FULL STACK PROJECT

(2021-2022)

Vision

PROJECT REPORT

Institute Of Engineering and Technology

Team Members

Shalini Kumari
(181500649)
Ritik Srivastava
(181500578)
Adarsh Kumar
(181500037)
Shashank Agrawal
(181500652)
Vanshika Agrawal
(181500780)

-Supervised By

Name- Mr. Pankaj Kapoor

Technical Trainer

Department of Computer Engineering & Applications

CONTENT

Problem Statement:

Vision is a platform where users can post about their happy memories. The post would act as a snapshot of a particular memory which can be revisited by using our platform. Human brain has a special part dedicated to memories. Our brain tends to remember the things that impact our brain too much and brought big change in our life. But there are other memories also which are not that impactful but which have played significant role in shaping our life. It is important to recollect these memories at certain point of our life to gain the motivation and inspiration of moving ahead.

Reason:

Using the vision platform, the users will make a post about a particular memory of their life. This post will be stored in our platform and can be seen by the user as a collection of memories. This platform will help the user to recollect the memories whenever they feel lack of inspiration. As the name of the platform suggest it will give vision to the user. It is sometimes important to look back and plan for the future.

Human tends to make mistakes but a wise human learns from the mistake and grows. Our platform can also become the life story of the person which will display small and big moments, mistakes, achievements. It is a free service that users can avail without any fees.

Objective:

Vision is a platform where users can post about their happy memories. The post would act as a

snapshot of a particular memory which can be revisited by using our platform. Human brain has a special part dedicated to memories. Our brain tends to remember the things that impact our brain too much and brought big change in our life. But there are other memories also which are not that impactful but which have played significant role in shaping our life. It is important to recollect these memories at certain point of our life to gain the motivation and inspiration of moving ahead.
.

Literature:

There is no platform where users can specifically can post about their memories. There are many social media platforms but none of them focuses on the life story of the person. People tend to post fake stories on social media platforms to impress their friends. Our platform is dedicated to individual but not to someone trying impress his/her friend circle. Our platform is a self-note which can read by the person when they feel demotivated.

Future Scope:

- Filter can be added in future to sort the memories by date and time.
- Voice notes can also be added so that the memories can be impactful.
- The platform can be used to share memories to other platforms.

Requirements:

1. Hardware:
   Ram required- 8GB or above
   Processor- core i5 or above

2. Software:
   Linux or Windows Operating System
   MERN Stack
   Atom or Vs.-code
   ES6 supportable browser for debugging.

3. Technology Used:
   MERN Stack.

Methodology

The idea of this methodology is to develop a web-application system that helps users to post their memories which cane be read by them in future.

In this web application users can register/sign-up themself and can post their memories which can be read by others also. All users post can be seen at one place. Posts are editable and also can be deleted. User have logout and re-login options also.

Modules: Description:

1)Login/Sign-up Page	Here a user can either login if he/she has already registered on this website or can register themself by their id and make a new password.
2)Main Page	On this page there is form by which posts can be added. And these posts are posted on this page only.

Testing:

We can test React components similar to testing other JavaScript code. There are a few ways to test React components. Broadly, they divide into two categories:

- **Rendering component trees** in a simplified test environment and asserting on their output.
- **Running a complete app** in a realistic browser environment (also known as "end-to-end" tests).

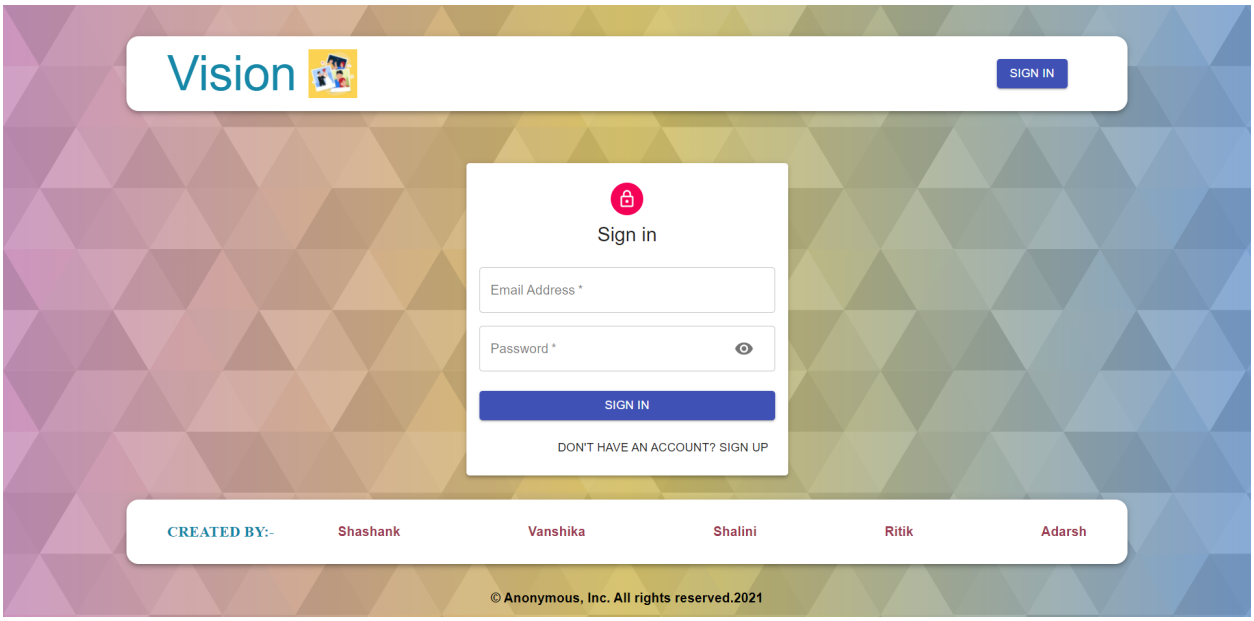
When choosing testing tools, it is worth considering a few trade-offs:

- **Iteration speed vs Realistic environment:** Some tools offer a very quick feedback loop between making a change and seeing the result, but don't model the browser behaviour precisely. Other tools might use a real browser environment, but reduce the iteration speed and are flakier on a continuous integration server.
- **How much to mock:** With components, the distinction between a "unit" and "integration" test can be blurry. If you're testing a form, should its test also test the buttons inside of it? Or should a button component have its own test suite? Should refactoring a button ever break the form test?

Jest is a JavaScript test runner that lets you access the DOM via jsdom. While jsdom is only an approximation of how the browser works, it is often good enough for testing React components. Jest provides a great iteration speed combined with powerful features like mocking modules and timers so you can have more control over how the code executes.

