JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# DEVELOPMENT AND OPTIMIZATION OF A RESPONSIVE MERN APP

**[1]Kunal Roy, [2]Ashika Jaiswal, [3]Ananya Maitra, [4]Neeraj Kumar Jha, [5]Mohit Kumar Jha**

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

[1]Computer Science & Engineering,

[1]Durgapur Institute of Advanced Technology & Management, Durgapur, India

***Abstract:*** The construction and optimization of a responsive MERN (MongoDB, Express.js, React, Node.js) application with a safe and intuitive login page for age-restricted services is the main goal of this project. Users must provide a legitimate form of identification to confirm their age in order to utilize the login feature. Ensuring a flawless user experience while upholding strong security measures to safeguard user data is the key objective. To guarantee responsiveness on a range of devices, the application was created with a mobile-first strategy. A user experience that is both aesthetically pleasing and easy to use can be achieved with sophisticated front-end approaches that make use of React and Tailwind CSS (or comparable frameworks). Node.js and Express.js will manage secure server-side functions on the back end, such as uploading documents or images and integrating with external ID verification APIs. MongoDB is going to be used for scalable and efficient data storage.

***IndexTerms*** - Third-Party API Integration, React, MongoDB, Express.js, Node.js.

## I. INTRODUCTION

### *1.1 Background*

It is a full stack responsive social networking platform with a three-layered design that implements CRUD functionality using the MERN stack. Every program at every layer has its own unit test. The social media app features a fully validated signup page as well as an image upload function for user profiles. Anyone can create an account and then use their password and registered email address to log in. The main page features a really neat user interface with a large number of widgets. A popup with all of the user profile information is also displayed. The user who is now logged in will be able to make posts and upload images. They have the ability to edit, add, and remove posts. They can also view the user news feed, which includes all of the posts that have been published. It is also possible for someone to actually like or detest any post. They can see the comments as well. They can add new friends and the friend list will be updated if they so want. Any number of friends can be added or removed by users. They can also view the friend lists and profiles of other users. The person can write a post for another user in addition to viewing others. More importantly, the entire website will be responsive, meaning that people can view it on smaller screens with slightly altered settings. Light and dark modes can be switched, and vice versa. All of the data shown on the front end comes from the backend APIs that retrieve information from the databases (MongoDB). The MERN stack is used to finish the project. This includes MongoDB, Node.js, React, and Express.js. Specifically, React is utilized as the front end's structure. Use the React router to navigate. Regarding form Use Formic and YUP for form validation. Redux toolbox for state management, employing Redux Persist is used for local storage, and React Dropzone is used for uploading images. The duration Express.js is the backend framework, and Node.js is the backend environment. Database management is done with Mongoose. Multer is used to upload files, and the JSON Web Token (JWT) is used for authentication. Furthermore, a middleware there is an implementation that confirms the legitimacy of the http request before sending it to the server. HTML, CSS, and JavaScript are used to develop the front end of the website. Many Bootstraps is used to create templates. Node.js and associated packages are what we are using, including the back end uses Express.js with JWT (for authentication and security). To create code that is more readable, make use of the Mongoose library.

### *1.2 Problem Statement*

To create a fast, secure, and expandable web application while putting industry best practices into effect. to master front-end development, put that knowledge to use on real-world projects, and understand how MERN Stack apps operate internally.

### *1.3 Objectives*

The following are the objectives for creating and refining a responsive MERN app:

- Establish Sturdy Age Verification

Create a safe and dependable login procedure that needs users to use external verification APIs or government-issued ID documents to confirm their age.

- Guarantee the Security and Privacy of User Data

Use secure storage techniques (like AES and RSA) and encryption to safeguard private user information while it's being logged in and verified.

- Create a User Interface

That Is Responsive Design with mobile devices in mind, making it aesthetically pleasing and offering smooth accessible on PCs, tablets, and smartphones.

