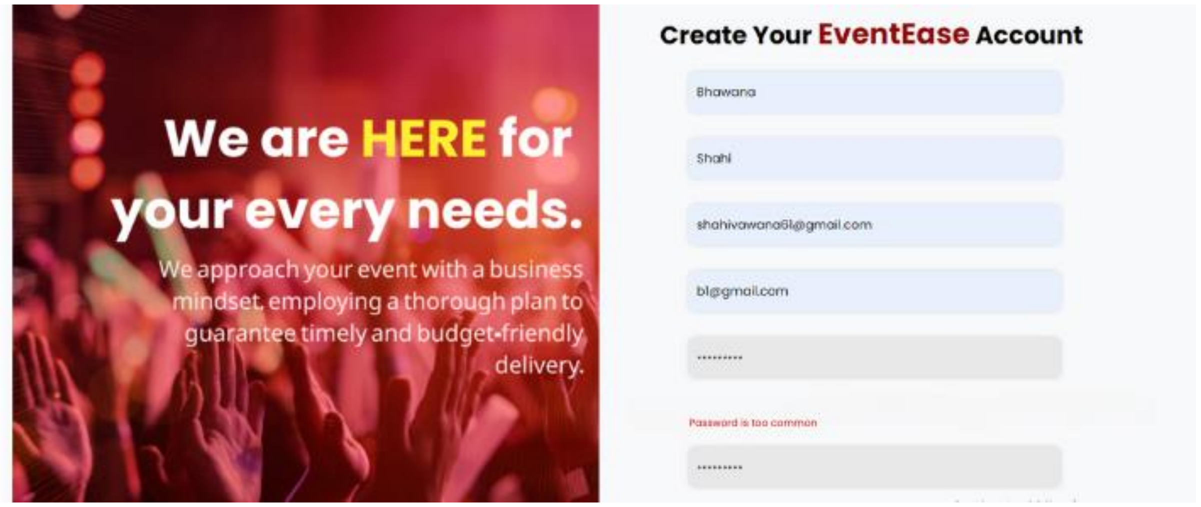
Avoid Common Passwords

Background

Preventing the usage of popular passwords is a crucial security precaution in this application. By preventing passwords like "password," "abc," and "123," which are simple to guess, the application strengthens its defense against potential weaknesses.

Design

The application checks to see if the user's selected password matches any entries on the restricted list when creating or updating a password. The user is prompted to choose a stronger password if a match is found.


We are HERE for your every needs.
We approach your event with a business mindset, employing a thorough plan to guarantee timely and budget-friendly delivery.

Create Your EventEase Account

Bhawana

Shahi

shahivawana61@gmail.com

bl@gmail.com

•••••••••

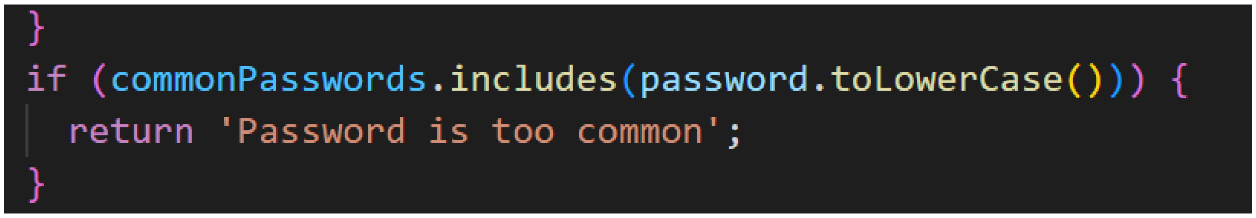
Password is too common

•••••••••

Figure 12: Validation for common passwords

Implementation

The password validator makes sure that users don't include personal information in their passwords by comparing the password to a list of keywords. It is used with the "CommonPasswordValidator" to stop people from using common passwords.



```
}
if (commonPasswords.includes(password.toLowerCase())) {
  return 'Password is too common';
}
```

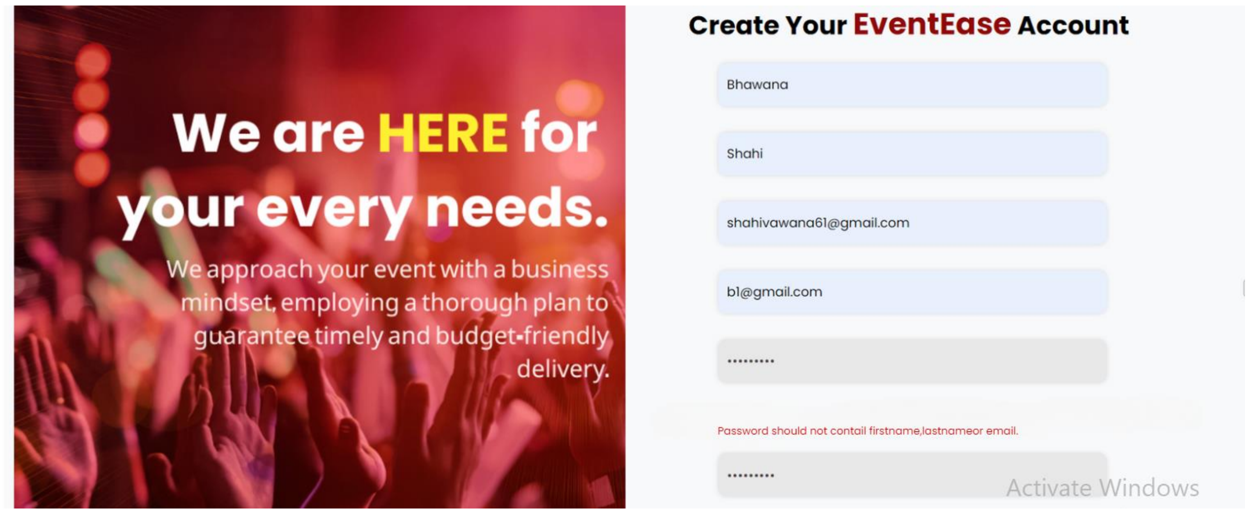
Figure 13: Code for common passwords

Avoid Personal Information

Background

Using readily available personal information such as addresses, birthdates, or names can seriously compromise the security of passwords. Through proactive disincentives, the application strengthens its protection against potential security lapses and unwanted access.