React Testing Library is a set of helpers that let you test React components without relying on their implementation details. This approach makes refactoring a breeze and also nudges you towards best practices for accessibility. Although it doesn't provide a way to "shallowly" render a component without its children, a test runner like Jest lets you do this by mocking.

Client
ACTIONS
auth.js




```
1  import { AUTH } from "../constants/actionTypes";
2  import * as api from "../api/index.js";
3
4  export const signin = (formData, router) => async
   (dispatch) => {
5    try {
6      const { data } = await api.signIn(formData);
7
8      dispatch({ type: AUTH, data });
9
10     router.push("/");
11   } catch (error) {
12     console.log(error);
13   }
14 };
15
16 export const signup = (formData, router) => async
   (dispatch) => {
17   try {
18     const { data } = await api.signUp(formData);
```

```
19
20      dispatch({ type: AUTH, data });
21
22      router.push("/");
23    } catch (error) {
24      console.log(error);
25    }
26  };
```

posts.js

```
1  import {
2    FETCH_ALL,
3    CREATE,
4    UPDATE,
5    DELETE,
6    LIKE,
7  } from "../constants/actionTypes";
8  import * as api from "../api/index.js";
9
10 export const getPosts = () => async (dispatch) => {
11   try {
12     const { data } = await api.fetchPosts();
13
14     dispatch({ type: FETCH_ALL, payload: data });
15   } catch (error) {
16     console.log(error);
17   }
18 };
19
20 export const createPost = (post) => async (dispatch) => {
21   try {
22     const { data } = await api.createPost(post);
23
24     dispatch({ type: CREATE, payload: data });
25   } catch (error) {
26     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 likePost = (id) => async (dispatch) => {
  const user = JSON.parse(localStorage.getItem("profile"));

  try {
    const { data } = await api.likePost(id, user?.token);

    dispatch({ type: LIKE, payload: data });
  } catch (error) {
    console.log(error);
  }
};

export const deletePost = (id) => async (dispatch) => {
  try {
    await await api.deletePost(id);

    dispatch({ type: DELETE, payload: id });
  } catch (error) {
    console.log(error);
  }
};
```

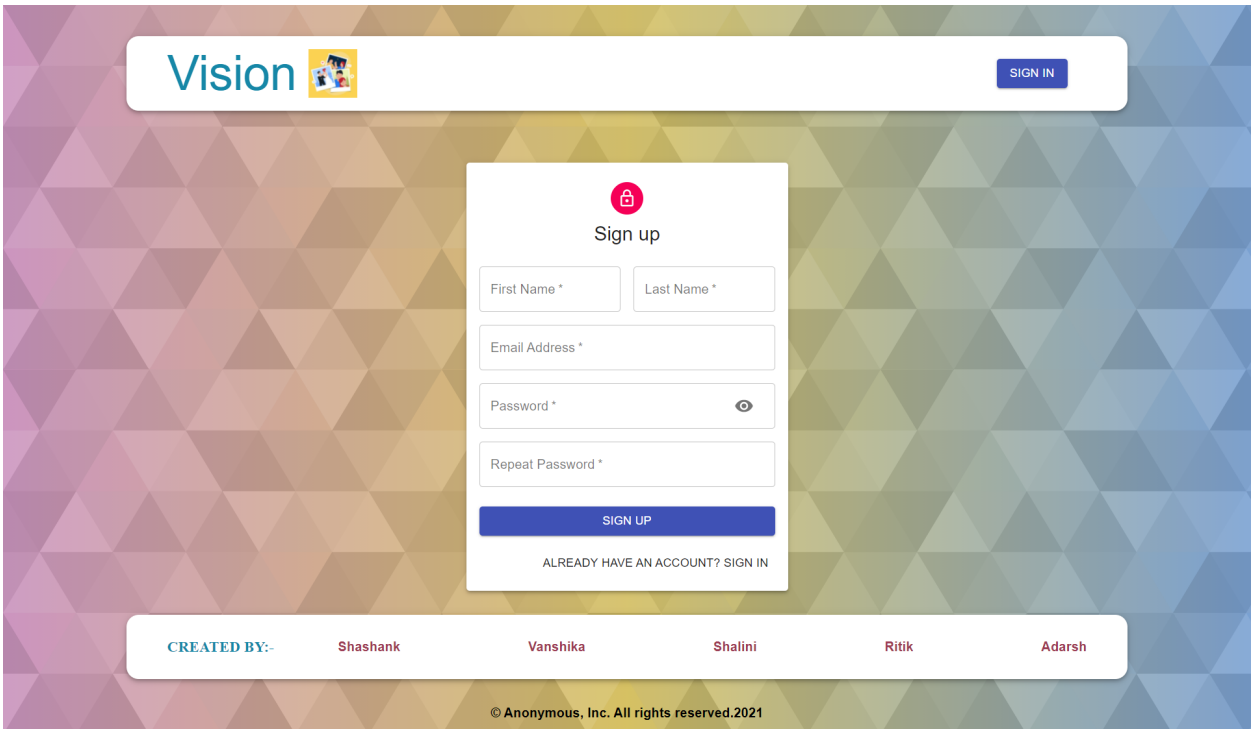
API

index.js

```
import axios from "axios";

const url = "http://localhost:5000/posts";

export const fetchPosts = () => axios.get(url);
export const createPost = (newPost) => {
  console.log(newPost);
  axios.post(url, newPost);
};
export const likePost = (id) =>
axios.patch(`${url}/${id}/likePost`);
export const updatePost = (id, updatedPost) =>
  axios.patch(`${url}/${id}`, updatedPost);
export const deletePost = (id) =>
axios.delete(`${url}/${id}`);

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

  return req;
});

export const signIn = (formData) =>
  axios.post("http://localhost:5000/user/signin",
formData);
export const signUp = (formData) =>
  axios.post("http://localhost:5000/user/signup",
formData);
```