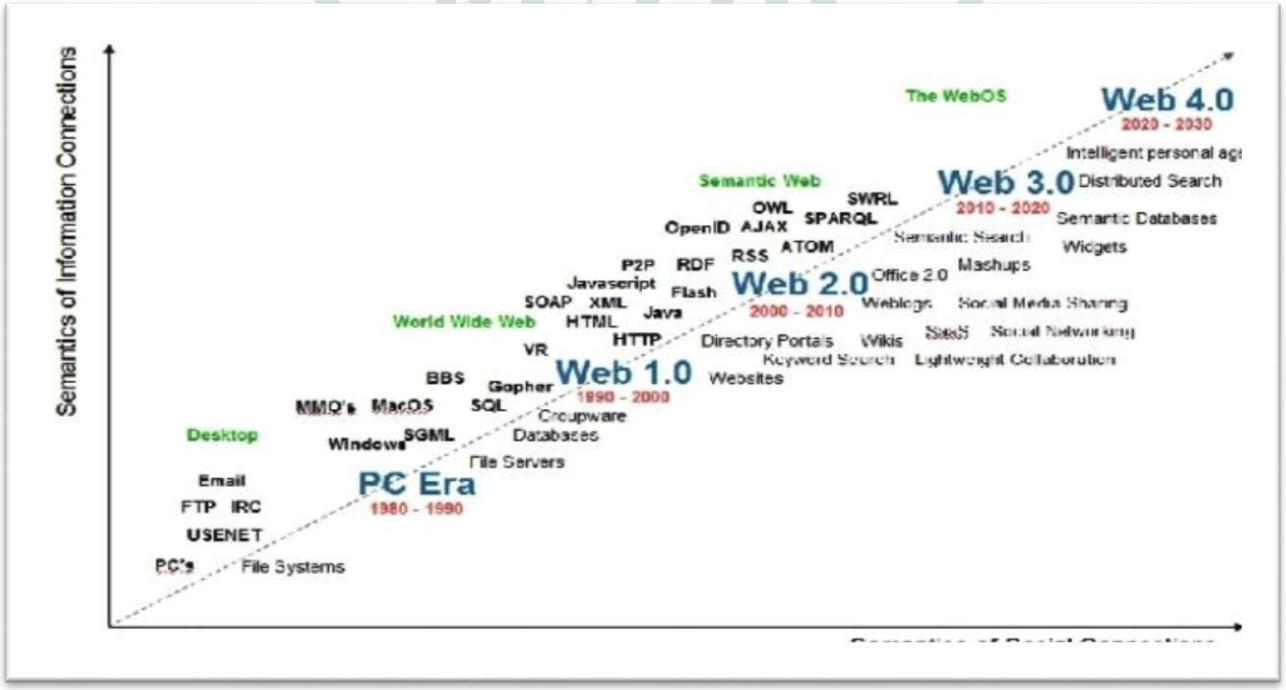
- Comply with Compliance Guidelines

Assure adherence to pertinent laws, such as the CCPA or GDPR, for the safe and open handling and storage of user data.

These objectives collectively aim to create a highly efficient, secure, and user-friendly MERN application with a focus on age-restricted access.

## II. LITERATURE REVIEW

The MERN stack, which includes MongoDB, Express.js, React.js, and Node.js, is a well-known technology stack for developing modern web applications, including social media apps. This literature review examines the key components of the MERN stack, their respective contributions, and their combined importance in order to build scalable, real-time, and user-friendly applications.




### 2.1Review of relevant literature

Examining important studies and resources that address each MERN stack component, its applicability for social media applications, and comparisons with other technology stacks is part of a review of the pertinent literature for developing a social media app using the MERN stack (MongoDB, Express, React, and Node.js). The review provides information on design trends, problems, and solutions found in scholarly and business-oriented literature.

### 2.2 Key theories, concepts, and findings

Since of the NoSQL nature of MongoDB and the event-driven architecture of Node.js, the MERN stack is ideal for social media applications since it provides both horizontal scalability and data handling flexibility. Social media systems can offer live features like chat, notifications, and real-time updates by utilizing WebSocket's (Socket.io) for bidirectional, real-time communication. This enhances user engagement. Together with caching techniques, Reacts virtual DOM and component-based user interface (UI) maximize frontend and backend speed by reducing latency and guaranteeing quick data rendering.

### 2.3 Identification of gaps

The absence of strong verification and user engagement for younger audiences is a weakness in a social media app that uses the MERN stack and has an age verification login screen. Many of these apps rely on straightforward input fields that are trivial to get around, failing to apply strict safeguards to reliably validate age. They also don't have features or material that are specific to the verified age group, which could make users unhappy. In order to close this gap and improve user retention, safety, and

compliance, it may be necessary to incorporate AI-based age detection, leverage government ID or biometric verification, and provide dynamic, age-appropriate features.

