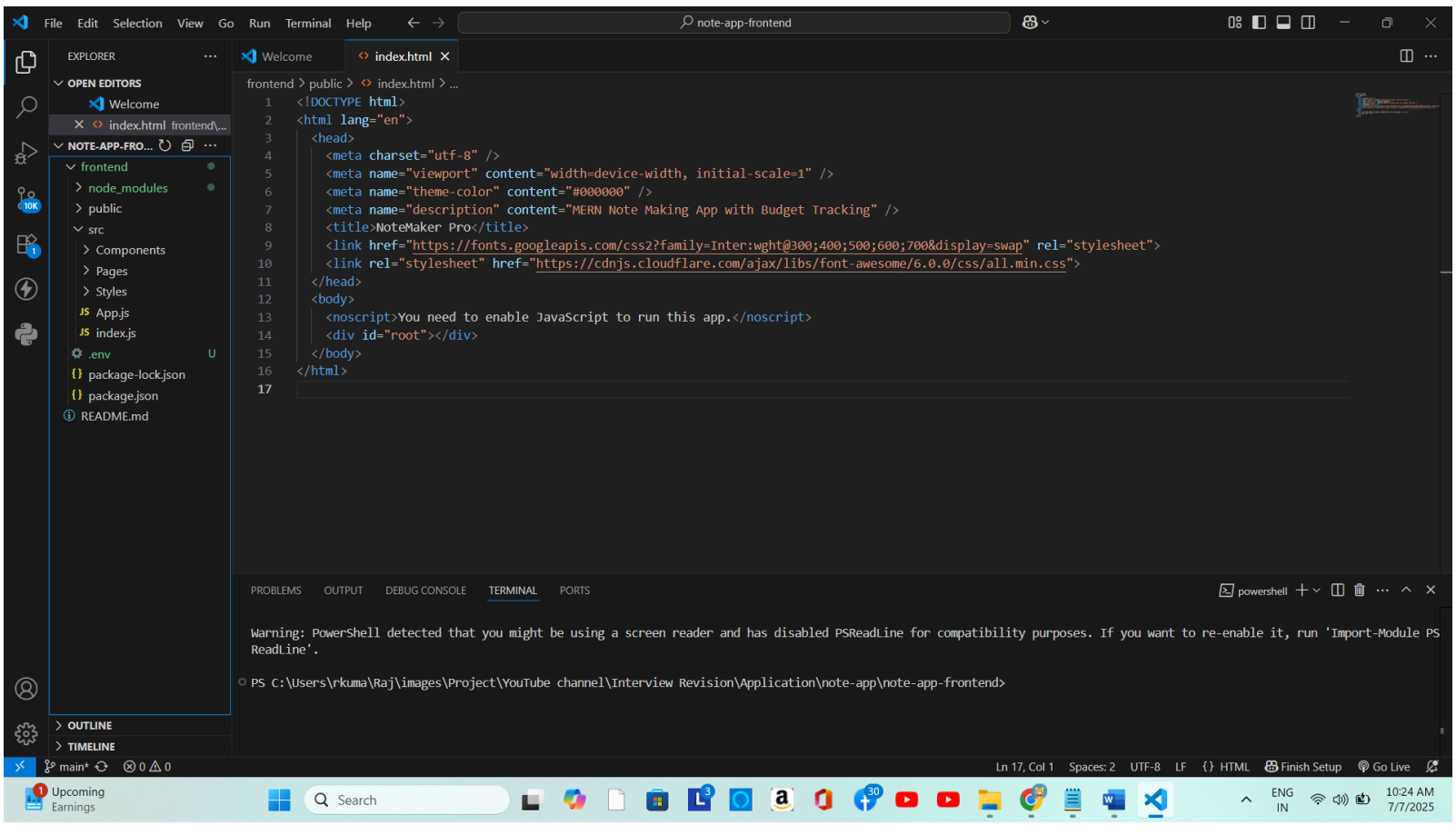

In Public Folder

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="MERN Note Making App with Budget Tracking" />
    <title>NoteMaker Pro</title>
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

In SRC Folder

In Component Folder

AllRouter.jsx

```
import { BrowserRouter as Router, Routes, Route, Navigate } from "react-router-dom"
import { useAppContext } from "./ContextAPI"
import Header from "./Header"
import NavBar from "./NavBar"
import Footer from "./Footer"
import Home from "../Pages/Home"
import About from "../Pages/About"
import Service from "../Pages/Service"
```

```
import Contact from "../Pages/Contact"
import SingleUser from "../Pages/SingleUser"
import Loading from "./Loading"
import Error from "./Error"
import GoToTop from "./GoToTop"

const AllRouters = () => {
  const { loading, error } = useAppContext()

  if (loading) return <Loading />
  if (error) return <Error message={error} />

  return (
    <Router>
      <Header />
      <NavBar />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/service" element={<Service />} />
        <Route path="/contact" element={<Contact />} />
        <Route path="/profile" element={<SingleUser />} />
        <Route path="*" element={<Navigate to="/" replace />} />
      </Routes>
      <Footer />
      <GoToTop />
    </Router>
  )
}

export default AllRouters
```

ContextAPI.js

```
"use client"

import { createContext, useContext, useReducer, useEffect } from "react"

const AppContext = createContext()

const initialState = {
  user: null,
  token: localStorage.getItem("token") || null,
  notes: [],
  loading: false,
  error: null,
  budgetSummary: {
```

```
    totalIncome: 0,
    totalExpenses: 0,
    balance: 0,
    currency: "₹",
  },
}

const appReducer = (state, action) => {
  switch (action.type) {
    case "SET_LOADING":
      return { ...state, loading: action.payload }
    case "SET_ERROR":
      return { ...state, error: action.payload, loading: false }
    case "SET_USER":
      return { ...state, user: action.payload, loading: false }
    case "SET_TOKEN":
      return { ...state, token: action.payload }
    case "SET_NOTES":
      return { ...state, notes: action.payload, loading: false }
    case "ADD_NOTE":
      return { ...state, notes: [action.payload, ...state.notes] }
    case "UPDATE_NOTE":
      return {
        ...state,
        notes: state.notes.map((note) => (note._id === action.payload._id ? action.payload : note)),
      }
    case "DELETE_NOTE":
      return {
        ...state,
        notes: state.notes.filter((note) => note._id !== action.payload),
      }
    case "SET_BUDGET_SUMMARY":
      return { ...state, budgetSummary: action.payload }
    case "LOGOUT":
      return { ...initialState, token: null }
    default:
      return state
  }
}

export const AppProvider = ({ children }) => {
  const [state, dispatch] = useReducer(appReducer, initialState)

  useEffect(() => {
    if (state.token) {
      localStorage.setItem("token", state.token)
```

```
    } else {
      localStorage.removeItem("token")
    }
  }, [state.token])

  const value = {
    ...state,
    dispatch,
  }

  return <AppContext.Provider value={value}>{children}</AppContext.Provider>
}

export const useAppContext = () => {
  const context = useContext(AppContext)
  if (!context) {
    throw new Error("useAppContext must be used within AppProvider")
  }
  return context
}
```

Error.js

```
"use client"
import styled from "styled-components"
import { Button } from "../Styles/button"

const ErrorContainer = styled.div`
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  text-align: center;
  padding: 2rem;
`

const ErrorIcon = styled.div`
  font-size: 6rem;
  margin-bottom: 2rem;
`

const ErrorTitle = styled.h1`
  font-size: 3rem;
  margin-bottom: 1rem;
`
```

```
const ErrorMessage = styled.p`
  font-size: 1.6rem;
  margin-bottom: 3rem;
  opacity: 0.9;
`

const Error = ({ message = "Something went wrong" }) => {
  const handleRetry = () => {
    window.location.reload()
  }

  return (
    <ErrorContainer>
      <ErrorIcon> ⚠️ </ErrorIcon>
      <ErrorTitle>Oops!</ErrorTitle>
      <ErrorMessage>{message}</ErrorMessage>
      <Button onClick={handleRetry}>Try Again</Button>
    </ErrorContainer>
  )
}

export default Error
```