Design



We are HERE for your every needs.

We approach your event with a business mindset, employing a thorough plan to guarantee timely and budget-friendly delivery.

Create Your EventEase Account

Bhawana

Shahi

shahivawana61@gmail.com

b1@gmail.com

·········

Password should not contail firstname,lastnameor email.

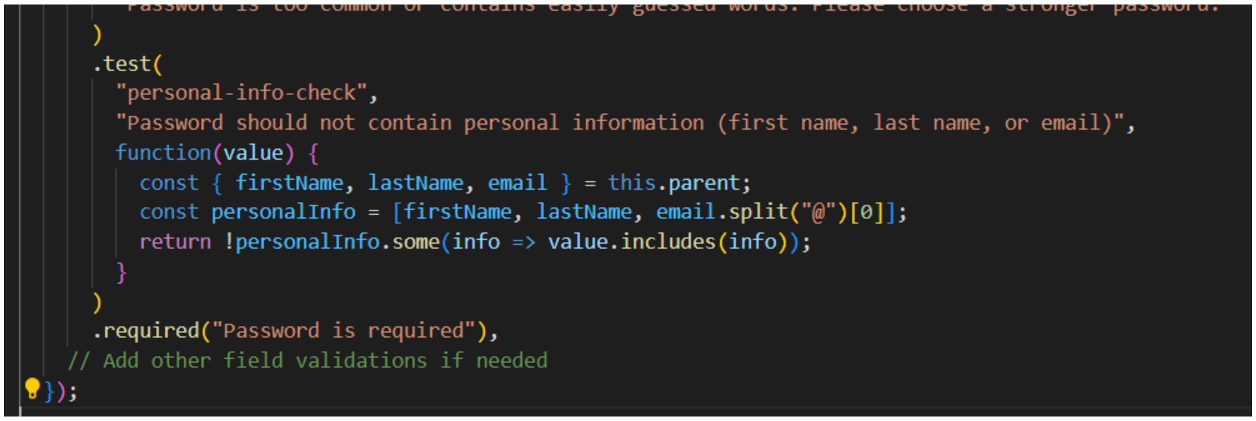
·········

Figure 14: Validation for personal information

The application thoroughly examines the suggested password for any direct matches or linkages to readily identifiable personal information while creating or updating a password. The application proactively suggests that the user select a different password if a match is discovered.

Implementation

The password that is entered cannot match any other value in any other field throughout the password choosing procedure. The user can select a password based on what the system requires.



```
        "Password is too common or contains easily guessed words. Please choose a stronger password."
      )
      .test(
        "personal-info-check",
        "Password should not contain personal information (first name, last name, or email)",
        function(value) {
          const { firstName, lastName, email } = this.parent;
          const personalInfo = [firstName, lastName, email.split("@")[0]];
          return !personalInfo.some(info => value.includes(info));
        }
      )
      .required("Password is required"),
    // Add other field validations if needed
});
```

Figure 15: Code for Personal information

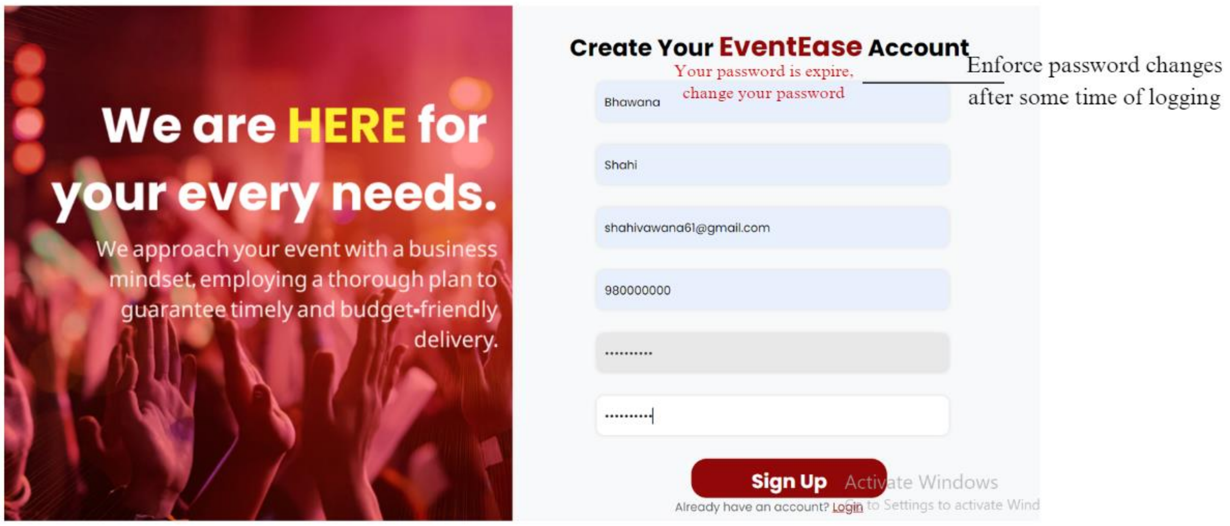
Enforce Password Changes

Background

By mandating regular password updates, the application aims to mitigate the risk of unauthorized access and stay ahead of potential security threats.

Design

The objective is to enhance security while minimizing disruption to users. This is accomplished by implementing a clear framework that prompts users to change their passwords at set intervals.



We are HERE for your every needs.

We approach your event with a business mindset, employing a thorough plan to guarantee timely and budget-friendly delivery.

Create Your EventEase Account

Your password is expire, change your password

Enforce password changes after some time of logging

Bhawana

Shahi

shahivawana61@gmail.com

980000000

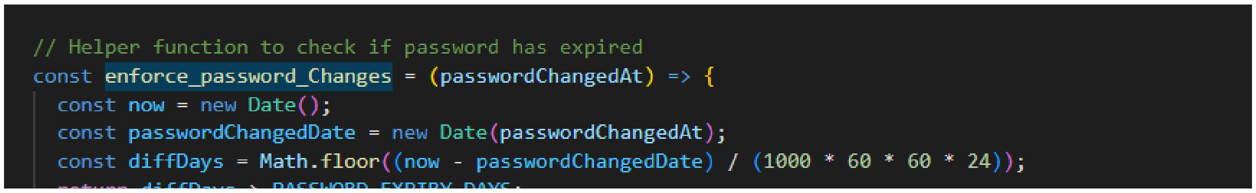
....

