A **mini chat website** is a perfect project to practice the basics of **Node.js**, **MongoDB**, and **Vue.js**. Here's a suggested roadmap to help you achieve your goal with a small but complete real-time chat app:

## 🧠 ****Core Features for Your Mini Chat Project****

### ✅ Backend (Node.js + Express + MongoDB):

* User registration and login (simple auth)
* Store chat messages in MongoDB
* Use **Socket.IO** for real-time messaging
* API to get previous messages (chat history)

### ✅ Frontend (Vue.js):

* Register/Login UI
* Chat interface (message input + chat history)
* Real-time message updates using **Socket.IO client**

## 📁 Suggested Folder Structure

chat-app/

├── backend/ # Node.js backend

│ ├── server.js

│ ├── routes/

│ ├── models/

│ └── controllers/

├── frontend/ # Vue.js frontend (Vite or Vue CLI)

│ ├── src/

│ └── public/

└── README.md

## ⚙️ Key Technologies

| **Layer** | **Stack** |
| --- | --- |
| Frontend | Vue 3, Socket.IO client, Axios |
| Backend | Node.js, Express, Socket.IO, Mongoose |
| Database | MongoDB (local or MongoDB Atlas) |

## 🗓️ Project Plan (Milestones)

### Day 1–2: Setup Project

* Setup backend (Node + Express)
* Setup MongoDB and test connection
* Setup Vue project

### Day 3–4: Basic User System

* User model (username only or simple password)
* Registration/Login routes
* JWT or session auth (optional for simple project)

### Day 5–6: Real-time Messaging

* Integrate Socket.IO (send/receive messages)
* Save messages in MongoDB
* Fetch message history on page load

### Day 7: Final Touches

* Improve UI (basic styles)
* Display usernames and timestamps
* Test and deploy (optional)

Would you like a starter template or code snippet for the backend or frontend setup?

## ✅ ****Day 1–2: Setup Project****

### 📁 Folder Structure

chat-app/

├── backend/ # Node.js + Express

├── frontend/ # Vue 3 app

## 1. ✅ Setup Backend (Node + Express)

### Step-by-step:

mkdir chat-app

cd chat-app

mkdir backend

cd backend

npm init -y

npm install express mongoose cors dotenv

### Create basic files:

touch server.js .env

mkdir routes models

### server.js (basic setup):

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

require('dotenv').config();

const app = express();

app.use(cors());

app.use(express.json());

// Sample route

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

res.send('Chat backend is running!');

});

// MongoDB connection

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

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

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

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

### .env file:

MONGO\_URI=mongodb://localhost:27017/chatdb

PORT=5000

Replace the URI with your MongoDB Atlas URI if you're not using local MongoDB.

## 2. ✅ Setup MongoDB

* If local: make sure MongoDB is running (e.g., mongod)
* If remote: use [MongoDB Atlas](https://www.mongodb.com/cloud/atlas/register)
* Use mongoose.connect() as shown above to test

## 3. ✅ Setup Vue 3 Frontend

### From root chat-app folder:

npm create vue@latest frontend

* Choose:
  + **Vue 3**
  + **No TypeScript** (or yes, your choice)
  + **No router** (you can add later)
  + **No Pinia** (or yes if you want)
  + **No testing** for now

cd frontend

npm install

npm run dev

## ✅ Test Everything

* Backend: visit http://localhost:5000 → should show "Chat backend is running!"
* Frontend: visit http://localhost:5173 (or the port Vite shows) → should show Vue starter page
* MongoDB: should log "MongoDB connected"