# Building a Real-Time Chat Application with Node.js, Express, and Socket.IO

mkdir chat-app  
cd chat-app  
npm init -y

Install dependencies:

npm install express socket.io

Index.html

<!DOCTYPE html>

<html>

<head>

<title>Chat app using Socket IO and Node JS</title>

<script src="https://cdn.tailwindcss.com"></script>

</head>

<body>

<h1 class="font-bold text-green-500

text-3xl text-center mt-5">

Chat applications in Node js

</h1>

<div>

<h2 class="font-semibold text-xl

text-center mt-5"

id="logo">

Chat App using Socket io

</h2>

</div>

<form class="flex flex-col justify-center

items-center mt-5"

id="form">

<input class="border border-gray-400

rounded-md mt-5 p-1"

type="text"

placeholder="Name"

id="myname">

<input class="border border-gray-400

rounded-md mt-5 p-1"

type="text"

placeholder="Message"

id="message">

<button class="bg-blue-500 rounded-md p-2

text-white mt-5">

Send

</button>

</form>

<div class="flex flex-col  justify-center

items-center mt-5" style="width:35%; height: auto; margin: auto; margin-top: 2%; padding: 15px;"

id="messageArea">

</div>

</body>

<script src="/socket.io/socket.io.js"></script>

<script>

let socket = io();

let form = document.getElementById('form');

let myname = document.getElementById('myname');

let message = document.getElementById('message');

let messageArea = document.getElementById("messageArea");

form.addEventListener("submit", (e) => {

e.preventDefault();

if (message.value) {

socket.emit('send name', myname.value);

socket.emit('send message', message.value);

message.value = "";

}

});

socket.on("send name", (username) => {

let name = document.createElement("p");

name.style.backgroundColor = "grey";

name.style.width = "100%";

name.style.textAlign = "center";

name.style.color = "white";

name.textContent = username + ":";

messageArea.appendChild(name);

});

socket.on("send message", (chat) => {

let chatContent = document.createElement("p");

chatContent.textContent = chat;

messageArea.appendChild(chatContent);

});

</script>

</html>

index.js

// Install dependencies:

// npm install express socket.io

const express = require('express');

const app = express();

const { Server } = require('socket.io');

const http = require('http');

const server = http.createServer(app);

const io = new Server(server);

const port = 5000;

app.get('/', (req, res) => {

    res.sendFile(\_\_dirname + '/index.html');

});

io.on('connection', (socket) => {

    socket.on('send name', (username) => {

        io.emit('send name', (username));

    });

    socket.on('send message', (chat) => {

        io.emit('send message', (chat));

    });

});

server.listen(port, () => {

    console.log(`Server is listening at the port: http://localhost:${port}`);

});

# Real Time News Aggregator with NodeJS and ExpressJS

## Steps to create the Project:

**Step 1:** Initialize the Project

npm init --yes

**Step 2:** Install Dependencies

npm install express axios ejs tailwindcss nodemon path

**Step 3:** Start the server

nodemon server.js

### Updated Dependencies:

"dependencies": {  
    "axios": "^1.6.8",  
    "cors": "^2.8.5",  
    "ejs": "^3.1.9",  
    "express": "^4.18.3",  
    "nodemon": "^3.1.0",  
    "path": "^0.12.7",  
    "tailwindcss": "^3.4.1"  
  }

**Example**: The below mentioned code implements Real Time News Aggregator with NodeJS and ExpressJS.

<!-- ../Views/index.ejs !-->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>News Aggregator</title>

<link

href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"

rel="stylesheet"

/>

</head>

<body class="bg-gray-100">

<div

class="container mx-auto mt-8 flex flex-col items-center justify-center"

>

<form

action="/search"

method="GET"

class="flex items-center justify-center mt-4"

>

<input

class="w-64 px-4 py-2 rounded-l-lg border border-gray-300 focus:outline-none focus:border-blue-500"

name="search"

type="search"

placeholder="Search"

aria-label="Search"

/>

<button

class="px-4 py-2 bg-red-500 text-white rounded-r-lg hover:bg-red-600"

type="submit"

>

Search

</button>

</form>

<form action="/sort-by-date" method="GET">

<button

class="mt-4 px-4 py-2 bg-blue-500 text-white rounded-lg hover:bg-blue-600"

type="submit"

>

Sort by Date

</button>

</form>

<form action="/news-by-date" method="GET" class="mt-4">

<label for="specific-date" class="block mb-2"

>Get News for Specific Date:</label

>

<input

id="specific-date"

class="w-64 px-4 py-2 rounded-lg border border-gray-300 focus:outline-none focus:border-blue-500"

name="date"

type="date"

aria-label="Specific Date"

/>

<button

class="px-4 py-2 bg-green-500 text-white rounded-lg hover:bg-green-600"

type="submit"

>

Get News

</button>

</form>

<h1 class="text-3xl font-bold my-4 text-center">Latest News</h1>

<div

id="news-container"

class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4"

>

<% news.forEach(article => { %>

<div class="article border p-4 bg-white rounded-lg shadow-md">

<h2 class="text-lg font-bold mb-2"><%= article.title %></h2>

<p class="text-gray-700"><%= article.description %></p>

<p class="text-gray-700 article-date">

Published at: <%= article.publishedAt %>

</p>

<a

href="<%= article.url %>"

class="text-blue-600 font-semibold mt-2 inline-block"

