index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/src/assets/TT.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="CRM System" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/css/
bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-SgOJa3DmI69IUzQ2PVdRZhwQ+dy64/
BUtbMJw1MZ8t5HZApcHrRKUc4W0kG879m7"
      crossorigin="anonymous"
    />
    <title>CRM System</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/js/
bootstrap.bundle.min.js"
      integrity="sha384-k6d4wzSIapyDyv1kpU366/
PK5hCdSbCRGRCMv+eplOQJWyd1fbcAu9OCUj5zNLiq"
      crossorigin="anonymous"
    ></script>
  </body>
</html>
```

package-lock.json

```
{
  "name": "client",
  "version": "0.0.0",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "name": "client",
      "version": "0.0.0",
      "dependencies": {
        "@heroicons/react": "^2.0.18",
        "aos": "^2.3.4",
        "axios": "^1.4.0",
        "chart.js": "^4.4.9",
        "date-fns": "^4.1.0",
        "lottie-react": "^2.4.1",
        "lucide-react": "^0.511.0",
        "react": "^18.2.0",
        "react-chartjs-2": "^5.3.0",
        "react-dom": "^18.2.0",
        "react-hot-toast": "^2.4.1",
        "react-icons": "^4.10.1",
```

```
        "react-image-crop": "^11.0.10",
        "react-redux": "^9.2.0",
        "react-router-dom": "^6.14.1",
        "socket.io-client": "^4.8.1",
        "xlsx": "^0.18.5"
      },
      "devDependencies": {
        "@eslint/js": "^9.22.0",
        "@types/react": "^19.0.10",
        "@types/react-dom": "^19.0.4",
        "@vitejs/plugin-react": "^4.3.4",
        "autoprefixer": "^10.4.21",
        "eslint": "^9.22.0",
        "eslint-plugin-react-hooks": "^5.2.0",
        "eslint-plugin-react-refresh": "^0.4.19",
        "globals": "^16.0.0",
        "postcss": "^8.5.3",
        "tailwindcss": "^3.1.0",
        "vite": "^6.3.1"
      }
    },
    "node_modules/@alloc/quick-lru": {
      "version": "5.2.0",
      "resolved": "https://registry.npmjs.org/@alloc/quick-lru/-/quick-
lru-5.2.0.tgz",
      "integrity": "sha512-UrcABB+4bUrFABwbluTIBErXwvbsU/
V7TZWfmbgJfbkwiBuziS9gxdODUyuiecfdGQ85jglMW6juS3+z5TsKLw==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/@ampproject/remapping": {
      "version": "2.3.0",
      "resolved": "https://registry.npmjs.org/@ampproject/remapping/-/
remapping-2.3.0.tgz",
      "integrity": "sha512-30iZtAPgz+LTIYoeivqYo853f02jBYSd5uGnGpkFV0M3xOt9aN73erk
gYAmZU43x4VfqcnLxW9Kpg3R5LC4YYw==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.24"
      },
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@babel/code-frame": {
      "version": "7.26.2",
      "resolved": "https://registry.npmjs.org/@babel/code-frame/-/code-
frame-7.26.2.tgz",
      "integrity": "sha512-RJlIHRueQgwWitWgF8OdFYGZX328Ax5BCemNGlqHfplnRT9ESi8JkFl
vaVYbS+UubVY6dpv87Fs2u5M29iNFVQ==",
      "dev": true,
```

```
      "license": "MIT",
      "dependencies": {
        "@babel/helper-validator-identifier": "^7.25.9",
        "js-tokens": "^4.0.0",
        "picocolors": "^1.0.0"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/compat-data": {
      "version": "7.26.8",
      "resolved": "https://registry.npmjs.org/@babel/compat-data/-/compat-
data-7.26.8.tgz",
      "integrity": "sha512-
oH5UPLMWR3L2wEFLnFJ1TZXqHufiTKAiLfqw5zkhS4dKXLJ10yVztfil/twG8EDTA4F/
tvVNw9nOl4ZMslB8rQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/core": {
      "version": "7.26.10",
      "resolved": "https://registry.npmjs.org/@babel/core/-/core-7.26.10.tgz",
      "integrity": "sha512-vMqyb7XCDMPvJFFOaT9kxtiRh42GwlZEg1/
uIgtZshS5a/8OaduUfCi7kynKgc3Tw/6Uo2D+db9qBttghhmxwQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@ampproject/remapping": "^2.2.0",
        "@babel/code-frame": "^7.26.2",
        "@babel/generator": "^7.26.10",
        "@babel/helper-compilation-targets": "^7.26.5",
        "@babel/helper-module-transforms": "^7.26.0",
        "@babel/helpers": "^7.26.10",
        "@babel/parser": "^7.26.10",
        "@babel/template": "^7.26.9",
        "@babel/traverse": "^7.26.10",
        "@babel/types": "^7.26.10",
        "convert-source-map": "^2.0.0",
        "debug": "^4.1.0",
        "gensync": "^1.0.0-beta.2",
        "json5": "^2.2.3",
        "semver": "^6.3.1"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/babel"
      }
    },
    "node_modules/@babel/core/node_modules/semver": {
      "version": "6.3.1",
      "resolved": "https://registry.npmjs.org/semver/-/semver-6.3.1.tgz",
      "integrity": "sha512-
```

```
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/
MprG9yArRkyrQxTO6XjMzA==",
      "dev": true,
      "license": "ISC",
      "bin": {
        "semver": "bin/semver.js"
      }
    },
    "node_modules/@babel/generator": {
      "version": "7.27.0",
      "resolved": "https://registry.npmjs.org/@babel/generator/-/
generator-7.27.0.tgz",
      "integrity": "sha512-VybsKvpiN1gU1sdMZIp7FcqphVVKEwcuj02x73uvcHE0PTihx1nlBco
wYWhDwjpoAXRv43+gDzyggGnn1XZhVw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/parser": "^7.27.0",
        "@babel/types": "^7.27.0",
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.25",
        "jsesc": "^3.0.2"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-compilation-targets": {
      "version": "7.27.0",
      "resolved": "https://registry.npmjs.org/@babel/helper-compilation-targets/-/
helper-compilation-targets-7.27.0.tgz",
      "integrity": "sha512-LVk7fbXml0H2xH34dFzKQ7TDZ2G4/
rVTOrq9V+icbbadjbVxxeFeDsNHv2SrZeWoA+6ZiTyWYWtScEIW07EAcA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/compat-data": "^7.26.8",
        "@babel/helper-validator-option": "^7.25.9",
        "browserslist": "^4.24.0",
        "lru-cache": "^5.1.1",
        "semver": "^6.3.1"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-compilation-targets/node_modules/semver": {
      "version": "6.3.1",
      "resolved": "https://registry.npmjs.org/semver/-/semver-6.3.1.tgz",
      "integrity": "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/
MprG9yArRkyrQxTO6XjMzA==",
      "dev": true,
      "license": "ISC",
      "bin": {
        "semver": "bin/semver.js"
      }
    },
    "node_modules/@babel/helper-module-imports": {
```

```
      "version": "7.25.9",
      "resolved": "https://registry.npmjs.org/@babel/helper-module-imports/-/
helper-module-imports-7.25.9.tgz",
      "integrity": "sha512-tnUA4RsrmflIM6W6RFTLFSXITtl0wKjgpnLgXyowocVPrbYrLUXSBXD
gTs8BlbmIzIdlBySRQjINYs2BAkiLtw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/traverse": "^7.25.9",
        "@babel/types": "^7.25.9"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-module-transforms": {
      "version": "7.26.0",
      "resolved": "https://registry.npmjs.org/@babel/helper-module-transforms/-/
helper-module-transforms-7.26.0.tgz",
      "integrity": "sha512-xO+xu6B5K2czEnQye6BHA7DolFFmS3LB7stHZFaOLb1pAwO1HWLS8fX
A+eh0A2yIvltPVmx3eNNDBJA2SLHXFw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-module-imports": "^7.25.9",
        "@babel/helper-validator-identifier": "^7.25.9",
        "@babel/traverse": "^7.25.9"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "peerDependencies": {
        "@babel/core": "^7.0.0"
      }
    },
    "node_modules/@babel/helper-plugin-utils": {
      "version": "7.26.5",
      "resolved": "https://registry.npmjs.org/@babel/helper-plugin-utils/-/helper-
plugin-utils-7.26.5.tgz",
      "integrity": "sha512-RS+jZcRdZdRFzMyr+wcsaqOmld1/EqTghfaBGQQd/WnRdzdlvSZ//
kF7U8VQTxf1ynZ4cjUcYgjVGx13ewNPMg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-string-parser": {
      "version": "7.25.9",
      "resolved": "https://registry.npmjs.org/@babel/helper-string-parser/-/
helper-string-parser-7.25.9.tgz",
      "integrity": "sha512-4A/SCr/2KLd5jrtOMFzaKjVtAei3+2r/NChoBNoZ3EyP/
+GlhoaEGoWOZUmFmoITP7zOJyHIMm+DYRd8o3PvHA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
```

```
    "node_modules/@babel/helper-validator-identifier": {
      "version": "7.25.9",
      "resolved": "https://registry.npmjs.org/@babel/helper-validator-
identifier/-/helper-validator-identifier-7.25.9.tgz",
      "integrity": "sha512-Ed61U6XJc3CVRfkERJWDz4dJwKe7iLmmJsbOGu9wSloNSFttHV0I8g6
UAgb7qnK5ly5bGLPd4oXZlxCdANBOWQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helper-validator-option": {
      "version": "7.25.9",
      "resolved": "https://registry.npmjs.org/@babel/helper-validator-option/-/
helper-validator-option-7.25.9.tgz",
      "integrity": "sha512-e/zv1co8pp55dNdEcCynfj9X7nyUKUXoUEwfXqaZt0omVOmDe9oOTdK
StH4GmAw6zxMFs50ZayuMfHDKlO7Tfw==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/helpers": {
      "version": "7.27.0",
      "resolved": "https://registry.npmjs.org/@babel/helpers/-/
helpers-7.27.0.tgz",
      "integrity": "sha512-U5eyP/CTFPuNE3qk+WZMxFkp/4zUzdceQlfzf7DdGdhp+Fezd7HD+i8
Y24ZuTMKX3wQBld449jijbGq6OdGNQg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/template": "^7.27.0",
        "@babel/types": "^7.27.0"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/parser": {
      "version": "7.27.0",
      "resolved": "https://registry.npmjs.org/@babel/parser/-/parser-7.27.0.tgz",
      "integrity": "sha512-iaepho73/2Pz7w2eMS0Q5f83+0RKI7i4xmiYeBmDzfRVbQtTOG7Ts0S
4HzJVsTMGI9keU8rNfuZr8DKfSt7Yyg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/types": "^7.27.0"
      },
      "bin": {
        "parser": "bin/babel-parser.js"
      },
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@babel/plugin-transform-react-jsx-self": {
      "version": "7.25.9",
```

```
      "resolved": "https://registry.npmjs.org/@babel/plugin-transform-react-jsx-
self/-/plugin-transform-react-jsx-self-7.25.9.tgz",
      "integrity": "sha512-y8quW6p0WHkEhmErnfe58r7x0A70uKphQm8Sp8cV7tjNQwK56sNVK0M
73LK3WuYmsuyrftut4xAkjjgU0twaMg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-plugin-utils": "^7.25.9"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "peerDependencies": {
        "@babel/core": "^7.0.0-0"
      }
    },
    "node_modules/@babel/plugin-transform-react-jsx-source": {
      "version": "7.25.9",
      "resolved": "https://registry.npmjs.org/@babel/plugin-transform-react-jsx-
source/-/plugin-transform-react-jsx-source-7.25.9.tgz",
      "integrity": "sha512-+iqjT8xmXhhYv4/
uiYd8FNQsraMFZIfxVSqxxVSZP0WbbSAWvBXAul0m/zu+7Vv4O/3WtApy9pmaTMiumEZgfg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-plugin-utils": "^7.25.9"
      },
      "engines": {
        "node": ">=6.9.0"
      },
      "peerDependencies": {
        "@babel/core": "^7.0.0-0"
      }
    },
    "node_modules/@babel/template": {
      "version": "7.27.0",
      "resolved": "https://registry.npmjs.org/@babel/template/-/
template-7.27.0.tgz",
      "integrity": "sha512-2ncevenBqXI6qRMukPlXwHKHchC7RyMuu4xv5JBXRfOGVcTy1mXCD12
qrp7Jsoxll1EV3+9sE4GugBVRjT2jFA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/code-frame": "^7.26.2",
        "@babel/parser": "^7.27.0",
        "@babel/types": "^7.27.0"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/traverse": {
      "version": "7.27.0",
      "resolved": "https://registry.npmjs.org/@babel/traverse/-/
traverse-7.27.0.tgz",
      "integrity": "sha512-19lYZFzYVQkkHkl4Cy4WrAVcqBkgvV2YM2TU3xG6DIwO7O3ecbDPfW3
yM3bjAGcqcQHi+CCtjMR3dIEHxsd6bA==",
      "dev": true,
      "license": "MIT",
```

```
      "dependencies": {
        "@babel/code-frame": "^7.26.2",
        "@babel/generator": "^7.27.0",
        "@babel/parser": "^7.27.0",
        "@babel/template": "^7.27.0",
        "@babel/types": "^7.27.0",
        "debug": "^4.3.1",
        "globals": "^11.1.0"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@babel/traverse/node_modules/globals": {
      "version": "11.12.0",
      "resolved": "https://registry.npmjs.org/globals/-/globals-11.12.0.tgz",
      "integrity": "sha512-WOBp/EEGUiIsJSp7wcv/
y6MO+lV9UoncWqxuFfm8eBwzWNgyfBd6Gz+IeKQ9jCmyhoH99g15M3T+QaVHFjizVA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=4"
      }
    },
    "node_modules/@babel/types": {
      "version": "7.27.0",
      "resolved": "https://registry.npmjs.org/@babel/types/-/types-7.27.0.tgz",
      "integrity": "sha512-H45s8fVLYjbhFH62dIJ3WtmJ6RSPt/3DRO0ZcT2SUiYiQyz3BLVb9AD
EnLl91m74aQPS3AzzeajZHYOalWe3bg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/helper-string-parser": "^7.25.9",
        "@babel/helper-validator-identifier": "^7.25.9"
      },
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/@esbuild/aix-ppc64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/aix-ppc64/-/aix-
ppc64-0.25.3.tgz",
      "integrity": "sha512-W8bFfPA8DowP8l//
sxjJLSLkD8iEjMc7cBVyP+u4cEv9sM7mdUCkgsj+t0n/BWPFtv7WWCN5Yzj0N6FJNUUqBQ==",
      "cpu": [
        "ppc64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "aix"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/android-arm": {
```

```
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/android-arm/-/android-
arm-0.25.3.tgz",
      "integrity": "sha512-PuwVXbnP87Tcff5I9ngV0lmiSu40xw1At6i3GsU77U7cjDDB4s0X2cy
FuBiDa1SBk9DnvWwnGvVaGBqoFWPb7A==",
      "cpu": [
        "arm"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "android"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/android-arm64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/android-arm64/-/android-
arm64-0.25.3.tgz",
      "integrity": "sha512-
XelR6MzjlZuBM4f5z2IQHK6LkK34Cvv6Rj2EntER3lwCBFdg6h2lKbtRjpTTsdEjD/
WSe1q8UyPBXP1x3i/wYQ==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "android"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/android-x64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/android-x64/-/android-
x64-0.25.3.tgz",
      "integrity": "sha512-ogtTpYHT/g1GWS/zKM0cc/tIebFjm1F9Aw1boQ2Y0eUQ+J89d0jFY//
s9ei9jVIlkYi8AfOjiixcLJSGNSOAdQ==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "android"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/darwin-arm64": {
      "version": "0.25.3",
```

```
      "resolved": "https://registry.npmjs.org/@esbuild/darwin-arm64/-/darwin-
arm64-0.25.3.tgz",
      "integrity": "sha512-eESK5yfPNTqpAmDfFWNsOhmIOaQA59tAcF/EfYvo5/
QWQCzXn5iUSOnqt3ra3UdzBv073ykTtmeLJZGt3HhA+w==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "darwin"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/darwin-x64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/darwin-x64/-/darwin-
x64-0.25.3.tgz",
      "integrity": "sha512-Kd8glo7sIZtwOLcPbW0yLpKmBNWMANZhrC1r6K+
+uDR2zyzb6AeOYtI6udbtabmQpFaxJ8uduXMAo1gs5ozz8A==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "darwin"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/freebsd-arm64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/freebsd-arm64/-/freebsd-
arm64-0.25.3.tgz",
      "integrity": "sha512-EJiyS70BYybOBpJth3M0KLOus0n+RRMKTYzhYhFeMwp7e/
RaajXvP+BWlmEXNk6uk+KAu46j/kaQzr6au+JcIw==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "freebsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/freebsd-x64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/freebsd-x64/-/freebsd-
x64-0.25.3.tgz",
```

```
      "integrity": "sha512-Q+wSjaLpGxYf7zC0kL0nDlhsfuFkoN+EXrx2KSB33RhinWzejOd6Avg
mP5JbkgXKmjhmpfgKZq24pneodYqE8Q==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "freebsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-arm": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-arm/-/linux-
arm-0.25.3.tgz",
      "integrity": "sha512-dUOVmAUzuHy2ZOKIHIKHCm58HKzFqd+puLaS424h6I85GlSDRZIA5yc
Bixb3mFgM0Jdh+ZOSB6KptX30DD8YOQ==",
      "cpu": [
        "arm"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-arm64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-arm64/-/linux-
arm64-0.25.3.tgz",
      "integrity": "sha512-xCUgnNYhRD5bb1C1nqrDV1PfkwgbswTTBRbAd8aH5PhYzikdf/
ddtsYyMXFfGSsb/6t6QaPSzxtbfAZr9uox4A==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-ia32": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-ia32/-/linux-
ia32-0.25.3.tgz",
      "integrity": "sha512-yplPOpczHOO4jTYKmuYuANI3WhvIPSVANGcNUeMlxH4twz/
TeXuzEP41tGKNGWJjuMhotpGabeFYGAOU2ummBw==",
```

