



Gujarat Technological University

Chandkheda, Ahmedabad
Affiliated



V.V.P. Engineering College, Rajkot

A
Project Report
on

Memories-App

Under the course of
SUMMER INTERNSHIP (3170001)
B.E. Semester-VII

(Computer Engineering)

Sr. No	Name of student	Enrolment No.
1	Makwana Dishen Alpeshbhai	180470107035

Dipesh Joshi

(Assistant Prof.)

Dr. Tejas Patalia

(HOD)

Academic Year

(2021-22)



V.V.P. ENGINEERING COLLEGE

Department Of Computer Engineering

Vision of the Institute

- To be an exemplary institute, transforming students into competent professionals with human values.

Mission of the Institute

- To provide a conducive academic environment for strengthening technical capabilities of the students.
- To strengthen linkage with industries, alumni and professional bodies.
- To organize various co-curricular and extra-curricular activities for overall development of the students.
- To practice good governance and conduct value- based activities for making students responsible citizens.

Vision of the Department

- Transforming students into globally efficient professionals with moral values.

Mission of the Department

- To provide a strong foundation of computer engineering through effective teaching learning process.
- To enhance industry linkage & alumni network for better placement and real-world exposure.
- To provide various opportunities & platforms for all round development of students & encourage them for value-based practices.



V.V.P. ENGINEERING COLLEGE

RAJKOT

Certificate

This is to certify that

Mr. DISHEN MAKWANA, Enrollment No: 180470107035, Branch: Computer Engineering, Semester: 7 has satisfactorily completed the course in the subject: **SUMMER INTERNSHIP (3170001)** within the four walls of V.V.P. Engineering College, Rajkot.

Date of Submission:

Prof. Dipesh Joshi,
Staff In-Charge

Head of Department,
Department of Computer Engineering,
V.V.P. Engineering College

Index

Sr No.	Context	Page No.
1	Introduction	5
2	Building the API's using Express	6
3	Building the frontend User interface using ReactJS	9
4	Flow Diagram	11
5	Building the real-time update with socket.io	12

Introduction

Objective

- The objective of this project is to create a social media platform (i.e. Memories-App) in which people can create account, can add friends, can add new friends, can remove friends, can upload post as images or text with descriptions(captions), can view post of their friends, can like post of their friends, can maintain their profile and can also see their friends profile on their profile page, also see their friends profile on their profile page.

Project Context

- Memories-App is a real world simple social media application just like Instagram, Facebook, etc, but with less features.
- This project is best for those who want to dive deeper into full stack using Nodejs, Express, React and MongoDB after learning HTML, CSS and JS.
- Building a full stack application single-handedly is a tough task but learning and building such applications will help you master your skills.
- Building this project will be a challenging task where you will get to learn and explore all about the MVC pattern, NoSQL Database (MongoDB) and much more.

Project Stages

We can divide the project based on the stack used:

- ExpressJS: Framework for creating api's that will connect Memories-App and MongoDB .
- MongoDB: Using NoSQL Database.
- NodeJs: Back-end JavaScript runtime environment, that is my application's server.
- Socket.IO: Building realtime update post in web
- ReactJs: A Javascript library for building User Interface. (Front-end)
- Postman: A great tool for testing restful api's.
- MondoDB Atlas: To deploy, operate and scale MongoDB in the Cloud.

Application

- You can use this among your friends or family and use it as a real world small scale social media application.

Building the API's using Express:

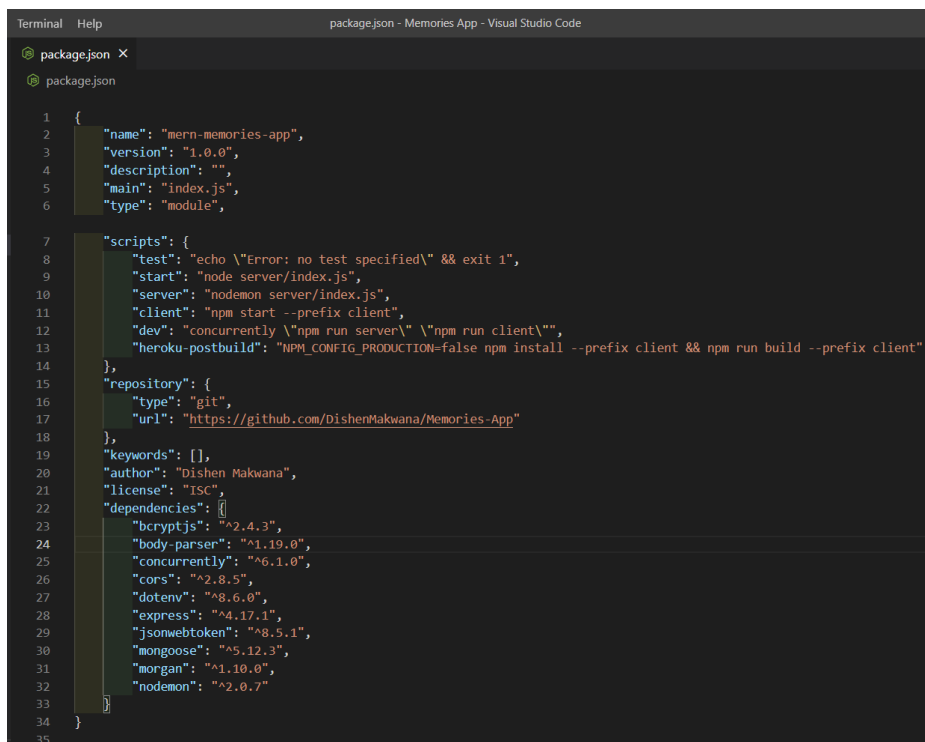
So the first phase of the project is to build the API's using the Express, NodeJs, MongoDB, and Postman.

Requirements

- Start building a server using express. Express.js, which is a lightweight framework for creating web servers, makes it easier to organize your application's functionality with middlewares and routes.
- **Install Mongoose:** Mongoose provides a straight-forward, schema-based solution to model your application data.
- **Install Nodemon:** Nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.
- **Helmet:** Helmet helps you secure your Express.js apps by setting various HTTP headers.
- **Morgan:** HTTP request logger middleware for node.js.
- **Multer:** Middleware for handling `multipart/form-data`.

After installing all the packages, our project folder must consist of index.js files along with package.json, package-lock.json and node_modules.

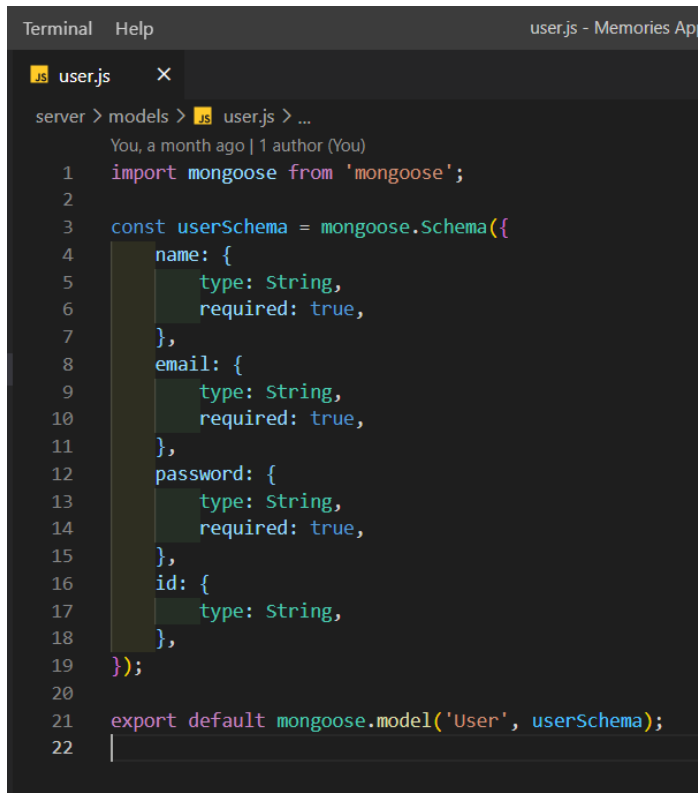
On Exploring package.json file you will find all your packages installed in dependencies as shown below.



```
1 {
2   "name": "mern-memories-app",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7
8   "scripts": {
9     "test": "echo \"Error: no test specified\" && exit 1",
10    "start": "node server/index.js",
11    "server": "nodemon server/index.js",
12    "client": "npm start --prefix client",
13    "dev": "concurrently \"npm run server\" \"npm run client\"",
14    "heroku-postbuild": "NPM_CONFIG_PRODUCTION=false npm install --prefix client && npm run build --prefix client"
15  },
16  "repository": {
17    "type": "git",
18    "url": "https://github.com/DishenMakwana/Memories-App"
19  },
20  "keywords": [],
21  "author": "Dishen Makwana",
22  "license": "ISC",
23  "dependencies": {
24    "bcryptjs": "^2.4.3",
25    "body-parser": "^1.19.0",
26    "concurrently": "^6.1.0",
27    "cors": "^2.8.5",
28    "dotenv": "^8.6.0",
29    "express": "^4.17.1",
30    "jsonwebtoken": "^8.5.1",
31    "mongoose": "^5.12.3",
32    "morgan": "^1.10.0",
33    "nodemon": "^2.0.7"
34  }
35 }
```

