

1.INTRODUCTION

1.1 About Application:

Memories is a CRUD, responsive web app designed using MERN stack. It is a simple social media app where people can post pictures, edit and delete their post and, like other's posts.

1.2 Definition of the Problem:

This application was built to use my knowledge of MERN app in one application. This is a personal growth application, with the motivation to build something new. I used my lesson of file destructing(both frontend and backend), that is a part of a IT project in a company.

1.3 Cost Benefit Analysis:

This application requires Internet connection which is basically available with almost everyone. This app will also run on mobile phones without any compromising on its UI look on the end user's side. The application is safe, so no threat of virus. Besides this, it has no ads contained in it. So, there is no extra use of your data. In conclusion, there is hardly any cost beared by the user for using this application except for its internet service.

2. COMPONENTS USED

2.1 Software Used:

- VS Code 1.57.1: Editor by Microsoft for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
- Nodejs 14.17.0: open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.
- MongoDB cloud Atlas version : fully-managed non-relational cloud database where my details of post like creator, creation date, like count, post id, etc are stored
- Windows 10 OS

2.2 Hardware Used:

- HP i3 5th Generation Laptop

3. MERN STACK ENVIRONMENT

MERN stands for MongoDB, ExpressJS, React, NodeJs. It is full stack solution to build a 3-tier architecture application. The main language used here is Javascript for both frontend and backend.

- **Mongodb** : Mongo DB is NoSQL, document base database. It is quite different from the traditional relational database where the data is stored in the form of rows and columns. Mongo DB is highly adaptable and scalable that is why it is used in volatile and rapid development process. Mongo DB stores data in the form of documents.
- **Express JS** : Express JS is a light weight framework designed for node JS to develop rapid applications.
- **React** : React JS is a front-end library developed by Facebook which works on the concept of virtual DOM rendering.
- **Node JS**: Node JS is the technology using which JavaScript can be used at server side also.

MERN stack is basically used in the startup culture where the requirements are either clearly defined or very much volatile. MERN is famous for its adaptably and easy maintenance of codebase.

3.1 Dependencies included :

Frontend packages imports:

- **@material-ui/core**: design beautiful react app fast
- **@material-ui/icons**: to show icons like password, lock etc, to make UI more attractive
- **axios**: make API requests
- **jwt-decode**: to decode jwt tokens, to set and check token's expiry
- **moment**: library to work with time and date
- **react**: package to build interface in react app
- **react-dom**: provides **DOM** specific methods that can be used at the top level of a web app to enable an efficient way of managing **DOM** elements of the web page.
- **react-file-base64**: convert image to string
- **react-google-login**: use google sign in react app

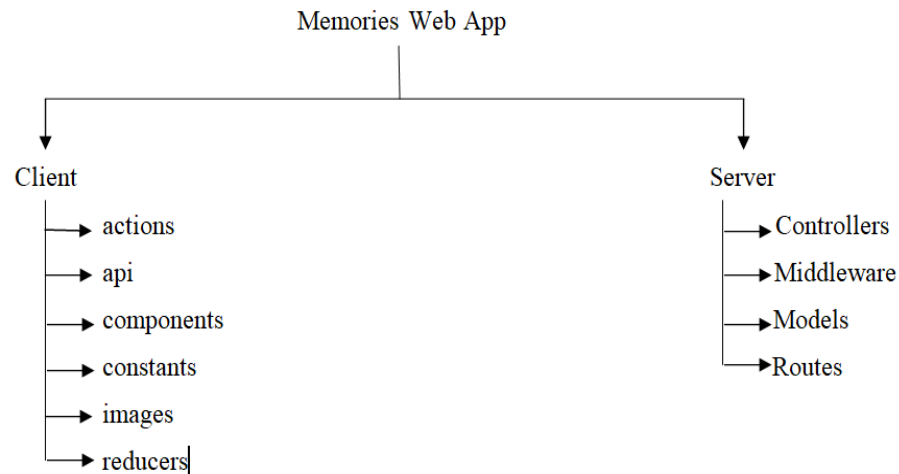
- react-redux: lets your React components read data from a Redux store, and dispatch actions to the store to update state.
- react-router-dom: handles dynamic routing
- redux: updates application state using actions
- redux-thunk: used for asynchronous actions using redux

Backend packages imports:

- bcryptjs: it hashes our password to increase security. The password can be decrypted using its compare() method.
- body-parser: send post request
- cors: enable cross-origin request
- dotenv: to store sensitive info like database connection credentials so that when committing to git, this file is not uploaded, but our app can still has access to by importing this file where that data is required to make connection.
- express: framework to create routing of app
- jsonwebtoken: securely transfer payloads between parties using a token
- mongoose: create models for our post
- nodemon: automatically updates changes of server without manually being done by us.

4. FILE DESTRUCTURING

Before starting any kind of project mind making and clear design is one of the key features for efficient product development. For this project I have wireframed the structure of the project by taking care all of its functionality and core parts.



(fig 4.1. File destructuring)

Server follows the order :

1. middleware: this will contain logic that will be executed first before we move on our next action. In our case , it has logic of authentication, before creating, updating or deleting a post, or liking a post , we first check if user is logged in. Until this logic is runs to success you won't be allowed to move above mentioned events.
2. routes : here we create routes for our backend application, which is used by express middleware to connect these routes to our application.
3. Controllers: It contains handlers for our routes to increase scalability and readability. In other words, instead of cluttering the routes of each feature with their logic written in the same route statement only, we pass that logic inside a function written in a file of controller, and call that corresponding function from the route file. This makes the code look cleaner,

Client follows the order :

1. api : code to implement calls to api
2. actions : we create actions, which is a function that returns an action. An action must have 'type' property and 'payload' which basically some data. This type is used by reducers.
3. reducers : we add redux capability because all actions and backend work is done by redux, it will dispatch the action. In simpler terms, it is a function that accepts state and action. Based on action type, we run a logic ,i.e., we return either action or a state changed by the action .

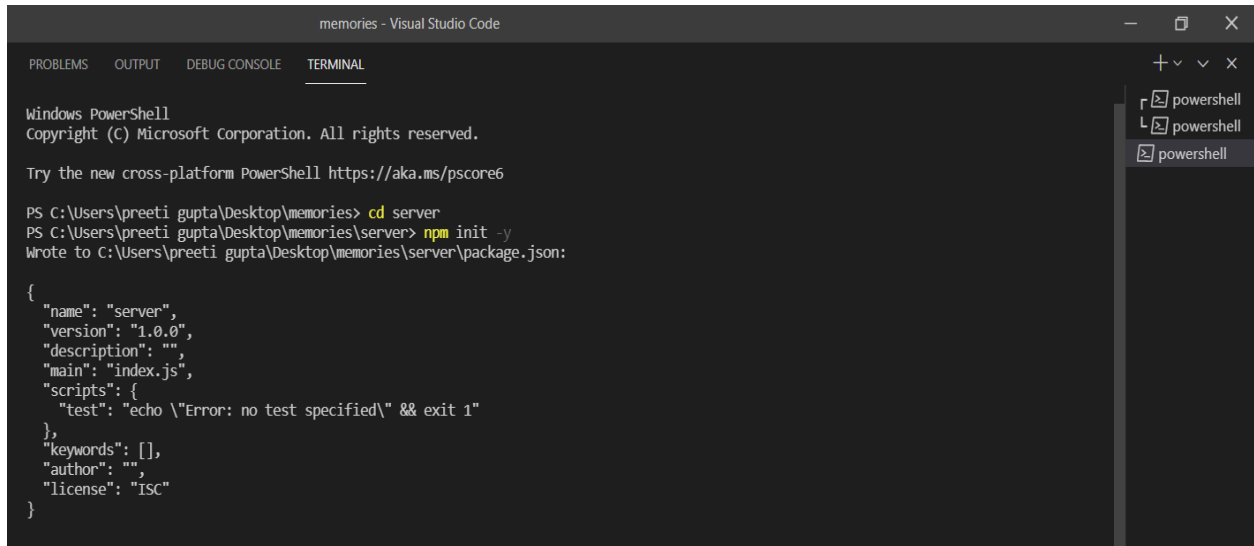
components : folder structure that works well in medium or large application. Components of my app are Form, Posts and Post. If one component uses another component, then the one being used is put inside the one using it. Here , Post folder is kept inside Posts folder.

images : store images used by our app while designing.

constants : store constant values so as to avoid errors arising from typos.

5. PROJECT SETUP

Backend Setup :



The screenshot shows a Visual Studio Code window with a terminal pane open. The terminal is running Windows PowerShell. The user has navigated to the 'server' directory and initialized a new Node.js project using 'npm init -y'. The terminal displays the contents of the generated 'package.json' file, which includes fields for name, version, description, main, scripts, keywords, author, and license.

```
memories - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

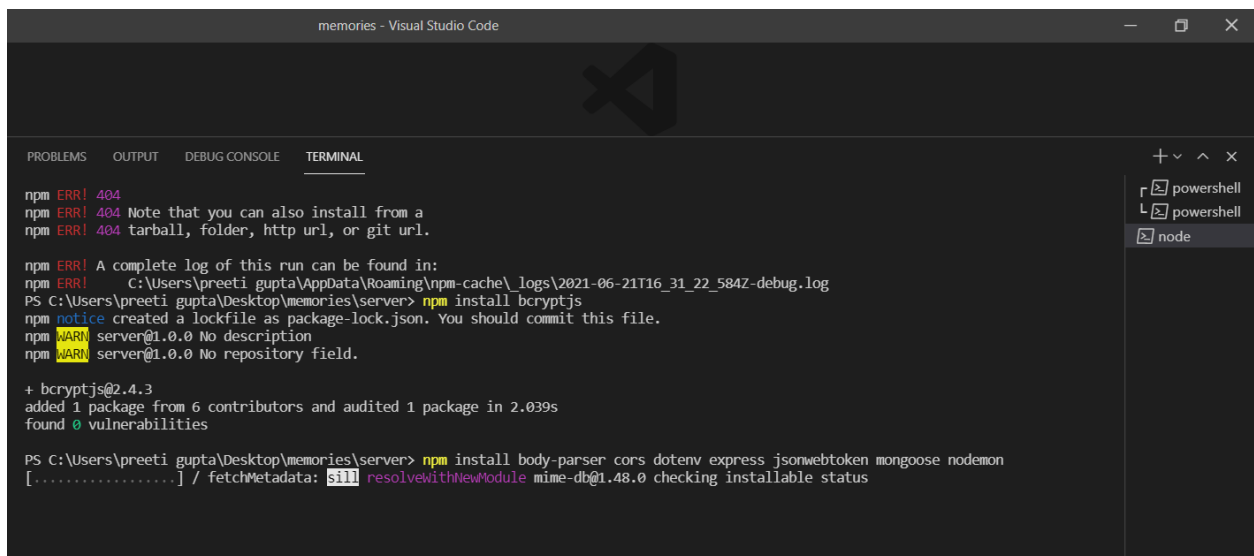
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\preeti gupta\Desktop\memories> cd server
PS C:\Users\preeti gupta\Desktop\memories\server> npm init -y
Wrote to C:\Users\preeti gupta\Desktop\memories\server\package.json:

{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

(fig 5.1. Setup node to get all node_modules folder)



The screenshot shows the same Visual Studio Code window with the terminal pane. The user has run 'npm install bcryptjs' and then 'npm install body-parser cors dotenv express jsonwebtoken mongoose nodemon'. The terminal output shows the installation progress, including the creation of a lockfile and the addition of packages to the project.