```
      "cpu": [
        "ia32"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-loong64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-loong64/-/linux-
loong64-0.25.3.tgz",
      "integrity": "sha512-P4BLP5/fjyihmXCELRGrLd793q/lBtKMQl8ARGpDxgzgIKJDRJ/
u4r1A/HgpBpKpKZelGct2PGI4T+axcedf6g==",
      "cpu": [
        "loong64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-mips64el": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-mips64el/-/linux-
mips64el-0.25.3.tgz",
      "integrity": "sha512-eRAOV2ODpu6P5divMEMa26RRqb2yUoYsuQQOuFUexUoQndm4MdpXXDB
bUoKIc0iPa4aCO7gIhtnYomkn2x+bag==",
      "cpu": [
        "mips64el"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-ppc64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-ppc64/-/linux-
ppc64-0.25.3.tgz",
      "integrity": "sha512-
ZC4jV2p7VbzTlnl8nZKLcBkfzIf4Yad1SJM4ZMKYnJqZFD4rTI+pBG65u8ev4jk3/
MPwY9DvGn50wi3uhdaghg==",
      "cpu": [
```

```
        "ppc64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-riscv64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-riscv64/-/linux-
riscv64-0.25.3.tgz",
      "integrity": "sha512-LDDODcFzNtECTrUUbVCs6j9/
bDVqy7DDRsuIXJg6so+mFksgwG7ZVnTruYi5V+z3eE5y+BJZw7VvUadkbfg7QA==",
      "cpu": [
        "riscv64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-s390x": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-s390x/-/linux-
s390x-0.25.3.tgz",
      "integrity": "sha512-s+w/
NOY2k0yC2p9SLen+ymflgcpRkvwwa02fqmAwhBRI3SC12uiS10edHHXlVWwfAagYSY5UpmT/
zISXPMW3tQ==",
      "cpu": [
        "s390x"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/linux-x64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/linux-x64/-/linux-
x64-0.25.3.tgz",
      "integrity": "sha512-
nQHDz4pXjSDC6UfOE1Fw9Q8d6GCAd9KdvMZpfVGWSJztYCarRgSDfOVBY5xwhQXseiyxapkiSJi/5/
ja8mRFFA==",
      "cpu": [
```

```
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/netbsd-arm64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/netbsd-arm64/-/netbsd-
arm64-0.25.3.tgz",
      "integrity":
"sha512-1QaLtOWq0mzK6tzzp0jRN3eccmN3hezey7mhLnzC6oNlJoUJz4nym5ZD7mDnS/
LZQgkrhEbEiTn515lPeLpgWA==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "netbsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/netbsd-x64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/netbsd-x64/-/netbsd-
x64-0.25.3.tgz",
      "integrity": "sha512-i5Hm68HXHdgv8wkrt+10Bc50zM0/eonPb/a/
OFVfB6QvpiirCo5gBA5bz7S2SHuU+Y4LWn/zehzNX14Sp4r27g==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "netbsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/openbsd-arm64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/openbsd-arm64/-/openbsd-
arm64-0.25.3.tgz",
      "integrity": "sha512-zGAVApJEYTbOC6H/3QBr2mq3upG/LBEXr85/
pTtKiv2IXcgKV0RT0QA/hSXZqSvLEpXeIxah7LczB4lkiYhTAQ==",
      "cpu": [
        "arm64"
```

```
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "openbsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/openbsd-x64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/openbsd-x64/-/openbsd-
x64-0.25.3.tgz",
      "integrity": "sha512-fpqctI45NnCIDKBH5AXQBsD0NDPbEFczK98hk/
aa6HJxbl+UtLkJV2+Bvy5hLSLk3LHmqt0NTkKNso1A9y1a4w==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "openbsd"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/sunos-x64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/sunos-x64/-/sunos-
x64-0.25.3.tgz",
      "integrity": "sha512-ROJhm7d8bk9dMCUZjkS8fgzsPAZEjtRJqCAmVgB0gMrvG7hfmPmz9k1
rwO4jSiblFjYmNvbECL9uhaPzONMfgA==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "sunos"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/win32-arm64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/win32-arm64/-/win32-
arm64-0.25.3.tgz",
      "integrity": "sha512-YWcow8peiHpNBiIXHwaswPnAXLsLVygFwCB3A7Bh5jRkIBFWHGmNQ48
AlX4xDvQNoMZlPYzjVOQDYEzWCqufMQ==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
```

```
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/win32-ia32": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/win32-ia32/-/win32-
ia32-0.25.3.tgz",
      "integrity": "sha512-qspTZOIGoXVS4DpNqUYUs9UxVb04khS1Degaw/
MnfMe7goQ3lTfQ13Vw4qY/Nj0979BGvMRpAYbs/BAxEvU8ew==",
      "cpu": [
        "ia32"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@esbuild/win32-x64": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/@esbuild/win32-x64/-/win32-
x64-0.25.3.tgz",
      "integrity": "sha512-ICgUR+kPimx0vvRzf+N/7L7tVSQeE3BYY+NhHRHXS1kBuPO7z2+7ea2
HbhDyZdTephgvNvKrlDDKUexuCVBVvg==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
      ],
      "engines": {
        "node": ">=18"
      }
    },
    "node_modules/@eslint-community/eslint-utils": {
      "version": "4.6.1",
      "resolved": "https://registry.npmjs.org/@eslint-community/eslint-utils/-/
eslint-utils-4.6.1.tgz",
      "integrity": "sha512-KTsJMmobmbrFLe3LDh0PC2FXpcSYJt/
MLjlkh/9LEnmKYLSYmT/0EW9JWANjeoemiuZrmogti0tW5Ch+qNUYDw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "eslint-visitor-keys": "^3.4.3"
      },
      "engines": {
```

```
        "node": "^12.22.0 || ^14.17.0 || >=16.0.0"
      },
      "funding": {
        "url": "https://opencollective.com/eslint"
      },
      "peerDependencies": {
        "eslint": "^6.0.0 || ^7.0.0 || >=8.0.0"
      }
    },
    "node_modules/@eslint-community/regexpp": {
      "version": "4.12.1",
      "resolved": "https://registry.npmjs.org/@eslint-community/regexpp/-/
regexpp-4.12.1.tgz",
      "integrity": "sha512-CCZCDJuduB9OUkFkY2IgppNZMi2lBQgD2qzwXkEia16cge2pijY/
aXi96CJMquDMn3nJdlPV1A5KrJEXwfLNzQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": "^12.0.0 || ^14.0.0 || >=16.0.0"
      }
    },
    "node_modules/@eslint/config-array": {
      "version": "0.20.0",
      "resolved": "https://registry.npmjs.org/@eslint/config-array/-/config-
array-0.20.0.tgz",
      "integrity": "sha512-fxlS1kkIjx8+vy2SjuCB94q3htSNrufYTXubwiBFeaQHbH6Ipi43gFJ
q2zCMt6PHhImH3Xmr0NksKDvchWlpQQ==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@eslint/object-schema": "^2.1.6",
        "debug": "^4.3.1",
        "minimatch": "^3.1.2"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/config-helpers": {
      "version": "0.2.1",
      "resolved": "https://registry.npmjs.org/@eslint/config-helpers/-/config-
helpers-0.2.1.tgz",
      "integrity": "sha512-RI17tsD2frtDu/3dmI7QRrD4bedNKPM08ziRYaC5AhkGrzIAJelm9kJ
U1TznK+apx6V+cqRz8tfpEeG3oIyjxw==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/core": {
      "version": "0.13.0",
      "resolved": "https://registry.npmjs.org/@eslint/core/-/core-0.13.0.tgz",
      "integrity": "sha512-yfkgDw1KR66rkT5A8ci4irzDysN7FRpq3ttJolR88OqQikAWqwA8j5V
Zyas+vjyBNFIJ7MfybJ9plMILI2UrCw==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@types/json-schema": "^7.0.15"
```

```
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/eslintrc": {
      "version": "3.3.1",
      "resolved": "https://registry.npmjs.org/@eslint/eslintrc/-/
eslintrc-3.3.1.tgz",
      "integrity": "sha512-gtF186CXhIl1p4pJNGZw8Yc6RlshoePRvE0X91oPGb3vZ8pM3qOS9W9
NGPat9LziaBV7XrJWGylNQXkGcnM3IQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ajv": "^6.12.4",
        "debug": "^4.3.2",
        "espree": "^10.0.1",
        "globals": "^14.0.0",
        "ignore": "^5.2.0",
        "import-fresh": "^3.2.1",
        "js-yaml": "^4.1.0",
        "minimatch": "^3.1.2",
        "strip-json-comments": "^3.1.1"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      },
      "funding": {
        "url": "https://opencollective.com/eslint"
      }
    },
    "node_modules/@eslint/eslintrc/node_modules/argparse": {
      "version": "2.0.1",
      "resolved": "https://registry.npmjs.org/argparse/-/argparse-2.0.1.tgz",
      "integrity":
"sha512-8+9WqebbFzpX9OR+Wa6O29asIogeRMzcGtAINdpMHHyAg10f05aSFVBbcEqGf/PXw1EjAZ+q2/
bEBg3DvurK3Q==",
      "dev": true,
      "license": "Python-2.0"
    },
    "node_modules/@eslint/eslintrc/node_modules/globals": {
      "version": "14.0.0",
      "resolved": "https://registry.npmjs.org/globals/-/globals-14.0.0.tgz",
      "integrity": "sha512-oahGvuMGQlPw/
ivIYBjVSrWAfWLBeku5tpPE2fOPLi+WHffIWbuh2tCjhyQhTBPMf5E9jDEH4FOmTYgYwbKwtQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=18"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/@eslint/eslintrc/node_modules/js-yaml": {
      "version": "4.1.0",
      "resolved": "https://registry.npmjs.org/js-yaml/-/js-yaml-4.1.0.tgz",
      "integrity": "sha512-wpxZs9NoxZaJESJGIZTyDEaYpl0FKSA+FB9aJiyemKhMwkxQg63h4T1
KJgUGHpTqPDNRcmmYLugrRjJlBtWvRA==",
```

```
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "argparse": "^2.0.1"
      },
      "bin": {
        "js-yaml": "bin/js-yaml.js"
      }
    },
    "node_modules/@eslint/js": {
      "version": "9.25.1",
      "resolved": "https://registry.npmjs.org/@eslint/js/-/js-9.25.1.tgz",
      "integrity": "sha512-dEIwmjntEx8u3Uvv+kr3PDeeArL8Hw07H9kyYxCjnM9pBjfEhk6uLXS
chxxzgiwtRhhzVzqmUSDFBOi1TuZ7qg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/object-schema": {
      "version": "2.1.6",
      "resolved": "https://registry.npmjs.org/@eslint/object-schema/-/object-
schema-2.1.6.tgz",
      "integrity": "sha512-RBMg5FRL0I0gs51M/guSAj5/
e14VQ4tpZnQNWwuDT66P14I43ItmPfIZRhO9fUVIPOAQXU47atlywZ/czoqFPA==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@eslint/plugin-kit": {
      "version": "0.2.8",
      "resolved": "https://registry.npmjs.org/@eslint/plugin-kit/-/plugin-
kit-0.2.8.tgz",
      "integrity": "sha512-ZAoA40rNMPwSm+AeHpCq8STiNAwzWLJuP8Xv4CHIc9wv/
PSuExjMrmjfYNj682vW0OOiZ1HKxzvjQr9XZIisQA==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@eslint/core": "^0.13.0",
        "levn": "^0.4.1"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      }
    },
    "node_modules/@heroicons/react": {
      "version": "2.2.0",
      "resolved": "https://registry.npmjs.org/@heroicons/react/-/react-2.2.0.tgz",
      "integrity": "sha512-LMcepvRaS9LYHJGsF0zzmgKCUim/X3N/
DQKc4jepAXJ7l8QxJ1PmxJzqplF2Z3FE4PqBAIGyJAQ/w4B5dsqbtQ==",
      "license": "MIT",
      "peerDependencies": {
        "react": ">= 16 || ^19.0.0-rc"
      }
    },
    "node_modules/@humanfs/core": {
```

```
      "version": "0.19.1",
      "resolved": "https://registry.npmjs.org/@humanfs/core/-/core-0.19.1.tgz",
      "integrity": "sha512-5DyQ4+1JEUzejeK1JGICcideyfUbGixgS9jNgex5nqkW+cY7WZhxBig
mieN5Qnw9ZosSNVC9KQKyb+GUaGyKUA==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": ">=18.18.0"
      }
    },
    "node_modules/@humanfs/node": {
      "version": "0.16.6",
      "resolved": "https://registry.npmjs.org/@humanfs/node/-/node-0.16.6.tgz",
      "integrity": "sha512-YuI2ZHQL78Q5HbhDiBA1X4LmYdXCKCMQIfw0pw7piHJwyREFebJUvrQ
N4cMssyES6x+vfUbx1CIpaQUKYdQZOw==",
      "dev": true,
      "license": "Apache-2.0",
      "dependencies": {
        "@humanfs/core": "^0.19.1",
        "@humanwhocodes/retry": "^0.3.0"
      },
      "engines": {
        "node": ">=18.18.0"
      }
    },
    "node_modules/@humanfs/node/node_modules/@humanwhocodes/retry": {
      "version": "0.3.1",
      "resolved": "https://registry.npmjs.org/@humanwhocodes/retry/-/
retry-0.3.1.tgz",
      "integrity": "sha512-JBxkERygn7Bv/GbN5Rv8Ul6LVknS+5Bp6RgDC/O8gEBU/yeH5Ui5C/
OlWrTb6qct7LjjfT6Re2NxB0ln0yYybA==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": ">=18.18"
      },
      "funding": {
        "type": "github",
        "url": "https://github.com/sponsors/nzakas"
      }
    },
    "node_modules/@humanwhocodes/module-importer": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/@humanwhocodes/module-importer/-/
module-importer-1.0.1.tgz",
      "integrity": "sha512-bxveV4V8v5Yb4ncFTT3rPSgZBOpCkjfK0y4oVVVJwIuDVBRMDXrPyXR
L988i5ap9m9bnyEEjWfm5WkBmtffLfA==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": ">=12.22"
      },
      "funding": {
        "type": "github",
        "url": "https://github.com/sponsors/nzakas"
      }
    },
    "node_modules/@humanwhocodes/retry": {
      "version": "0.4.2",
```

```
      "resolved": "https://registry.npmjs.org/@humanwhocodes/retry/-/
retry-0.4.2.tgz",
      "integrity": "sha512-xeO57FpIu4p1Ri3Jq/EXq4ClRm86dVF2z/
+kvFnyqVYRavTZmaFaUBbWCOuuTh0o/g7DSsk6kc2vrS4Vl5oPOQ==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": ">=18.18"
      },
      "funding": {
        "type": "github",
        "url": "https://github.com/sponsors/nzakas"
      }
    },
    "node_modules/@isaacs/cliui": {
      "version": "8.0.2",
      "resolved": "https://registry.npmjs.org/@isaacs/cliui/-/cliui-8.0.2.tgz",
      "integrity": "sha512-O8jcjabXaleOG9DQ0+ARXWZBTfnP4WNAqzuiJK7ll44AmxGKv/
J2M4TPjxjY3znBCfvBXFzucm1twdyFybFqEA==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "string-width": "^5.1.2",
        "string-width-cjs": "npm:string-width@^4.2.0",
        "strip-ansi": "^7.0.1",
        "strip-ansi-cjs": "npm:strip-ansi@^6.0.1",
        "wrap-ansi": "^8.1.0",
        "wrap-ansi-cjs": "npm:wrap-ansi@^7.0.0"
      },
      "engines": {
        "node": ">=12"
      }
    },
    "node_modules/@jridgewell/gen-mapping": {
      "version": "0.3.8",
      "resolved": "https://registry.npmjs.org/@jridgewell/gen-mapping/-/gen-
mapping-0.3.8.tgz",
      "integrity": "sha512-imAbBGkb+ebQyxKgzv5Hu2nmROxoDOXHh80evxdoXNOrvAnVx7zimzc
1Oo5h9RlfV4vPXaE2iM5pOFbvOCClWA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@jridgewell/set-array": "^1.2.1",
        "@jridgewell/sourcemap-codec": "^1.4.10",
        "@jridgewell/trace-mapping": "^0.3.24"
      },
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@jridgewell/resolve-uri": {
      "version": "3.1.2",
      "resolved": "https://registry.npmjs.org/@jridgewell/resolve-uri/-/resolve-
uri-3.1.2.tgz",
      "integrity": "sha512-bRISgCIjP20/
tbWSPWMEi54QVPRZExkuD9lJL+UIxUKtwVJA8wW1Trb1jMs1RFXo1CBTNZ/5hpC9QvmKWdopKw==",
      "dev": true,
      "license": "MIT",
      "engines": {
```