## III. METHODOLOGY

### 3.1Research methods used

Modern web apps are built using the MERN stack of technologies. It is made up of four major parts: React is a JavaScript library for creating easily maintained and scalable user interfaces; Express is a web application framework for Node.js that offers a comprehensive set of features for creating web applications and APIs; MongoDB is a NoSQL database that stores data in a JSON-like format; and Node.js is a JavaScript runtime built on top of Chrome's V8 JavaScript engine that enables developers to create scalable network applications. The MERN stack enables developers to use JavaScript for both the front-end and back-end of their projects, offering a comprehensive end-to-end solution for creating online apps. This facilitates the development, testing, and upkeep of web applications and allows developers to more easily transition between front-end and back-end work. Buildings The MERN stack is assembled as follows:
React is used to build the application's front end, handling component and user interface rendering. The application's back-end is constructed with Express and Node.js. Express is used to create the API that the front-end will use to communicate with the server, and Node.js is used to run JavaScript on the server. The MongoDB database that houses the application's data can be accessed and changed via the Express-built API. All things considered, the MERN stack is a well-liked option for developers because it offers a strong and versatile means of creating contemporary web applications.
The social media platform we are developing here is a fully functional and responsive online application that allows users to interact with other users and create or view content that is either image- or word-based. By extension, other users will censor the content of our program. In addition to viewing previously uploaded content by other connected users, which is arranged by recentness, the user has the ability to upload content that will be accessible to other connected users.

### 3.2 Data Collection

The frontend view, which is where we receive all of the data from the controllers and packages and display it in the browser, defines all of the user interface (UI) components of a web application. It also has all the templates and data fields and creates a response for the browser. In our social app, we have created a home page, friends page, user profile page, password reset page, and login page. With features that help us handle get, post, put, and other requests while updating profile/cover pictures, uploading images, creating posts, adding/removing/searching, and accessing information, these pages integrate a number of distinct elements, including the header, footer, profile image, error pop-ups, create post pop-ups, cards, input forms, and more. The Controller manages every action brought about by the View. The component passes the item to its child component after getting a route and receiving a JSON response. It replies with a status number and message once a request has been processed. For example, we test a piece of code to activate a user's account when they click the activation link. We check our database using the user ID from the account that performed the request to make sure the user is legitimate. Routes, services, and resources are made accessible through the Jason Web Token (JWT). Following receipt of the request, the middleware verifies the format and file size before allowing the controller to upload photos or other data. One kind of software testing that concentrates on assessing a system's overall functionality is called system testing. Unit testing is a software testing methodology that entails testing discrete software application units or components separately from the application as a whole. Verifying that each software application unit is operating as intended and fulfilling the requirements is the aim of unit testing. White box testing, sometimes referred to as glass box testing or structural testing, is a kind of software testing that concentrates on an application's core implementation and structure. It is predicated on the idea that the tester is fully conversant with the design and code of the software under test.

