

1. ABSTRACT

Our application “Confessionity” is a web-based application. This system has a centralized database to keep all the posts and comments of the users and as well as their account information such as password and email. Confessionity will help users to confess anything without compromising their identity.

This website is very helpful for people who are fighting with depression and other mental issues due to a thing or deed which haunts them and cannot seek any advice from others as their identity will be revealed. Our application helps them to seek advice on any kind of things and without compromising identity.

Our is website also doesn't allow anyone to post vulgar or insulting text or content as it can lead harm to other users. We have used a sentiment analysis engine which helps us to achieve that.

Currently there are few platforms which offer pseudonymous activity but here in Confessionity completely anonymous activity.

2. INTRODUCTION

There is currently no existing platform that allows users to confess with complete anonymity. There are many facebook groups that does so, where they get your confessions and then they post, but the admin knows everything as facebook is not anonymous.

To overcome these problems, we came up with a solution, and named it “Confessionity”.

1.1 Domain Description

The following tools are used in this project:

1. React

React is a popular JavaScript library for building user interfaces. It was developed by Facebook and is widely used for creating dynamic and interactive web applications. React follows a component-based approach, where UIs are divided into reusable and self-contained components that update efficiently based on changes in data. It utilizes a virtual DOM (Document Object Model) to efficiently render updates, resulting in faster and more responsive applications. React employs a declarative syntax, allowing developers to describe how the UI should look based on the application state. It supports both client-side and server-side rendering, making it versatile for different environments. With its strong ecosystem and

vast community support, React has become a go-to choice for developers seeking efficient and scalable UI development.

2. MongoDB

MongoDB is a popular NoSQL document-oriented database system that revolutionizes the way data is stored and accessed. Designed for scalability and high-performance, MongoDB excels in handling large volumes of unstructured and semi-structured data. With its flexible schema and JSON-like documents, it provides developers with the freedom to model and store data in a way that suits their application requirements. MongoDB boasts a rich query language and supports a wide range of operations, including indexing, aggregation, and geospatial queries. Its horizontal scaling capabilities make it well-suited for distributed architectures and cloud environments. MongoDB's extensive ecosystem includes a robust set of drivers, frameworks, and tools, making it a popular choice for building modern, scalable, and flexible applications.

3. Express

Express.js is a minimalistic and flexible web application framework for Node.js. It provides a lightweight and unopinionated approach to building web servers and APIs. With Express.js, developers can easily handle routes, middleware, and HTTP requests, making it ideal for building single-page applications, RESTful APIs, and microservices. It offers a modular structure, allowing

developers to choose from a vast ecosystem of middleware and plugins to add functionality as needed. Express.js promotes code simplicity and readability, enabling rapid development and easy maintenance. It also supports template engines for server-side rendering and provides robust error handling and routing capabilities. With its extensive community support and widespread adoption, Express.js remains a popular choice for creating fast and scalable web applications in Node.js.

4. Node

Node.js is a powerful and versatile JavaScript runtime environment built on the V8 engine. It allows developers to run JavaScript code on the server-side, enabling them to build scalable and high-performance applications. Node.js uses an event-driven, non-blocking I/O model, making it efficient and suitable for real-time applications, APIs, and microservices. It offers a rich ecosystem of modules and packages through its package manager, npm, allowing developers to easily extend the functionality of their applications. With its single-threaded, asynchronous nature, Node.js excels at handling concurrent requests and performing I/O operations efficiently. It is highly flexible, enabling the development of both server-side and full-stack applications. Node.js has gained significant popularity due to its speed, scalability, and ability to unify front-end and back-end development using JavaScript.

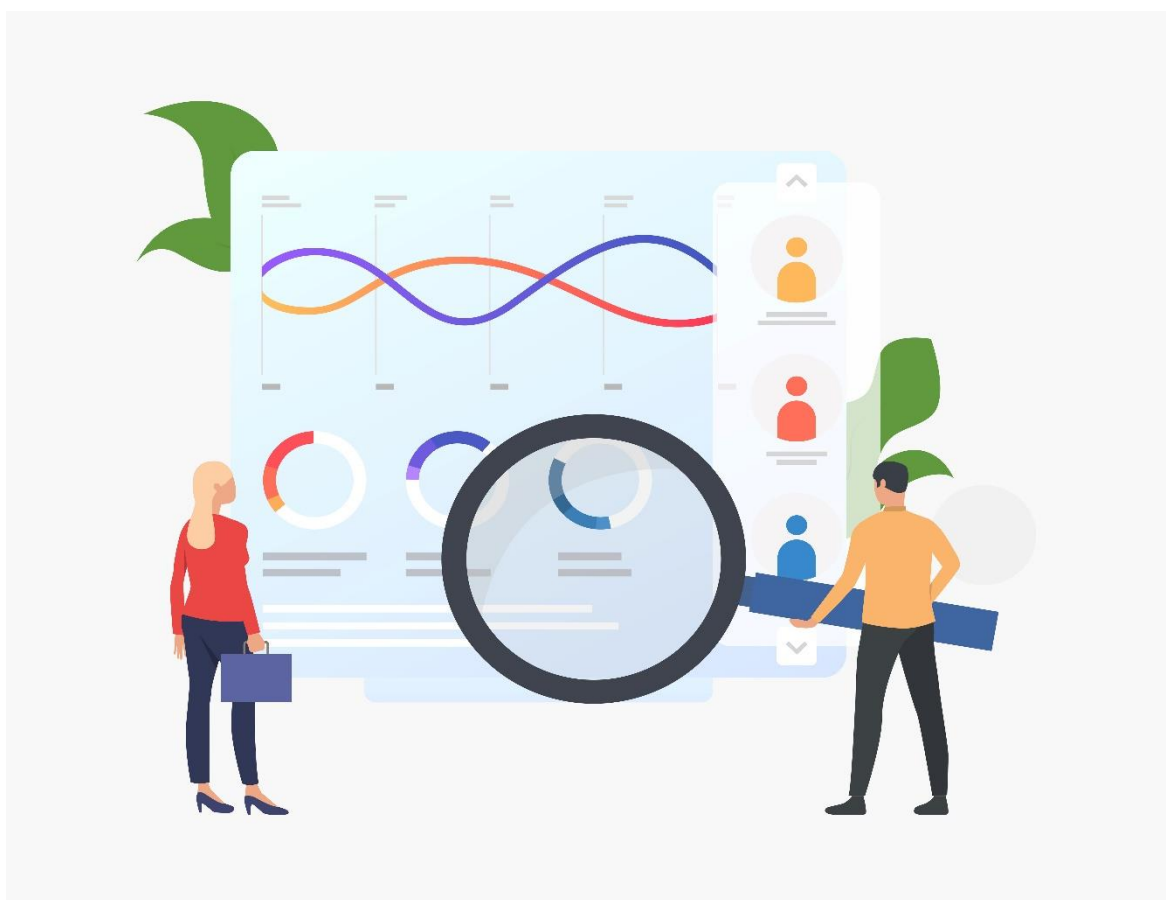
5. CSS

CSS (Cascading Style Sheets) is a fundamental language for web design and presentation. It is used to control the appearance and layout of HTML elements on web pages. With CSS, developers can define colors, fonts, spacing, borders, and other visual properties of elements, allowing for precise control over the presentation of web content. CSS follows a cascading rule, where styles can be defined in different ways and locations, providing flexibility and modularity. It supports selectors to target specific elements or groups of elements, and it offers a wide range of properties and values to customize the look and feel of a website. CSS is a crucial tool for creating responsive and visually appealing websites, providing separation between content and design and enhancing the overall user experience.

1.2 Motivation

In today's world many people are fighting with depression for not being able to confess something after knowing its consequences and many cases which even leads to suicide.

This problem motivated us to develop a platform such that people can talk to each other about their confessions and seek advises or further steps to be taken to tackle the deed they have committed. In doing so, keep them completely anonymous from the open world.



1.3 Scope

The objective of this application is to make depression not a common thing among today's generation or the so-called "GEN-Z" and save millions of people by simply [providing them with a community, completely anonymous, where they can discuss or confess freely and seek help or advise.