```
        "node": ">=6.0.0"
      }
    },
    "node_modules/@jridgewell/set-array": {
      "version": "1.2.1",
      "resolved": "https://registry.npmjs.org/@jridgewell/set-array/-/set-
array-1.2.1.tgz",
      "integrity": "sha512-R8gLRTZeyp03ymzP/6Lil/28tGeGEzhx1q2k703KGWRAI1VdvPIXdG7
0VJc2pAMw3NA6JKL5hhFu1sJX0Mnn/A==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.0.0"
      }
    },
    "node_modules/@jridgewell/source-map": {
      "version": "0.3.6",
      "resolved": "https://registry.npmjs.org/@jridgewell/source-map/-/source-
map-0.3.6.tgz",
      "integrity": "sha512-1ZJTZebgqllO79ue2bm3rIGud/
bOe0pP5BjSRCRxxYkEZS8STV7zN84UBbiYu7jy+eCKSnVIUgoWWE/tt+shMQ==",
      "dev": true,
      "license": "MIT",
      "optional": true,
      "peer": true,
      "dependencies": {
        "@jridgewell/gen-mapping": "^0.3.5",
        "@jridgewell/trace-mapping": "^0.3.25"
      }
    },
    "node_modules/@jridgewell/sourcemap-codec": {
      "version": "1.5.0",
      "resolved": "https://registry.npmjs.org/@jridgewell/sourcemap-codec/-/
sourcemap-codec-1.5.0.tgz",
      "integrity": "sha512-gv3ZRaISU3fjPAgNsriBRqGWQL6quFx04YMPW/
zD8XMLsU32mhCCbfbO6KZFLjvYpCZ8zyDEgqsgf+PwPaM7GQ==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/@jridgewell/trace-mapping": {
      "version": "0.3.25",
      "resolved": "https://registry.npmjs.org/@jridgewell/trace-mapping/-/trace-
mapping-0.3.25.tgz",
      "integrity": "sha512-vNk6aEwybGtawWmy/
PzwnGDOjCkLWSD2wqvjGGAgOAwCGWySYXfYoxt00IJkTF+8Lb57DwOb3Aa0o9CApepiYQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@jridgewell/resolve-uri": "^3.1.0",
        "@jridgewell/sourcemap-codec": "^1.4.14"
      }
    },
    "node_modules/@kurkle/color": {
      "version": "0.3.4",
      "resolved": "https://registry.npmjs.org/@kurkle/color/-/color-0.3.4.tgz",
      "integrity": "sha512-M5UknZPHRu3DEDWoipU6sE8PdkZ6Z/
S+v4dD+Ke8IaNlpdSQah50lz1KtcFBa2vsdOnwbbnxJwVM4wty6udA5w==",
      "license": "MIT"
    },
```

```
    "node_modules/@nodelib/fs.scandir": {
      "version": "2.1.5",
      "resolved": "https://registry.npmjs.org/@nodelib/fs.scandir/-/
fs.scandir-2.1.5.tgz",
      "integrity": "sha512-vq24Bq3ym5HEQm2NKCr3yXDwjc7vTsEThRDnkp2DK9p1uqLR+DHurm/
NOTo0KG7HYHU7eppKZj3MyqYuMBf62g==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@nodelib/fs.stat": "2.0.5",
        "run-parallel": "^1.1.9"
      },
      "engines": {
        "node": ">= 8"
      }
    },
    "node_modules/@nodelib/fs.stat": {
      "version": "2.0.5",
      "resolved": "https://registry.npmjs.org/@nodelib/fs.stat/-/
fs.stat-2.0.5.tgz",
      "integrity": "sha512-RkhPPp2zrqDAQA/2jNhnztcPAlv64XdhIp7a7454A5ovI7Bukxgt7MX
7udwAu3zg1DcpPU0rz3VV1SeaqvY4+A==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">= 8"
      }
    },
    "node_modules/@nodelib/fs.walk": {
      "version": "1.2.8",
      "resolved": "https://registry.npmjs.org/@nodelib/fs.walk/-/
fs.walk-1.2.8.tgz",
      "integrity": "sha512-
oGB+UxlgWcgQkgwo8GcEGwemoTFt3FIO9ababBmaGwXIoBKZ+GTy0pP185beGg7Llih/
NSHSV2XAs1lnznocSg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@nodelib/fs.scandir": "2.1.5",
        "fastq": "^1.6.0"
      },
      "engines": {
        "node": ">= 8"
      }
    },
    "node_modules/@pkgjs/parseargs": {
      "version": "0.11.0",
      "resolved": "https://registry.npmjs.org/@pkgjs/parseargs/-/
parseargs-0.11.0.tgz",
      "integrity": "sha512-
+1VkjdD0QBLPodGrJUeqarH8VAIvQODIbwh9XpP5Syisf7YoQgsJKPNFoqqLQlu+VQ/
tVSshMR6loPMn8U+dPg==",
      "dev": true,
      "license": "MIT",
      "optional": true,
      "engines": {
        "node": ">=14"
      }
    },
```

```
    "node_modules/@remix-run/router": {
      "version": "1.23.0",
      "resolved": "https://registry.npmjs.org/@remix-run/router/-/
router-1.23.0.tgz",
      "integrity": "sha512-O3rHJzAQKamUz1fvE0Qaw0xSFqsA/
yafi2iqeE0pvdFtCO1viYx8QL6f3Ln/aCCTLxs68SLf0KPM9eSeM8yBnA==",
      "license": "MIT",
      "engines": {
        "node": ">=14.0.0"
      }
    },
    "node_modules/@rollup/rollup-android-arm-eabi": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-android-arm-eabi/-/
rollup-android-arm-eabi-4.40.1.tgz",
      "integrity": "sha512-
kxz0YeeCrRUHz3zyqvd7n+TVRlNyTifBsmnmNPtk3hQURUyG9eAB+usz6DAwagMusjx/
zb3AjvDUvhFGDAexGw==",
      "cpu": [
        "arm"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "android"
      ]
    },
    "node_modules/@rollup/rollup-android-arm64": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-android-arm64/-/
rollup-android-arm64-4.40.1.tgz",
      "integrity": "sha512-PPkxTOisoNC6TpnDKatjKkjRMsdaWIhyuMkA4UsBXT9WEZY4uHezBTj
s6Vl4PbqQQeu6oION1w2voYZv9yquCw==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "android"
      ]
    },
    "node_modules/@rollup/rollup-darwin-arm64": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-darwin-arm64/-/
rollup-darwin-arm64-4.40.1.tgz",
      "integrity": "sha512-VWXGISWFY18v/0JyNUy4A46KCFCb9NVsH+1100XP31lud+TzlezBbz2
4CYzbnA4x6w4hx+NYCXDfnvDVO6lcAA==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "darwin"
      ]
```

```
    },
    "node_modules/@rollup/rollup-darwin-x64": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-darwin-x64/-/rollup-
darwin-x64-4.40.1.tgz",
      "integrity": "sha512-nIwkXafAI1/QCS7pxSpv/
ZtFW6TXcNUEHAIA9EIyw5OzxJZQ1YDrX+CL6JAIQgZ33CInl1R6mHet9Y/UZTg2Bw==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "darwin"
      ]
    },
    "node_modules/@rollup/rollup-freebsd-arm64": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-freebsd-arm64/-/
rollup-freebsd-arm64-4.40.1.tgz",
      "integrity": "sha512-BdrLJ2mHTrIYdaS2I99mriyJfGGenSaP+UwGi1kB9BLOCu9SR8Zpbkm
malKIALnRw24kM7qCN0IOm6L0S44iWw==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "freebsd"
      ]
    },
    "node_modules/@rollup/rollup-freebsd-x64": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-freebsd-x64/-/rollup-
freebsd-x64-4.40.1.tgz",
      "integrity": "sha512-VXeo/
puqvCG8JBPNZXZf5Dqq7BzElNJzHRRw3vjBE27WujdzuOPecDPc/
+1DcdcTptNBep3861jNq0mYkT8Z6Q==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "freebsd"
      ]
    },
    "node_modules/@rollup/rollup-linux-arm-gnueabihf": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-arm-
gnueabihf/-/rollup-linux-arm-gnueabihf-4.40.1.tgz",
      "integrity": "sha512-ehSKrewwsESPt1TgSE/
na9nIhWCosfGSFqv7vwEtjyAqZcvbGIg4JAcV7ZEh2tfj/IlfBeZjgOXm35iOOjadcg==",
      "cpu": [
        "arm"
      ],
```

```
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-arm-musleabihf": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-arm-
musleabihf/-/rollup-linux-arm-musleabihf-4.40.1.tgz",
      "integrity": "sha512-m39iO/aaurh5FVIu/F4/
Zsl8xppd76S4qoID8E+dSRQvTyZTOI2gVk3T4oqzfq1PtcvOfAVlwLMK3KRQMaR8lg==",
      "cpu": [
        "arm"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-arm64-gnu": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-arm64-gnu/-/
rollup-linux-arm64-gnu-4.40.1.tgz",
      "integrity": "sha512-Y+GHnGaku4aVLSgrT0uWe2o2Rq8te9hi+MwqGF9r9ORgXhmHK5Q71N7
57u0F8yU1OIwUIFy6YiJtKjtyktk5hg==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-arm64-musl": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-arm64-musl/-/
rollup-linux-arm64-musl-4.40.1.tgz",
      "integrity": "sha512-jEwjn3jCA+tQGswK3aEWcD09/7M5wGwc6+flhva7dsQNRZZTe30vkal
gIzV4tjkopsTS9Jd7Y1Bsj6a4lzz8gQ==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-loongarch64-gnu": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-loongarch64-
gnu/-/rollup-linux-loongarch64-gnu-4.40.1.tgz",
```

```
      "integrity": "sha512-ySyWikVhNzv+BV/
IDCsrraOAZ3UaC8SZB67FZlqVwXwnFhPihOso9rPOxzZbjp81suB1O2Topw+6Ug3JNegejQ==",
      "cpu": [
        "loong64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-powerpc64le-gnu": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-powerpc64le-
gnu/-/rollup-linux-powerpc64le-gnu-4.40.1.tgz",
      "integrity": "sha512-BvvA64QxZlh7WZWqDPPdt0GH4bznuL6uOO1pmgPnnv86rpUpc8ZxgZw
cEgXvo02GRIZX1hQ0j0pAnhwkhwPqWg==",
      "cpu": [
        "ppc64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-riscv64-gnu": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-riscv64-gnu/-/
rollup-linux-riscv64-gnu-4.40.1.tgz",
      "integrity": "sha512-EQSP+8+1VuSulm9RKSMKitTav89fKbHymTf25n5+Yr6gAPZxYWpj3Dz
AsQqoaHAk9YX2lwEyAf9S4W8F4l3VBQ==",
      "cpu": [
        "riscv64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-riscv64-musl": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-riscv64-musl/-/
rollup-linux-riscv64-musl-4.40.1.tgz",
      "integrity": "sha512-n/vQ4xRZXKuIpqukkMXZt9RWdl+2zgGNx7Uda8NtmLJ06NL8jiHxUaw
bwC+hdSq1rrw/9CghCpEONor+l1e2gA==",
      "cpu": [
        "riscv64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
```

```
    },
    "node_modules/@rollup/rollup-linux-s390x-gnu": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-s390x-gnu/-/
rollup-linux-s390x-gnu-4.40.1.tgz",
      "integrity": "sha512-h8d28xzYb98fMQKUz0w2fMc1XuGzLLjdyxVIbhbil4ELfk5/
orZlSTpF/xdI9C8K0I8lCkq+1En2RJsawZekkg==",
      "cpu": [
        "s390x"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-x64-gnu": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-x64-gnu/-/
rollup-linux-x64-gnu-4.40.1.tgz",
      "integrity": "sha512-XiK5z70PEFEFqcNj3/zRSz/qX4bp4QIraTy9QjwJAb/
Z8GM7kVUsD0Uk8maIPeTyPCP03ChdI+VVmJriKYbRHQ==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-linux-x64-musl": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-linux-x64-musl/-/
rollup-linux-x64-musl-4.40.1.tgz",
      "integrity": "sha512-2BRORitq5rQ4Da9blVovzNCMaUlyKrzMSvkVR0D4qPuOy/
+pMCrh1d7o01RATwVy+6Fa1WBw+da7QPeLWU/1mQ==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "linux"
      ]
    },
    "node_modules/@rollup/rollup-win32-arm64-msvc": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-win32-arm64-msvc/-/
rollup-win32-arm64-msvc-4.40.1.tgz",
      "integrity": "sha512-
b2bcNm9Kbde03H+q+Jjw9tSfhYkzrDUf2d5MAd1bOJuVplXvFhWz7tRtWvD8/
ORZi7qSCy0idW6tf2HgxSXQSg==",
      "cpu": [
        "arm64"
      ],
```

```
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
      ]
    },
    "node_modules/@rollup/rollup-win32-ia32-msvc": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-win32-ia32-msvc/-/
rollup-win32-ia32-msvc-4.40.1.tgz",
      "integrity": "sha512-
DfcogW8N7Zg7llVEfpqWMZcaErKfsj9VvmfSyRjCyo4BI3wPEfrzTtJkZG6gKP/Z92wFm6rz2aDO7/
JfiR/whA==",
      "cpu": [
        "ia32"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
      ]
    },
    "node_modules/@rollup/rollup-win32-x64-msvc": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/@rollup/rollup-win32-x64-msvc/-/
rollup-win32-x64-msvc-4.40.1.tgz",
      "integrity": "sha512-
ECyOuDeH3C1I8jH2MK1RtBJW+YPMvSfT0a5NN0nHfQYnDSJ6tUiZH3gzwVP5/Kfh/
+Tt7tpWVF9LXNTnhTJ3kA==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "win32"
      ]
    },
    "node_modules/@socket.io/component-emitter": {
      "version": "3.1.2",
      "resolved": "https://registry.npmjs.org/@socket.io/component-emitter/-/
component-emitter-3.1.2.tgz",
      "integrity": "sha512-9BCxFwvbGg/
RsZK9tjXd8s4UcwR0MWeFQ1XEKIQVVvAGJyINdrqKMcTRyLoK8Rse1GjzLV9cwjWV1olXRWEXVA==",
      "license": "MIT"
    },
    "node_modules/@types/babel__core": {
      "version": "7.20.5",
      "resolved": "https://registry.npmjs.org/@types/babel__core/-/
babel__core-7.20.5.tgz",
      "integrity": "sha512-qoQprZvz5wQFJwMDqeseRXWv3rqMvhgpbXFfVyWhbx9X47POIA6i/
+dXefEmZKoAgOaTdaIgNSMqMIU61yRyzA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/parser": "^7.20.7",
```

```
        "@babel/types": "^7.20.7",
        "@types/babel__generator": "*",
        "@types/babel__template": "*",
        "@types/babel__traverse": "*"
      }
    },
    "node_modules/@types/babel__generator": {
      "version": "7.27.0",
      "resolved": "https://registry.npmjs.org/@types/babel__generator/-/
babel__generator-7.27.0.tgz",
      "integrity": "sha512-ufFd2Xi92OAVPYsy+P4n7/
U7e68fex0+Ee8gSG9KX7eo084CWiQ4sdxktvdl0bOPupXtVJPY19zk6EwWqUQ8lg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/types": "^7.0.0"
      }
    },
    "node_modules/@types/babel__template": {
      "version": "7.4.4",
      "resolved": "https://registry.npmjs.org/@types/babel__template/-/
babel__template-7.4.4.tgz",
      "integrity": "sha512-h/NUaSyG5EyxBIp8YRxo4RMe2/
qQgvyowRwVMzhYhBCONbW8PUsg4lkFMrhgZhUe5z3L3MiLDuvyJ/CaPa2A8A==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/parser": "^7.1.0",
        "@babel/types": "^7.0.0"
      }
    },
    "node_modules/@types/babel__traverse": {
      "version": "7.20.7",
      "resolved": "https://registry.npmjs.org/@types/babel__traverse/-/
babel__traverse-7.20.7.tgz",
      "integrity": "sha512-dkO5fhS7+/oos4ciWxyEyjWe48zmG6wbCheo/
G2ZnHx4fs3EU6YC6UM8rk56gAjNJ9P3MTH2jo5jb92/K6wbng==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/types": "^7.20.7"
      }
    },
    "node_modules/@types/estree": {
      "version": "1.0.7",
      "resolved": "https://registry.npmjs.org/@types/estree/-/estree-1.0.7.tgz",
      "integrity": "sha512-w28IoSUCJpidD/
TGviZwwMJckNESJZXFu7NBZ5YJ4mEUnNraUn9Pm8HSZm/jDF1pDWYKspWE7oVphigUPRakIQ==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/@types/json-schema": {
      "version": "7.0.15",
      "resolved": "https://registry.npmjs.org/@types/json-schema/-/json-
schema-7.0.15.tgz",
      "integrity": "sha512-5+fP8P8MFNC+AyZCDxrB2pkZFPGzqQWUzpSeuuVLvm8VMcorNYavBqo
FcxK8bQz4Qsbn4oUEEem4wDLfcysGHA==",
      "dev": true,
      "license": "MIT"
```

```
    },
    "node_modules/@types/node": {
      "version": "22.15.2",
      "resolved": "https://registry.npmjs.org/@types/node/-/node-22.15.2.tgz",
      "integrity": "sha512-uKXqKN9beGoMdBfcaTY1ecwz6ctxuJAcUlwE55938g0ZJ8lRxwAZqRz
2AJ4pzpt5dHdTPMB863UZ0ESiFUcP7A==",
      "dev": true,
      "license": "MIT",
      "optional": true,
      "peer": true,
      "dependencies": {
        "undici-types": "~6.21.0"
      }
    },
    "node_modules/@types/react": {
      "version": "19.1.2",
      "resolved": "https://registry.npmjs.org/@types/react/-/react-19.1.2.tgz",
      "integrity": "sha512-oxLPMytKchWGbnQM9O7D67uPa9paTNxO7jVoNMXgkkErULBPhPARCfk
KL9ytcIJJRGjbsVwW4ugJzyFFvm/Tiw==",
      "devOptional": true,
      "license": "MIT",
      "dependencies": {
        "csstype": "^3.0.2"
      }
    },
    "node_modules/@types/react-dom": {
      "version": "19.1.2",
      "resolved": "https://registry.npmjs.org/@types/react-dom/-/react-
dom-19.1.2.tgz",
      "integrity": "sha512-XGJkWF41Qq305SKWEILa1O8vzhb3aOo3ogBlSmiqNko/
WmRb6QIaweuZCXjKygVDXpzXb5wyxKTSOsmkuqj+Qw==",
      "dev": true,
      "license": "MIT",
      "peerDependencies": {
        "@types/react": "^19.0.0"
      }
    },
    "node_modules/@types/use-sync-external-store": {
      "version": "0.0.6",
      "resolved": "https://registry.npmjs.org/@types/use-sync-external-store/-/
use-sync-external-store-0.0.6.tgz",
      "integrity": "sha512-
zFDAD+tlpf2r4asuHEj0XH6pY6i0g5NeAHPn+15wk3BV6JA69eERFXC1gyGThDkVa1zCyKr5jox1+2LbV/
AMLg==",
      "license": "MIT"
    },
    "node_modules/@vitejs/plugin-react": {
      "version": "4.4.1",
      "resolved": "https://registry.npmjs.org/@vitejs/plugin-react/-/plugin-
react-4.4.1.tgz",
      "integrity": "sha512-IpEm5ZmeXAP/
osiBXVVP5KjFMzbWOonMs0NaQQl+xYnUAcq4oHUBsF2+p4MgKWG4YMmFYJU8A6sxRPuowllm6w==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@babel/core": "^7.26.10",
        "@babel/plugin-transform-react-jsx-self": "^7.25.9",
        "@babel/plugin-transform-react-jsx-source": "^7.25.9",
        "@types/babel__core": "^7.20.5",
```