COMPONENTS

Auth.js




```
import React, { useState } from "react";
import { useDispatch } from "react-redux";
import {
  Avatar,
  Button,
  Paper,
  Grid,
  Typography,
  Container,
} from "@material-ui/core";
import { useHistory } from "react-router-dom";
import { GoogleLogin } from "react-google-login";
import LockOutlinedIcon from
  "@material-ui/icons/LockOutlined";

import Icon from "./Icon";
import { signin, signup } from "../../actions/auth";
import { AUTH } from "../../constants/actionTypes";
import useStyles from "./styles";
import Input from "./Input";
```

```
20
21 const initialState = {
22   firstName: "",
23   lastName: "",
24   email: "",
25   password: "",
26   confirmPassword: "",
27 };
28
29 const SignUp = () => {
30   const [form, setForm] = useState(initialState);
31   const [isSignup, setIsSignup] = useState(false);
32   const dispatch = useDispatch();
33   const history = useHistory();
34   const classes = useStyles();
35
36   const [showPassword, setShowPassword] = useState(false);
37   const handleShowPassword = () =>
   setShowPassword(!showPassword);
38
39   const switchMode = () => {
40     setForm(initialState);
41     setIsSignup((prevIsSignup) => !prevIsSignup);
42     setShowPassword(false);
43   };
44
45   const handleSubmit = (e) => {
46     e.preventDefault();
47
48     if (isSignup) {
49       dispatch(signup(form, history));
50     } else {
51       dispatch(signin(form, history));
52     }
53   };
54
```

```
55   const googleSuccess = async (res) => {
56     const result = res?.profileObj;
57     const token = res?.tokenId;
58
59     try {
60       dispatch({ type: AUTH, data: { result, token } });
61
62       history.push("/");
63     } catch (error) {
64       console.log(error);
65     }
66   };
67
68   const googleError = () =>
69     alert("Google Sign In was unsuccessful. Try again
   later");
70
71   const handleChange = (e) =>
72     setForm({ ...form, [e.target.name]: e.target.value });
73
74   return (
75     <Container component="main" maxWidth="xs">
76       <Paper className={classes.paper} elevation={3}>
77         <Avatar className={classes.avatar}>
78           <LockOutlinedIcon />
79         </Avatar>
80         <Typography component="h1" variant="h5">
81           {isSignup ? "Sign up" : "Sign in"}
82         </Typography>
83         <form className={classes.form}
   onSubmit={handleSubmit}>
84           <Grid container spacing={2}>
85             {isSignup && (
86               <>
87                 <Input
88                   name="firstName"
```

```
89                 label="First Name"
90                 handleChange={handleChange}
91                 autoFocus
92                 half
93               />
94               <Input
95                 name="lastName"
96                 label="Last Name"
97                 handleChange={handleChange}
98                 half
99               />
100             </>
101           )}
102           <Input
103             name="email"
104             label="Email Address"
105             handleChange={handleChange}
106             type="email"
107           />
108           <Input
109             name="password"
110             label="Password"
111             handleChange={handleChange}
112             type={showPassword ? "text" : "password"}
113             handleShowPassword={handleShowPassword}
114           />
115           {isSignup && (
116             <Input
117               name="confirmPassword"
118               label="Repeat Password"
119               handleChange={handleChange}
120               type="password"
121             />
122           )}
123         </Grid>
124         <Button
```

```
            type="submit"
            fullWidth
            variant="contained"
            color="primary"
            className={classes.submit}
          >
            {isSignup ? "Sign Up" : "Sign In"}
          </Button>

          <Grid container justify="flex-end">
            <Grid item>
              <Button onClick={switchMode}>
                {isSignup
                  ? "Already have an account? Sign in"
                  : "Don't have an account? Sign Up"}
              </Button>
            </Grid>
          </Grid>
        </form>
      </Paper>
    </Container>
  );
};
export default SignUp;
```

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;
```

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 onClick={handleShowPassword}>
                    {type === "password" ? <Visibility /> :
  <VisibilityOff />}
                  </IconButton>
                </InputAdornment>
              ),
            }
          : null
      }
    />
  </Grid>
);

export default Input;
```

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),
  },
}));
```

Footer.js

```
import React from "react";
import { AppBar, Typography} from "@material-ui/core";
import useStyles from "./styles";

const Footer = () => {

  const classes = useStyles();



  return (
    <div>
    <AppBar className={classes.appBar} position="static"
color="inherit">

        <h3 className={classes.heading}>CREATED BY:-</h3>

      <Typography className={classes.root}>
  <a className={classes.git}
href="https://github.com/ShashankAgrawal003" alt="shashank
```

```
  github" target="_blank" rel="noreferrer">
    <strong>Shashank</strong>
    </a>
    </Typography>

    <br></br>

    <Typography className={classes.root}>
    <a className={classes.git}
  href="https://github.com/vanshika0307" alt="vanshika
  github" target="_blank" rel="noreferrer">
    <strong>Vanshika</strong>
    </a>
    </Typography>

    <br></br>

    <Typography className={classes.root}>
    <a className={classes.git}
  href="https://github.com/shalinikumari50" alt="shalini
  github" target="_blank" rel="noreferrer">
    <strong>Shalini</strong>
    </a>
    </Typography>

    <br></br>

<Typography className={classes.root}>
<a className={classes.git}
  href="https://github.com/Ritik-code" alt="ritik github"
  target="_blank" rel="noreferrer" >
    <strong>Ritik</strong>
    </a>
</Typography>

    <br></br>

```

```
49   <Typography className={classes.root}>
50   <a className={classes.git}
   href="https://github.com/ADDY-AK" alt="adarsh github"
   target="_blank" rel="noreferrer">
51   <strong>Adarsh</strong>
52   </a>
53   </Typography>
54     </AppBar>
55
56     <Typography className={classes.copyright}
   align="center" noWrap>
57   <strong display="block">© Anonymous, Inc. All rights
   reserved.2021</strong>
58   </Typography>
59     </div>
60   );
61 };
62
63 export default Footer;
```

Form.js

```
1  import React, { useState, useEffect } from "react";
2  import { TextField, Button, Typography, Paper } from
   "@material-ui/core";
3  import { useDispatch, useSelector } from "react-redux";
4  import FileBase from "react-file-base64";
5
6  import { createPost, updatePost } from
   "../../actions/posts";
7  import useStyles from "./styles";
8
9  const Form = ({ currentId, setCurrentId }) => {
10   const [postData, setPostData] = useState({
11     title: "",
12     message: "",
13     tags: "",
```

```
    selectedFile: "",
  });
  const post = useSelector((state) =>
    currentId ? state.posts.find((message) => message._id
=== currentId) : null
  );
  const dispatch = useDispatch();
  const classes = useStyles();
  const user = JSON.parse(localStorage.getItem("profile"));

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

  const clear = () => {
    setCurrentId(0);
    setPostData({ title: "", message: "", tags: "",
selectedFile: "" });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (currentId === 0) {
      dispatch(createPost({ ...postData, name:
user?.result?.name }));
      clear();
    } else {
      dispatch(
        updatePost(currentId, { ...postData, name:
user?.result?.name })
      );
      clear();
    }
  };

```

