A Project Report

on

Online Chatting Web Application

Oasis

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Dr. Kirti Shukla
Associate Professor
Department of Computer Science and Engineering

Submitted By

18SCSE1140025- Tanmay Srivastava

18SCSE1140061 – Himanshu Kumar Singh

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA

DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE’S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **“Online chatting web application:Oasis”** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** Submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Dr Kirti Shukla, Associate Professor**, **Department of Computer Science and Engineering** of School of Computing Science and Engineering, Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1140025 – Tanmay Srivastava
18SCSE1140061 – Himanshu Kumar singh

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Dr. Kirti Shukla, Associate Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **18SCSE1140025 – Tanmay Srivastava, 18SCSE1140061 – Himanshu Kumar Singh** has been held on ________ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

ABSTRACT

In today's expeditiously moving world, it is extremely crucial to be able to communicate in the most efficacious and simple manner. As most business is also moving to the online world it is very important to have an efficient way of communicating. This dissertation describes the process of the development of a chat application for developers, from a mere idea to a working cloud service. The author has built a real time platform that makes it easy to have a group conversation between a projects' members, share code and stay up to date with their latest repository updates. This project dispenses an approachable way for people to view orders communicate. In contemplation to establish a real-time chatting web application, several technologies have been studied and acknowledged.

Technologies that have been included are, React.js, MongoDB, Node.js, Express.js. This is a project to ease the manage a group chat among people and establish a web application where a person is delivered with an exhaustive web application and also to understand the technologies used to demonstrate such an application. This paper will discuss each of the fundamental technologies to create and implement an chatting web application.

The software world is evolving quickly, and oftentimes people find themselves left behind, even the most experienced ones. The purpose of this project is to create a full application with the latest technologies, but also to keep up to date with the ever-changing development ecosystem.

Full-stack development currently has many different stacks and technologies to learn. It is very easy to get overwhelmed and get distracted. React is one of the most popularly used Front-end libraries used by companies such as Facebook, Pinterest, Uber, Instagram and many more.
As communication is becoming more and more important day by day and keeping track of messages and data of people you are communicating with is also very important.
By creating a web app which can do things like above is the main goal of our project.

[illegible]

Signature of Guide

Table of Contents

Chapter :1

Introduction

In the real world, communication plays a very vital role. People have been communicating with each other through various applications or mediums. In the beginning, people communicated with each other using letters or other sources, as these mediums could take much time to deliver the content. Cell phones are another medium of communication but the drawback /disadvantage is for any limited or small message which needs to be passed to another user than a phone call is not a perfect way. The developers/engineers then looked to implement a text-based communication which would allow an in instant communication service.

Chat applications have become one of the most important and popular applications on smartphones. It has the capability of exchanging text messages, images, and files which it cost-free for the users to communicate with each other. All messages must be protected. The aim of the paper is to propose a chat application that provides End-to-End security that lets safely exchange of private information with each other without worrying about data. In addition to the protection of storage. A list of requirements to make a secure chat application is presented in this paper and based on these requirements, the application was designed. The proposed chat application was compared with other popular applications based on those requirements as well as it has been tested as proof for providing End-to-End security.

In 1984, the concept of SMS was developed in the Franco-German GSM cooperation by Friedhelm Hillebrand and Bernard Ghillebaert. The limitation of SMS was the limited size i.e., 128 bytes, after the accent of smartphones from 10yearsmany messaging applications have been developed. Some are Bluetooth-based and some were internet-based such as WhatsApp, We Chat and others. Android is an operating system for mobiles that was developed by Google. This operating system permits the applications to be utilized on mobiles. As it was developed by Google, android users can create mobile applications and can be sold through android application stores such as play store. Firebase is a NoSQL database that makes use of sockets which allows the users to store and retrieve the data from the database. An Android version should be greater than 2.3, android studio 1.5 or higher version, and android studio project are the prerequisites to connect the firebase to an android application. Firebase provides various kind of services such as: Firebase Authentication: Firebase Authentication is useful to both developers and users. Developing and maintaining sign-in set-up may be a bit difficult and time taking.

Firebase provides an easy API for sign in. It also provides the data backup using real time databases. Firebase cloud: For storing the data such as video, text, pictures building the infrastructure would be difficult and expensive for a new developer so the firebase provides the platform of cloud storage . Real-time database: It is a cloud-facilitated NoSQL database. Aside from the authentication, cloud service and real-time databases firebase also provides a service for crash reporting Crash Reporting: when some unexpected crashes occur in any applications it may be difficult to conclude why the application crashed. Firebase provides crash reporting service to deal with these crashes. This paper is concerned of a software application for the establishment of a real-time communication services between operators/users. Chat application many-to-many type of communication system where the users will able to exchange the messages among themselves. Users can create the chatroom according to the requirement or can also join to the existing chatrooms

The name of our project is “Oasis”, it is a web application built using the MERN stack for real-time messaging.
Oasis is a chat app, similar to programs such as Skype or TeamSpeak, or professional communications platforms like Slack.
It uses React as front-end and node js as back-end. And it is using MongoDB as a database.

The goal of this project was to build a scalable web application and to demonstrate the working of the MERN stack.

Oasis is an implementation of a web-based online messaging application with features to make groups or communities in form of channels. The technologies used in making this project are Mongo DB, React Js/REDUX, Node Js, Express, and Push
The frontend to end part of the application is built using Reacts. And for the backend, we have used Node Js. Mongo Deserves as the database for this application. These are the most popular technologies for building a scalable a web application at the current time. The user interface is built very simple and easy to understand so a new user will not face any difficulties regarding the UI. The signup and login is enabled via Google using fires base, so it’s very easy to log in.
A registered user can create their own channels, which can be public or private.
A "channel" is nothing but a container for several "chats" (where the conversations happen). Project administrators might want to create their own rooms for each of their projects, having one or more chats to discuss its development. Having more than one chat is useful to divide the different topics inside a project, instead of having conversations of mixed contents altogether.

The Name of our project is “Oasis”. It is a web application built using the Mern stack for real-time messaging. Oasis is a chat app, similar to a program such as Skype or TeamSpeak or a professional communication platform like slack.
It uses React as the front end and node.js as the back end. And it is using Mongo DB as a database

The goal of this project was to build a scalable web application and to demonstrate the working of the Mern stack. With the rapid development of mobile phones, mobile devices have become one of an integral part of daily activities. In recent years, chat applications have evolved and made a major change in social media because of their distinctive features that attract audiences. It provides Real-time messaging and offers different services including, exchange text messages, images, files and etc. Moreover, it supports cross platforms such as Android and iOS. There are currently hundred million of users smartphone are using chat applications on monthly basis.

There are two types of architecture in those applications, client-server and peer-to-peer networks. In a peer-to-peer network, there is no central server and each user has his/her own data storage. On the contrary, there are dedicated servers and clients in a client-server network and the data is stored on a central server. Security and privacy in chat applications have an amount of importance but few people take it seriously. In a test done by the Electronic Frontier Foundation, most of the popular messaging applications failed to meet most security standards. These applications might be using the conversations informationtion for certain purposes. Moreover, reading private conversations is certainly unacceptable in terms of privacy. Most applications only used Transport Layer Security (TLS) for secure channels. The main contributions of this paper are the following:

1- Propose client-server mobile chat application which supports the status of the communicating parties whether online or offline.
2- Provide a friendship request service.
3. Secure key exchange, then calculate the session key.
4. Secure exchange of end-to-end messages.
5. Analysis and Test the proposed chat.

Chapter 2

Functionality/Working of Project

Present’s business net foundations began in 1990. At the end of 1990, Tim Berners-Lee created simple standards of the world wide web and lots of tools for efficient net utilization. Those equipment encompass Hyper Text Transfer Protocol (HTTP), Hyper Text Markup Language (HTML), the first net browser and code editor, first web server, and first net web page which defined new time period, world huge internet and approach of developing personal web page . From 1990, the internet was evolving fast, and the progress of its improvement may be defined through four generations. The first net generation web pages have been static, and now not frequently updated, and customers ought to best read internet content. The principle motto turned into examine the simplest web. All net pages had been written with HTML and the principle conversation protocol become HTTP. The second gen technology begins 2004, and phrases that make it are numerous social networks, blogs, the possibility of the person growing internet web page contents, and enhancement of user enjoy browsing net interface.

Throughout that period, well-known social networks have been found, like facebook, Twitter, LinkedIn, and others. Those social networks have enabled user connections around the world. And at the same time, new technologies, such as JavaScript, Document Object Model (DOM), Ajax, Cascading Style Sheets (CSS), eXtensible HTML (XHTML), eXtensible Markup Language (XML), emerged. , eXtensible Stylesheet Language (XSL), and Flash enabling the launch and delivery of web services, without problems with web distribution. The third generation of the web began in 2010, and was marked with a semantic web (adding semantics to the web), personalized content, intelligent search, and computer-generated creative ability to create a variety of content. Ontologies are used to represent meaning and reasoning. Apart from ontologies, the technology also used in third generation web Resource

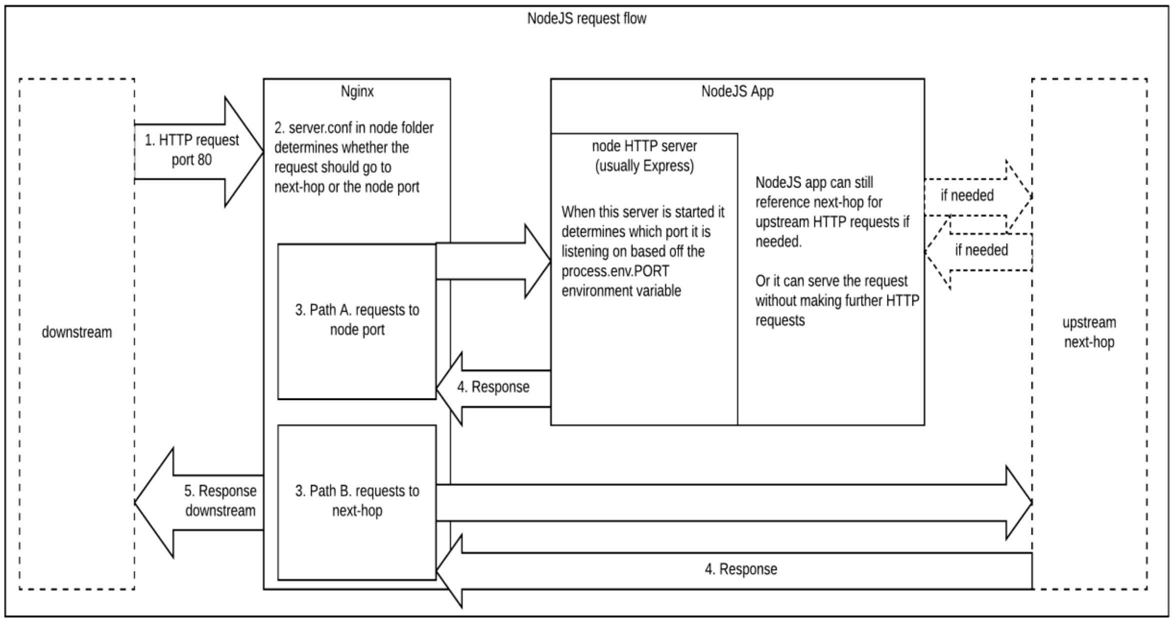
Description Framework (RDF), Web Ontology Language (OWL), and more. In the 4th generation, people can drive Internet 4.0 with a working web.
In the current 3.0 web, the use of search engines is still important, and that gives us information, in its major web content programs, that we can use according to our needs. Web 4.0 will be different. Once fully developed, it will not have the few steps required when using web 3.0, this way its use can be straightforward and unobtrusive.

Design is the universal language in the visual world, and web design refers to the user interface of the web page. The main purpose and goal of the design are to put content into focus, so users easily reach and use web content. Because of various technological changes and trends, web design has changed a lot, from first web generation web page which showed contents using a simple textual web page, through a second-generation web page with lots of graphics, vivid colors to achieve memorable web pages, and finally towards today's simple and intuiive web design. Web page design should always be modern and have updated content.

To set up the web application MERN stack was utilized. The web application was created utilizing the MERN stack utilizing Mongoose and MongoDB data base. Chrome developer's tools were used while testing utilizing revival devices for reproduction.
The accompanying segment examines MERN stack parts and their execution.