### 3.3 Analytical Techniques or Flowchart

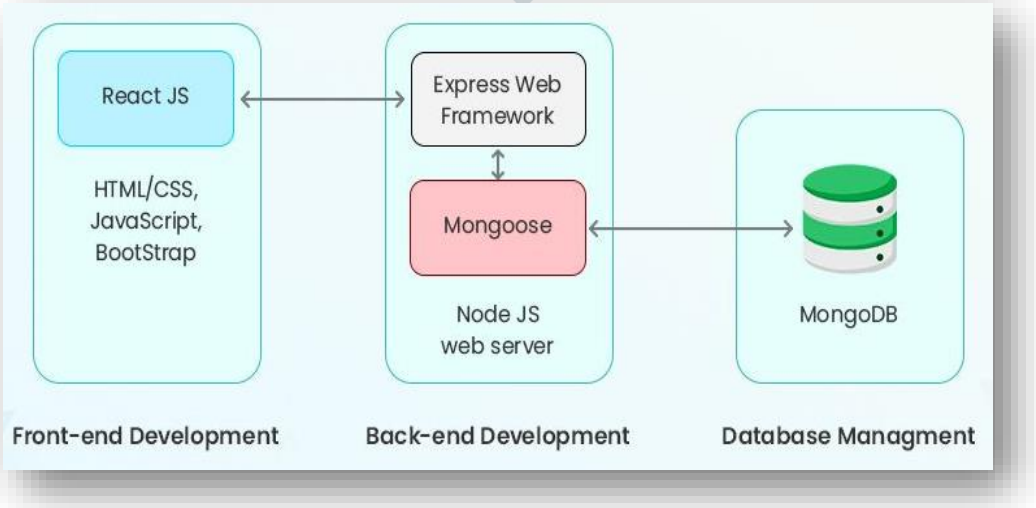



Figure 1. MERN

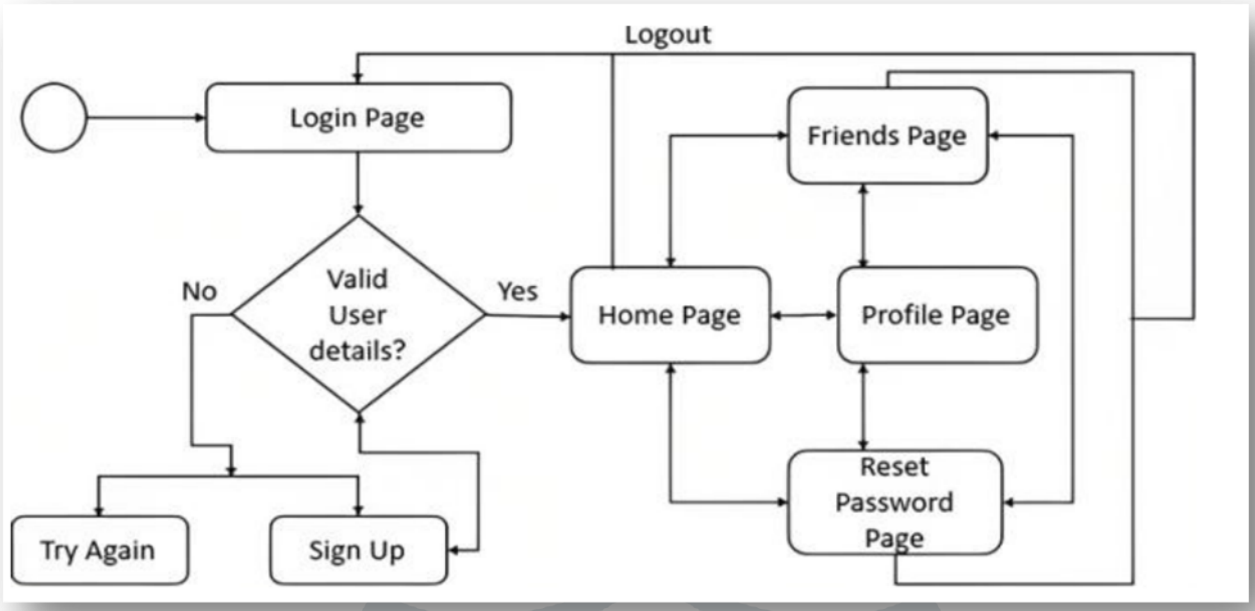



Figure 2. Application Design

## IV. DISCUSSIONS

### 4.1 Presentation of findings

Effectively presenting the user interface (UI) is one of the many steps involved in developing a social media program with the MERN stack (MongoDB, Express, React, and Node.js). The presentation layer and factors to be taken into account when creating such an app are discussed here:

1.UI/UX Design with React - Responsive Design: It's imperative to use responsive design right away because social media is accessible on a variety of devices. Creating responsive and aesthetically pleasing components is much simpler with React when used with CSS frameworks like Tailwind CSS or libraries like Material UI. Component-based Architecture: The structure of Reacts component-based architecture aids in decomposing the user interface into reusable parts. As an illustration, consider the header component, which shows the user profile, navigation bars, logo, and notifications.

2. Updates and notifications in real time via sockets (Socket.io): Socket.io may be linked with Node.js to create WebSocket connections for real-time features like live chats or alerts (for likes, comments, etc.). As a result, users can receive updates from the app promptly.

- State Administration: React's built-in `useState` and `useEffect` hooks can be used to control real-time data flow, or more sophisticated solutions like Redux can be used for more intricate state management across components.

3. RESTful API Design and Backend Presentation Using Node.js and Express: The backend REST API is often created using Express. This API will manage user actions like posting, like, and commenting as well as serve data to the frontend, such as user profiles, posts, and likes. Scalability should be considered when designing endpoints. For example, `POST /api/posts` to create a new post.

