Multi- Factor Authentication (MFA) Using MERN Stack

Sneha k[1], Subasree D[2], Yuvasri M[3], Sashitika R[4], Mrs. iyswarya k[5]
[1, 2] Dept of Information Technology
[1, 2, 3,4, 5] Sri Shakthi Institute of Engineering and Technology (Autonomous) Coimbatore-641062

Abstract- *Protecting stoner data and preventing unwanted access are crucial in today's digital world. Watchwords can be stolen or guessed, therefore traditional word- based authentication is no longer enough. This concept uses the MERN mound—MongoDB, Express.js, Reply, and Node.js—to integrate Multi-Factor Authentication (MFA) into an online application. By requiring drug users to provide several pieces of verification before allowing access, MFA improves security. This technique mixes the stoner's known (word) and unknown (OTP) commodities, which are typically sent via SMS, dispatch, or an authenticator app. OTP generation, verification, and stoner login are handled by the backend (Node.js with Express).*

I. INTRODUCTION

Protecting user data is more crucial than ever in the modern digital environment. Passwords can be readily guessed, stolen, or leaked, therefore using them alone is no longer secure. The requirement that users supply two or more types of verification—usually a combination of something they know (password), something they have (OTP or device), or something they are (biometric)—is how Multi-Factor Authentication (MFA) improves security. Unauthorized users find it far more difficult to obtain access thanks to this multi-layered strategy. Modern banking, enterprise, and web applications all make extensive use of MFA. In order to increase security and foster user trust, this project focuses on integrating MFA into a MERN stack (MongoDB, Express.js, React, and Node.js) application utilizing either email/SMS OTP or an authenticator app.

OBJECTIVE

1. Boost Security is put in place a second line of defense in addition to the conventional password-based authentication.
2. Prevent Unauthorized Access to make sure that sensitive information or services are only accessible by authorized users.
3. Create and incorporate MFA capability into a web application using the MERN stack (MongoDB, Express.js, React, and Node.js).
4. Allow OTP-based verification using email, SMS, or authenticator apps (like Google Authenticator) to support multiple verification methods.
5. Boost System Reliability and User Trust by implementing strong authentication, users can feel more secure about the application's security.

II. LITERATURE SURVEY

Secure authentication is one of the most important aspects of developing online applications in the current digital era. Using just usernames and passwords for single- factor authentication has shown to be inadequate in light of the growing number of identity thefts and data breaches. This problem is addressed by Multi-Factor Authentication (MFA), which requires several verification techniques before allowing users access. MFA usually combines the user's knowledge (password), possessions (phone or OTP), and occasionally identity (biometric verification). Any application's overall security is greatly strengthened by this tiered approach.

Static password authentication, which has been repeatedly demonstrated to be weak, was a major component of earlier systems. Modern applications have begun implementing MFA using email, SMS-based codes, and Time-based One-Time Passwords (TOTP) in order to reduce these vulnerabilities.

Static password authentication, which has been repeatedly demonstrated to be weak, was a major component of earlier systems. Modern applications have begun implementing MFA employing email verification techniques, SMS-based codes, and Time-based One- Time Passwords (TOTP) in order to reduce these dangers. Open-source libraries like otplib and speakeasy give developers the means to create time-bound, secure OTPs. These tools' simplicity and efficacy have led to their widespread use in MFA systems today.

Recent academic projects and industry implementations have successfully deployed MFA in MERN applications using a combination of JWT for session management, bcrypt for password hashing, and nodemailer for

sending OTPs. Firebase Authentication, Authy, and Google Authenticator have also been integrated with Node.js to enable TOTP-based second-factor authentication. Studies have shown that using MFA decreases the likelihood of account breaches by up to 99.9%, according to Microsoft Security Reports. Implementing MFA is not just a security upgrade but a compliance requirement for many sectors such as healthcare, banking, and e-commerce.

Recent research projects and industry implementations have successfully deployed MFA in MERN apps utilizing a combination of JWT for session management, bcrypt for password hashing, and nodemailer for issuing OTPs. To support TOTP-based second- factor authentication, Node.js has also been integrated with Firebase Authentication, Authy, and Google Authenticator. According to Microsoft Security Reports, research indicates that implementing MFA can reduce the risk of account breaches by as much as 99.9%.