A. NodeJS
Originally built for Google Chrome and later on open-sourced, Node JS is a cross-platform run-time JavaScript environment used for executing JavaScript code outside of a browser. JavaScript was originally used for front-end scripting, but Node JS has enabled developers to use it to write command-line tools and back-end scripts for the purpose of creating dynamic web page content before the page is sent to the user's web browser.

Node JS was designed with the idea of allowing developers to build scalable network applications.
Node.js configuration utilizes the event-driven as the essential center idea for its current circumstance, which gave us the different number of APIs that are event-based and offbeat in nature which has helped us in building the site utilizing node.js for our back-end advancement. As we utilized Node.js, it utilized the comparing callback work as per our web application's business rationale. These callback capacities are executed non concurrently, which implies that albeit these capacities have all the earmarks of being enlisted successively in the logic, they try not to rely upon the code written in which they show up, yet maybe hang tight for the execution of the comparing event to fire. The fundamental benefit of Event-driven and offbeat writing computer programs is that it utilizes single-strung engineering. The execution of the call-back function code is managed without waiting for a specific code to finish, and the restricted assets were utilized for different errands that should have been executed as our web applications business rationale.
This plan was reasonable for our back-end advancement, which was likewise the objective of our system. In server development, taking care of coordinated requests was a significant undertaking, and blocking had the lead, to not completely utilizing the assets or squandering it. Through single string architecture, no concurrent callback capacities, we worked on the use of assets and advanced our site execution which likewise gave us the ideal outcomes while testing. From the upheld module given by Node.js, we can see that large numbers of the capacities, including document activities, are executed non concurrently, which is not quite the same as different languages. To


NodeJS request flow
downstream
1. HTTP request port 80
Nginx
2. server.conf in node folder determines whether the request should go to next-hop or the node port
3. Path A. requests to node port
3. Path B. requests to next-hop
5. Response downstream
NodeJS App
node HTTP server (usually Express)
When this server is started it determines which port it is listening on based off the process.env.PORT environment variable
NodeJS app can still reference next-hop for upstream HTTP requests if needed.
Or it can serve the request without making further HTTP requests
4. Response
if needed
if needed
upstream next-hop
4. Response

facilitate the advancement of the server, Node.js utilizes particularly enormous network modules, Including HTTP, DNS, NET, UDP, HTTPS, TLS, and so on, engineers can set up a Web server utilizing these network modules.

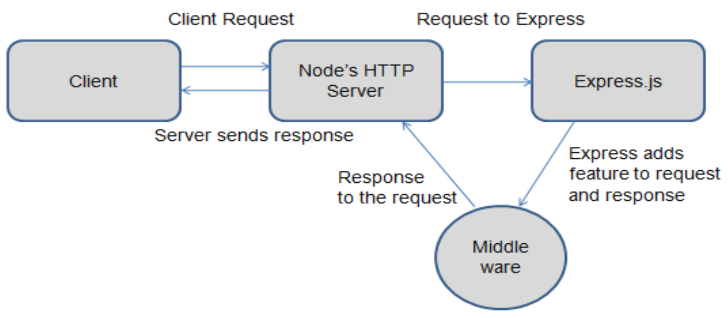
B. Express JS

Express is also free, open-source software, it can be classified as a web application framework for Node.js. To be more precise, Express JS is made for developing web apps and APIs.

Instead of manually writing full web server code in Node.js, developers use this MERN component to simplify the coding process. The best feature of this framework is that developers don't repeat the same code over and over, as they would with writing Node.js code in the HTTP module.

We utilized Express as it is a Node.js structure. While building the application we concentrated on that as opposed to making heaps of hub modules and composing the code with NodeJS, Express simplified it and more straightforward to compose the back-end code what's more carry out it in an organized arrangement. Express aided us in planning our web applications and APIs needed in our project as it upholds numerous middlewares which makes the code more limited and more straightforward to compose. Asynchronous programming furthermore Single-threaded architecture are the greatest benefits of utilizing Express in our application. For our application vigorous Programming interface Created another envelope to begin our express venture and the ventures for it are, we needed to add an order in the order brief to instate the package.json document. From that point forward, we had to acknowledge the default settings and proceed. npm init is the order to begin.

Fig 3. Express JS request flow



C. React JS
ReactJS is a JavaScript library used for building user interfaces. Originally created by a software engineer who worked for Facebook, React was later on open-sourced.
This specific library is often used for creating views rendered in HTML. The views that you create in Reach declarative, which means that you don't have to deploy additional time on managing the changes and effects they have on the data.

React uses a full-featured programming language (JavaScript) to construct repetitive or conditional DOM elements. With React, the same code can run on both the server and the browser.

React.JS is the front-end library of the JavaScript programming language. We utilized React.JS for building our client interface for the web application, as it is utilized for the development of a single page application since it can deliver powerfully changing information at a rapid. React permits engineers to code in JS and make User Interface parts.

We contemplated virtual DOM protests in React.JS, which we carried out in our task. Any progressions we made in our web based chatting application caused the whole User-Interface to re-render the virtual DOM. This permits us to analyze the potential distinction between the DOM Object and Virtual DOM. We utilized JSX, It made our code more straightforward and easier to write in React application. React.JS utilizes Components. Parts are the structure squares of User-Interface wherein every part had a rationale identified with our online chatting application and it contributed to the general User-Interface of our web application. Parts can be reused, and it helped our code for web application more straightforward to be perceived by different engineers and generally web application better at execution.

We began our React application by first introducing create react-application utilizing npm or yarn. npm introduce make react application worldwide OR yarn worldwide adds make react application are the two orders for utilizing npm or yarn individually. From that point onward, we made a new react application by utilizing create react application "our app name" Then, at that point, explore into the our application name organizer and type yarn start or on the other hand npm start to run your application.

D. Mongo DB

MongoDB is a free open-source, cross-platform document-oriented database program. It is classified as a No SQL database program, which means that data is stored in flexible documents with JSON-based query language. This also means that the size of the content number of fields in the documents tends to vary. The whole data is structured in a way to be prone to change over time.
MongoDB is known as a flexible solution that is always easy to scale.
We utilized MongoDB for our venture MongoDB is a data set where each record is a document design. In the background on the server, MongoDB changes over our JSON information into a parallel adaptation of it which is essentially put away and questioned all the more proficiently. MongoDB employments BSON to question information base. MongoDB stores BSON format both inside, and over the network, however that implies we can't consider MongoDB a JSON data set. we can address any information in JSON format which can be locally put away in MongoDB, and recovered simply in JSON format. As we contemplated and carried out MongoDB we can say that it is adaptable and permits its clients to make mapping, data sets, tables, and so forth Subsequent to introducing MongoDB we had a choice, of utilizing Mongo shell as it gives us a JavaScript interface through which the clients can associate and do any activities identifying with querying. MongoDB is a document-oriented database, so it is simple to list archives. also that is the explanation it handles response at a quicker pace.

MongoDB is Scalable In the MongoDB database, we handled large data by dividing it into a nested documented structure. MongoDB is a database server that allows us to run multiple database on it. Simple commands such as 'use' command is used to create a database. done using a use command: use database name; use database name; Creating a table: If the collection table doesn't exist then a new collection table will be created: db. create collection(collection name);.

The current work for the chatting web application development project is finished utilizing MERN stack advancements.

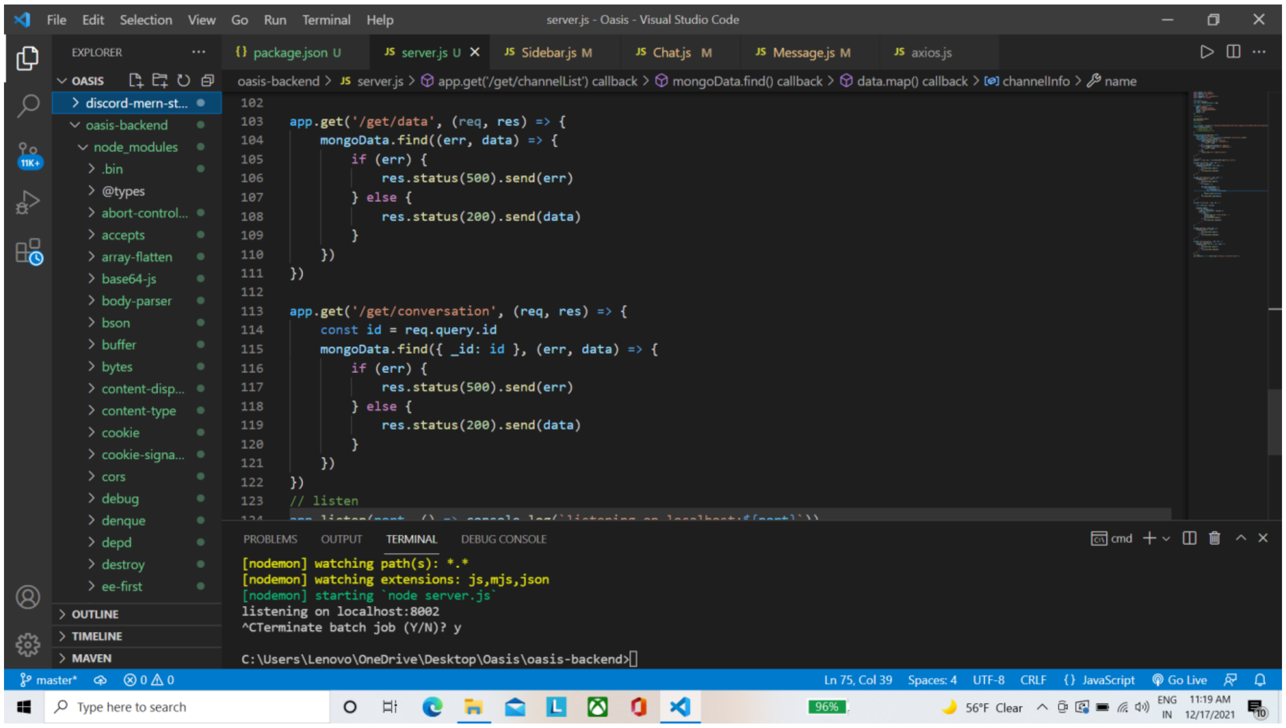
This venture plans to give a basic audit of the significant writing in the web based informing field and furthermore to portray key parts of the approach that we have applied all through the venture.

This venture figured out how to comprehend different issues that emerge while building a web application.

We Studied that web applications are not straightforward programming antiquities. Dominating the fundamental innovations/stack for establishing any web application is required. Zeroing in on other challenges that surface all through the development cycle. These are the first and generally essential steps that will guarantee that the final web application, will be created by the necessities of the market and will be modified to the needs of its customer. Further exploration what's more spotlight were given on programming devices for testing.

