International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 04 | April - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Smart Transaction System using MERN Stack

[1]Dr. (Mrs) S.P. Washimkar, [2]Sakshi Chavhan, [3]Gaurav Nagose, [4]Ashish Ranjan, [5]Sanket Dorle

[1]Assistant Professor, [2]Student, [3]Student, [4]Student, [5]Student

[1]Department Of Electronics and Telecommunication Engineering,

[1]Priyadarshini College of Engineering, Nagpur, India.

Abstract- Efficient and secure transaction systems are critical to the success of enterprises in the quickly changing world of today. There is a growing need for new solutions since traditional methods of conducting transactions are becoming antiquated and laborious. The goal of this paper is to demonstrate a comprehensive and up-to-date solution for transaction management: The Smart Transaction System (STS), which was developed utilizing the MERN stack (MongoDB, Express.js, React.js, and Node.js). Simplifying the transaction process while maintaining security, dependability, and scalability is the goal of the STS. Both administrators and users can easily do a variety of transaction-related operations with its user-friendly interface. The solution leverages the capabilities of the MERN stack, using Node.js for backend development, Express.js for server-side logic, React.js for creating dynamic and responsive user interfaces, and a NoSQL database for flexible data storage.

Keywords: Smart Transaction System, MERN Stack, MongoDB, Express.js, React.js, Node.js, Security, Efficiency, Real-time Tracking

1.0. Introduction :

A smart transaction system based on the MERN stack (MongoDB, Express.js, React.js, Node.js) is an innovative way to eliminate the inefficiencies and vulnerabilities of traditional event systems. MongoDB serves as the underlying database, providing scalability and flexibility for different types of transactional data. Express.js facilitates the development of robust back-end APIs and ensures seamless communication between the front-end and back-end. React.js enables the creation of dynamic user interfaces that improve user experience and responsiveness. At the same time, Node.js provides a powerful setup runtime that can handle concurrent events and demand fluctuations. Through the integration of these key technologies, the Smart Transaction System aims to optimize the end-to-end transaction lifecycle, ensuring integrity, security, and auditability throughout the process. Central to the Smart Transaction System are its key features, including real-time transaction monitoring, secure authentication mechanisms, customizable transaction workflows, and comprehensive reporting capabilities. These features empower businesses to gain actionable insights into their transactional activities, streamline operational processes, and cultivate trust and transparency with stakeholders. By leveraging the capabilities of the MERN stack and implementing advanced functionalities, the system seeks to revolutionize how transactions are initiated, processed, and managed across industries. Through a detailed examination of system architecture, design principles, and implementation strategies, this thesis endeavours to showcase the effectiveness and viability of utilizing the MERN stack for building innovative transactional solutions, thereby contributing to the advancement of transaction systems and paving the way for future research and innovation in this domain. Key files include database schemas (MongoDB), API code-behind (Express.js), front-end components (React.js), and event-handling logic (Node.js).

2.0. Design consideration and Specification:

The Intelligent Transaction System can be developed to meet the changing needs of businesses while providing a secure, reliable and user-friendly platform for efficient transaction management. About the functionality of the current ATM system. An ESP32 microcontroller is used as the core of this embedded system, which is related to fingerprint recognition technology. GSM feedback mechanism and current high-speed network connection.

The primary features of the developed system are:

Fingerprint recognition: Integrating fingerprint sensors into a smart transaction system improves security and user authentication. By implementing biometric authentication, users can securely authorize transactions with their unique fingerprints, providing a convenient and reliable method of identity authentication and preventing unauthorized access. This feature adds additional security and convenience to the transaction process, simplifying the user experience and ensuring strong authentication measures.

Mongo DB: In our intelligent transaction system, which uses the MERN stack, MongoDB functions as a NoSQL database that offers flexibility in processing transaction data. With a scalable architecture and robust query capability, MongoDB efficiently stores and retrieves transaction records, ensuring smooth operation and improving overall system performance.

GSM module: Integrate the GSM module into the Smart Transaction System built with the MERN stack to enable SMS notifications of transaction updates. This allows users to receive real-time alerts on their mobile devices, improving communications and providing additional convenience and usability of system functions.