....

Figure 16: For enforce password changes

Implementation

The request and customer are passed as inputs to the enforce_password_change function in this implementation. It calculates how long it has been since the user last engaged, and if it is more than a certain amount of time—one minute, in this example—it displays a notice, reroutes the user to their profile page, and updates the last login timestamp.



```
// Helper function to check if password has expired
const enforce_password_Changes = (passwordChangedAt) => {
  const now = new Date();
  const passwordChangedDate = new Date(passwordChangedAt);
  const diffDays = Math.floor((now - passwordChangedDate) / (1000 * 60 * 60 * 24));
  return diffDays > PASSWORD_EXPIRY_DAYS;
```

Figure 17: Code for enforce password changes

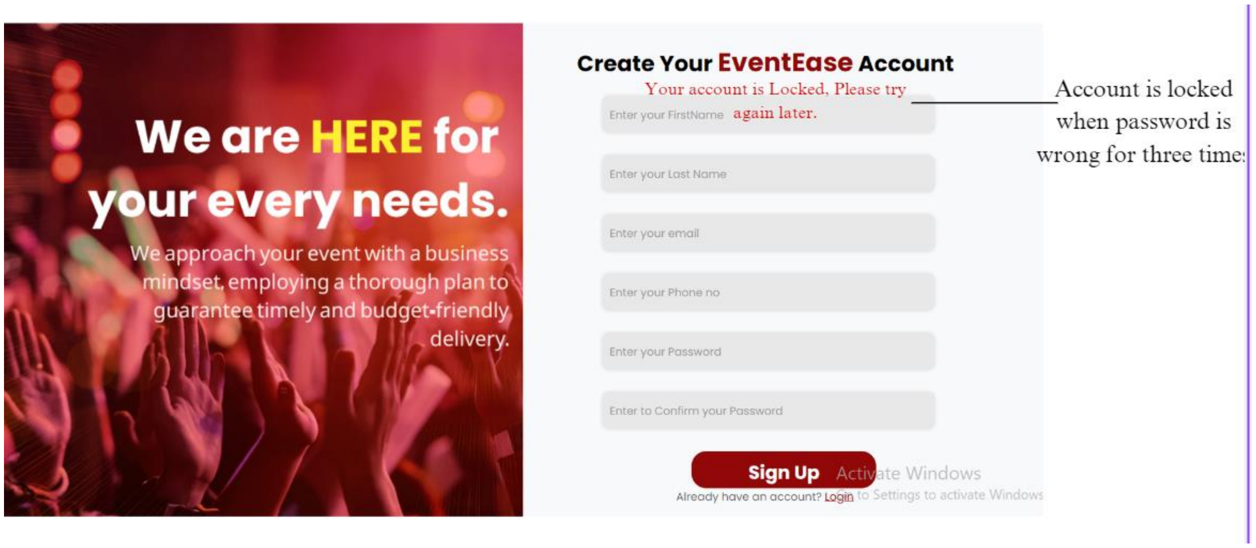
Limit Login Attempts

Background

The program can effectively lower the likelihood of hostile actors trying to guess user credentials by restricting the number of login attempts permitted within a specified time window.

Design

A maximum of three login attempts per user are permitted under the "Limit Login Attempts" function. A user's account gets temporarily locked and is inaccessible for a predetermined amount of time if they surpass this limit. By taking this precaution, you may protect user accounts from unwanted access and thwart brute force attacks.


We are HERE for your every needs.
We approach your event with a business mindset, employing a thorough plan to guarantee timely and budget-friendly delivery.
Sign Up
Activate Windows
Go to Settings to activate Windows

Create Your EventEase Account

Your account is Locked, Please try again later.

Enter your FirstName

Enter your Last Name

Enter your email

Enter your Phone no

Enter your Password

Enter to Confirm your Password

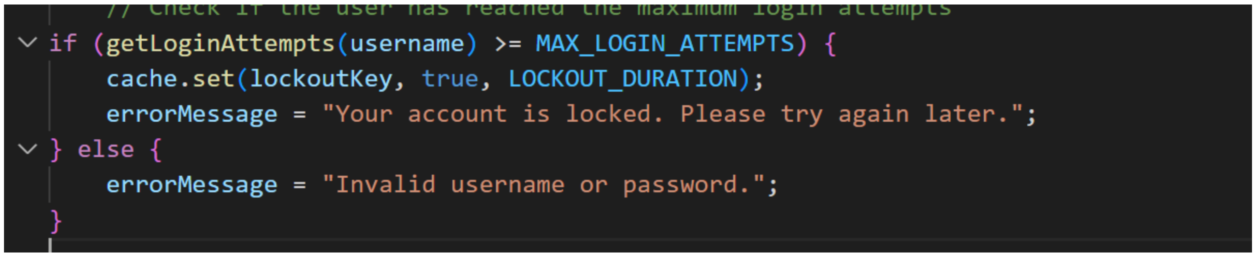
Already have an account? Login

Account is locked when password is wrong for three time

Figure 18: Validation for limit login attempts

Implementation

During each login attempt, the middleware tracks the number of failed logins attempts for the user's account within a specified time window.



```
    // check if the user has reached the maximum login attempts
if (getLoginAttempts(username) >= MAX_LOGIN_ATTEMPTS) {
    cache.set(lockoutKey, true, LOCKOUT_DURATION);
    errorMessage = "Your account is locked. Please try again later.";
} else {
    errorMessage = "Invalid username or password.";
}
```

Figure 19: Code for Limit login attempts

Encrypted Passwords

Background

Passwords stored in plain text are vulnerable to security breaches because they reveal private user data. Passwords ought to be salted and hashed before being entered into a database to increase security.