```
46   if (!user?.result?.name) {
47     return (
48       <Paper className={classes.paper}>
49         <Typography variant="h6" align="center">
50           Please Sign In to create your own memories and
   like other's memories.
51         </Typography>
52       </Paper>
53     );
54   }
55
56   return (
57     <Paper className={classes.paper}>
58       <form
59         autoComplete="off"
60         noValidate
61         className={`${classes.root} ${classes.form}`}
62         onSubmit={handleSubmit}
63       >
64         <Typography variant="h6">
65           {currentId ? `Editing "${post.title}"` :
   "Creating a Memory"}
66         </Typography>
67         <TextField
68           name="title"
69           variant="outlined"
70           label="Title"
71           fullWidth
72           value={postData.title}
73           onChange={(e) => setPostData({ ...postData,
   title: e.target.value })}
74         />
75         <TextField
76           name="message"
77           variant="outlined"
78           label="Message"
```

```
          fullWidth
          multiline
          rows={4}
          value={postData.message}
          onChange={(e) =>
            setPostData({ ...postData, message:
  e.target.value })
          }
        />
        <TextField
          name="tags"
          variant="outlined"
          label="Tags (coma separated)"
          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"
```

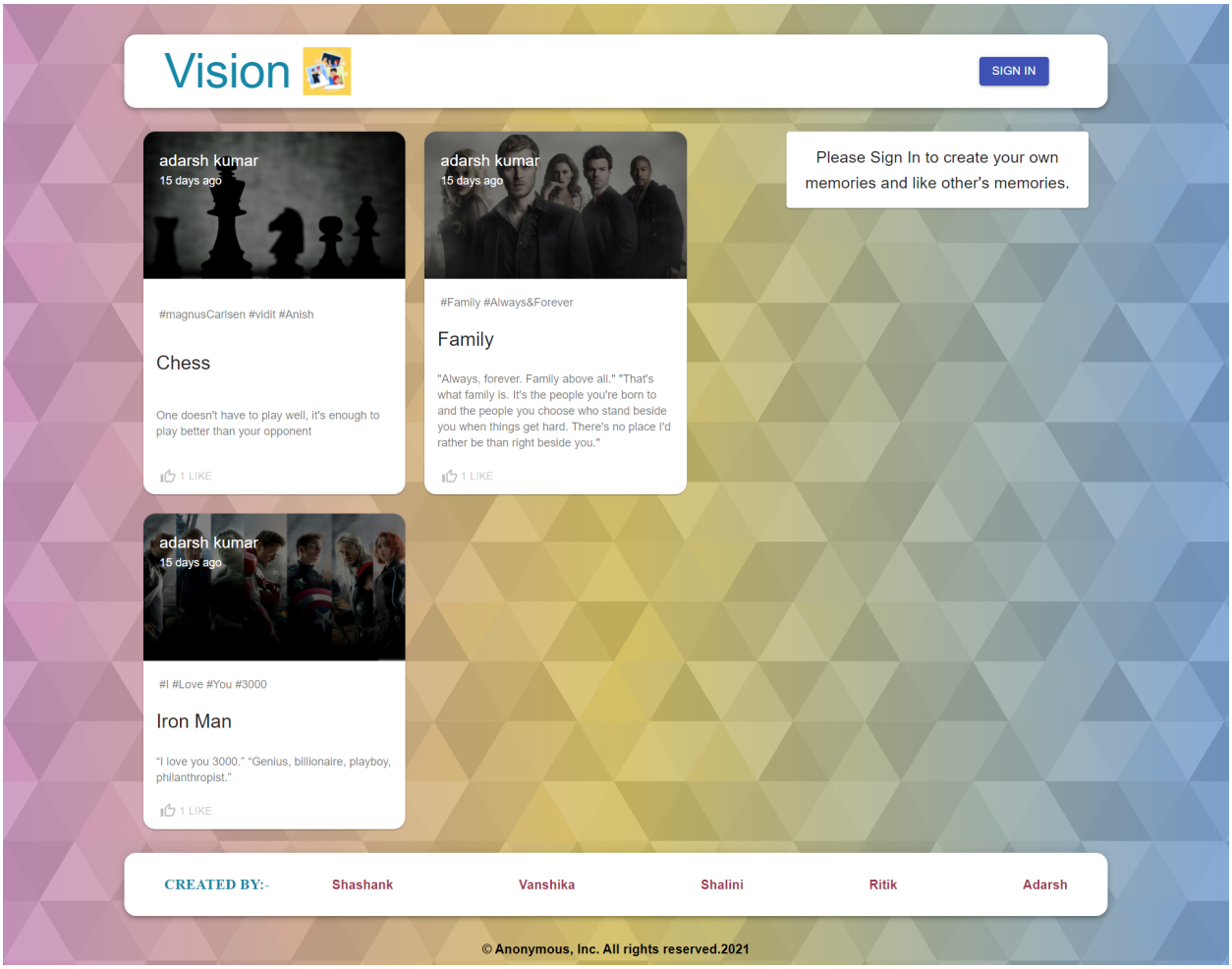
```
          fullWidth
        >
          Submit
        </Button>
        <Button
          variant="contained"
          color="secondary"
          size="small"
          onClick={clear}
          fullWidth
        >
          Clear
        </Button>
      </form>
    </Paper>
  );
};

export default Form;
```

styles.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,
  },
}));
```

Home.js




```
import React, { useState, useEffect } from "react";
import { Container, Grow, Grid } from "@material-ui/core";
import { useDispatch } from "react-redux";

```

```
5  import { getPosts } from "../../actions/posts";
6  import Posts from "../Posts/Posts";
7  import Form from "../Form/Form";
8
9  const Home = () => {
10   const [currentId, setCurrentId] = useState(0);
11   const dispatch = useDispatch();
12
13   useEffect(() => {
14     dispatch(getPosts());
15   }, [currentId, dispatch]);
16
17   return (
18     <Grow in>
19       <Container>
20         <Grid
21           container
22           justify="space-between"
23           alignItems="stretch"
24           spacing={3}
25         >
26           <Grid item xs={12} sm={7}>
27             <Posts setCurrentId={setCurrentId} />
28           </Grid>
29           <Grid item xs={12} sm={4}>
30             <Form currentId={currentId}
   setCurrentId={setCurrentId} />
31           </Grid>
32         </Grid>
33       </Container>
34     </Grow>
35   );
36 };
37
38 export default Home;
```

