Full Stack Engineering

Project Report

Semester-VI (Batch-2022)

MindEase

Supervised By:

Rahul Sir

Submitted By:

Lalit Jindal-2210991847(G-24)

Krishana Raj-2210991816(G-24)

Khushi Agarwal-2210991799(G-24)

Liza-2210991853(G-24)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

ABSTRACT

MindEase is a next-generation mental health platform designed to provide accessible, secure, and anonymous interactions between users, volunteers, and consultants. The platform integrates AI-driven recommendations, real-time chat, and personalized resources such as books, music, and wellness videos. It aims to create a supportive digital ecosystem where users can seek help, track their mental health progress, and access a variety of wellness tools in a stigma-free environment.

The platform leverages advanced technologies such as AI-driven mood tracking, real-time chat systems with anonymity options, and secure data encryption to ensure privacy and reliability. Through AI-powered sentiment analysis, users receive personalized recommendations for mental health resources, self-care exercises, and professional guidance. The interactive user dashboard provides a seamless experience, allowing individuals to engage with consultants and volunteers effortlessly.

MindEase is built using a scalable MERN stack architecture to handle high user traffic while maintaining efficiency. The integration of cloud services like AWS/Firebase ensures smooth data storage and processing. Moreover, the use of WebSockets enables real-time communication, ensuring instant support for users in need.

This report presents the project's background, objectives, problem definition, methodology, results, and future improvements. It details how MindEase is positioned to address key challenges in mental health accessibility and engagement while maintaining a secure and personalized user experience.

Table of Content

Introduction

Mental health issues have become a global concern, affecting millions of individuals worldwide. Many people struggle in silence due to societal stigma, financial barriers, or the lack of easily accessible mental health support. Peer support networks have proven to be highly effective in helping individuals navigate emotional distress by connecting them with those who have successfully overcome similar challenges.

Mind Ease is a web-based platform designed to provide mental health support by facilitating connections between individuals facing mental health difficulties and volunteers who have experienced and overcome similar challenges. By leveraging modern technologies, including React.js for the frontend, Node.js and Express.js for the backend, and MongoDB as the database, the platform offers a safe, secure, and supportive community where users can seek guidance and emotional support. The addition of an AI-powered virtual assistant enhances accessibility, ensuring that users can find immediate assistance when needed.

2. Background and Significance

Mental health conditions such as anxiety, depression, and stress-related disorders are increasing at an alarming rate. However, many individuals hesitate to seek professional help due to fear of judgment, financial constraints, or lack of awareness. In response to these challenges, peer support communities have emerged as a practical and empathetic solution for mental health care.

Mind Ease is significant because it combines human empathy with technological innovation to create an engaging, user-friendly, and effective mental health support system. The importance of community-driven support cannot be overstated, as personal experiences and shared stories can foster hope, resilience, and healing. The platform’s key advantage is that it reduces isolation by creating meaningful connections, ensuring that no one has to navigate their struggles alone.

3. Objectives

The primary goal of Mind Ease is to bridge the gap between mental health sufferers and individuals who can provide support. The specific objectives of the project include:

To facilitate peer-based mental health support by connecting users with volunteers who have successfully navigated similar challenges.

To provide an AI-powered virtual assistant capable of offering instant emotional support, guidance, and resource recommendations.

To create an interactive, user-friendly, and accessible platform that ensures seamless communication between users and volunteers.

To eliminate mental health stigma by encouraging open discussions and fostering a supportive community.

To enhance accessibility to mental health resources by providing a free, digital-first solution for individuals across different locations.

4. Features and Functionality

To achieve these objectives, Mind Ease incorporates the following features:

a. User Registration and Profile Management

Users can create accounts with personal details, mental health concerns, and preferences.

Volunteers can register by providing background information about their past struggles and recovery journey.

b. Peer Support Matching System

A smart matching algorithm connects users with volunteers based on shared experiences, mental health needs, and language preferences.

Users can view volunteer profiles before initiating a conversation.

c. AI-Powered Virtual Assistant

Provides instant responses to users' questions and concerns.

Suggests coping strategies and self-care techniques.

Directs users to relevant resources, including emergency helplines if necessary.

d. Secure Messaging System

Encrypted, real-time chat ensures confidential and private communication between users and volunteers.

Users can schedule voice or video sessions for deeper interactions.

e. Community Forum and Resources

A moderated forum where users can share experiences, insights, and advice.

Educational articles, guided meditations, and self-help exercises are available.

5. Technology Stack

To ensure high performance, scalability, and security, Mind Ease is developed using the MERN

(MongoDB, Express.js, React.js, Node.js) stack:

Frontend: React.js (for a dynamic and responsive UI)

Backend: Node.js and Express.js (for API and business logic handling)

Database: MongoDB (for storing user data, messages, and resources)

AI Integration: Machine learning-based chatbot for virtual assistance

Authentication: JWT (JSON Web Tokens) for secure login and user sessions

Hosting & Deployment: Cloud-based servers for global accessibility

Problem Definition and Requirements

Problem Statement:

Mental health resources are often inaccessible due to social stigma, long waiting periods for professional consultations, and a lack of tailored content. MindEase seeks to create an inclusive and supportive platform where users can seek help without fear of judgment. Many individuals hesitate to reach out for support due to concerns about privacy, limited access to certified professionals, and the unavailability of structured self-help resources. The challenge lies in integrating AI-driven content personalization, real-time user engagement, and secure consultations into one seamless system.

Software & Hardware Requirements:

Frontend: React.js (for dynamic UI, responsive design, and optimized performance)

Backend: Node.js, Express.js (handling API requests, authentication, and real-time messaging)

Database: MongoDB (storing user data, chat logs, wellness resources, and system logs)

Security: JWT, bcrypt.js (user authentication and encryption for privacy protection)

Cloud Services: AWS/Firebase (scalability, hosting, real-time updates, and secure cloud storage)

Hardware: Server with a minimum 8GB RAM, SSD storage, and cloud deployment capability to handle high user loads and concurrency.

Proposed Design / Methodology

System Architecture:

MindEase follows a MERN stack architecture, ensuring scalability and efficient user interactions. The platform consists of:

Frontend: Built with React.js, utilizing reusable components, modular design, and state management with Redux for seamless performance. The UI is designed to be minimalistic and distraction-free, ensuring a user-friendly and engaging experience.

Backend: Node.js with Express.js, handling authentication, database queries, and API responses with optimized routing and middleware processing. The backend ensures secure and efficient data handling, supporting high-volume concurrent requests.

Database: MongoDB, structured for efficient retrieval, high availability, and scalable NoSQL architecture to handle diverse user data. The database is optimized with indexing, caching, and partitioning strategies to enhance performance.

AI-Powered Modules: Incorporating machine learning for mood prediction, content personalization, and sentiment analysis. These modules enhance user engagement by dynamically suggesting resources and tracking emotional progress.

File Structure:

Frontend: Organized into Components, Pages, Redux Store, Styles, Utility Functions, and API Services.

Backend: Modular structure including Controllers, Routes, Models, Middleware, Services, and Configurations.

Database Models: Efficiently structured models for storing User Profiles, Chat Messages, Resource Libraries, Feedback, Analytics, and Session Logs.

Algorithms Used:

AI-based Recommendation Engine: Suggests mental health resources, guided sessions, and expert advice based on user preferences and interactions. This engine is trained on user behavior data, sentiment analysis results, and historical engagement patterns.

Real-time Chat System: Uses WebSockets for instant messaging, ensuring encrypted and private conversations. The chat system supports multimedia messages, voice notes, and emergency contact features.

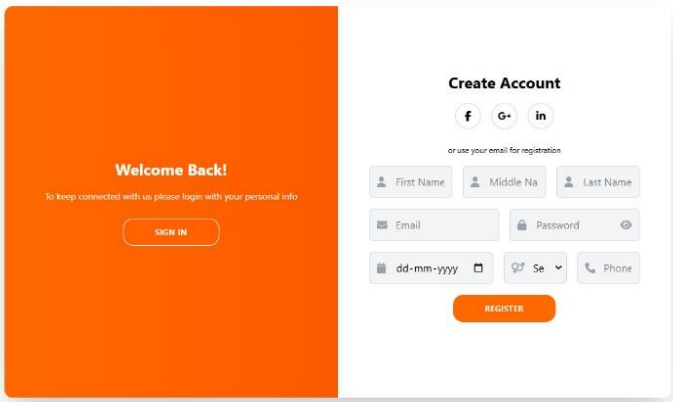
Sentiment Analysis Model: Uses Natural Language Processing (NLP) to analyze user emotions and provide appropriate recommendations. The model detects mood fluctuations and adjusts content delivery to enhance user support.