Design

By converting passwords into irreversible hash values, this design idea makes it extremely difficult for attackers to recover the original passwords.

Implementations

To add encrypted passwords to a MongoDB database, first open the MongoDB shell and make changes to the database configuration. By adding a new field to the user collection, you can store distinct random salts and improve security by guarding against different kinds of assaults.

Feature 2: Audit Logs

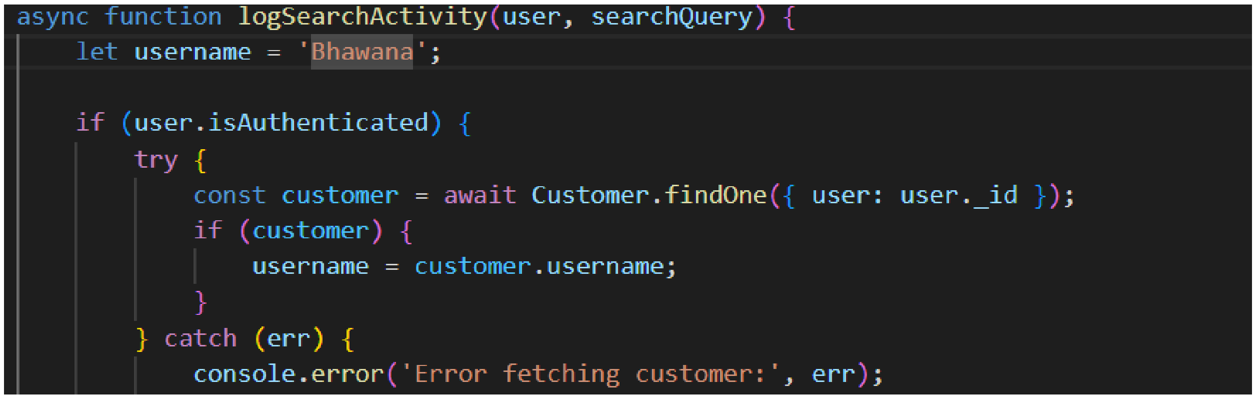
Background

Audit logs are essential for ensuring the security and accountability of a system or application. They offer a comprehensive record of all access attempts, user activities, and actions within the system. Implementing audit logs improves transparency, aids in monitoring, and helps identify security incidents.

Design

To gather and preserve relevant data regarding user interactions and system activities, an organized method was used in the design of the audit logs feature. Every log entry ought to contain information like this:

- The system account or user connected to the event.
- Actions like login or logout.
- An account of what happened.



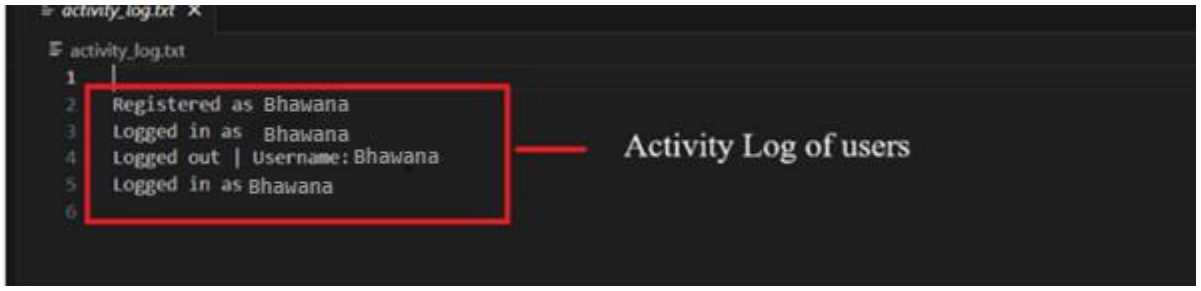
```
async function logSearchActivity(user, searchQuery) {
    let username = 'Bhawana';

    if (user.isAuthenticated) {
        try {
            const customer = await Customer.findOne({ user: user._id });
            if (customer) {
                username = customer.username;
            }
        } catch (err) {
            console.error('Error fetching customer:', err);
```

Figure 20: Code of Audits logs

Implementation

The system generates a log entry with the necessary information whenever a user completes an activity that needs to be recorded (such logging in, updating data, or changing settings) and adds it to the audit log file.



```
Registered as Bhawana
Logged in as  Bhawana
Logged out | Username:Bhawana
Logged in as Bhawana
```

Figure 21: User activity in application

Feature 3: User Access Level

Background

To effectively manage user interactions and permissions, the application incorporates three different user levels: guest user, registered user, and admin user.

Design

User access levels are designed in a way that allows users to be categorized based on their responsibilities and the actions they are allowed to take inside the online application.

Guest User

Guest Users can only examine products in the web application design to improve security and encourage user involvement. A Guest User will immediately be forwarded to the login page if they attempt to make a purchase or access the checkout procedure.