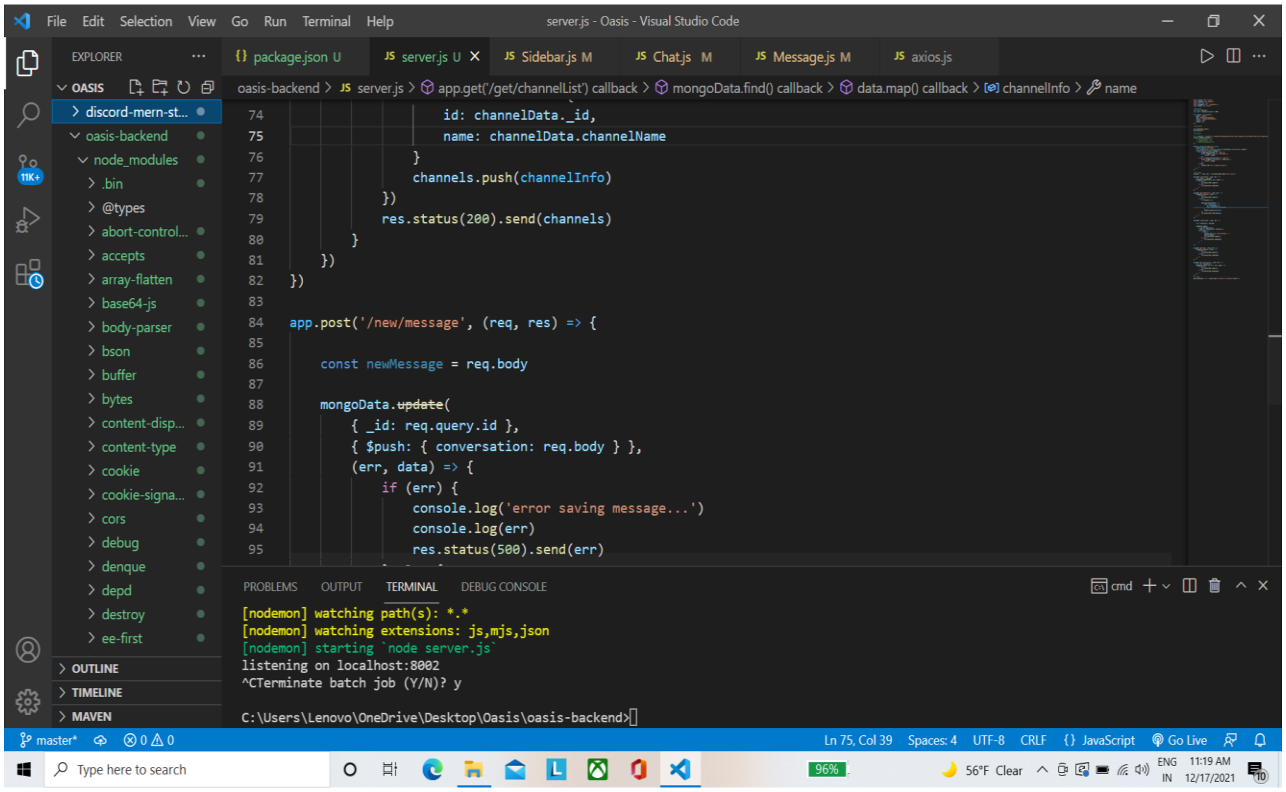
All the essential choices were made on how the site will be constructed relying upon the aftereffects of the issue examination stage since they assumed a significant part in portraying the particular client necessities for the web application.

ScreenShotsots


server.js - Oasis - Visual Studio Code
package.json U
server.js U
Sidebar.js M
Chat.js M
Message.js M
axios.js
oasis-backend > server.js > app.get('/get/channelList') callback > mongoData.find() callback > data.map() callback > channelInfo > name
master*
Ln 75, Col 39 Spaces: 4 UTF-8 CRLF JavaScript Go Live

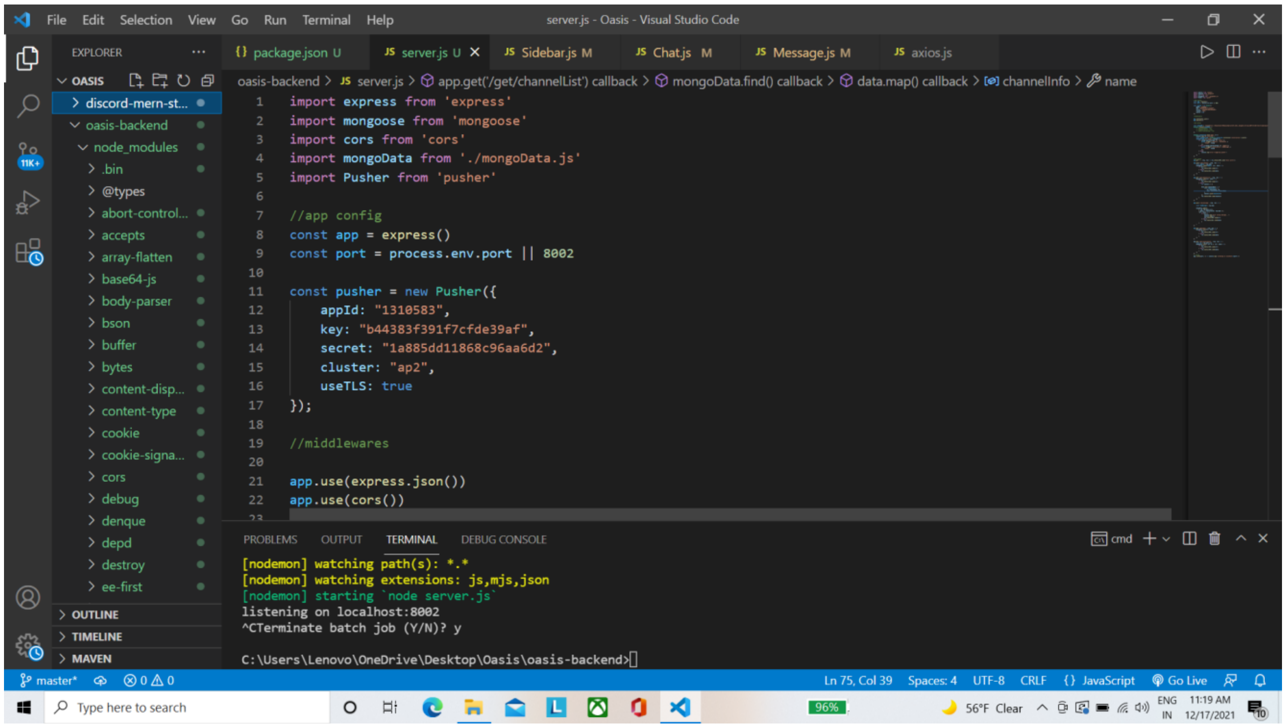
```
app.get('/get/data', (req, res) => {
    mongoData.find((err, data) => {
        if (err) {
            res.status(500).send(err)
        } else {
            res.status(200).send(data)
        }
    })
})

app.get('/get/conversation', (req, res) => {
    const id = req.query.id
    mongoData.find({ _id: id }, (err, data) => {
        if (err) {
            res.status(500).send(err)
        } else {
            res.status(200).send(data)
        }
    })
})
// listen
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```


server.js - Oasis - Visual Studio Code
package.json U
server.js U
Sidebar.js M
Chat.js M
Message.js M
axios.js
oasis-backend > server.js > app.get('/get/channelList') callback > mongoData.find() callback > data.map() callback > channelInfo > name
master*
Ln 75, Col 39 Spaces: 4 UTF-8 CRLF JavaScript Go Live

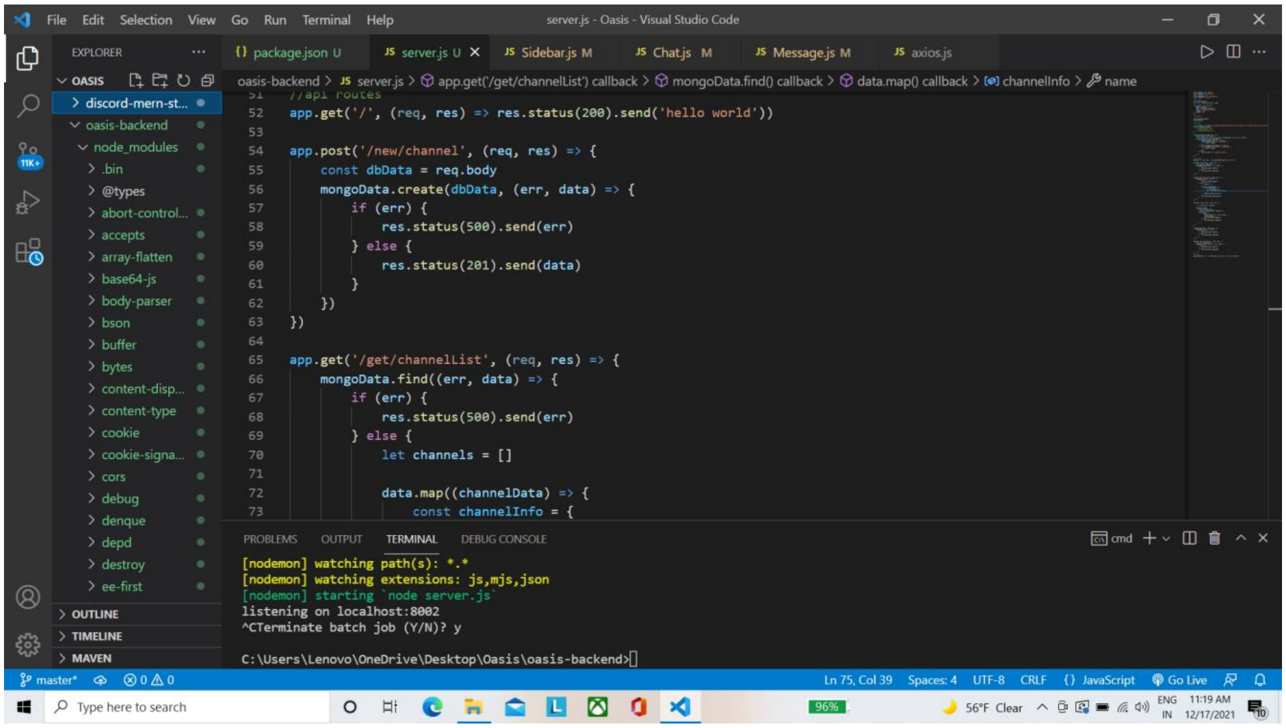
```
                    id: channelData._id,
                    name: channelData.channelName
                }
                channels.push(channelInfo)
            })
            res.status(200).send(channels)
        }
    })
})

app.post('/new/message', (req, res) => {

    const newMessage = req.body

    mongoData.~~update~~(
        { _id: req.query.id },
        { $push: { conversation: req.body } },
        (err, data) => {
            if (err) {
                console.log('error saving message...')
                console.log(err)
                res.status(500).send(err)
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```