III. METHODOLOGY

Initially, this project uses the MERN stack (MongoDB, Express.js, React, and Node.js) to create Multi-Factor Authentication (MFA) in a methodical manner. Users first register by entering their password and basic information, which is then safely hashed with bcrypt and saved in MongoDB. By confirming the username and password during login, the system carries out primary authentication. The system generates a One-Time Password (OTP) as the next step.

Second, this OTP can be generated as a time-based token using the speakeasy library for use with authenticator apps, or it can be provided to the user by SMS using Twilio or email using Node mailer. After that, the user is asked to enter the OTP, which the server verifies.

Thirdly, the system grants access to protected resources by issuing a secure session or JWT token if the OTP is accurate and hasn't expired.

Lastly, by including an additional layer of verification, this methodology guarantees improved security while preserving a seamless and intuitive user experience.

IV. EXISTING SYSTEM

A single-factor system, usually consisting of a username and password, is the only method used for user authentication in the majority of conventional web services. Despite being straightforward and popular, this approach has a number of serious security dangers, including phishing, brute-force attacks, password theft, and data breaches. Attackers can quickly access user accounts and private data if a password has been obtained. Basic safeguards like CAPTCHA and password strength checks are included in some applications, but they are insufficient to fend off complex attacks. Stronger, multi- layered authentication methods like Multi-Factor Authentication (MFA) must be implemented since applications are extremely vulnerable to identity theft and illegal access due to the current system's lack of a second verification layer.

Disdvantages

- Passwords that are easily hacked are susceptible to password leaks and brute- force assaults.
- Phishing susceptibility: Users may be duped into disclosing their login information.
- Absence of a second line of defense Complete access is given once a password has been compromised.
- Reusing passwords: People frequently use the same passwords on several sites, which raises the risk.
- Weak password practices: A lot of people select easy-to-guess passwords that are straightforward or widely used.
- Absence of user activity alerts: Users might not be aware of improper usage of their credentials.
- Absence of identity verification: Does not verify whether the account is being accessed by the legitimate user.
- Inadequate defense against social engineering: Unable to identify or stop attacks focused on people.

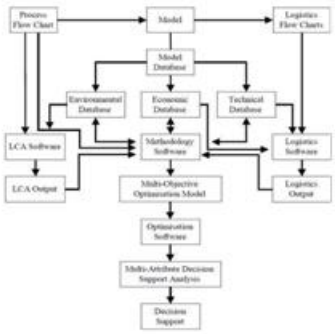
proposed system

The proposed system is a secure, full-stack web application that implements using the — MongoDB, Express.js, React.js, and Node.js. The main objective of this system is to enhance login security by requiring users to verify their identity through : a password and a One-Time Password (OTP). This proposed solution aims to overcome the vulnerabilities of traditional username- password systems and provide a robust method to protect user data from unauthorized access.

In this system, the , offering a responsive and user-friendly interface. It includes features like secure login, registration, and an OTP verification screen. The user first enters their registered email and password, which are verified using on the backend. If the credentials are valid, the system will generate a random OTP using packages like speakeasy or otp-generator, and send it to the user's email using . Only upon

successful OTP verification will the user be granted access to the system.

The , which manage routes for login, registration, password hashing, token generation, and OTP verification. Passwords are stored securely using , and are used for session management and route protection. This prevents unauthorized access to protected APIs or pages even if the frontend is manipulated. Additionally, all sensitive operations are secured using environment variables (dotenv) and best practices such as rate limiting, CORS handling, and error sanitization. is used to store user data, including email, hashed passwords, and OTP metadata. It supports fast data access, scalability, and flexible schemas that are ideal for modern applications. MongoDB also supports built-in security features like TLS encryption and role-based access control, enhancing the safety of stored user data.