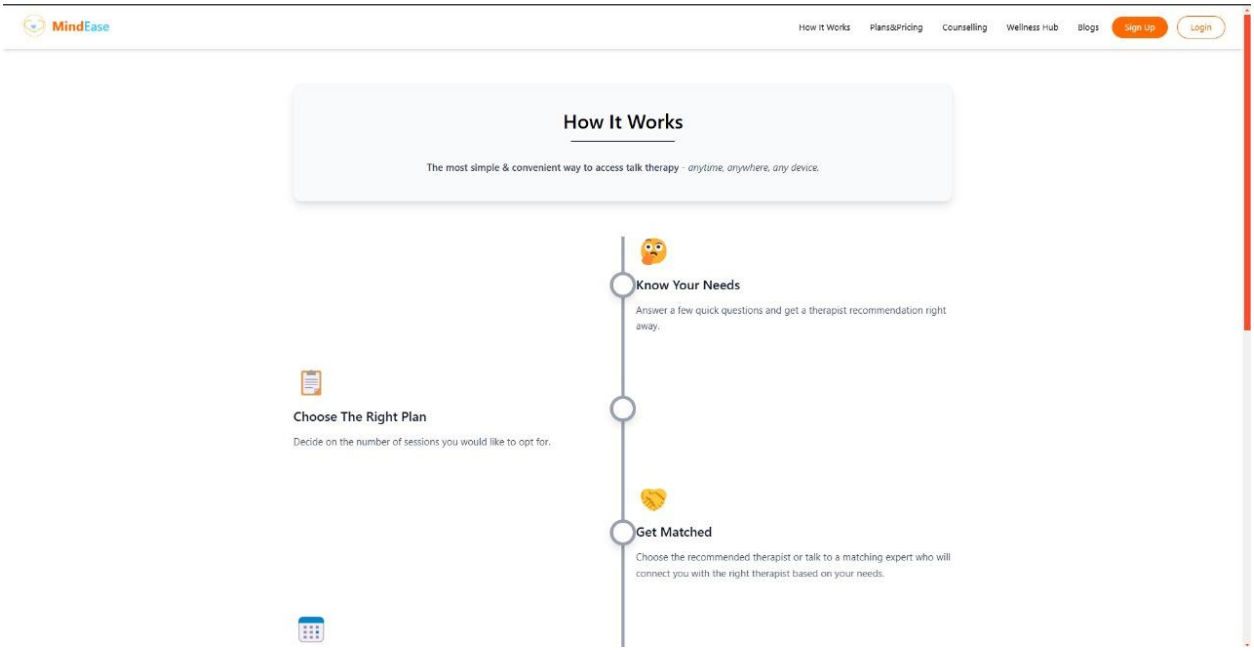
Machine Learning-based Mood Tracker: Predicts mood patterns based on historical data and user interactions. It employs time-series analysis to monitor emotional changes and suggest relevant coping strategies.

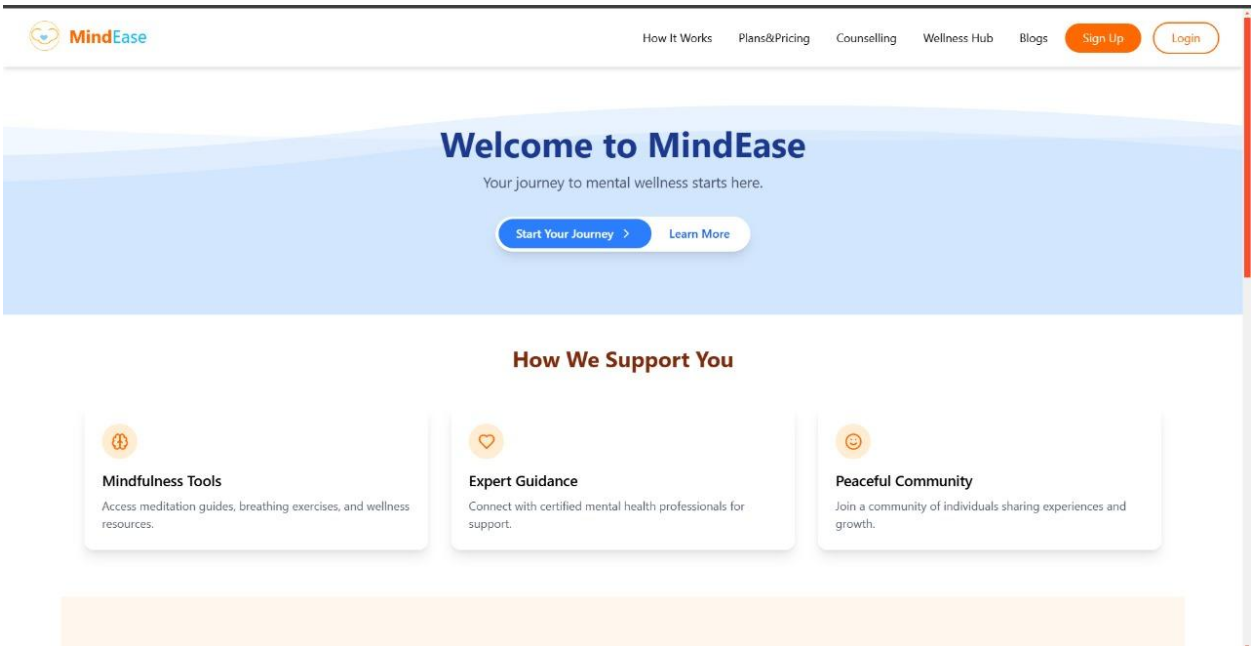
Anomaly Detection System: Identifies signs of distress or crisis based on user interaction patterns and triggers alerts for immediate intervention.

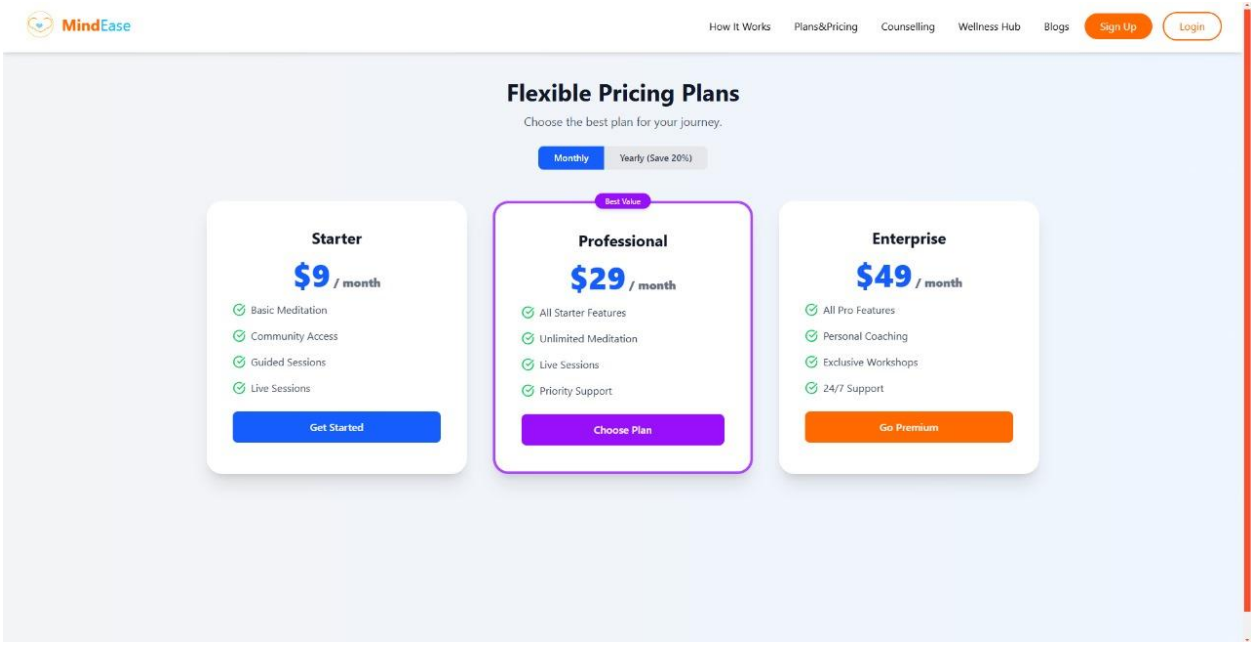
Personalized Learning Model: Adapts user content recommendations based on feedback, engagement levels, and preferences, ensuring a tailored support experience