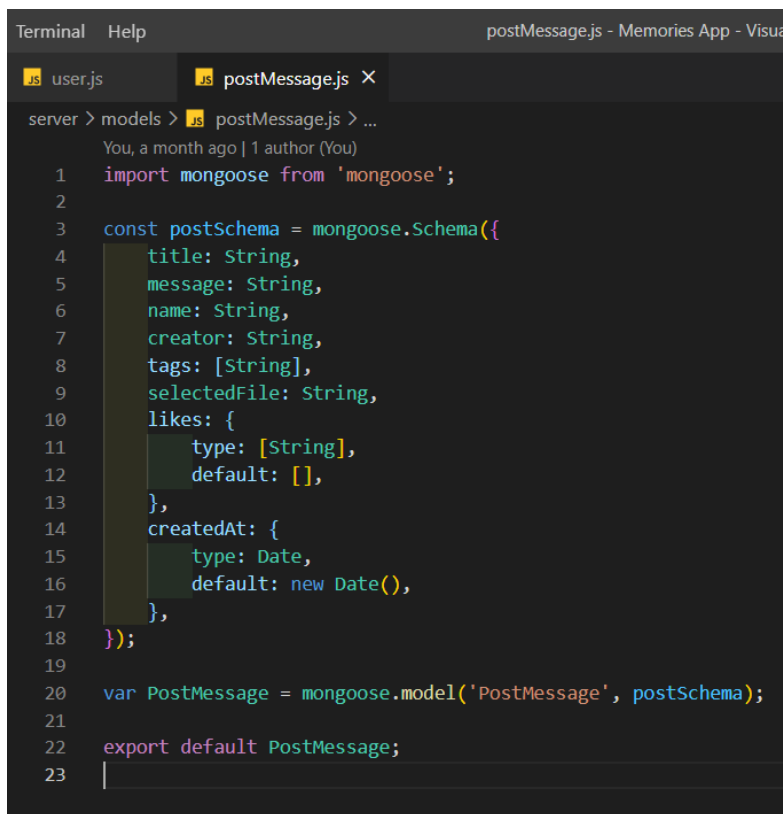
Now we need to create all the models that will connect our application to our database which is MongoDB Atlas cloud.