```
        "react-refresh": "^0.17.0"
      },
      "engines": {
        "node": "^14.18.0 || >=16.0.0"
      },
      "peerDependencies": {
        "vite": "^4.2.0 || ^5.0.0 || ^6.0.0"
      }
    },
    "node_modules/@vitejs/plugin-react/node_modules/react-refresh": {
      "version": "0.17.0",
      "resolved": "https://registry.npmjs.org/react-refresh/-/react-
refresh-0.17.0.tgz",
      "integrity": "sha512-z6F7K9bV85EfseRCp2bzrpyQ0Gkw1uLoCel9XBVWPg/
TjRj94SkJzUTGfOa4bs7iJvBWtQG0Wq7wnI0syw3EBQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/acorn": {
      "version": "8.14.1",
      "resolved": "https://registry.npmjs.org/acorn/-/acorn-8.14.1.tgz",
      "integrity": "sha512-OvQ/2pUDKmgfCg+
+xsTX1wGxfTaszcHVcTctW4UJB4hibJx2HXxxO5UmVgyjMa+ZDsiaf5wWLXYpRWMmBI0QHg==",
      "dev": true,
      "license": "MIT",
      "bin": {
        "acorn": "bin/acorn"
      },
      "engines": {
        "node": ">=0.4.0"
      }
    },
    "node_modules/acorn-jsx": {
      "version": "5.3.2",
      "resolved": "https://registry.npmjs.org/acorn-jsx/-/acorn-jsx-5.3.2.tgz",
      "integrity": "sha512-rq9s+JNhf0IChjtDXxllJ7g41oZk5SlXtp0LHwyA5cejwn7vKmKp4pP
ri6YEePv2PU65sAsegbXtIinmDFDXgQ==",
      "dev": true,
      "license": "MIT",
      "peerDependencies": {
        "acorn": "^6.0.0 || ^7.0.0 || ^8.0.0"
      }
    },
    "node_modules/adler-32": {
      "version": "1.3.1",
      "resolved": "https://registry.npmjs.org/adler-32/-/adler-32-1.3.1.tgz",
      "integrity": "sha512-ynZ4w/nUUv5rrsR8UUGoe1VC9hZj6V5hU9Qw1HlMDJGEJw5S7TfTErW
TjMys6M7vr0YWcPqs3qAr4ss0nDfP+A==",
      "license": "Apache-2.0",
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/ajv": {
      "version": "6.12.6",
      "resolved": "https://registry.npmjs.org/ajv/-/ajv-6.12.6.tgz",
```

```
      "integrity": "sha512-j3fVLgvTo527anyYyJOGTYJbG+vnnQYvE0m5mmkc1TK+nxAppkCLMIL
0aZ4dblVCNoGShhm+kzE4ZUykBoMg4g==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "fast-deep-equal": "^3.1.1",
        "fast-json-stable-stringify": "^2.0.0",
        "json-schema-traverse": "^0.4.1",
        "uri-js": "^4.2.2"
      },
      "funding": {
        "type": "github",
        "url": "https://github.com/sponsors/epoberezkin"
      }
    },
    "node_modules/ansi-regex": {
      "version": "6.1.0",
      "resolved": "https://registry.npmjs.org/ansi-regex/-/ansi-regex-6.1.0.tgz",
      "integrity": "sha512-7HSX4QQb4CspciLpVFwyRe79O3xsIZDDLER21kERQ71oaPodF8jL725
AgJMFAYbooIqolJoRLuM81SpeUkpkvA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=12"
      },
      "funding": {
        "url": "https://github.com/chalk/ansi-regex?sponsor=1"
      }
    },
    "node_modules/ansi-styles": {
      "version": "4.3.0",
      "resolved": "https://registry.npmjs.org/ansi-styles/-/ansi-
styles-4.3.0.tgz",
      "integrity": "sha512-zbB9rCJAT1rbjiVDb2hqKFHNYLxgtk8NURxZ3IZwD3F6NtxbXZQCnnS
i1Lkx+IDohdPlFp222wVALIheZJQSEg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "color-convert": "^2.0.1"
      },
      "engines": {
        "node": ">=8"
      },
      "funding": {
        "url": "https://github.com/chalk/ansi-styles?sponsor=1"
      }
    },
    "node_modules/any-promise": {
      "version": "1.3.0",
      "resolved": "https://registry.npmjs.org/any-promise/-/any-
promise-1.3.0.tgz",
      "integrity": "sha512-7UvmKalWRt1wgjL1RrGxoSJW/0QZFIegpeGvZG9kjp8vrRu55XTHbwn
qq2GpXm9uLbcuhxm3IqX9OB4MZR1b2A==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/anymatch": {
      "version": "3.1.3",
      "resolved": "https://registry.npmjs.org/anymatch/-/anymatch-3.1.3.tgz",
```

```
      "integrity": "sha512-
KMReFUr0B4t+D+OBkjR3KYqvocp2XaSzO55UcB6mgQMd3KbcE+mWTyvVV7D/
zsdEbNnV6acZUutkiHQXvTr1Rw==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "normalize-path": "^3.0.0",
        "picomatch": "^2.0.4"
      },
      "engines": {
        "node": ">= 8"
      }
    },
    "node_modules/aos": {
      "version": "2.3.4",
      "resolved": "https://registry.npmjs.org/aos/-/aos-2.3.4.tgz",
      "integrity": "sha512-zh/ahtR2yME4I51z8IttIt4lC1Nw0ktsFtmeDzID1m9naJnWXhCoARa
CgNOGXb5CLy3zm+wqmRAEgMYB5E2HUw==",
      "license": "MIT",
      "dependencies": {
        "classlist-polyfill": "^1.0.3",
        "lodash.debounce": "^4.0.6",
        "lodash.throttle": "^4.0.1"
      }
    },
    "node_modules/arg": {
      "version": "5.0.2",
      "resolved": "https://registry.npmjs.org/arg/-/arg-5.0.2.tgz",
      "integrity": "sha512-PYjyFOLKQ9y57JvQ6QLo8dAgNqswh8M1RMJYdQduT6xbWSgK36P/Z/
v+p888pM69jMMfS8Xd8F6I1kQ/I9HUGg==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/asynckit": {
      "version": "0.4.0",
      "resolved": "https://registry.npmjs.org/asynckit/-/asynckit-0.4.0.tgz",
      "integrity": "sha512-Oei9OH4tRh0YqU3GxhX79dM/
mwVgvbZJaSNaRk+bshkj0S5cfHcgYakreBjrHwatXKbz+IoIdYLxrKim2MjW0Q==",
      "license": "MIT"
    },
    "node_modules/autoprefixer": {
      "version": "10.4.21",
      "resolved": "https://registry.npmjs.org/autoprefixer/-/
autoprefixer-10.4.21.tgz",
      "integrity": "sha512-O+A6LWV5LDHSJD3LjHYoNi4VLsj/
Whi7k6zG12xTYaU4cQ8oxQGckXNX8cRHK5yOZ/ppVHe0ZBXGzSV9jXdVbQ==",
      "dev": true,
      "funding": [
        {
          "type": "opencollective",
          "url": "https://opencollective.com/postcss/"
        },
        {
          "type": "tidelift",
          "url": "https://tidelift.com/funding/github/npm/autoprefixer"
        },
        {
          "type": "github",
          "url": "https://github.com/sponsors/ai"
```

```
        }
      ],
      "license": "MIT",
      "dependencies": {
        "browserslist": "^4.24.4",
        "caniuse-lite": "^1.0.30001702",
        "fraction.js": "^4.3.7",
        "normalize-range": "^0.1.2",
        "picocolors": "^1.1.1",
        "postcss-value-parser": "^4.2.0"
      },
      "bin": {
        "autoprefixer": "bin/autoprefixer"
      },
      "engines": {
        "node": "^10 || ^12 || >=14"
      },
      "peerDependencies": {
        "postcss": "^8.1.0"
      }
    },
    "node_modules/axios": {
      "version": "1.9.0",
      "resolved": "https://registry.npmjs.org/axios/-/axios-1.9.0.tgz",
      "integrity": "sha512-re4CqKTJaURpzbLHtIi6XpDv20/CnpXOtjRY5/
CU32L8gU8ek9UIivcfvSWvmKEngmVbrUtPpdDwWDWL7DNHvg==",
      "license": "MIT",
      "dependencies": {
        "follow-redirects": "^1.15.6",
        "form-data": "^4.0.0",
        "proxy-from-env": "^1.1.0"
      }
    },
    "node_modules/balanced-match": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/balanced-match/-/balanced-
match-1.0.2.tgz",
      "integrity": "sha512-3oSeUO0TMV67hN1AmbXsK4yaqU7tjiHlbxRDZOpH0KW9+CeX4bRAaX0
Anxt0tx2MrpRpWwQaPwIlISEJhYU5Pw==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/binary-extensions": {
      "version": "2.3.0",
      "resolved": "https://registry.npmjs.org/binary-extensions/-/binary-
extensions-2.3.0.tgz",
      "integrity": "sha512-Ceh+7ox5qe7LJuLHoY0feh3pHuUDHAcRUeyL2VYghZwfpkNIy/
+8Ocg0a3UuSoYzavmylwuLWQOf3hl0jjMMIw==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/brace-expansion": {
      "version": "1.1.11",
```

```
      "resolved": "https://registry.npmjs.org/brace-expansion/-/brace-
expansion-1.1.11.tgz",
      "integrity": "sha512-iCuPHDFgrHX7H2vEI/5xpz07zSHB00TpugqhmYtVmMO6518mCuRMoOY
FldEBl0g187ufozdaHgWKcYFb61qGiA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "balanced-match": "^1.0.0",
        "concat-map": "0.0.1"
      }
    },
    "node_modules/braces": {
      "version": "3.0.3",
      "resolved": "https://registry.npmjs.org/braces/-/braces-3.0.3.tgz",
      "integrity": "sha512-yQbXgO/
OSZVD2IsiLlro+7Hf6Q18EJrKSEsdoMzKePKXct3gvD8oLcOQdIzGupr5Fj+EDe8gO/
lxc1BzfMpxvA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "fill-range": "^7.1.1"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/browserslist": {
      "version": "4.24.4",
      "resolved": "https://registry.npmjs.org/browserslist/-/
browserslist-4.24.4.tgz",
      "integrity": "sha512-
KDi1Ny1gSePi1vm0q4oxSF8b4DR44GF4BbmS2YdhPLOEqd8pDviZOGH/
GsmRwoWJ2+5Lr085X7naowMwKHDG1A==",
      "dev": true,
      "funding": [
        {
          "type": "opencollective",
          "url": "https://opencollective.com/browserslist"
        },
        {
          "type": "tidelift",
          "url": "https://tidelift.com/funding/github/npm/browserslist"
        },
        {
          "type": "github",
          "url": "https://github.com/sponsors/ai"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "caniuse-lite": "^1.0.30001688",
        "electron-to-chromium": "^1.5.73",
        "node-releases": "^2.0.19",
        "update-browserslist-db": "^1.1.1"
      },
      "bin": {
        "browserslist": "cli.js"
      },
      "engines": {
```

```
        "node": "^6 || ^7 || ^8 || ^9 || ^10 || ^11 || ^12 || >=13.7"
      }
    },
    "node_modules/buffer-from": {
      "version": "1.1.2",
      "resolved": "https://registry.npmjs.org/buffer-from/-/buffer-
from-1.1.2.tgz",
      "integrity": "sha512-E+XQCRwSbaaiChtv6k6Dwgc+bx+Bs6vuKJHHl5kox/
BaKbhiXzqQOwK4cO22yElGp2OCmjwVhT3HmxgyPGnJfQ==",
      "dev": true,
      "license": "MIT",
      "optional": true,
      "peer": true
    },
    "node_modules/call-bind-apply-helpers": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/call-bind-apply-helpers/-/call-bind-
apply-helpers-1.0.2.tgz",
      "integrity": "sha512-Sp1ablJ0ivDkSzjcaJdxEunN5/XvksFJ2sMBFfq6x0ryhQV/2b/
KwFe21cMpmHtPOSij8K99/wSfoEuTObmuMQ==",
      "license": "MIT",
      "dependencies": {
        "es-errors": "^1.3.0",
        "function-bind": "^1.1.2"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/callsites": {
      "version": "3.1.0",
      "resolved": "https://registry.npmjs.org/callsites/-/callsites-3.1.0.tgz",
      "integrity": "sha512-P8BjAsXvZS+VIDUI11hHCQEv74YT67YUi5JJFNWIqL235sBmjX4+qx9
Muvls5ivyNENctx46xQLQ3aTuE7ssaQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/camelcase-css": {
      "version": "2.0.1",
      "resolved": "https://registry.npmjs.org/camelcase-css/-/camelcase-
css-2.0.1.tgz",
      "integrity": "sha512-QOSvevhslijgYwRx6Rv7zKdMF8lbRmx+uQGx2+vDc+KI/
eBnsy9kit5aj23AgGu3pa4t9AgwbnXWqS+iOY+2aA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">= 6"
      }
    },
    "node_modules/caniuse-lite": {
      "version": "1.0.30001715",
      "resolved": "https://registry.npmjs.org/caniuse-lite/-/caniuse-
lite-1.0.30001715.tgz",
      "integrity": "sha512-7ptkFGMm2OAOgvZpwgA4yjQ5SQbrNVGdRjzH0pBdy1Fasvcr+KAeECm
bCAECzTuDuoX0FCY8KzUxjf9+9kfZEw==",
      "dev": true,
```

```
      "funding": [
        {
          "type": "opencollective",
          "url": "https://opencollective.com/browserslist"
        },
        {
          "type": "tidelift",
          "url": "https://tidelift.com/funding/github/npm/caniuse-lite"
        },
        {
          "type": "github",
          "url": "https://github.com/sponsors/ai"
        }
      ],
      "license": "CC-BY-4.0"
    },
    "node_modules/cfb": {
      "version": "1.2.2",
      "resolved": "https://registry.npmjs.org/cfb/-/cfb-1.2.2.tgz",
      "integrity": "sha512-KfdUZsSOw19/ObEWasvBP/Ac4reZvAGauZhs6S/
gqNhXhI7cKwvlH7ulj+dOEYnca4bm4SGo8C1bTAQvnTjgQA==",
      "license": "Apache-2.0",
      "dependencies": {
        "adler-32": "~1.3.0",
        "crc-32": "~1.2.0"
      },
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/chalk": {
      "version": "4.1.2",
      "resolved": "https://registry.npmjs.org/chalk/-/chalk-4.1.2.tgz",
      "integrity": "sha512-oKnbhFyRIXpUuez8iBMmyEa4nbj4IOQyuhc/wy9kY7/
WVPcwIO9VA668Pu8RkO7+0G76SLROeyw9CpQ061i4mA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ansi-styles": "^4.1.0",
        "supports-color": "^7.1.0"
      },
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/chalk/chalk?sponsor=1"
      }
    },
    "node_modules/chart.js": {
      "version": "4.4.9",
      "resolved": "https://registry.npmjs.org/chart.js/-/chart.js-4.4.9.tgz",
      "integrity": "sha512-EyZ9wWKgpAU0fLJ43YAEIF8sr5F2W3LqbS40ZJyHIner2lY14ufqv2V
Mp69MAiZ2rpwxEUxEhIH/0U3xyRynxg==",
      "license": "MIT",
      "dependencies": {
        "@kurkle/color": "^0.3.0"
      },
      "engines": {
        "pnpm": ">=8"
```

```
      }
    },
    "node_modules/chokidar": {
      "version": "3.6.0",
      "resolved": "https://registry.npmjs.org/chokidar/-/chokidar-3.6.0.tgz",
      "integrity": "sha512-7VT13fmjotKpGipCW9JEQAusEPE+Ei8nl6/
g4FBAmIm0GOOLMua9NDDo/DWp0ZAxCr3cPq5ZpBqmPAQgDda2Pw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "anymatch": "~3.1.2",
        "braces": "~3.0.2",
        "glob-parent": "~5.1.2",
        "is-binary-path": "~2.1.0",
        "is-glob": "~4.0.1",
        "normalize-path": "~3.0.0",
        "readdirp": "~3.6.0"
      },
      "engines": {
        "node": ">= 8.10.0"
      },
      "funding": {
        "url": "https://paulmillr.com/funding/"
      },
      "optionalDependencies": {
        "fsevents": "~2.3.2"
      }
    },
    "node_modules/chokidar/node_modules/glob-parent": {
      "version": "5.1.2",
      "resolved": "https://registry.npmjs.org/glob-parent/-/glob-
parent-5.1.2.tgz",
      "integrity": "sha512-
AOIgSQCepiJYwP3ARnGx+5VnTu2HBYdzbGP45eLw1vr3zB3vZLeyed1sC9hnbcOc9/
SrMyM5RPQrkGz4aS9Zow==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "is-glob": "^4.0.1"
      },
      "engines": {
        "node": ">= 6"
      }
    },
    "node_modules/classlist-polyfill": {
      "version": "1.2.0",
      "resolved": "https://registry.npmjs.org/classlist-polyfill/-/classlist-
polyfill-1.2.0.tgz",
      "integrity": "sha512-GzIjNdcEtH4ieA2S8NmrSxv7DfEV5fmixQeyTmqmRmRJPGpRBaSnA2a
0VrCjyT8iW8JjEdMbKzDotAJf+ajgaQ==",
      "license": "Unlicense"
    },
    "node_modules/codepage": {
      "version": "1.15.0",
      "resolved": "https://registry.npmjs.org/codepage/-/codepage-1.15.0.tgz",
      "integrity": "sha512-3g6NUTPd/YtuuGrhMnOMRjFc+LJw/bnMp3+0r/
Wcz3IXUuCosKRJvMphm5+Q+bvTVGcJJuRvVLuYba+WojaFaA==",
      "license": "Apache-2.0",
      "engines": {
```