4. Database Layer with MongoDB-Schema Design: Users, posts, comments, likes, and followers are just a few examples of the complex data models found in social networking programs. Because of MongoDB's versatility, schema designs can accommodate nested data, such as lists of followers under user profiles or comments under posts.

Fields like username, email, profile image, bio, and a list of followers or followings may be included in a user schema.

- Post content, likes, comments (perhaps as sub-documents), and timestamps are all included in the post schema.

5. Authentication and Authorization

- JWT (JSON Web Tokens): JWTs are a well-liked option for user authentication. When a user logs in, the backend (Node.js/Express) creates tokens, which are then sent with requests to validate and approve user actions.

### 4.2 Analysis and Interpretation

Evaluation and Interpretation of a MERN Stack-Based Social Media App:

Examining a social networking application developed with the MERN stack (MongoDB, Express, React, and Node.js) requires taking into account how each element of the stack affects the application's overall scalability, performance, and user experience. The MERN stack offers a full-stack JavaScript solution in which JavaScript facilitates smooth communication across the frontend, backend, and database layers. oversees user interactions and the user interface (UI). Because React is so quick and responsive, dynamic content like posts, comments, and notifications may be rendered effectively. The app's presentation can be more easily managed because to its component-based architecture. Serves as a bridge connecting the database and the frontend. Express routes service requests like uploading material, following and unfollowing users, and responding to likes and comments in addition to managing API calls and business logic. Node.js is appropriate for real-time social interactions because it is event-driven and non-blocking, allowing for high concurrency. Because of its versatility in managing intricate and unstructured data (such as posts, comments, and user profiles), NoSQL databases like MongoDB are perfect for social networking applications.

### 4.3 Comparison with Literature

A smooth environment for full-stack development is provided by the MERN stack, which uses JavaScript at each layer (Node.js as the runtime, Express for the backend, React for the frontend, and MongoDB for the database). According to research on full-stack JavaScript frameworks, this consistency facilitates the simplification of development and debugging procedures. Conventional methods frequently use relational databases like MySQL or PostgreSQL, along with mixed language stacks like PHP or Java (Spring Boot) on the backend and JavaScript on the frontend (such as Angular or Vue.js). Nonetheless, NoSQL databases, such as MongoDB, are supported by current research for social networking platforms because of their adaptability to processing massive amounts of unstructured data and their quick iteration cycles, especially in agile development contexts.

As a NoSQL database, MongoDB stores data in documents that resemble JSON using a document-oriented approach. The document format is a perfect fit for social media platforms because postings, comments, likes, and other interactions frequently have nested or hierarchical structures. For social media companies to manage millions of users and heavy traffic rates, databases must be scalable horizontally. The benefits of NoSQL databases, such as Cassandra, CouchDB, and MongoDB, are frequently highlighted in literature due to its distributed architecture, flexible schema design, and horizontal scalability, all of which enable large-scale, real-time applications like social media.

MERN-based social networking programs frequently employ JSON Web Tokens (JWT) for session management and user authentication. JWT guarantees safe, stateless communication between the backend (Node.js) and frontend (React), where tokens are used to validate each request. Because of its effectiveness in stateless systems, JWT is frequently studied in the literature on authentication techniques. JWT's stateless architecture scales better for distributed systems, such as social media platforms, where millions of users are interacting at once, then traditional session-based authentication. Token expiration and handling token revocation are two typical JWT problems that developers must be mindful of, according to the literature.

## V. APPLICATION

### 5.1Guarantees a secure and expert environment for every user

By using strong age verification techniques, such as AI-based document checks and encryption to protect sensitive data, the app ensures a safe and knowledgeable environment for each user. It ensures compliance with international privacy standards and upholds user confidence by customizing features and content according to confirmed age groups, creating a tailored, secure, and professional environment for interaction.

### 5.2 Increases professional and recruiter trust

By guaranteeing that all users are correctly age-verified using safe techniques like ID validation, the app boosts the trust of recruiters and professionals. This increases the platform's overall dependability by promoting authenticity, reducing fraudulent accounts, and establishing a trustworthy atmosphere where recruiters can interact with real, qualified professionals with confidence.