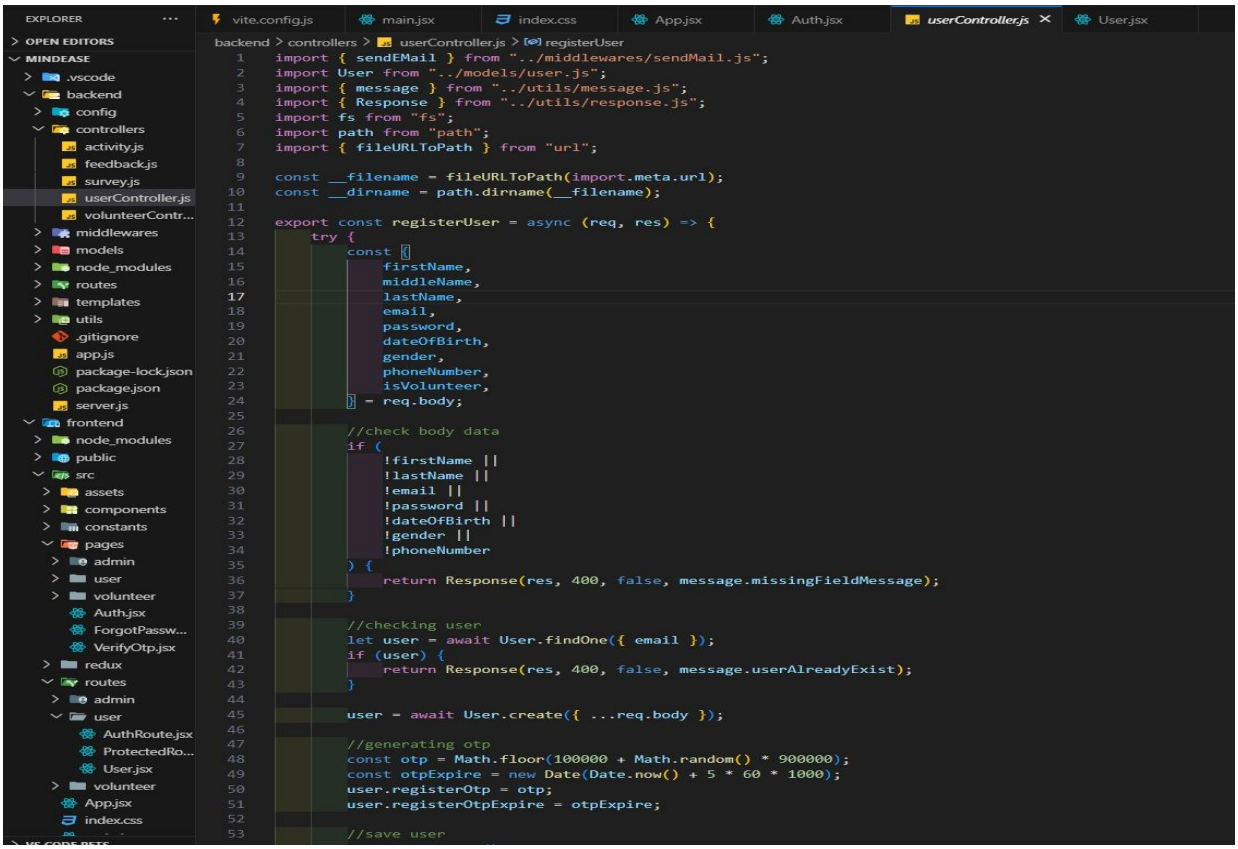
Results

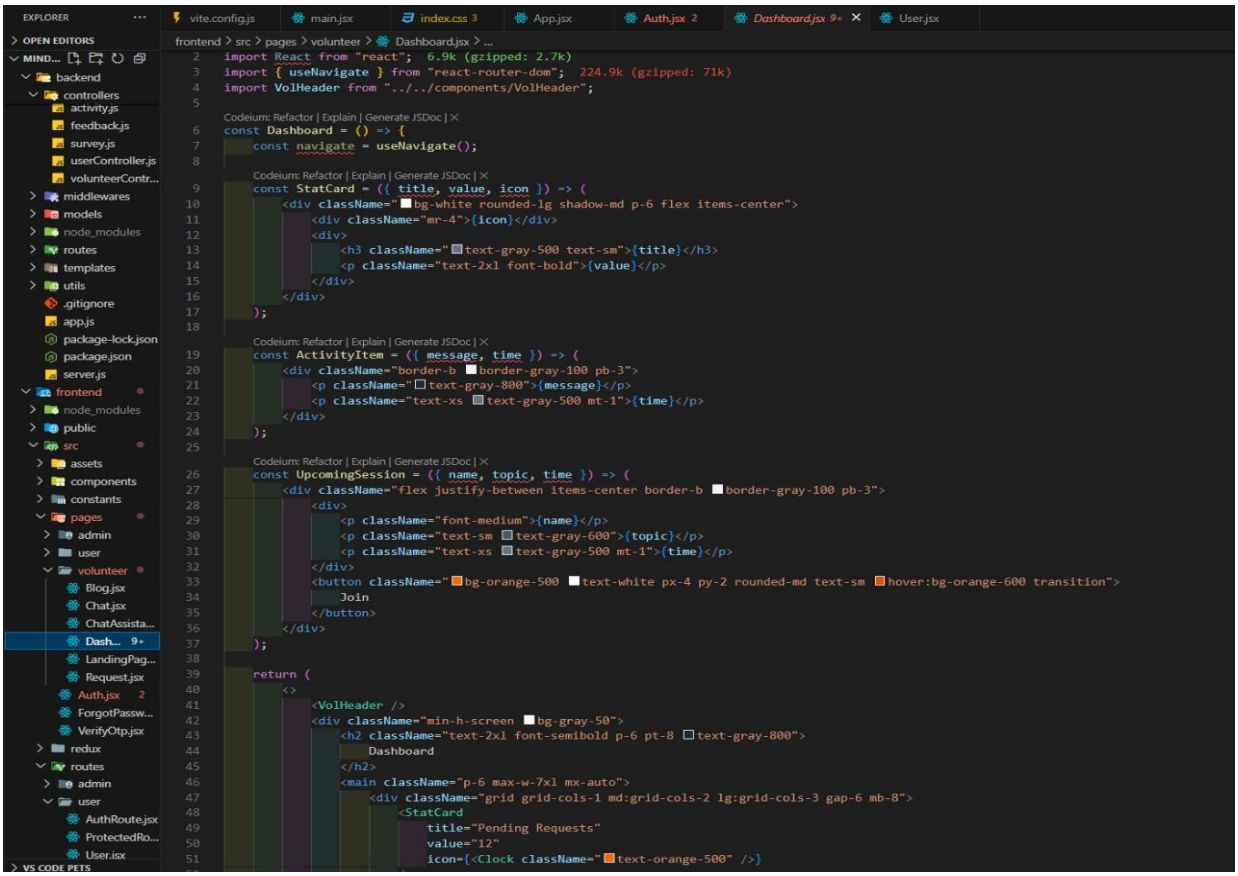

Welcome Back!
To keep connected with us please login with your personal info
SIGN IN
Create Account
or use your email for registration
First Name
Middle Na
Last Name
Email
Password
dd-mm-yyyy
Se
Phone
REGISTER


MindEase
How It Works
Plans&Pricing
Counselling
Wellness Hub
Blogs
Sign Up
Login
How It Works
The most simple & convenient way to access talk therapy - anytime, anywhere, any device.
Know Your Needs
Answer a few quick questions and get a therapist recommendation right away.
Choose The Right Plan
Decide on the number of sessions you would like to opt for.
Get Matched
Choose the recommended therapist or talk to a matching expert who will connect you with the right therapist based on your needs.


MindEase
How It Works
Plans&Pricing
Counselling
Wellness Hub
Blogs
Sign Up
Login
Welcome to MindEase
Your journey to mental wellness starts here.
Start Your Journey
Learn More
How We Support You
Mindfulness Tools
Access meditation guides, breathing exercises, and wellness resources.
Expert Guidance
Connect with certified mental health professionals for support.
Peaceful Community
Join a community of individuals sharing experiences and growth.