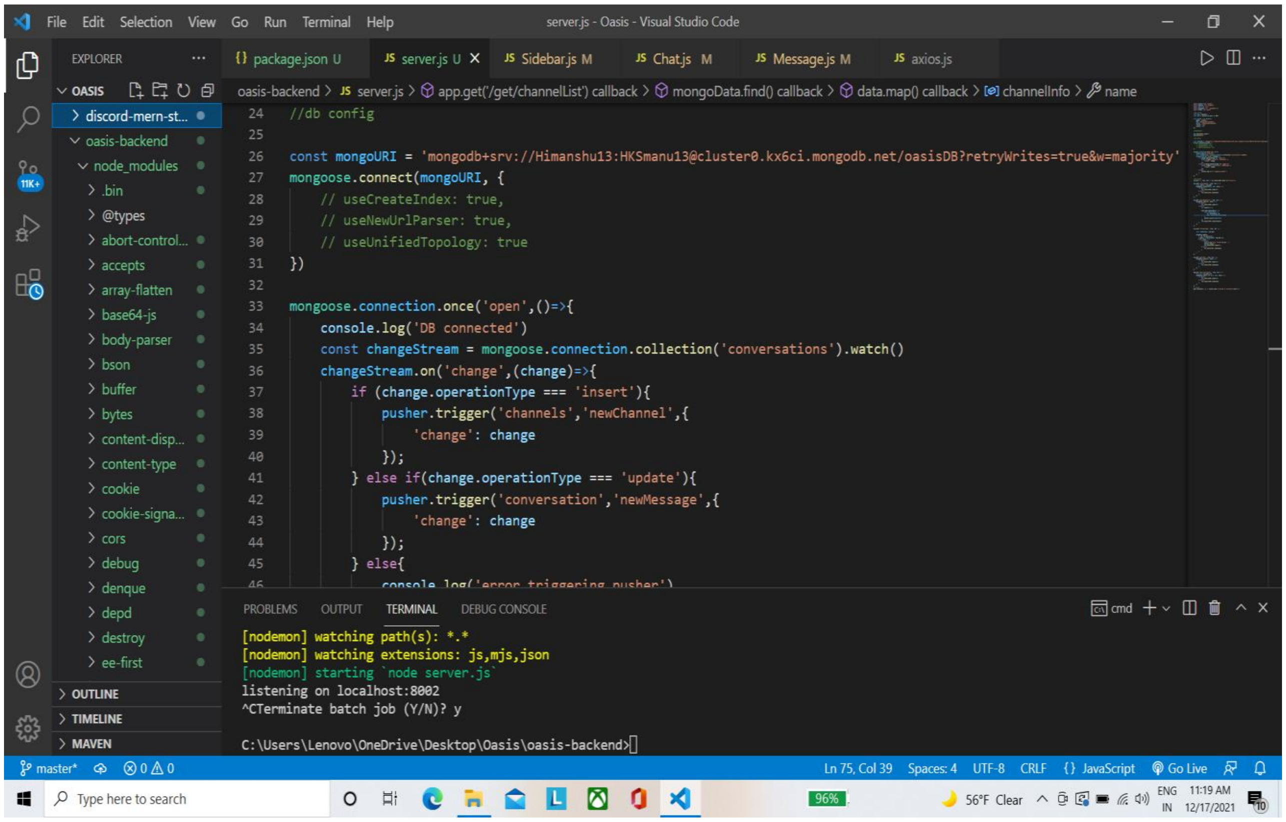
```
import express from 'express'
import mongoose from 'mongoose'
import cors from 'cors'
import mongoData from './mongoData.js'
import Pusher from 'pusher'

//app config
const app = express()
const port = process.env.port || 8002

const pusher = new Pusher({
    appId: "1310583",
    key: "b44383f391f7cfde39af",
    secret: "1a885dd11868c96aa6d2",
    cluster: "ap2",
    useTLS: true
});

//middlewares

app.use(express.json())
app.use(cors())
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```



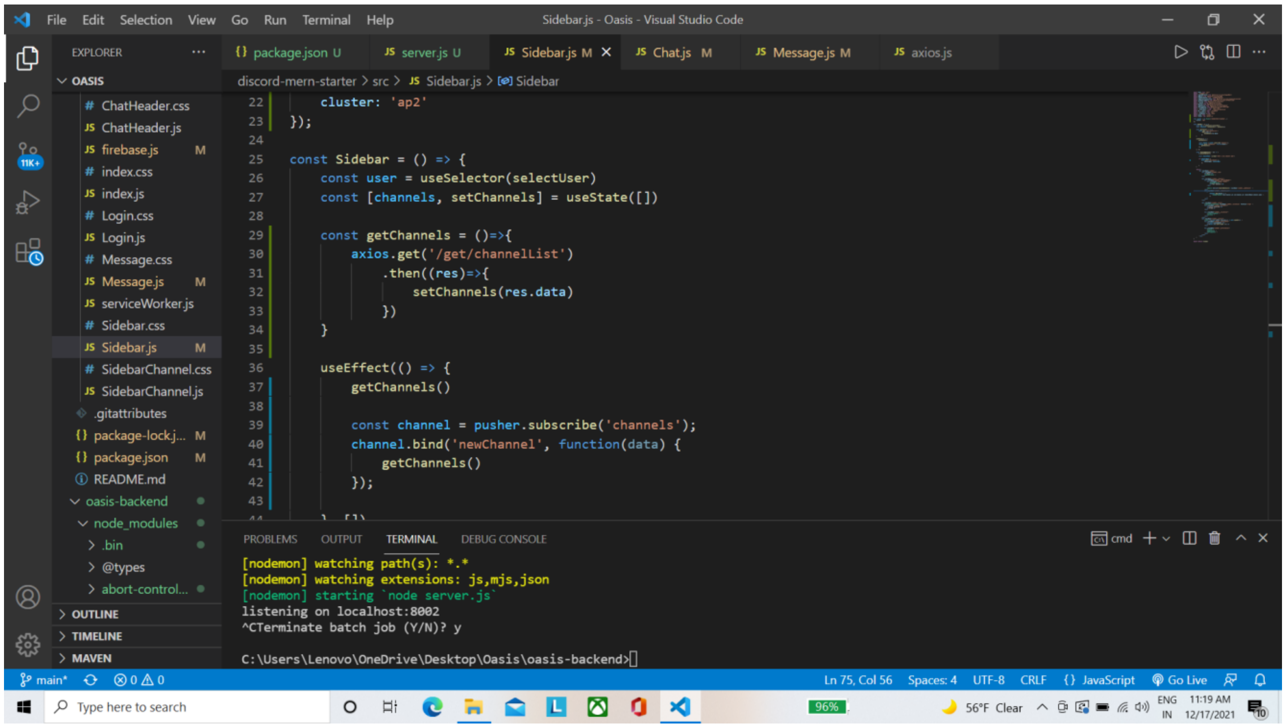
```
//api routes
app.get('/', (req, res) => res.status(200).send('hello world'))

app.post('/new/channel', (req, res) => {
    const dbData = req.body
    mongoData.create(dbData, (err, data) => {
        if (err) {
            res.status(500).send(err)
        } else {
            res.status(201).send(data)
        }
    })
})

app.get('/get/channelList', (req, res) => {
    mongoData.find((err, data) => {
        if (err) {
            res.status(500).send(err)
        } else {
            let channels = []

            data.map((channelData) => {
                const channelInfo = {
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```



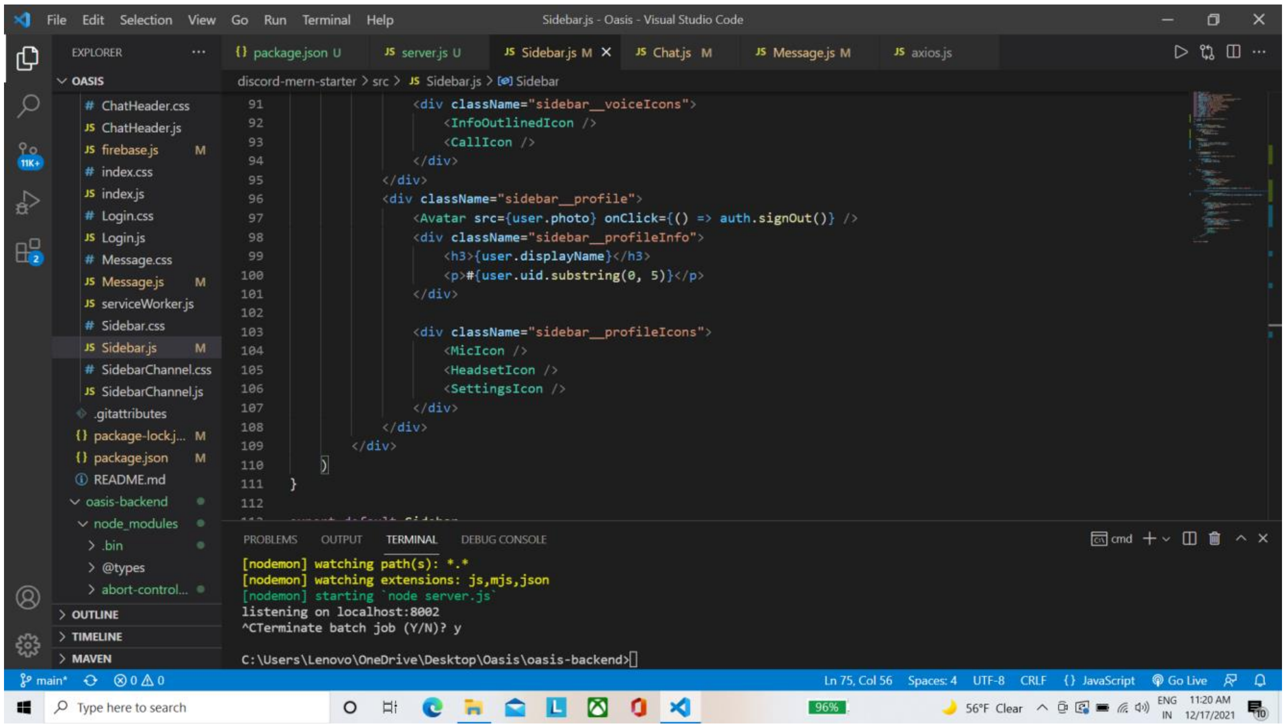
```
//db config

const mongoURI = 'mongodb+srv://Himanshu13:HKSmanu13@cluster0.kx6ci.mongodb.net/oasisDB?retryWrites=true&w=majority'
mongoose.connect(mongoURI, {
    // useCreateIndex: true,
    // useNewUrlParser: true,
    // useUnifiedTopology: true
})

mongoose.connection.once('open',()=>{
    console.log('DB connected')
    const changeStream = mongoose.connection.collection('conversations').watch()
    changeStream.on('change',(change)=>{
        if (change.operationType === 'insert'){
            pusher.trigger('channels','newChannel',{
                'change': change
            });
        } else if(change.operationType === 'update'){
            pusher.trigger('conversation','newMessage',{
                'change': change
            });
        } else{
            console.log('error triggering pusher')
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```



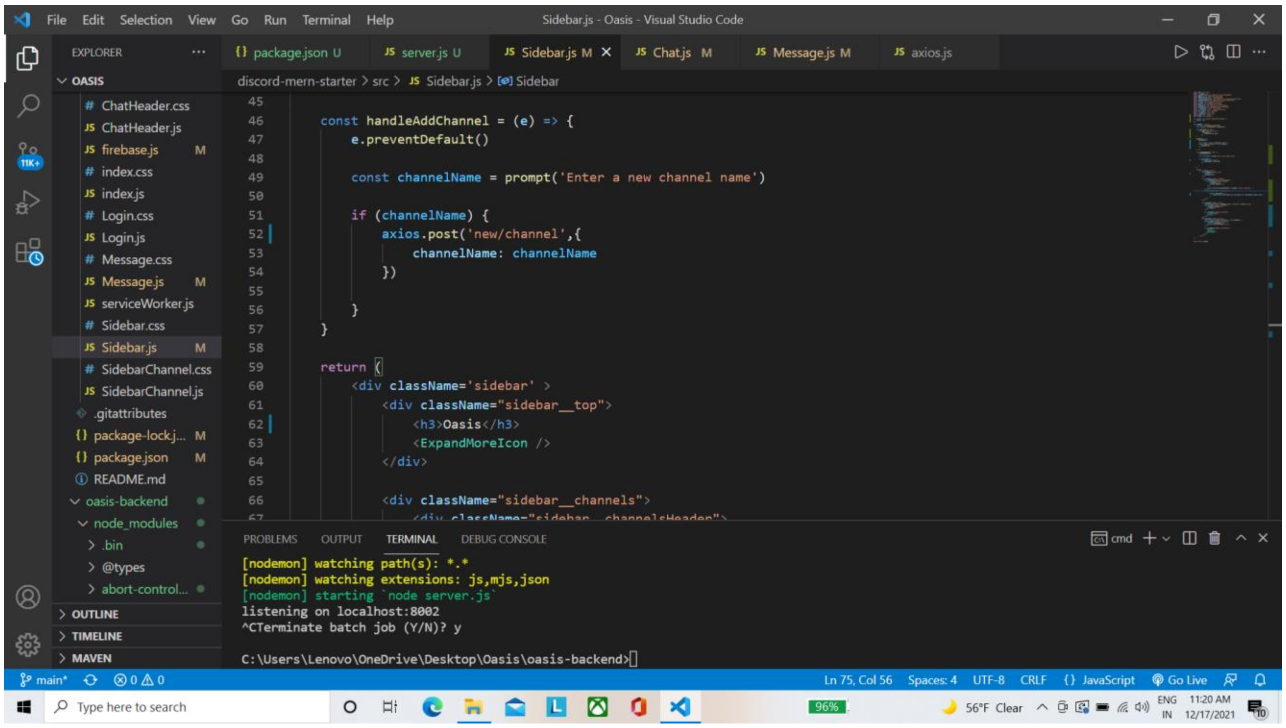
```
    cluster: 'ap2'
});

const Sidebar = () => {
    const user = useSelector(selectUser)
    const [channels, setChannels] = useState([])

    const getChannels = ()=>{
        axios.get('/get/channelList')
            .then((res)=>{
                setChannels(res.data)
            })
    }

    useEffect(() => {
        getChannels()

        const channel = pusher.subscribe('channels');
        channel.bind('newChannel', function(data) {
            getChannels()
        });
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```