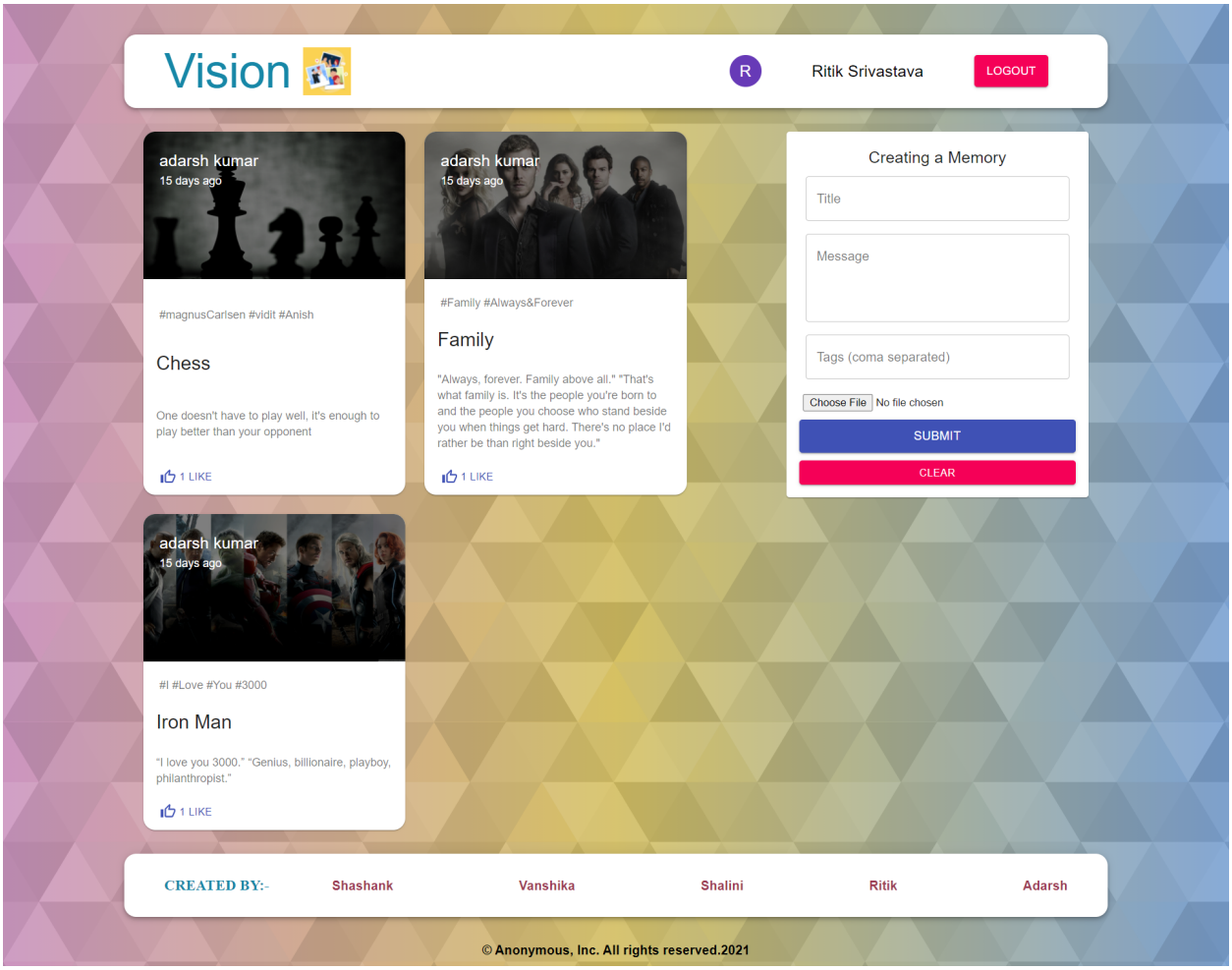
Navbar.js

```
import React, { useState, useEffect } from "react";
import { AppBar, Typography, Toolbar, Avatar, Button } from
"@material-ui/core";
import { Link, useHistory, useLocation } from
"react-router-dom";
import { useDispatch } from "react-redux";
import decode from "jwt-decode";

import memories from "../../images/memorie.jpg";
import * as actionType from "../../constants/actionTypes";
import useStyles from "./styles";

const Navbar = () => {
  const [user, setUser] =
useState(JSON.parse(localStorage.getItem("profile")));
  const dispatch = useDispatch();
  const location = useLocation();
  const history = useHistory();
  const classes = useStyles();

  const logout = () => {
    dispatch({ type: actionType.LOGOUT });

    history.push("/auth");

    setUser(null);
  };

  useEffect(() => {
    const token = user?.token;

    if (token) {
      const decodedToken = decode(token);

      if (decodedToken.exp * 1000 < new Date().getTime())
logout();
```

```
33      }
34
35      setUser(JSON.parse(localStorage.getItem("profile")));
36    }, [location]);
37
38    return (
39      <AppBar className={classes.appBar} position="static"
    color="inherit">
40        <div className={classes.brandContainer}>
41          <Typography
42            component={Link}
43            to="/"
44            className={classes.heading}
45            variant="h2"
46            align="center"
47          >
48            Memories
49          </Typography>
50          <img className={classes.image} src={memories}
    alt="icon" height="60" />
51        </div>
52        <Toolbar className={classes.toolbar}>
53          {user?.result ? (
54            <div className={classes.profile}>
55              <Avatar
56                className={classes.purple}
57                alt={user?.result.name}
58                src={user?.result.imageUrl}
59              >
60                {user?.result.name.charAt(0)}
61              </Avatar>
62              <Typography className={classes.userName}
    variant="h6">
63                {user?.result.name}
64              </Typography>
65              <Button
```

```
66                variant="contained"
67                className={classes.logout}
68                color="secondary"
69                onClick={logout}
70              >
71                Logout
72              </Button>
73            </div>
74          ) : (
75            <Button
76              component={Link}
77              to="/auth"
78              variant="contained"
79              color="primary"
80            >
81              Sign In
82            </Button>
83          )}
84        </Toolbar>
85      </AppBar>
86    );
87  };
88
89  export default Navbar;
```