MindEase
How It Works
Plans&Pricing
Counselling
Wellness Hub
Blogs
Sign Up
Login
Flexible Pricing Plans
Choose the best plan for your journey.
Monthly
Yearly (Save 20%)
Starter
$9 / month
Basic Meditation
Community Access
Guided Sessions
Live Sessions
Get Started
Best Value
Professional
$29 / month
All Starter Features
Unlimited Meditation
Live Sessions
Priority Support
Choose Plan
Enterprise
$49 / month
All Pro Features
Personal Coaching
Exclusive Workshops
24/7 Support
Go Premium


EXPLORER
OPEN EDITORS
MINDEASE
.vscode
backend
config
controllers
activity.js
feedback.js
survey.js
userController.js
volunteerContr...
middlewares
models
node_modules
routes
templates
utils
.gitignore
app.js
package-lock.json
package.json
server.js
frontend
node_modules
public
src
assets
components
constants
pages
admin
user
volunteer
Auth.jsx
ForgotPassw...
VerifyOtp.jsx
redux
routes
admin
user
AuthRoute.jsx
ProtectedRo...
User.jsx
volunteer
App.jsx
index.css
vite.config.js
main.jsx
index.css
App.jsx
Auth.jsx
userController.js
User.jsx
backend > controllers > userController.js > registerUser
import { sendEMail } from "../middlewares/sendMail.js";
import User from "../models/user.js";
import { message } from "../utils/message.js";
import { Response } from "../utils/response.js";
import fs from "fs";
import path from "path";
import { fileURLToPath } from "url";
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
export const registerUser = async (req, res) => {
try {
const {
firstName,
middleName,
lastName,
email,
password,
dateOfBirth,
gender,
phoneNumber,
isVolunteer,
} = req.body;
//check body data
if (
!firstName ||
!lastName ||
!email ||
!password ||
!dateOfBirth ||
!gender ||
!phoneNumber
) {
return Response(res, 400, false, message.missingFieldMessage);
}
//checking user
let user = await User.findOne({ email });
if (user) {
return Response(res, 400, false, message.userAlreadyExist);
}
user = await User.create({ ...req.body });
//generating otp
const otp = Math.floor(100000 + Math.random() * 900000);
const otpExpire = new Date(Date.now() + 5 * 60 * 1000);
user.registerOtp = otp;
user.registerOtpExpire = otpExpire;
//save user


EXPLORER
vite.config.js
main.jsx
index.css 3
App.jsx
Auth.jsx 2
Dashboard.jsx 9+
User.jsx
frontend > src > pages > volunteer > Dashboard.jsx > ...
import React from "react"; 6.9k (gzipped: 2.7k)
import { useNavigate } from "react-router-dom"; 224.9k (gzipped: 71k)
import VolHeader from "../../components/VolHeader";
Codeium: Refactor | Explain | Generate JSDoc | ×
const Dashboard = () => {
const navigate = useNavigate();
Codeium: Refactor | Explain | Generate JSDoc | ×
const StatCard = ({ title, value, icon }) => (

{icon}

<h3 className="text-gray-500 text-sm">{title}</h3>
{value}

);
Codeium: Refactor | Explain | Generate JSDoc | ×
const ActivityItem = ({ message, time }) => (

{message}
{time}

);
Codeium: Refactor | Explain | Generate JSDoc | ×
const UpcomingSession = ({ name, topic, time }) => (