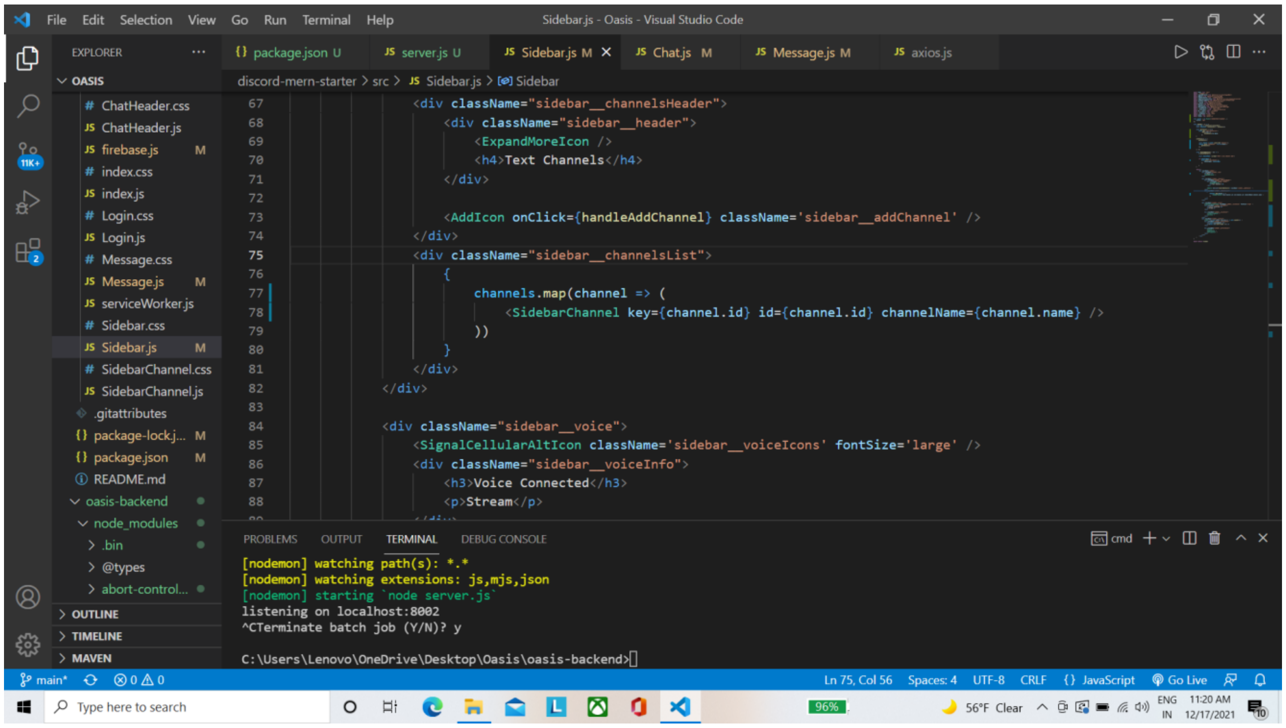
```
                <div className="sidebar__voiceIcons">
                    <InfoOutlinedIcon />
                    <CallIcon />
                </div>
            </div>
            <div className="sidebar__profile">
                <Avatar src={user.photo} onClick={() => auth.signOut()} />
                <div className="sidebar__profileInfo">
                    <h3>{user.displayName}</h3>
                    <p>#{user.uid.substring(0, 5)}</p>
                </div>

                <div className="sidebar__profileIcons">
                    <MicIcon />
                    <HeadsetIcon />
                    <SettingsIcon />
                </div>
            </div>
        </div>
    )
}

```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```



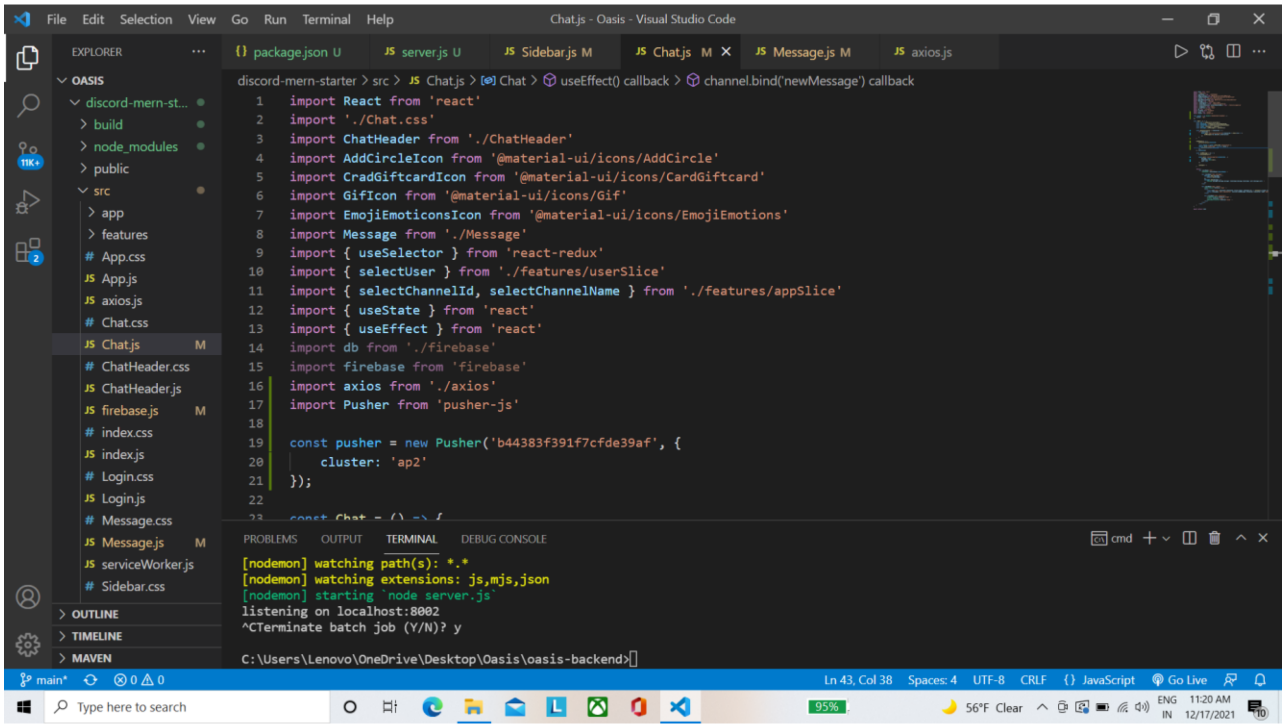
```

    const handleAddChannel = (e) => {
        e.preventDefault()

        const channelName = prompt('Enter a new channel name')

        if (channelName) {
            axios.post('new/channel',{
                channelName: channelName
            })

        }
    }

    return (
        <div className='sidebar' >
            <div className="sidebar__top">
                <h3>Oasis</h3>
                <ExpandMoreIcon />
            </div>

            <div className="sidebar__channels">
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```



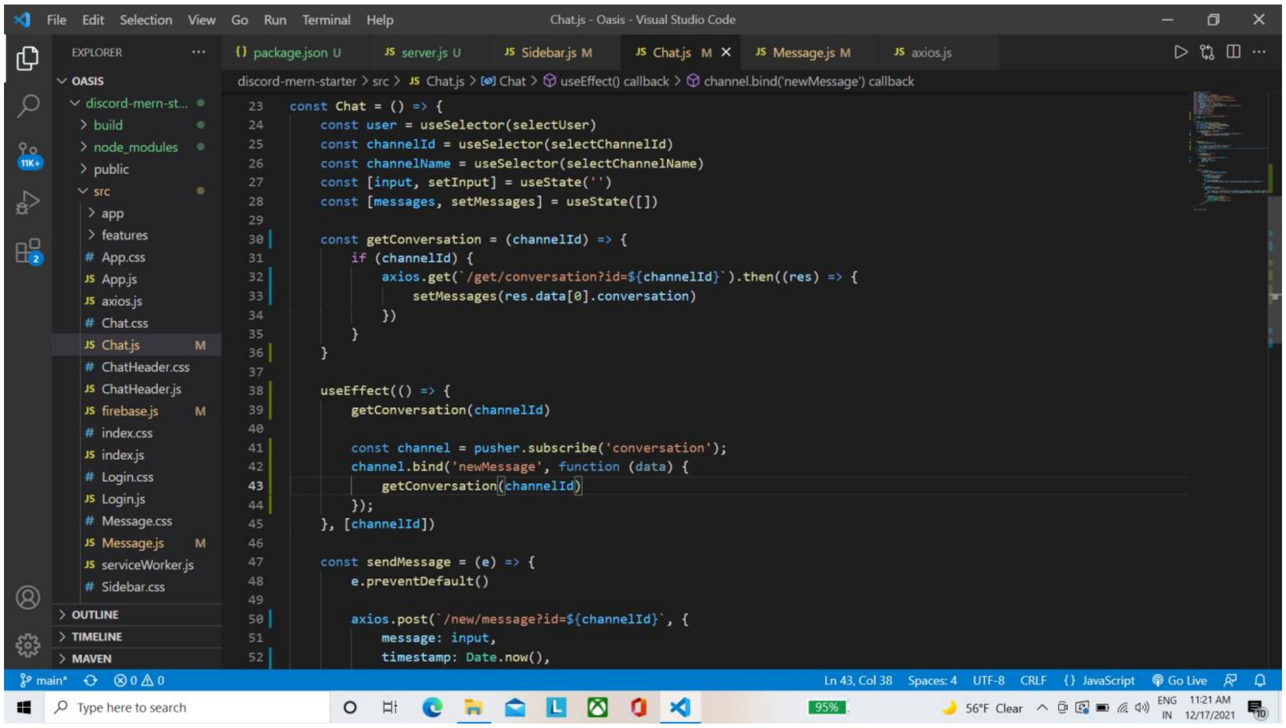
```
            <div className="sidebar__channelsHeader">
                <div className="sidebar__header">
                    <ExpandMoreIcon />
                    <h4>Text Channels</h4>
                </div>

                <AddIcon onClick={handleAddChannel} className='sidebar__addChannel' />
            </div>
            <div className="sidebar__channelsList">
                {
                    channels.map(channel => (
                        <SidebarChannel key={channel.id} id={channel.id} channelName={channel.name} />
                    ))
                }
            </div>
        </div>

        <div className="sidebar__voice">
            <SignalCellularAltIcon className='sidebar__voiceIcons' fontSize='large' />
            <div className="sidebar__voiceInfo">
                <h3>Voice Connected</h3>
                <p>Stream</p>
```

```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```

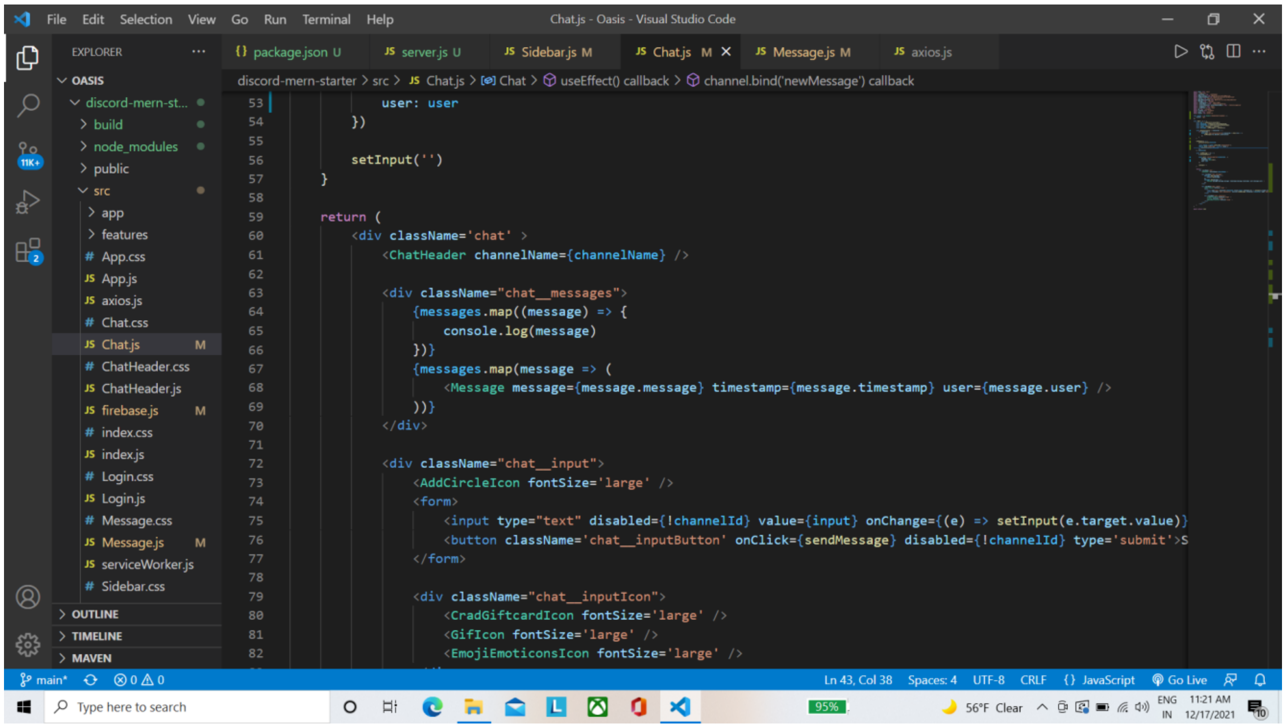


```
import React from 'react'
import './Chat.css'
import ChatHeader from './ChatHeader'
import AddCircleIcon from '@material-ui/icons/AddCircle'
import CradGiftcardIcon from '@material-ui/icons/CardGiftcard'
import GifIcon from '@material-ui/icons/Gif'
import EmojiEmoticonsIcon from '@material-ui/icons/EmojiEmotions'
import Message from './Message'
import { useSelector } from 'react-redux'
import { selectUser } from './features/userSlice'
import { selectChannelId, selectChannelName } from './features/appSlice'
import { useState } from 'react'
import { useEffect } from 'react'
import db from './firebase'
import firebase from 'firebase'
import axios from './axios'
import Pusher from 'pusher-js'

const pusher = new Pusher('b44383f391f7cfde39af', {
    cluster: 'ap2'
});
```