Footer.js

```
import styled from "styled-components"

const FooterContainer = styled.footer`
  background: #2c3e50;
  color: white;
  padding: 4rem 0 2rem;
  margin-top: 6rem;
`

const FooterContent = styled.div`
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 2rem;
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 3rem;
`

const FooterSection = styled.div`
  h3 {
    font-size: 2rem;
```

```
  margin-bottom: 1.5rem;
  color: #667eea;
}

p, li {
  font-size: 1.4rem;
  line-height: 1.6;
  margin-bottom: 1rem;
}

ul {
  list-style: none;
}

a {
    color: #bdc3c7;
    transition: color 0.3s ease;

    &:hover {
      color: #667eea;
    }
  }
`

const FooterBottom = styled.div`
  border-top: 1px solid #34495e;
  margin-top: 3rem;
  padding-top: 2rem;
  text-align: center;
  color: #bdc3c7;
`

const Footer = () => {
  return (
    <FooterContainer>
      <FooterContent>
        <FooterSection>
          <h3>NoteMaker Pro</h3>
          <p>
            Your ultimate digital note-taking companion with budget tracking capabilities. Organize your thoughts and
            finances in one place.
          </p>
        </FooterSection>

        <FooterSection>
          <h3>Features</h3>
```

```
  <ul>
    <li>
      <a href="#notes">Smart Notes</a>
    </li>
    <li>
      <a href="#budget">Budget Tracking</a>
    </li>
    <li>
      <a href="#security">Secure Storage</a>
    </li>
    <li>
      <a href="#mobile">Mobile Access</a>
    </li>
  </ul>
</FooterSection>

<FooterSection>
  <h3>Support</h3>
  <ul>
    <li>
      <a href="#help">Help Center</a>
    </li>
    <li>
      <a href="#contact">Contact Us</a>
    </li>
    <li>
      <a href="#privacy">Privacy Policy</a>
    </li>
    <li>
      <a href="#terms">Terms of Service</a>
    </li>
  </ul>
</FooterSection>

<FooterSection>
  <h3>Connect</h3>
  <ul>
    <li>
      <a href="#twitter">Twitter</a>
    </li>
    <li>
      <a href="#facebook">Facebook</a>
    </li>
    <li>
      <a href="#linkedin">LinkedIn</a>
    </li>
```

```
          <li>
            <a href="#github">GitHub</a>
          </li>
        </ul>
      </FooterSection>
    </FooterContent>

    <FooterBottom>
      <p>&copy; 2024 NoteMaker Pro. All rights reserved. Made with ❤️ in India</p>
    </FooterBottom>
  </FooterContainer>
 )
}

export default Footer
```

GlobalStyle.js

```
import styled, { createGlobalStyle } from "styled-components"

export const GlobalStyle = createGlobalStyle`
 * {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
 }

 body {
  font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', sans-serif;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  min-height: 100vh;
  color: #333;
 }

html {
 font-size: 62.5%;
}

a {
 text-decoration: none;
 color: inherit;
}

button {
 border: none;
 cursor: pointer;
 outline: none;
}
```

```
input, textarea {
  border: none;
  outline: none;
}

::-webkit-scrollbar {
  width: 8px;
}

::-webkit-scrollbar-track {
  background: #f1f1f1;
}

::-webkit-scrollbar-thumb {
    background: #888;
    border-radius: 4px;
  }

  ::-webkit-scrollbar-thumb:hover {
    background: #555;
  }
`

export const Container = styled.div`
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 2rem;

  @media (max-width: 768px) {
    padding: 0 1rem;
  }
`

export const Grid = styled.div`
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 2rem;
  margin: 2rem 0;

  @media (max-width: 768px) {
    grid-template-columns: 1fr;
    gap: 1rem;
  }
`
```

```
export const Card = styled.div`
  background: white;
  border-radius: 12px;
  padding: 2rem;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  transition: transform 0.3s ease, box-shadow 0.3s ease;

  &:hover {
    transform: translateY(-5px);
    box-shadow: 0 8px 25px rgba(0, 0, 0, 0.15);
  }

  @media (max-width: 768px) {
    padding: 1.5rem;
  }
`

export const Flex = styled.div`
  display: flex;
  justify-content: ${(props) => props.justify || "flex-start"};
  align-items: ${(props) => props.align || "stretch"};
  gap: ${(props) => props.gap || "0"};
  flex-direction: ${(props) => props.direction || "row"};

  @media (max-width: 768px) {
    flex-direction: ${(props) => props.mobileDirection || props.direction || "column"};
  }
`
```

GoToTop.js

```
"use client"

import { useState, useEffect } from "react"
import styled from "styled-components"

const GoToTopButton = styled.button`
  position: fixed;
  bottom: 2rem;
  right: 2rem;
  background: #667eea;
  color: white;
  border: none;
  border-radius: 50%;
  width: 50px;
  height: 50px;
  font-size: 2rem;
```

```
cursor: pointer;
box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
transition: all 0.3s ease;
z-index: 1000;
display: ${(props) => (props.visible ? "flex" : "none")};
align-items: center;
justify-content: center;

&:hover {
  background: #5a67d8;
  transform: translateY(-2px);
  box-shadow: 0 6px 16px rgba(0, 0, 0, 0.4);
}

&:active {
  transform: translateY(0);
}
`

const GoToTop = () => {
  const [visible, setVisible] = useState(false)

  useEffect(() => {
    const handleScroll = () => {
      setVisible(window.scrollY > 300)
    }

    window.addEventListener("scroll", handleScroll)
    return () => window.removeEventListener("scroll", handleScroll)
  }, [])

  const scrollToTop = () => {
    window.scrollTo({
      top: 0,
      behavior: "smooth",
    })
  }

  return (
    <GoToTopButton visible={visible} onClick={scrollToTop}>
      <i className="fas fa-arrow-up"></i>
    </GoToTopButton>
  )
}

export default GoToTop
```

Header.js

```
"use client"


import { useState } from "react"
import styled from "styled-components"
import { useAppContext } from "./ContextAPI"
import { Button } from "../Styles/button"


const HeaderContainer = styled.header`
  background: rgba(255, 255, 255, 0.95);
  backdrop-filter: blur(10px);
  padding: 1rem 0;
  position: sticky;
  top: 0;
  z-index: 1000;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
`


const HeaderContent = styled.div`
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 2rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
`


const Logo = styled.h1`
  font-size: 2.4rem;
  font-weight: 700;
  color: #667eea;
  cursor: pointer;
`


const UserSection = styled.div`
  display: flex;
  align-items: center;
  gap: 1rem;
`


const AuthForm = styled.div`
  display: flex;
  gap: 1rem;
  align-items: center;
`
```

```
const Input = styled.input`
  padding: 0.8rem 1.2rem;
  border: 1px solid #ddd;
  border-radius: 6px;
  font-size: 1.4rem;
  width: 200px;

  @media (max-width: 768px) {
    width: 150px;
  }
`

const Header = () => {
  const { user, dispatch } = useAppContext()
  const [showAuth, setShowAuth] = useState(false)
  const [isLogin, setIsLogin] = useState(true)
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
  })

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

  const handleSubmit = async (e) => {
    e.preventDefault()
    dispatch({ type: "SET_LOADING", payload: true })

    try {
      const url = isLogin ? "/api/auth/login" : "/api/auth/register"
      const response = await fetch(`http://localhost:5000${url}`, {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(formData),
      })

      const data = await response.json()

      if (response.ok) {
```

```
      dispatch({ type: "SET_TOKEN", payload: data.token })
      dispatch({ type: "SET_USER", payload: data })
      setShowAuth(false)
      setFormData({ name: "", email: "", password: "" })
    } else {
      dispatch({ type: "SET_ERROR", payload: data.message })
    }
  } catch (error) {
    dispatch({ type: "SET_ERROR", payload: "Something went wrong" })
  }
}

const handleLogout = () => {
  dispatch({ type: "LOGOUT" })
}

return (
  <HeaderContainer>
    <HeaderContent>
      <Logo>NoteMaker Pro</Logo>

      <UserSection>
        {user ? (
          <>
            <span>Welcome, {user.name}!</span>
            <Button onClick={handleLogout}>Logout</Button>
          </>
        ) : (
          <>
            {!showAuth ? (
              <Button onClick={() => setShowAuth(true)}>Login</Button>
) : (
  <AuthForm>
    <form onSubmit={handleSubmit} style={{ display: "flex", gap: "1rem", alignItems: "center" }}>
      {!isLogin && (
        <Input
          type="text"
          name="name"
          placeholder="Name"
          value={formData.name}
          onChange={handleInputChange}
          required
        />
      )}
      <Input
        type="email"
```

```
          name="email"
          placeholder="Email"
          value={formData.email}
          onChange={handleInputChange}
          required
        />
        <Input
          type="password"
          name="password"
          placeholder="Password"
          value={formData.password}
          onChange={handleInputChange}
          required
        />
        <Button type="submit">{isLogin ? "Login" : "Register"}</Button>
        <Button
          type="button"
          onClick={() => setIsLogin(!isLogin)}
          style={{ background: "transparent", color: "#667eea" }}
        >
          {isLogin ? "Register" : "Login"}
        </Button>
        <Button type="button" onClick={() => setShowAuth(false)} style={{ background: "#ff4757" }}>
          Cancel
        </Button>
      </form>
    </AuthForm>
  )}
 </>
)}
</UserSection>
    </HeaderContent>
  </HeaderContainer>
 )
}

export default Header
```