Our application is bound to provide complete anonymity and we take only your email to verify that you're a genuine person who wants to use our website and confess.



3. BACKGROUND RELATED WORK

In recent years, there has been an increasing emphasis on using technology in a good way to improve various aspects of our society. The need for such an efficient, secure and accessible platform will lead to a better society.

Existing Online Voting Systems

Currently there are no such platforms that offer complete anonymity for users to confess. For example, there are few facebook groups that allow users to confess something to the admin of the group and then the admin posts his/her confessions in the group timeline and other users can then suggest them through comments section. In this process the admin will get to know the details of the person who is confessing and the other users will also think twice to comment as their identity will also be revealed and others can associate him/her with that confession individually.

Limitations and Challenges

While developing this application we came across several limitations and challenges within the domain. Some of the common issues include:

Security - Ensuring the integrity and confidentiality of the platform so as to maintain a friendly environment within the application.

Trust -People may think that we will be using their data to show them ads and in case any hacker attacks out centralized database system, their data will be exposed. But we are true to our commitment, that we will not be taking any kind of personal information from our users. We only take their email id so as to verify the authenticity of the user.

Need for an anonymous confessing platform

Considering the limitations and challenges present in existing this system, there is a clear need for the development of a comprehensive and robust solution like the "Confessionity". This project aims to address the identified limitations and challenges by incorporating security measures, improving usability and accessibility, ensuring scalability and performance, and implementing effective confessions. By leveraging the use of email ids of users the "Confessionity" endeavors to provide a trustworthy, efficient, and inclusive platform for maintaining the authenticity of the user and as well as any bots and bugs.

Through the development of this project, we strive to make our society or rather the current generation mentally stable and strong and better.

4. METHODOLOGY

3.1 Problem Formulation

The first step to start any project is to formulate the problem existing in present projects like security concerns, anonymity. In providing complete anonymous environment, we must study the problems like,

If the user wants to post any hateful comments, he/she will not be able to post it until it conforms out sentiment analysis.

No sign of any details about the user so as to provide an environment where users can ask and recommend freely.

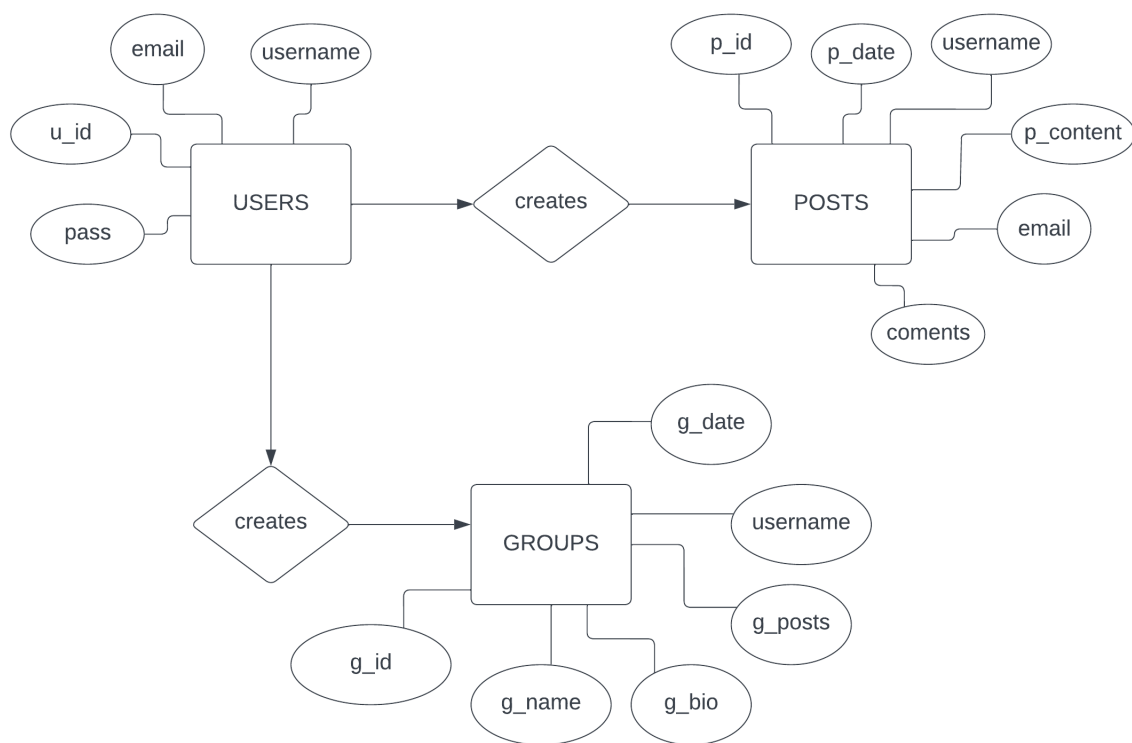
It will help the youngsters to fall less in depression and hence can live a peaceful and stress life.

3.2 Algorithm Description

3.2.1 List of Schemas Used in Database

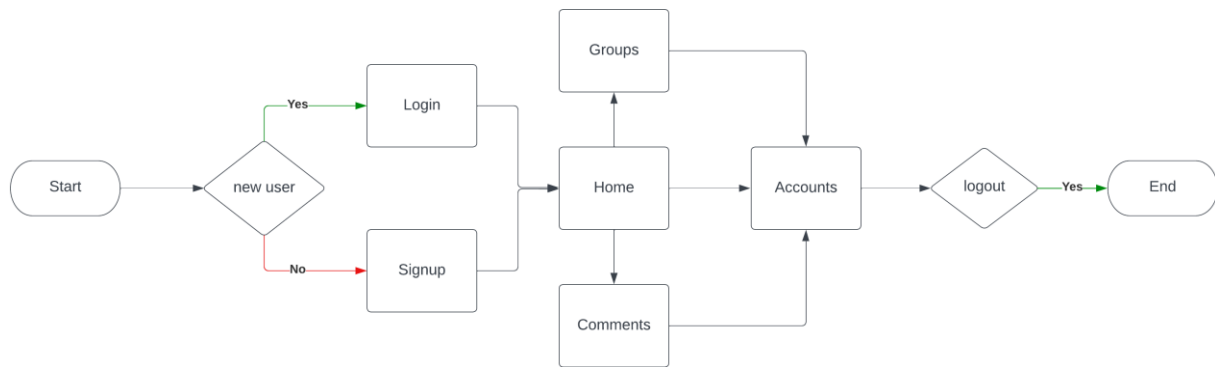
Schema	Names Of Attributes
USERS	username, email, u_id, pass
POSTS	p_id, p_date, username, p_content, email, comments
GROUPS	g_date, username, g_posts, g_id, g_name, g_bio