```
memories - Visual Studio Code

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

npm ERR! 404
npm ERR! 404 Note that you can also install from a
npm ERR! 404 tarball, folder, http url, or git url.

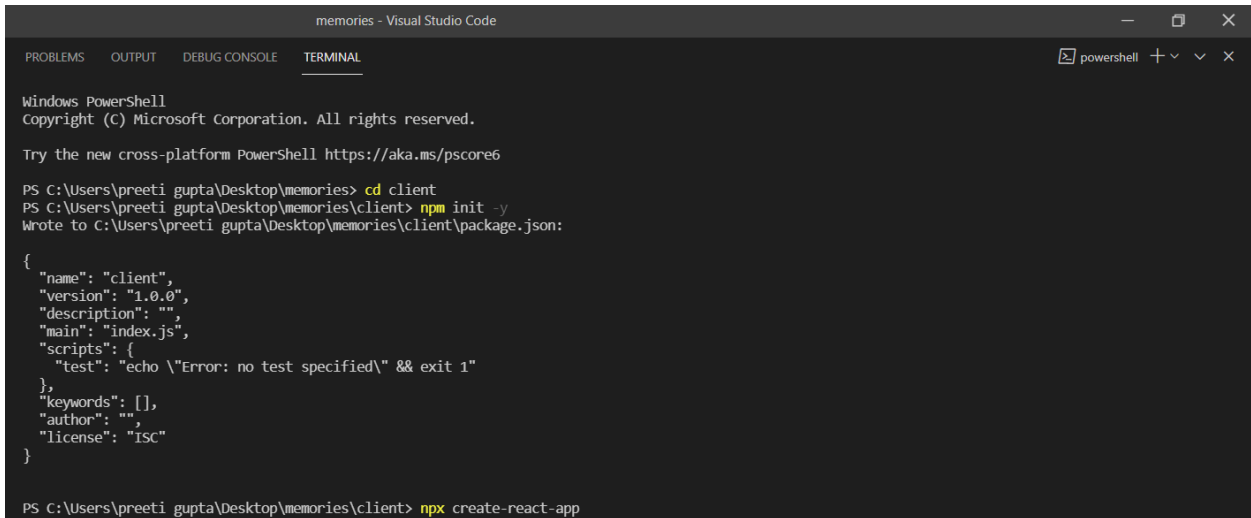
npm ERR! A complete log of this run can be found in:
npm ERR! C:\Users\preeti gupta\AppData\Roaming\npm-cache\logs\2021-06-21T16_31_22_584Z-debug.log
PS C:\Users\preeti gupta\Desktop\memories\server> npm install bcryptjs
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN server@1.0.0 No description
npm WARN server@1.0.0 No repository field.

+ bcryptjs@2.4.3
added 1 package from 6 contributors and audited 1 package in 2.039s
found 0 vulnerabilities

PS C:\Users\preeti gupta\Desktop\memories\server> npm install body-parser cors dotenv express jsonwebtoken mongoose nodemon
[.....] / fetchMetadata: sill resolveWithNewModule mime-db@1.48.0 checking installable status
```

(fig 5.2. Adding all server dependencies)

Frontend Setup:



```
memories - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

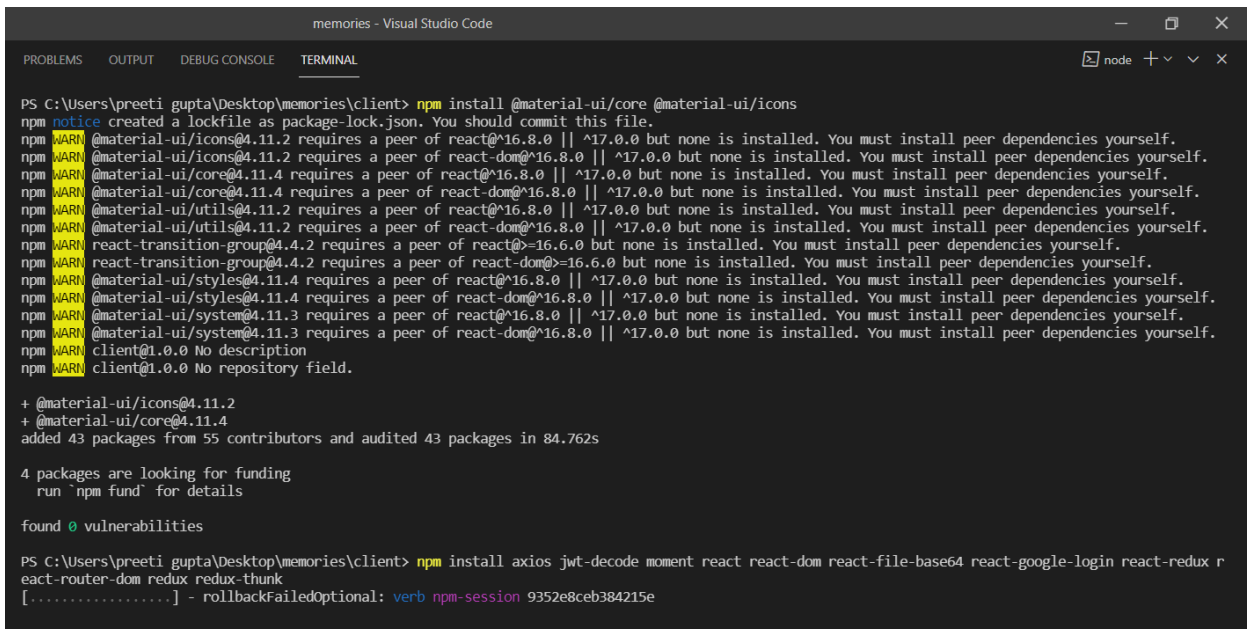
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\preeti gupta\Desktop\memories> cd client
PS C:\Users\preeti gupta\Desktop\memories\client> npm init -y
Wrote to C:\Users\preeti gupta\Desktop\memories\client\package.json:

{
  "name": "client",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS C:\Users\preeti gupta\Desktop\memories\client> npx create-react-app
```

(fig 5.3. Setting up node for client)



```
memories - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
node
PS C:\Users\preeti gupta\Desktop\memories\client> npm install @material-ui/core @material-ui/icons
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN @material-ui/icons@4.11.2 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/icons@4.11.2 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/core@4.11.4 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/core@4.11.4 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/utils@4.11.2 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/utils@4.11.2 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN react-transition-group@4.4.2 requires a peer of react-dom@^16.6.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/styles@4.11.4 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/styles@4.11.4 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/system@4.11.3 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN @material-ui/system@4.11.3 requires a peer of react-dom@^16.8.0 || ^17.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN client@1.0.0 No description
npm WARN client@1.0.0 No repository field.

+ @material-ui/icons@4.11.2
+ @material-ui/core@4.11.4
added 43 packages from 55 contributors and audited 43 packages in 84.762s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

PS C:\Users\preeti gupta\Desktop\memories\client> npm install axios jwt-decode moment react react-dom react-file-base64 react-google-login react-redux r
eact-router-dom redux redux-thunk
[.....] - rollbackFailedOptional: verb npm-session 9352e8ceb384215e
```

(fig 5.4. Adding all client dependencies)

6. DATABASE SCHEMA

Mongo DB is a NoSQL database which consists of documents. Here the DB is known as collections. So, in the project I have created two schemas or collections. These are user and postMessages.

‘postMessages’ Schema schema has fields like title, message, name, creator, tags, selectedFile, likes and createdAt.

```
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:[Number],
12     default:[]
13   },
14   createdAt:{
15     type: Date,
16     default: new Date()
17   }
18 })
19 );
20
```

(fig 6.1. postMessages schema)

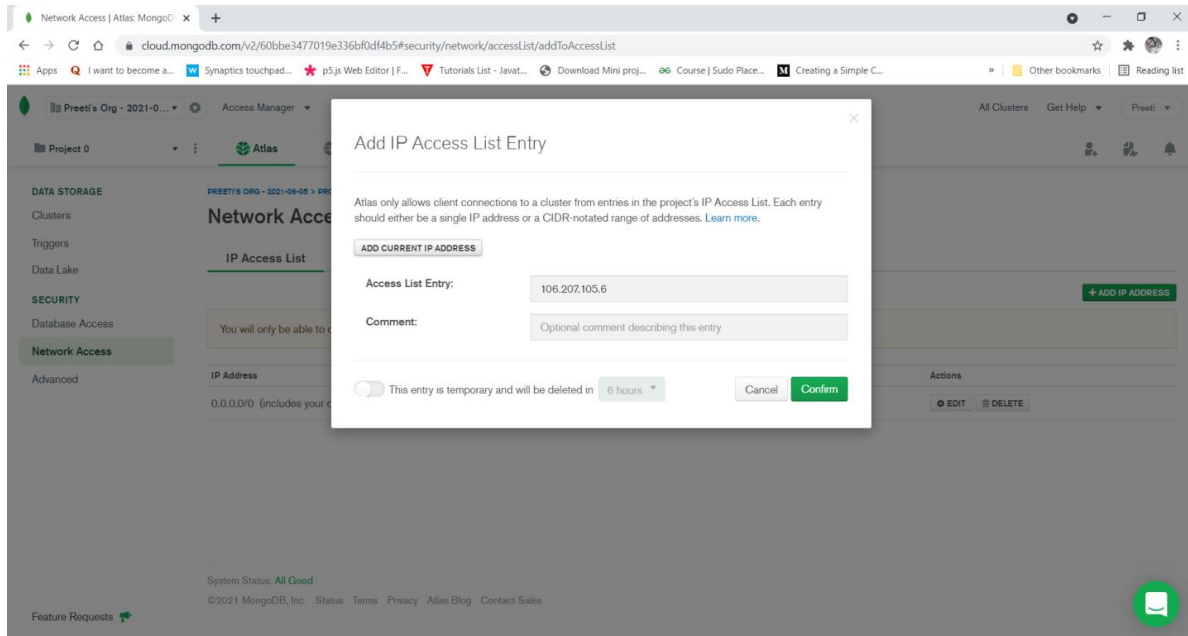
‘user’ schema has fields like name, email, password, id.