HeroSecton.js

```
import styled from "styled-components"
import { Button } from "../Styles/button"

const HeroContainer = styled.section`
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  padding: 8rem 0;
```

```
  text-align: center;
  position: relative;
  overflow: hidden;
`

const HeroContent = styled.div`
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 2rem;
  position: relative;
  z-index: 2;
`

const HeroTitle = styled.h1`
  font-size: 4.8rem;
  font-weight: 700;
  margin-bottom: 2rem;
  line-height: 1.2;

  @media (max-width: 768px) {
    font-size: 3.2rem;
  }
`

const HeroSubtitle = styled.p`
  font-size: 2rem;
  margin-bottom: 3rem;
  opacity: 0.9;

  @media (max-width: 768px) {
    font-size: 1.6rem;
  }
`

const HeroButtons = styled.div`
  display: flex;
  gap: 2rem;
  justify-content: center;
  align-items: center;

  @media (max-width: 768px) {
    flex-direction: column;
    gap: 1rem;
  }
`
```

```
const FeatureGrid = styled.div`
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 2rem;
  margin-top: 6rem;
`

const FeatureCard = styled.div`
  background: rgba(255, 255, 255, 0.1);
  backdrop-filter: blur(10px);
  padding: 3rem 2rem;
  border-radius: 12px;
  text-align: center;
  border: 1px solid rgba(255, 255, 255, 0.2);
`

const FeatureIcon = styled.div`
  font-size: 4rem;
  margin-bottom: 1rem;
`

const FeatureTitle = styled.h3`
  font-size: 2rem;
  margin-bottom: 1rem;
`

const FeatureDescription = styled.p`
  font-size: 1.4rem;
  opacity: 0.8;
`

const HeroSection = () => {
  const features = [
    {
      icon: " 📝 ",
      title: "Smart Notes",
      description: "Create, organize, and manage your notes with intelligent categorization",
    },
    {
      icon: " 💰 ",
      title: "Budget Tracking",
      description: "Track your expenses and income with financial note-taking",
    },
    {
      icon: " 🔒 ",
      title: "Secure & Private",
```

```
    description: "Your notes are protected with JWT authentication and encryption",
  },
  {
    icon: " 📱 ",
    title: "Mobile Friendly",
    description: "Access your notes anywhere with responsive design",
  },
]

return (
  <HeroContainer>
    <HeroContent>
      <HeroTitle>Your Digital Note-Taking Companion</HeroTitle>
      <HeroSubtitle>Organize your thoughts, track your finances, and boost your productivity</HeroSubtitle>
      <HeroButtons>
        <Button size="large">Get Started Free</Button>
        <Button variant="outline" size="large">
          Learn More
        </Button>
      </HeroButtons>

      <FeatureGrid>
        {features.map((feature, index) => (
          <FeatureCard key={index}>
            <FeatureIcon>{feature.icon}</FeatureIcon>
            <FeatureTitle>{feature.title}</FeatureTitle>
            <FeatureDescription>{feature.description}</FeatureDescription>
          </FeatureCard>
        ))}
      </FeatureGrid>
    </HeroContent>
  </HeroContainer>
  )
}

export default HeroSection
```

Loading.js

```
import styled, { keyframes } from "styled-components"

const spin = keyframes`
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
`

const LoadingContainer = styled.div`
```

```
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
`

const LoadingSpinner = styled.div`
  border: 4px solid rgba(255, 255, 255, 0.3);
  border-top: 4px solid white;
  border-radius: 50%;
  width: 50px;
  height: 50px;
  animation: ${spin} 1s linear infinite;
`

const LoadingText = styled.p`
  color: white;
  font-size: 1.8rem;
  margin-left: 2rem;
`

const Loading = () => {
  return (
    <LoadingContainer>
      <LoadingSpinner />
      <LoadingText>Loading...</LoadingText>
    </LoadingContainer>
  )
}

export default Loading
```

NavBar.js

```
"use client"

import { useState } from "react"
import styled from "styled-components"
import { Link, useLocation } from "react-router-dom"

const Nav = styled.nav`
  background: rgba(255, 255, 255, 0.9);
  backdrop-filter: blur(10px);
  padding: 1rem 0;
  position: sticky;
  top: 70px;
```

```
  z-index: 999;
`

const NavContent = styled.div`
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 2rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
`

const NavLinks = styled.ul`
  display: flex;
  list-style: none;
  gap: 2rem;

  @media (max-width: 768px) {
    flex-direction: column;
    position: absolute;
    top: 100%;
    left: 0;
    right: 0;
    background: white;
    padding: 2rem;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    transform: ${(props) => (props.isOpen ? "translateX(0)" : "translateX(-100%)")};
    transition: transform 0.3s ease;
  }
`

const NavLink = styled(Link)`
  font-size: 1.6rem;
  font-weight: 500;
  color: ${(props) => (props.active ? "#667eea" : "#333")};
  transition: color 0.3s ease;
  border-bottom: ${(props) => (props.active ? "2px solid #667eea" : "none")};
  padding-bottom: 0.5rem;

  &:hover {
    color: #667eea;
  }
`

const MenuToggle = styled.button`
  display: none;
```

```
  background: none;
  font-size: 2rem;
  color: #333;

  @media (max-width: 768px) {
    display: block;
  }
`

const NavBar = () => {
  const location = useLocation()
  const [isOpen, setIsOpen] = useState(false)

  const navItems = [
    { path: "/", label: "Home" },
    { path: "/about", label: "About" },
    { path: "/service", label: "Services" },
    { path: "/contact", label: "Contact" },
    { path: "/profile", label: "Profile" },
  ]

  return (
    <Nav>
      <NavContent>
        <NavLinks isOpen={isOpen}>
          {navItems.map((item) => (
            <li key={item.path}>
              <NavLink to={item.path} active={location.pathname === item.path} onClick={() => setIsOpen(false)}>
                {item.label}
              </NavLink>
            </li>
          ))}
        </NavLinks>
        <MenuToggle onClick={() => setIsOpen(!isOpen)}>
          <i className={`fas fa-${isOpen ? "times" : "bars"}`}></i>
        </MenuToggle>
      </NavContent>
    </Nav>
  )
}

export default NavBar
```