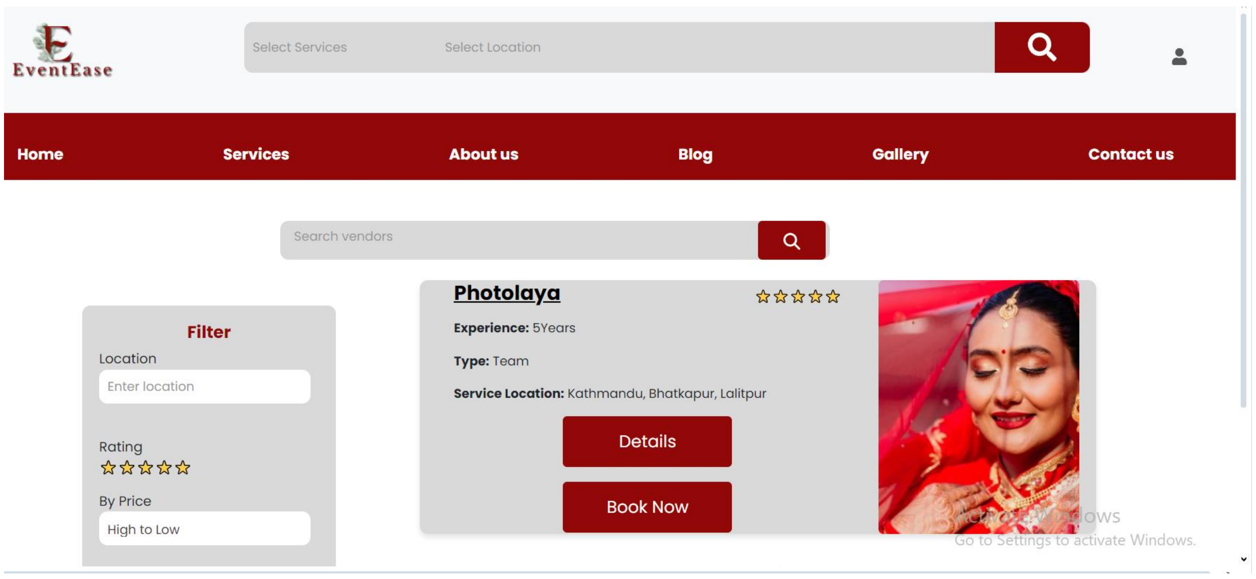


Figure 22: Guest User

Registered User

After logging in, registered users can use the following features of the web application:

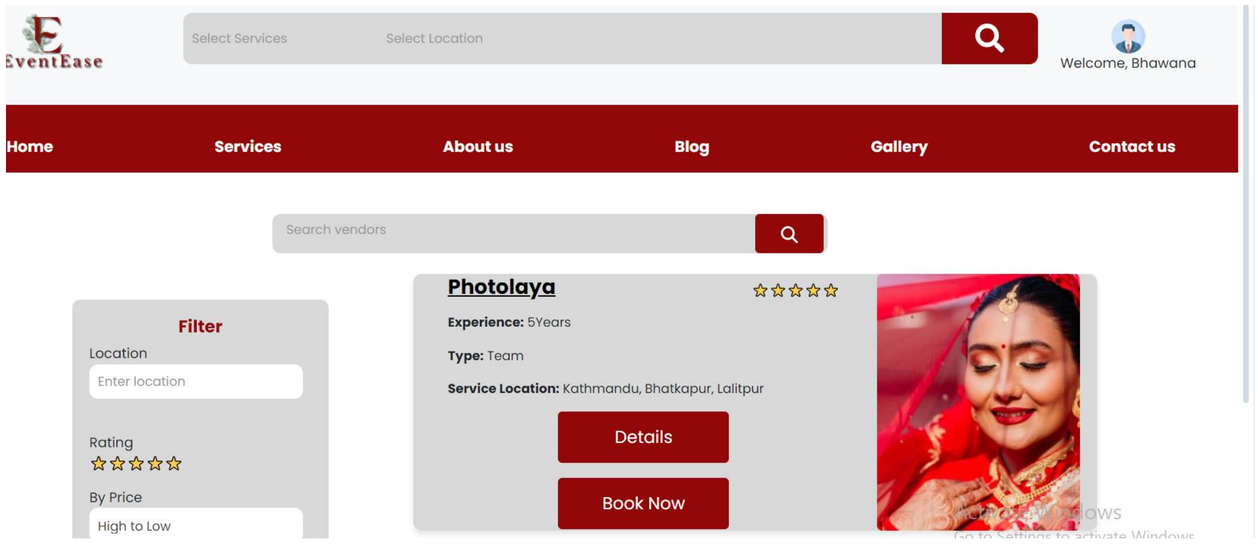
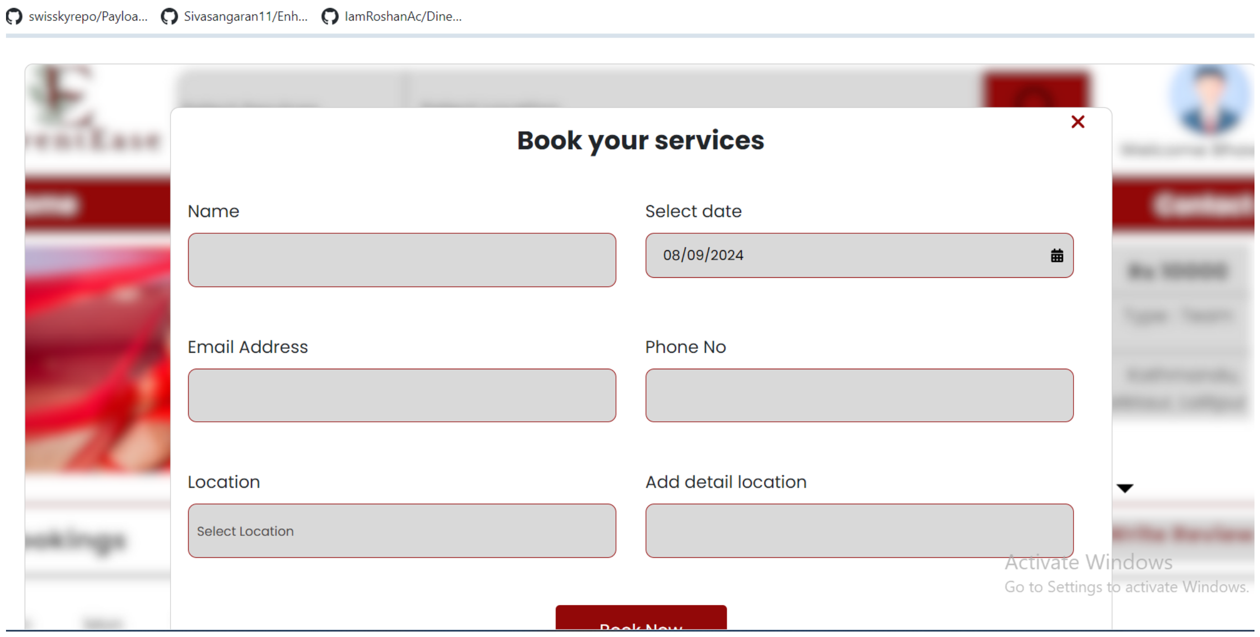


Figure 23: Registered User



Administrator

View Users

Within the online application, the Administrator has access to and viewing of a list of all registered users. They may monitor user activities, assign roles, and carry out account administration with this tool.