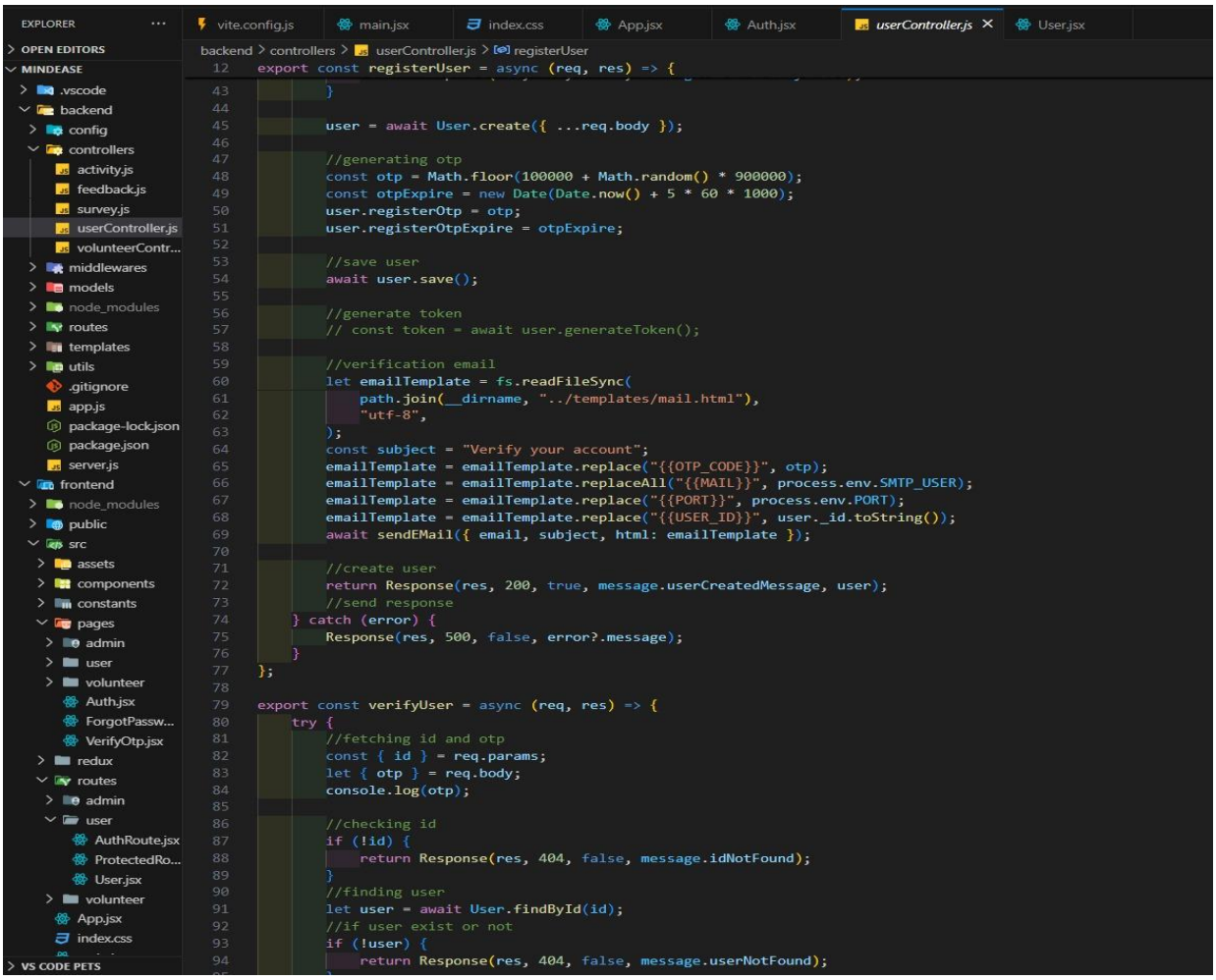
{name}
{topic}
{time}

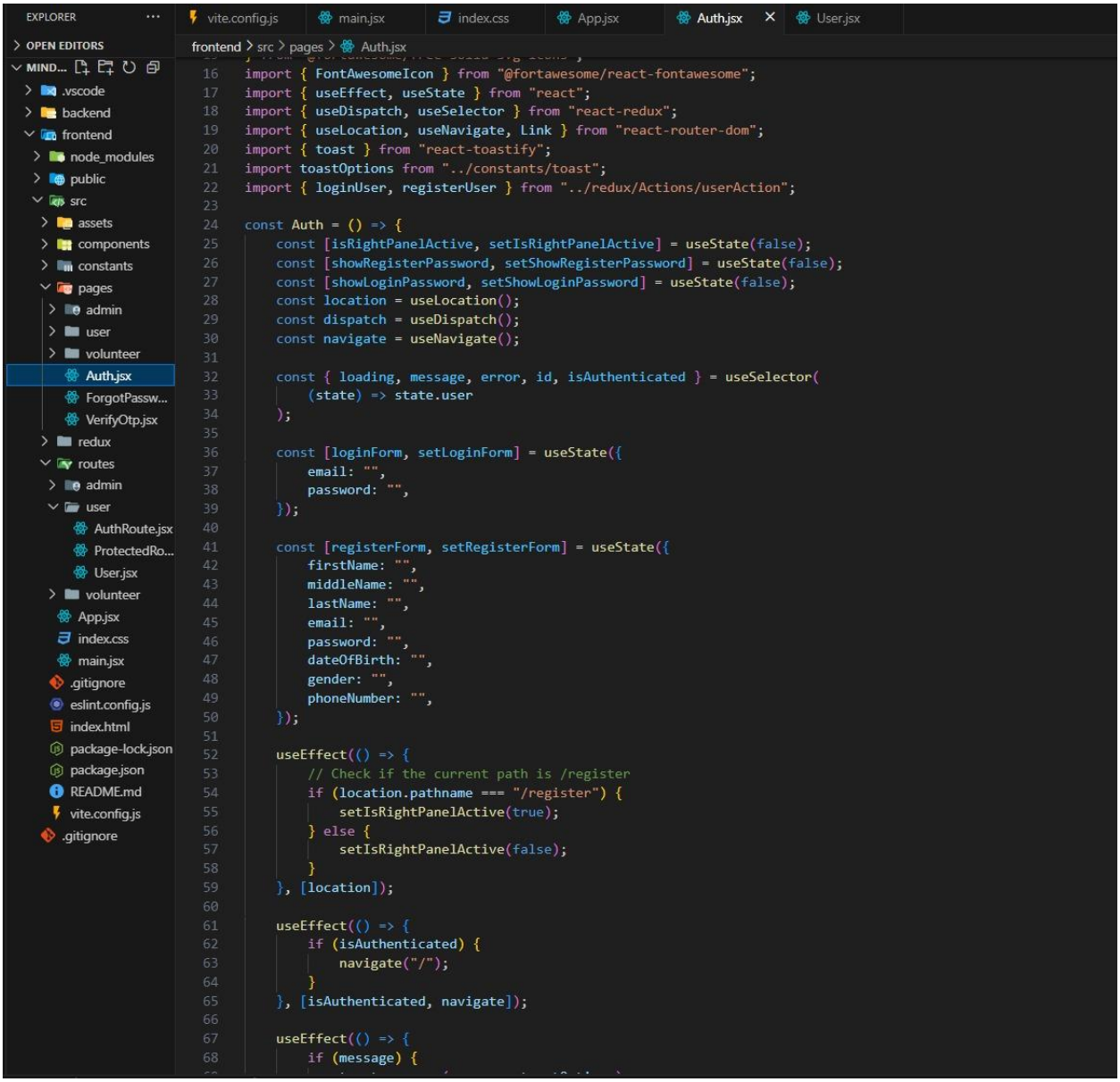
<button className="bg-orange-500 text-white px-4 py-2 rounded-md text-sm hover:bg-orange-600 transition">
Join
</button>

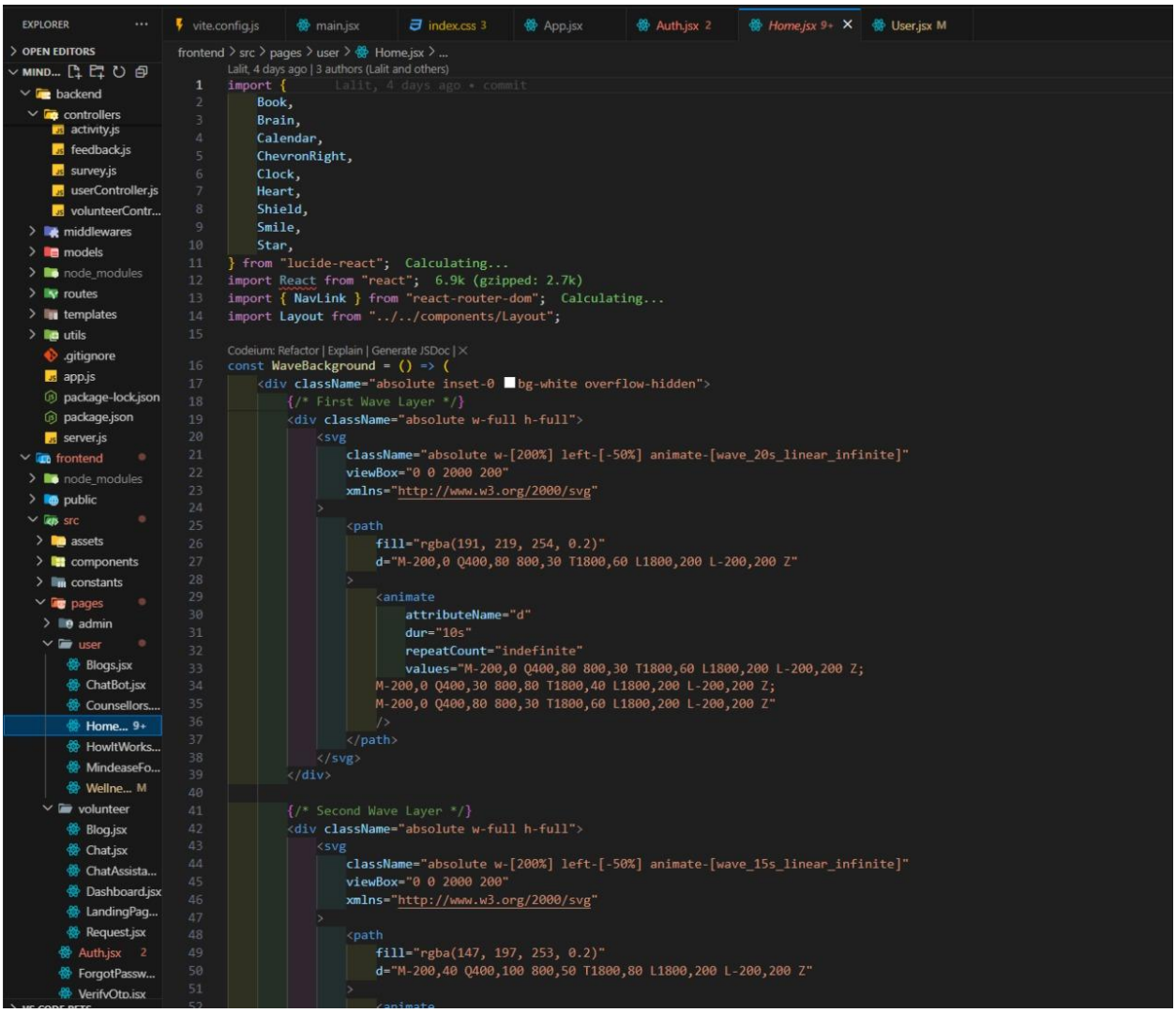
);
return (
<>
<VolHeader />

<h2 className="text-2xl font-semibold p-6 pt-8 text-gray-800">
Dashboard
</h2>
<main className="p-6 max-w-7xl mx-auto">

<StatCard
title="Pending Requests"
value="12"
icon={<Clock className="text-orange-500" />}
OPEN EDITORS
MIND...
backend
controllers
activity.js
feedback.js
survey.js
userController.js
volunteerContr...
middlewares
models
node_modules
routes
templates
utils
.gitignore
app.js
package-lock.json
package.json
server.js
frontend
node_modules
public
src
assets
components
constants
pages
admin
user
volunteer
Blog.jsx
Chat.jsx
ChatAssista...
Dash... 9+
LandingPag...
Request.jsx
Auth.jsx 2
ForgotPassw...
VerifyOtp.jsx
redux
routes
admin
user
AuthRoute.jsx
ProtectedRo...
User.jsx
VS CODE PETS