3.2.2 Entity-Relationship Diagram (ER Diagram)



Entity-Relationship Diagram

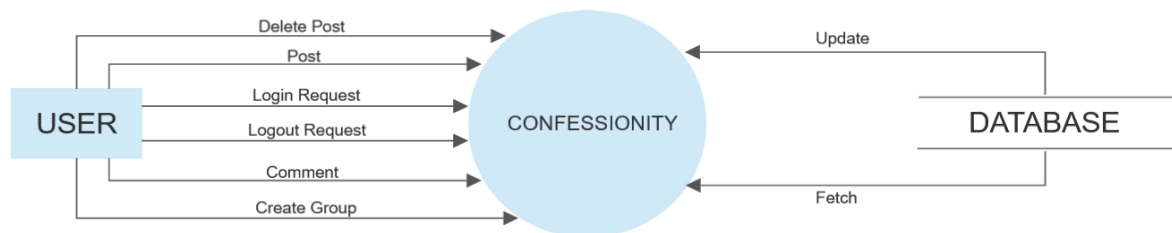
3.2.3 Flowchart



Flowchart

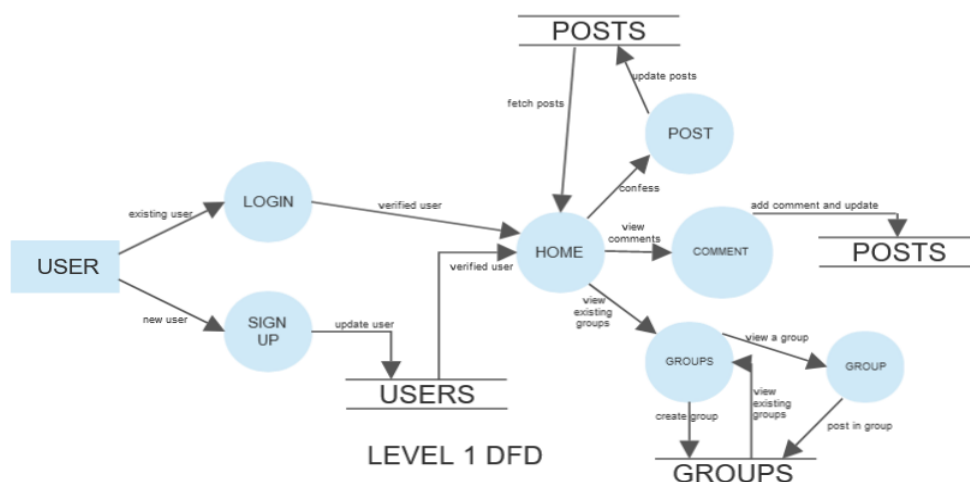
3.2.4 Data Flow Diagram (DFD)

- Level 0 DFD (Context Diagram)



Level 0 DFD

- Level 1 DFD



LEVEL 1 DFD

5. Implementation

Code for Home Screen –

```
import React, { useContext, useEffect, useState } from "react";
import "./Home.css";
import Navbar from "../Components/Navbar";
import axios from "axios";
import { Button, CircularProgress, Fab } from "@mui/material";
import { useContext } from "../App";
import { useNavigate } from "react-router-dom";
import Cookies from "universal-cookie";
import { BASE_URL } from "../utility/baseUrl";

function Home() {
  const cookies = new Cookies();
  const navigate = useNavigate();
  const { emailid, setEmailid, setUser, user } = useContext(userContext);
  const [posts, setPosts] = useState();
  const [comment, setComment] = useState();
  const [loading, setLoading] = useState(false);

  const handleDate = (dateee) => {
    const d = new Date(dateee);
    const date = d.toISOString().split("T")[0];
    const time = d.toString().split(" ")[0];
    return `${date} ${time}`;
  };

  useEffect(() => {
    const getdata = async () => {
      await axios
        .get(`${BASE_URL}/`)
        .then((res) => {
          setPosts(res.data);
          console.log(new Date(Date.now() + 2592000));

          console.log(res.data);
        });
    };
  });
}
```

```

    })
    .catch((e) => {
      console.log(e);
    });
  };

const verifyToken = async () => {
  const token = cookies.get("token");
  if (token) {
    const { data } = await axios.post(`${BASE_URL}/verifyToken`, {
      token: token,
    });
    if (data) {
      setEmailid(data?.email);
      setUser(data?.userid);
    }
  }
};

verifyToken();
getdata();
}, []);

const handleComment = (e) => {
  setComment(e.target.value);
};

const sendComment = async (params) => {
  setComment("");
  await axios.post(`${BASE_URL}/comments`, params);
};

return (
  <>
  <Navbar />
  <div className="progress">
    {!posts ? (
      <div className="preloader">

```

```

    <CircularProgress className="blue" />
    <div className="preloader-text">shhhhhh! Its loading</div>
  </div>
) : (
  ""
)
}
</div>
<div className="home">
  {posts
    ?.slice(0)
    .reverse()
    .map((post) => {
      return (
        <div className="home-all-posts">
          <div
            className="home-post"
            onClick={() => {
              const url = `/comment/${post?._id}`;
              navigate(url);
            }}
          >
            <div className="post-info">
              <span className="username">{post?.userid}</span>
              <span className="time">{handleDate(post?.date)}</span>
            </div>
            <div className="post-content">{post?.content}</div>
          </div>
          <div className="quick-comment">
            <div className="hero-comment">comments...</div>
            {user ? (
              <div className="qc">
                <input
                  type="text"
                  placeholder="write your comment..."
                  className="qc-input"
                  onChange={handleComment}
                />

```

```

<Button
  variant="contained"
  onClick={async () => {
    setLoading(true);
    if (comment.trim() != 0) {
      const encodedParams = new URLSearchParams();
      encodedParams.set("text", comment);

      const options = {
        method: "POST",
        url: "https://text-sentiment.p.rapidapi.com/analyze",
        headers: {
          "content-type":
            "application/x-www-form-urlencoded",
          "X-RapidAPI-Key":
            "28c02bb353msh2998e70e582daf6p1fc5a4jsnac03ff4e2649",
          "X-RapidAPI-Host":
            "text-sentiment.p.rapidapi.com",
        },
        data: encodedParams,
      };

      try {
        const response = await axios.request(options);
        const negs =
          response?.data?.neg_percent.split("%")[0];
        const neg = parseInt(negs);
        if (neg < 50) {
          sendComment({
            id: post?._id,
            comment: comment,
          });
        } else {
          alert(
            "due to use of negative comments, we cannot post."
          );
        }
      }
    }
  }}
/>

```



```

    }
  } catch (error) {
    alert("there was an error. Please try again.");
  }
  setLoading(false);
}
navigate(`/comment/${post?._id}`);
}}
>
  send
</Button>
</div>
): (
  ""
)}

```

```

<div className="latest-comment">
  {post?.comments[0] ? (
    <div className="first-comment">
      <div className="qc-comment">
        {post?.comments?.slice(-1)}
      </div>
      <div
        className="show-more-comments"
        onClick={() => {
          const url = `/comment/${post?._id}`;
          navigate(url);
        }}
      >
        <u>
          <i>more...</i>
        </u>
      </div>
    </div>
  ) : (
    "no comments yet..."
  )}

```

```
        </div>
      </div>
    </div>
  );
  }}}
</div>
</>
);
}
```

```
export default Home;
```

```
.groups-btn {
  z-index: 99999;
  position: sticky;
  /* left: 10px; */
  right: 10px;
  top: 5rem;
}
```

```
.login-msg {
  background-color: #242424;
  color: #646cff;
  font-size: large;
  display: flex;
  justify-content: center;
  align-items: center;
  width: auto;
  height: 100vh;
}
```

```
.home {
  margin-bottom: 1.5rem;
  width: auto;
  display: flex;
  flex-direction: column;
  align-items: center;
```

```
}
```

```
.home-all-posts {  
  display: flex;  
  flex-direction: column;  
  margin-top: 2rem;  
  padding: 20px;  
  width: 80rem;  
  height: max-content;  
  border: 2px solid black;  
  border-radius: 20px;  
}
```

```
.quick-comment {  
  margin: 1.5rem 0;  
  padding: 1rem;  
  border-top: 2px solid #242424;  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  flex-direction: column;  
}
```

```
.qc {  
  width: 75rem;  
  display: flex;  
  justify-content: space-between;  
}
```

```
.show-more-comments{  
  cursor: pointer;  
  font-size: smaller;  
}
```

```
.hero-comment{  
  font-size: large;
```

```
font-weight: 500;
width: 75rem;
padding-left: .5rem;
text-align: left;
}

.qc-input {
padding: .5rem;
width: 69rem;
border-radius: 1rem;
outline: none;
border: 1px solid #242424;
}

.latest-comment {
margin-top: 1rem;
padding: 1rem;
border: 1px solid #646cff;
border-radius: 1.5rem;
width: 75rem;
display: flex;
justify-content: flex-start;
}

.home-post {
flex: .9;
width: 80rem;
height: max-content;
/* border: 2px solid black; */
border-radius: 20px;
}

.comment-btn {
flex: .1;
}

.post-info {
```

```
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
}
```

```
.username {
    font-size: large;
    font-weight: 700;
}
```

```
.time {
    font-size: small;
}
```

```
.progress {
    display: flex;
    justify-content: center;
    align-content: center;
}
```

```
.blue {
    text-align: center;
    align-content: center;
}
```

```
.preloader {
    width: auto;
    height: 85vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