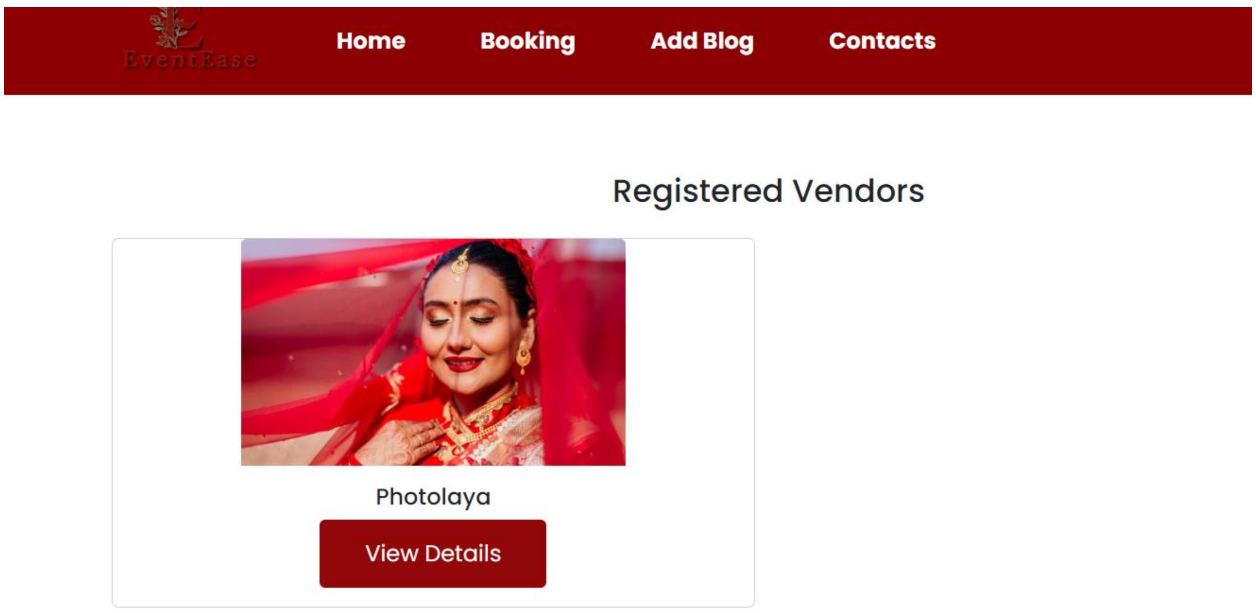
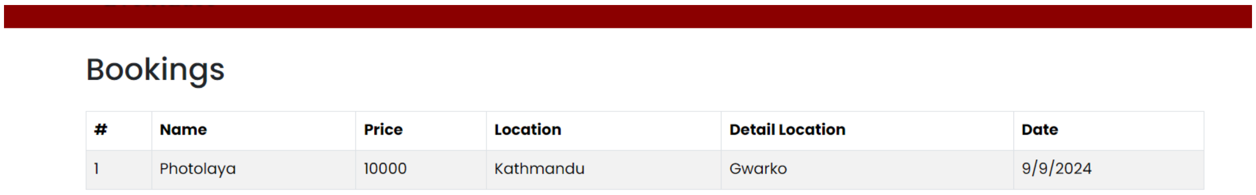
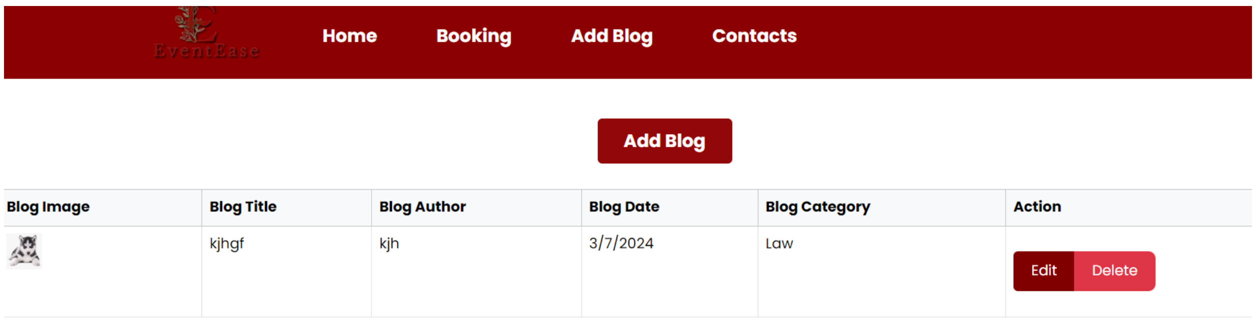


Figure 24: Admin can view Registered vendors


Bookings

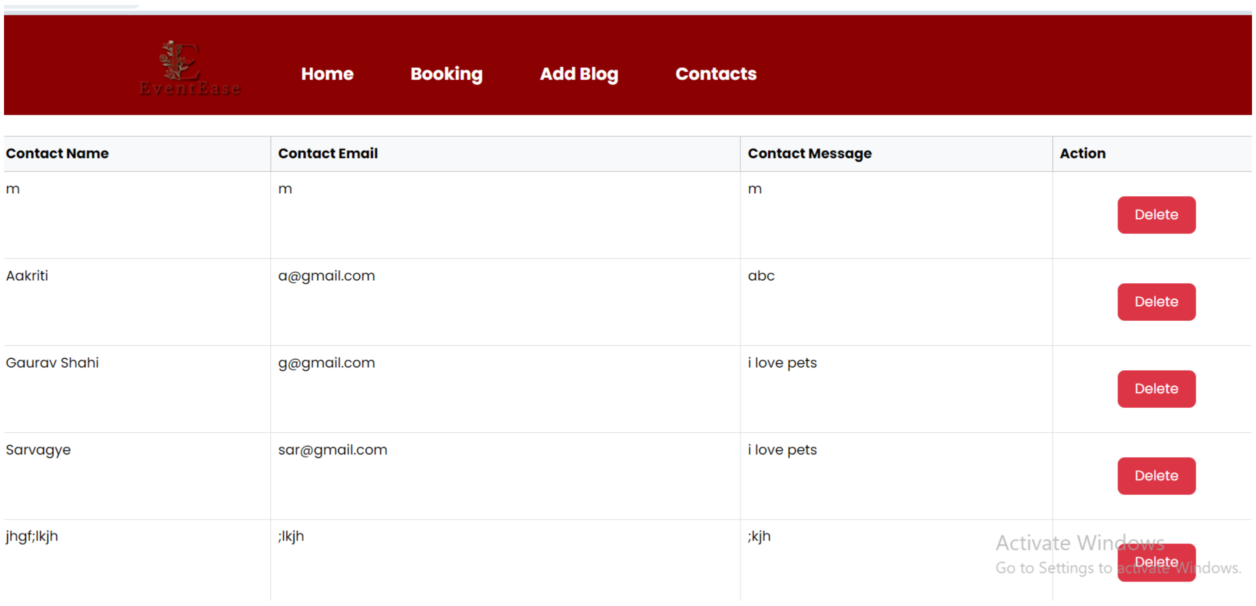
#	Name	Price	Location	Detail Location	Date
1	Photolaya	10000	Kathmandu	Gwarko	9/9/2024