```
        "node": ">=0.8"
      }
    },
    "node_modules/color-convert": {
      "version": "2.0.1",
      "resolved": "https://registry.npmjs.org/color-convert/-/color-
convert-2.0.1.tgz",
      "integrity": "sha512-RRECPsj7iu/
xb5oKYcsFHSppFNnsj/52OVTRKb4zP5onXwVF3zVmmToNcOfGC+CRDpfK/U584fMg38ZHCaElKQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "color-name": "~1.1.4"
      },
      "engines": {
        "node": ">=7.0.0"
      }
    },
    "node_modules/color-name": {
      "version": "1.1.4",
      "resolved": "https://registry.npmjs.org/color-name/-/color-name-1.1.4.tgz",
      "integrity": "sha512-dOy+3AuW3a2wNbZHIuMZpTcgjGuLU/uBL/
ubcZF9OXbDo8ff4O8yVp5Bf0efS8uEoYo5q4Fx7dY9OgQGXgAsQA==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/combined-stream": {
      "version": "1.0.8",
      "resolved": "https://registry.npmjs.org/combined-stream/-/combined-
stream-1.0.8.tgz",
      "integrity": "sha512-
FQN4MRfuJeHf7cBbBMJFXhKSDq+2kAArBlmRBvcvFE5BB1HZKXtSFASDhdlz9zOYwxh8lDdnvmMOe/
+5cdoEdg==",
      "license": "MIT",
      "dependencies": {
        "delayed-stream": "~1.0.0"
      },
      "engines": {
        "node": ">= 0.8"
      }
    },
    "node_modules/commander": {
      "version": "4.1.1",
      "resolved": "https://registry.npmjs.org/commander/-/commander-4.1.1.tgz",
      "integrity": "sha512-NOKm8xhkzAjzFx8B2v5OAHT+u5pRQc2UCa2Vq9jYL/31o2wi9mxBA7L
IFs3sV5VSC49z6pEhfbMULvShKj26WA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">= 6"
      }
    },
    "node_modules/concat-map": {
      "version": "0.0.1",
      "resolved": "https://registry.npmjs.org/concat-map/-/concat-map-0.0.1.tgz",
      "integrity": "sha512-/Srv4dswyQNBfohGpz9o6Yb3Gz3SrUDqBH5rTuhGR7ahtlbYKnVxw2b
CFMRljaA7EXHaXZ8wsHdodFvbkhKmqg==",
      "dev": true,
      "license": "MIT"
```

```
    },
    "node_modules/convert-source-map": {
      "version": "2.0.0",
      "resolved": "https://registry.npmjs.org/convert-source-map/-/convert-source-
map-2.0.0.tgz",
      "integrity": "sha512-Kvp459HrV2FEJ1CAsi1Ku+MY3kasH19TFykTz2xWmMeq6bk2NU3XXvf
J+Q61m0xktWwt+1HSYf3JZsTms3aRJg==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/crc-32": {
      "version": "1.2.2",
      "resolved": "https://registry.npmjs.org/crc-32/-/crc-32-1.2.2.tgz",
      "integrity": "sha512-
ROmzCKrTnOwybPcJApAA6WBWij23HVfGVNKqqrZpuyZOHqK2CwHSvpGuyt/
UNNvaIjEd8X5IFGp4Mh+Ie1IHJQ==",
      "license": "Apache-2.0",
      "bin": {
        "crc32": "bin/crc32.njs"
      },
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/cross-spawn": {
      "version": "7.0.6",
      "resolved": "https://registry.npmjs.org/cross-spawn/-/cross-
spawn-7.0.6.tgz",
      "integrity": "sha512-uV2QOWP2nWzsy2aMp8aRibhi9dlzF5Hgh5SHaB9OiTGEyDTiJJyx0uy
51QXdyWbtAHNua4XJzUKca3OzKUd3vA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "path-key": "^3.1.0",
        "shebang-command": "^2.0.0",
        "which": "^2.0.1"
      },
      "engines": {
        "node": ">= 8"
      }
    },
    "node_modules/cssesc": {
      "version": "3.0.0",
      "resolved": "https://registry.npmjs.org/cssesc/-/cssesc-3.0.0.tgz",
      "integrity": "sha512-/Tb/JcjK111nNScGob5MNtsntNM1aCNUDipB/
TkwZFhyDrrE47SOx/18wF2bbjgc3ZzCSKW1T5nt5EbFoAz/Vg==",
      "dev": true,
      "license": "MIT",
      "bin": {
        "cssesc": "bin/cssesc"
      },
      "engines": {
        "node": ">=4"
      }
    },
    "node_modules/csstype": {
      "version": "3.1.3",
      "resolved": "https://registry.npmjs.org/csstype/-/csstype-3.1.3.tgz",
      "integrity": "sha512-M1uQkMl8rQK/
```

```
szD0LNhtqxIPLpimGm8sOBwU7lLnCpSbTyY3yeU1Vc7l4KT5zT4s/yOxHH5O7tIuuLOCnLADRw==",
      "license": "MIT"
    },
    "node_modules/date-fns": {
      "version": "4.1.0",
      "resolved": "https://registry.npmjs.org/date-fns/-/date-fns-4.1.0.tgz",
      "integrity": "sha512-Ukq0owbQXxa/U3EGtsdVBkR1w7KOQ5gIBqdH2hkvknzZPYvBxb/
aa6E8L7tmjFtkwZBu3UXBbjIgPo/Ez4xaNg==",
      "license": "MIT",
      "funding": {
        "type": "github",
        "url": "https://github.com/sponsors/kossnocorp"
      }
    },
    "node_modules/debug": {
      "version": "4.4.0",
      "resolved": "https://registry.npmjs.org/debug/-/debug-4.4.0.tgz",
      "integrity": "sha512-6WTZ/IxCY/T6BALoZHaE4ctp9xm+Z5kY/
pzYaCHRFeyVhojxlrm+46y68HA6hr0TcwEssoxNiDEUJQjfPZ/RYA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ms": "^2.1.3"
      },
      "engines": {
        "node": ">=6.0"
      },
      "peerDependenciesMeta": {
        "supports-color": {
          "optional": true
        }
      }
    },
    "node_modules/deep-is": {
      "version": "0.1.4",
      "resolved": "https://registry.npmjs.org/deep-is/-/deep-is-0.1.4.tgz",
      "integrity": "sha512-oIPzksmTg4/MriiaYGO+okXDT7ztn/w3Eptv/
+gSIdMdKsJo0u4CfYNFJPy+4SKMuCqGw2wxnA+URMg3t8a/bQ==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/delayed-stream": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/delayed-stream/-/delayed-
stream-1.0.0.tgz",
      "integrity": "sha512-ZySD7Nf91aLB0RxL4KGrKHBXl7Eds1DAmEdcoVawXnLD7SDhpNgtuII
2aAkg7a7QS41jxPSZ17p4VdGnMHk3MQ==",
      "license": "MIT",
      "engines": {
        "node": ">=0.4.0"
      }
    },
    "node_modules/detect-libc": {
      "version": "2.0.4",
      "resolved": "https://registry.npmjs.org/detect-libc/-/detect-
libc-2.0.4.tgz",
      "integrity": "sha512-3UDv+G9CsCKO1WKMGw9fwq/
SWJYbI0c5Y7LU1AXYoDdbhE2AHQ6N6Nb34sG8Fj7T5APy8qXDCKuuIHd1BR0tVA==",
      "dev": true,
```

```
      "license": "Apache-2.0",
      "optional": true,
      "peer": true,
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/didyoumean": {
      "version": "1.2.2",
      "resolved": "https://registry.npmjs.org/didyoumean/-/didyoumean-1.2.2.tgz",
      "integrity": "sha512-gxtyfqMg7GKyhQmb056K7M3xszy/
myH8w+B4RT+QXBQsvAOdc3XymqDDPHx1BgPgsdAA5SIifona89YtRATDzw==",
      "dev": true,
      "license": "Apache-2.0"
    },
    "node_modules/dlv": {
      "version": "1.1.3",
      "resolved": "https://registry.npmjs.org/dlv/-/dlv-1.1.3.tgz",
      "integrity": "sha512-+HlytyjlPKnIG8XuRG8WvmBP8xs8P71y+SKKS6ZXWoEgLuePxtDoUEi
H7WkdePWrQ5JBpE6aoVqfZfJUQkjXwA==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/dunder-proto": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/dunder-proto/-/dunder-
proto-1.0.1.tgz",
      "integrity": "sha512-KIN/nDJBQRcXw0MLVhZE9iQHmG68qAVIBg9CqmUYjmQIhgij9U5MFvr
qkUL5FbtyyzZuOeOt0zdeRe4UY7ct+A==",
      "license": "MIT",
      "dependencies": {
        "call-bind-apply-helpers": "^1.0.1",
        "es-errors": "^1.3.0",
        "gopd": "^1.2.0"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/eastasianwidth": {
      "version": "0.2.0",
      "resolved": "https://registry.npmjs.org/eastasianwidth/-/
eastasianwidth-0.2.0.tgz",
      "integrity": "sha512-I88TYZWc9XiYHRQ4/3c5rjjfgkjhLyW2luGIheGERbNQ6OY7yTybanS
pDXZa8y7VUP9YmDcYa+eyq4ca7iLqWA==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/electron-to-chromium": {
      "version": "1.5.143",
      "resolved": "https://registry.npmjs.org/electron-to-chromium/-/electron-to-
chromium-1.5.143.tgz",
      "integrity": "sha512-
QqklJMOFBMqe46k8iIOwA9l2hz57V2OKMmP5eSWcUvwx+mASAsbU+wkF1pHjn9ZVSBPrsYWr4/
W/95y5SwYg2g==",
      "dev": true,
      "license": "ISC"
    },
    "node_modules/emoji-regex": {
```

```
      "version": "9.2.2",
      "resolved": "https://registry.npmjs.org/emoji-regex/-/emoji-
regex-9.2.2.tgz",
      "integrity": "sha512-L18DaJsXSUk2+42pv8mLs5jJT2hqFkFE4j21wOmgbUqsZ2hL72NsUU7
85g9RXgo3s0ZNgVl42TiHp3ZtOv/Vyg==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/engine.io-client": {
      "version": "6.6.3",
      "resolved": "https://registry.npmjs.org/engine.io-client/-/engine.io-
client-6.6.3.tgz",
      "integrity": "sha512-T0iLjnyNWahNyv/lcjS2y4oE358tVS/SYQNxYXGAJ9/GLgH4VCvOQ/
mhTjqU88mLZCQgiG8RIegFHYCdVC+j5w==",
      "license": "MIT",
      "dependencies": {
        "@socket.io/component-emitter": "~3.1.0",
        "debug": "~4.3.1",
        "engine.io-parser": "~5.2.1",
        "ws": "~8.17.1",
        "xmlhttprequest-ssl": "~2.1.1"
      }
    },
    "node_modules/engine.io-client/node_modules/debug": {
      "version": "4.3.7",
      "resolved": "https://registry.npmjs.org/debug/-/debug-4.3.7.tgz",
      "integrity": "sha512-Er2nc/
H7RrMXZBFCEim6TCmMk02Z8vLC2Rbi1KEBggpo0fS6l0S1nnapwmIi3yW/
+GOJap1Krg4w0Hg80oCqgQ==",
      "license": "MIT",
      "dependencies": {
        "ms": "^2.1.3"
      },
      "engines": {
        "node": ">=6.0"
      },
      "peerDependenciesMeta": {
        "supports-color": {
          "optional": true
        }
      }
    },
    "node_modules/engine.io-parser": {
      "version": "5.2.3",
      "resolved": "https://registry.npmjs.org/engine.io-parser/-/engine.io-
parser-5.2.3.tgz",
      "integrity": "sha512-
HqD3yTBfnBxIrbnM1DoD6Pcq8NECnh8d4As1Qgh0z5Gg3jRRIqijury0CL3ghu/
edArpUYiYqQiDUQBIs4np3Q==",
      "license": "MIT",
      "engines": {
        "node": ">=10.0.0"
      }
    },
    "node_modules/es-define-property": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/es-define-property/-/es-define-
property-1.0.1.tgz",
      "integrity": "sha512-e3nRfgfUZ4rNGL232gUgX06QNyyez04KdjFrF+LTRoOXmrOgFKDg4BC
```

```
dsjW8EnT69eqdYGmRpJwiPVYNrCaW3g==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/es-errors": {
      "version": "1.3.0",
      "resolved": "https://registry.npmjs.org/es-errors/-/es-errors-1.3.0.tgz",
      "integrity": "sha512-Zf5H2Kxt2xjTvbJvP2ZWLEICxA6j+hAmMzIlypy4xcBg1vKVnx89Wy0
GbS+kf5cwCVFFzdCFh2XSCFNULS6csw==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/es-object-atoms": {
      "version": "1.1.1",
      "resolved": "https://registry.npmjs.org/es-object-atoms/-/es-object-
atoms-1.1.1.tgz",
      "integrity": "sha512-FGgH2h8zKNim9ljj7dankFPcICIK9Cp5bm+c2gQSYePhpaG5+esrLOD
ihIorn+Pe6FGJzWhXQotPv73jTaldXA==",
      "license": "MIT",
      "dependencies": {
        "es-errors": "^1.3.0"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/es-set-tostringtag": {
      "version": "2.1.0",
      "resolved": "https://registry.npmjs.org/es-set-tostringtag/-/es-set-
tostringtag-2.1.0.tgz",
      "integrity": "sha512-j6vWzfrGVfyXxge+O0x5sh6cvxAog0a/4Rdd2K36zCMV5eJ+/
+tOAngRO8cODMNWbVRdVlmGZQL2YS3yR8bIUA==",
      "license": "MIT",
      "dependencies": {
        "es-errors": "^1.3.0",
        "get-intrinsic": "^1.2.6",
        "has-tostringtag": "^1.0.2",
        "hasown": "^2.0.2"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/esbuild": {
      "version": "0.25.3",
      "resolved": "https://registry.npmjs.org/esbuild/-/esbuild-0.25.3.tgz",
      "integrity": "sha512-qKA6Pvai73+M2FtftpNKRxJ78GIjmFXFxd/1DVBqGo/
qNhLSfv+G12n9pNoWdytJC8U00TrViOwpjT0zgqQS8Q==",
      "dev": true,
      "hasInstallScript": true,
      "license": "MIT",
      "bin": {
        "esbuild": "bin/esbuild"
      },
      "engines": {
```

```
        "node": ">=18"
      },
      "optionalDependencies": {
        "@esbuild/aix-ppc64": "0.25.3",
        "@esbuild/android-arm": "0.25.3",
        "@esbuild/android-arm64": "0.25.3",
        "@esbuild/android-x64": "0.25.3",
        "@esbuild/darwin-arm64": "0.25.3",
        "@esbuild/darwin-x64": "0.25.3",
        "@esbuild/freebsd-arm64": "0.25.3",
        "@esbuild/freebsd-x64": "0.25.3",
        "@esbuild/linux-arm": "0.25.3",
        "@esbuild/linux-arm64": "0.25.3",
        "@esbuild/linux-ia32": "0.25.3",
        "@esbuild/linux-loong64": "0.25.3",
        "@esbuild/linux-mips64el": "0.25.3",
        "@esbuild/linux-ppc64": "0.25.3",
        "@esbuild/linux-riscv64": "0.25.3",
        "@esbuild/linux-s390x": "0.25.3",
        "@esbuild/linux-x64": "0.25.3",
        "@esbuild/netbsd-arm64": "0.25.3",
        "@esbuild/netbsd-x64": "0.25.3",
        "@esbuild/openbsd-arm64": "0.25.3",
        "@esbuild/openbsd-x64": "0.25.3",
        "@esbuild/sunos-x64": "0.25.3",
        "@esbuild/win32-arm64": "0.25.3",
        "@esbuild/win32-ia32": "0.25.3",
        "@esbuild/win32-x64": "0.25.3"
      }
    },
    "node_modules/escalade": {
      "version": "3.2.0",
      "resolved": "https://registry.npmjs.org/escalade/-/escalade-3.2.0.tgz",
      "integrity": "sha512-WUj2qlxaQtO4g6Pq5c29GTcWGDyd8itL8zTlipgECz3JesAiiOKotd8
JU6otB3PACgG6xkJUyVhboMS+bje/jA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/escape-string-regexp": {
      "version": "4.0.0",
      "resolved": "https://registry.npmjs.org/escape-string-regexp/-/escape-
string-regexp-4.0.0.tgz",
      "integrity": "sha512-TtpcNJ3XAzx3Gq8sWRzJaVajRs0uVxA2YAkdb1jm2YkPz4G6egUFAyA
3n5vtEIZefPk5Wa4UXbKuS5fKkJWdgA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/eslint": {
      "version": "9.25.1",
      "resolved": "https://registry.npmjs.org/eslint/-/eslint-9.25.1.tgz",
```