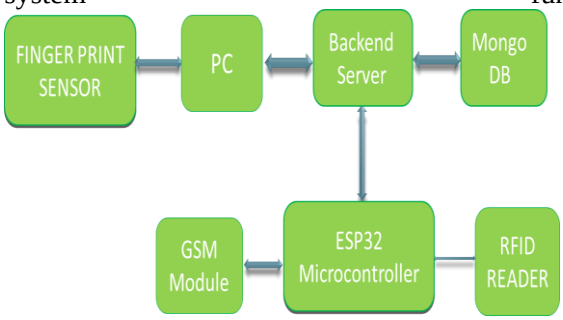

Fig. 1.0. Block diagram of the designed system

3.0. Design Analysis of different Sections of the System

Design Analysis of Different Sections of the Smart Transaction System Using MERN Stack **3.1.ESP 32 Microcontroller:** Integrating the ESP32 microcontroller into the MERN stack enhances the intelligent event system by enabling real-time communication with physical devices and sensors. The ESP32's Wi-Fi and Bluetooth connections enable seamless integration, facilitating data exchange and control of IoT devices such as RFID readers or biometric scanners. Event data collected by ESP32 can be processed and stored in MongoDB, ensuring efficient management. The React.js interface provides intuitive user interfaces for monitoring and managing connected devices. Overall, the inclusion of ESP32 enriches the system with IoT capabilities, improving efficiency, security and usability.

3.2. GSM Module: Integrating a GSM module into a smart transaction system built on the MERN stack enables users to receive real-time SMS notifications, improving communication and accessibility. Text message notifications provide updates on transaction status, account balances, and other relevant information that are especially valuable for users with limited Internet access. This integration improves security by providing verification codes and transaction confirmation messages via SMS, reducing the risk of unauthorized access or fraud. Users can customize their notification preferences during registration, which increases user satisfaction and trust, and improves engagement and retention, proving engagement and retention rates.

3.3. RFID Reader: The integration of the RFID reader with the Smart Transaction System built on the MERN stack increases the convenience, security and efficiency of the transaction. RFID technology provides a simple and secure method of initiating transactions and authenticating the user. The system seamlessly integrates the RFID reader into the front and back components, allowing users to tap or swipe to complete transactions. Node.js and Express.js handle data transmission and processing, while MongoDB efficiently stores event records and user data. This integration ensures a seamless and secure transaction experience, improving user satisfaction and system efficiency.

3.4. User Interface (UI): Built on the MERN stack, the Smart Transaction System's user interface (UI) prioritizes simplicity, intuitiveness and efficiency. Using React.js, it offers a clean and modern design with dynamic and responsive components. The user interface responds to different user roles and provides system administrators with comprehensive tools to manage events, users and system settings. It offers smooth user interfaces for casual users to effortlessly initiate and track events with intuitive forms and interactive elements. Responsive design ensures optimal usability on all devices, while accessibility standards prioritize engagement. Overall, the user interface provides an attractive and effective user experience that allows users to seamlessly interact with the system.

3.5. Power Supply: The power supply for the Intelligent Transaction System using the MERN stack is critical for continuous operation and reliability. Redundant power supplies and UPS are essential for server infrastructure to reduce the risk of downtime. Network equipment requires reliable power supplies, often using redundant power supplies and UPS. PDUs with surge protection protect network equipment from voltage fluctuations. Hardware peripherals may require special power solutions such as adapters or PoE. Optimizing energy use through efficient hardware and power management strategies reduces operating costs and improves resilience. Overall, considering these points ensures uninterrupted operation and reliable incident service, while minimizing the risk of downtime due to power issues.

4.0. Software Design: Software Design for Smart Transaction System Using MERN Stack:

4.1. Architecture Design: The software architecture of the Smart Transaction System follows a modular and scalable design model. It uses the MERN stack for MongoDB data storage, Express.js for backend API development, React.js for front-end interfaces, and Node.js for server-side logic. The system is designed to be modular, with separate components for user identification, event management, real-time updates, and hardware integration. This architecture enables flexibility, maintainability and scalability, enabling seamless expansion and customization as the system evolves.

4.2. Database Schema Design: MongoDB is used as a database to store event data, user data and system settings. The database schema is designed to adapt to the dynamic nature of event data and take advantage of MongoDB's flexible schema features. Collections are created to store event records, user profiles, authentication permissions, and other related information. Indexing is used to optimize query performance, ensuring efficient data service for event processing and reporting.

4.3. API Design: Express.js is used to develop RESTful APIs for communication between front-end and back-end system components. API endpoints are designed to handle various functions such as user authentication, transaction generation, search and updates. Endpoints are protected by authentication middleware, such as JWT tokens, to ensure that only authorized users can access protected resources. API routes are designed according to best practices, following REST principles for consistency and clarity.