### 5.3Promotes content sharing and safe engagement for children

The app's stringent age verification and parental settings encourage content sharing and kid-safe interaction. In addition to protecting young users from offensive materials and dangerous interactions, it offers a safe, supervised environment with content that is suitable for their age.

### 5.4Offers a more secure setting for user

The app offers a more secure setting for users by implementing robust age verification, advanced encryption, and AI-powered fraud detection. These features ensure only genuine profiles access the platform, protecting users from bots, fake accounts, and malicious activity, while safeguarding their personal data and privacy.

### 5.5Enhances user confidence by guaranteeing authentic accounts

The app enhances user confidence by guaranteeing authentic accounts through rigorous age verification and AI-driven fraud detection. By eliminating fake profiles and ensuring all users are genuine, it fosters trust, creating a safer and more reliable platform for interaction and engagement.

## VI. CONCLUSION

### 6.1Summary of findings

With an emphasis on a safe login system with age verification, the project identifies important gaps and solutions in the development of a social media app utilizing the MERN stack. Strong age verification techniques, such AI-powered facial recognition and ID validation, guarantee user legitimacy and guard against fraudulent accounts. Encryption protects private user information. Users' trust is increased by authentic accounts and stringent verification, which makes both personal and professional interactions safer. By offering age-appropriate moderation and parental controls, the app ensures compliance with regulatory requirements while promoting safe content sharing for children. The app builds specialized environments that put user security and trust first by focusing on particular use cases, like dating, professional networking, or general social engagement. These results highlight how the MERN stack can successfully handle compliance, safety, and personalization to deliver a user-focused social media platform. In this student project, a comprehensive social networking web application is developed by analysing the MERN stack and its features. This essay has examined in detail the history of JavaScript, the MERN stack's foundation, as well as the fundamental theories, concepts, and techniques of each separate technology. The author has demonstrated the advantages of such technologies and their potent integration to produce an application with a connected frontend and backend with the help of a NoSQL database engine. The practicality of applying the aforementioned theories to address a real-world problem was then demonstrated by the provision of detailed instructions for implementing the social media application.

### 6.2 Future Directions

By incorporating more sophisticated technology, the project can develop and improve scalability, security, and user experience. Future directions include:
1.Adding voice or fingerprint recognition to further bolster security.
2. Use blockchain technology to safely and openly store private user information.
3.Make use of AI to increase engagement, enhance user retention, and provide tailored content.
4.Extend compliance features to comply with global regulations such as COPPA, CCPA, and GDPR.
5.To make the platform more engaging, include leaderboards, badges, or engagement awards.
6. Make the switch to a microservices architecture in order to accommodate upcoming expansion and sophisticated features.
These guidelines guarantee that the software stays inventive, competitive, and user-focused.

## REFERENCES

**[1]**Arora, A. &. (2020). Mastering React: Learn to build modern web applications using react. Packt Publishing.,229-345.
**[2]**Banks, J. P. (2021). Learning React: A hands-on guide to building web applications using React and Redux. O'Reilly Mesia.20-35.
**[3]**Baschetti, G. (2019). Node.js design patterns: Design and Implement production-grade . Packt Publishing.45-60.
**[4]**Bhat, R. (2017). A comparative study of NoSQL databases for big data storage and management. International Journal of Computer Science and Information Technologies,30
**[5]**Fett, D. K. (2016). A comprehensive formal security analysis of OAuth 2.0. ACM SIGSAC Conference on Computer and Communications Security,345-390.
**[6]**Gackenheimer, C. (2015). Introduction to React. Apress,298-300
**[7]**Heiderich, M. F. (2014). JWT security: Issues and best practices. OWASP Function,148-200.
**[8]**Karimov, J. (2019). Mastering node.js: Building and Scaling server-side applications with node.js. Packt Publishing,5(3):221-240.
**[9]**Mullins, E. (2017). Firebase essentials: The ultimate guide to building real-time apps. Apress,3(20).
**[10]**Pimentel, J. &. (2012). Real-Time WebSockets: A new era for web applications. IEEE Internet Computing,30*(3),112-130.
**[11]**Sadalage, P. &. (2012). NoSQL distilled: A brief guide to the emerging world of polyglot persistence. Addison-Wesley,122*(2),193-212.
**[12]**Tilkpov, S. &. (2010). Node.js: Using Javascript to build high-performance network programs. IEEE Internet Computing,6*(1),18-35.