Posts.js




```
import React from "react";
import {
  Card,
  CardActions,
  CardContent,
  CardMedia,
  Button,
  Typography,
} from "@material-ui/core/";
import ThumbUpAltIcon from "@material-ui/icons/ThumbUpAlt";
import DeleteIcon from "@material-ui/icons/Delete";
import MoreHorizIcon from "@material-ui/icons/MoreHoriz";
import ThumbUpAltOutlined from
  "@material-ui/icons/ThumbUpAltOutlined";
import { useDispatch } from "react-redux";
```

```
15 import moment from "moment";
16
17 import { likePost, deletePost } from
   "../../../actions/posts";
18 import useStyles from "./styles";
19
20 const Post = ({ post, setCurrentId }) => {
21   const dispatch = useDispatch();
22   const classes = useStyles();
23   const user = JSON.parse(localStorage.getItem("profile"));
24
25   const Likes = () => {
26     if (post.likes.length > 0) {
27       return post.likes.find(
28         (like) => like === (user?.result?.googleId ||
   user?.result?._id)
29       ) ? (
30         <>
31           <ThumbUpAltIcon fontSize="small" />
32            
33           {post.likes.length > 2
34             ? `You and ${post.likes.length - 1} others`
35             : `${post.likes.length} like${post.likes.length
   > 1 ? "s" : ""}`}
36         </>
37       ) : (
38         <>
39           <ThumbUpAltOutlined fontSize="small" />
40            {post.likes.length} {post.likes.length ===
   1 ? "Like" : "Likes"}
41         </>
42       );
43     }
44
45     return (
46       <>
```

```
        <ThumbUpAltOutlined fontSize="small" />
         Like
      </>
    );
  };

  return (
    <Card className={classes.card}>
      <CardMedia
        className={classes.media}
        image={
          post.selectedFile ||

"https://user-images.githubusercontent.com/194400/49531010-48dad180-f8b1-11e8-8d89-1e61320e1d82.png"
        }
        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>
      )}
```

```
81        <div className={classes.details}>
82          <Typography variant="body2" color="textSecondary"
  component="h2">
83            {post.tags.map((tag) => `#${tag} `)}
84          </Typography>
85        </div>
86        <Typography
87          className={classes.title}
88          gutterBottom
89          variant="h5"
90          component="h2"
91        >
92          {post.title}
93        </Typography>
94        <CardContent>
95          <Typography variant="body2" color="textSecondary"
  component="p">
96            {post.message}
97          </Typography>
98        </CardContent>
99        <CardActions className={classes.cardActions}>
100         <Button
101           size="small"
102           color="primary"
103           disabled={!user?.result}
104           onClick={() => dispatch(likePost(post._id))}
105         >
106           <Likes />
107         </Button>
108         {(user?.result?.googleId === post?.creator ||
109           user?.result?._id === post?.creator) && (
110           <Button
111             size="small"
112             color="secondary"
113             onClick={() => dispatch(deletePost(post._id))}
114           >
```

```
            <DeleteIcon fontSize="small" /> Delete
          </Button>
        )}
      </CardActions>
    </Card>
  );
};

export default Post;
```

posts.js

```
import React from "react";
import { Grid, CircularProgress } from "@material-ui/core";
import { useSelector } from "react-redux";

import Post from "./Post/Post";
import useStyles from "./styles";

const Posts = ({ setCurrentId }) => {
  const posts = useSelector((state) => state.posts);
  const classes = useStyles();

  return !posts.length ? (
    <CircularProgress />
  ) : (
    <Grid
      className={classes.container}
      container
      alignItems="stretch"
      spacing={3}
    >
      {posts.map((post) => (
        <Grid key={post._id} item xs={12} sm={6} md={6}>
          <Post post={post} setCurrentId={setCurrentId} />
        </Grid>
```

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

export default Posts;
```

CONSTANTS

actionTypes.js

```
export const CREATE = "CREATE";
export const UPDATE = "UPDATE";
export const DELETE = "DELETE";
export const FETCH_ALL = "FETCH_ALL";
export const LIKE = "LIKE";

export const AUTH = "AUTH";
export const LOGOUT = "LOGOUT";
```

REDUCERS

auth.js

```
import * as actionType from "../constants/actionTypes";

const authReducer = (state = { authData: null }, action) =>
{
  switch (action.type) {
    case actionType.AUTH:
      localStorage.setItem("profile", JSON.stringify({
...action?.data }));

      return { ...state, authData: action.data, loading:
false, errors: null };
    case actionType.LOGOUT:
      localStorage.clear();

      return { ...state, authData: null, loading: false,
```

```
  errors: null };
    default:
      return state;
  }
};

export default authReducer;
```

index.js

```
import { combineReducers } from "redux";

import posts from "./posts";
import auth from "./auth";

export const reducers = combineReducers({ posts, auth });
```

posts.js

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

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

```
      return posts.map((post) =>
        post._id === action.payload._id ? action.payload :
  post
      );
    case DELETE:
      return posts.filter((post) => post._id !==
  action.payload);
    default:
      return posts;
  }
};
```

App.js

```
import React from "react";
import { Container } from "@material-ui/core";
import { BrowserRouter, Switch, Route } from
  "react-router-dom";

import Home from "./components/Home/Home";
import Navbar from "./components/Navbar/Navbar";
import Footer from "./components/Footer/Footer"
import Auth from "./components/Auth/Auth";

const App = () => (
  <BrowserRouter>
    <Container maxWidth="lg">
      <Navbar />
      <Switch>
        <Route path="/" exact component={Home} />
        <Route path="/auth" exact component={Auth} />
      </Switch>
      <Footer/>
    </Container>
  </BrowserRouter>
);

```

```
export default App;
```

App.css

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
```

```
  }
  to {
    transform: rotate(360deg);
  }
}
```