```
2
3  const userSchema = mongoose.Schema({
4    name : {type:String,required:true},
5    email : {type:String,required:true},
6    password : {type:String,required:true},
7    id : {type:String}
8  })
9
```

(fig 6.2. User schema)

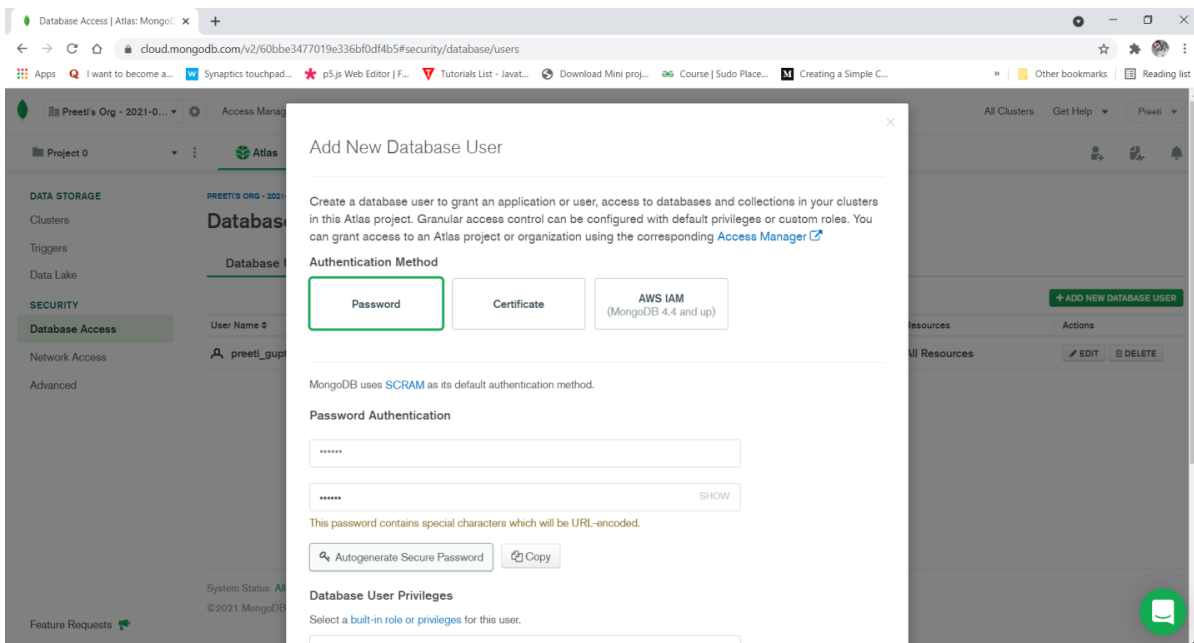
Mongodb setup :

1. Under Network Access, add IP address -> add current IP address -> confirm



(fig 6.3. Setting network access)

2. Under Database Access , add new database user -> enter password -> add user



(fig 6.4. Setting database access)

3. Finally under clusters, click create clusters, which takes few minutes to complete.

7. SOURCE CODE

SERVER CODE:

controllers/user.js:

```
import mongoose from 'mongoose';

const userSchema = mongoose.Schema({
  name : {type:String,required:true},
  email : {type:String,required:true},
  password : {type:String,required:true},
  id : {type:String}
})

export default mongoose.model("User",userSchema);
```

controllers/posts.js:

```
import PostMessage from '../models/postMessages.js';
import mongoose from 'mongoose';
import express from 'express';

const router = express.Router();

export const getPosts = async(req,res)=>{
  try {
    const postMessages= await PostMessage.find();
    res.status(200).json(postMessages);

  } catch (error) {
    res.status(404).json({message:error.message});
  }
}

export const createPost = async (req,res)=>{
  const post = req.body;
  const newPost= new PostMessage({...post,creator:req.userId,createdAt: new Date().toISOString()});
  try {
    await newPost.save();
    //https://www.restapitutorial.com/httpstatuscodes.html - check http codes meaning
    res.status(201).json(newPost);
  } catch (error) {
    res.status(409).json({message:error.message});
  }
}

export const updatePost = async(req, res) =>{
  const {id : _id} =req.params;
  const post= req.body;

  if(!mongoose.Types.ObjectId.isValid(_id)) return res.status(404).send("No Post with that ID exists");
  const updatedPost = await PostMessage.findByIdAndUpdate(_id,{...post,_id},{new:true});
  res.json(updatedPost);
}

export const deletePost = async (req,res) =>{
  const {id } =req.params;
  if(!mongoose.Types.ObjectId.isValid(id)) return res.status(404).send("No Post with that ID exists");
  await PostMessage.findByIdAndRemove(id);
  res.json({message: 'Post deleted successfully'});
}

export const likePost =async (req,res)=>{
  const {id} =req.params;
```

```
if(!req.userId) return res.json({message:'User not authenticated. Please sign in before you like the post.'});

if(!mongoose.Types.ObjectId.isValid(id)) return res.status(404).send("No Post with that ID exists");
const post = await PostMessage.findById(id);

//check if user id is in list of likes in the post
const index = post.likes.findIndex((id)=> id === String(req.userId));

if( index === -1) //if user not present in like list, allow him to like
{
  post.likes.push(req.userId);
}
else //he can only dislike the post
{
  post.likes = post.likes.filter((id)=> id !== String(req.userId));
}

const updatedPost = await PostMessage.findByIdAndUpdate(id,post,{new:true});
res.json(updatedPost);
}

export default router;
```

middleware/auth.js:

```
import React,{useState} from 'react';
import {Avatar,Button,Paper,Container,Grid,Typography} from '@material-ui/core';
import useStyles from './styles';
import LockOutlinedIcon from '@material-ui/icons/LockOutlined';
import Input from './Input';
import {GoogleLogin} from 'react-google-login';
import Icon from './icon';
import {useDispatch} from 'react-redux';
import {useHistory} from 'react-router-dom';
import {signup,signin} from '../actions/auth';

const initialState = {firstName:'',lastName:'',email:'',password:'',confirmPassword:''};

const Auth = () => {
  const classes = useStyles();
  const [showPassword,setShowPassword] = useState(false);
  const [formData,setFormData] = useState(initialState);
  const [isSignUp, setisSignUp] = useState(false);
  const dispatch=useDispatch();
  const history =useHistory();

  const handleShowPassword = () => setShowPassword(!showPassword);

  const handleSubmit =(e)=>{
    e.preventDefault();
    if(isSignUp){
      dispatch(signup(formData,history));
    }else{
      dispatch(signin(formData,history));
    }

    console.log(formData);
  };

  const handleChange = (e)=>{
    setFormData({...formData,[e.target.name]:e.target.value});
  };

  const googleSuccess = async (res)=>{
    // console.log(res);
  }
}
```

VSSUT 2021 pg. 13

```
        onFailure={googleFailure}
        cookiePolicy="single_host_origin"
      />
      <Grid container justify="flex-end" >
        <Button onClick={switchMode}>
{isSignUp ? 'Already have an account? Sign In' : "Don't have an account? Sign Up"}
        </Button>

      </Grid>
    </form>
  </Paper>
</Container>
)
}

export default Auth
```

routes/posts.js:

```
import express from 'express';
import {getPosts,createPost,updatePost,deletePost,likePost} from '../controllers/posts.js';
import auth from '../middleware/auth.js';

const router= express.Router();
router.get('/',getPosts);
router.post('/',auth,createPost);
router.patch('/:id',auth,updatePost);
router.delete('/:id',auth,deletePost);
router.patch('/:id/likePost',auth,likePost)
export default router;
```

routes/users.js:

```
import express from 'express';
import {signin,signup} from '../controllers/user.js';

const router= express.Router();

router.post('/signup',signup);
router.post('/signin',signin);

export default router;
```

models/postMessages.js:

```
import mongoose from 'mongoose';

const postSchema = mongoose.Schema({
  title: String,
  message: String,
  name:String,
  creator: String,
  tags: [String],
  selectedFile: String,
  likes: {
    type:[Number],
    default:[]
  },
  createdAt:{
    type: Date,
    default: new Date()
  }
});

//convert schema to model
const PostMessge = mongoose.model('PostMessage',postSchema);

export default PostMessge;
```

models/user.js:

```
import mongoose from 'mongoose';

const userSchema = mongoose.Schema({
  name : {type:String,required:true},
  email : {type:String,required:true},
  password : {type:String,required:true},
  id : {type:String}
})

export default mongoose.model("User",userSchema);
```

server/index.js:

```
import express from 'express';
import bodyParser from 'body-parser';
import cors from 'cors';
import mongoose from 'mongoose';
import postRoutes from './routes/posts.js';
import userRoutes from './routes/users.js';
import dotenv from 'dotenv';

const app=express();
dotenv.config();

app.use(bodyParser.json({limit:"30mb", extended:true}));
app.use(bodyParser.urlencoded({limit:"30mb", extended:true}));
app.use(cors());
app.use('/posts',postRoutes);
app.use('/user',userRoutes);

app.get('/',(req,res)=>{
  res.send('Welcome to Memories API');
});

//https://www.mongodb.com/cloud/atlas
const PORT = process.env.PORT || 5000;

mongoose.connect(process.env.CONNECTION_URL,{useNewUrlParser:true,useUnifiedTopology:true})
.then(()=>app.listen(PORT,()=>console.log(`Server running on port : ${PORT}`)))
.catch((error)=>console.log(error.message));
mongoose.set('useFindAndModify',false);
```

CLIENT CODE:

actions/auth.js:

```
import * as api from '../api';
import {AUTH} from '../constants/actionTypes';

export const signin = (formData,history) => async(dispatch) =>{
  try {
    //log in the user
    const {data} = await api.signIn(formData);
    dispatch({type:AUTH,data});

    history.push('/');
  } catch (error) {
    console.log(error);
  }
};

export const signup = (formData,history) => async(dispatch) =>{
  try {
    //sign up the user
    const {data} = await api.signUp(formData);
    dispatch({type:AUTH,data});

    history.push('/');
  } catch (error) {
    console.log(error);
  }
}
```

actions/posts.js:

```
import * as api from '../api';
import {CREATE, FETCH_ALL,UPDATE,LIKE,DELETE} from '../constants/actionTypes';
//Action Creators
export const getPosts=()=>async (dispatch)=>{
  try {
    const {data} = await api.fetchPosts();
    // const action ={type:"FETCH_ALL",payload:[]};
    // return action;
    dispatch({type:FETCH_ALL,payload:data});
  } catch (error) {
    console.log(error.message);
  }
}

export const createPost = (post) => async(dispatch) =>{
  try {
    const {data} = await api.createPost(post);
    dispatch({type:CREATE,payload:data});

  } catch (error) {
    console.log(error);
  }
}

export const updatePost = (id,post) =>async(dispatch) => {
  try {
    const {data} = await api.updatePost(id,post);
    dispatch({type:UPDATE,payload:data});

  } catch (error) {
    console.log(error);
  }
}
```



```
}

export const deletePost=(id) => async(dispatch) =>{
  try {
    await api.deletePost(id);
    dispatch({type:DELETE,payload:id});
  } catch (error) {
    console.log(error);
  }
}

export const likePost = (id) => async(dispatch) =>{
  try
  {
    const {data} = await api.likePost(id);
    dispatch({type:LIKE,payload:data});

  } catch (error) {
    console.log(error);
  }
}
```

api/index.js:

```
import axios from 'axios';

const API = axios.create({baseUrl:'http://localhost:5000'});
const url = 'https://memories-master.herokuapp.com/posts';

API.interceptors.request.use((req)=>{
  if(localStorage.getItem('profile')){
    req.headers.Authorization = `Bearer ${JSON.parse(localStorage.getItem('profile')).token}`;
  }

  return req;
})

export const fetchPosts = ()=> API.get('/posts');
export const createPost= (newPost)=> API.post('/posts',newPost);
export const updatePost = (id,updatedPost) => API.patch(`/posts/${id}`,updatedPost);
export const deletePost = (id) => API.delete(`/posts/${id}`);
export const likePost = (id) => API.patch(`/posts/${id}/likePost`);

export const signIn = (formData) => API.post('/user/signin',formData);
export const signUp = (formData) => API.post('/user/signup',formData);
```

components/Auth/auth.js:

```
import React,{useState} from 'react';
import {Avatar,Button,Paper,Container,Grid,Typography} from '@material-ui/core';
import useStyles from './styles';
import LockOutlinedIcon from '@material-ui/icons/LockOutlined';
import Input from './Input';
import {GoogleLogin} from 'react-google-login';
import Icon from './icon';
import {useDispatch} from 'react-redux';
import {useHistory} from 'react-router-dom';
import {signup,signin} from '../../actions/auth';

const initialState = {firstName:'',lastName:'',email:'',password:'',confirmPassword:''};

const Auth = () => {
  const classes = useStyles();
  const [showPassword,setShowPassword] = useState(false);
  const [formData,setFormData] = useState(initialState);
  const [isSignUp, setisSignUp] = useState(false);
```

```

const dispatch=useDispatch();
const history =useHistory();

const handleShowPassword = () => setShowPassword(!showPassword);

const handleSubmit =(e)=>{
  e.preventDefault();
  if(isSignUp){
    dispatch(signup(formData,history));

  }else{

    dispatch(signin(formData,history));
  }

  console.log(formData);
};

const handleChange = (e)=>{

  setFormData({...formData,[e.target.name]:e.target.value});
};

const googleSuccess = async (res)=>{
  // console.log(res);

  const result = res?.profileObj;
  const token = res?.tokenId;

  try {
    dispatch({type:'AUTH' ,data:{result,token}});
    history.push('/');
  } catch (error) {
    console.log(error);
  }
};

const googleFailure = (error)=>{

  console.log("Google Log In Failed. Try again later");
  console.log(error);
};

const switchMode = ()=>{
  setFormData(initialState);
  setIsSignUp((prevIsSignUp)=>!prevIsSignUp);
  setShowPassword(false);
};

return (
  <Container component="main" maxWidth="xs" >
    <Paper className={classes.paper} elevation={3} >
      <Avatar className={classes.avatar}>
        <LockOutlinedIcon/>
      </Avatar>
      <Typography variant="h5">{isSignUp? "Sign Up" : "Sign In"}</Typography>
      <form className={classes.form} onSubmit={handleSubmit} >
        <Grid container spacing={2} >
          {
            isSignUp && (
              <>
                <Input name="firstName" label="First Name" handleChange={handleChange} autoFocus
us half/>
                <Input name="lastName" label="Last Name" handleChange={handleChange} half />
              </>
            )
          }
        </Grid>
      </form>
    </Paper>
  </Container>
);

```

MEMORIES MERN CRUD APP

```

        <Input name="email" label="Email Address" type="email" handleChange={handleChange} />
        <Input name="password" label="Password" type={showPassword ? "text" : "password"} han
leChange={handleChange} handleShowPassword={handleShowPassword} />
        {isSignUp && <Input name="confirmPassword" label="Confirm Password" handleChange={hand
leChange} type="password" />}
      </Grid>
      <Button type="submit" fullWidth variant="contained" className={classes.submit} color="
primary" >
        {isSignUp ? 'Sign Up' : 'Sign In'}
      </Button>
      <GoogleLogin
        clientId="556873350930-98veh1ii1332fk1css4okckernii992j.apps.googleusercontent.com"
        render={
          (renderProps) => (
            <Button className={classes.googleButton} color="primary" fullWidth onClick={re
nderProps.onClick} disabled={renderProps.disabled} startIcon={<Icon/>}
              variant="contained" >
                Google Sign In
              </Button>
            </Button>
          )
        }
        onSuccess={googleSucess}
        onFailure={googleFailure}
        cookiePolicy="single_host_origin"
      />
      <Grid container justify="flex-end" >
        <Button onClick={switchMode}>
{isSignUp ? 'Already have an account? Sign In' : "Don't have an account? Sign Up"}
        </Button>
      </Grid>
    </form>
  </Paper>
</Container>
)
}

export default Auth
```

components/Auth/icon.js:

```
import React from 'react';

const icon = () => (
  <svg style={{ width: '20px', height: '20px' }} viewBox="0 0 24 24">
    <path
      fill="currentColor"
      d="M21.35,11.1H12.18V13.83H18.69C18.36,17.64 15.19,19.27 12.19,19.27C8.36,19.27 5,16.25 5,12C5,7.9 8
.2,4.73 12.2,4.73C15.29,4.73 17.1,6.7 17.1,6.7L19,4.72C19,4.72 16.56,2 12.1,2C6.42,2 2.03,6.8 2.03,12C2.03
,17.05 6.16,22 12.25,22C17.6,22 21.5,18.33 21.5,12.91C21.5,11.76 21.35,11.1 21.35,11.1V11.1Z"
    />
  </svg>
);

export default icon;
```

components/Auth/Input.js:

```
import React from 'react';
import {TextField,Grid,InputAdornment,IconButton} from '@material-ui/core';
import Visibility from '@material-ui/icons/Visibility';
import VisibilityOff from '@material-ui/icons/VisibilityOff';
```

```
const Input = ({name, handleChange, label, half, autoFocus, type, handleShowPassword}) => (  
  
  <Grid item xs={12} sm={half ? 6:12} >  
    <TextField  
      name={name}  
      onChange={handleChange}  
      variant="outlined"  
      required  
      fullWidth  
      label={label}  
      autoFocus={autoFocus}  
      type={type}  
      InputProps = {name === 'password' ? {  
        endAdornment:(  
          < InputAdornment position="end">  
            <IconButton onChange={handleShowPassword}>  
              {type === "password" ? <Visibility/> : <VisibilityOff/> }  
            </IconButton>  
          </InputAdornment>  
        )  
      } : null }  
    </Grid>  
  </Grid>  
  </Grid>  
);  
  
export default Input;
```

components/Auth/styles.js:

```
import { makeStyles } from '@material-ui/core/styles';  
  
export default makeStyles((theme) => ({  
  paper: {  
    marginTop: theme.spacing(8),  
    display: 'flex',  
    flexDirection: 'column',  
    alignItems: 'center',  
    padding: theme.spacing(2),  
  },  
  root: {  
    '& .MuiTextField-root': {  
      margin: theme.spacing(1),  
    },  
  },  
  avatar: {  
    margin: theme.spacing(1),  
    backgroundColor: theme.palette.secondary.main,  
  },  
  form: {  
    width: '100%', // Fix IE 11 issue.  
    marginTop: theme.spacing(3),  
  },  
  submit: {  
    margin: theme.spacing(3, 0, 2),  
  },  
  googleButton: {  
    marginBottom: theme.spacing(2),  
  },  
}));
```

components/Form/Form.js:

```
import React, { useState,useEffect } from "react";
import useStyles from "../styles";
import { TextField, Button, Typography, Paper } from "@material-ui/core";
import FileBase from "react-file-base64";
import {useDispatch,useSelector} from 'react-redux';
import {createPost,updatePost} from '../actions/posts'

const Form = ({currentId,setCurrentId}) => {
  const post= useSelector((state)=> currentId ? state.posts.find((p)=>p._id === currentId):null);
  const classes = useStyles();
  const user = JSON.parse(localStorage.getItem('profile'));
  const dispatch = useDispatch();
  const [postData, setPostData] = useState({
    title: "",
    message: "",
    tags: "",
    selectedFile: "",
  });

  useEffect(()=>{
    if(post) setPostData(post);
  },[post]);

  const handleSubmit = (e) => {
    e.preventDefault();
    if(currentId){
      dispatch(updatePost(currentId,{...postData,name: user?.result?.name}));
    }
    else{
      dispatch(createPost({...postData,name: user?.result?.name}));
    }
    clear();
  };

  const clear = () => {
    setCurrentId(null);
    setPostData({
      title: "",
      message: "",
      tags: "",
      selectedFile: "",
    });
  };

  if(!user?.result?.name){
    return(
      <Paper className={classes.paper} >
        <Typography variant="h6" align="center" >
          Please Sign In to create your memory or to like someone's post.
        </Typography>
      </Paper>
    )
  };

  return (
    <Paper className={classes.paper}>
      <form
        autoComplete="off"
        noValidate
        className={` ${classes.root} ${classes.form}`}
        onSubmit={handleSubmit}
      >
```

```

<Typography variant="h6">{currentId ? 'Editing' : 'Creating'} a Memory</Typography>
{ /* <TextField
  name="creator"
  variant="outlined"
  label="Creator"
  fullWidth
  value={postData.creator}
  onChange={e =>
    setPostData({ ...postData, creator: e.target.value })
  }
*/ }

<TextField
  name="title"
  variant="outlined"
  label="Title"
  fullWidth
  value={postData.title}
  onChange={e => setPostData({ ...postData, title: e.target.value })}
/>