Now in Pages Folder

About.js

```
import styled from "styled-components"
```

```
import { Container, Card, Grid } from "../Components/GlobalStyle"

const AboutContainer = styled.div`
  min-height: 100vh;
  padding: 4rem 0;
`

const AboutHero = styled.section`
  text-align: center;
  margin-bottom: 6rem;
`

const AboutTitle = styled.h1`
  font-size: 4rem;
  color: white;
  margin-bottom: 2rem;
  text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
`

const AboutSubtitle = styled.p`
  font-size: 1.8rem;
  color: rgba(255,255,255,0.9);
  max-width: 600px;
  margin: 0 auto;
`

const FeatureCard = styled(Card)`
  text-align: center;
  height: 100%;

  h3 {
    color: #667eea;
    margin-bottom: 1rem;
  }

  .icon {
    font-size: 4rem;
    margin-bottom: 1rem;
  }
`

const TeamSection = styled.section`
  background: rgba(255,255,255,0.1);
  padding: 6rem 0;
  margin: 6rem 0;
  border-radius: 20px;
```

```
`

const TeamMember = styled(Card)`
  text-align: center;

  img {
    width: 150px;
    height: 150px;
    border-radius: 50%;
    object-fit: cover;
    margin-bottom: 1rem;
  }

  h3 {
    color: #2c3e50;
    margin-bottom: 0.5rem;
  }

  p {
    color: #7f8c8d;
    font-style: italic;
  }
`

const About = () => {
  const features = [
    {
      icon: " 🚀 ",
      title: "Fast & Efficient",
      description: "Lightning-fast performance with modern React architecture and optimized backend APIs.",
    },
    {
  icon: " 🔐 ",
  title: "Secure by Design",
  description: "Enterprise-grade security with JWT authentication and encrypted data storage.",
},
{
  icon: " 💡 ",
  title: "Smart Organization",
  description: "Intelligent categorization and tagging system to keep your notes organized.",
},
{
  icon: " 💰 ",
  title: "Financial Integration",
  description: "Built-in expense tracking and budget management with Indian Rupee support.",
},
```

```
  {
    icon: " 📱 ",
    title: "Cross-Platform",
    description: "Responsive design that works seamlessly across all devices and screen sizes.",
  },
  {
    icon: " 🎨 ",
    title: "Customizable",
    description: "Personalize your experience with custom colors, themes, and layout options.",
  },
]

const teamMembers = [
  {
    name: "Priya Sharma",
    role: "Full Stack Developer",
    image: "/placeholder.svg?height=150&width=150",
    bio: "Expert in MERN stack development with 5+ years of experience.",
  },
  {
    name: "Rahul Kumar",
    role: "UI/UX Designer",
    image: "/placeholder.svg?height=150&width=150",
    bio: "Creative designer passionate about creating intuitive user experiences.",
  },
  {
    name: "Anita Patel",
    role: "Backend Engineer",
    image: "/placeholder.svg?height=150&width=150",
    bio: "Database optimization expert with focus on scalable Node.js applications.",
  },
]

return (
  <AboutContainer>
    <Container>
      <AboutHero>
        <AboutTitle>About NoteMaker Pro</AboutTitle>
        <AboutSubtitle>
          We're building the future of digital note-taking, combining powerful organization tools with intelligent
          financial tracking to help you stay productive and organized.
        </AboutSubtitle>
      </AboutHero>

      <section>
        <h2 style={{ textAlign: "center", fontSize: "3rem", color: "white", marginBottom: "3rem" }}>
```

```
    Why Choose NoteMaker Pro?
  </h2>
  <Grid>
    {features.map((feature, index) => (
      <FeatureCard key={index}>
        <div className="icon">{feature.icon}</div>
        <h3>{feature.title}</h3>
        <p>{feature.description}</p>
      </FeatureCard>
    ))}
  </Grid>
</section>

<TeamSection>
  <Container>
    <h2 style={{ textAlign: "center", fontSize: "3rem", color: "white", marginBottom: "3rem" }}>
      Meet Our Team
    </h2>
    <Grid>
      {teamMembers.map((member, index) => (
        <TeamMember key={index}>
          <img src={member.image || "/placeholder.svg"} alt={member.name} />
          <h3>{member.name}</h3>
          <p>{member.role}</p>
          <p>{member.bio}</p>
        </TeamMember>
      ))}
    </Grid>
  </Container>
</TeamSection>

<section style={{ textAlign: "center", padding: "4rem 0" }}>
  <h2 style={{ color: "white", marginBottom: "2rem" }}>Our Mission</h2>
  <p
    style={{
      fontSize: "1.6rem",
      color: "rgba(255,255,255,0.9)",
      maxWidth: "800px",
      margin: "0 auto",
      lineHeight: "1.8",
    }}
  >
    At NoteMaker Pro, we believe that everyone deserves access to powerful, intuitive tools that help them
    organize their thoughts and manage their finances. Our mission is to create a seamless digital experience
    that combines the best of note-taking with smart financial tracking, all while maintaining the highest
    standards of security and user privacy.
```

```
        </p>
      </section>
    </Container>
  </AboutContainer>
 )
}

export default About
```

In Contact.js

```
"use client"

import { useState } from "react"
import styled from "styled-components"
import { Container, Card, Flex } from "../Components/GlobalStyle"
import { Button } from "../Styles/button"

const ContactContainer = styled.div`
  min-height: 100vh;
  padding: 4rem 0;
`

const ContactHero = styled.section`
  text-align: center;
  margin-bottom: 6rem;
`

const ContactTitle = styled.h1`
  font-size: 4rem;
  color: white;
  margin-bottom: 2rem;
  text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
`

const ContactSubtitle = styled.p`
  font-size: 1.8rem;
  color: rgba(255,255,255,0.9);
  max-width: 600px;
  margin: 0 auto;
`

const ContactForm = styled.form`
  background: white;
  padding: 3rem;
  border-radius: 12px;
  box-shadow: 0 8px 32px rgba(0,0,0,0.1);
```

```
  margin-bottom: 4rem;
`

const FormGroup = styled.div`
  margin-bottom: 2rem;
`

const Label = styled.label`
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 600;
  color: #2c3e50;
  font-size: 1.4rem;
`

const Input = styled.input`
  width: 100%;
  padding: 1.2rem;
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  font-size: 1.4rem;
  transition: border-color 0.3s ease;

  &:focus {
    border-color: #667eea;
    outline: none;
  }
`

const TextArea = styled.textarea`
  width: 100%;
  padding: 1.2rem;
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  font-size: 1.4rem;
  min-height: 150px;
  resize: vertical;
  transition: border-color 0.3s ease;

  &:focus {
    border-color: #667eea;
    outline: none;
  }
`

const ContactInfo = styled(Card)`
```

```
text-align: center;
margin-bottom: 2rem;

.icon {
  font-size: 3rem;
  color: #667eea;
  margin-bottom: 1rem;
}

h3 {
  color: #2c3e50;
  margin-bottom: 1rem;
  font-size: 1.8rem;
}

  p {
    color: #7f8c8d;
    font-size: 1.4rem;
  }
`


const SuccessMessage = styled.div`
  background: #d4edda;
  color: #155724;
  padding: 1rem;
  border-radius: 8px;
  margin-bottom: 2rem;
  text-align: center;
  font-weight: 500;
`


const Contact = () => {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    subject: "",
    message: "",
  })
  const [isSubmitting, setIsSubmitting] = useState(false)
  const [showSuccess, setShowSuccess] = useState(false)

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

```
}

const handleSubmit = async (e) => {
  e.preventDefault()
  setIsSubmitting(true)

  // Simulate form submission
  setTimeout(() => {
    setIsSubmitting(false)
    setShowSuccess(true)
    setFormData({
      name: "",
      email: "",
      subject: "",
      message: "",
    })

    // Hide success message after 5 seconds
    setTimeout(() => {
      setShowSuccess(false)
    }, 5000)
  }, 2000)
}

const contactInfo = [
  {
    icon: " 📧 ",
    title: "Email Us",
    info: "support@notemakerPro.com",
    description: "Get in touch for support or inquiries",
  },
  {
    icon: " 📱 ",
    title: "Call Us",
    info: "+91 98765 43210",
    description: "Available Mon-Fri, 9AM-6PM IST",
  },
  {
    icon: " 🏢 ",
    title: "Visit Us",
    info: "Bangalore, Karnataka, India",
    description: "Our headquarters in the Silicon Valley of India",
  },
  {
    icon: " 💬 ",
    title: "Live Chat",
```

```
    info: "Available 24/7",
    description: "Chat with our support team anytime",
  },
]

return (
  <ContactContainer>
    <Container>
      <ContactHero>
        <ContactTitle>Get In Touch</ContactTitle>
        <ContactSubtitle>
          Have questions? We'd love to hear from you. Send us a message and we'll respond as soon as possible.
        </ContactSubtitle>
      </ContactHero>

      <Flex gap="4rem" direction="row" mobileDirection="column">
        <div style={{ flex: 1 }}>
          <ContactForm onSubmit={handleSubmit}>
            <h2 style={{ marginBottom: "2rem", color: "#2c3e50" }}>Send us a Message</h2>

            {showSuccess && <SuccessMessage>Thank you for your message! We'll get back to you soon.</SuccessMessage>}

            <Flex gap="2rem" direction="row" mobileDirection="column">
              <FormGroup style={{ flex: 1 }}>
                <Label htmlFor="name">Full Name</Label>
                <Input
                  type="text"
                  id="name"
                  name="name"
                  value={formData.name}
                  onChange={handleInputChange}
                  required
                  placeholder="Enter your full name"
                />
              </FormGroup>

              <FormGroup style={{ flex: 1 }}>
                <Label htmlFor="email">Email Address</Label>
                <Input
                  type="email"
                  id="email"
                  name="email"
                  value={formData.email}
                  onChange={handleInputChange}
                  required
                  placeholder="Enter your email address"
```

```
    />
  </FormGroup>
</Flex>

<FormGroup>
  <Label htmlFor="subject">Subject</Label>
  <Input
    type="text"
    id="subject"
    name="subject"
    value={formData.subject}
    onChange={handleInputChange}
    required
    placeholder="What's this about?"
  />
</FormGroup>

<FormGroup>
  <Label htmlFor="message">Message</Label>
  <TextArea
    id="message"
    name="message"
    value={formData.message}
    onChange={handleInputChange}
    required
    placeholder="Tell us more about your inquiry..."
  />
</FormGroup>

<Button type="submit" disabled={isSubmitting} size="large">
  {isSubmitting ? "Sending..." : "Send Message"}
</Button>
  </ContactForm>
</div>