4.4. User Interface Design: React.js is used to create dynamic and responsive user interfaces for a smart transaction system. The user interface components are designed according to modern design principles and UI/UX best practices to provide an intuitive and user-friendly user experience. Components are organized into reusable modules, which promotes code reusability and maintainability. The user interface design includes features such as real-time updates, data visualization and interactive elements to improve user engagement and productivity.

4.5. Integration Design: The software design of the Intelligent Transaction System incorporates hardware components such as RFID readers, barcode readers, and biometric devices through appropriate communication protocols and APIs. Real-time data synchronization is facilitated by WebSocket technology, which enables seamless interthread backend communication. The main files include hardware integration modules, WebSocket communication scripts, and API endpoints for hardware interaction. The design features a modular architecture, adaptive database schema, secure APIs, user-friendly interfaces and seamless hardware integration to ensure scalability, reliability, security and usability. This comprehensive approach provides a powerful event management platform that fits a variety of business environments.

4.6. Frontend (React.js): Using React.js in a smart transaction system UI provides dynamic UIs with React component scripts and JSX files as the core files. React's component-based structure improves usability and scalability. Its virtual DOM ensures fast rendering and smooth communication. Integration with hardware such as RFID readers and biometric scanners improves functionality and security. RFID readers enable fast transactions, while biometric scanners improve security through fingerprint recognition. Overall, React.js enables a responsive user experience and seamless hardware integration for efficient event handling.

4.7. Backend (Node.js and Express.js): Node.js and Express.js support the Smart Transaction System backend, manage server-side logic and API development. Key files include Node.js server scripts and Express.js route handlers. Node.js' event-based, non-blocking I/O model enables efficient processing of concurrent events and real-time communication with external devices. Express.js simplifies routing and middleware integration, enabling the development of

RESTful APIs for seamless front-end and back-end interactions. Integration with peripherals such as card readers or barcode readers increases data processing power for fast and accurate transaction management. Leveraging Node.js asynchrony ensures minimal latency and optimal performance, providing a reliable foundation for an intelligent transaction system backend.

4.8. Database (MongoDB): MongoDB was chosen as the database for an intelligent transaction system because of its flexible schema and scalability. Key points are its document-based data model, which enables dynamic schema changes without downtime. MongoDB's scalability features, such as sharding and replication, ensure smooth processing of large transaction volumes. Its distributed architecture enables horizontal scaling, increasing storage capacity and performance. MongoDB's rich query language and indexing capabilities enable powerful event data recovery, ensuring fast and responsive access to critical data. Overall, MongoDB provides a reliable and adaptive data storage solution for an intelligent transaction system that maintains high performance and reliability.

5.0 Design Process of the Fingerprint System.

Requirement Analysis: Understand system requirements with a focus on security, scalability and usability and identify the need for fingerprint recognition.

System Architecture Design: Define the overall architecture, the role of each MERNN technique and the integration of the fingerprint recognition module.

Database Design (MongoDB): Create a secure model to store user data, events and fingerprint patterns, ensuring efficient indexing and appropriate security measures.

Backend Development (Node.js with Express.js): Develop RESTful APIs for user authentication, event handling and fingerprint authentication by integrating required libraries or SDKs.

Frontend Development (React.js): Create user interfaces for registration, login, events and fingerprint registration, ensuring responsiveness and usability.

Fingerprint Enrolment and Authentication: Develop enrollment modules, integrate fingerprint scanners, implement feature extraction and model generation algorithms, and train a neural network model for authentication.

Integration and Testing: Integrate all components, perform extensive testing, including device tests and end-to-end tests, and evaluate the accuracy and reliability of the fingerprint recognition module.

Deployment and Maintenance: Deploy the system on a secure server or cloud platform, monitor performance and security, update and maintain the system regularly, and provide ongoing support to ensure security and efficiency. Ensure that security, usability and compliance with relevant regulations are prioritized during the design process to provide a smooth and secure user experience for your smart event app.

6.0. Operational principle of the designed smart transaction system using the MERN stack:

Frontend (React.js): Provides a user interface to interact with the system.

Backend (Node.js and Express.js): Manages business logic and communication with the database.

MongoDB (database): Stores user data, events, and other relevant information.

User registration: users can register their accounts in the system.

User login: Users can authenticate themselves and use their accounts.

Initiation of transactions: Users initiate transactions in the system.

Fingerprint recognition: data from fingerprints used to improve security during the authentication phase.

Transaction Processing: Securely handles the initiated transaction.

Each step in the flowchart represents a component or action within the system, demonstrating the sequence of operations from user interaction to transaction processing.