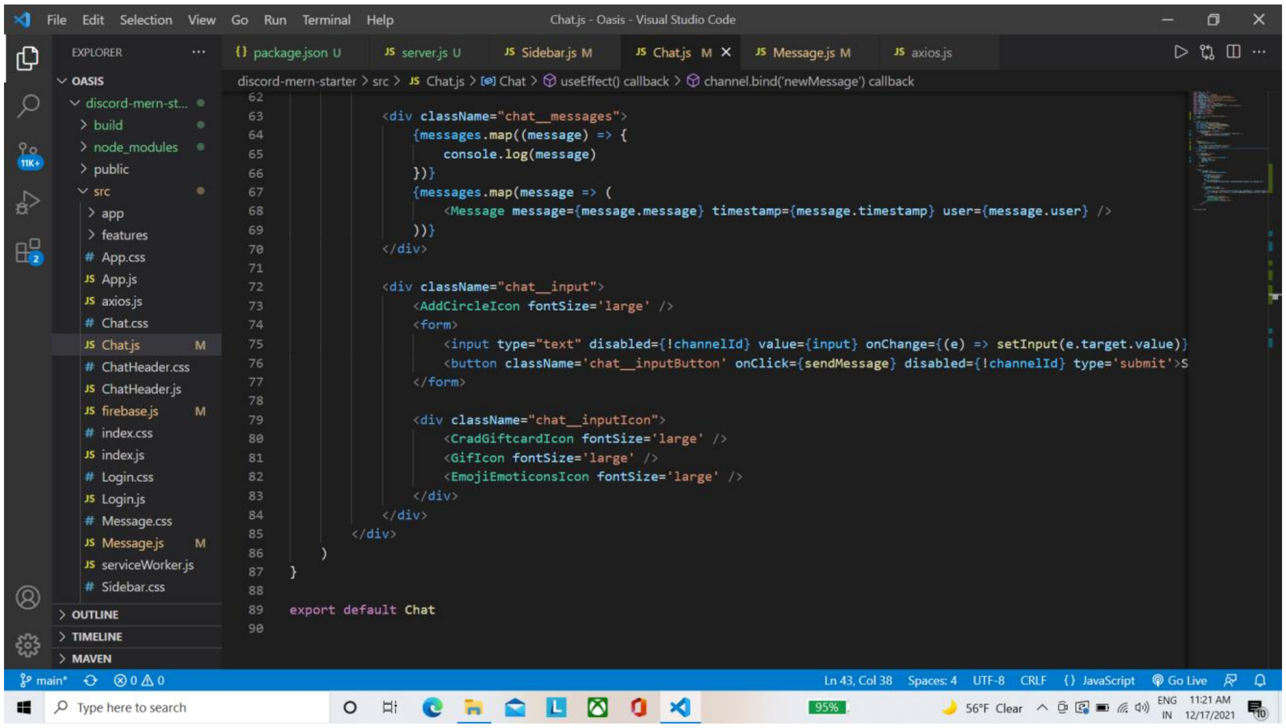
```
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on localhost:8002
^CTerminate batch job (Y/N)? y

C:\Users\Lenovo\OneDrive\Desktop\Oasis\oasis-backend>
```



```
const Chat = () => {
    const user = useSelector(selectUser)
    const channelId = useSelector(selectChannelId)
    const channelName = useSelector(selectChannelName)
    const [input, setInput] = useState('')
    const [messages, setMessages] = useState([])

    const getConversation = (channelId) => {
        if (channelId) {
            axios.get(`/get/conversation?id=${channelId}`).then((res) => {
                setMessages(res.data[0].conversation)
            })
        }
    }

    useEffect(() => {
        getConversation(channelId)

        const channel = pusher.subscribe('conversation');
        channel.bind('newMessage', function (data) {
            getConversation(channelId)
        });
    }, [channelId])

    const sendMessage = (e) => {
        e.preventDefault()

        axios.post(`/new/message?id=${channelId}`, {
            message: input,
            timestamp: Date.now(),
```

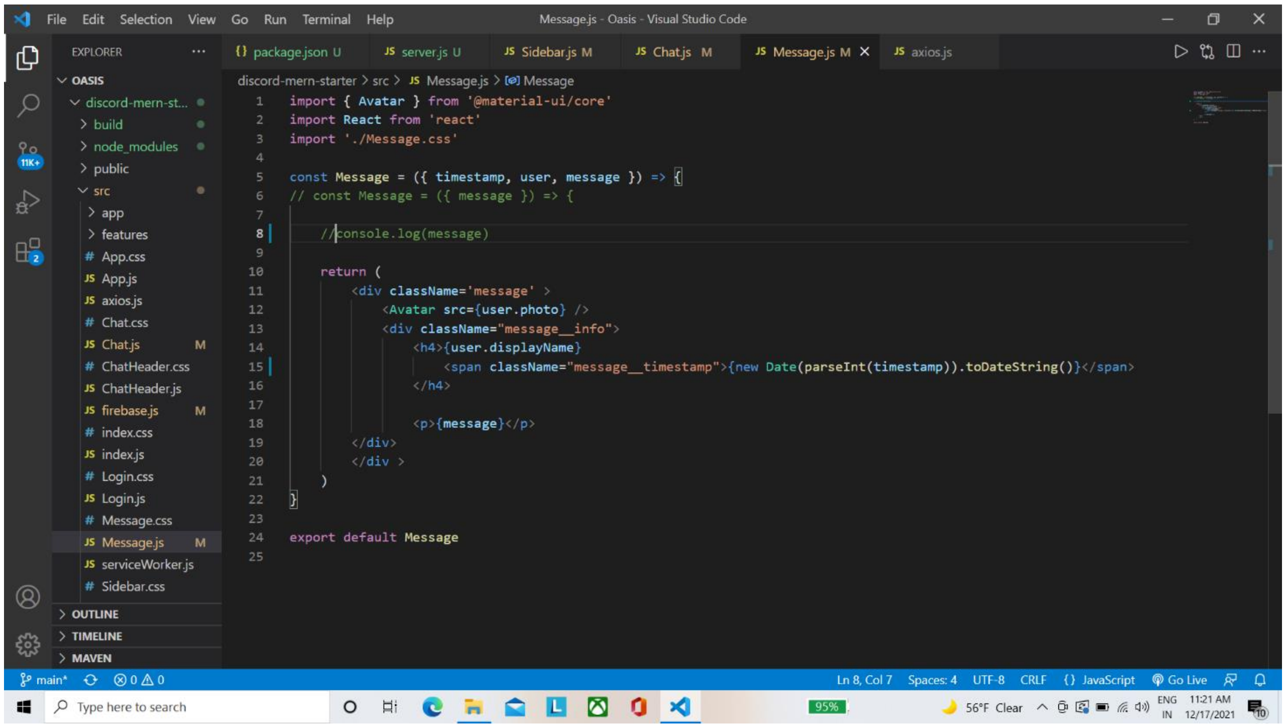


```
            user: user
        })

        setInput('')
    }

    return (
        <div className='chat' >
            <ChatHeader channelName={channelName} />

            <div className="chat__messages">
                {messages.map((message) => {
                    console.log(message)
                })}
                {messages.map(message => (
                    <Message message={message.message} timestamp={message.timestamp} user={message.user} />
                ))}
            </div>

            <div className="chat__input">
                <AddCircleIcon fontSize='large' />
                <form>
                    <input type="text" disabled={!channelId} value={input} onChange={(e) => setInput(e.target.value)}
                    <button className='chat__inputButton' onClick={sendMessage} disabled={!channelId} type='submit'>S
                </form>

                <div className="chat__inputIcon">
                    <CradGiftcardIcon fontSize='large' />
                    <GifIcon fontSize='large' />
                    <EmojiEmoticonsIcon fontSize='large' />
```



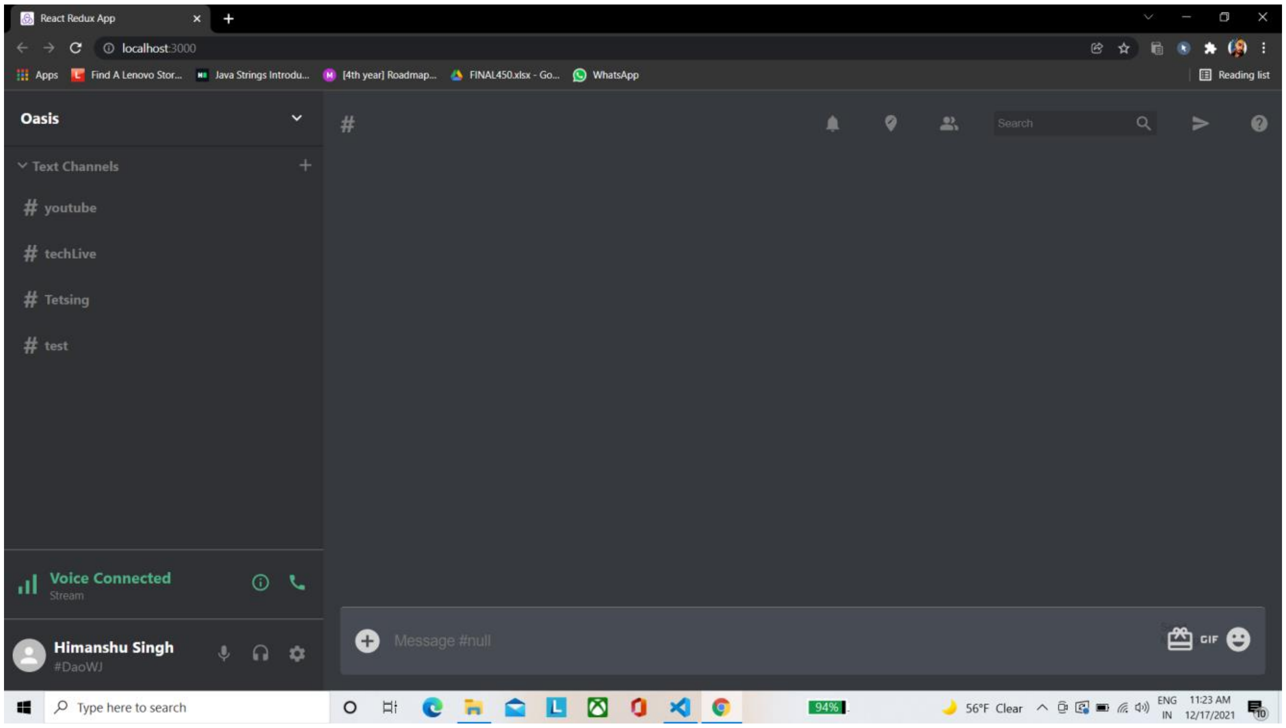
discord-mern-starter > src > Chat.js > Chat > useEffect() callback > channel.bind('newMessage') callback