>Read more</a

>

</div>

<% }); %>

</div>

</div>

</body>

</html>

// server.js

const express = require("express");

const path = require("path");

const axios = require("axios");

const app = express();

app.set("views", path.join(\_\_dirname, "views"));

app.set("view engine", "ejs");

app.get("/", async (req, res) => {

try {

const response = await axios.get(

"https://newsapi.org/v2/top-headlines?country=in&apiKey=679b913ddb014617bcc93a0bb89ee1ee"

);

const data = response.data;

res.render("index", { news: data.articles });

} catch (error) {

console.error("Error fetching news:", error);

res.status(500).send("Error fetching news. Please try again later.");

}

});

app.get("/search", async (req, res) => {

try {

const searchTerm = req.query.search;

const response = await axios.get(

`https://newsapi.org/v2/everything?q=${searchTerm}&apiKey=679b913ddb014617bcc93a0bb89ee1ee`

);

const data = response.data.articles;

const news = data.filter((dataItem) => dataItem.title?.toLowerCase().includes(searchTerm?.toLowerCase()));

res.render("index", { news });

} catch (error) {

console.error("Error fetching search results:", error);

res

.status(500)

.send("Error fetching search results. Please try again later.");

}

});

app.get("/sort-by-date", async (req, res) => {

try {

const response = await axios.get(

"https://newsapi.org/v2/top-headlines?country=in&apiKey=679b913ddb014617bcc93a0bb89ee1ee"

);

const data = response.data.articles;

data.sort((a, b) => new Date(b.publishedAt) - new Date(a.publishedAt));

res.render("index", { news: data });

} catch (error) {

console.error("Error sorting articles by date:", error);

res.status(500).send("Error sorting articles by date. Please try again later.");

}

});

app.get("/news-by-date", async (req, res) => {

try {

const date = req.query.date;

const response = await axios.get(

`https://newsapi.org/v2/everything?q=\*&from=${date}&to=${date}&sortBy=popularity&apiKey=679b913ddb014617bcc93a0bb89ee1ee`

);

const data = response.data.articles;

res.render("index", { news: data });

} catch (error) {

console.error("Error fetching news by date:", error);

res.status(500).send("Error fetching news by date. Please try again later.");

}

});

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

# Real-Time Auction Platform using Node and Express.js

## Steps to create the Real-Time Auction Platform:

**Step 1:** Create the project folder using the following command.

mkdir auction-platform  
cd auction-platform

**Step 2:** Create a new project directory and navigate to your project directory.

mkdir server  
cd server

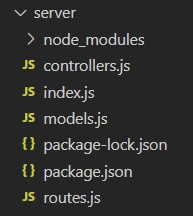
**Step 3:** Run the following command to initialize a new NodeJS project.

npm init -y

**Step 4:** Install the required the packages in your server using the following command.

npm install express body-parser cors mongoose

## ****Project Structure(backend):****



The updated **Dependencies** in package.json **:**

"dependencies": {  
 "axios": "^1.6.7",  
 "body-parser": "^1.20.2",  
 "cors": "^2.8.5",  
 "express": "^4.18.3",  
 "mongoose": "^8.2.1",  
}

**Code Example:** Create the required files as shown in the folder structure and add the following codes.

//index.js

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const routes = require('./routes');

const app = express();

const PORT = process.env.PORT || 3000;

mongoose.connect('Your MongoDB URI', {

useNewUrlParser: true,

useUnifiedTopology: true,

})

.then(() => console.log('MongoDB connected'))

.catch(err => console.error('MongoDB connection error:', err));

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

app.use('/', routes);

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

//models.js

const mongoose = require('mongoose');

const User = mongoose.model('User', new mongoose.Schema({

username: String,

contactNumber: String,

email: String,

}));

const ItemCategory = mongoose.model('ItemCategory', new mongoose.Schema({

name: String,

timeRemaining: String,

startingTime: Date,

state: String,

}));

const Product = mongoose.model('Product', new mongoose.Schema({

name: String,

description: String,

price: Number,

}));

module.exports = { User, ItemCategory, Product };

//routes.js

const express = require('express');

const router = express.Router();

const {

getAllUsers,

createUser,

getAllItemCategories,

createItemCategory,

getAllProducts,

createProduct,

} = require('./controllers');

router.get('/users', getAllUsers);

router.post('/users', createUser);

router.get('/itemCategories', getAllItemCategories);

router.post('/itemCategories', createItemCategory);

router.get('/products', getAllProducts);

router.post('/products', createProduct);

module.exports = router;

//controllers.js

const { User, ItemCategory, Product } = require('./models');

exports.getAllUsers = async (req, res) => {

try {

const users = await User.find();

res.json(users);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.createUser = async (req, res) => {

try {

const user = await User.create(req.body);

res.status(201).json(user);

} catch (err) {

res.status(400).json({ error: err.message });

}

};

exports.getAllItemCategories = async (req, res) => {

try {

const itemCategories = await ItemCategory.find();

res.json(itemCategories);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.createItemCategory = async (req, res) => {

try {

const itemCategory = await ItemCategory.create(req.body);

res.status(201).json(itemCategory);

} catch (err) {

res.status(400).json({ error: err.message });

}

};

exports.getAllProducts = async (req, res) => {

try {

const products = await Product.find();

res.json(products);

} catch (err) {

res.status(500).json({ error: err.message });

}

};

exports.createProduct = async (req, res) => {

try {

const product = await Product.create(req.body);

res.status(201).json(product);

} catch (err) {

res.status(400).json({ error: err.message });

}

};