<div style={{ flex: 1 }}>
  <h2 style={{ marginBottom: "2rem", color: "white", textAlign: "center" }}>Other Ways to Reach Us</h2>
  {contactInfo.map((info, index) => (
    <ContactInfo key={index}>
      <div className="icon">{info.icon}</div>
      <h3>{info.title}</h3>
      <p style={{ fontWeight: "bold", color: "#2c3e50" }}>{info.info}</p>
      <p>{info.description}</p>
    </ContactInfo>
  ))}
</div>
```

```
</Flex>

<section style={{ marginTop: "6rem", textAlign: "center" }}>
  <h2 style={{ color: "white", marginBottom: "2rem" }}>Follow Us</h2>
  <Flex justify="center" gap="2rem">
    <a href="#" style={{ color: "white", fontSize: "2rem" }}>
      <i className="fab fa-twitter"></i>
    </a>
    <a href="#" style={{ color: "white", fontSize: "2rem" }}>
      <i className="fab fa-facebook"></i>
    </a>
    <a href="#" style={{ color: "white", fontSize: "2rem" }}>
      <i className="fab fa-linkedin"></i>
    </a>
    <a href="#" style={{ color: "white", fontSize: "2rem" }}>
      <i className="fab fa-instagram"></i>
    </a>
  </Flex>
</section>
</Container>
</ContactContainer>
)
}

export default Contact
```

Home.js

```
"use client"

import { useState, useEffect } from "react"
import styled from "styled-components"
import { useAppContext } from "../Components/ContextAPI"
import { Button } from "../Styles/button"
import { Container, Grid, Card, Flex } from "../Components/GlobalStyle"
import HeroSection from "../Components/HeroSection"

const HomeContainer = styled.div`
  min-height: 100vh;
`

const Section = styled.section`
  padding: 4rem 0;
  background: ${(props) => props.bg || "transparent"};
`

const SectionTitle = styled.h2`
```

```
  font-size: 3.2rem;
  text-align: center;
  margin-bottom: 3rem;
  color: ${(props) => props.color || "#333"};
`

const NoteCard = styled(Card)`
  background: ${(props) => props.color || "white"};
  border-left: 4px solid ${(props) => (props.isFinancial ? "#27ae60" : "#667eea")};
  position: relative;

  ${(props) =>
    props.isPinned &&
    `
    &::before {
      content: '📌';
      position: absolute;
      top: 1rem;
      right: 1rem;
      font-size: 1.6rem;
    }
  `}
`

const NoteTitle = styled.h3`
  font-size: 2rem;
  margin-bottom: 1rem;
  color: #2c3e50;
`

const NoteContent = styled.p`
  font-size: 1.4rem;
  line-height: 1.6;
  margin-bottom: 1.5rem;
  color: #555;
`

const NoteActions = styled.div`
  display: flex;
  gap: 1rem;
  margin-top: 1rem;
`

const ActionButton = styled.button`
  background: ${(props) => (props.variant === "danger" ? "#e74c3c" : "#667eea")};
  color: white;
```

```
  border: none;
  padding: 0.5rem 1rem;
  border-radius: 4px;
  cursor: pointer;
  font-size: 1.2rem;
  transition: background 0.3s ease;

  &:hover {
    background: ${(props) => (props.variant === "danger" ? "#c0392b" : "#5a67d8")};
  }
`

const NoteForm = styled.form`
  background: white;
  padding: 2rem;
  border-radius: 12px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  margin-bottom: 3rem;
`

const FormGroup = styled.div`
  margin-bottom: 1.5rem;
`

const Label = styled.label`
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 500;
  color: #2c3e50;
`

const Input = styled.input`
  width: 100%;
  padding: 1rem;
  border: 1px solid #ddd;
  border-radius: 6px;
  font-size: 1.4rem;

  &:focus {
    border-color: #667eea;
    outline: none;
  }
`

const TextArea = styled.textarea`
  width: 100%;
```

```
  padding: 1rem;
  border: 1px solid #ddd;
  border-radius: 6px;
  font-size: 1.4rem;
  min-height: 120px;
  resize: vertical;

  &:focus {
    border-color: #667eea;
    outline: none;
  }
`

const Select = styled.select`
  width: 100%;
  padding: 1rem;
  border: 1px solid #ddd;
  border-radius: 6px;
  font-size: 1.4rem;

  &:focus {
    border-color: #667eea;
    outline: none;
  }
`

const BudgetSummary = styled.div`
  background: linear-gradient(135deg, #27ae60 0%, #2ecc71 100%);
  color: white;
  padding: 2rem;
  border-radius: 12px;
  margin-bottom: 3rem;
  text-align: center;
`

const BudgetItem = styled.div`
  margin-bottom: 1rem;

  h4 {
    font-size: 1.4rem;
    margin-bottom: 0.5rem;
  }

  p {
    font-size: 2rem;
    font-weight: 700;
```

```
  }
`


const Home = () => {
  const { user, notes, budgetSummary, dispatch, token } = useAppContext()
  const [formData, setFormData] = useState({
    title: "",
    content: "",
    category: "personal",
    isFinancial: false,
    amount: "",
    transactionType: "expense",
    color: "#ffffff",
  })
  const [editingNote, setEditingNote] = useState(null)

useEffect(() => {
  if (user && token) {
    fetchNotes()
    fetchBudgetSummary()
  }
}, [user, token])


const fetchNotes = async () => {
  try {
    const response = await fetch(`${process.env.REACT_APP_BASE_URL}/api/notes`, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    const data = await response.json()
    dispatch({ type: "SET_NOTES", payload: data })
  } catch (error) {
    console.error("Error fetching notes:", error)
  }
}


const fetchBudgetSummary = async () => {
  try {
    const response = await fetch(`${process.env.REACT_APP_BASE_URL}/api/notes/budget/summary`, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    const data = await response.json()
    dispatch({ type: "SET_BUDGET_SUMMARY", payload: data })
```

```
  } catch (error) {
    console.error("Error fetching budget summary:", error)
  }
}

const handleInputChange = (e) => {
  const { name, value, type, checked } = e.target
  setFormData({
    ...formData,
    [name]: type === "checkbox" ? checked : value,
  })
}

const handleSubmit = async (e) => {
  e.preventDefault()
   const url = editingNote ? `${process.env.REACT_APP_BASE_URL}/api/notes/${editingNote._id}` :
`${process.env.REACT_APP_BASE_URL}/api/notes`

   const method = editingNote ? "PUT" : "POST"

   try {
    const response = await fetch(url, {
      method,
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${token}`,
      },
      body: JSON.stringify(formData),
    })

    const data = await response.json()

if (editingNote) {
  dispatch({ type: "UPDATE_NOTE", payload: data })
} else {
  dispatch({ type: "ADD_NOTE", payload: data })
}

setFormData({
  title: "",
  content: "",
  category: "personal",
  isFinancial: false,
  amount: "",
  transactionType: "expense",
  color: "#ffffff",
```

```
    })
    setEditingNote(null)

    if (formData.isFinancial) {
      fetchBudgetSummary()
    }
  } catch (error) {
    console.error("Error saving note:", error)
  }
}

const handleEdit = (note) => {
  setEditingNote(note)
  setFormData({
    title: note.title,
    content: note.content,
    category: note.category,
    isFinancial: note.isFinancial,
    amount: note.amount.toString(),
    transactionType: note.transactionType,
    color: note.color,
  })
}

const handleDelete = async (id) => {
  if (window.confirm("Are you sure you want to delete this note?")) {
    try {
      await fetch(`${process.env.REACT_APP_BASE_URL}/api/notes/${id}`, {
        method: "DELETE",
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      dispatch({ type: "DELETE_NOTE", payload: id })
      fetchBudgetSummary()
    } catch (error) {
      console.error("Error deleting note:", error)
    }
  }
}

if (!user) {
  return (
    <HomeContainer>
      <HeroSection />
    </HomeContainer>
```

```
  )
}

return (
  <HomeContainer>
    <Container>
      <Section>
        <SectionTitle>My Notes Dashboard</SectionTitle>

        {budgetSummary && (
          <BudgetSummary>
            <h3>Budget Summary</h3>
            <Flex justify="space-around" align="center">
              <BudgetItem>
                <h4>Total Income</h4>
                <p>
                  {budgetSummary.currency}
                  {budgetSummary.totalIncome}
                </p>
              </BudgetItem>
              <BudgetItem>
                <h4>Total Expenses</h4>
                <p>
                  {budgetSummary.currency}
                  {budgetSummary.totalExpenses}
                </p>
              </BudgetItem>
              <BudgetItem>
                <h4>Balance</h4>
                <p style={{ color: budgetSummary.balance >= 0 ? "#2ecc71" : "#e74c3c" }}>
                  {budgetSummary.currency}
                  {budgetSummary.balance}
                </p>
              </BudgetItem>
            </Flex>
          </BudgetSummary>
        )}