<TextField
  name="message"
  variant="outlined"
  label="Message"
  fullWidth
  value={postData.message}
  onChange={e =>
    setPostData({ ...postData, message: e.target.value })
  }
/>

<TextField
  name="tags"
  variant="outlined"
  label="Tags(Comma Seperated)"
  fullWidth
  value={postData.tags}
  onChange={e => setPostData({ ...postData, tags: e.target.value.split(',')})}
/>

<div className={classes.fileInput}>
  <FileBase
    type="file"
    multiple={false}
    onDone={({ base64 }) =>
      setPostData({ ...postData, selectedFile: base64 })
    }
  />
</div>
  <Button className={classes.buttonSubmit} variant="contained" color="primary" size="large"
type="submit" fullWidth>
    Submit
  </Button>
  <Button variant="contained" color="secondary" size="small" onClick={clear} fullWidth>
    Clear
  </Button>

</form>
</Paper>
);
};
export default Form;

```

components/Form/Form.js:

```
import { makeStyles } from '@material-ui/core/styles';

export default makeStyles((theme) => ({
  root: {
    '& .MuiTextField-root': {
      margin: theme.spacing(1),
    },
  },
  paper: {
    padding: theme.spacing(2),
  },
  form: {
    display: 'flex',
    flexWrap: 'wrap',
    justifyContent: 'center',
  },
  fileInput: {
    width: '97%',
    margin: '10px 0',
  },
  buttonSubmit: {
    marginBottom: 10,
  },
})));
```

components/Home/Home.js:

```
import React, { useState, useEffect } from 'react';
import { Container, Grow, Grid } from '@material-ui/core';
import Posts from '../Posts/Posts';
import Form from '../Form/Form';
import { useDispatch } from 'react-redux';
import { getPosts } from '../actions/posts';
import useStyles from '../Posts/styles';

const Home = () => {
  const classes = useStyles();
  const [currentId, setCurrentId] = useState(null);
  const dispatch = useDispatch();
  useEffect(() => {
    dispatch(getPosts());
  }, [currentId, dispatch]);
  return (
    <Grow in>
      <Container >
        <Grid className={classes.mainContainer} container justify="space-between" alignItems="stretch" spacing={3}>
          <Grid item xs={12} sm={7}>
            <Posts setCurrentId={setCurrentId} />
          </Grid>
          <Grid item xs={12} sm={4}>
            <Form currentId={currentId} setCurrentId={setCurrentId} />
          </Grid>
        </Grid>
      </Container>
    </Grow>
  )
}

export default Home
```

components/Navbar/Navbar.js:

```
import React,{useState,useEffect} from 'react';
import {AppBar,Typography,Toolbar,Button,Avatar} from '@material-ui/core';
import useStyles from './styles';
import memories from '../images/memories.png';
import {Link,useHistory,useLocation} from 'react-router-dom';
import {useDispatch} from 'react-redux';
import decode from 'jwt-decode';

const Navbar = () => {
  const classes = useStyles();
  //user immediately fetched
  const [user,setUser] = useState(JSON.parse(localStorage.getItem('profile'))) ;
  const dispatch = useDispatch();
  const history =useHistory();
  const location = useLocation();
  // console.log(user);
  const logout =()=>{
    dispatch({type:'LOGOUT'});
    history.push('/');
    setUser(null);
  }
  useEffect(()=>{
    const token = user?.token;
    //JWT CODE HERE
    if(token){
      const decodedToken = decode(token);
      if(decodedToken.exp * 1000 < new Date().getTime()) logout();
    }

    setUser(JSON.parse(localStorage.getItem('profile')));
  },[location]);

  return (
    <AppBar className={classes.appBar} position='static' color="inherit">
      <div className={classes.brandContainer}>
        <Typography component={Link} to="/" className={classes.heading} variant='h2' align='center'>
          Memories
        </Typography>
        <img className={classes.image} src={memories} alt="memories" height='60' />
      </div>
      <Toolbar className={classes.toolbar}>
        {user?
          <div className={classes.profile}>
            <Avatar className={classes.purple} alt={user.result.name} src={user.result.imageUrl} />
            <Typography className={classes.userName} variant="h6" >{user.result.name}</Typography>
            <Button variant="contained" className={classes.logout} color="secondary" onClick={logout} >Log Out</Button>
          </div>
          :(
            <Button component={Link} to="/auth" variant="contained" color="primary" >Sign In</Button>
          )
        }
      </Toolbar>
    </AppBar>
  )
}

export default Navbar
```


components/Navbar/styles.js:

```
import { makeStyles } from '@material-ui/core/styles';
import { deepPurple } from '@material-ui/core/colors';

export default makeStyles((theme) => ({
  appBar: {
    borderRadius: 15,
    margin: '30px 0',
    display: 'flex',
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    padding: '10px 50px',
  },
  heading: {
    color: 'rgba(0,183,255, 1)',
    textDecoration: 'none',
  },
  image: {
    marginLeft: '15px',
  },
  toolbar: {
    display: 'flex',
    justifyContent: 'flex-end',
    width: '400px',
  },
  profile: {
    display: 'flex',
    justifyContent: 'space-between',
    width: '400px',
  },
  userName: {
    display: 'flex',
    alignItems: 'center',
  },
  brandContainer: {
    display: 'flex',
    alignItems: 'center',
  },
  purple: {
    color: theme.palette.getContrastText(deepPurple[500]),
    backgroundColor: deepPurple[500],
  },
})));
```

components/Posts/posts.js:

```
import React from 'react';
import { Grid, CircularProgress } from '@material-ui/core';
import Post from './Post/Post';
import useStyles from './styles';
import { useSelector } from 'react-redux';

const Posts = ({ setCurrentId }) => {
  const posts = useSelector((state) => state.posts);
  const classes = useStyles();
  console.log(posts);
  return (
    !posts.length ? <CircularProgress/> : (
      <Grid className={classes.mainContainer} container alignItems="stretch" spacing={3}>
        {posts.map((post) => (
          <Grid key={post._id} item xs={12} sm={6}>
            <Post post={post} setCurrentId={setCurrentId} />
          </Grid>
        ))}
      </Grid>
    )
  );
};
```

```
        </Grid>
      ))
    }
  </Grid>
)

);
}
export default Posts;
```

components/Posts/styles.js:

```
import { makeStyles } from '@material-ui/core/styles';

export default makeStyles((theme) => ({
  mainContainer: {
    display: 'flex',
    alignItems: 'center',
  },
  smMargin: {
    margin: theme.spacing(1),
  },
  actionDiv: {
    textAlign: 'center',
  },
})));
```

components/Posts/Post/styles.js:

```
import { makeStyles } from '@material-ui/core/styles';

export default makeStyles({
  media: {
    height: 0,
    padding: '56.25%',
    backgroundColor: 'rgba(0, 0, 0, 0.5)',
    backgroundBlendMode: 'darken',
  },
  border: {
    border: 'solid',
  },
  fullHeightCard: {
    height: '100%',
  },
  card: {
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'space-between',
    borderRadius: '15px',
    height: '100%',
    position: 'relative',
  },
  overlay: {
    position: 'absolute',
    top: '20px',
    left: '20px',
    color: 'white',
  },
  overlay2: {
    position: 'absolute',
    top: '20px',
    right: '20px',
    color: 'white',
  },
});
```

```

    },
    grid: {
      display: 'flex',
    },
    details: {
      display: 'flex',
      justifyContent: 'space-between',
      margin: '20px',
    },
    title: {
      padding: '0 16px',
    },
    cardActions: {
      padding: '0 16px 8px 16px',
      display: 'flex',
      justifyContent: 'space-between',
    },
  },
});

```

components/Posts/Post/Post.js:

```

import React from 'react';
import {Card, CardActions, CardContent, CardMedia, Button, Typography} from '@material-ui/core';
import ThumbUpAltIcon from '@material-ui/icons/ThumbUpAlt';
import ThumbUpAltOutlined from '@material-ui/icons/ThumbUpAltOutlined';
import DeleteIcon from '@material-ui/icons/Delete';
import MoreHorizIcon from '@material-ui/icons/MoreHoriz';
import moment from 'moment';
import useStyles from './styles';
import {useDispatch} from 'react-redux';
import {deletePost, likePost} from '../../actions/posts';

const Post=({post, setCurrentId})=>{
  const classes = useStyles();
  const dispatch = useDispatch();
  const user = JSON.parse(localStorage.getItem('profile'));
  const Likes = () => {
    if (post.likes.length > 0) {
      return post.likes.find((like) => like === (user?.result?.googleId || user?.result?._id))
        ? (
            <<ThumbUpAltIcon fontSize="small" />&nbsp;{post.likes.length > 2 ? `You and ${post.likes.le
length - 1} others` : `${post.likes.length} like${post.likes.length > 1 ? 's' : ''}`} </>
          ) : (
            <<ThumbUpAltOutlined fontSize="small" />&nbsp;{post.likes.length} {post.likes.length === 1
? 'Like' : 'Likes'}</>
          );
    }
    return <<ThumbUpAltOutlined fontSize="small" />&nbsp;Like</>;
  };

  return(
    <Card className={classes.card}>
      <CardMedia className={classes.media} image={post.selectedFile} title={post.title} />
      <div className={classes.overlay}>
        <Typography variant="h6">{post.name}</Typography>
        <Typography variant="body2">{moment(post.createdAt).fromNow()}</Typography>
      </div>
      {(user?.result?.googleId === post?.creator || user?.result?._id === post?.creator) && (
        <div className={classes.overlay2}>
          <Button onClick={() => setCurrentId(post._id)} style={{ color: 'white' }} size="small">
            <MoreHorizIcon fontSize="default" />
          </Button>
        </div>
      )}
      <div className={classes.details} >

```

```

        <Typography variant="body2" color="textSecondary">{post.tags.map((tag)=>`#${tag} `)}</Typograp
hy>
    </div>
    <Typography className={classes.title} variant="h5" gutterBottom>{post.title}</Typography>
    <CardContent>
    <Typography variant="body2" color="textSecondary" component="p" >{post.message}</Typography>
    </CardContent>
    <CardActions className={classes.cardActions}>
    <Button size="small" color="primary" disabled={!user?.result} onClick={()=>dispatch(likePost(
post._id))} >
        <Likes/>
        {/* <ThumbUpAltIcon fontSize="small" />
        &nbsp; Like &nbsp; */}
        {post.likeCount} */}
    </Button>
    {(user?.result?.googleId === post?.creator || user?.result?._id === post?.creator) && (
    <Button size="small" color="secondary" onClick={() => dispatch(deletePost(post._id))}>
        <DeleteIcon fontSize="small" /> Delete
    </Button>
    )}
    </CardActions>
    </Card>
    );
}
export default Post;

```

constants/actionTypes.js:

```

export const CREATE = 'CREATE';
export const UPDATE = 'UPDATE';
export const DELETE = 'DELETE';
export const LIKE = 'LIKE';
export const FETCH_ALL = 'FETCH_ALL';
export const AUTH = 'AUTH';
export const LOGOUT = 'LOGOUT';

```

reducers/auth.js:

```

import {AUTH,LOGOUT} from '../constants/actionTypes';

const authReducer = (state={authData:null},action) =>{
  switch(action.type){
    case AUTH :
      // ?. is called optional chaining. this doesn't throw an error if action value doesn't exist
      // console.log(action?.data);
      localStorage.setItem('profile',JSON.stringify({...action?.data}));
      return {...state, authData : action?.data};

    case LOGOUT:
      localStorage.clear();
      return {...state, authData : null};
    default:
      return state;
  }
}
export default authReducer;

```

reducers/index.js:

```

import {combineReducers} from 'redux';
import posts from './posts';
import auth from './auth';

export default combineReducers({posts, auth});

```

reducers/posts.js:

```
import {CREATE, FETCH_ALL,UPDATE,LIKE,DELETE} from '../constants/actionTypes';

export default (posts = [], action) => {
  switch (action.type) {
    case DELETE:
      return posts.filter((post)=> post._id !== action.payload)
    case UPDATE:
    case LIKE:
      return posts.map((post)=> post._id === action.payload._id ? action.payload : post);
    case FETCH_ALL:
      return action.payload;
    case CREATE:
      return [...posts,action.payload];
    default:
      return posts;
  }
};
```

client/src/App.js:

```
import React from 'react';

import Navbar from '../components/Navbar/Navbar';
import {Container} from '@material-ui/core';

import {BrowserRouter,Switch,Route} from 'react-router-dom';
import Home from '../components/Home/Home';
import Auth from '../components/Auth/Auth';

const App=() =>{
  return(
    <BrowserRouter>

      <Container maxWidth='lg'>
        <Navbar/>
        <Switch>
          <Route path="/" exact component={Home} />
          <Route path="/auth" exact component={Auth} />
        </Switch>

      </Container>
    </BrowserRouter>
  );
}
export default App;
```

client/src/styles.js:

```
import { makeStyles } from '@material-ui/core/styles';

export default makeStyles((theme) => ({
  appBar: {
    borderRadius: 15,
    margin: '30px 0',
    display: 'flex',
    flexDirection: 'row',
    justifyContent: 'center',
    alignItems: 'center',
  },
  heading: {
    color: 'rgba(0,183,255, 1)',
  },
});
```

```
image: {
  marginLeft: '15px',
},
[theme.breakpoints.down('sm')]:{
  mainContainer :{
    flexDirection : "column-reverse"
  }
}
}
}
}));
```

client/src/index.js:

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import {Provider} from 'react-redux';
import {createStore, applyMiddleware,compose} from 'redux';
import thunk from 'redux-thunk';
import reducers from './reducers';
import './index.css';

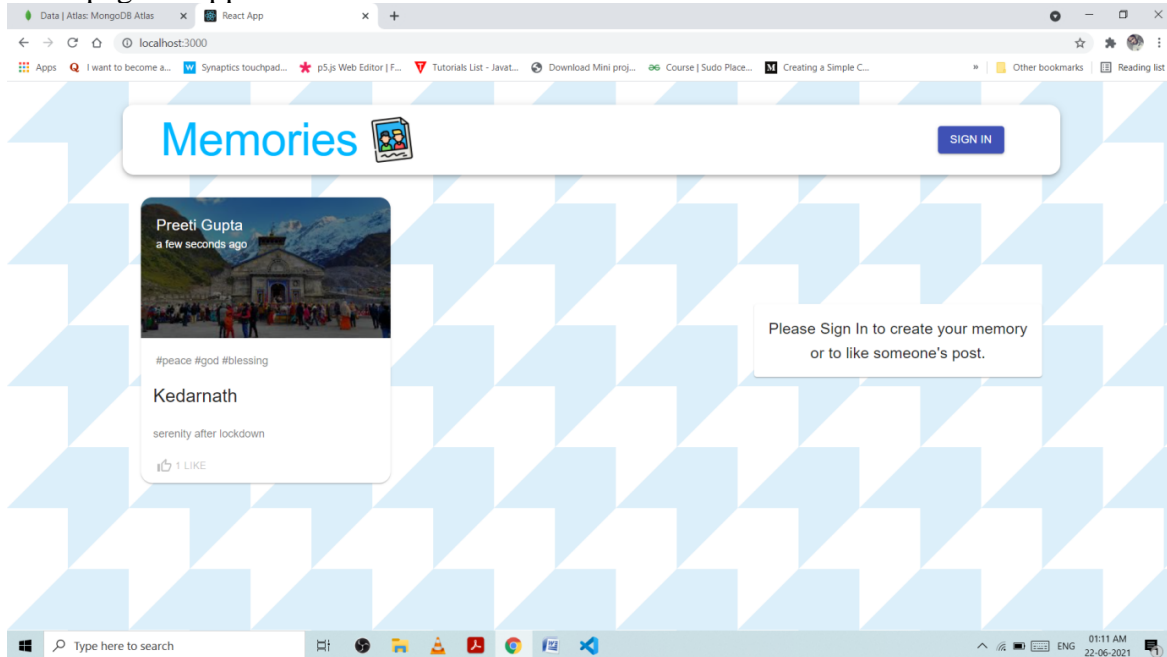
const store = createStore(reducers,compose(applyMiddleware(thunk)));
ReactDOM.render(
  <Provider store={store}>
    <App/>
  </Provider>,
  document.getElementById('root'));
```

client/src/index.js:

```
body{
  background-color: #ffffff;
background-
image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg' width='160' height='160' viewBox='0 0 200 200'%3E%3Cpolygon fill='%23DCEFFA' points='100 0 0 100 100 100 100 200 200 100 200 0'/%3E%3C/svg%3E");
}
```

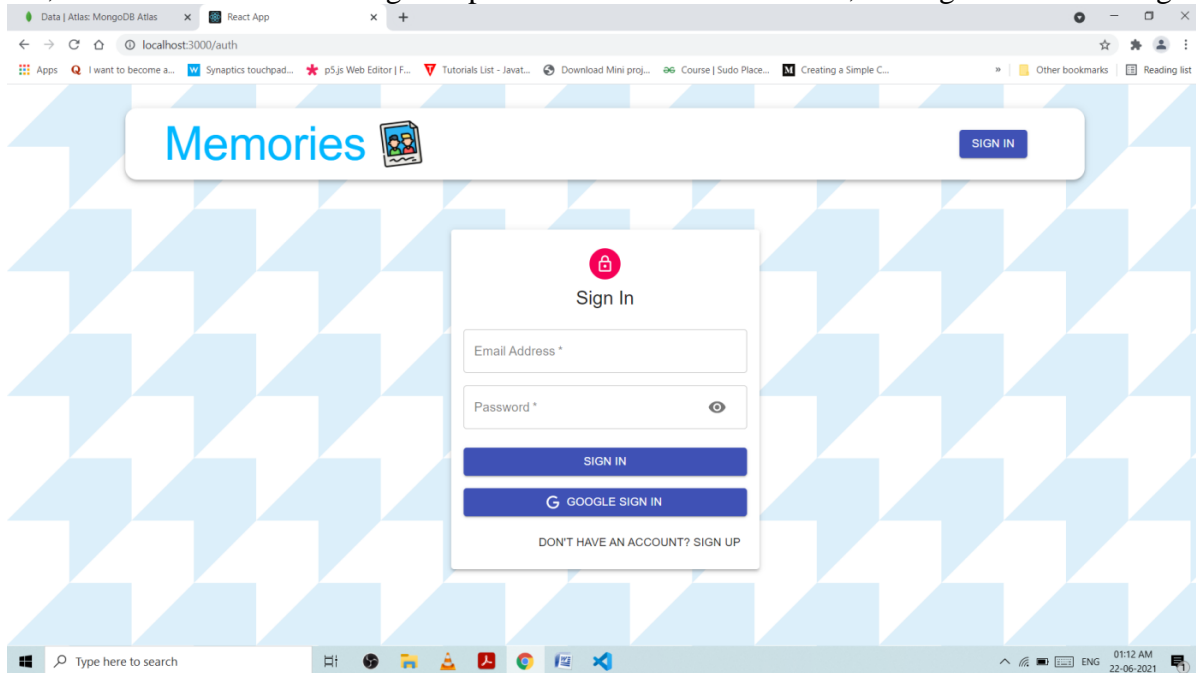
8. OUTPUTS

1. Until the user logs in, s/he can not do anything except for seeing all posts. This is the home page of app.



(fig 8.1. Home Page)

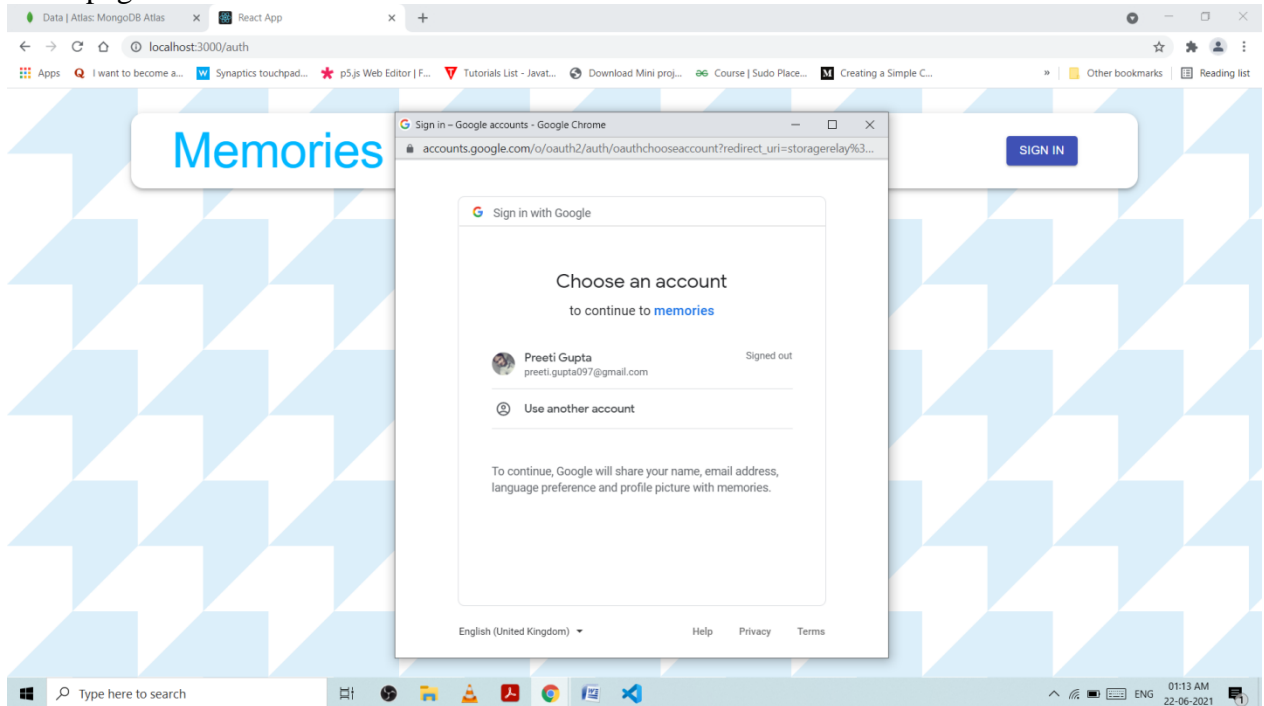
2. On clicking sign In, we get following sign in page, if user exists, s/he can directly login. If not, s/he can either sign up and create account ,or sign in with google.