EXPLORER
OPEN EDITORS
MINDEASE
.vscode
backend
config
controllers
activity.js
feedback.js
survey.js
userController.js
volunteerContr...
middlewares
models
node_modules
routes
templates
utils
.gitignore
app.js
package-lock.json
package.json
server.js
frontend
node_modules
public
src
assets
components
constants
pages
admin
user
volunteer
Auth.jsx
ForgotPassw...
VerifyOtp.jsx
redux
routes
admin
user
AuthRoute.jsx
ProtectedRo...
User.jsx
volunteer
App.jsx
index.css
VS CODE PETS
vite.config.js
main.jsx
index.css
App.jsx
Auth.jsx
userController.js
User.jsx
backend > controllers > userController.js > registerUser
export const registerUser = async (req, res) => {
}
user = await User.create({ ...req.body });
//generating otp
const otp = Math.floor(100000 + Math.random() * 900000);
const otpExpire = new Date(Date.now() + 5 * 60 * 1000);
user.registerOtp = otp;
user.registerOtpExpire = otpExpire;
//save user
await user.save();
//generate token
// const token = await user.generateToken();
//verification email
let emailTemplate = fs.readFileSync(
path.join(__dirname, "../templates/mail.html"),
"utf-8",
);
const subject = "Verify your account";
emailTemplate = emailTemplate.replace("{{OTP_CODE}}", otp);
emailTemplate = emailTemplate.replaceAll("{{MAIL}}", process.env.SMTP_USER);
emailTemplate = emailTemplate.replace("{{PORT}}", process.env.PORT);
emailTemplate = emailTemplate.replace("{{USER_ID}}", user._id.toString());
await sendEMail({ email, subject, html: emailTemplate });
//create user
return Response(res, 200, true, message.userCreatedMessage, user);
//send response
} catch (error) {
Response(res, 500, false, error?.message);
}
};
export const verifyUser = async (req, res) => {
try {
//fetching id and otp
const { id } = req.params;
let { otp } = req.body;
console.log(otp);
//checking id
if (!id) {
return Response(res, 404, false, message.idNotFound);
}
//finding user
let user = await User.findById(id);
//if user exist or not
if (!user) {
return Response(res, 404, false, message.userNotFound);


EXPLORER
OPEN EDITORS
MIND...
.vscode
backend
frontend
node_modules
public
src
assets
components
constants
pages
admin
user
volunteer
Auth.jsx
ForgotPassw...
VerifyOtp.jsx
redux
routes
admin
user
AuthRoute.jsx
ProtectedRo...
User.jsx
volunteer
App.jsx
index.css
main.jsx
.gitignore
eslint.config.js
index.html
package-lock.json
package.json
README.md
vite.config.js
.gitignore
vite.config.js
main.jsx
index.css
App.jsx
Auth.jsx
User.jsx
frontend > src > pages > Auth.jsx
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { useLocation, useNavigate, Link } from "react-router-dom";
import { toast } from "react-toastify";
import toastOptions from "../constants/toast";
import { loginUser, registerUser } from "../redux/Actions/userAction";
const Auth = () => {
const [isRightPanelActive, setIsRightPanelActive] = useState(false);
const [showRegisterPassword, setShowRegisterPassword] = useState(false);
const [showLoginPassword, setShowLoginPassword] = useState(false);
const location = useLocation();
const dispatch = useDispatch();
const navigate = useNavigate();
const { loading, message, error, id, isAuthenticated } = useSelector(
(state) => state.user
);
const [loginForm, setLoginForm] = useState({
email: "",
password: "",
});
const [registerForm, setRegisterForm] = useState({
firstName: "",
middleName: "",
lastName: "",
email: "",
password: "",
dateOfBirth: "",
gender: "",
phoneNumber: "",
});
useEffect(() => {
// Check if the current path is /register
if (location.pathname === "/register") {
setIsRightPanelActive(true);
} else {
setIsRightPanelActive(false);
}
}, [location]);
useEffect(() => {
if (isAuthenticated) {
navigate("/");
}
}, [isAuthenticated, navigate]);
useEffect(() => {
if (message) {

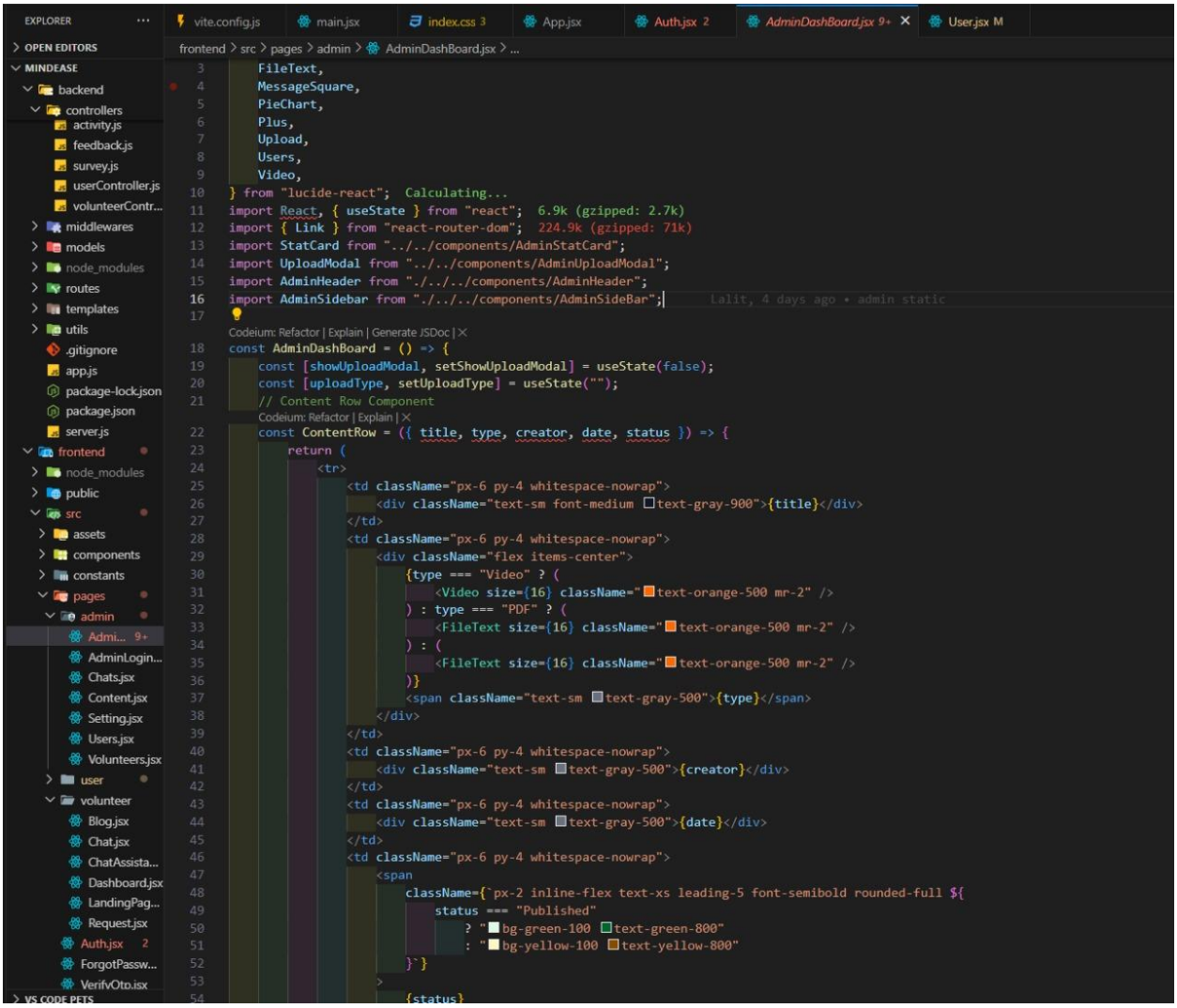

EXPLORER
OPEN EDITORS
MIND...
backend
controllers
activity.js
feedback.js
survey.js
userController.js
volunteerContr...
middlewares
models
node_modules
routes
templates
utils
.gitignore
app.js
package-lock.json
package.json
server.js
frontend
node_modules
public
src
assets
components
constants
pages
admin
user
Blogs.jsx
ChatBot.jsx
Counsellors....
Home... 9+
HowItWorks...
MindeaseFo...
Wellne... M
volunteer
Blog.jsx
Chat.jsx
ChatAssista...
Dashboard.jsx
LandingPag...
Request.jsx
Auth.jsx 2
ForgotPassw...
VerifyOto.jsx
vite.config.js
main.jsx
index.css 3
App.jsx
Auth.jsx 2
Home.jsx 9+
User.jsx M
frontend > src > pages > user > Home.jsx > ...
Lalit, 4 days ago | 3 authors (Lalit and others)
import {
Lalit, 4 days ago • commit
Book,
Brain,
Calendar,
ChevronRight,
Clock,
Heart,
Shield,
Smile,
Star,
} from "lucide-react"; Calculating...
import React from "react"; 6.9k (gzipped: 2.7k)
import { NavLink } from "react-router-dom"; Calculating...
import Layout from "../../components/Layout";
Codeium: Refactor | Explain | Generate JSDoc |
const WaveBackground = () => (