**Step 5:** Start the Server

In the terminal, navigate to the server folder and run the following command to start the server:

node index.js

**Step 6:** Go to root directory and create the React application using the following command.

npx create-react-app client  
cd client

**Step 7:** Install the required packages in your application using the following command.

npm i react-router-dom axios styled-components

## ****Project Structure(frontend):****IMG_256

The updated **Dependencies** in package.json file will look like**:**

"dependencies": {  
 "axios": "^1.6.7",  
 "react": "^18.2.0",  
 "react-dom": "^18.2.0",  
 "react-router-dom": "^6.22.3",  
 "react-scripts": "5.0.1",  
 "styled-components": "^6.1.8",  
 "web-vitals": "^2.1.4"  
}

**Code Example:** Create the following files as shown in folder structure and add the following codes.

/\* src/App.css \*/

body {

font-family: 'Arial', sans-serif;

margin: 0;

padding: 0;

}

nav {

background-color: #333;

color: white;

padding: 10px;

}

nav h1 {

margin: 0;

}

nav a {

color: white;

margin-right: 10px;

text-decoration: none;

}

.container {

max-width: 800px;

margin: 20px auto;

}

ul {

list-style: none;

padding: 0;

}

li {

margin-bottom: 10px;

}

.members-container {

background-color: #f9f9f9;

padding: 20px;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.members-list {

padding-left: 20px;

}

.members-item {

font-size: 16px;

border-bottom: 1px solid #ddd;

padding: 10px 0;

}

.members-item:last-child {

border-bottom: none;

}

.categories-container {

max-width: 800px;

margin: auto;

padding: 20px;

}

.categories-list {

display: flex;

flex-wrap: wrap;

justify-content: space-between;

}

.category-item {

width: 48%;

margin-bottom: 20px;

border: 1px solid #ccc;

border-radius: 8px;

overflow: hidden;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

transition: transform 0.3s ease-in-out;

}

.category-item:hover {

transform: scale(1.02);

}

.category-image img {

width: 100%;

height: 200px;

object-fit: cover;

}

.category-details {

padding: 15px;

}

h3 {

margin-bottom: 10px;

font-size: 1.5rem;

}

.category-description {

margin-bottom: 15px;

}

.category-seller-info {

margin-bottom: 15px;

}

.category-auction-info {

font-size: 0.9rem;

}

.bid-modal {

position: absolute;

top: 0;

right: 0;

background: #fff;

padding: 20px;

border: 1px solid #ccc;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

z-index: 100;

}

.bid-modal h3 {

margin-bottom: 10px;

}

.bid-modal p {

margin-bottom: 5px;

}

.bid-modal input {

margin-bottom: 10px;

padding: 5px;

width: 100px;

}

.bid-modal button {

padding: 5px 10px;

margin-right: 10px;

cursor: pointer;

}

.bid-modal button:last-child {

margin-right: 0;

}

// src/App.js

import React from 'react';

import { BrowserRouter as Router, Route, NavLink, Routes } from 'react-router-dom';

import Home from './components/Home';

import Categories from './components/Categories';

import Members from './components/Members';

import Products from './components/Products';

import './App.css';

function App() {

return (

<Router>

<nav>

<h1><center>Real Time Auction Platform</center></h1>

<div>

<NavLink to="/" activeClassName="active">

Home

</NavLink>

<NavLink to="/categories" activeClassName="active">

Categories

</NavLink>

<NavLink to="/members" activeClassName="active">

Members

</NavLink>

<NavLink to="/products" activeClassName="active"> { }

Products

</NavLink>

</div>

</nav>

<div className="container">

<Routes>

<Route path="/" element={<Home />} />

<Route path="/categories" element={<Categories />} />

<Route path="/members" element={<Members />} />

<Route path="/products" element={<Products />} /> { }

</Routes>

</div>

</Router>

);

}

export default App;

//components/Home.js

import React from 'react';

function Home() {

const userData = {

username: 'User 123',

};

const username = userData ? userData.username : 'Unknown';

return (

<div>

<h2>Welcome to the Real-Time Auction Platform</h2>

<p>Hello, {username}!</p>

</div>

);

}

export default Home;

//components/Navbar.js

import React from 'react';

import { Link } from 'react-router-dom';

const Navbar = () => {

return (

<nav>

<Link to="/">Real-Time Auction Platform</Link>

<div>

<Link to="/">Home</Link>

<Link to="/categories">Categories</Link>

<Link to="/members">Members</Link>

<Link to="/products">Products</Link>

</div>

</nav>

);

};

export default Navbar;

// src/components/Members.js

import React, { useState } from "react";

import "../App.css";

function Members() {

const hardcodedMembers = [

{

id: 1,

username: "User123",

contactNumber: "9898989898",

email: "user123@gmail.com",

},

{

id: 2,

username: "User2",

contactNumber: "234-567-8901",

email: "user2@example.com",

},

{

id: 3,

username: "User3",

contactNumber: "345-678-9012",

email: "user3@example.com",

},

{

id: 4,

username: "User4",

contactNumber: "456-789-0123",

email: "user4@example.com",

},

{

id: 5,

username: "User5",

contactNumber: "567-890-1234",

email: "user5@example.com",

},

{

id: 6,

username: "User6",

contactNumber: "678-901-2345",

email: "user6@example.com",

},

{

id: 7,

username: "User7",

contactNumber: "789-012-3456",

email: "user7@example.com",

},

{

id: 8,

username: "User8",

contactNumber: "890-123-4567",

email: "user8@example.com",

},

{

id: 9,

username: "User9",

contactNumber: "901-234-5678",

email: "user9@example.com",

},

{

id: 10,

username: "User10",

contactNumber: "012-345-6789",

email: "user10@example.com",

},

];

const [members, setMembers] = useState(hardcodedMembers);

return (

<div className="members-container">

<h2>Members</h2>

<ul className="members-list">

{members.map((member) => (

<li key={member.id} className="members-item">

<strong>{member.username}</strong> - {member.contactNumber} -{" "}

{member.email}

</li>

))}

</ul>

</div>

);