```
      "integrity": "sha512-E6Mtz9oGQWDCpV12319d59n4tx9zOTXSTmc8BLVxBx+G/0RdM5MvEEJ
LU9c0+aleoePYYgVTOsRblx433qmhWQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@eslint-community/eslint-utils": "^4.2.0",
        "@eslint-community/regexpp": "^4.12.1",
        "@eslint/config-array": "^0.20.0",
        "@eslint/config-helpers": "^0.2.1",
        "@eslint/core": "^0.13.0",
        "@eslint/eslintrc": "^3.3.1",
        "@eslint/js": "9.25.1",
        "@eslint/plugin-kit": "^0.2.8",
        "@humanfs/node": "^0.16.6",
        "@humanwhocodes/module-importer": "^1.0.1",
        "@humanwhocodes/retry": "^0.4.2",
        "@types/estree": "^1.0.6",
        "@types/json-schema": "^7.0.15",
        "ajv": "^6.12.4",
        "chalk": "^4.0.0",
        "cross-spawn": "^7.0.6",
        "debug": "^4.3.2",
        "escape-string-regexp": "^4.0.0",
        "eslint-scope": "^8.3.0",
        "eslint-visitor-keys": "^4.2.0",
        "espree": "^10.3.0",
        "esquery": "^1.5.0",
        "esutils": "^2.0.2",
        "fast-deep-equal": "^3.1.3",
        "file-entry-cache": "^8.0.0",
        "find-up": "^5.0.0",
        "glob-parent": "^6.0.2",
        "ignore": "^5.2.0",
        "imurmurhash": "^0.1.4",
        "is-glob": "^4.0.0",
        "json-stable-stringify-without-jsonify": "^1.0.1",
        "lodash.merge": "^4.6.2",
        "minimatch": "^3.1.2",
        "natural-compare": "^1.4.0",
        "optionator": "^0.9.3"
      },
      "bin": {
        "eslint": "bin/eslint.js"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      },
      "funding": {
        "url": "https://eslint.org/donate"
      },
      "peerDependencies": {
        "jiti": "*"
      },
      "peerDependenciesMeta": {
        "jiti": {
          "optional": true
        }
      }
    },
```

```
    "node_modules/eslint-plugin-react-hooks": {
      "version": "5.2.0",
      "resolved": "https://registry.npmjs.org/eslint-plugin-react-hooks/-/eslint-
plugin-react-hooks-5.2.0.tgz",
      "integrity": "sha512-+f15FfK64YQwZdJNELETdn5ibXEUQmW1DZL6KXhNnc2heoy/
sg9VJJeT7n8TlMWouzWqSWavFkIhHyIbIAEapg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=10"
      },
      "peerDependencies": {
        "eslint": "^3.0.0 || ^4.0.0 || ^5.0.0 || ^6.0.0 || ^7.0.0 || ^8.0.0-0 ||
^9.0.0"
      }
    },
    "node_modules/eslint-plugin-react-refresh": {
      "version": "0.4.20",
      "resolved": "https://registry.npmjs.org/eslint-plugin-react-refresh/-/
eslint-plugin-react-refresh-0.4.20.tgz",
      "integrity": "sha512-XpbHQ2q5gUF8BGOX4dHe+71qoirYMhApEPZ7sfhF/
dNnOF1UXnCMGZf79SFTBO7Bz5YEIT4TMieSlJBWhP9WBA==",
      "dev": true,
      "license": "MIT",
      "peerDependencies": {
        "eslint": ">=8.40"
      }
    },
    "node_modules/eslint-scope": {
      "version": "8.3.0",
      "resolved": "https://registry.npmjs.org/eslint-scope/-/eslint-
scope-8.3.0.tgz",
      "integrity": "sha512-pUNxi75F8MJ/
GdeKtVLSbYg4ZI34J6C0C7sbL4YOp2exGwen7ZsuBqKzUhXd0qMQ362yET3z+uPwKeg/0C2XCQ==",
      "dev": true,
      "license": "BSD-2-Clause",
      "dependencies": {
        "esrecurse": "^4.3.0",
        "estraverse": "^5.2.0"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      },
      "funding": {
        "url": "https://opencollective.com/eslint"
      }
    },
    "node_modules/eslint-visitor-keys": {
      "version": "3.4.3",
      "resolved": "https://registry.npmjs.org/eslint-visitor-keys/-/eslint-
visitor-keys-3.4.3.tgz",
      "integrity": "sha512-
wpc+LXeiyiisxPlEkUzU6svyS1frIO3Mgxj1fdy7Pm8Ygzguax2N3Fa/D/ag1WqbOprdI+uY6wMUl8/
a2G+iag==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": "^12.22.0 || ^14.17.0 || >=16.0.0"
      },
```

```
      "funding": {
        "url": "https://opencollective.com/eslint"
      }
    },
    "node_modules/eslint/node_modules/eslint-visitor-keys": {
      "version": "4.2.0",
      "resolved": "https://registry.npmjs.org/eslint-visitor-keys/-/eslint-
visitor-keys-4.2.0.tgz",
      "integrity": "sha512-UyLnSehNt62FFhSwjZlHmeokpRK59rcz29j+F1/
aDgbkbRTk7wIc9XzdoasMUbRNKDM0qQt/+BJ4BrpFeABemw==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      },
      "funding": {
        "url": "https://opencollective.com/eslint"
      }
    },
    "node_modules/eslint/node_modules/find-up": {
      "version": "5.0.0",
      "resolved": "https://registry.npmjs.org/find-up/-/find-up-5.0.0.tgz",
      "integrity": "sha512-78/PXT1wlLLDgTzDs7sjq9hzz0vXD+zn+7wypEe4fXQxCmdmqfGsEPQ
xmiCSQI3ajFV91bVSsvNtrJRiW6nGng==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "locate-path": "^6.0.0",
        "path-exists": "^4.0.0"
      },
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/eslint/node_modules/locate-path": {
      "version": "6.0.0",
      "resolved": "https://registry.npmjs.org/locate-path/-/locate-
path-6.0.0.tgz",
      "integrity": "sha512-iPZK6eYjbxRu3uB4/WZ3EsEIMJFMqAoopl3R+zuq0UjcAm/
MO6KCweDgPfP3elTztoKP3KtnVHxTn2NHBSDVUw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "p-locate": "^5.0.0"
      },
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/eslint/node_modules/p-limit": {
      "version": "3.1.0",
      "resolved": "https://registry.npmjs.org/p-limit/-/p-limit-3.1.0.tgz",
      "integrity": "sha512-TYOanM3wGwNGsZN2cVTYPArw454xnXj5qmWF1bEoAc4+cU/
ol7GVh7odevjp1FNHduHc3KZMcFduxU5Xc6uJRQ==",
```

```
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "yocto-queue": "^0.1.0"
      },
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/eslint/node_modules/p-locate": {
      "version": "5.0.0",
      "resolved": "https://registry.npmjs.org/p-locate/-/p-locate-5.0.0.tgz",
      "integrity": "sha512-LaNjtRWUBY++zB5nE/
NwcaoMylSPk+S+ZHNB1TzdbMJMny6dynpAGt7X/tl/QYq3TIeE6nxHppbo2LGymrG5Pw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "p-limit": "^3.0.2"
      },
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/espree": {
      "version": "10.3.0",
      "resolved": "https://registry.npmjs.org/espree/-/espree-10.3.0.tgz",
      "integrity": "sha512-0QYC8b24HWY8zjRnDTL6RiHfDbAWn63qb4LMj1Z4b076A4une81+z03
Kg7l7mn/48PUTqoLptSXez8oknU8Clg==",
      "dev": true,
      "license": "BSD-2-Clause",
      "dependencies": {
        "acorn": "^8.14.0",
        "acorn-jsx": "^5.3.2",
        "eslint-visitor-keys": "^4.2.0"
      },
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      },
      "funding": {
        "url": "https://opencollective.com/eslint"
      }
    },
    "node_modules/espree/node_modules/eslint-visitor-keys": {
      "version": "4.2.0",
      "resolved": "https://registry.npmjs.org/eslint-visitor-keys/-/eslint-
visitor-keys-4.2.0.tgz",
      "integrity": "sha512-UyLnSehNt62FFhSwjZlHmeokpRK59rcz29j+F1/
aDgbkbRTk7wIc9XzdoasMUbRNKDM0qQt/+BJ4BrpFeABemw==",
      "dev": true,
      "license": "Apache-2.0",
      "engines": {
        "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
      },
```

```
      "funding": {
        "url": "https://opencollective.com/eslint"
      }
    },
    "node_modules/esquery": {
      "version": "1.6.0",
      "resolved": "https://registry.npmjs.org/esquery/-/esquery-1.6.0.tgz",
      "integrity": "sha512-ca9pw9fomFcKPvFLXhBKUK90ZvGibiGOvRJNbjljY7s7uq/5YO4BOzc
YtJqExdx99rF6aAcnRxHmcUHcz6sQsg==",
      "dev": true,
      "license": "BSD-3-Clause",
      "dependencies": {
        "estraverse": "^5.1.0"
      },
      "engines": {
        "node": ">=0.10"
      }
    },
    "node_modules/esrecurse": {
      "version": "4.3.0",
      "resolved": "https://registry.npmjs.org/esrecurse/-/esrecurse-4.3.0.tgz",
      "integrity": "sha512-KmfKL3b6G+RXvP8N1vr3Tq1kL/oCFgn2NYXEtqP8/
L3pKapUA4G8cFVaoF3SU323CD4XypR/ffioHmkti6/Tag==",
      "dev": true,
      "license": "BSD-2-Clause",
      "dependencies": {
        "estraverse": "^5.2.0"
      },
      "engines": {
        "node": ">=4.0"
      }
    },
    "node_modules/estraverse": {
      "version": "5.3.0",
      "resolved": "https://registry.npmjs.org/estraverse/-/estraverse-5.3.0.tgz",
      "integrity": "sha512-MMdARuVEQziNTeJD8DgMqmhwR11BRQ/
cBP+pLtYdSTnf3MIO8fFeiINEbX36ZdNlfU/7A9f3gUw49B3oQsvwBA==",
      "dev": true,
      "license": "BSD-2-Clause",
      "engines": {
        "node": ">=4.0"
      }
    },
    "node_modules/esutils": {
      "version": "2.0.3",
      "resolved": "https://registry.npmjs.org/esutils/-/esutils-2.0.3.tgz",
      "integrity": "sha512-kVscqXk4OCp68SZ0dkgEKVi6/8ij300KBWTJq32P/
dYeWTSwK41WyTxalN1eRmA5Z9UU/LX9D7FWSmV9SAYx6g==",
      "dev": true,
      "license": "BSD-2-Clause",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/fast-deep-equal": {
      "version": "3.1.3",
      "resolved": "https://registry.npmjs.org/fast-deep-equal/-/fast-deep-
equal-3.1.3.tgz",
      "integrity": "sha512-f3qQ9oQy9j2AhBe/H9VC91wLmKBCCU/
```

```
gDOnKNAYG5hswO7BLKj09Hc5HYNz9cGI++xlpDCIgDaitVs03ATR84Q==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/fast-glob": {
      "version": "3.3.3",
      "resolved": "https://registry.npmjs.org/fast-glob/-/fast-glob-3.3.3.tgz",
      "integrity": "sha512-7MptL8U0cqcFdzIzwOTHoilX9x5BrNqye7Z/
LuC7kCMRio1EMSyqRK3BEAUD7sXRq4iT4AzTVuZdhgQ2TCvYLg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@nodelib/fs.stat": "^2.0.2",
        "@nodelib/fs.walk": "^1.2.3",
        "glob-parent": "^5.1.2",
        "merge2": "^1.3.0",
        "micromatch": "^4.0.8"
      },
      "engines": {
        "node": ">=8.6.0"
      }
    },
    "node_modules/fast-glob/node_modules/glob-parent": {
      "version": "5.1.2",
      "resolved": "https://registry.npmjs.org/glob-parent/-/glob-
parent-5.1.2.tgz",
      "integrity": "sha512-
AOIgSQCepiJYwP3ARnGx+5VnTu2HBYdzbGP45eLw1vr3zB3vZLeyed1sC9hnbcOc9/
SrMyM5RPQrkGz4aS9Zow==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "is-glob": "^4.0.1"
      },
      "engines": {
        "node": ">= 6"
      }
    },
    "node_modules/fast-json-stable-stringify": {
      "version": "2.1.0",
      "resolved": "https://registry.npmjs.org/fast-json-stable-stringify/-/fast-
json-stable-stringify-2.1.0.tgz",
      "integrity": "sha512-lhd/wF+Lk98HZoTCtlVraHtfh5XYijIjalXck7saUtuanSDyLMxnHhS
XEDJqHxD7msR8D0uCmqlkwjCV8xvwHw==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/fast-levenshtein": {
      "version": "2.0.6",
      "resolved": "https://registry.npmjs.org/fast-levenshtein/-/fast-
levenshtein-2.0.6.tgz",
      "integrity": "sha512-DCXu6Ifhqcks7TZKY3Hxp3y6qphY5SJZmrWMDrKcERSOXWQdMhU9Ig/
PYrzyw/ul9jOIyh0N4M0tbC5hodg8dw==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/fastq": {
      "version": "1.19.1",
      "resolved": "https://registry.npmjs.org/fastq/-/fastq-1.19.1.tgz",
```

```
      "integrity": "sha512-GwLTyxkCXjXbxqIhTsMI2Nui8huMPtnxg7krajPJAjnEG/
iiOS7i+zCtWGZR9G0NBKbXKh6X9m9UIsYX/N6vvQ==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "reusify": "^1.0.4"
      }
    },
    "node_modules/file-entry-cache": {
      "version": "8.0.0",
      "resolved": "https://registry.npmjs.org/file-entry-cache/-/file-entry-
cache-8.0.0.tgz",
      "integrity": "sha512-XXTUwCvisa5oacNGRP9SfNtYBNAMi+RPwBFmblZEF7N7swHYQS6/
Zfk7SRwx4D5j3CH211YNRco1DEMNVfZCnQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "flat-cache": "^4.0.0"
      },
      "engines": {
        "node": ">=16.0.0"
      }
    },
    "node_modules/fill-range": {
      "version": "7.1.1",
      "resolved": "https://registry.npmjs.org/fill-range/-/fill-range-7.1.1.tgz",
      "integrity": "sha512-YsGpe3WHLK8ZYi4tWDg2Jy3ebRz2rXowDxnld4bkQB00cc/1Zw9AWnC
0i9ztDJitivtQvaI9KaLyKrc+hBW0yg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "to-regex-range": "^5.0.1"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/flat-cache": {
      "version": "4.0.1",
      "resolved": "https://registry.npmjs.org/flat-cache/-/flat-cache-4.0.1.tgz",
      "integrity": "sha512-f7ccFPK3SXFHpx15UIGyRJ/
FJQctuKZ0zVuN3frBo4HnK3cay9VEW0R6yPYFHC0AgqhukPzKjq22t5DmAyqGyw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "flatted": "^3.2.9",
        "keyv": "^4.5.4"
      },
      "engines": {
        "node": ">=16"
      }
    },
    "node_modules/flatted": {
      "version": "3.3.3",
      "resolved": "https://registry.npmjs.org/flatted/-/flatted-3.3.3.tgz",
      "integrity": "sha512-GX+ysw4PBCz0PzosHDepZGANEuFCMLrnRTiEy9McGjmkCQYwRq4A/
X786G/fjM/+OjsWSU1ZrY5qyARZmO/uwg==",
      "dev": true,
      "license": "ISC"
```

```
    },
    "node_modules/follow-redirects": {
      "version": "1.15.9",
      "resolved": "https://registry.npmjs.org/follow-redirects/-/follow-
redirects-1.15.9.tgz",
      "integrity": "sha512-
gew4GsXizNgdoRyqmyfMHyAmXsZDk6mHkSxZFCzW9gwlbtOW44CDtYavM+y+72qD/
Vq2l550kMF52DT8fOLJqQ==",
      "funding": [
        {
          "type": "individual",
          "url": "https://github.com/sponsors/RubenVerborgh"
        }
      ],
      "license": "MIT",
      "engines": {
        "node": ">=4.0"
      },
      "peerDependenciesMeta": {
        "debug": {
          "optional": true
        }
      }
    },
    "node_modules/foreground-child": {
      "version": "3.3.1",
      "resolved": "https://registry.npmjs.org/foreground-child/-/foreground-
child-3.3.1.tgz",
      "integrity": "sha512-gIXjKqtFuWEgzFRJA9WCQeSJLZDjgJUOMCMzxtvFq/37KojM1BFGufq
sCy0r4qSQmYLsZYMeyRqzIWOMup03sw==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "cross-spawn": "^7.0.6",
        "signal-exit": "^4.0.1"
      },
      "engines": {
        "node": ">=14"
      },
      "funding": {
        "url": "https://github.com/sponsors/isaacs"
      }
    },
    "node_modules/form-data": {
      "version": "4.0.2",
      "resolved": "https://registry.npmjs.org/form-data/-/form-data-4.0.2.tgz",
      "integrity": "sha512-hGfm/slu0ZabnNt4oaRZ6uREyfCj6P4fT/
n6A1rGV+Z0VdGXjfOhVUpkn6qVQONHGIFwmveGXyDs75+nr6FM8w==",
      "license": "MIT",
      "dependencies": {
        "asynckit": "^0.4.0",
        "combined-stream": "^1.0.8",
        "es-set-tostringtag": "^2.1.0",
        "mime-types": "^2.1.12"
      },
      "engines": {
        "node": ">= 6"
      }
    },
```