DataBase Models :



```
Terminal  Help  user.js - Memories App

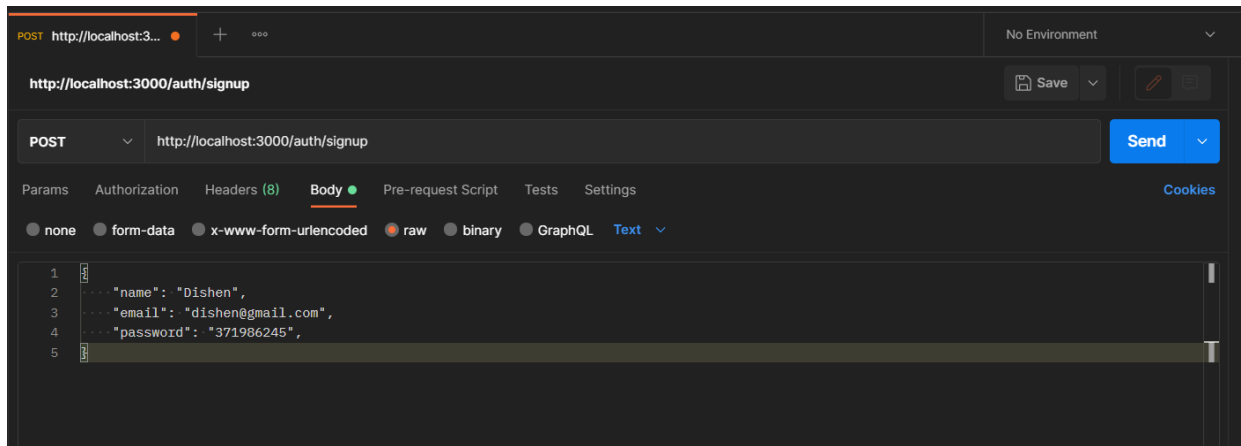
user.js  X
server > models > user.js > ...
You, a month ago | 1 author (You)
1  import mongoose from 'mongoose';
2
3  const userSchema = mongoose.Schema({
4    name: {
5      type: String,
6      required: true,
7    },
8    email: {
9      type: String,
10     required: true,
11   },
12   password: {
13     type: String,
14     required: true,
15   },
16   id: {
17     type: String,
18   },
19 });
20
21 export default mongoose.model('User', userSchema);
22 |
```



```
Terminal  Help  postMessage.js - Memories App - Visual Studio Code

user.js  postMessage.js  X
server > models > postMessage.js > ...
You, a month ago | 1 author (You)
1  import mongoose from 'mongoose';
2
3  const postSchema = mongoose.Schema({
4    title: String,
5    message: String,
6    name: String,
7    creator: String,
8    tags: [String],
9    selectedFile: String,
10    likes: {
11      type: [String],
12      default: [],
13    },
14    createdAt: {
15      type: Date,
16      default: new Date(),
17    },
18  });
19
20 var PostMessage = mongoose.model('PostMessage', postSchema);
21 export default PostMessage;
22 |
23 |
```

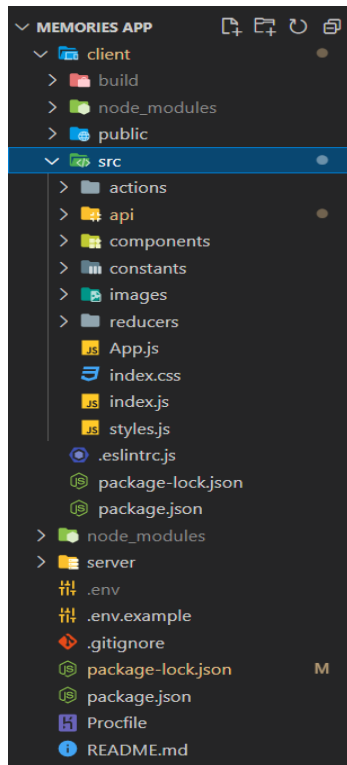
As here you can see I have created four Models:- User.js, PostMessage.js.
By this we define Schema of our collection.
After doing so we need to create routes of all schema to our database.



Building the frontend User interface using ReactJS:

In order to use React in production, you need NPM and Node.js installed. Now we need to create our app using:- `npx create-react-app chat-react`.

Our folder chat-react will be created and will look like this:



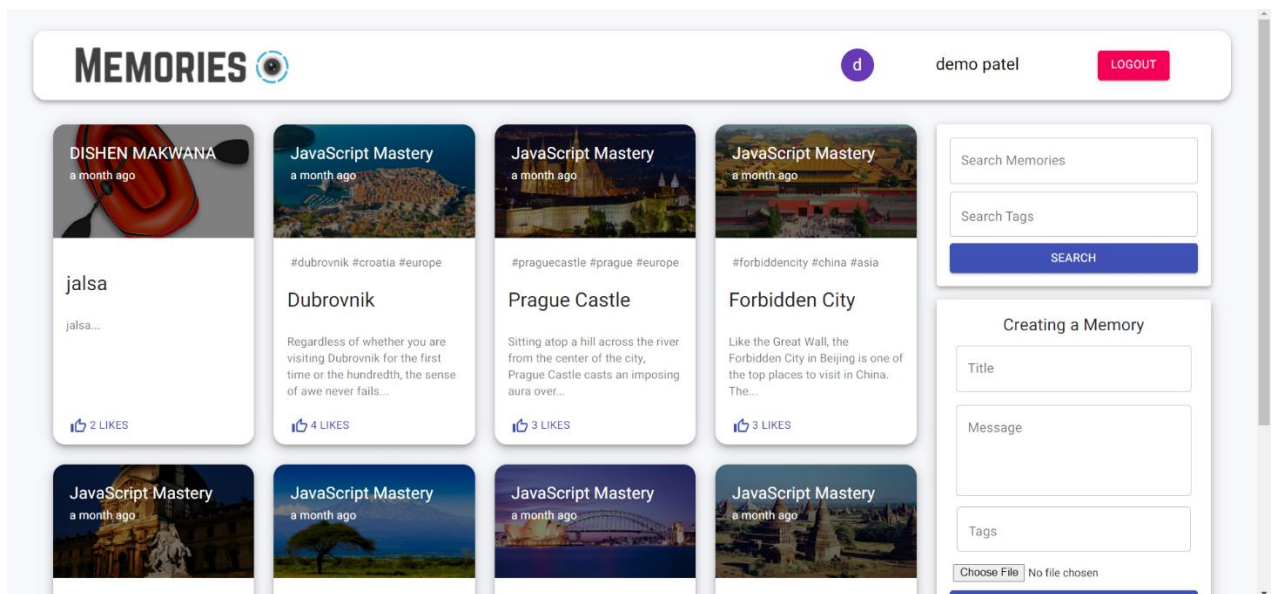
We will start developing the application by creating components of our Memories-App, so we can reuse them in any part of our application.

