**ABSTRACT**

**Researcher**: Dhanshree Ramrao Patil, Varsha Milind Dhole, Prajakta Rameshrao Ganjapurkar

**Presentation** **Title**: MERN Chat App (Web application)

**Research Focus:** Software development using new emerging technologies

**Presentation** **Type**: Project Presentation

**Abstract**: In a request-response scenario, there is no way for a server to send data to the client without having the client to request something first. A client would have to continuously ask for changes in regular intervals (polling), which is not what we consider real-time. Imagine a chatting app, where you would only see new messages every 30 seconds. There are ways for a server to notify a client upon new events via HTTP messages. You could push messages to a service worker running in the client’s browser or have the client subscribe to server-sent events (SSE). With Web Sockets however, we get this feature for free and can easily broadcast new messages received to every other client connected.

On the client-side users can create accounts that will be stored in the database. Then users can authenticate with the given credentials, if those are correct the server sends a unique token to the client. The client stores it for use on restricted backend route requests.

Once authenticated, the server creates a socket bidirectional connection with the client to facilitate the chat functionality.

Every time a user sends a message, this goes to the server which redirects it to the desired respondent. Also every time a user enters or leaves chat, the server announces all the connected clients.

Asynchronous JavaScript - **JavaScript** is a single-threaded programming language which means only one thing can happen at a time. That's where **asynchronous JavaScript** comes into play. Using **asynchronous JavaScript** (such as callbacks, promises, and **async**/await), you can perform long network requests without blocking the main thread.

**Features**:

* Register and authenticate users using JSON web tokens
* You can chat with all the registered users
* Open 1on1 private conversations with a user of your choice
* Notifications for messages on background conversations using firebase
* Have a pleasant UI/UX built to look awesome from desktop to mobile view(responsive)

**Technologies**:

* [NodeJS](https://nodejs.org/) - JavaScript backend/server-side solution of choice
* [Express](https://expressjs.com/) - Node framework that makes handling http requests with ease
  + [JsonWebToken](https://www.npmjs.com/package/jsonwebtoken) - package that helps with generating JWTs for secure authentication
* [MongoDB](https://www.mongodb.com/) - data storage solution that just speaks JSON and pairs very well with Node
  + [Mongoose](http://mongoosejs.com/) - package that helps with object modeling and manages connection between server and database
  + [Bcryptjs](https://www.npmjs.com/package/bcryptjs) - for salting and hashing the user password to be stored in the database
* [Socket.io](https://socket.io/) - web sockets implementation, fast and reliable real-time communication engine
* ReactJS - rich frontend web js library, helps creating fast, reliable web applications
  + Material-UI - CSS/JS framework, makes it easy to develop responsive, well polished web apps
  + AuthGaurd - To authenticate user if token is present then an then only application allows to login

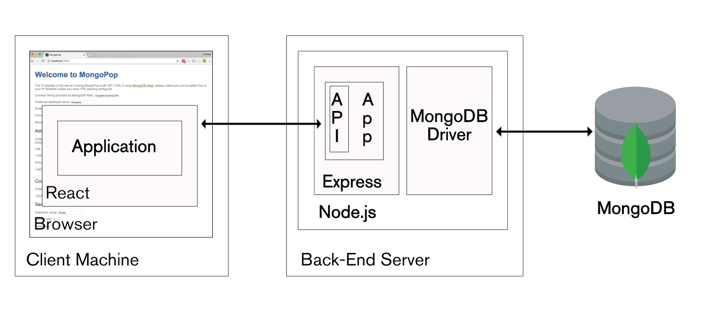


fig: Block diagram of MERN web application