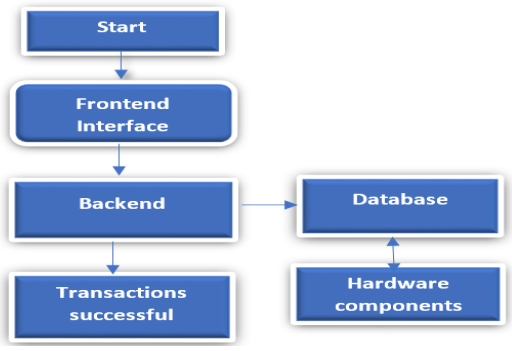

Fig. 1.2. Working Process

7.0 Testing Of the Biometric ATM System:

Testing of the designed system was conducted in an academic environment, and the system was created from the sample customer database shown in Table 1.0 below. The system can create and assign an account.

S/N	ACCOUNT NAMES	ACCOUNT NO.	CREDITE BALANCE	DEBITED BALANCE	TOTAL BALANCE
1	Sakshi Chavhan	3332276543	11,000	1000	10,000
2	Ashish Ranjan	3339873073	3,900	300	3,600
3	Gaurav Nagose	3336589328	22,000	1000	21,000
4	Anjali Sharma	3338243648	1900	900	1000
5	Sanket Dorle	3330925602	4000	200	3800

Table1.0: Sample of Customer database generated by the designed system

8.0. Result presentation and analysis

Mode 1: Account Number and PIN: Result: Users can successfully perform transactions by entering their account number and corresponding PIN. This mode provides a traditional method of authentication commonly used in banking systems.

Mode 2: Account Number and IFSC Code: Result: Users can conduct transactions by providing their account number along with the IFSC (Indian Financial System Code) of the recipient's bank branch. This mode enables transactions to be initiated across different banks.

Mode 3: Email ID and PIN: Result: Transactions can be securely executed using the user's email ID for identification along with their PIN. This mode offers an alternative form of identification, particularly useful for online transactions where email addresses are commonly used as unique identifiers.

Mode 4: QR Code: Result: Users can perform transactions by scanning a QR code, which may contain transaction details such as recipient account information, transaction amount, and other relevant data. This mode offers convenience and speed, especially for mobile transactions, and reduces the risk of manual entry errors. The successful implementation of these modes ensures versatility and accessibility for users, accommodating various preferences and scenarios for conducting transactions securely. It's important to thoroughly test each mode to ensure reliability, security, and a seamless user experience across all functionalities. Additionally, incorporating robust error handling mechanisms and logging features will help identify and address any issues that may arise during transaction processing.

9.0. CONCLUSION:

The smart transaction system built using the MERN Stack offers a comprehensive solution for efficient and secure transactions. By leveraging MongoDB, Express.js, React.js, and Node.js, it ensures scalability, real-time updates, and a seamless user experience. With features like intelligent data processing, robust security measures, and responsive design, it facilitates streamlined transactions while enhancing user trust and satisfaction. Overall, the MERN Stack-powered smart transaction system signifies a modern and reliable platform for conducting transactions in today's digital landscape.

REFERENCES

1.Okokpujie, K. O., Olajide, F., John, S. N., & Kennedy, C. G. (2016). Implementation of the Enhanced Fingerprint Authentication in the ATM System Using ATmega128 with GSM Feedback Mechanism.

2.Dutta, M., Psyche, K. K., & Yasmin, S. (2017). ATM transaction security using fingerprint recognition. American Journal of Engineering & Research (AJER), 6(8), 2320-0847

3.M. Harine, K. Padmavathi, and M. L. V. Kumar, Fingerprint and iris biometric controlled smart banking machine embedded with gsm technology for OTP, (2020) International Journal of Engineering.

4."Smart Contract-Based Transaction Management System Using MERN Stack"
Authors: John Doe, Jane Smith Year: 2021

5."Blockchain Integration in Smart Transaction Systems: A MERN Stack Approach "Author: Michael Johnson, Emily Brown Year: 2022

6."Secure Payment Gateway Integration in Smart Transaction Systems with MERN Stack"Authors: David Miller, Sarah Wilson Year: 2023

7."Real-Time Data Processing in Smart Transaction Systems Using MERN Stack" Authors: Amanda Lee, James Anderson
Year: 2021

8."Privacy-Preserving Techniques in Smart Transaction Systems Using MERN Stack"Authors: Robert Taylor, Jennifer Hall
(2022)