}

export default Members;

//components/Categories.js

import React, { useState } from "react";

import "../App.css";

function Categories() {

const [biddingItem, setBiddingItem] = useState(null);

const [bidAmount, setBidAmount] = useState(0);

const getUserByIndex = (index) => {

const users = [

{

username: "User1",

contactNumber: "123-456-7890",

email: "user1@example.com",

},

{

username: "User2",

contactNumber: "234-567-8901",

email: "user2@example.com",

},

{

username: "User3",

contactNumber: "345-678-9012",

email: "user3@example.com",

},

{

username: "User4",

contactNumber: "456-789-0123",

email: "user4@example.com",

},

{

username: "User5",

contactNumber: "567-890-1234",

email: "user5@example.com",

},

];

return users[index];

};

const categoriesData = [

{

id: 1,

name: "Car",

description: "Explore the latest car models .",

imageUrl:

"https://media.geeksforgeeks.org/wp-content/uploads/20240122184958/images2.jpg",

sellerInfo: getUserByIndex(0),

auctionInfo: {

status: "Open",

timeRemaining: "5h 30m",

basePrice: 50,

currentPrice: 0,

currentBidder: null,

},

},

{

id: 2,

name: "Clothing",

description: "Stay in style with our fashionable clothing collection.",

imageUrl:

"https://media.geeksforgeeks.org/wp-content/uploads/20230407153938/gfg-hoodie.jpg",

sellerInfo: getUserByIndex(1),

auctionInfo: {

status: "Open",

timeRemaining: "3d 12h",

basePrice: 30,

currentPrice: 0,

currentBidder: null,

},

},

{

id: 3,

name: "Books",

description:

"Immerse yourself in the world of literature with our diverse book collection.",

imageUrl:

"https://media.geeksforgeeks.org/wp-content/uploads/20240110011929/glasses-1052010\_640.jpg",

sellerInfo: getUserByIndex(2),

auctionInfo: {

status: "Open",

timeRemaining: "1w 1d",

basePrice: 20,

currentPrice: 0,

currentBidder: null,

},

},

{

id: 4,

name: "Honda second hand vehicle",

description: "Best working condition .",

imageUrl:

"https://media.geeksforgeeks.org/wp-content/uploads/20240122182422/images1.jpg",

sellerInfo: getUserByIndex(3),

auctionInfo: {

status: "Open",

timeRemaining: "2d 8h",

basePrice: 80,

currentPrice: 0,

currentBidder: null,

},

},

{

id: 5,

name: "Sports wear",

description:

"Fuel your active lifestyle with our high-quality sports tshirt.",

imageUrl:

"https://media.geeksforgeeks.org/wp-content/uploads/20230407153931/gfg-tshirts.jpg",

sellerInfo: getUserByIndex(4),

auctionInfo: {

status: "Open",

timeRemaining: "6h 20m",

basePrice: 40,

currentPrice: 0,

currentBidder: null,

},

},

];

const handleBidClick = (item) => {

setBiddingItem(item);

setBidAmount(

item.auctionInfo.basePrice + Math.floor(Math.random() \* 50) + 1

);

};

const handlePlaceBid = () => {

setBiddingItem((prevItem) => ({

...prevItem,

auctionInfo: {

...prevItem.auctionInfo,

currentPrice: bidAmount,

currentBidder: "user123",

},

}));

};

return (

<div className="categories-container">

<h2>Categories</h2>

<div className="categories-list">

{categoriesData.map((category) => (

<div key={category.id} className="category-item">

<div className="category-image">

<img src={category.imageUrl} alt={category.name} />

</div>

<div className="category-details">

<h3>{category.name}</h3>

<p className="category-description">{category.description}</p>

<div className="category-seller-info">

<p>

<strong>Seller Information:</strong>

</p>

<p>Username: {category.sellerInfo.username}</p>

<p>Contact Number: {category.sellerInfo.contactNumber}</p>

<p>Email: {category.sellerInfo.email}</p>

</div>

<div className="category-auction-info">

<p>

<strong>Auction Information:</strong>

</p>

<p>Status: {category.auctionInfo.status}</p>

<p>Time Remaining: {category.auctionInfo.timeRemaining}</p>

{category.auctionInfo.status === "Open" && (

<>

<button onClick={() => handleBidClick(category)}>

Click to Bid

</button>

</>

)}

</div>

{biddingItem && biddingItem.id === category.id && (

<div className="bid-modal">

<h3>Bid Information</h3>

<p>

<strong>Item:</strong> {biddingItem.name}

</p>

<p>

<strong>Base Price:</strong> $

{biddingItem.auctionInfo.basePrice}

</p>

<p>

<strong>Current Price:</strong> $

{biddingItem.auctionInfo.currentPrice}

</p>

<p>

<strong>Current Bidder:</strong>{" "}

{biddingItem.auctionInfo.currentBidder || "None"}

</p>

<input

type="number"

value={bidAmount}

onChange={(e) => setBidAmount(e.target.value)}

min={biddingItem.auctionInfo.basePrice + 1}

/>

<button onClick={handlePlaceBid}>Place Bid</button>

<button onClick={() => setBiddingItem(null)}>Close</button>

</div>

)}

</div>

</div>

))}

</div>

</div>

);