```
    "node_modules/frac": {
      "version": "1.1.2",
      "resolved": "https://registry.npmjs.org/frac/-/frac-1.1.2.tgz",
      "integrity": "sha512-w/XBfkibaTl3YDqASwfDUqkna4Z2p9cFSr1aHDt0WoMTECnRfBOv2WA
rlZILlqgWlmdIlALXGpM2AOhEk5W3IA==",
      "license": "Apache-2.0",
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/fraction.js": {
      "version": "4.3.7",
      "resolved": "https://registry.npmjs.org/fraction.js/-/
fraction.js-4.3.7.tgz",
      "integrity": "sha512-ZsDfxO51wGAXREY55a7la9LScWpwv9RxIrYABrlvOFBlH/
ShPnrtsXeuUIfXKKOVicNxQ+o8JTbJvjS4M89yew==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": "*"
      },
      "funding": {
        "type": "patreon",
        "url": "https://github.com/sponsors/rawify"
      }
    },
    "node_modules/fsevents": {
      "version": "2.3.3",
      "resolved": "https://registry.npmjs.org/fsevents/-/fsevents-2.3.3.tgz",
      "integrity": "sha512-5xoDfX+fL7faATnagmWPpbFtwh/
R77WmMMqqHGS65C3vvB0YHrgF+B1YmZ3441tMj5n63k0212XNoJwzlhffQw==",
      "dev": true,
      "hasInstallScript": true,
      "license": "MIT",
      "optional": true,
      "os": [
        "darwin"
      ],
      "engines": {
        "node": "^8.16.0 || ^10.6.0 || >=11.0.0"
      }
    },
    "node_modules/function-bind": {
      "version": "1.1.2",
      "resolved": "https://registry.npmjs.org/function-bind/-/function-
bind-1.1.2.tgz",
      "integrity": "sha512-7XHNxH7qX9xG5mIwxkhumTox/
MIRNcOgDrxWsMt2pAr23WHp6MrRlN7FBSFpCpr+oVO0F744iUgR82nJMfG2SA==",
      "license": "MIT",
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/gensync": {
      "version": "1.0.0-beta.2",
      "resolved": "https://registry.npmjs.org/gensync/-/gensync-1.0.0-beta.2.tgz",
      "integrity":
"sha512-3hN7NaskYvMDLQY55gnW3NQ+mesEAepTqlg+VEbj7zzqEMBVNhzcGYYeqFo/
TlYz6eQiFcp1HcsCZO+nGgS8zg==",
```

```
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6.9.0"
      }
    },
    "node_modules/get-intrinsic": {
      "version": "1.3.0",
      "resolved": "https://registry.npmjs.org/get-intrinsic/-/get-
intrinsic-1.3.0.tgz",
      "integrity": "sha512-9fSjSaos/
fRIVIp+xSJlE6lfwhES7LNtKaCBIamHsjr2na1BiABJPo0mOjjz8GJDURarmCPGqaiVg5mfjb98CQ==",
      "license": "MIT",
      "dependencies": {
        "call-bind-apply-helpers": "^1.0.2",
        "es-define-property": "^1.0.1",
        "es-errors": "^1.3.0",
        "es-object-atoms": "^1.1.1",
        "function-bind": "^1.1.2",
        "get-proto": "^1.0.1",
        "gopd": "^1.2.0",
        "has-symbols": "^1.1.0",
        "hasown": "^2.0.2",
        "math-intrinsics": "^1.1.0"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/get-proto": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/get-proto/-/get-proto-1.0.1.tgz",
      "integrity": "sha512-sTSfBjoXBp89JvIKIefqw7U2CCebsc74kiY6awiGogKtoSGbgjYE/G/
+l9sF3MWFPNc9IcoOC4ODfKHfxFmp0g==",
      "license": "MIT",
      "dependencies": {
        "dunder-proto": "^1.0.1",
        "es-object-atoms": "^1.0.0"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/glob": {
      "version": "10.4.5",
      "resolved": "https://registry.npmjs.org/glob/-/glob-10.4.5.tgz",
      "integrity": "sha512-7Bv8RF0k6xjo7d4A/PxYLbUCfb6c+Vpd2/
mB2yRDlew7Jb5hEXiCD9ibfO7wpk8i4sevK6DFny9h7EYbM3/sHg==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "foreground-child": "^3.1.0",
        "jackspeak": "^3.1.2",
        "minimatch": "^9.0.4",
        "minipass": "^7.1.2",
        "package-json-from-dist": "^1.0.0",
```

```
        "path-scurry": "^1.11.1"
      },
      "bin": {
        "glob": "dist/esm/bin.mjs"
      },
      "funding": {
        "url": "https://github.com/sponsors/isaacs"
      }
    },
    "node_modules/glob-parent": {
      "version": "6.0.2",
      "resolved": "https://registry.npmjs.org/glob-parent/-/glob-
parent-6.0.2.tgz",
      "integrity": "sha512-XxwI8EOhVQgWp6iDL+3b0r86f4d6AX6zSU55HfB4ydCEuXLXc5FcYeO
u+nnGftS4TEju/11rt4KJPTMgbfmv4A==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "is-glob": "^4.0.3"
      },
      "engines": {
        "node": ">=10.13.0"
      }
    },
    "node_modules/glob/node_modules/brace-expansion": {
      "version": "2.0.1",
      "resolved": "https://registry.npmjs.org/brace-expansion/-/brace-
expansion-2.0.1.tgz",
      "integrity": "sha512-XnAIvQ8eM+kC6aULx6wuQiwVsnzsi9d3WxzV3FpWTGA19F621kwdbsA
cFKXgKUHZWsy+mY6iL1sHTxWEFCytDA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "balanced-match": "^1.0.0"
      }
    },
    "node_modules/glob/node_modules/minimatch": {
      "version": "9.0.5",
      "resolved": "https://registry.npmjs.org/minimatch/-/minimatch-9.0.5.tgz",
      "integrity": "sha512-G6T0ZX48xgozx7587koeX9Ys2NYy6Gmv//
P89sEte9V9whIapMNF4idKxnW2QtCcLiTWlb/wfCabAtAFWhhBow==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "brace-expansion": "^2.0.1"
      },
      "engines": {
        "node": ">=16 || 14 >=14.17"
      },
      "funding": {
        "url": "https://github.com/sponsors/isaacs"
      }
    },
    "node_modules/globals": {
      "version": "16.0.0",
      "resolved": "https://registry.npmjs.org/globals/-/globals-16.0.0.tgz",
      "integrity": "sha512-iInW14XItCXET01CQFqudPOWP2jYMl7T+QRQT+UNcR/iQncN/
F0UNpgd76iFkBPgNQb4+X3LV9tLJYzwh+Gl3A==",
      "dev": true,
```

```
      "license": "MIT",
      "engines": {
        "node": ">=18"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/goober": {
      "version": "2.1.16",
      "resolved": "https://registry.npmjs.org/goober/-/goober-2.1.16.tgz",
      "integrity": "sha512-erjk19y1U33+XAMe1VTvIONHYoSqE4iS7BYUZfHaqeohLmnC0FdxEh7
rQU+6MZ4OajItzjZFSRtVANrQwNq6/g==",
      "license": "MIT",
      "peerDependencies": {
        "csstype": "^3.0.10"
      }
    },
    "node_modules/gopd": {
      "version": "1.2.0",
      "resolved": "https://registry.npmjs.org/gopd/-/gopd-1.2.0.tgz",
      "integrity": "sha512-ZUKRh6/kUFoAiTAtTYPZJ3hw9wNxx+BIBOijnlG9PnrJsCcSjs1wyyD
6vJpaYtgnzDrKYRSqf3OO6Rfa93xsRg==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/has-flag": {
      "version": "4.0.0",
      "resolved": "https://registry.npmjs.org/has-flag/-/has-flag-4.0.0.tgz",
      "integrity": "sha512-EykJT/
Q1KjTWctppgIAgfSO0tKVuZUjhgMr17kqTumMl6Afv3EISleU7qZUzoXDFTAHTDC4NOoG/
ZxU3EvlMPQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/has-symbols": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/has-symbols/-/has-
symbols-1.1.0.tgz",
      "integrity": "sha512-1cDNdwJ2Jaohmb3sg4OmKaMBwuC48sYni5HUw2DvsC8LjGTLK9h+eb1
X6RyuOHe4hT0ULCW68iomhjUoKUqlPQ==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/has-tostringtag": {
      "version": "1.0.2",
```

```
      "resolved": "https://registry.npmjs.org/has-tostringtag/-/has-
tostringtag-1.0.2.tgz",
      "integrity": "sha512-
NqADB8VjPFLM2V0VvHUewwwsw0ZWBaIdgo+ieHtK3hasLz4qeCRjYcqfB6AQrBggRKppKF8L52/
VqdVsO47Dlw==",
      "license": "MIT",
      "dependencies": {
        "has-symbols": "^1.0.3"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/hasown": {
      "version": "2.0.2",
      "resolved": "https://registry.npmjs.org/hasown/-/hasown-2.0.2.tgz",
      "integrity": "sha512-0hJU9SCPvmMzIBdZFqNPXWa6dqh7WdH0cII9y+CyS8rG3nL48Bclra9
HmKhVVUHyPWNH5Y7xDwAB7bfgSjkUMQ==",
      "license": "MIT",
      "dependencies": {
        "function-bind": "^1.1.2"
      },
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/ignore": {
      "version": "5.3.2",
      "resolved": "https://registry.npmjs.org/ignore/-/ignore-5.3.2.tgz",
      "integrity": "sha512-hsBTNUqQTDwkWtcdYI2i06Y/
nUBEsNEDJKjWdigLvegy8kDuJAS8uRlpkkcQpyEXL0Z/pjDy5HBmMjRCJ2gq+g==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">= 4"
      }
    },
    "node_modules/import-fresh": {
      "version": "3.3.1",
      "resolved": "https://registry.npmjs.org/import-fresh/-/import-
fresh-3.3.1.tgz",
      "integrity": "sha512-TR3KfrTZTYLPB6jUjfx6MF9WcWrHL9su5TObK4ZkYgBdWKPOFoSoQId
EuTuR82pmtxH2spWG9h6etwfr1pLBqQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "parent-module": "^1.0.0",
        "resolve-from": "^4.0.0"
      },
      "engines": {
        "node": ">=6"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
```

```
    "node_modules/import-fresh/node_modules/resolve-from": {
      "version": "4.0.0",
      "resolved": "https://registry.npmjs.org/resolve-from/-/resolve-
from-4.0.0.tgz",
      "integrity": "sha512-pb/MYmXstAkysRFx8piNI1tGFNQIFA3vkE3Gq4EuA1dF6gHp/
+vgZqsCGJapvy8N3Q+4o7FwvquPJcnZ7RYy4g==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=4"
      }
    },
    "node_modules/imurmurhash": {
      "version": "0.1.4",
      "resolved": "https://registry.npmjs.org/imurmurhash/-/
imurmurhash-0.1.4.tgz",
      "integrity": "sha512-JmXMZ6wuvDmLiHEml9ykzqO6lwFbof0GG4IkcGaENdCRDDmMVnny7s5
HsIgHCbaq0w2MyPhDqkhTUgS2LU2PHA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.8.19"
      }
    },
    "node_modules/is-binary-path": {
      "version": "2.1.0",
      "resolved": "https://registry.npmjs.org/is-binary-path/-/is-binary-
path-2.1.0.tgz",
      "integrity": "sha512-ZMERYes6pDydyuGidse7OsHxtbI7WVeUEozgR/g7rd0xUimYNlvZRE/
K2MgZTjWy725IfelLeVcEM97mmtRGXw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "binary-extensions": "^2.0.0"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/is-core-module": {
      "version": "2.16.1",
      "resolved": "https://registry.npmjs.org/is-core-module/-/is-core-
module-2.16.1.tgz",
      "integrity": "sha512-UfoeMA6fIJ8wTYFEUjelnaGI67v6+N7qXJEvQuIGa99l4xsCruSYOVS
Q0uPANn4dAzm8lkYPaKLrrijLq7x23w==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "hasown": "^2.0.2"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/is-extglob": {
      "version": "2.1.1",
```

```
      "resolved": "https://registry.npmjs.org/is-extglob/-/is-extglob-2.1.1.tgz",
      "integrity": "sha512-SbKbANkN603Vi4jEZv49LeVJMn4yGwsbzZworEoyEiutsN3nJYdbO36
zfhGJ6QEDpOZIFkDtnq5JRxmvl3jsoQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/is-fullwidth-code-point": {
      "version": "3.0.0",
      "resolved": "https://registry.npmjs.org/is-fullwidth-code-point/-/is-
fullwidth-code-point-3.0.0.tgz",
      "integrity": "sha512-zymm5+u+sCsSWyD9qNaejV3DFvhCKclKdizYaJUuHA83RLjb7nSuGnd
dCHGv0hk+KY7BMAlsWeK4Ueg6EV6XQg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/is-glob": {
      "version": "4.0.3",
      "resolved": "https://registry.npmjs.org/is-glob/-/is-glob-4.0.3.tgz",
      "integrity": "sha512-xelSayHH36ZgE7ZWhli7pW34hNbNl8Ojv5KVmkJD4hBdD3th8Tfk9vY
asLM+mXWOZhFkgZfxhLSnrwRr4elSSg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "is-extglob": "^2.1.1"
      },
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/is-number": {
      "version": "7.0.0",
      "resolved": "https://registry.npmjs.org/is-number/-/is-number-7.0.0.tgz",
      "integrity": "sha512-41Cifkg6e8TylSpdtTpeLVMqvSBEVzTttHvERD741+pnZ8ANv0004MR
L43QKPDlK9cGvNp6NZWZUBlbGXYxxng==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.12.0"
      }
    },
    "node_modules/isexe": {
      "version": "2.0.0",
      "resolved": "https://registry.npmjs.org/isexe/-/isexe-2.0.0.tgz",
      "integrity": "sha512-RHxMLp9lnKHGHRng9QFhRCMbYAcVpn69smSGcq3f36xjgVVWThj4qqL
bTLlq7Ssj8B+fIQ1EuCEGI2lKsyQeIw==",
      "dev": true,
      "license": "ISC"
    },
    "node_modules/jackspeak": {
      "version": "3.4.3",
      "resolved": "https://registry.npmjs.org/jackspeak/-/jackspeak-3.4.3.tgz",
      "integrity": "sha512-OGlZQpz2yfahA/Rd1Y8Cd9SIEsqvXkLVoSw/
cgwhnhFMDbsQFeZYoJJ7bIZBS9BcamUW96asq/npPWugM+RQBw==",
```

```
      "dev": true,
      "license": "BlueOak-1.0.0",
      "dependencies": {
        "@isaacs/cliui": "^8.0.2"
      },
      "funding": {
        "url": "https://github.com/sponsors/isaacs"
      },
      "optionalDependencies": {
        "@pkgjs/parseargs": "^0.11.0"
      }
    },
    "node_modules/jiti": {
      "version": "2.4.2",
      "resolved": "https://registry.npmjs.org/jiti/-/jiti-2.4.2.tgz",
      "integrity": "sha512-rg9zJN+G4n2nfJl5MW3BMygZX56zKPNVEYYqq7adpmMh4Jn2QNEwhvQ
lFy6jPVdcod7txZtKHWnyZiA3a0zP7A==",
      "dev": true,
      "license": "MIT",
      "optional": true,
      "peer": true,
      "bin": {
        "jiti": "lib/jiti-cli.mjs"
      }
    },
    "node_modules/js-tokens": {
      "version": "4.0.0",
      "resolved": "https://registry.npmjs.org/js-tokens/-/js-tokens-4.0.0.tgz",
      "integrity": "sha512-RdJUflcE3cUzKiMqQgsCu06FPu9UdIJO0beYbPhHN4k6apgJtifcoCt
T9bcxOpYBtpD2kCM6Sbzg4CausW/PKQ==",
      "license": "MIT"
    },
    "node_modules/jsesc": {
      "version": "3.1.0",
      "resolved": "https://registry.npmjs.org/jsesc/-/jsesc-3.1.0.tgz",
      "integrity": "sha512-/sM3dO2FOzXjKQhJuo0Q173wf2KOo8t4I8vHy6lF9poUp7bKT0/
NHE8fPX23PwfhnykfqnC2xRxOnVw5XuGIaA==",
      "dev": true,
      "license": "MIT",
      "bin": {
        "jsesc": "bin/jsesc"
      },
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/json-buffer": {
      "version": "3.0.1",
      "resolved": "https://registry.npmjs.org/json-buffer/-/json-
buffer-3.0.1.tgz",
      "integrity": "sha512-4bV5BfR2mqfQTJm+V5tPPdf+ZpuhiIvTuAB5g8kcrXOZpTT/
QwwVRWBywX1ozr6lEuPdbHxwaJlm9G6mI2sfSQ==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/json-schema-traverse": {
      "version": "0.4.1",
      "resolved": "https://registry.npmjs.org/json-schema-traverse/-/json-schema-
traverse-0.4.1.tgz",
```

```
      "integrity": "sha512-xbbCH5dCYU5T8LcEhhuh7HJ88HXuW3qsI3Y0zOZFKfZEHcpWiHU/
Jxzk629Brsab/mMiHQti9wMP+845RPe3Vg==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/json-stable-stringify-without-jsonify": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/json-stable-stringify-without-
jsonify/-/json-stable-stringify-without-jsonify-1.0.1.tgz",
      "integrity": "sha512-Bdboy+l7tA3OGW6FjyFHWkP5LuByj1Tk33Ljyq0axyzdk9//
JSi2u3fP1QSmd1KNwq6VOKYGlAu87CisVir6Pw==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/json5": {
      "version": "2.2.3",
      "resolved": "https://registry.npmjs.org/json5/-/json5-2.2.3.tgz",
      "integrity": "sha512-XmOWe7eyHYH14cLdVPoyg+GOH3rYX+
+KpzrylJwSW98t3Nk+U8XOl8FWKOgwtzdb8lXGf6zYwDUzeHMWfxasyg==",
      "dev": true,
      "license": "MIT",
      "bin": {
        "json5": "lib/cli.js"
      },
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/keyv": {
      "version": "4.5.4",
      "resolved": "https://registry.npmjs.org/keyv/-/keyv-4.5.4.tgz",
      "integrity": "sha512-oxVHkHR/
EJf2CNXnWxRLW6mg7JyCCUcG0DtEGmL2ctUo1PNTin1PUil+r/+4r5MpVgC/fn1kjsx7mjSujKqIpw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "json-buffer": "3.0.1"
      }
    },
    "node_modules/levn": {
      "version": "0.4.1",
      "resolved": "https://registry.npmjs.org/levn/-/levn-0.4.1.tgz",
      "integrity": "sha512-+bT2uH4E5LGE7h/n3evcS/sQlJXCpIp6ym8OWJ5eV6+67Dsql/
LaaT7qJBAt2rzfoa/5QBGBhxDix1dMt2kQKQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "prelude-ls": "^1.2.1",
        "type-check": "~0.4.0"
      },
      "engines": {
        "node": ">= 0.8.0"
      }
    },
    "node_modules/lightningcss": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss/-/
lightningcss-1.29.2.tgz",
      "integrity": "sha512-6b6gd/RUXKaw5keVdSEtqFVdzWnU5jMxTUjA2bVcMNPLwSQ08Sv/
UodBVtETLCn7k4S1Ibxwh7k68IwLZPgKaA==",
```