Figure 25:Admin can view Bookings



Blog Image	Blog Title	Blog Author	Blog Date	Blog Category	Action
	kjhgf	kjh	3/7/2024	Law	Edit Delete

Figure 26:Admin can view blogs



Contact Name	Contact Email	Contact Message	Action
m	m	m	Delete
Aakriti	a@gmail.com	abc	Delete
Gaurav Shahi	g@gmail.com	i love pets	Delete
Sarvagye	sar@gmail.com	i love pets	Delete
jhgf;lkjh	;lkjh	;kjh	Delete

Implementation

For every access level, different user models or profiles can be created by the backend code. For instance, the Registered User model would have access to booking management and reviews, the Administrator model would have access to administrative services, and the Guest User model would just be able to see products. Before allowing a user to do an action, the application can confirm the user's role. The system will identify a Guest User's role and direct them to the login page if they attempt to make a booking.

Feature 4: Session Management

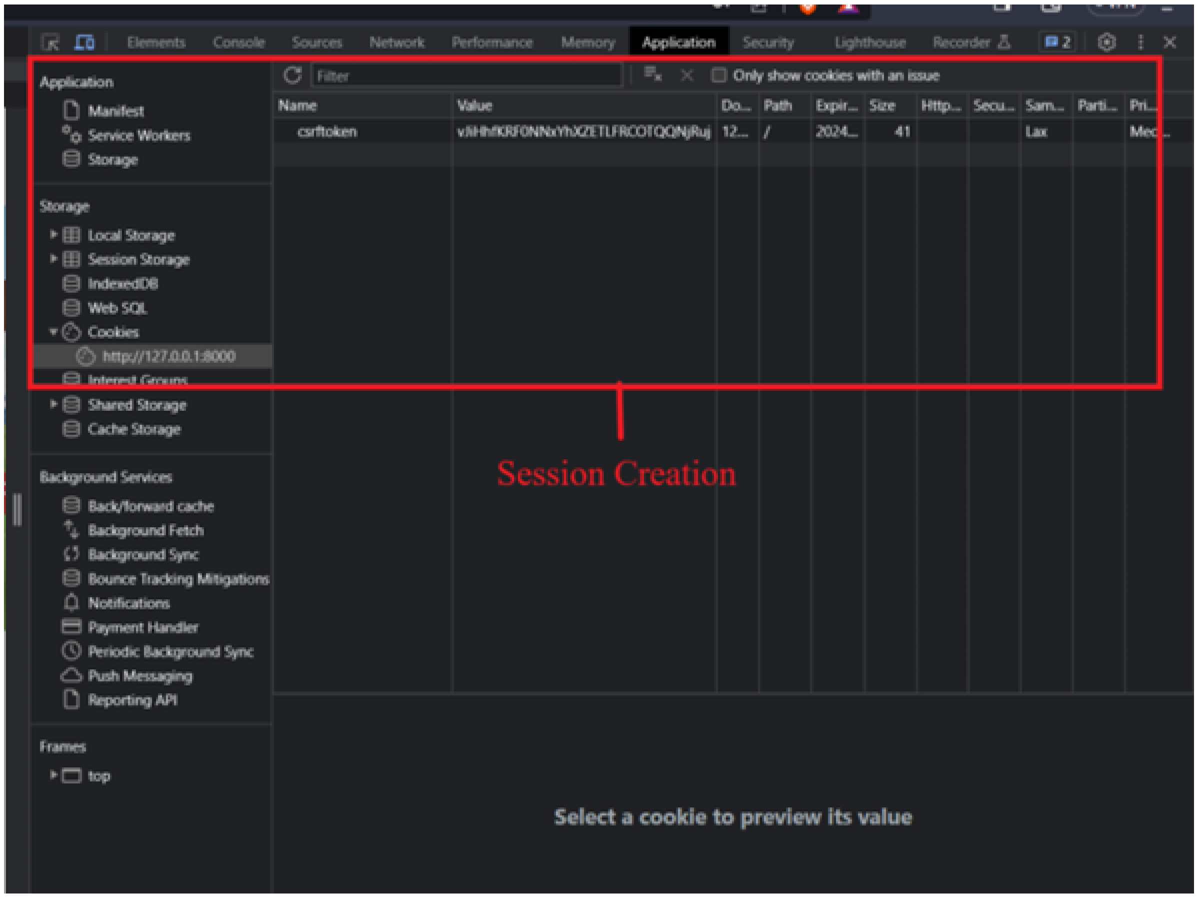
Session Creation

Background

It entails setting up and overseeing individual sessions for every user who communicates with the program. The creation of the session enables the system to monitor and control their visitation actions.

Design

Each user is given a unique session ID by the application after they successfully log in. To identify and differentiate between various user sessions, utilize this session ID.