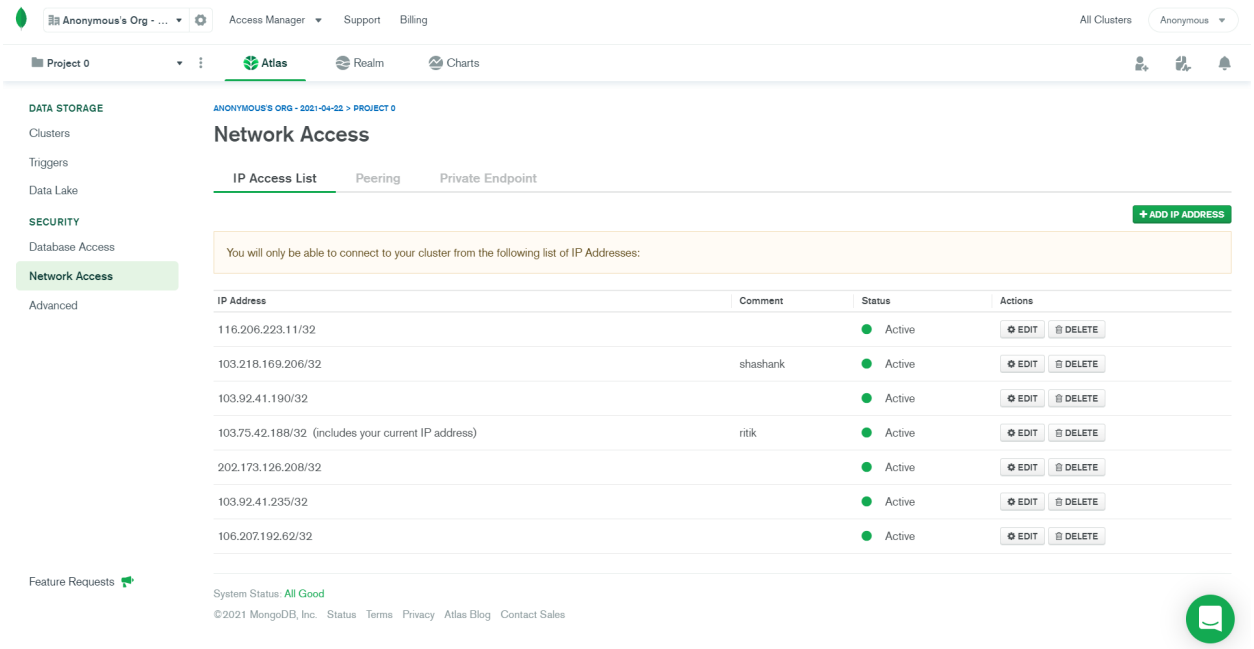
index.js

```
import React from "react";
import ReactDOM from "react-dom";
import { Provider } from "react-redux";
import { createStore, applyMiddleware, compose } from
"redux";
import thunk from "redux-thunk";

import { reducers } from "./reducers";
import App from "./App";
import "./index.css";

const store = createStore(reducers,
compose(applyMiddleware(thunk)));

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById("root")
);
```

Server

CONTROLLERS

posts.js



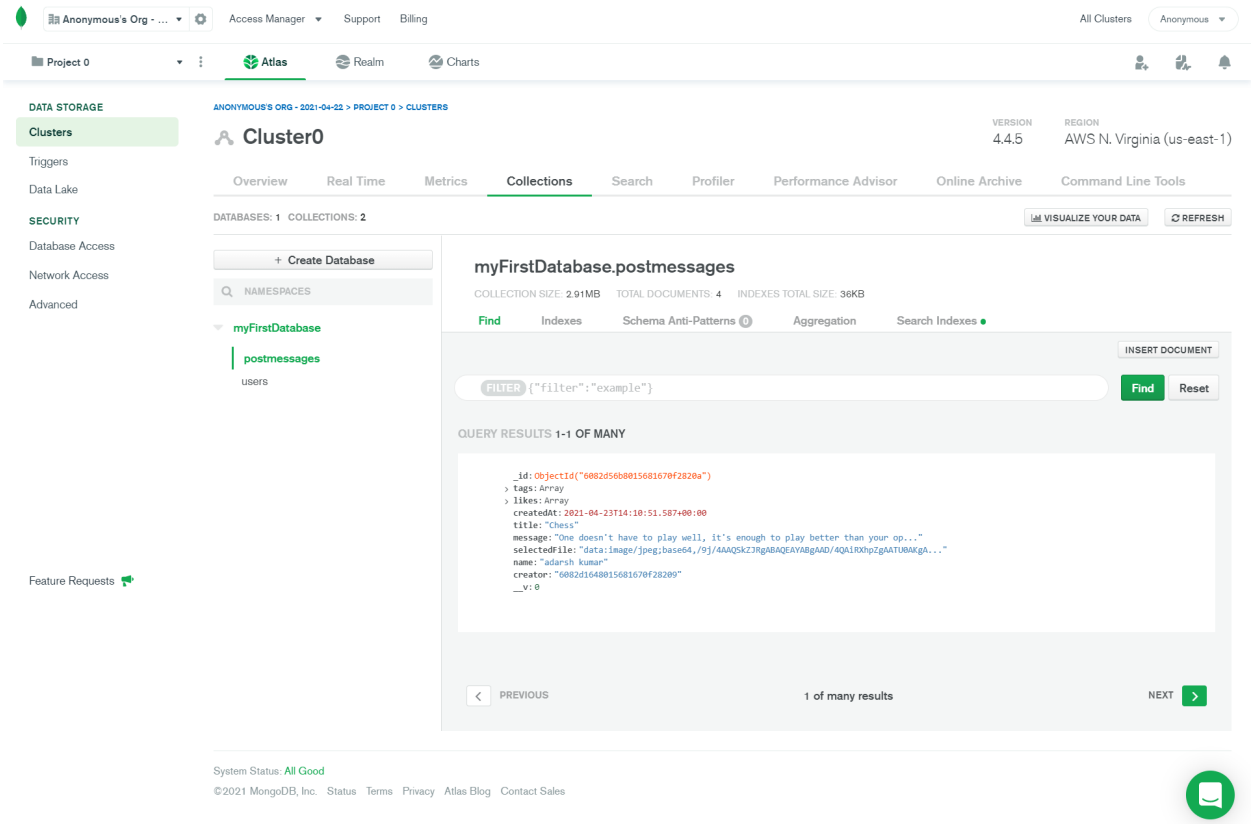

```
1  import express from "express";
2  import mongoose from "mongoose";
3
4  import PostMessage from "../models/postMessage.js";
5
6  const router = express.Router();
7
8  export const getPosts = async (req, res) => {
9    try {
10     const postMessages = await PostMessage.find();
11
12     res.status(200).json(postMessages);
13   } catch (error) {
14     res.status(404).json({ message: error.message });
15   }
16 };
17
18 export const getPost = async (req, res) => {
19   const { id } = req.params;
20
21   try {
22     const post = await PostMessage.findById(id);
```

```
23
24      res.status(200).json(post);
25    } catch (error) {
26      res.status(404).json({ message: error.message });
27    }
28  };
29
30  export const createPost = async (req, res) => {
31    const post = req.body;
32
33    const newPostMessage = new PostMessage({
34      ...post,
35      creator: req.userId,
36      createdAt: new Date().toISOString(),
37    });
38
39    try {
40      await newPostMessage.save();
41
42      res.status(201).json(newPostMessage);
43    } catch (error) {
44      res.status(409).json({ message: error.message });
45    }
46  };
47
48  export const updatePost = async (req, res) => {
49    const { id } = req.params;
50    const { title, message, creator, selectedFile, tags } =
    req.body;
51
52    if (!mongoose.Types.ObjectId.isValid(id))
53      return res.status(404).send(`No post with id: ${id}`);
54
55    const updatedPost = { creator, title, message, tags,
    selectedFile, _id: id };
56
57    await PostMessage.findByIdAndUpdate(id, updatedPost, {
```

```
  new: true });
58
59   res.json(updatedPost);
60 };
61
62 export const deletePost = async (req, res) => {
63   const { id } = req.params;
64
65   if (!mongoose.Types.ObjectId.isValid(id))
66     return res.status(404).send(`No post with id: ${id}`);
67
68   await PostMessage.findByIdAndRemove(id);
69
70   res.json({ message: "Post deleted successfully." });
71 };
72
73 export const likePost = async (req, res) => {
74   const { id } = req.params;
75
76   if (!req.userId) {
77     return res.json({ message: "Unauthenticated" });
78   }
79
80   if (!mongoose.Types.ObjectId.isValid(id))
81     return res.status(404).send(`No post with id: ${id}`);
82
83   const post = await PostMessage.findById(id);
84
85   const index = post.likes.findIndex((id) => id ===
  String(req.userId));
86
87   if (index === -1) {
88     post.likes.push(req.userId);
89   } else {
90     post.likes = post.likes.filter((id) => id !==
  String(req.userId));
91   }
```

```
92   const updatedPost = await
   PostMessage.findByIdAndUpdate(id, post, {
93     new: true,
94   });
95   res.status(200).json(updatedPost);
96 };
97
98 export default router;
```