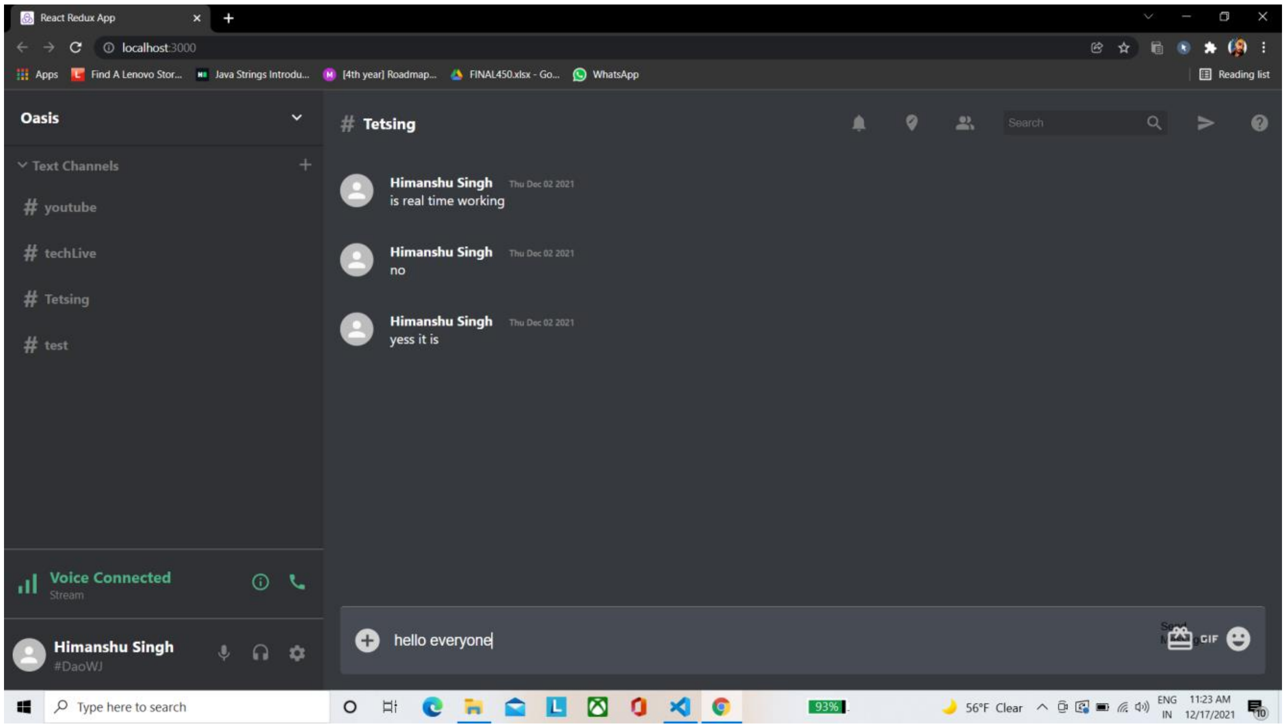
```
            <div className="chat__messages">
                {messages.map((message) => {
                    console.log(message)
                })}
                {messages.map(message => (
                    <Message message={message.message} timestamp={message.timestamp} user={message.user} />
                ))}
            </div>

            <div className="chat__input">
                <AddCircleIcon fontSize='large' />
                <form>
                    <input type="text" disabled={!channelId} value={input} onChange={(e) => setInput(e.target.value)}
                    <button className='chat__inputButton' onClick={sendMessage} disabled={!channelId} type='submit'>S
                </form>

                <div className="chat__inputIcon">
                    <CradGiftcardIcon fontSize='large' />
                    <GifIcon fontSize='large' />
                    <EmojiEmoticonsIcon fontSize='large' />
                </div>
            </div>
        </div>
    )
}

export default Chat
```



discord-mern-starter > src > Message.js > Message

```
import { Avatar } from '@material-ui/core'
import React from 'react'
import './Message.css'

const Message = ({ timestamp, user, message }) => {
// const Message = ({ message }) => {

    //console.log(message)

    return (
        <div className='message' >
            <Avatar src={user.photo} />
            <div className="message__info">
                <h4>{user.displayName}
                    <span className="message__timestamp">{new Date(parseInt(timestamp)).toDateString()}</span>
                </h4>

                <p>{message}</p>
        </div>
        </div >
    )
}

export default Message
```





Himanshu Singh Thu Dec 02 2021
is real time working

Himanshu Singh Thu Dec 02 2021
no

Himanshu Singh Thu Dec 02 2021
yess it is

hello everyone

Proposed architecture

Secure Mobile Chat Requirements
In this section, we propose a set of requirements to make

Secure chat application:
req1: Password stored on the chat server should be encrypted.
req2: Providing either secure session or TLS. Secure session is a unique key for each session. Ensures that communication is with the right person and no man-in-the-middle can read the messages.
req3: Messages must be encrypted to maintain security and privacy.
req4: Local storage must be protected by encryption.
req5: Messages are not stored on the chat server but stored on the user's device.
req6: It is not allowed to exchange messages if they are not friends.

Proposed Architecture

The proposed architecture is designed to be Client-Server chat application. In client side, when a user sets up the application, the user either selects registration or log-in. In server side, the chat server consists of users' server and a message server. User's server that manages user's credentials. Message server handles messages between users by using Firebase Cloud Messaging (FCM).If the recipient is offline, the messages will be stored temporarily on the FCM queue for a specific period of time, and when recipient becomes online these messages are forwarded to him then deleted from the queue.

Registration an account

Before starting the application, there must have a lock screen to configure the Keystore that provides a secure container to store the local storage key to make more difficult for extraction it from the device by unauthorized persons or other applications . Each account has only one device and it is distinguished by device id. In addition, Email and username are unique. Name, email and password are required to register a new account.

After typing the registration information, the password is encrypted by using XSalsa20 algorithm then the user credentials are sent to the server. After verification, the server generates a unique identifier that acts as the user ID. After that, the acknowledgement message is received for successful registration to the client application and the client information is stored in local storage.

The application generates a set of keys:
(a) Key for encrypting the password.
(b) A public key pair for calculating session key.
(c) Symmetric storage key for encrypting/decrypting local storage contains contact list, chat history and key store.

Login

Email and password are required for user authentication. After typing the authentication information, the password is encrypted then the user credentials are sent to the server. The server checks if the email and password are valid. After validation, JSON Web Token (JWT)is created and sent to the client to store it. When a client makes a request at the later time, JWT is passed with the request.

This saves a lot of users' battery by avoiding requesting to the server for new messages. It provides TLS for securing channel. At the beginning of running the application for the first time gets the following:

(1) The application connects to the FCM server and registers itself.
(2) When successful registration, FCM provides a registration token to the device. This registration token uniquely identifies each device.
(3) The application sends the registration token to the server to store it in the MongoDB database.

Session key Setup

To add users to contact list either by username or by email address. For sending a request to a friend on the assumption that the first user knows the username or email of the second user due to the username and email are a unique for each user and the second user should have already registered in the server. Presumably, the first user is called Alice and the second is called Bob.

When the send request, Bob name is typed by Alice and her public key is fetched from the local storage then the request is sent to the server. When a request is received, it appears as a notification .If the friendship request is accepted by Bob, his Private the key is fetched with Alice's public key to calculate the session key by using Elliptic Curve Diffie-Hellman

Exchanging Messages

When a message is typed, the application encrypts the message using the XSalsa20 encryption algorithm to encrypt the message body and Poly1305 to compute a Message Authentication Code (MAC).

Each message has its own separate key and nonce which brings better security for every single message in such discovering one of the keys cannot decrypt previous messages. After encrypting the message, it is encrypted again using the recipient's session key then it is sent to the server.

After the message is received from FCM, the MAC of the encrypted message is calculated and compares it with the received MAC to verify the integrity of the message. If the results are not the same, it is rejected and does not show to the user otherwise it is decrypted by the sender session key. Next, the message body is verified in the same steps above. Now the key and nonce to decrypt the message are known. The message is then decrypted and stored in the local storage and displayed to the recipient.

Server-Side Implementation

Server-side has relied on Node JS and Mongo DB database. Node JS is fast, capable of handling a large number of simultaneous connections with high throughput, which is equivalent to high scalability. Mongo DB and Node JS have often used together because of their using

MESSAGING

A chat application should allow both sending and receiving process in simultaneous way. This is achieved in this application with java multithreading concept. GROUP CchatsAnother important feature in chat application is group chat which is implemented in this application. It allows people to chat. Message will be sent to all the users in chat room along with the name of the user who has sent the message Users who are available in the chat room will receive the message. VII.

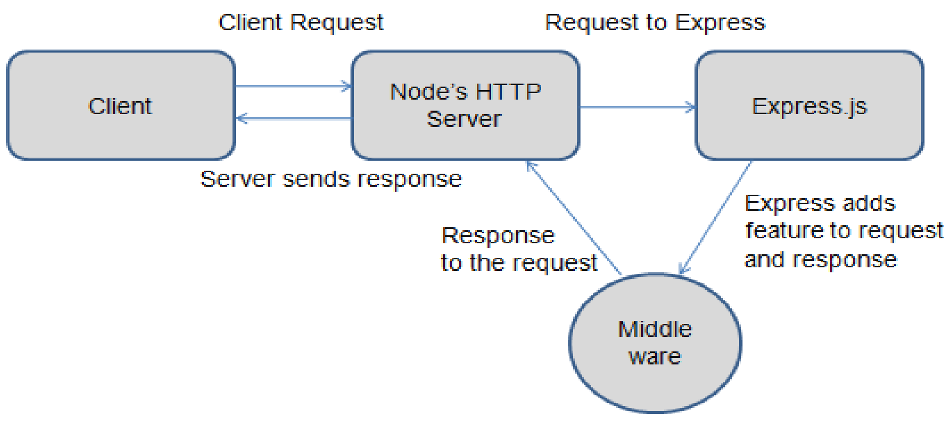
ARCHITECTURE

This application has been implemented based on client/server model.

A. SERVER

A server may be a computer dedicated to running a server application.

Organizations have dedicated computer for server application which has to be maintained periodically and has to be monitored continuously for traffic loads would never let them go down which affects the company's revenue. Most organizations have a separate monitoring system to keep an eye over their server so that they can find their server downtime before its clients. These server computers accept clients over network connections that are requested. The server responds back by sending responses being requested.



There are many different server applications that vary based on their dedicated work. Some are involved for accepting requests and performing all dedicated works like business application servers while others are just to bypass the request like a proxy server.

These server computers must have a faster Central processing unit, faster and more plentiful RAM, and bigger hard disc drive. More obvious distinctions include redundancy in power supplies, network connections, and RAID also as Modular design.

B. CLIENT

A client is a software application code or a system that requests another application that is running on dedicated machine called Server. These clients need not be connected to the server through wired communication. Wireless communication takes place in this process. Client with a network connection can send a request to the server.

CHAPTER 3

SYSTEM REQUIREMENTS

It is basically build using Mern stack for real time messaging. It uses react as front end and node.js as backend and it is using mongo DB as database. Mern stack is a mixture of technologies used to create web application. Any web application. Any web application will be made utilizing various technologies like frame work, libraries and database.

Mern stack includes the following open source components

Mongo DB

Express JS

React JS/redux

Node Js

Benefits of a Mern stack-

Java script is everywhere. It is used both on the front end and back end side. Because of this there is no need for context switching

Mern is java script based dev. only need to master a single coding language which makes things a million time easier.

Building dynamic web application in no time

Mern stack for web application-

Efficiency- Highly efficient code with less time can be developed

Testing-It is crucial and the first string for the software development process.