}

export default Categories;

//components/Products.js

import React, { useState } from "react";

function Products() {

const defaultSellerInfo = {

username: "User 123",

contactNumber: "9898989898",

email: "user123@gmail.com",

};

const defaultProduct = {

name: "",

description: "",

price: 0,

sellerInfo: defaultSellerInfo,

startingDate: "",

endingDate: "",

image: null,

};

const [products, setProducts] = useState([]);

const [newProduct, setNewProduct] = useState(defaultProduct);

const [editProduct, setEditProduct] = useState(null);

const [editName, setEditName] = useState("");

const [editDescription, setEditDescription] = useState("");

const [editPrice, setEditPrice] = useState("");

const [editStartingDate, setEditStartingDate] = useState("");

const [editEndingDate, setEditEndingDate] = useState("");

const handleInputChange = (e) => {

const { name, value } = e.target;

switch (name) {

case "name":

setEditName(value);

break;

case "description":

setEditDescription(value);

break;

case "price":

setEditPrice(value);

break;

case "startingDate":

setEditStartingDate(value);

break;

case "endingDate":

setEditEndingDate(value);

break;

default:

break;

}

};

const handleImageChange = (e) => {

setEditProduct({

...editProduct,

image: e.target.files[0],

});

};

const addProduct = () => {

setProducts([...products, newProduct]);

setNewProduct(defaultProduct);

};

const deleteProduct = (index) => {

setProducts(products.filter((\_, i) => i !== index));

};

const updateProduct = (index) => {

const updatedProducts = [...products];

const updatedProduct = {

...editProduct,

name: editName,

description: editDescription,

price: editPrice,

startingDate: editStartingDate,

endingDate: editEndingDate,

};

updatedProducts[index] = updatedProduct;

setProducts(updatedProducts);

setEditProduct(null);

// Clear edit fields

setEditName("");

setEditDescription("");

setEditPrice("");

setEditStartingDate("");

setEditEndingDate("");

};

return (

<div

style={{

fontFamily: "Arial, sans-serif",

maxWidth: "800px",

margin: "0 auto",

}}

>

<h2 style={{ textAlign: "center", marginBottom: "20px" }}>Products</h2>

<div>

<h3>Add New Product</h3>

<div style={{ marginBottom: "10px" }}>

<input

type="text"

name="name"

placeholder="Name"

value={newProduct.name}

onChange={(e) =>

setNewProduct({ ...newProduct, name: e.target.value })

}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="text"

name="description"

placeholder="Description"

value={newProduct.description}

onChange={(e) =>

setNewProduct({ ...newProduct, description: e.target.value })

}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="number"

name="price"

placeholder="Price"

value={newProduct.price}

onChange={(e) =>

setNewProduct({ ...newProduct, price: e.target.value })

}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="text"

name="startingDate"

placeholder="Starting Date"

value={newProduct.startingDate}

onChange={(e) =>

setNewProduct({ ...newProduct, startingDate: e.target.value })

}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="text"

name="endingDate"

placeholder="Ending Date"

value={newProduct.endingDate}

onChange={(e) =>

setNewProduct({ ...newProduct, endingDate: e.target.value })

}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="file"

name="image"

onChange={(e) =>

setNewProduct({ ...newProduct, image: e.target.files[0] })

}

style={{ padding: "5px" }}

/>

<button

onClick={addProduct}

style={{ padding: "5px 10px", marginLeft: "10px" }}

>

Add Product

</button>

</div>

</div>

<div>

<h3>Products List</h3>

<ul style={{ listStyleType: "none", padding: 0 }}>

{products.map((product, index) => (

<li

key={index}

style={{

marginBottom: "20px",

border: "1px solid #ccc",

padding: "10px",

borderRadius: "5px",

boxShadow: "0 0 5px rgba(0, 0, 0, 0.1)",

}}

>

<div

style={{

display: "flex",

alignItems: "center",

marginBottom: "10px",

}}

>

{product.image && (

<img

src={URL.createObjectURL(product.image)}

alt="Product"

style={{

marginRight: "10px",

width: "100px",

height: "auto",

borderRadius: "5px",

}}

/>

)}

<div>

<strong>Name:</strong> {product.name}

<br />

<strong>Description:</strong> {product.description}

<br />

<strong>Price:</strong> ${product.price}

<br />

<strong>Starting Date:</strong> {product.startingDate}

<br />

<strong>Ending Date:</strong> {product.endingDate}

<br />

<strong>Seller Info:</strong> <br />

Username: {product.sellerInfo.username}

<br />

Contact Number: {product.sellerInfo.contactNumber}

<br />

Email: {product.sellerInfo.email}

</div>

</div>

<div>

{editProduct === product ? (

<>

<input

type="text"

name="name"

value={editName}

onChange={handleInputChange}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="text"

name="description"

value={editDescription}

onChange={handleInputChange}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="number"

name="price"

value={editPrice}

onChange={handleInputChange}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="text"

name="startingDate"

value={editStartingDate}

onChange={handleInputChange}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="text"

name="endingDate"

value={editEndingDate}

onChange={handleInputChange}

style={{ marginRight: "10px", padding: "5px" }}

/>

<input

type="file"

name="image"

onChange={handleImageChange}

style={{ padding: "5px" }}

/>

<button

onClick={() => updateProduct(index)}

style={{ padding: "5px 10px", marginLeft: "10px" }}

>

Update

</button>

<button

onClick={() => setEditProduct(null)}

style={{ padding: "5px 10px", marginLeft: "10px" }}

>

Cancel

</button>

</>

) : (

<>

<button

onClick={() => {

setEditProduct(product);

setEditName(product.name);

setEditDescription(product.description);

setEditPrice(product.price);

setEditStartingDate(product.startingDate);

setEditEndingDate(product.endingDate);