Components Descriptions:



- feed: This will be the main component which is the parent of two components, that are: share and post. This will fetch all the timeline posts from the database with the help of apis.
- post: This component will be under the feed component where the user can see his own posts which he has posted and also his friend's posts. This component will fetch

data from the post api and display it to the user. This will allow user's followers to like the post or to dislike the post.

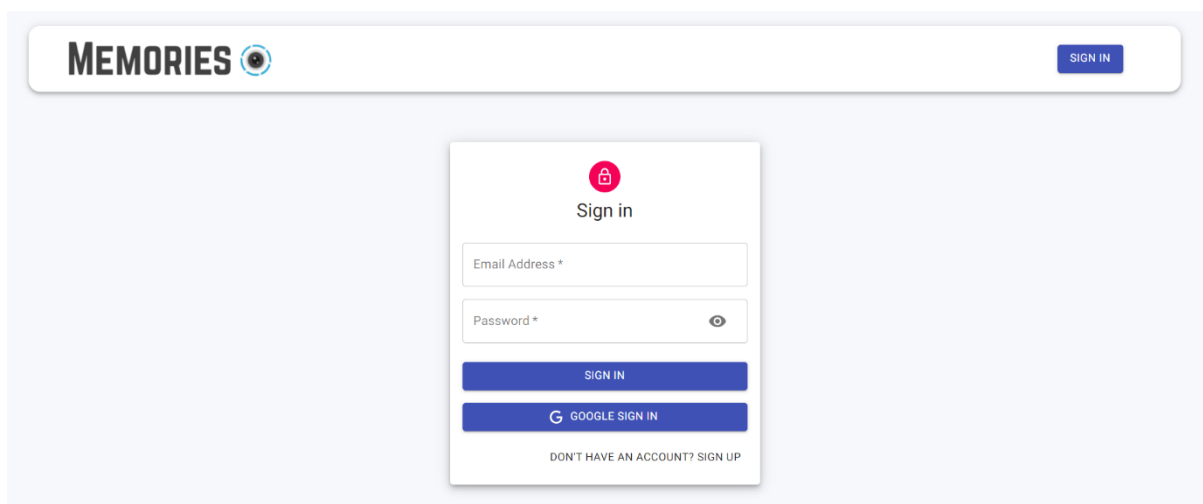


- share: This component will allow users to share the post along with post description and post image.

Pages:

There will be many web pages in Memories-App:- Home, Login, Profile, Register.