{/* First Wave Layer */}

<svg
className="absolute w-[200%] left-[-50%] animate-[wave_20s_linear_infinite]"
viewBox="0 0 2000 200"
xmlns="http://www.w3.org/2000/svg"
>
<path
fill="rgba(191, 219, 254, 0.2)"
d="M-200,0 Q400,80 800,30 T1800,60 L1800,200 L-200,200 Z"
>
<animate
attributeName="d"
dur="10s"
repeatCount="indefinite"
values="M-200,0 Q400,80 800,30 T1800,60 L1800,200 L-200,200 Z;
M-200,0 Q400,30 800,80 T1800,40 L1800,200 L-200,200 Z;
M-200,0 Q400,80 800,30 T1800,60 L1800,200 L-200,200 Z"
/>
</path>
</svg>

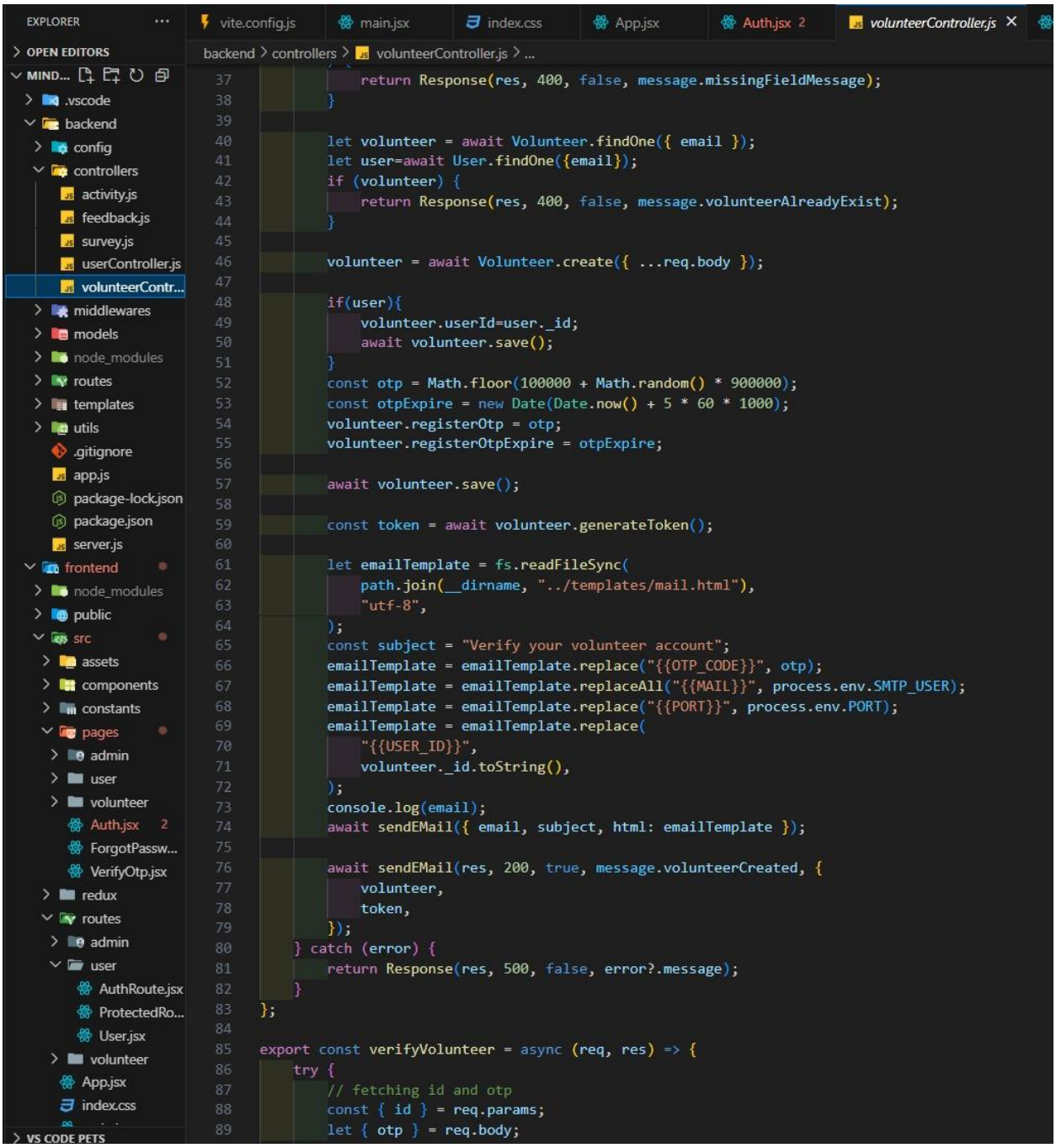
{/* Second Wave Layer */}

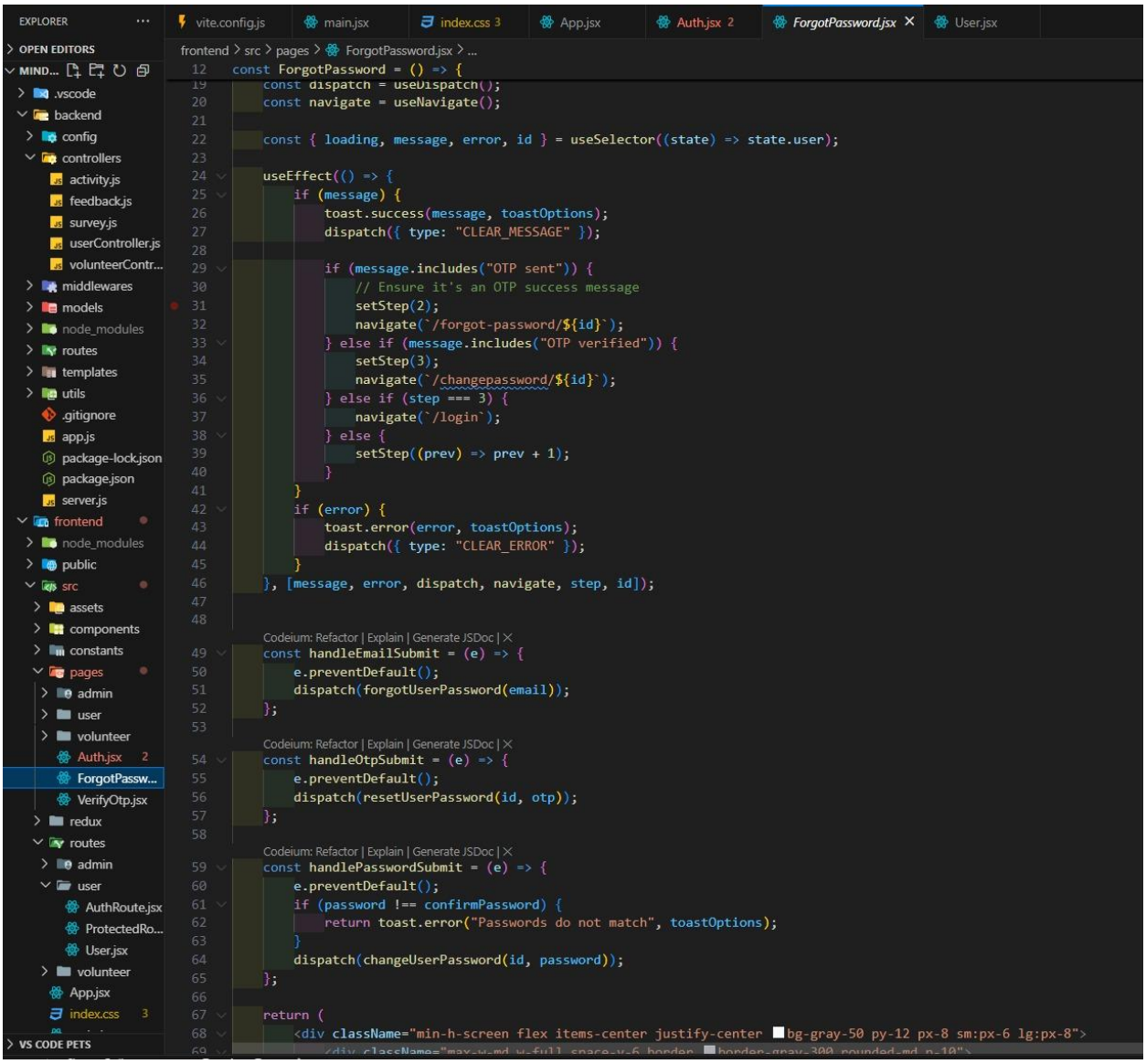
<svg
className="absolute w-[200%] left-[-50%] animate-[wave_15s_linear_infinite]"
viewBox="0 0 2000 200"
xmlns="http://www.w3.org/2000/svg"
>
<path
fill="rgba(147, 197, 253, 0.2)"
d="M-200,40 Q400,100 800,50 T1800,80 L1800,200 L-200,200 Z"
>