```
.preloader-text {
    margin-top: 1.5rem;
    text-align: center;
}
```

```
}

@media (max-width: 480px) {
  .home-all-posts {
    flex-direction: column;
    width: 20rem;
  }

  .username {
    font-size: smaller;
  }

  .home-post {
    width: auto;
  }

  .comment-btn {
    margin-top: 1rem;
  }

  .hero-comment{
    width: 18rem;
    margin-bottom: .8rem;
  }

  .qc{
    flex-direction: column;
    align-items: center;
  }

  .qc-input{
    width: 18rem;
    margin-bottom: 1rem;
  }

  .latest-comment{
    width: 18rem;
  }
}
```

Code for Login Screen –

```
import React, { useState, useContext } from "react";
import "./Login.css";
import { Button, CircularProgress } from "@mui/material";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import { userContext } from "../App";
import Cookies from "universal-cookie";
import { BASE_URL } from "../utility/baseUrl";

function Login() {
  const { emailid, errorText, setErrorText, setEmailid, setUser, user } =
    useContext(userContext);
  const navigate = useNavigate();
  const [email, setEmail] = useState();
  const [pass, setPass] = useState();
  const [error, setError] = useState();
  const [loading, setLoading] = useState(false);
  const cookies = new Cookies();

  const updateEmail = (e) => {
    setEmail(e.target.value);
  };

  const updatePass = (e) => {
    setPass(e.target.value);
  };

  const login = async () => {
    setLoading(true);
    await axios
      .get(
        `${BASE_URL}/login?email=${email}&pass=${pass}`
      )
      .then((res) => {
        cookies.set("token", res?.data?.token, { maxAge: 604800 });
        const status = res?.data.status;
```

```

const message = res?.data.message;
const userid = res?.data.userid;
const email = res?.data.email;
if (status === "success") {
  setErrorText(undefined);
  setEmailid(email);
  setLoading(false);
  setUser(userid);
  navigate("/", { replace: true });
} else {
  setLoading(false);
  setError(message);
  console.log(error);
}
});
};

return (
  <>
    <div className="error-text">{errorText}</div>
    <div className="login">
      <div className="login-form">
        <div className="hero">welcome back!</div>
        <div className="details">
          <input
            onChange={updateEmail}
            className="email"
            type="email"
            placeholder="email..."
          />
          <input
            onChange={updatePass}
            className="pass"
            type="password"
            placeholder="pass..."
          />
        </div>
      </div>
    </div>
  </div>

```



```

    <div className="error-message">{error}</div>
    <div className="btn">
      {loading ? (
        <CircularProgress />
      ) : (
        <Button onClick={login} variant="contained">
          login
        </Button>
      )}
    </div>
    <span>
      dont have an account? <a href="/signup">signup</a>{" "}
    </span>
  </div>
</div>
</>
);
}

```

```
export default Login;
```

```

.error-text{
  display: flex;
  justify-content: center;
  align-items: center;
  width: auto;
  height: 1.5rem;
  color: rgb(250, 72, 72);
  font-size: small;
  font-weight: 600;
}

```

```

.login{
  width: auto;
  height: 100vh;
  display: flex;
  align-items: center;
}

```

```
    justify-content: center;
}

.login-form{
  border: 2px solid black;
  border-radius: 25px;
  padding: 2rem;
  display: flex;
  flex-direction: column;
  align-items: center;
}

.hero{
  font-size: xx-large;
  font-weight: 600;
  margin-bottom: 0.3rem;
}

.details{
  display: flex;
  flex-direction: column;
  align-items: center;
}

.email, .pass {
  width: 15rem;
  margin: 0.8rem 0;
  padding: 10px;
  border: none;
  border-bottom: 1px solid black;
}

.error-message{
  color: rgb(250, 72, 72);
  font-size: smaller;
}
```

```
span{
  margin-top: 9px;
  font-size: smaller;
}

.btn{
  margin-top: 0.9rem;
}
```

Code for Signup Screen –

```
import React, { useContext, useState } from "react";
import "./Signup.css";
import { Button, CircularProgress } from "@mui/material";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import Cookies from "universal-cookie";
import { userContext } from "../App";
import { BASE_URL } from "../utility/baseUrl";

function Signup() {
  const { emailid, setEmailid, setUser, user } = useContext(userContext);
  const cookies = new Cookies();

  const navigate = useNavigate();
  const [email, setEmail] = useState();
  const [pass, setPass] = useState();
  const [cpass, setCpass] = useState();
  const [error, setError] = useState();
  const [match, setMatch] = useState();
  const [loading, setLoading] = useState(false);

  const updateEmail = (e) => {
    setEmail(e.target.value);
    console.log(email);
  };
}
```

```

const updatePass = (e) => {
  setPass(e.target.value);
  console.log(pass);
};

const updateCpass = (e) => {
  setCpass(e.target.value);
  console.log(pass);
};

const signup = async () => {
  setLoading(true);
  if (pass === cpass) {
    setMatch(undefined);
    await axios
      .post(`${BASE_URL}/signup`, {
        email: email,
        password: pass,
      })
      .then((res) => {
        cookies.set("token", res?.data?.token);
        const status = res?.data.status;
        const message = res?.data.message;
        const userid = res?.data.userid;
        const email = res?.data.email;
        console.log(status);
        if (status === "success") {
          setEmailid(email);
          setUser(userid);
          setLoading(false);
          navigate("/welcome", { replace: true });
        } else {
          setLoading(false);
          setError(message);
        }
      });
  }
};

```

```

    } else {
      setLoading(false);
      setMatch("password does not match");
    }
  };

return (
  <div className="signup">
    <div className="signup-form">
      <div className="hero">lets confession!</div>
      <div className="details">
        <input
          onChange={updateEmail}
          className="email"
          type="email"
          placeholder="email..."
        />
        <div className="error-message">{error}</div>
        <input
          onChange={updatePass}
          className="pass"
          type="password"
          placeholder="pass..."
        />
        <input
          onChange={updateCpass}
          className="confirmpass"
          type="password"
          placeholder="repeat pass..."
        />
        <div className="pass-match">{match}</div>
      </div>
      <div className="btn">
        {loading ? (
          <CircularProgress />
        ) : (
          <Button onClick={signup} variant="contained">

```

```
        signup
      </Button>
    )}
  </div>
  <span>
    already have an account? <a href="/login">login</a>{" "}
  </span>
</div>
</div>
);
}
```

```
export default Signup;
```

```
.signup{
  width: auto;
  height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
}
```

```
.signup-form{
  border: 2px solid black;
  border-radius: 25px;
  padding: 2rem;
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

```
.hero{
  font-size: xx-large;
  font-weight: 600;
  margin-bottom: 0.3rem;
}
```

```

.details{
  display: flex;
  flex-direction: column;
  align-items: center;
}

.email, .pass, .confirmpass {
  width: 15rem;
  margin: 0.8rem 0;
  padding: 10px;
  border: none;
  border-bottom: 1px solid black;
}

.pass-match, .error-message{
  color: rgb(250, 72, 72);
  font-size: smaller;
}

.btn{
  margin-top: 0.9rem;
}

```

Code for Posts Screen –

```

import React, { useState, useContext, useEffect } from "react";
import "./Post.css";
import { Button, CircularProgress } from "@mui/material";
import { createTheme, ThemeProvider } from "@mui/material/styles";
import Navbar from "../Components/Navbar";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import { userContext } from "../App";
import Cookies from "universal-cookie";
import { BASE_URL } from "../utility/baseUrl";

```

```

function Post() {
  const cookies = new Cookies();
  const { emailid, setErrorText, setEmailid, setUser, user } =
    useContext(userContext);

  const navigate = useNavigate();
  const [content, setContent] = useState("");
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    const verifyToken = async () => {
      const token = cookies.get("token");
      console.log(token);
      if (token) {
        const { data } = await axios.post(`${BASE_URL}/verifyToken`, {
          token: token,
        });
        if (data) {
          setEmailid(data?.email);
          setUser(data?.userid);
        }
      } else {
        setErrorText("login to continue");
        navigate("/login", { replace: true });
      }
    };
    verifyToken();
  }, []);

  const updatecontent = (e) => {
    setContent(e.target.value);
  };

  const postData = async () => {
    setLoading(true);
    if (content.trim() !== "") {
      const encodedParams = new URLSearchParams();
    }
  };