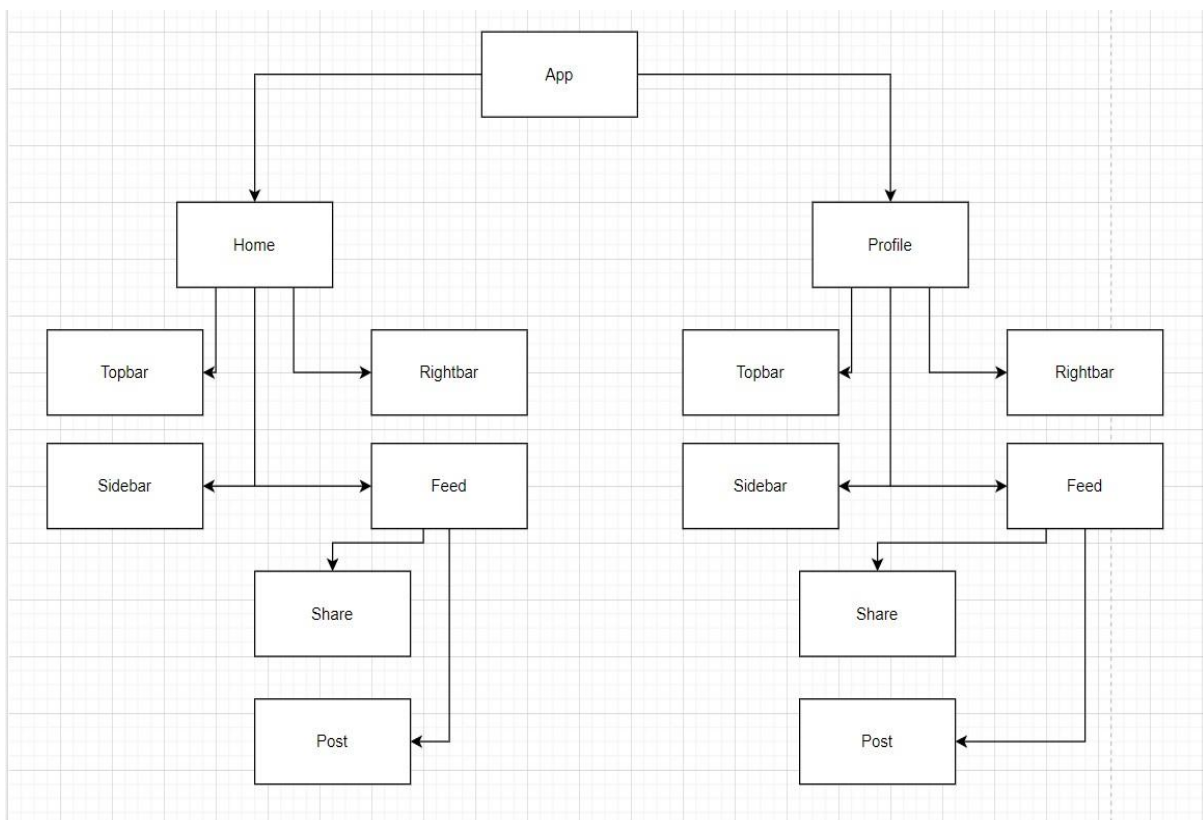
Login Page:



Register Page:

The screenshot shows the 'Sign up' page of the 'MEMORIES' app. The page has a light blue background. At the top left is the 'MEMORIES' logo with a camera icon. At the top right is a 'SIGN IN' button. The central form is titled 'Sign up' with a red lock icon. It contains the following fields: 'First Name *', 'Last Name *', 'Email Address *', 'Password *' (with an eye icon for toggling visibility), and 'Repeat Password *'. Below these fields are two buttons: 'SIGN UP' and 'GOOGLE SIGN IN'. At the bottom of the form, it says 'ALREADY HAVE AN ACCOUNT? SIGN IN'.

Flow Diagram:



This shows the flow diagram of Memories-App where user's info will be shared among all components.

Building the realtime update with socket.io:

Socket.IO enables real-time, bidirectional and event-based communication. It works on every platform, browser or device, focusing equally on reliability and speed.



The client will try to establish a WebSocket connection if possible and will fall back on HTTP long polling if not.

WebSocket is a communication protocol which provides a full-duplex and low-latency channel between the server and the browser. More information can be found [here](#).

So, in the best-case scenario, provided that:

- the browser supports WebSocket (97% of all browsers in 2020)
- there is no element (proxy, firewall, ...) preventing WebSocket connections between the client and the server you can consider the Socket.IO client as a “slight” wrapper around the WebSocket API.

References & Project links

- **GitHub** - <https://github.com/DishenMakwana/Memories-App>
- **Live** - <https://memories-app-dm.herokuapp.com/>
<https://memories-app-dm.netlify.app/>
- <https://nodejs.org/en/>
- <https://expressjs.com/>
- <https://www.w3schools.com/>
- <https://reactjs.org/>
- <https://socket.io/>
- <https://developer.mozilla.org/en-US/>