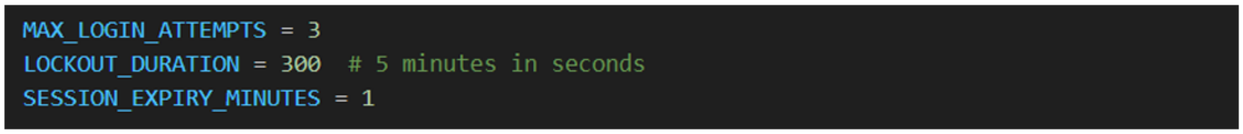


Name	Value	Do...	Path	Expir...	Size	Http...	Secu...	Sam...	Parti...	Pri...
csrftoken	vIiHhfKRFONNxYhXZETLFRCOTQQNjRuj	12...	/	2024...	41			Lax		Med...

Figure 27: Session Creation

Implementation

In this application, the process of session creation involves using the built-in session management tools of the MERN stack. When a user logs in, a session is created, and a session ID is assigned to them. This ID is stored in the user's browser as a secure cookie, ensuring that subsequent requests from the user include the session ID, allowing the server to identify and manage the user's session.



```
MAX_LOGIN_ATTEMPTS = 3
LOCKOUT_DURATION = 300  # 5 minutes in seconds
SESSION_EXPIRY_MINUTES = 1
```

Figure 28: Session Creation Implementation

Session Expiration

Background

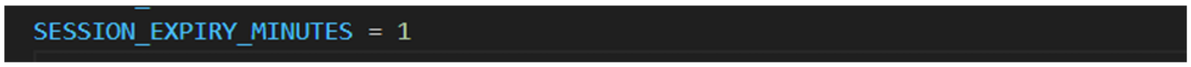
Session expiration is a crucial security feature that protects user data and stops unwanted access. It entails determining the maximum amount of time that a user's session can be kept active following a spell of inactivity. The user is automatically logged out of the application after the session ends and needs to log in again to use it.

Design

Session expiration is designed with user experience and security in mind. The program reduces the possibility of unauthorized access if a user leaves their device unattended by imposing a time limit on active sessions. This method makes sure that the user's session will terminate automatically after a predetermined amount of inactivity, even if they fail to log out.

Implementation

The session expiration time in this application can be set using the MERN stack's built-in session management capabilities. This can be done by specifying the time of inactivity that causes a session to expire in the session settings. The user is taken to the login page upon the expiration of a session, where they must reenter their credentials to access the program. By adding another layer of authentication, security is improved and unwanted access is avoided.



```
SESSION_EXPIRY_MINUTES = 1
```

Figure 29: Session expiration

Feature 5: Encrypted User Information

Background

A significant security risk arises when private user data, such as passwords and personal information, is stored in a database in plaintext.

Design

By utilizing the SHA-256 cryptographic hash function, SHA-256 hashing is intended to improve password security and ensure data integrity. This method uses several components that come together to give the application a strong layer of security.

Implementation

To implement encrypted passwords in a MongoDB database, start by modifying the database structure. Use the SHA-256 function to hash user information. This method ensures that hashed passwords are unique, even if users have the same password, thereby enhancing security.

Conclusion

To sum up, the project successfully specified and integrated several security elements for a web application used for event management system. The software uses a range of programming languages and frameworks to create a safe website, all while paying close attention to security during development. This effective development of a secure IT system shows how cybersecurity concepts may be successfully applied to practical uses. Together, the features that have been put into place provide a robust, user-centered, and secure management platform that tackles the constantly changing cybersecurity landscape. Together, the project's methodology, design decisions, and useful applications provide a complete solution that improves data security and user happiness.

References

Web security and Website Security | Fortinet. (n.d.). Fortinet. https://www.fortinet.com/resources/cyberglossary/what-is-web-security

MERN Stack - JavatPoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/mern-stack

GeeksforGeeks. (2023, November 13). *MERN Stack*. GeeksforGeeks. https://www.geeksforgeeks.org/mern-stack/

Update Password Encrypted by bcrypt MERN. (n.d.). Stack Overflow. https://stackoverflow.com/questions/75184130/update-password-encrypted-by-bcrypt-mern

D, V. (2023, May 1). How to hash Password string in a MERN app? - Vahid D - Medium. *Medium*. https://medium.com/@vahidnety/how-to-hash-password-string-in-a-mern-app-986ed03be8cf

Patil, C. (2024, March 24). Password encryption and decryption in Node.js using bcrypt package. *Medium*. https://medium.com/@patilchetan2110/password-encryption-and-decryption-in-node-js-using-bcrypt-package-5a7b1952d49d

Create databases and collections - MongoDB for VS code. (n.d.). https://www.mongodb.com/docs/mongodb-vscode/playground-databases/

GeeksforGeeks. (2024, June 27). *How to create database and collection in MongoDB*. GeeksforGeeks. https://www.geeksforgeeks.org/how-to-create-database-collection-in-mongodb/

5 tips to improve computer security when working from home. (n.d.). https://neklo.com/blog/common-computer-security-threats

NI Business Info. (n.d.). *Common cyber security measures | nibusinessinfo.co.uk*. https://www.nibusinessinfo.co.uk/content/common-cyber-security-measures

What is a Computer Security Service? AllSafe IT. (2024, March 20). https://www.allsafeit.com/blog/what-is-a-computer-security-service

What is web application security and how does it work? | Synopsys. (n.d.). https://www.synopsys.com/glossary/what-is-web-application-security.html#:~:text=Web%20application%20security%20defends%20against,implementation-level%20bugs%20are%20addressed.

W3Schools.com. (n.d.). https://www.w3schools.com/cybersecurity/cybersecurity_web_applications.php

CyberTalents. (n.d.). *What is Web Application Security? Everything you Need to Know*. CyberTalents Blog. https://cybertalents.com/blog/web-application-security

Blog | GirlsCanHack. (n.d.). GirlsCanHack. https://www.girlscanhack.com/blog?gad_source=1&gclid=Cj0KCQjwlIG2BhC4ARIsADBgpVR8k_Ec_7HGrZTY5JTaW8rXEhXON8W8PlxiMnBGPxcaEDAZ9Gc288QaAuNVEALw_wcB

Project Links

YouTube link: https://youtu.be/rmR8hiRr2fU
GitHub link: https://github.com/Bhawanashahi/Security-Cw2.git