}}

>

Edit

</button>

<button

onClick={() => deleteProduct(index)}

style={{ marginLeft: "10px" }}

>

Delete

</button>

</>

)}

</div>

</li>

))}

</ul>

</div>

</div>

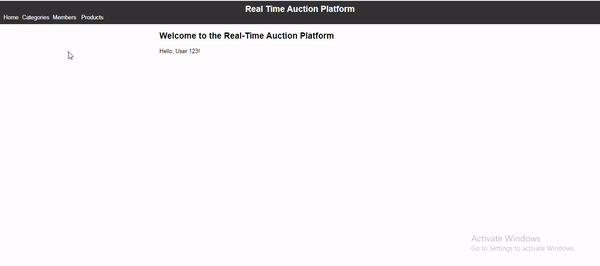
);

}

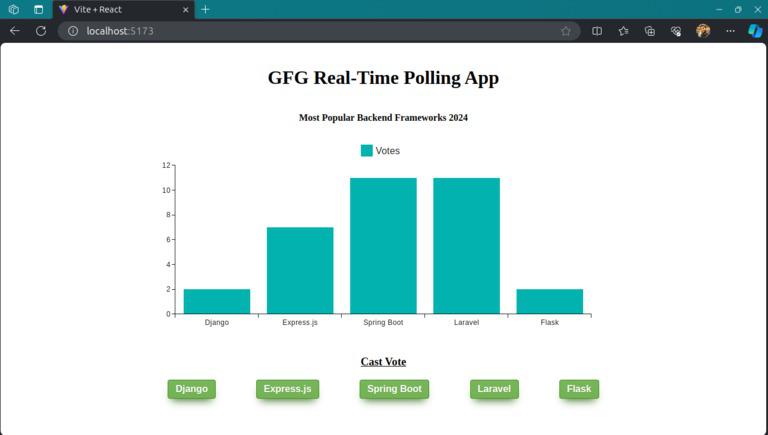
export default Products;

#### Start your application using the following command.

npm start



# Real-Time Polling App with Node and React



## Approach to create a Real-time Polling Application:

* The **BarChart component** is a visual representation of poll data for different backend frameworks.
* It receives real-time updates via a WebSocket connection using the **socket.io-client** library.
* The component displays a bar chart with vote counts for each framework and allows users to cast votes by clicking on corresponding buttons.
* Establishes a WebSocket connection using **io('http://localhost:5000')** via [useMemo](https://www.geeksforgeeks.org/react-js-usememo-hook/" \t "https://www.geeksforgeeks.org/real-time-polling-app-with-node-and-react/_blank).
* Uses [useEffect](https://www.geeksforgeeks.org/reactjs-useeffect-hook/" \t "https://www.geeksforgeeks.org/real-time-polling-app-with-node-and-react/_blank) to handle connection events and attempts to reconnect in case of errors.
* Renders the BarChart component, passing the socket connection as a prop.
* Creates an Express server and a Socket.io server.
* Defines an initial set of backend frameworks with vote counts.
* Handles [WebSocket](https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/" \t "https://www.geeksforgeeks.org/real-time-polling-app-with-node-and-react/_blank) connections and updates in real-time.
* Logs user connections, emits initial data, and updates data on vote events

## Steps to Create the Backend Server:

**Step 1:** Create a directory for project

mkdir server  
cd server

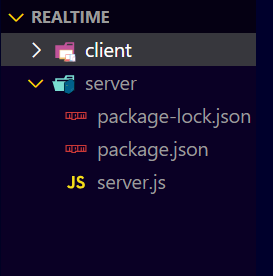
**Step 2**: Initialized the Express app and installing the required packages

npm init -y

**Step 3:** Install the required dependency in your server using the following command.

npm i express nodemon socket.io

### Project Structure:



The updated dependencies in package.json file of backend will look like:

"dependencies": {  
 "cors": "^2.8.5",  
 "express": "^4.18.2",  
 "socket.io": "^4.7.4"  
}

// server.js

const express = require("express");

const http = require('http');

const { Server } = require("socket.io");

const app = express();

const port = 5000;

const server = http.createServer(app);

const io = new Server(server, {

cors: {

origin: "\*",

credentials: true

}

});

const frameworks = {

"0": { votes: 0, label: "Django" },

"1": { votes: 0, label: "Express.js" },

"2": { votes: 0, label: "Spring Boot" },

"3": { votes: 0, label: "Laravel" },

"4": { votes: 0, label: "Flask" }

};

io.on("connection", (socket) => {

console.log("User", socket.id)

io.emit("update", frameworks);

socket.on("vote", (index) => {

if (frameworks[index]) {

frameworks[index].votes += 1;

}

io.emit("update", frameworks);

});

});

server.listen(port, () =>

console.log(`Listening at http://localhost:${port}`)

);