```



```

encodedParams.set("text", content.trim());

const options = {
  method: "POST",
  url: "https://text-sentiment.p.rapidapi.com/analyze",
  headers: {
    "content-type": "application/x-www-form-urlencoded",
    "X-RapidAPI-Key":
      "28c02bb353msh2998e70e582daf6p1fc5a4jsnac03ff4e2649",
    "X-RapidAPI-Host": "text-sentiment.p.rapidapi.com",
  },
  data: encodedParams,
};

try {
  const response = await axios.request(options);
  // if(response.data?.)
  const negs = response?.data?.neg_percent.split("%")[0];
  const neg = parseInt(negs);

  if (neg < 50) {
    console.log(neg);
    await axios
      .post(`${BASE_URL}/post`, {
        postContent: content,
        email: emailid,
        userid: user,
      })
      .then(function (response) {
        setLoading(false);
      })
      .catch(function (error) {
        // console.log(error);
      });
    alert("Posted Successfull");
    setContent("");
    navigate("/");
  }
}

```

```

    } else {
      setLoading(false);
      alert("due to use of negetive comments, we cannot post.");
    }
  } catch (error) {
    setLoading(false);
    alert("there was a error. Please try again");
  }
} else {
  setLoading(false);
  alert("write something before you post!!!");
}
};

```

```

const theme = createTheme({
  palette: {
    violet: {
      main: "#B2A4FF",
    },
  },
});

```

```

return (
  <>
    <Navbar />
    <div className="post">
      <div className="post-create">
        <textarea
          value={content}
          onChange={updatecontent}
          className="textarea"
          id="post"
          cols="60"
          rows="15"
          placeholder="write your post here..."
        ></textarea>
      </div>
    </div>
  </>
)

```

```

    <div className="post-btn">
      {loading ? (
        <CircularProgress />
      ) : (
        <ThemeProvider theme={theme}>
          <Button onClick={postData} color="violet" variant="contained">
            post
          </Button>
        </ThemeProvider>
      )}
    </div>
  </div>
</>
);
}

```

```
export default Post;
```

```

.post{
  width: auto;
  height: 85vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-evenly;
}

```

```

.textarea{
  font-size: medium;
  outline: none;
  padding: 1rem;
  border-radius: 20px;
  border: 2px solid black;
}

```

```

@media (max-width: 480px) {
  textarea {

```

```
    height: 15rem;
    width: 20rem;
  }
}
```

Code for My Posts Screen –

```
import React, { useEffect, useState, useContext } from "react";
import "./Myposts.css";
import { Button, CircularProgress } from "@mui/material";
import { createTheme, ThemeProvider } from "@mui/material/styles";
import Navbar from "../Components/Navbar";
import axios from "axios";
import { useContext } from "../App";
import { useNavigate } from "react-router-dom";
import Cookies from "universal-cookie";
import { BASE_URL } from "../utility/baseUrl";

function Myposts() {
  const cookies = new Cookies();
  const navigate = useNavigate();

  const { emailid, setErrorText, setEmailid, setUser, user } =
    useContext(userContext);
  const email = "random";
  const [myPosts, setMyPosts] = useState();
  const [effect, setEffect] = useState(true);

  const handleDate = (dateee) => {
    const d = new Date(dateee);
    const date = d.toISOString().split("T")[0];
    const time = d.toString().split(" ")[0];
    return `${date} ${time}`;
  };
}
```

```

useEffect(() => {
  const verifyToken = async () => {
    const token = cookies.get("token");
    console.log(token);
    if (token) {
      const { data } = await axios.post(`${BASE_URL}/verifyToken`, {
        token: token,
      });
      if (data) {
        setEmailid(data?.email);
        setUser(data?.userid);
      }
    } else {
      setErrorText("login to continue");
      navigate("/login", { replace: true });
    }
  };
  verifyToken();
  const getdata = async () => {
    await axios
      .get(`${BASE_URL}/my-post?email=${emailid}`)
      .then((res) => {
        setMyPosts(res.data);
        console.log(res.data);
      })
      .catch((e) => {
        console.log(e);
      });
  };

  getdata();
}, [effect]);

const theme = createTheme({
  palette: {
    delete: {
      main: "#F13E3E",

```

```

    },
    },
  });

return (
  <>
    <Navbar />
    <div className="progress">{!myPosts? (
      <div className="preloader">
        <CircularProgress/>
        <div className="preloader-text">
          shhhhh! its loading
        </div>
      </div>
    ):""}</div>
    <div className="myposts">
      {myPosts
        ?.slice(0)
        .reverse()
        .map((post) => {
          return (
            <div key={post?._id} className="all-posts">
              <div className="post-info">
                <span className="username">{post?.userid}</span>
                {/* <span className="email-mye">{post?.email}</span> */}
                <span className="time">{handleDate(post?.date)}</span>
              </div>
              <div className="post-content">{post?.content}</div>
              <div className="post-delete">
                <ThemeProvider theme={theme}>
                  <Button
                    onClick={() => {
                      axios.delete(`${BASE_URL}/delete-post`, {
                        data: { postId: post._id },
                      });
                      alert("post deleted")
                      setEffect(!effect);
                    }}
                  />
                </ThemeProvider>
              </div>
            </div>
          );
        })
      }
    </div>
  )
);

```

```

        }}
        variant="contained"
      >
        Delete
      </Button>
    </ThemeProvider>
  </div>
  <div className="my-comments-post">
    <span className="comment-hero">comments...</span>
    {post?.comments
      ?.slice(0)
      .reverse()
      .map((comment) => {
        return <div className="single-my-comment">{comment}</div>;
      })
    }
  </div>
</div>
);
}}
</div>
</>
);
}

```

```
export default Myposts;
```

```

.myposts {
  width: auto;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  margin-bottom: 2rem;
}

```

```

.all-posts {
  margin-top: 2rem;
}

```

```
padding: 20px;
width: 80rem;
height: max-content;
border: 2px solid black;
border-radius: 20px;
}

.post-info {
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
}

.username {
  font-size: large;
  font-weight: 700;
}

.email-mye {
  font-size: small;
  font-weight: 700;
}

.time {
  font-size: small;
}

.post-delete {
  padding-top: 10px;
  display: flex;
  justify-content: flex-start;
}

.my-comments-post {
  padding: 2rem;
}
```



```
.single-my-comment {
  padding: 1rem;
  border: 2px solid black;
  border-radius: 1.5rem;
  margin: 1rem 0;
}

/* .preloader{
  width: auto;
  height: 100vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}

.preloader-text {
  text-align: center;
} */

@media (max-width: 480px) {
  .all-posts {
    flex-direction: column;
    width: 20rem;
    /* background-color: #646cff; */
  }

  .username {
    font-size: smaller;
    color: black;
  }

  .home-post {
    width: auto;
  }

  .comment-btn {
```

```
        margin-top: 1rem;
    }

}
```

Code for Accounts Screen –

```
import React, { useContext, useEffect } from "react";
import Navbar from "../Components/Navbar";
import "../Account.css";
import { useNavigate } from "react-router-dom";
import { userContext } from "../App";
import Cookies from "universal-cookie";
import axios from "axios";
import { Button, ThemeProvider, createTheme } from "@mui/material";
import { BASE_URL } from "../utility/baseUrl";
```

```
function Account() {
  const theme = createTheme({
    palette: {
      violet: {
        main: "#6f5fc9",
      },
    },
  });
  const cookies = new Cookies();
  const navigate = useNavigate();
  const { emailid, setEmailid, setUser, user } = useContext(userContext);

  useEffect(() => {
    const verifyToken = async () => {
      const token = cookies.get("token");
      if (token) {
```

```

const { data } = await axios.post(
  `${BASE_URL}/verifyToken`,
  {
    token: token,
  }
);
if (data) {
  // console.log(data);
  setEmailid(data?.email);
  setUser(data?.userid);
}
console.log(token);
};
console.log("email", emailid);
verifyToken();
if (!emailid) {
  navigate("/login", { replace: true });
}
}, []);

const handleLogout = (e) => {
  setEmailid(undefined);
  setUser(undefined);
  cookies.remove("token");
  alert("sorry to see you go! 😞");
  navigate("/", { replace: true });
};

return (
  <>
    <Navbar />
    {console.log({ user })}
    <div className="container">
      <div className="left-userid">username: {user}</div>
      <div className="right-email">email: {emailid}</div>
    </div>
  </>
)