(fig 8.2. Sign In screen)

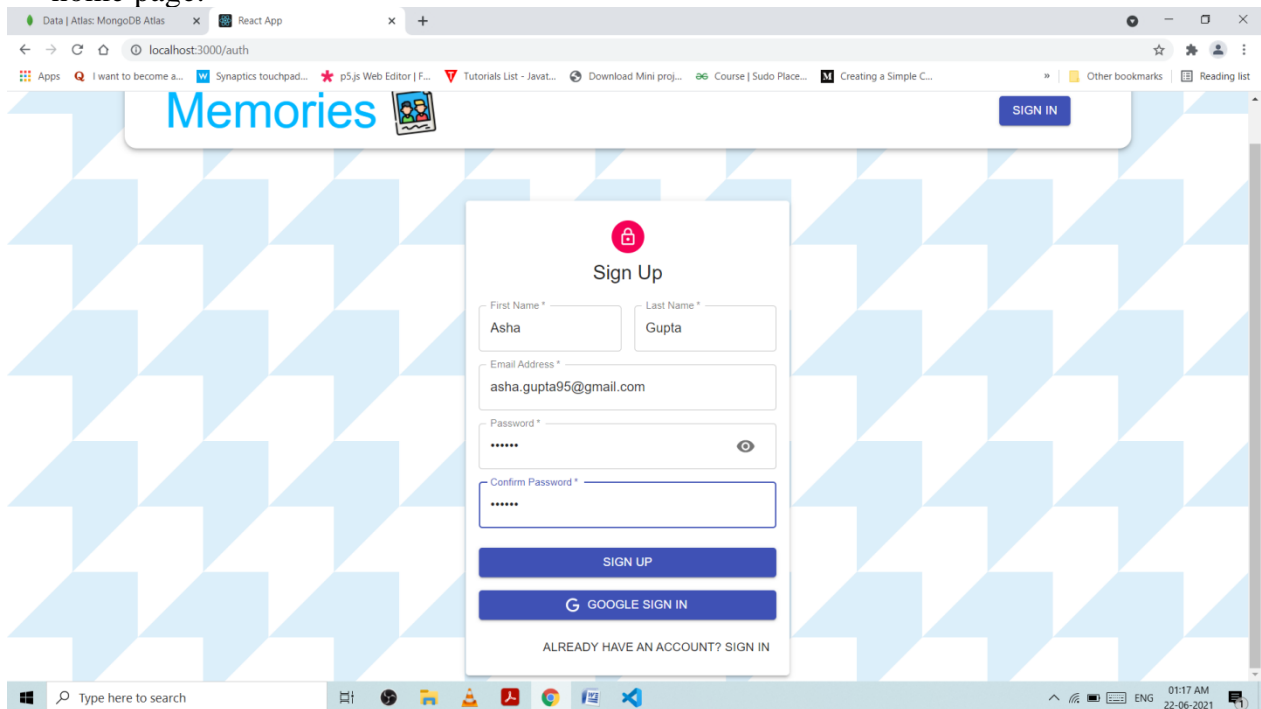
MEMORIES MERN CRUD APP

3. If user signs with google, following page shows. On successful login, we are redirected to home page.



(fig 8.3. Google Sign In)

4. If user signs up, following page is shown. On successful sign up, we are redirected to home page.



(fig 8.4. Sign Up screen)

- [illegible]

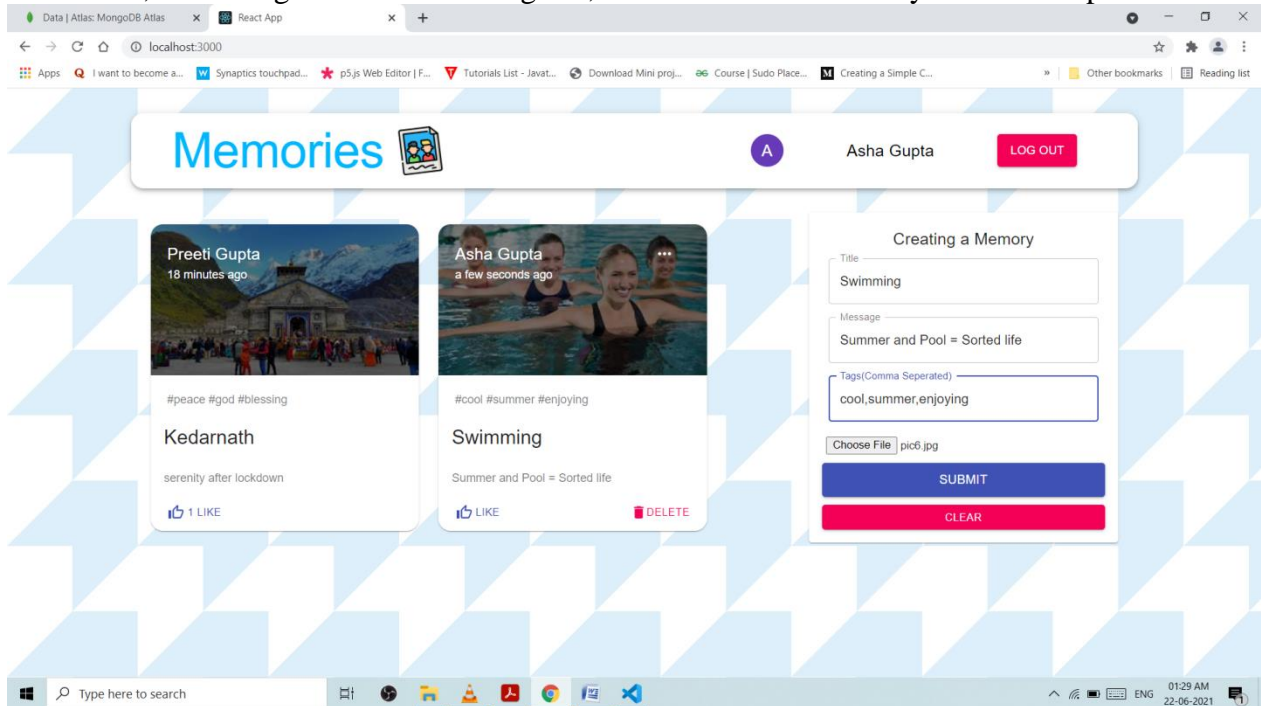
6. Once sign up completes or user logs on through google, we get Home Page. Now, we not only sees all posts, but also create, delete and edit our post, and not to forget like someone's post. A user is allowed to like a post only once.

-

VSSUT 2021

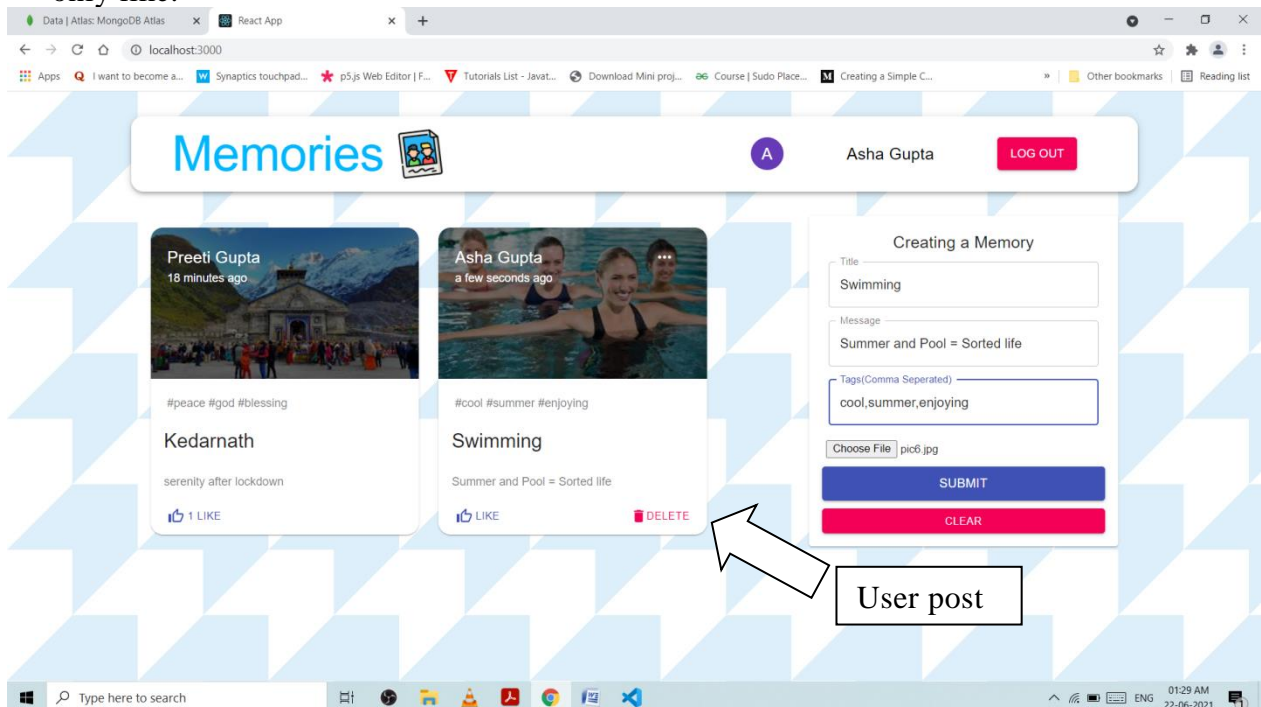
MEMORIES MERN CRUD APP

7. User created a post. The image uploaded is converted to string. Once the user clicks submit, the data gets stored in mongodb, and almost immediately we see our post.