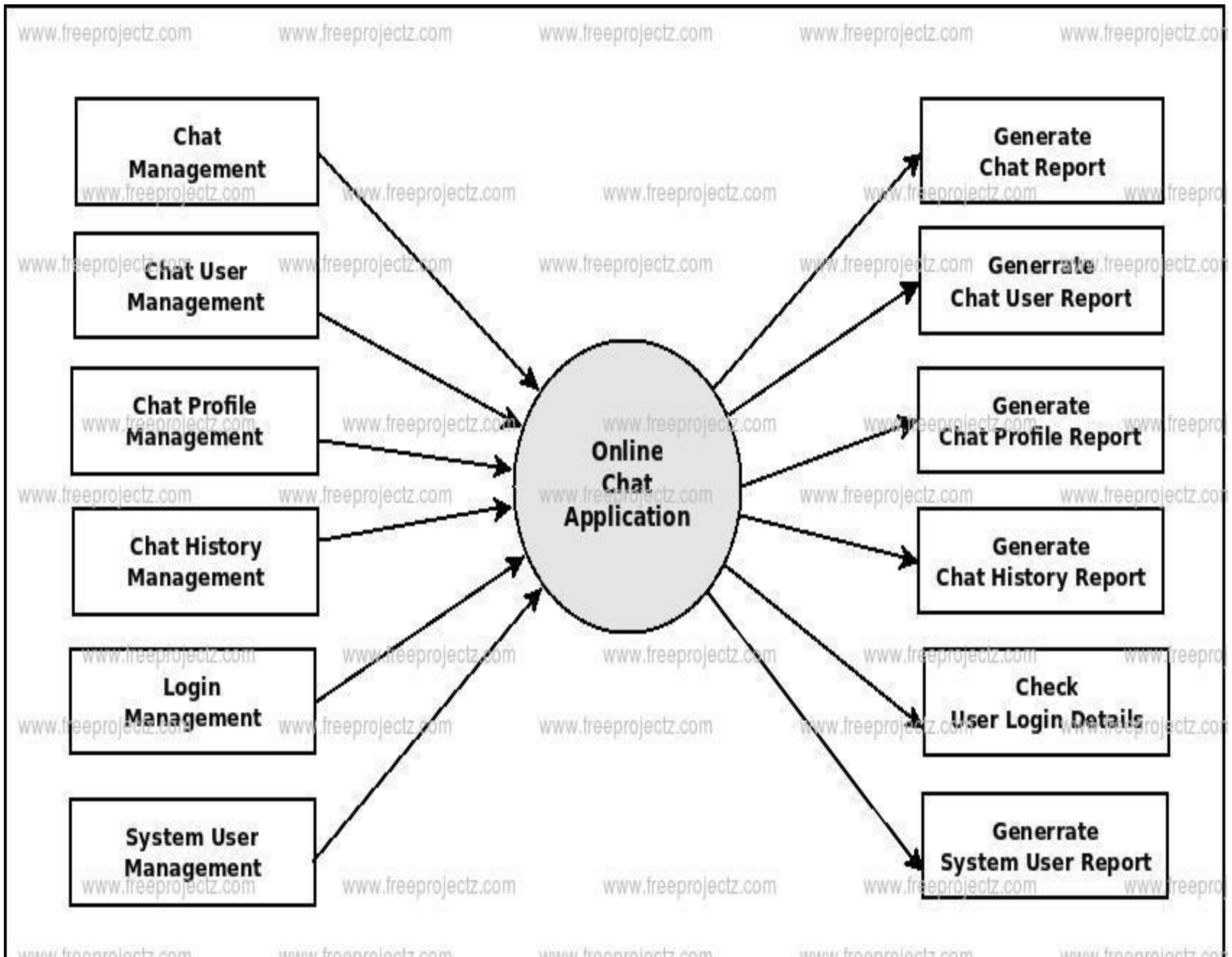
Implementation-There is a common html like hybrid code formatting in react.js is called JSX

Required Tools

Software and Hardware Requirements:
Software Requirements:

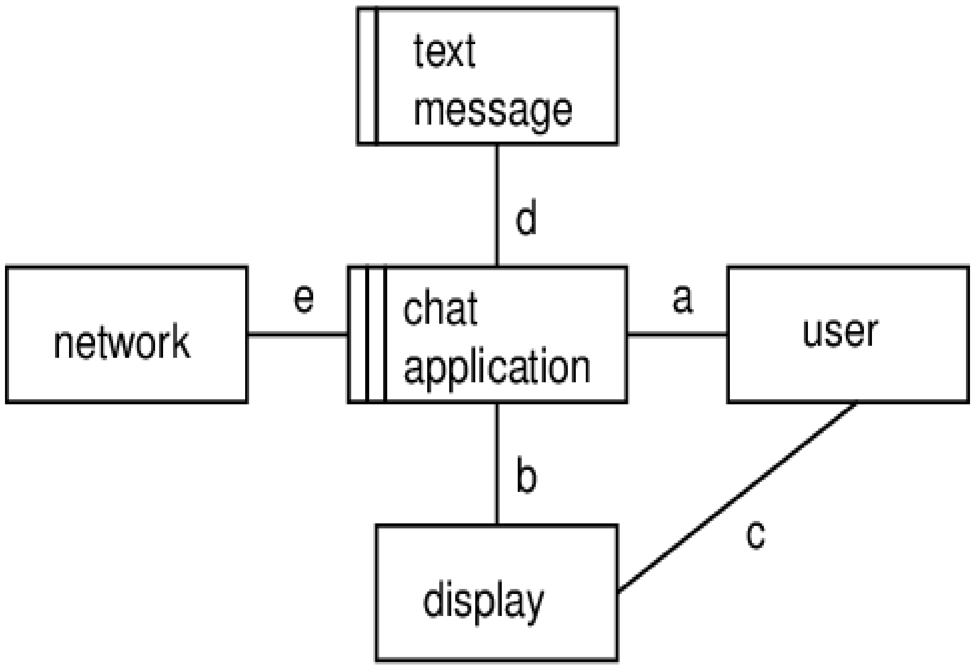
- Python3.5+
- Visual Studio Code,
- Chrome
- Java
- Android studio
- Firebase

Hardware Requirements:
● OS:Windows / Linux/ Mac
● Processor: intel i5
● RAM: 4GB
● ROM: 500GB
● Graphics Card: Good But Not Necessary



First Level DFD - Online Chat Application

Context Digram



Design is the universal language in the visual world, and web design refers to the user interface of the web page. The main purpose and goal of the design are to put content into focus, so users easily reach and use web content . Because of various technological changes and trends, web design has changed a lot, from first web generation web page which showed contents using a simple textual web page.

Chapter 4
Results and Discussion

It may help collecting perfect management in detail. In very short time, the collection will obvious simple and sensible. It will help a person to know the management of passes year perfectly and vividly.

It also helps in current all works relative to online chat application. It will be also reduced the cost of collecting the management &collection procedure will go on smoothly

The currently used online chatting web application systems have following shortcomings:

1) We can add printer in future.

2) We can give more advance software for online chat application including more

   facilities.

3) We will host the platform on online servers to make it accessible worldwide.

4) Integrate multiple load balancers to distribute the load of system

5) Create the master and slave database structure to reduce the overload of the database

   queries.

6) Implement the backup mechanism for taking backup of codebase and database on

   regular basis on different servers

The above-mentioned points are the enhancements that and be done to increase the applicability and usage of this project.Here we can maintain the records of chat profile and char user .Enhancements can be done to maintain all the chat profile, chat user, chat history , group chat, smiles chat.

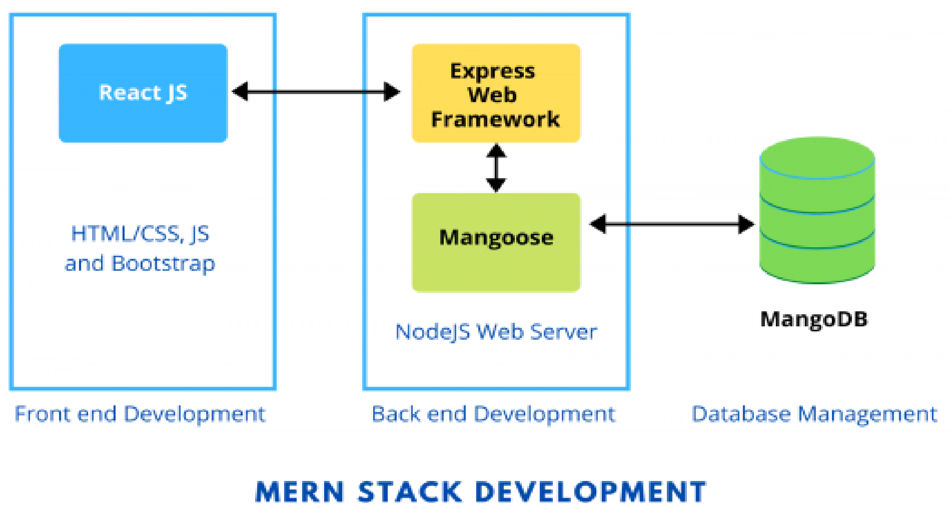
The current work for the chatting web application development project is finished utilizing MERN stack advancements. This venture plans to give a basic audit of the significant writing in the web based informing field and furthermore to portray key parts of the approach that we have applied all through the venture. This venture figured out how to comprehend different issues that emerge while building a web application.

We Studied that web applications are not straightforward programming antiquities. Dominating the fundamental innovations/stack for establishing any web application is required. Zeroing in on other challenges that surface all through the development cycle. These are the first and generally essential steps that will guarantee that the final web application, will be created by the necessities of the market and will be modified to the needs of its customer.

Further exploration of what's more spotlight were given on programming devices for testing. All the essential choices were made on how the site will be constructed relying upon the aftereffects of the issue examination stage since they assumed a significant part in portraying the particular client necessities for the web application.

As this world is into the internet and nothing happens without it. This application will have a huge impact. It can reach people. Private organizations like IT parks, Colleges, Institutions prefer to have separate chat applications over a public ones. Hence this application can be implemented over there. Thus, this application has a huge impact on people, mostly in private networks. This provides good scope for developing a better application with additional features than other traditional ones in the world.

Institutions prefer to have separate chat applications over a public ones. Hence this application can be implemented over there. Thus, this application has a huge impact on people, mostly in private networks. This provides good scope for developing a better application with additional features than other traditional ones in the world.



Future scope

The current work for the chatting web application development project is finished utilizing MERN stack advancements. This venture plans to give a basic audit of the significant writing in the web based informing field and furthermore to portray key parts of the approach that we have applied all through the venture. This venture figured out how to comprehend different issues that emerge while building a web application.

We Studied that web applications are not straightforward programming antiquities. Dominating the fundamental innovations/stack for establishing any web application is required. Zeroing in on other challenges that surface all through the development cycle.

Chatting is a very commonly used application among users. General users use instant messaging services to communicate with other individual users. In our project, we have provided many enhanced features for a chat application. The features like painting along with chat would be fun to use and interactive feature for a general user.

For professional users it would be very useful for communicating important flowcharts, diagrammatic representation of some problem, making important symbols, etc.

It opens up a wide variety of uses for individuals. The predictive texting feature would help a user to chat easily. Various figures of various formats could be opened and sent to another user. It would also give freedom of using any tool for drawing. The chat application is so aimed that the people could have a better experience of chatting.

It has the potential to attract more and more users to interact and connec t. These are the first and generally essential steps that will guarantee that the final web application, will be created by the necessities of the market and will be modified to the needs of its customer. Further exploration of what's more spotlight was given on programming devices for testing. All the essential choices were made on how the site will be constructed relying upon the aftereffects of the issue examination stage since they assumed a significant part in portraying the particular client necessities for the web application.

References

[1] M. R. Solanki, A. Dongaonkar, A adventure of human comfort: web1.zero to net 4.0, global journal of research and clinical Innovation(IJRSI), quantity III, difficulty IX, pp. seventy five-78, 2016

[2] Javeed, A. (2019). overall performance Optimization techniques for ReactJS. 2019

[3] J. M. Spool, content and design are inseparable paintings companions, 2014. Retrieved September 29, 2017, fromhttps://articles.uie.com/ content and design

[4] Bozikovic, H., Stula, M. (2018). internet design beyond, present and future. 2018 41st global conference on information and communiquegeneration, Electronics and Microelectronics (MIPRO).

[5] Carter, B. (2014). HTML structure, a novel development system(arms): An technique for internet improvement. 2014

[6] Sterling, A. (2019). NodeJS and Angular equipment for JSON-LD. 2019 IEEE 13th

[7] Laksono, D. (2018). trying out Spatial facts Deliverance in sq. and NoSQL Database the use of NodeJS Fullstack internet App. 2018

[8] Patil, M. M., Hanni, A., Tejeshwar, C. H., Patil, P. (2017). A qualitative evaluation of the overall performance of MongoDB vs MySQL database primarily basedon insertion and retriewal operations using an internet/android application to explore load balancing Sharding in MongoDB and its advantages

[9] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, photograph analogies, in Proc. twenty eighth Annu. Conf. Comput. Graph. engage. Tech., 2001, pp. 327-340.

[10] ”Tags utilized in HTML”. world extensive web Consortium. November three, 1992. Retrieved November 16, 2008.

[11] R. Irony, D. Cohen-Or, and D. Lischinski, Colorization by means of example, in Proc. Eurograph. Symp. Rendering, vol. 2. 2005, pp. 201-210.

[12] First mention of HTML Tags at the www-communicate mailing listing". global extensive net Consortium. October 29, 1991. Retrieved April 8, 2007.

[13] JavaScript specification. Retrieved from http://www.w3.org/standards/ webdesign/script, November 1, 2014.