        <NoteForm onSubmit={handleSubmit}>
          <h3>{editingNote ? "Edit Note" : "Create New Note"}</h3>

          <FormGroup>
            <Label htmlFor="title">Title</Label>
            <Input type="text" id="title" name="title" value={formData.title} onChange={handleInputChange} required />
          </FormGroup>
```

```
<FormGroup>
  <Label htmlFor="content">Content</Label>
  <TextArea id="content" name="content" value={formData.content} onChange={handleInputChange} required />
</FormGroup>

<Flex gap="2rem">
  <FormGroup style={{ flex: 1 }}>
    <Label htmlFor="category">Category</Label>
    <Select id="category" name="category" value={formData.category} onChange={handleInputChange}>
      <option value="personal">Personal</option>
      <option value="work">Work</option>
      <option value="finance">Finance</option>
      <option value="shopping">Shopping</option>
      <option value="other">Other</option>
    </Select>
  </FormGroup>

  <FormGroup style={{ flex: 1 }}>
    <Label htmlFor="color">Color</Label>
    <Input type="color" id="color" name="color" value={formData.color} onChange={handleInputChange} />
  </FormGroup>
</Flex>

<FormGroup>
  <Label>
    <input type="checkbox" name="isFinancial" checked={formData.isFinancial} onChange={handleInputChange} />{" "}
    Financial Transaction
  </Label>
</FormGroup>

{formData.isFinancial && (
  <Flex gap="2rem">
    <FormGroup style={{ flex: 1 }}>
      <Label htmlFor="amount">Amount (₹)</Label>
      <Input
        type="number"
        id="amount"
        name="amount"
        value={formData.amount}
        onChange={handleInputChange}
        min="0"
        step="0.01"
      />
    </FormGroup>

    <FormGroup style={{ flex: 1 }}>
```

```
      <Label htmlFor="transactionType">Type</Label>
      <Select
        id="transactionType"
        name="transactionType"
        value={formData.transactionType}
        onChange={handleInputChange}
      >
        <option value="income">Income</option>
        <option value="expense">Expense</option>
      </Select>
    </FormGroup>
  </Flex>
)}

<Button type="submit">{editingNote ? "Update Note" : "Create Note"}</Button>
{editingNote && (
  <Button
    type="button"
    onClick={() => {
      setEditingNote(null)
      setFormData({
        title: "",
        content: "",
        category: "personal",
        isFinancial: false,
        amount: "",
        transactionType: "expense",
        color: "#ffffff",
      })
    }}
    style={{ marginLeft: "1rem", background: "#95a5a6" }}
  >
    Cancel
  </Button>
)}
</NoteForm>

<Grid>
  {notes.map((note) => (
    <NoteCard key={note._id} color={note.color} isFinancial={note.isFinancial} isPinned={note.isPinned}>
      <NoteTitle>{note.title}</NoteTitle>
      <NoteContent>{note.content}</NoteContent>

      <div style={{ marginBottom: "1rem" }}>
        <span
          style={{
```

```
    display: "inline-block",
    background: "#ecf0f1",
    padding: "0.2rem 0.8rem",
    borderRadius: "20px",
    fontSize: "1.2rem",
    marginRight: "1rem",
  }}
>
  {note.category}
</span>

{note.isFinancial && (
  <span
    style={{
      display: "inline-block",
      background: note.transactionType === "income" ? "#27ae60" : "#e74c3c",
      color: "white",
      padding: "0.2rem 0.8rem",
      borderRadius: "20px",
      fontSize: "1.2rem",
    }}
  >
    {note.transactionType === "income" ? "+" : "-"}₹{note.amount}
  </span>
 )}
</div>

<NoteActions>
  <ActionButton onClick={() => handleEdit(note)}>
    <i className="fas fa-edit"></i> Edit
  </ActionButton>
        <ActionButton variant="danger" onClick={() => handleDelete(note._id)}>
          <i className="fas fa-trash"></i> Delete
        </ActionButton>
      </NoteActions>
    </NoteCard>
  ))}
 </Grid>

 {notes.length === 0 && (
  <div style={{ textAlign: "center", padding: "4rem 0" }}>
    <h3>No notes yet!</h3>
    <p>Create your first note to get started.</p>
  </div>
 )}
</Section>
```

```
      </Container>
    </HomeContainer>
  )
}

export default Home
```

Service.js

```
"use client"

import React from "react"
import styled from "styled-components"
import { Container, Card, Grid } from "../Components/GlobalStyle"
import { Button } from "../Styles/button"

const ServiceContainer = styled.div`
  min-height: 100vh;
  padding: 4rem 0;
`

const ServiceHero = styled.section`
  text-align: center;
  margin-bottom: 6rem;
`

const ServiceTitle = styled.h1`
  font-size: 4rem;
  color: white;
  margin-bottom: 2rem;
  text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
`

const ServiceSubtitle = styled.p`
  font-size: 1.8rem;
  color: rgba(255,255,255,0.9);
  max-width: 600px;
  margin: 0 auto;
`

const ServiceCard = styled(Card)`
  text-align: center;
  height: 100%;
  position: relative;

  .icon {
    font-size: 4rem;
```

```
  margin-bottom: 1rem;
}

h3 {
  color: #667eea;
  margin-bottom: 1rem;
  font-size: 2.4rem;
}

.price {
  font-size: 3rem;
  color: #27ae60;
  font-weight: bold;
  margin: 1rem 0;
}

.features {
  text-align: left;
  margin: 2rem 0;
}

.features li {
  margin-bottom: 0.5rem;
  padding-left: 1rem;
  position: relative;
}

.features li:before {
  content: '✓';
  position: absolute;
  left: 0;
    color: #27ae60;
    font-weight: bold;
  }
`

const PricingSection = styled.section`
  background: rgba(255,255,255,0.1);
  padding: 6rem 0;
  margin: 6rem 0;
  border-radius: 20px;
`

const FAQSection = styled.section`
  margin: 6rem 0;
`
```

```
const FAQItem = styled(Card)`
  margin-bottom: 2rem;
  cursor: pointer;

  h3 {
    color: #2c3e50;
    margin-bottom: 1rem;
    display: flex;
    justify-content: space-between;
    align-items: center;
  }

  .answer {
    color: #7f8c8d;
    line-height: 1.6;
    display: ${(props) => (props.isOpen ? "block" : "none")};
  }
`

const Service = () => {
  const [openFAQ, setOpenFAQ] = React.useState(null)

  const services = [
    {
      icon: "📝",
      title: "Smart Note Taking",
      description: "Advanced note-taking with rich text editing, categorization, and intelligent search.",
      features: [
        "Rich text editor with formatting",
        "Smart categorization system",
        "Advanced search and filtering",
        "Note templates and shortcuts",
        "Collaboration features",
      ],
    },
    {
      icon: "💰",
      title: "Budget Management",
      description: "Comprehensive financial tracking with expense categorization and budget planning.",
      features: [
        "Expense and income tracking",
        "Budget planning and alerts",
        "Financial reporting and analytics",
        "Multi-currency support",
        "Bank account integration",
```

```
  ],
 },
 {
  icon: " 🔐 ",
  title: "Security & Privacy",
  description: "Enterprise-grade security with end-to-end encryption and privacy controls.",
  features: [
   "End-to-end encryption",
   "Two-factor authentication",
   "Privacy controls and settings",
   "Regular security audits",
   "GDPR compliance",
  ],
 },
 {
   icon: " 📊 ",
   title: "Analytics & Insights",
   description: "Powerful analytics to understand your note-taking patterns and financial habits.",
   features: [
    "Usage analytics and insights",
    "Financial trend analysis",
    "Productivity metrics",
    "Custom reports and dashboards",
    "Export and sharing options",
   ],
  },
]

const pricingPlans = [
 {
   name: "Free",
  price: "₹0",
  period: "forever",
  features: [
   "Up to 100 notes",
   "Basic categorization",
   "Simple expense tracking",
   "Mobile app access",
   "Email support",
  ],
  recommended: false,
 },
 {
  name: "Pro",
  price: "₹299",
  period: "per month",
```