user.js




```
1 import bcrypt from "bcryptjs";
2 import jwt from "jsonwebtoken";
3
4 import UserModal from "../models/user.js";
5
6 const secret = "test";
7
8 export const signin = async (req, res) => {
```

```
  const { email, password } = req.body;

  try {
    const oldUser = await UserModal.findOne({ email });

    if (!oldUser)
      return res.status(404).json({ message: "User doesn't exist" });

    const isPasswordCorrect = await bcrypt.compare(password, oldUser.password);

    if (!isPasswordCorrect)
      return res.status(400).json({ message: "Invalid credentials" });

    const token = jwt.sign({ email: oldUser.email, id: oldUser._id }, secret, {
      expiresIn: "1h",
    });

    res.status(200).json({ result: oldUser, token });
  } catch (err) {
    res.status(500).json({ message: "Something went wrong" });
  }
};

export const signup = async (req, res) => {
  const { email, password, firstName, lastName } = req.body;

  try {
    const oldUser = await UserModal.findOne({ email });

    if (oldUser)
      return res.status(400).json({ message: "User already
```

```
  exists" });

    const hashedPassword = await bcrypt.hash(password, 12);

    const result = await UserModal.create({
      email,
      password: hashedPassword,
      name: `${firstName} ${lastName}`,
    });

    const token = jwt.sign({ email: result.email, id:
result._id }, secret, {
      expiresIn: "1h",
    });

    res.status(201).json({ result, token });
  } catch (error) {
    res.status(500).json({ message: "Something went wrong"
});

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

MIDDLEWARE

auth.js

```
import jwt from "jsonwebtoken";

const secret = "test";

const auth = async (req, res, next) => {
  try {
    const token = req.headers.authorization.split(" ")[1];
    const isCustomAuth = token.length < 500;

    let decodedData;

```

```
    if (token && isCustomAuth) {
      decodedData = jwt.verify(token, secret);

      req.userId = decodedData?.id;
    } else {
      decodedData = jwt.decode(token);

      req.userId = decodedData?.sub;
    }

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

export default auth;
```

MODELS

postMessage.js

```
import mongoose from "mongoose";

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

var PostMessage = mongoose.model("PostMessage",
```

```
  postSchema);

export default PostMessage;
```

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);
```

ROUTES

posts.js

```
import express from "express";

import {
  getPosts,
  getPost,
  createPost,
  updatePost,
  likePost,
  deletePost,
} from "../controllers/posts.js";

const router = express.Router();
import auth from "../middleware/auth.js";

router.get("/", getPosts);
router.post("/", auth, createPost);
router.patch("/:id", auth, updatePost);
router.delete("/:id", auth, deletePost);
router.patch("/:id/likePost", auth, likePost);
```

```
20
21 export default router;
```

user.js

```
1  import express from "express";
2  const router = express.Router();
3
4  import { signin, signup } from "../controllers/user.js";
5
6  router.post("/signin", signin);
7  router.post("/signup", signup);
8
9  export default router;
```

index.js

```
1  import express from "express";
2  import bodyParser from "body-parser";
3  import mongoose from "mongoose";
4  import cors from "cors";
5
6  import postRoutes from "./routes/posts.js";
7  import userRouter from "./routes/user.js";
8
9  const app = express();
10
11 app.use(express.json({ limit: "30mb", extended: true }));
12 app.use(express.urlencoded({ limit: "30mb", extended: true
   }));
13 app.use(cors());
14
15 app.use("/posts", postRoutes);
16 app.use("/user", userRouter);
17
18 const CONNECTION_URL =
19
   "mongodb+srv://AnonymousDatabase:Anonymous@123@cluster0.p2p
```

```
  lb.mongodb.net/myFirstDatabase?retryWrites=true&w=majority"
  ;
20 const PORT = process.env.PORT || 5000;
21
22
23 mongoose
24   .connect(CONNECTION_URL, { useNewUrlParser: true,
  useUnifiedTopology: true })
25   .then(() =>
26     app.listen(PORT, () =>
27       console.log(`Server Running on Port:
  http://localhost:${PORT}`)
28     )
29   )
30   .catch((error) => console.log(`${error} did not
  connect`));
31
32 mongoose.set("useFindAndModify", false);
```

Scope for Extension in Major Project:

The web-application can be extended to cover a wide range of features like containing voice notes, video and other forms of multimedia. The memories can act as a future legacy to the future generation of the person. Posts can be shared to others platforms so the memories with other persons can also be shared. Filtering of the post will hep the person to save their time.

Conclusion:

In this pandemic the mental health of the people has been adversely affected. Many people are not able to connect with people or outside world and has no platform to share their vision. Our platform vision helps them connect them with their past memories and motivate them for future. Sometimes the motivation that be need lies within us and not outside.

Vision is a platform where users can post about their happy memories and recollect them when they need inspiration from their memories.

References:

1. Reference for React Documentation -> https://reactjs.org/
2. For UI ideas -> https://dribbble.com/
3. About Atom -> https://atom.io/docs
4. About Vs.-Code -> https://code.visualstudio.com/docs
5. Some of the contents are from the Internet.