```

```

    <div className="logout-btn">
      <ThemeProvider theme={theme}>
        <Button onClick={handleLogout} color="violet" variant="contained">
          logout
        </Button>
      </ThemeProvider>
    </div>
    <div className="hero-text">
      <span>
        your <span className="anonymity">anonymity</span>, our responsibility
      </span>
    </div>
  </>
);
}

```

```
export default Account;
```

```

.container {
  width: auto;
  height: 10rem;
  display: flex;
  align-items: center;
}

```

```

.left-userid,
.right-email {
  font-size: larger;
  flex: 0.5;
  text-align: center;
}

```

```

.hero-text {
  margin-top: 6rem;
  display: flex;
  align-items: center;
  width: auto;
}

```

```
    height: 10rem;
    font-size: xx-large;
    font-weight: 600;
    justify-content: center;
}

.anonymity {
    color: #8875f1;
}

.logout-btn {
    width: auto;
    display: flex;
    justify-content: center;
    align-items: center;
}

.progress>span {
    margin: 0;
    padding: 0;
}

/* .preloader {
    width: auto;
    height: 100vh;
    display: flex;
    flex-direction: column;
    justify-content: space-around;
    align-items: center;
}

.preloader-text {
    text-align: center;
} */

@media (max-width: 480px) {
    .container {
```

```

        margin-top: 2rem;
        flex-direction: column;
    }

    .left-userid,
    .right-email {
        font-size: large;
    }

    .hero-text {
        margin-top: 1rem;
        font-size: x-large;
    }
}

```

Code for Groups Screen –

```

import React, { useContext, useEffect, useState } from "react";
import Navbar from "../Components/Navbar";
import { Button, CircularProgress, Input } from "@mui/material";
import "../Groups.css";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import { BASE_URL } from "../utility/baseUrl";
import Cookies from "universal-cookie";
import { useContext } from "../App";

function Groups() {
    const { emailid, setEmailid, setUser, user } = useContext(userContext);
    const cookies = new Cookies();
    const navigate = useNavigate();
    const [groups, setGroups] = useState();

    useEffect(() => {

```

```

const getgroups = async () => {
  await axios
    .get(`${BASE_URL}/groups`)
    .then((res) => {
      setGroups(res.data);
    });
};

const verifyToken = async () => {
  const token = cookies.get("token");
  if (token) {
    const { data } = await axios.post(`${BASE_URL}/verifyToken`, {
      token: token,
    });
    if (data) {
      // console.log(data);
      setEmailid(data?.email);
      setUser(data?.userid);
    }
  }
};

verifyToken();
getgroups();
}, []);

```

```

const handleCreateGrp = () => {
  navigate("/groups/create");
};

return (
  <>
    <Navbar />
    <div className="progress">{!groups? (
      <div className="preloader">
        <CircularProgress/>
        <div className="preloader-text">
          shhhhh! its loading
        </div>
      </div>
    )}
  </div>
)

```

```

):""}</div>
{user? (<div className="create-grp">
  <Button onClick={handleCreateGrp} variant="outlined">
    create
  </Button>
</div>):""}
<div className="all-grps">
  {groups?.map((group) => {
    return (
      <div
        onClick={() => {
          const url = `/groups/${group?._id}`;
          navigate(url);
        }}
        className="grp-info"
      >
        <div className="grp-name">{group?.name}</div>
        <div className="grp-bio">{group?.bio}</div>
      </div>
    );
  })}
</div>
</>
);
}

```

```
export default Groups;
```

```

.create-grp {
  position: fixed;
  bottom: 3rem;
  right: 3rem;
}

```

```

.all-grps {
  padding: 0 0 2rem 0;
  width: auto;
}

```



```
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
}

.grp-info {
margin-top: 2rem;
width: 80rem;
/* background-color: aqua; */
padding: 2rem;
border: 2px solid black;
border-radius: 20px;
display: flex;
justify-content: flex-start;
flex-direction: column;
}

.grp-name {
font-size: xx-large;
font-weight: 600;
}

.grp-bio {
margin-top: 0.6rem;
font-size: small;
}

@media (max-width: 480px) {
.grp-info {
width: 20rem;
}

.grp-name {
font-size: x-large;
font-weight: 600;
}
```

```

    .grp-bio {
      margin-top: 0.6rem;
      font-size: small;
    }
  }
}

```

Code for Specific Group Screen –

```

import React, { useContext, useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import axios from "axios";
import Navbar from "../Components/Navbar";
import "../Group.css";
import { Button, CircularProgress } from "@mui/material";
import { useContext } from "../App";
import { BASE_URL } from "../utility/baseUrl";

function Group() {
  const { emailid, setEmailid, setUser, user } = useContext(userContext);

  const [grpPost, setGrpPost] = useState();
  const [flag, setFlag] = useState(false);
  const [grpDetails, setGrpDetails] = useState();
  const [loading, setLoading] = useState(false);
  const params = useParams();
  const id = params.id;

  const handleDate = (dateee) => {
    const d = new Date(dateee);
    const date = d.toISOString().split("T")[0];
    const time = d.toTimeString().split(" ")[0];
    return `${date} ${time}`;
  };
}

```

```

useEffect(() => {
  const getDetails = async () => {
    await axios.get(`${BASE_URL}/group?id=${id}`).then((res) => {
      setGrpDetails(res.data);
      console.log(res.data);
    });
  };
  getDetails();
}, [flag]);

const handleGrpPost = async () => {
  setLoading(true);
  if (grpPost.trim !== "") {
    const encodedParams = new URLSearchParams();
    encodedParams.set("text", grpPost);

    const options = {
      method: "POST",
      url: "https://text-sentiment.p.rapidapi.com/analyze",
      headers: {
        "content-type": "application/x-www-form-urlencoded",
        "X-RapidAPI-Key":
          "28c02bb353msh2998e70e582daf6p1fc5a4jsnac03ff4e2649",
        "X-RapidAPI-Host": "text-sentiment.p.rapidapi.com",
      },
      data: encodedParams,
    };

    try {
      const response = await axios.request(options);
      const negs = response?.data?.neg_percent.split("%")[0];
      const neg = parseInt(negs);

      if (neg < 50) {
        const data = await axios.post(`${BASE_URL}/grp-post`, {
          admin: user,
          id: id,

```

```

        content: grpPost,
      });
      if (data) {
        setLoading(false);
        alert("posted successfully");
      }
      setGrpPost("");
      setFlag(!flag);
    } else {
      setLoading(false);
      alert("due to use of negetive comments, we cannot post.");
    }
  } catch (error) {
    setLoading(false);
    alert("There was error. Please Try again.");
  }
  setLoading(false);
}
};

const handleGrpInput = (e) => {
  setGrpPost(e.target.value);
};

return (
  <>
    <Navbar />
    <div className="progress"></div>
    <div className="grp-container">
      <div className="sin-grp-name">{grpDetails?.name}</div>
      {emailid ? (
        <div className="grp-post-input">
          <input
            type="text"
            value={grpPost}
            onChange={handleGrpInput}
            placeholder="write your post here..."
          />