Hardware requirements

- Processor: at least Intel Core i3
- RAM: 4 GB at minimum; 8 GB is advised for efficient development.
- Storage: Project files should have at least 100 MB available.
- Internet access is necessary to test OTP delivery and use email and SMS services

Software requirements

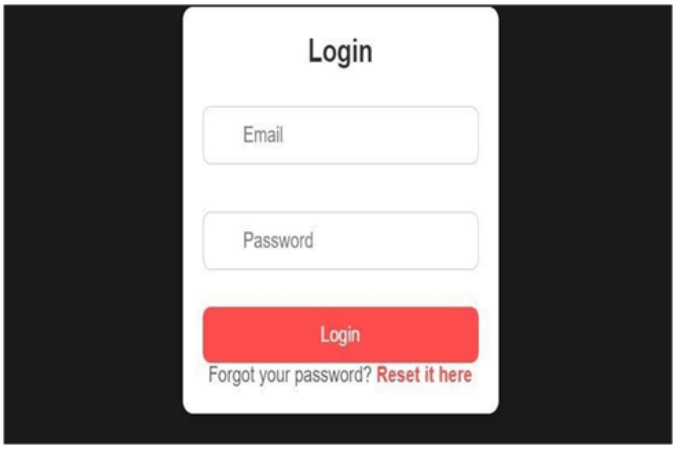
- MONGO DB
- EXPRESS.JS
- RREACT.JS
- NODE.JS
- SOCKET.IO

Module description

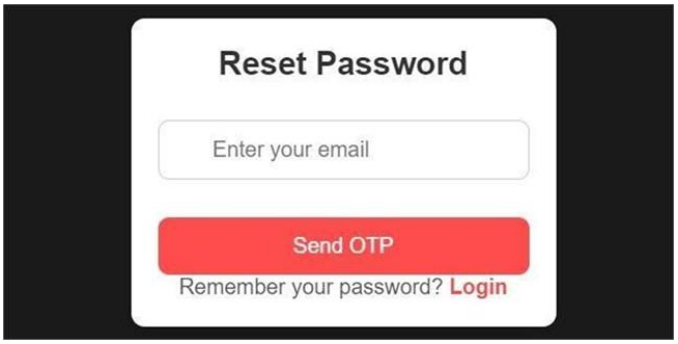
1. Fete the types of authentication factors and their beginning principles.
2. Dissect the advantages and disadvantages of colorful MFA ways.
3. Set up and apply MFA across colorful pall services and IT systems.
4. Examine MFA's donation to compliance and threat reduction.

Account page

LOGIN PAGE



RESET PASSWORD



V. CONCLUSION

By combining multiple authentication factors, multi-factor authentication (MFA) greatly lowers the risk of credential theft and security breaches. This module has given learners a thorough understanding of the concepts, configurations, and real-world applications of MFA. Armed with both theoretical knowledge and practical skills, students are now ready to implement strong authentication mechanisms that improve the overall security posture of contemporary IT systems. MFA is essential for bolstering access control and safeguarding digital resources from unauthorized access.

REFERENCES

[1] Russell, Steve (2023-02-2 . ITNOW. (1): 42–45.
[2] Jacomme, Charlie; Kremer, Steve (Februar . *ACMTransacoitnsonPrviacyandSecurtiy*. (2). New York City: Association for Computing Machinery: 1–34.
[3] kaitlin.boeckl@nist.gov (2016-06-28). NIST. Archived from on 2021-04-06. Retrieved 2021-04-06.
[4] Barrett, Brian (July 22, 2018). Retrieved 12 September 2020.
[5] Bruce Schneier (March 2005). .*Schneieron Security*. Retrieved 20 September 2016.
[6] Siadati, Hossein; Nguyen, Toan; Gupta, Payas; Jakobsson, Markus; Memon, Nasir\ *Compuetrs&Securyti*. : 14–28.
[7] Khandelwal, Swati . *The HackerNews*. Retrieved 2017-05-05.
[8] Hardcastle, Jessica Lyons (202 Retrieved 2023-08-24.
[9] Nichols, Shaun (10 July 2017). Retrieved 2017-07-11.
[10] Toorani, Mohsen; Beheshti, A. (2008). "SSMS - A secure SMS messaging 70 protocol for the m-payment systems