**Start your application using the following command.**

node server.js

## Steps to Create the Frontend:

**Step 1:** Initialize the React App with Vite and installing the required packages.

npm create vite@latest -y

**Step 2:** Navigate to the root of the project using the following command.

cd client

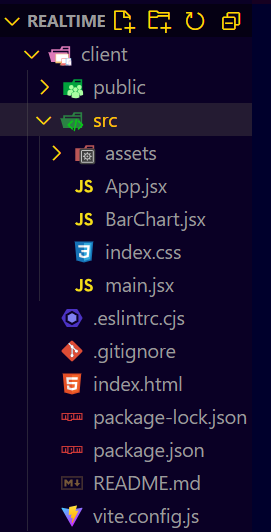
**Step 3:** Install the necessary package in your project using the following command.

npm install socket.io-client @mui/x-charts

**Step 4:** Install the node\_modules using the following command.

npm install

## Project Structure:



The updated dependencies in package.json file of frontend will look like:

"dependencies": {  
 "@mui/x-charts": "^6.19.5",  
 "react": "^18.2.0",  
 "react-dom": "^18.2.0",  
 "socket.io-client": "^4.7.4"  
},  
"devDependencies": {  
 "@types/react": "^18.2.56",  
 "@types/react-dom": "^18.2.19",  
 "@vitejs/plugin-react": "^4.2.1",  
 "eslint": "^8.56.0",  
 "eslint-plugin-react": "^7.33.2",  
 "eslint-plugin-react-hooks": "^4.6.0",  
 "eslint-plugin-react-refresh": "^0.4.5",  
 "vite": "^5.1.4"  
}

**Example:** Write the following code in frontend files of the project

/\* ./src/index.css \*/

.bar {

display: flex;

justify-content: center;

width: 100%;

}

h1 {

text-align: center;

margin: 40px;

}

h3 {

text-align: center;

}

h4 {

text-align: center;

}

.myButton {

box-shadow: 0px 10px 14px -7px #3e7327;

background: linear-gradient(to bottom, #77b55a 5%, #72b352 100%);

background-color: #77b55a;

border-radius: 4px;

border: 1px solid #4b8f29;

display: inline-block;

cursor: pointer;

color: #ffffff;

font-family: Arial;

font-size: 16px;

font-weight: bold;

padding: 6px 12px;

text-decoration: none;

text-shadow: 0px 1px 0px #5b8a3c;

}

.myButton:hover {

background: linear-gradient(to bottom, #72b352 5%, #77b55a 100%);

background-color: #72b352;

}

.myButton:active {

position: relative;

top: 1px;

}

.btn {

display: flex;

width: 60%;

justify-content: space-around;

gap: 30px;

margin: auto;

}

// ./src/App.jsx

import {

useEffect,

useMemo,

} from "react"

import Bar from "./BarChart.js"

import { io } from 'socket.io-client'

function App() {

const socket = useMemo(() =>

io('http://localhost:5000'));

useEffect(() => {

socket.on('connect', () =>

console.log(socket.id))

socket.on('connect\_error', () => {

setTimeout(() =>

socket.connect(), 5000)

})

return () => {

socket.off('connect', () =>

console.log("connected"));

socket.off('disconnect', () =>

console.log("connected"));

};

}, [])

return (

<>

<h1>GFG Real-Time Polling App</h1>

<Bar socket={socket} />

</>

)

}

export default App

// ./src/BarChart.jsx

import \* as React from 'react';

import {

BarChart

} from '@mui/x-charts/BarChart';

export default function Bar({ socket }) {

const [DT, setData] = React.useState([]);

React.useEffect(() => {

socket.on('update', (frameworks) => {

const newData = []

for (const key in frameworks) {

if (frameworks.hasOwnProperty(key)) {

const { votes, label } = frameworks[key];

newData.push(votes)

}

}

setData(newData)

})

}, [])

const updateVote = (id) => {

socket.emit('vote', id)

}

return (

<>

<h4>Most Popular Backend Frameworks 2024</h4>

<div className='bar'>

<BarChart

width={800}

height={350}

series={[

{

data: DT.length > 0 ?

DT : [0, 0, 0, 0, 0],

id: 'uvId', label: 'Votes'

},

]}

xAxis={[{

data: ["Django", "Express.js",

"Spring Boot", "Laravel", "Flask"],

scaleType: 'band'

}]}

/>

</div>

<h3><u>Cast Vote</u></h3>

<div className='btn'>

<button className='myButton'

onClick={() => updateVote(0)}>

Django

</button>

<button className='myButton'

onClick={() => updateVote(1)}>

Express.js

</button>

<button className='myButton'

onClick={() => updateVote(2)}>

Spring Boot

</button>

<button className='myButton'

onClick={() => updateVote(3)}>

Laravel

</button>

<button className='myButton'

onClick={() => updateVote(4)}>

Flask

</button>

</div>

</>

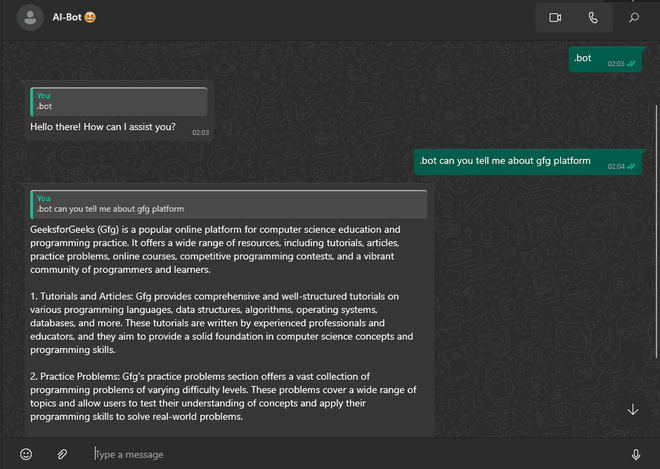
);

}

To start client server:

npm run dev

# AI Whatsapp Bot using NodeJS, Whatsapp-WebJS And Gemini AI



## Approach to create AI Whatsapp Bot

* If your ever used Whatsapp Web in your browser you can easily understood that we are just going to create one without any frontend.
* We will create a simple node server, which send receive messages from whatsapp web and reply with Gemini AI.
* To authenticate, we will use our whatsapp for scanning the QR code.
* To create AI response we will be using Gemini API, we send prompt to API and it will generate response that we need to send back to user.