```
  features: [
    "Unlimited notes",
    "Advanced search and filtering",
    "Comprehensive budget tracking",
    "Priority support",
    "Data export options",
    "Team collaboration",
  ],
  recommended: true,
},
{
  name: "Enterprise",
  price: "₹999",
  period: "per month",
  features: [
    "Everything in Pro",
    "Advanced security features",
    "Custom integrations",
    "Dedicated account manager",
    "SLA guarantees",
    "Custom training",
  ],
  recommended: false,
},
]

const faqs = [
 {
  question: "How secure is my data?",
  answer:
   "We use enterprise-grade security with end-to-end encryption. Your data is encrypted both in transit and at rest, and we never have access to your unencrypted information.",
 },
 {
  question: "Can I access my notes offline?",
  answer:
   "Yes, our mobile apps support offline access. You can create, edit, and view your notes without an internet connection. Changes will sync automatically when you reconnect.",
 },
 {
  question: "Is there a limit to how many notes I can create?",
  answer:
   "Free users can create up to 100 notes. Pro and Enterprise users have unlimited notes with additional features for organization and management.",
 },
 {
  question: "How does the budget tracking work?",
```

```
    answer:
      "You can add financial information to your notes, categorize expenses and income, and get comprehensive reports. The system supports multiple currencies and provides insights into your spending patterns.",
  },
  {
    question: "Can I collaborate with others?",
    answer:
      "Yes, Pro and Enterprise plans include collaboration features. You can share notes, create team workspaces, and work together on projects.",
  },
]

const toggleFAQ = (index) => {
  setOpenFAQ(openFAQ === index ? null : index)
}

return (
  <ServiceContainer>
    <Container>
      <ServiceHero>
        <ServiceTitle>Our Services</ServiceTitle>
        <ServiceSubtitle>
          Comprehensive digital solutions for note-taking, financial management, and productivity enhancement.
        </ServiceSubtitle>
      </ServiceHero>

      <section>
        <h2 style={{ textAlign: "center", fontSize: "3rem", color: "white", marginBottom: "3rem" }}>What We Offer</h2>
        <Grid>
          {services.map((service, index) => (
            <ServiceCard key={index}>
              <div className="icon">{service.icon}</div>
              <h3>{service.title}</h3>
              <p>{service.description}</p>
              <ul className="features">
                {service.features.map((feature, fIndex) => (
                  <li key={fIndex}>{feature}</li>
                ))}
              </ul>
            </ServiceCard>
          ))}
        </Grid>
      </section>

      <PricingSection>
        <Container>
          <h2 style={{ textAlign: "center", fontSize: "3rem", color: "white", marginBottom: "3rem" }}>
```

```
  Choose Your Plan
</h2>
<Grid>
  {pricingPlans.map((plan, index) => (
    <ServiceCard
      key={index}
      style={{
        border: plan.recommended ? "3px solid #667eea" : "none",
        position: "relative",
      }}
    >
      {plan.recommended && (
        <div
          style={{
            position: "absolute",
            top: "-10px",
            left: "50%",
            transform: "translateX(-50%)",
            background: "#667eea",
            color: "white",
            padding: "0.5rem 1rem",
            borderRadius: "20px",
            fontSize: "1.2rem",
            fontWeight: "bold",
          }}
        >
          Recommended
        </div>
      )}
      <h3>{plan.name}</h3>
      <div className="price">{plan.price}</div>
        <p>per {plan.period}</p>
        <ul className="features">
          {plan.features.map((feature, fIndex) => (
            <li key={fIndex}>{feature}</li>
          ))}
        </ul>
        <Button style={{ marginTop: "2rem" }}>{plan.name === "Free" ? "Get Started" : "Choose Plan"}</Button>
      </ServiceCard>
    ))}
  </Grid>
</Container>
</PricingSection>

<FAQSection>
  <h2 style={{ textAlign: "center", fontSize: "3rem", color: "white", marginBottom: "3rem" }}>
```

```
      Frequently Asked Questions
    </h2>
    {faqs.map((faq, index) => (
      <FAQItem key={index} isOpen={openFAQ === index} onClick={() => toggleFAQ(index)}>
        <h3>
          {faq.question}
          <span>{openFAQ === index ? "−" : "+"}</span>
        </h3>
        <div className="answer">
          <p>{faq.answer}</p>
        </div>
      </FAQItem>
    ))}
  </FAQSection>
</Container>
  </ServiceContainer>
 )
}

export default Service
```

SingleUser.js

```
"use client"

import { useState, useEffect } from "react"
import styled from "styled-components"
import { useAppContext } from "../Components/ContextAPI"
import { Container, Card } from "../Components/GlobalStyle"
import { Button } from "../Styles/button"

const ProfileContainer = styled.div`
  min-height: 100vh;
  padding: 4rem 0;
`

const ProfileHeader = styled.section`
  text-align: center;
  margin-bottom: 4rem;
`

const ProfileTitle = styled.h1`
  font-size: 4rem;
  color: white;
  margin-bottom: 2rem;
  text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
`
```

```
const ProfileCard = styled(Card)`
  max-width: 600px;
  margin: 0 auto 4rem;
  text-align: center;

  .avatar {
    width: 120px;
    height: 120px;
    border-radius: 50%;
    object-fit: cover;
    margin-bottom: 2rem;
    border: 4px solid #667eea;
  }

  h2 {
    color: #2c3e50;
    margin-bottom: 1rem;
  }

  .email {
    color: #7f8c8d;
    font-size: 1.6rem;
    margin-bottom: 2rem;
  }
`

const StatsGrid = styled.div`
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 2rem;
  margin-bottom: 4rem;
`

const StatCard = styled(Card)`
  text-align: center;

  .stat-number {
    font-size: 3rem;
    font-weight: bold;
    color: #667eea;
    margin-bottom: 0.5rem;
  }

  .stat-label {
    color: #7f8c8d;
```

```
    font-size: 1.4rem;
  }
`

const EditForm = styled.form`
  background: white;
  padding: 3rem;
  border-radius: 12px;
  box-shadow: 0 8px 32px rgba(0,0,0,0.1);
  max-width: 600px;
  margin: 0 auto;
`

const FormGroup = styled.div`
  margin-bottom: 2rem;
`

const Label = styled.label`
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 600;
  color: #2c3e50;
  font-size: 1.4rem;
`

const Input = styled.input`
  width: 100%;
  padding: 1.2rem;
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  font-size: 1.4rem;
  transition: border-color 0.3s ease;

  &:focus {
    border-color: #667eea;
    outline: none;
  }
`

const ActivityFeed = styled.div`
  max-width: 800px;
  margin: 0 auto;
`

const ActivityItem = styled(Card)`
  margin-bottom: 1rem;
```

```
padding: 1.5rem;
border-left: 4px solid #667eea;

.activity-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 0.5rem;
}

.activity-type {
  font-weight: 600;
  color: #2c3e50;
}

  .activity-time {
    color: #7f8c8d;
    font-size: 1.2rem;
  }

  .activity-description {
    color: #555;
    line-height: 1.6;
  }
`

const SingleUser = () => {
  const { user, notes, token, dispatch } = useAppContext()
  const [isEditing, setIsEditing] = useState(false)
  const [editData, setEditData] = useState({
    name: "",
    email: "",
    currentPassword: "",
    newPassword: "",
    confirmPassword: "",
  })
  const [userStats, setUserStats] = useState({
    totalNotes: 0,
    financialNotes: 0,
    categoriesUsed: 0,
    totalExpenses: 0,
    totalIncome: 0,
  })

  useEffect(() => {
    if (user) {
```

```
    setEditData({
      name: user.name || "",
      email: user.email || "",
      currentPassword: "",
      newPassword: "",
      confirmPassword: "",
    })
    calculateStats()
  }
}, [user, notes])

const calculateStats = () => {
  const financialNotes = notes.filter((note) => note.isFinancial)
  const categories = [...new Set(notes.map((note) => note.category))]
  const totalExpenses = financialNotes
    .filter((note) => note.transactionType === "expense")
    .reduce((sum, note) => sum + note.amount, 0)
  const totalIncome = financialNotes
    .filter((note) => note.transactionType === "income")
    .reduce((sum, note) => sum + note.amount, 0)

  setUserStats({
    totalNotes: notes.length,
    financialNotes: financialNotes.length,
    categoriesUsed: categories.length,
    totalExpenses,
    totalIncome,
  })
}

const handleInputChange = (e) => {
  setEditData({
    ...editData,
    [e.target.name]: e.target.value,
  })
}

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

  if (editData.newPassword && editData.newPassword !== editData.confirmPassword) {
    alert("New passwords do not match!")
    return
  }

  try {
```

```
// Simulate API call to update user profile
const updateData = {
  name: editData.name,
  email: editData.email,
  ...(editData.newPassword && {
    currentPassword: editData.currentPassword,
    newPassword: editData.newPassword,
  }),
}