EXPLORER
OPEN EDITORS
MINDEASE
backend
controllers
activity.js
feedback.js
survey.js
userController.js
volunteerContr...
middlewares
models
node_modules
routes
templates
utils
.gitignore
app.js
package-lock.json
package.json
server.js
frontend
node_modules
public
src
assets
components
constants
pages
admin
Admi... 9+
AdminLogin...
Chats.jsx
Content.jsx
Setting.jsx
Users.jsx
Volunteers.jsx
user
volunteer
Blog.jsx
Chat.jsx
ChatAssista...
Dashboard.jsx
LandingPag...
Request.jsx
Auth.jsx 2
ForgotPassw...
VerifyOtp.jsx
vite.config.js
main.jsx
index.css 3
App.jsx
Auth.jsx 2
AdminDashBoard.jsx 9+
User.jsx M
frontend > src > pages > admin > AdminDashBoard.jsx > ...
FileText,
MessageSquare,
PieChart,
Plus,
Upload,
Users,
Video,
} from "lucide-react"; Calculating...
import React, { useState } from "react"; 6.9k (gzipped: 2.7k)
import { Link } from "react-router-dom"; 224.9k (gzipped: 71k)
import StatCard from "../../components/AdminStatCard";
import UploadModal from "../../components/AdminUploadModal";
import AdminHeader from "./../../components/AdminHeader";
import AdminSidebar from "./../../components/AdminSideBar";
Lalit, 4 days ago • admin static
Codeium: Refactor | Explain | Generate JSDoc |
const AdminDashBoard = () => {
const [showUploadModal, setShowUploadModal] = useState(false);
const [uploadType, setUploadType] = useState("");
// Content Row Component
Codeium: Refactor | Explain |
const ContentRow = ({ title, type, creator, date, status }) => {
return (
<tr>
<td className="px-6 py-4 whitespace-nowrap">
{title}
</td>
<td className="px-6 py-4 whitespace-nowrap">

{type === "Video" ? (
<Video size={16} className="text-orange-500 mr-2" />
) : type === "PDF" ? (
<FileText size={16} className="text-orange-500 mr-2" />
) : (
<FileText size={16} className="text-orange-500 mr-2" />
)}
{type}

</td>
<td className="px-6 py-4 whitespace-nowrap">
{creator}
</td>
<td className="px-6 py-4 whitespace-nowrap">
{date}
</td>
<td className="px-6 py-4 whitespace-nowrap">
<span
className={`px-2 inline-flex text-xs leading-5 font-semibold rounded-full ${
status === "Published"
? "bg-green-100 text-green-800"
: "bg-yellow-100 text-yellow-800"
}`}
>
{status}


EXPLORER
OPEN EDITORS
MIND...
.vscode
backend
config
controllers
activity.js
feedback.js
survey.js
userController.js
volunteerContr...
middlewares
models
node_modules
routes
templates
utils
.gitignore
app.js
package-lock.json
package.json
server.js
frontend
node_modules
public
src
assets
components
constants
pages
admin
user
volunteer
Auth.jsx 2
ForgotPassw...
VerifyOtp.jsx
redux
routes
admin
user
AuthRoute.jsx
ProtectedRo...
User.jsx
volunteer
App.jsx
index.css
VS CODE PETS
vite.config.js
main.jsx
index.css
App.jsx
Auth.jsx 2
volunteerController.js
backend > controllers > volunteerController.js > ...
return Response(res, 400, false, message.missingFieldMessage);
}
let volunteer = await Volunteer.findOne({ email });
let user=await User.findOne({email});
if (volunteer) {
return Response(res, 400, false, message.volunteerAlreadyExist);
}
volunteer = await Volunteer.create({ ...req.body });
if(user){
volunteer.userId=user._id;
await volunteer.save();
}
const otp = Math.floor(100000 + Math.random() * 900000);
const otpExpire = new Date(Date.now() + 5 * 60 * 1000);
volunteer.registerOtp = otp;
volunteer.registerOtpExpire = otpExpire;
await volunteer.save();
const token = await volunteer.generateToken();
let emailTemplate = fs.readFileSync(
path.join(__dirname, "../templates/mail.html"),
"utf-8",
);
const subject = "Verify your volunteer account";
emailTemplate = emailTemplate.replace("{{OTP_CODE}}", otp);
emailTemplate = emailTemplate.replaceAll("{{MAIL}}", process.env.SMTP_USER);
emailTemplate = emailTemplate.replace("{{PORT}}", process.env.PORT);
emailTemplate = emailTemplate.replace(
"{{USER_ID}}",
volunteer._id.toString(),
);
console.log(email);
await sendEMail({ email, subject, html: emailTemplate });
await sendEMail(res, 200, true, message.volunteerCreated, {
volunteer,
token,
});
} catch (error) {
return Response(res, 500, false, error?.message);
}
};
export const verifyVolunteer = async (req, res) => {
try {
// fetching id and otp
const { id } = req.params;
let { otp } = req.body;


EXPLORER
vite.config.js
main.jsx
index.css 3
App.jsx
Auth.jsx 2
ForgotPassword.jsx
User.jsx
frontend > src > pages > ForgotPassword.jsx > ...
OPEN EDITORS
MIND...
.vscode
backend
config
controllers
activity.js
feedback.js
survey.js
userController.js
volunteerContr...
middlewares
models
node_modules
routes
templates
utils
.gitignore
app.js
package-lock.json
package.json
server.js
frontend
node_modules
public
src
assets
components
constants
pages
admin
user
volunteer
Auth.jsx 2
ForgotPassw...
VerifyOtp.jsx
redux
routes
admin
user
AuthRoute.jsx
ProtectedRo...
User.jsx
volunteer
App.jsx
index.css 3
VS CODE PETS
const ForgotPassword = () => {
const dispatch = useDispatch();
const navigate = useNavigate();
const { loading, message, error, id } = useSelector((state) => state.user);
useEffect(() => {
if (message) {
toast.success(message, toastOptions);
dispatch({ type: "CLEAR_MESSAGE" });
if (message.includes("OTP sent")) {
// Ensure it's an OTP success message
setStep(2);
navigate(`/forgot-password/${id}`);
} else if (message.includes("OTP verified")) {
setStep(3);
navigate(`/changepassword/${id}`);
} else if (step === 3) {
navigate(`/login`);
} else {
setStep((prev) => prev + 1);
}
}
if (error) {
toast.error(error, toastOptions);
dispatch({ type: "CLEAR_ERROR" });
}
}, [message, error, dispatch, navigate, step, id]);
Codeium: Refactor | Explain | Generate JSDoc |
const handleEmailSubmit = (e) => {
e.preventDefault();
dispatch(forgotUserPassword(email));
};
Codeium: Refactor | Explain | Generate JSDoc |
const handleOtpSubmit = (e) => {
e.preventDefault();
dispatch(resetUserPassword(id, otp));
};
Codeium: Refactor | Explain | Generate JSDoc |
const handlePasswordSubmit = (e) => {
e.preventDefault();
if (password !== confirmPassword) {
return toast.error("Passwords do not match", toastOptions);
}
dispatch(changeUserPassword(id, password));
};
return (

Auth.jsx 2
index.css 3

References:

Official Documentation: Documentation for libraries, frameworks, and tools used in the project, as well as APIs or services integrated.

Tutorials and Guides: Online tutorials, guides, blog posts, and educational videos that provided assistance or insights during development.

Code Repositories: GitHub repositories or other code repositories where code snippets, examples, or inspiration were found**.**

Forums and Communities: Online forums, such as Stack Overflow or Reddit, and developer communities where questions were asked, advice was sought, or discussions were participated in**.**

Personal Communication: Mentors, peers who provided guidance, feedback, or support during development.