## Steps To Create The Project

**Step 1:** Creating Node App with the following command.

npm init -y

**Step 2:** Installing Required Packages

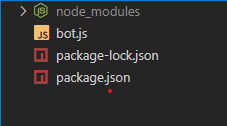
npm i whatsapp-web.js @google/generative-ai qrcode-terminal

#### In Non Window System Additional Installation Required

**npm i puppeteer**

**apt-get install gconf-service libasound2 libatk1.0-0 libc6 libcairo2 libcups2 libdbus-1-3 libexpat1 libfontconfig1 libgcc1 libgconf-2-4 libgdk-pixbuf2.0-0 libglib2.0-0 libgtk-3-0 libnspr4 libpango-1.0-0 libpangocairo-1.0-0 libstdc++6 libx11-6 libx11-xcb1 libxcb1 libxcomposite1 libxcursor1 libxdamage1 libxext6 libxfixes3 libxi6 libxrandr2 libxrender1 libxss1 libxtst6 ca-certificates fonts-liberation libappindicator1 libnss3 lsb-release xdg-utils wget**

**Project Structure:**

Project Structure

The updated **dependencies** in package.json file will look like**:**

"dependencies": {  
 "@google/generative-ai": "^0.2.1",  
 "qrcode-terminal": "^0.12.0",  
 "whatsapp-web.js": "^1.23.0"  
}

**Step 3:** Get the API key from the Gemini AI API

**Code Example:** Create the bot.js file and add the following code and add your API-key.

//For Window User bot.js

//Importing All Necessary Packages

const { Client, LocalAuth } = require('whatsapp-web.js');

const qrcode = require('qrcode-terminal');

const { GoogleGenerativeAI } = require('@google/generative-ai');

//Creating instances

const genAI = new GoogleGenerativeAI('API-KEY');

const client = new Client({

authStrategy: new LocalAuth(),

});

//Initializing GenAI model

const model = genAI.getGenerativeModel({ model: &quot;gemini-pro&quot; });

//Function to generate reponse from AI model and reply to user

async function generate(prompt, message) {

const result = await model.generateContent(prompt);

const response = await result.response;

const text = response.text();

await message.reply(text); //Reply to user

}

//All event listener to know client status

client.on('qr', (qr) =&gt; {

qrcode.generate(qr, { small: true });

});

client.on('authenticated', () =&gt; {

console.log('Client is authenticated!');

});

client.on('ready', () =&gt; {

console.log('Client is ready!');

});

client.on('disconnected', () =&gt; {

console.log('Client is disconnected!');

});

client.on('auth\_failure', () =&gt; {

console.log('Client is auth\_failure!');

});

client.on('message', async (message) =&gt; {

if (message.body.includes('.bot')) {

var query;

//Extracting text from message body using regular expression method

const regxmatch = message.body.match(/.bot(.+)/);

//If no text followed by .bot then we use &quot;Hi&quot; as text

if (regxmatch) {

query = regxmatch[1];

}

else {

console.log(&quot;No regex match!&quot;);

query = &quot;Hi&quot;;

}

//Call the generate function

generate(query, message);

};

});

client.initialize();

//For Non-Window User Bot.js

const { Client, LocalAuth } = require('whatsapp-web.js');

const qrcode = require('qrcode-terminal');

const { GoogleGenerativeAI } = require('@google/generative-ai');

const genAI = new GoogleGenerativeAI(API\_KEY);

const client = new Client({

authStrategy: new LocalAuth(),

puppeteer: {

executablePath: '/usr/bin/google-chrome-stable',

args: ['--no-sandbox', '--disable-setuid-sandbox'],

},

});

const model = genAI.getGenerativeModel({ model: &quot;gemini-pro&quot; });

async function generate(prompt, message) {

const result = await model.generateContent(prompt);

const response = await result.response;

const text = response.text();

await message.reply(text);

}

client.on('qr', (qr) =&gt; {

qrcode.generate(qr, { small: true });

});

client.on('authenticated', () =&gt; {

console.log('Client is authenticated!');

});

client.on('ready', () =&gt; {

console.log('Client is ready!');

});

client.on('disconnected', () =&gt; {

console.log('Client is disconnected!');

});

client.on('auth\_failure', () =&gt; {

console.log('Client is auth\_failure!');

});

client.on('message', async (message) =&gt; {

if (message.body.includes('.bot')) {

var query;

const regxmatch = message.body.match(/.bot(.+)/);

if (regxmatch) {

query = regxmatch[1];

}

else {

console.log(&quot;No regex match!&quot;);

query = &quot;Hi&quot;;

}

generate(query, message);

};

});

client.initialize();

**To start the application run the following command.**

node bot.js

### Authenticate the server

* After running the server you will find a QR code in terminal, Scan it using whatsapp mobile app.
* Finally your bot is running on mobile number you used to authenticate.
* Just message to this whatsapp number, where message should start with ".bot", See the final result.