// For demo purposes, just update the context
dispatch({
  type: "SET_USER",
  payload: {
    ...user,
      name: editData.name,
      email: editData.email,
    },
  })

  setIsEditing(false)
  setEditData({ ...editData, currentPassword: "", newPassword: "", confirmPassword: "" })
  alert("Profile updated successfully!")
 } catch (error) {
  console.error("Error updating profile:", error)
  alert("Failed to update profile. Please try again.")
 }
}

const generateActivityFeed = () => {
 const activities = []

// Add recent note activities
notes.slice(0, 10).forEach((note) => {
 activities.push({
  type: "Note Created",
  description: `Created note: "${note.title}"`,
  time: new Date(note.createdAt).toLocaleDateString(),
  category: note.category,
 })
})

// Add financial activities
const financialNotes = notes.filter((note) => note.isFinancial).slice(0, 5)
financialNotes.forEach((note) => {
 activities.push({
```

```
      type: note.transactionType === "income" ? "Income Added" : "Expense Recorded",
      description: `${note.transactionType === "income" ? "Earned" : "Spent"} ₹${note.amount} - ${note.title}`,
      time: new Date(note.createdAt).toLocaleDateString(),
      category: "finance",
    })
  })

  return activities.sort((a, b) => new Date(b.time) - new Date(a.time))
}

if (!user) {
  return (
    <ProfileContainer>
      <Container>
        <ProfileHeader>
          <ProfileTitle>Please log in to view your profile</ProfileTitle>
        </ProfileHeader>
      </Container>
    </ProfileContainer>
  )
}

return (
  <ProfileContainer>
    <Container>
      <ProfileHeader>
        <ProfileTitle>My Profile</ProfileTitle>
      </ProfileHeader>

      <ProfileCard>
        <img src={user.avatar || "/placeholder.svg?height=120&width=120"} alt="Profile Avatar" className="avatar" />
        <h2>{user.name}</h2>
        <p className="email">{user.email}</p>
        <Button onClick={() => setIsEditing(!isEditing)}>{isEditing ? "Cancel" : "Edit Profile"}</Button>
      </ProfileCard>

      {isEditing && (
        <EditForm onSubmit={handleSubmit}>
          <h3 style={{ marginBottom: "2rem", color: "#2c3e50" }}>Edit Profile</h3>

          <FormGroup>
            <Label htmlFor="name">Full Name</Label>
            <Input type="text" id="name" name="name" value={editData.name} onChange={handleInputChange} required />
          </FormGroup>

          <FormGroup>
```

```
  <Label htmlFor="email">Email Address</Label>
  <Input
    type="email"
    id="email"
    name="email"
    value={editData.email}
    onChange={handleInputChange}
    required
  />
</FormGroup>

<h4 style={{ marginBottom: "1rem", color: "#2c3e50" }}>Change Password (Optional)</h4>

<FormGroup>
  <Label htmlFor="currentPassword">Current Password</Label>
  <Input
    type="password"
    id="currentPassword"
    name="currentPassword"
    value={editData.currentPassword}
    onChange={handleInputChange}
    placeholder="Enter current password"
  />
</FormGroup>

<FormGroup>
  <Label htmlFor="newPassword">New Password</Label>
  <Input
    type="password"
    id="newPassword"
    name="newPassword"
    value={editData.newPassword}
    onChange={handleInputChange}
    placeholder="Enter new password"
  />
</FormGroup>

<FormGroup>
  <Label htmlFor="confirmPassword">Confirm New Password</Label>
  <Input
    type="password"
    id="confirmPassword"
    name="confirmPassword"
    value={editData.confirmPassword}
    onChange={handleInputChange}
    placeholder="Confirm new password"
```

```
      />
    </FormGroup>

    <Button type="submit" size="large">
      Update Profile
    </Button>
  </EditForm>
)}

<section style={{ marginTop: "4rem" }}>
  <h2 style={{ textAlign: "center", color: "white", marginBottom: "3rem" }}>Your Statistics</h2>
  <StatsGrid>
    <StatCard>
      <div className="stat-number">{userStats.totalNotes}</div>
      <div className="stat-label">Total Notes</div>
    </StatCard>
    <StatCard>
      <div className="stat-number">{userStats.financialNotes}</div>
      <div className="stat-label">Financial Notes</div>
    </StatCard>
    <StatCard>
      <div className="stat-number">{userStats.categoriesUsed}</div>
      <div className="stat-label">Categories Used</div>
    </StatCard>
    <StatCard>
      <div className="stat-number">₹{userStats.totalIncome}</div>
      <div className="stat-label">Total Income</div>
    </StatCard>
    <StatCard>
      <div className="stat-number">₹{userStats.totalExpenses}</div>
      <div className="stat-label">Total Expenses</div>
    </StatCard>
    <StatCard>
      <div className="stat-number">₹{userStats.totalIncome - userStats.totalExpenses}</div>
      <div className="stat-label">Net Balance</div>
    </StatCard>
  </StatsGrid>
</section>

<section style={{ marginTop: "4rem" }}>
  <h2 style={{ textAlign: "center", color: "white", marginBottom: "3rem" }}>Recent Activity</h2>
  <ActivityFeed>
    {generateActivityFeed().map((activity, index) => (
      <ActivityItem key={index}>
        <div className="activity-header">
          <span className="activity-type">{activity.type}</span>
```

```
              <span className="activity-time">{activity.time}</span>
            </div>
            <div className="activity-description">{activity.description}</div>
          </ActivityItem>
        ))}
      </ActivityFeed>
    </section>
  </Container>
</ProfileContainer>
)
}

export default SingleUser
```

Now in Style folder

Button.js

```
import styled from "styled-components"


export const Button = styled.button`
  background: ${(props) => {
    if (props.variant === "outline") return "transparent"
    if (props.variant === "danger") return "#e74c3c"
    return "#667eea"
  }};
  color: ${(props) => {
    if (props.variant === "outline") return "#667eea"
    return "white"
  }};
  border: ${(props) => {
    if (props.variant === "outline") return "2px solid #667eea"
    return "none"
  }};
  padding: ${(props) => {
    if (props.size === "large") return "1.5rem 3rem"
    if (props.size === "small") return "0.5rem 1rem"
    return "1rem 2rem"
  }};
  font-size: ${(props) => {
    if (props.size === "large") return "1.8rem"
    if (props.size === "small") return "1.2rem"
    return "1.4rem"
  }};
  border-radius: 8px;
  cursor: pointer;
  font-weight: 600;
```

```
transition: all 0.3s ease;
text-transform: uppercase;
letter-spacing: 1px;

&:hover {
  background: ${(props) => {
    if (props.variant === "outline") return "#667eea"
    if (props.variant === "danger") return "#c0392b"
    return "#5a67d8"
  }};
  color: ${(props) => {
    if (props.variant === "outline") return "white"
    return "white"
  }};
  transform: translateY(-2px);
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
}

&:active {
  transform: translateY(0);
}

&:disabled {
  background: #bdc3c7;
  cursor: not-allowed;
  transform: none;
  box-shadow: none;
}

@media (max-width: 768px) {
  padding: ${(props) => {
    if (props.size === "large") return "1.2rem 2rem"
    if (props.size === "small") return "0.4rem 0.8rem"
    return "0.8rem 1.5rem"
  }};
  font-size: ${(props) => {
    if (props.size === "large") return "1.6rem"
    if (props.size === "small") return "1.1rem"
    return "1.3rem"
  }};
 }
`
```

App.js

```
import { AppProvider } from "./Components/ContextAPI"
import { GlobalStyle } from "./Components/GlobalStyle"
```

```
import AllRouters from "./Components/AllRouters"


function App() {
  return (
    <AppProvider>
      <GlobalStyle />
      <AllRouters />
    </AppProvider>
  )
}

export default App
```

Index.js

```
import React from "react"
import ReactDOM from "react-dom/client"
import App from "./App"


const root = ReactDOM.createRoot(document.getElementById("root"))
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

.env

```
REACT_APP_BASE_URL=https://note-app-backend-zgfm.onrender.com
```

Package.json

```
{
  "name": "note-app-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.15.0",
    "styled-components": "^6.0.7"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": ["react-app", "react-app/jest"]
```

```
  },
  "browserslist": {
    "production": [">0.2%", "not dead", "not op_mini all"],
    "development": ["last 1 chrome version", "last 1 firefox version", "last 1 safari version"]
  },
  "devDependencies": {
    "react-scripts": "5.0.1"
  },
  "proxy": "http://localhost:5000"
}
```