(fig 8.7.creating post)

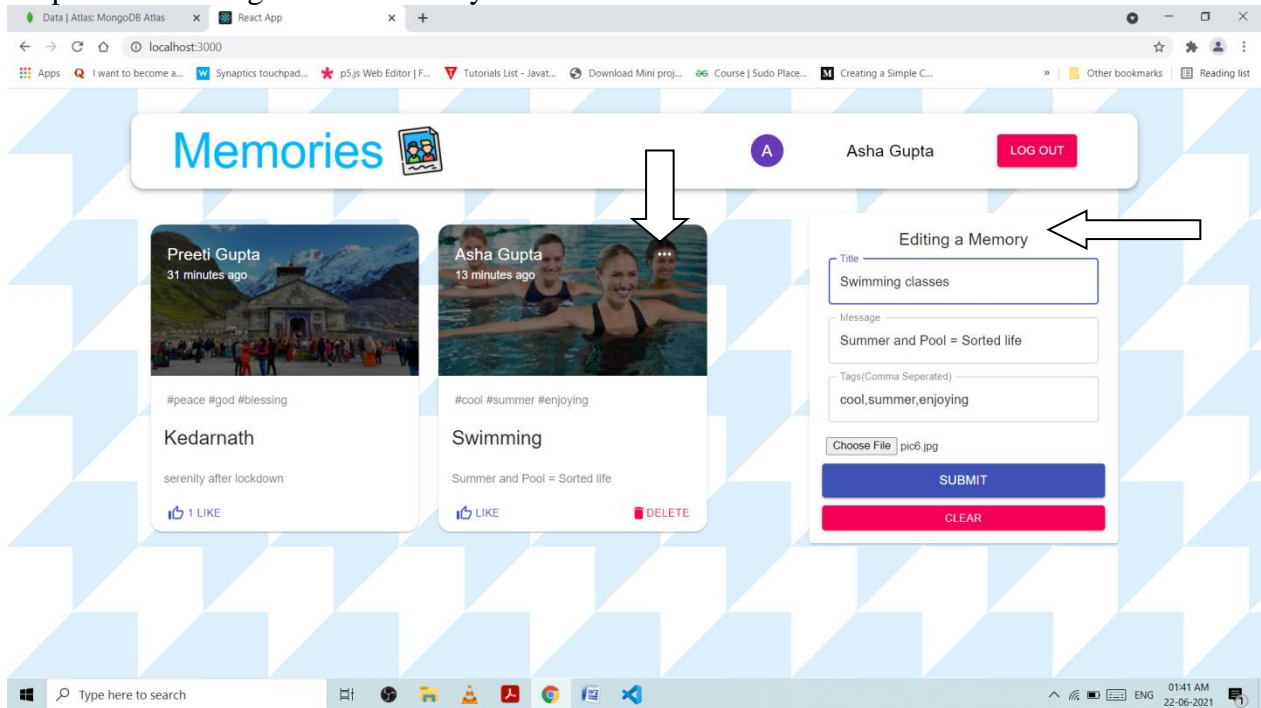
8. A user is allowed to like, delete and edit only his/her post. For others' post, they can only like.



(fig 8.8. User capability)

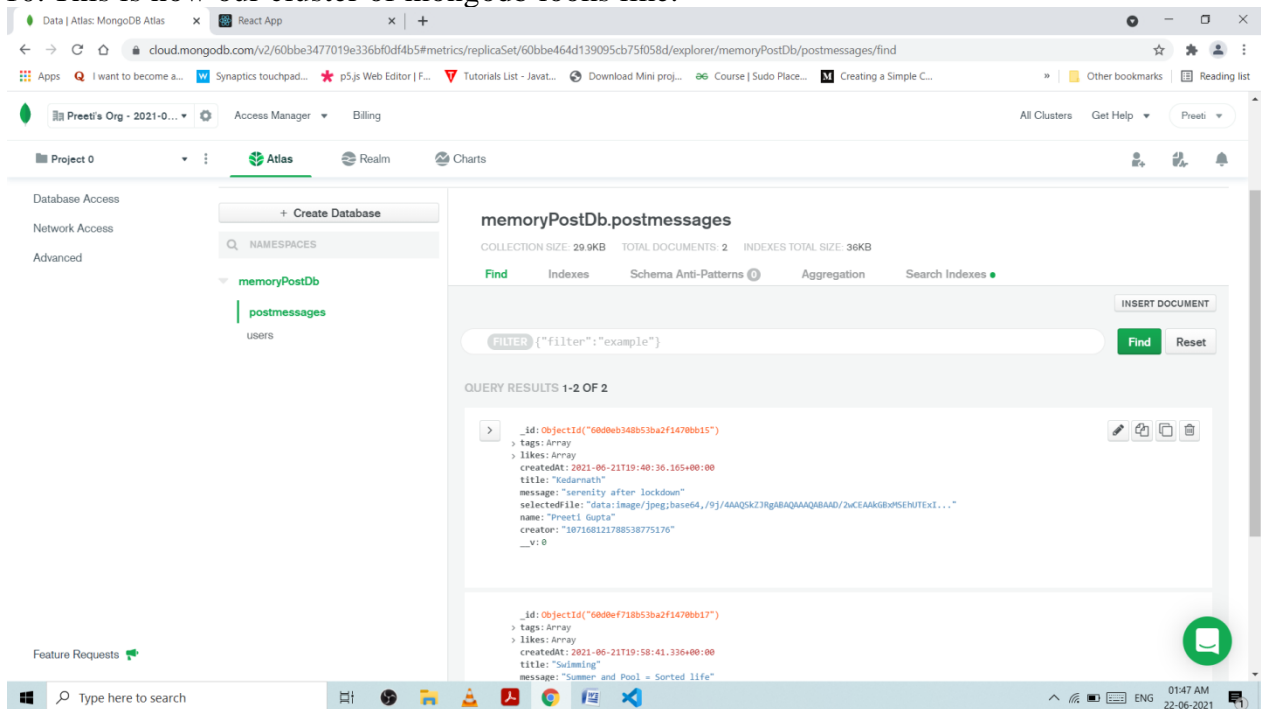
MEMORIES MERN CRUD APP

9. If the user clicks on the three dot on top right corner of his/her post, s/he can edit their post. The change is immediately reflected on screen.



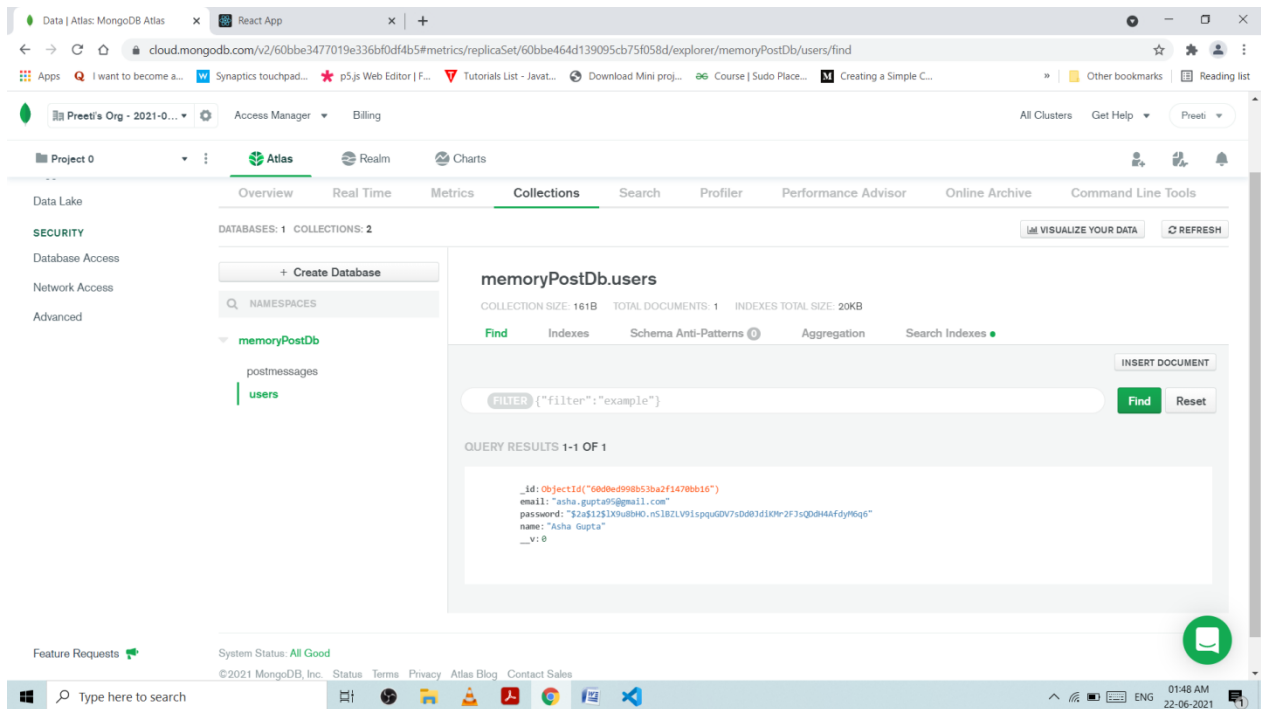
(fig 8.9.edit a post)

10. This is how our cluster of mongodb looks like.



(fig 8.10. Post data)

MEMORIES MERN CRUD APP



(fig 8.11. User data)

9. APPLICATION

- This application is simple social media web app.
- It can be used to create a post and upload events of our life.
- We can see/Read all posts ,including ourselves.
- We can edit/ipdate our post
- We can delete our post
- There is authentication through sign up and Google Login

10. CONCLUSION AND FUTURE SCOPE

Conclusion:

The project was build to connect with people, and also preserve our memories. This is just a simple UI social media app, with the idea to build CRUD app in MERN. There is lot of work needed to make it a fully functional social media app that we use today.

Future Scope:

****Enhancement:**

This application may need other features in the future. I, the developer, cannot guarantee that this application will be still suitable in all critical environment lying ahead. The application may go obsolete or it may not be supported by phones in coming future due to changing technology. Hence, I hereby mention that this application may require major or minor modifications in near future. On the other hand, new features may be added if present scenario doesn't fully satisfy the user needs in future.

11. BIBLIOGRAPHY

- <https://www.npmjs.com/>
- <https://www.mongodb.com/>
- <https://reactrouter.com>
- <https://www.jwt.io>
- <http://stackoverflow.com/>
- <https://www.youtube.com/watch?v=LKlO8vLvUao>