```

```

    {loading ? (
      <CircularProgress />
    ) : (
      <Button onClick={handleGrpPost} variant="contained">
        Post
      </Button>
    )}
  </div>
) : (
  ""
)

{!grpDetails ? (
  <div className="preloading">
    <CircularProgress />
    <div className="preloader-text">
      shhhhhh! Its loading
    </div>
  </div>
) : (
  ""
)

<div className="grp-posts">
  {grpDetails?.posts
    ?.slice(0)
    .reverse()
    .map((post) => {
      return (
        <div className="grp-all-posts">
          <div className="grp-post">
            <div className="post-info">
              <span className="username">{post?.username}</span>
              <span className="time">{handleDate(post?.date)}</span>
            </div>
            <div className="post-content">{post?.content}</div>
          </div>
        </div>
      )
    })
  }

```

```
        </div>
      );
    }
  </div>
</div>
</>
);
}
```

```
export default Group;
```

```
.grp-container {
  margin: 1rem;
  padding: 2rem;
  border-radius: 2rem;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}
```

```
.sin-grp-name {
  font-size: xx-large;
  font-weight: 600;
  text-align: center;
}
```

```
.grp-all-posts {
  display: flex;
  margin-top: 2rem;
  padding: 20px;
  width: 80rem;
  height: max-content;
  border: 2px solid black;
  border-radius: 20px;
}
```

```
.grp-post-input {  
  margin-top: 1rem;  
  width: 70rem;  
  display: flex;  
  justify-content: space-evenly;  
}
```

```
.grp-post-input>input {  
  border-radius: .4rem;  
  width: 60rem;  
  border: none;  
  border: 1px solid black;  
  background: transparent;  
  padding: .4rem 1rem;  
  text-align: center;  
  outline: none;  
}
```

```
.progress>span {  
  margin: 0;  
}
```

```
.preloading{  
  width: auto;  
  height: 50vh;  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
  align-items: center;  
}
```

```
@media (max-width: 480px) {  
  .grp-post-input {  
    width: 20rem;  
    flex-direction: column;  
    justify-content: center;  
    align-items: center;  
  }
```

```
}

.grp-post-input>input {
  width: auto;
  border-radius: .4rem;
  margin-bottom: 1.5rem;
}

.grp-all-posts {
  flex-direction: column;
  width: 20rem;
}

.grp-post {
  width: auto;
}

.post-content {
  width: auto;
}
}
```


6. Results and Discussions

5.1 Results

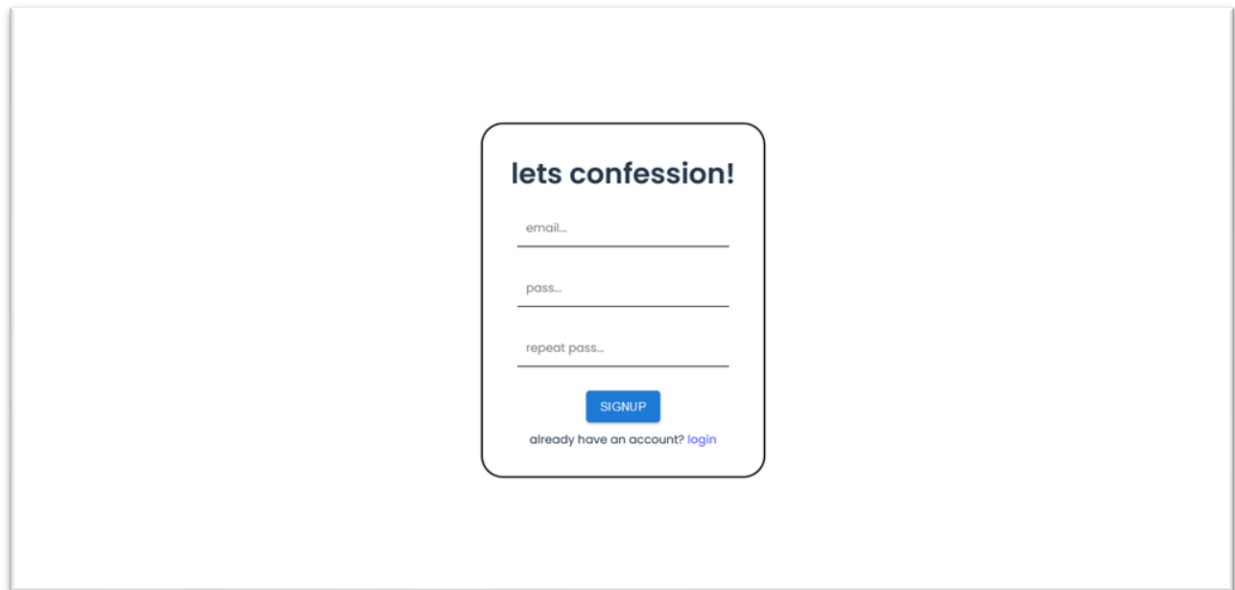
- ❖ This is the landing page where the people will not be logged in for the first time and can see the existing posts but cannot comment or post in any group.

The screenshot shows the Confessionity landing page. The header is purple with the text 'Confessionity' on the left, 'Home Groups' in the center, and a 'LOGIN' button on the right. Below the header, there are two post cards. The first post is by 'voluminous_jade_walrus' with a timestamp of '2023-05-13 21:51:05' and the text 'The best way to begin this new journey is a glass of wine and scrolling'. Below the post text is a 'comments...' section with a text input field containing 'bhal lagena' and a 'more...' link. The second post is by 'appropriate_copper_sole' with a timestamp of '2023-05-11 00:00:30' and the text 'A Lamborghini shall be mine.'.

- ❖ Login Screen, if the user is not registered then they can go to signup page.


The screenshot shows the login screen. It features a central white box with a black border. Inside the box, the text 'welcome back!' is at the top. Below it are two input fields labeled 'email...' and 'pass...'. A blue 'LOGIN' button is positioned below the password field. At the bottom of the box, the text 'dont have an account? [signup](#)' is displayed.

- ❖ Signup Screen, if the user is already registered, they can jump to login page.



A mockup of a signup screen. It features a central white card with rounded corners on a light gray background. The card has the heading "lets confession!" in bold. Below the heading are three input fields labeled "email...", "pass...", and "repeat pass...". A blue "SIGNUP" button is positioned below the input fields. At the bottom of the card, there is a link that says "already have an account? login".

- ❖ Home Screen (logged in), where user can view, comment on others posts, create groups and post in them. They can also delete their previous posts too.



A mockup of a home screen for a user who is logged in. The page has a purple header bar with the text "Confessionity" on the left and navigation links "Home Post My Posts Groups Account" on the right. The main content area displays two posts. The first post is by "voluminous_jade_walrus" and is dated "2023-05-13 21:51:05". The text of the post is "The best way to begin this new journey is a glass of wine and scrolling". Below the post is a "comments..." section with a text input field labeled "write your comment..." and a blue "SEND" button. A comment by "bhal lagena" is shown below the input field, with a "more..." link. The second post is by "appropriate_copper_sole" and is dated "2023-05-11 00:00:30". The text of the post is "A Lamborghini shall be mine."

❖ Post Screen - Users can post their content from here.



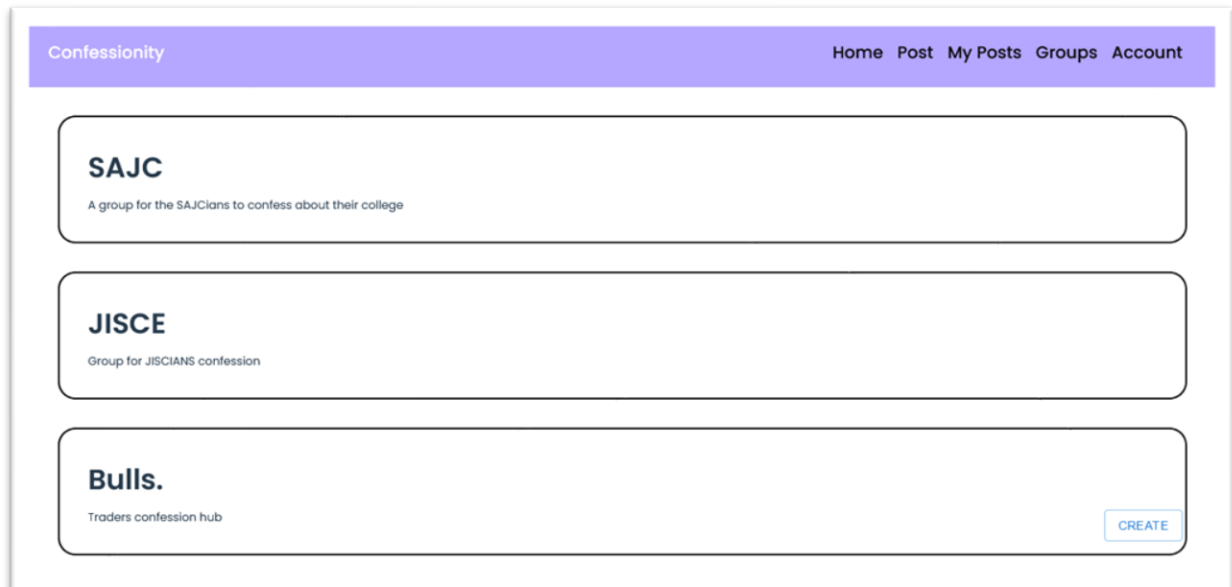
The image shows a web application interface for posting content. At the top, there is a purple navigation bar with the text 'Confessionity' on the left and a list of links 'Home Post My Posts Groups Account' on the right. Below the navigation bar, there is a large, empty rectangular text area with a thin black border and rounded corners. Inside this area, at the top left, is the placeholder text 'write your post here...'. At the bottom center of the text area, there is a small purple button with the text 'POST' in white capital letters.

❖ My Post Screen - Users can view their previous posts and delete them if required.

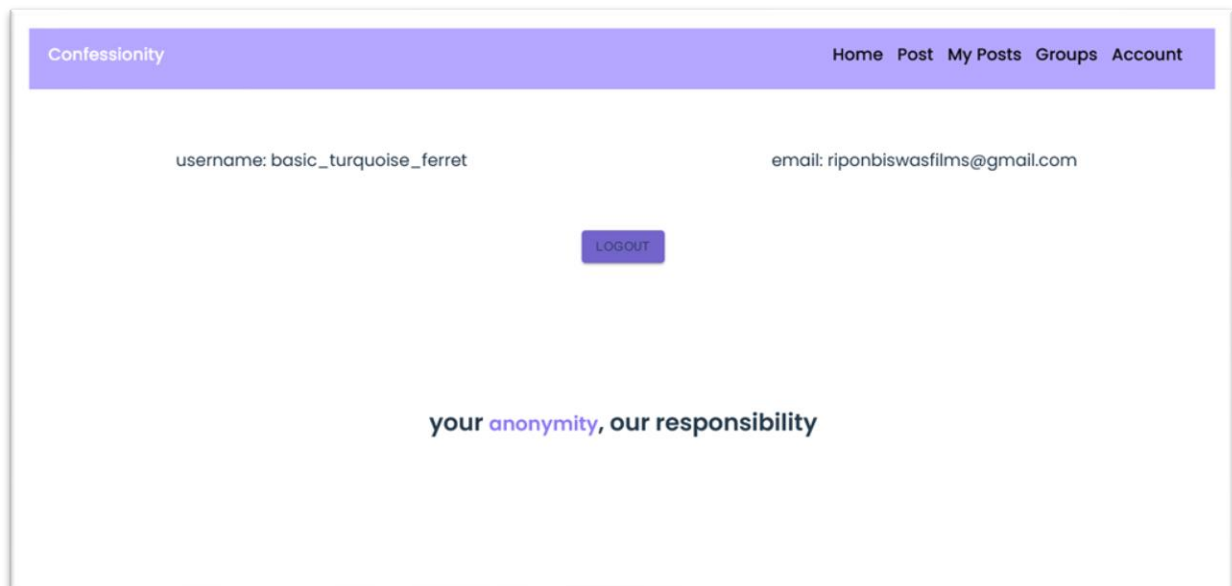


The image shows a web application interface for viewing and managing posts. At the top, there is a purple navigation bar with the text 'Confessionity' on the left and a list of links 'Home Post My Posts Groups Account' on the right. Below the navigation bar, there is a large rectangular container with a thin black border and rounded corners. Inside this container, at the top left, is the text 'basic_turquoise_ferret' in bold. Below this text is the date '2023-05-09 13:07:46' and a paragraph of text: 'This is my first post. A greate platform for us introverts. Waise mujhe samajhne ki kosish bhi mat karna. Main dil mein aata hu, dimag mein nahi.' Below the text is a small blue button with the text 'DELETE' in white capital letters. At the bottom of the container, there is a text input field with the placeholder text 'comments...'.

- ❖ Users can view the existing groups to join one or can create one of their own.



- ❖ Accounts page- Users can view their account details like username and password.



❖ Post details and comments screen - Users can view the existing comments and post a new comment too.

Confessionity

Home Post My Posts Groups Account

helpless_maroon_albatross
2023-05-11 19:40:55
I want to be a great trader.

write your comment here

>

comments...

mee tooo

Yes you can. Try harder.

❖ Specific Group Screen – Users can post in a specific group of their choice.

Confessionity

Home Post My Posts Groups Account

Bulls.

write your post here...

POST

basic_turquoise_ferret
2023-05-11 22:09:56
The thing is, I bring trading to emotions 😊

helpless_maroon_albatross
2023-05-11 19:42:41
Don't bring emotions in trading.

5.2 Discussion

As this website provides a better way of confessing securely. Hence, we suppose that this project has a greater scope and is important requirement is to provide a better mental health with less suicides.

The main feature of the project focuses on providing a more reliable, secured, fast, accurate and easy to handle system to confess ones deeds and get advices.

7. Advantages & Disadvantages of Confessionity

Advantages-

- **Anonymous** – Users can use this website without having to compromise their personal details and can confess anything freely without any tension.
- **No Hate /Abuses** – We have restricted our users to some comments or post which may relate to some kind of abusive behaviour or insults to maintain a friendly environment.
- **Easily Accessible** – Users can easily access this website using any PC or smartphone with a internet connection.
- **High Security** – The user's account details hashed using bcrypt to provide security.

Disadvantages-

- ❖ **May not get the right advise** – Users may not be able to suggest the right advise as per the users. They maybe dishonest and can provide false information.

8. Future Enhancements

Data can be managed on cloud so that it can be secured and managed more efficiently. This project is only a demonstration of the online confessing portal and this can be extended for commercial uses online across India by further enhancements.

For further verification of our users, we can use a OTP system to authenticate our user to prevent false and fishy accounts.

We can use the blockchain technology in our database rather than centralized database. Blockchain technology is like a fortress for data. The data is stored in multiple blocks, which are then linked together in a chain. This makes it very difficult to penetrate the fortress, as any attack would need to be made on all of the blocks in the chain. Additionally, the blocks are encrypted, which further protects the data from unauthorized access.

We can also introduce a feature where live calls can be provided to deal with depression or any other mental illness.

9. Conclusions

This website, named Confessionity, is a portal where users can confess their deeds without compromising their identity and in turn get advice from other users as to what to do next. This helps in clearing the load which they have by keeping it a secret.

There is a database maintained in which all the information of the accounts of the users are stored in the form of email and password only. We have taken email to verify the user to tackle false and fishy accounts our site.

By implementing this system or website, we can tackle depression up to a certain level as people can open up without having to take tension about their identity.

Last but not the least, we would like to thank our mentor, Prof. Rajesh Mishra for his constant guidance and support in the light of the completion of this project.

(Signature of Supervisor)

(Signature of External Examiner)

10. References

a. Websites-

- ❖ <https://react.dev/>
- ❖ <https://nodejs.org/en/docs>
- ❖ <https://expressjs.com/>
- ❖ <https://developer.mozilla.org/>
- ❖ <https://w3schools.com>
- ❖ <https://www.mongodb.com/>
- ❖ <https://courses.webdevsimplified.com/>
- ❖ <https://youtube.com/>

b. Books-

- ❖ Fundamentals of Software Engineering by Rajib Mall –
Published by PHI Learning.
- ❖ MongoDB: The Definitive Guide by Kristina Chodorow
and Michael Dirolf – Published by O'Reilly Media, Inc.

Thank
you