```
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "peer": true,
      "dependencies": {
        "detect-libc": "^2.0.3"
      },
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      },
      "optionalDependencies": {
        "lightningcss-darwin-arm64": "1.29.2",
        "lightningcss-darwin-x64": "1.29.2",
        "lightningcss-freebsd-x64": "1.29.2",
        "lightningcss-linux-arm-gnueabihf": "1.29.2",
        "lightningcss-linux-arm64-gnu": "1.29.2",
        "lightningcss-linux-arm64-musl": "1.29.2",
        "lightningcss-linux-x64-gnu": "1.29.2",
        "lightningcss-linux-x64-musl": "1.29.2",
        "lightningcss-win32-arm64-msvc": "1.29.2",
        "lightningcss-win32-x64-msvc": "1.29.2"
      }
    },
    "node_modules/lightningcss-darwin-arm64": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-darwin-arm64/-/
lightningcss-darwin-arm64-1.29.2.tgz",
      "integrity": "sha512-cK/eMabSViKn/PG8U/
a7aCorpeKLMlK0bQeNHmdb7qUnBkNPnL+oV5DjJUo0kqWsJUapZsM4jCfYItbqBDvlcA==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "darwin"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-darwin-x64": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-darwin-x64/-/
lightningcss-darwin-x64-1.29.2.tgz",
      "integrity": "sha512-j5qYxamyQw4kDXX5hnnCKMf3mLlHvG44f24Qyi2965/
Ycz829MYqjrVg2H8BidybHBp9kom4D7DR5VqCKDXS0w==",
      "cpu": [
        "x64"
```

```
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "darwin"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-freebsd-x64": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-freebsd-x64/-/
lightningcss-freebsd-x64-1.29.2.tgz",
      "integrity": "sha512-wDk7M2tM78Ii8ek9YjnY8MjV5f5JN2qNVO+/0BAGZRvXKtQrBC4/
cn4ssQIpKIPP44YXw6gFdpUF+Ps+RGsCwg==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "freebsd"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-linux-arm-gnueabihf": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-linux-arm-gnueabihf/-/
lightningcss-linux-arm-gnueabihf-1.29.2.tgz",
      "integrity": "sha512-IRUrOrAF2Z+KExdExe3Rz7NSTuuJ2HvCGlMKoquK5pjvo2JY4Rybr+N
rKnq0U0hZnx5AnGsuFHjGnNT14w26sg==",
      "cpu": [
        "arm"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "linux"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
```

```
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-linux-arm64-gnu": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-linux-arm64-gnu/-/
lightningcss-linux-arm64-gnu-1.29.2.tgz",
      "integrity": "sha512-KKCpOlmhdjvUTX/mBuaKemp0oeDIBBLFiU5Fnqxh1/
DZ4JPZi4evEH7TKoSBFOSOV3J7iEmmBaw/8dpiUvRKlQ==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "linux"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-linux-arm64-musl": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-linux-arm64-musl/-/
lightningcss-linux-arm64-musl-1.29.2.tgz",
      "integrity": "sha512-Q64eM1bPlOOUgxFmoPUefqzY1yV3ctFPE6d/
Vt7WzLW4rKTv7MyYNky+FWxRpLkNASTnKQUaiMJ87zNODIrrKQ==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "linux"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-linux-x64-gnu": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-linux-x64-gnu/-/
lightningcss-linux-x64-gnu-1.29.2.tgz",
      "integrity": "sha512-0v6idDCPG6epLXtBH/
RPkHvYx74CVziHo6TMYga8O2EiQApnUPZsbR9nFNrg2cgBzk1AYqEd95TlrsL7nYABQg==",
```

```
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "linux"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-linux-x64-musl": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-linux-x64-musl/-/
lightningcss-linux-x64-musl-1.29.2.tgz",
      "integrity": "sha512-rMpz2yawkgGT8RULc5S4WiZopVMOFWjiItBT7aSfDX4NQav6M44rhn5
hjtkKzB+wMTRlLLqxkeYEtQ3dd9696w==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "linux"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-win32-arm64-msvc": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-win32-arm64-msvc/-/
lightningcss-win32-arm64-msvc-1.29.2.tgz",
      "integrity": "sha512-nL7zRW6evGQqYVu/
bKGK+zShyz8OVzsCotFgc7judbt6wnB2KbiKKJwBE4SGoDBQ1O94RjW4asrCjQL4i8Fhbw==",
      "cpu": [
        "arm64"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "win32"
      ],
      "peer": true,
      "engines": {
```

```
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lightningcss-win32-x64-msvc": {
      "version": "1.29.2",
      "resolved": "https://registry.npmjs.org/lightningcss-win32-x64-msvc/-/
lightningcss-win32-x64-msvc-1.29.2.tgz",
      "integrity": "sha512-EdIUW3B2vLuHmv7urfzMI/
h2fmlnOQBk1xlsDxkN1tCWKjNFjfLhGxYk8C8mzpSfr+A6jFFIi8fU6LbQGsRWjA==",
      "cpu": [
        "x64"
      ],
      "dev": true,
      "license": "MPL-2.0",
      "optional": true,
      "os": [
        "win32"
      ],
      "peer": true,
      "engines": {
        "node": ">= 12.0.0"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/parcel"
      }
    },
    "node_modules/lilconfig": {
      "version": "3.1.3",
      "resolved": "https://registry.npmjs.org/lilconfig/-/lilconfig-3.1.3.tgz",
      "integrity": "sha512-/vlFKAoH5Cgt3Ie+JLhRbwOsCQePABiU3tJ1egGvyQ+33R/
vcwM2Zl2QR/LzjsBeItPt3oSVXapn+m4nQDvpzw==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=14"
      },
      "funding": {
        "url": "https://github.com/sponsors/antonk52"
      }
    },
    "node_modules/lines-and-columns": {
      "version": "1.2.4",
      "resolved": "https://registry.npmjs.org/lines-and-columns/-/lines-and-
columns-1.2.4.tgz",
      "integrity": "sha512-7ylylesZQ/
PV29jhEDl3Ufjo6ZX7gCqJr5F7PKrqc93v7fzSymt1BpwEU8nAUXs8qzzvqhbjhK5QZg6Mt/HkBg==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/lodash.debounce": {
      "version": "4.0.8",
      "resolved": "https://registry.npmjs.org/lodash.debounce/-/
lodash.debounce-4.0.8.tgz",
      "integrity": "sha512-
```

```
FT1yDzDYEoYWhnSGnpE/4Kj1fLZkDFyqRb7fNt6FdYOSxlUWAtp42Eh6Wb0rGIv/
m9Bgo7x4GhQbm5Ys4SG5ow==",
      "license": "MIT"
    },
    "node_modules/lodash.merge": {
      "version": "4.6.2",
      "resolved": "https://registry.npmjs.org/lodash.merge/-/
lodash.merge-4.6.2.tgz",
      "integrity": "sha512-0KpjqXRVvrYyCsX1swR/
XTK0va6VQkQM6MNo7PqW77ByjAhoARA8EfrP1N4+KlKj8YS0ZUCtRT/YUuhyYDujIQ==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/lodash.throttle": {
      "version": "4.1.1",
      "resolved": "https://registry.npmjs.org/lodash.throttle/-/
lodash.throttle-4.1.1.tgz",
      "integrity": "sha512-
wIkUCfVKpVsWo3JSZlc+8MB5it+2AN5W8J7YVMST30UrvcQNZ1Okbj+rbVniijTWE6FGYy4XJq/
rHkas8qJMLQ==",
      "license": "MIT"
    },
    "node_modules/loose-envify": {
      "version": "1.4.0",
      "resolved": "https://registry.npmjs.org/loose-envify/-/loose-
envify-1.4.0.tgz",
      "integrity": "sha512-lyuxPGr/Wfhrlem2CL/UcnUc1zcqKAImBDzukY7Y5F/
yQiNdko6+fRLevlw1HgMySw7f611UIY408EtxRSoK3Q==",
      "license": "MIT",
      "dependencies": {
        "js-tokens": "^3.0.0 || ^4.0.0"
      },
      "bin": {
        "loose-envify": "cli.js"
      }
    },
    "node_modules/lottie-react": {
      "version": "2.4.1",
      "resolved": "https://registry.npmjs.org/lottie-react/-/lottie-
react-2.4.1.tgz",
      "integrity": "sha512-LQrH7jlkigIIv+
+wIyrOYFLHSKQpEY4zehPicL9bQsrt1rnoKRYCYgpCUe5maqylNtacy58/sQDZTkwMcTRxZw==",
      "license": "MIT",
      "dependencies": {
        "lottie-web": "^5.10.2"
      },
      "peerDependencies": {
        "react": "^16.8.0 || ^17.0.0 || ^18.0.0 || ^19.0.0",
        "react-dom": "^16.8.0 || ^17.0.0 || ^18.0.0 || ^19.0.0"
      }
    },
    "node_modules/lottie-web": {
      "version": "5.12.2",
      "resolved": "https://registry.npmjs.org/lottie-web/-/lottie-web-5.12.2.tgz",
      "integrity": "sha512-uvhvYPC8kGPjXT3MyKMrL3JitEAmDMp30lVkuq/590Mw9ok6pWcFCwX
Jveo0t5uqYw1UREQHofD+jVpdjBv8wg==",
      "license": "MIT"
    },
    "node_modules/lru-cache": {
```

```
      "version": "5.1.1",
      "resolved": "https://registry.npmjs.org/lru-cache/-/lru-cache-5.1.1.tgz",
      "integrity": "sha512-KpNARQA3Iwv+jTA0utUVVbrh+Jlrr1Fv0e56GGzAFOXN7dk/
FviaDW8LHmK52DlcH4WP2n6gI8vN1aesBFgo9w==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "yallist": "^3.0.2"
      }
    },
    "node_modules/lucide-react": {
      "version": "0.511.0",
      "resolved": "https://registry.npmjs.org/lucide-react/-/lucide-
react-0.511.0.tgz",
      "integrity": "sha512-VK5a2ydJ7xm8GvBeKLS9mu1pVK6ucef9780JVUjw6bAjJL/
QXnd4Y0p7SPeOUMC27YhzNCZvm5d/QX0Tp3rc0w==",
      "license": "ISC",
      "peerDependencies": {
        "react": "^16.5.1 || ^17.0.0 || ^18.0.0 || ^19.0.0"
      }
    },
    "node_modules/math-intrinsics": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/math-intrinsics/-/math-
intrinsics-1.1.0.tgz",
      "integrity": "sha512-/IXtbwEk5HTPyEwyKX6hGkYXxM9nbj64B+ilVJnC/
R6B0pH5G4V3b0pVbL7DBj4tkhBAppbQUlf6F6Xl9LHu1g==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      }
    },
    "node_modules/merge2": {
      "version": "1.4.1",
      "resolved": "https://registry.npmjs.org/merge2/-/merge2-1.4.1.tgz",
      "integrity": "sha512-8q7VEgMJW4J8tcfVPy8g09NcQwZdbwFEqhe/
WZkoIzjn/3TGDwtOCYtXGxA3O8tPzpczCCDgv+P2P5y00ZJOOg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">= 8"
      }
    },
    "node_modules/micromatch": {
      "version": "4.0.8",
      "resolved": "https://registry.npmjs.org/micromatch/-/micromatch-4.0.8.tgz",
      "integrity": "sha512-
PXwfBhYu0hBCPw8Dn0E+WDYb7af3dSLVWKi3HGv84IdF4TyFoC0ysxFd0Goxw7nSv4T/
PzEJQxsYsEiFCKo2BA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "braces": "^3.0.3",
        "picomatch": "^2.3.1"
      },
      "engines": {
        "node": ">=8.6"
      }
    },
```

```
    "node_modules/mime-db": {
      "version": "1.52.0",
      "resolved": "https://registry.npmjs.org/mime-db/-/mime-db-1.52.0.tgz",
      "integrity": "sha512-sPU4uV7dYlvtWJxwwxHD0PuihVNiE7TyAbQ5SWxDCB9mUYvOgroQOwY
QQOKPJ8CIbE+1ETVlOoK1UC2nU3gYvg==",
      "license": "MIT",
      "engines": {
        "node": ">= 0.6"
      }
    },
    "node_modules/mime-types": {
      "version": "2.1.35",
      "resolved": "https://registry.npmjs.org/mime-types/-/mime-types-2.1.35.tgz",
      "integrity": "sha512-ZDY+bPm5zTTF+YpCrAU9nK0UgICYPT0QtT1NZWFv4s+
+TNkcgVaT0g6+4R2uI4MjQjzysHB1zxuWL50hzaeXiw==",
      "license": "MIT",
      "dependencies": {
        "mime-db": "1.52.0"
      },
      "engines": {
        "node": ">= 0.6"
      }
    },
    "node_modules/minimatch": {
      "version": "3.1.2",
      "resolved": "https://registry.npmjs.org/minimatch/-/minimatch-3.1.2.tgz",
      "integrity": "sha512-J7p63hRiAjw1NDEww1W7i37+ByIrOWO5XQQAzZ3VOcL0PNybwpfmV/
N05zFAzwQ9USyEcX6t3UO+K5aqBQOIHw==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "brace-expansion": "^1.1.7"
      },
      "engines": {
        "node": "*"
      }
    },
    "node_modules/minipass": {
      "version": "7.1.2",
      "resolved": "https://registry.npmjs.org/minipass/-/minipass-7.1.2.tgz",
      "integrity": "sha512-qOOzS1cBTWYF4BH8fVePDBOO9iptMnGUEZwNc/
cMWnTV2nVLZ7VoNWEPHkYczZA0pdoA7dl6e7FL659nX9S2aw==",
      "dev": true,
      "license": "ISC",
      "engines": {
        "node": ">=16 || 14 >=14.17"
      }
    },
    "node_modules/ms": {
      "version": "2.1.3",
      "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
      "integrity": "sha512-6FlzubTLZG3J2a/NVCAleEhjzq5oxgHyaCU9yYXvcLsvoVaHJq/
s5xXI6/XXP6tz7R9xAOtHnSO/tXtF3WRTlA==",
      "license": "MIT"
    },
    "node_modules/mz": {
      "version": "2.7.0",
      "resolved": "https://registry.npmjs.org/mz/-/mz-2.7.0.tgz",
      "integrity": "sha512-z81GNO7nnYMEhrGh9LeymoE4+Yr0Wn5McHIZMK5cfQCl+NDX08sCZgU
```

```
c9/6MHni9IWuFLm1Z3HTCXu2z9fN62Q==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "any-promise": "^1.0.0",
        "object-assign": "^4.0.1",
        "thenify-all": "^1.0.0"
      }
    },
    "node_modules/nanoid": {
      "version": "3.3.11",
      "resolved": "https://registry.npmjs.org/nanoid/-/nanoid-3.3.11.tgz",
      "integrity": "sha512-N8SpfPUnUp1bK+PMYW8qSWdl9U+wwNWI4QKxOYDy9JAro3WMX7p2OeV
RF9v+347pnakNevPmiHhNmZ2HbFA76w==",
      "dev": true,
      "funding": [
        {
          "type": "github",
          "url": "https://github.com/sponsors/ai"
        }
      ],
      "license": "MIT",
      "bin": {
        "nanoid": "bin/nanoid.cjs"
      },
      "engines": {
        "node": "^10 || ^12 || ^13.7 || ^14 || >=15.0.1"
      }
    },
    "node_modules/natural-compare": {
      "version": "1.4.0",
      "resolved": "https://registry.npmjs.org/natural-compare/-/natural-
compare-1.4.0.tgz",
      "integrity": "sha512-OWND8ei3VtNC9h7V60qff3SVobHr996CTwgxubgyQYEpg290h9J0buy
ECNNJexkFm5sOajh5G116RYA1c8ZMSw==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/node-releases": {
      "version": "2.0.19",
      "resolved": "https://registry.npmjs.org/node-releases/-/node-
releases-2.0.19.tgz",
      "integrity": "sha512-
xxOWJsBKtzAq7DY0J+DTzuz58K8e7sJbdgwkbMWQe8UYB6ekmsQ45q0M/
tJDsGaZmbC+l7n57UV8Hl5tHxO9uw==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/normalize-path": {
      "version": "3.0.0",
      "resolved": "https://registry.npmjs.org/normalize-path/-/normalize-
path-3.0.0.tgz",
      "integrity":
"sha512-6eZs5Ls3WtCisHWp9S2GUy8dqkpGi4BVSz3GaqiE6ezub0512ESztXUwUB6C6IKbQkY2Pnb/
mD4WYojCRwcwLA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.10.0"
```

```
      }
    },
    "node_modules/normalize-range": {
      "version": "0.1.2",
      "resolved": "https://registry.npmjs.org/normalize-range/-/normalize-
range-0.1.2.tgz",
      "integrity": "sha512-bdok/XvKII3nUpklnV6P2hxtMNrCboOjAcyBuQnWEhO665FwrSNRxU+
AqpsyvO6LgGYPspN+lu5CLtw4jPRKNA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/object-assign": {
      "version": "4.1.1",
      "resolved": "https://registry.npmjs.org/object-assign/-/object-
assign-4.1.1.tgz",
      "integrity": "sha512-rJgTQnkUnH1sFw8yT6VSU3zD3sWmu6sZhIseY8VX+GRu3P6F7Fu+JND
oXfklElbLJSnc3FUQHVe4cU5hj+BcUg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/object-hash": {
      "version": "3.0.0",
      "resolved": "https://registry.npmjs.org/object-hash/-/object-
hash-3.0.0.tgz",
      "integrity": "sha512-RSn9F68PjH9HqtltsSnqYC1XXoWe9Bju5+213R98cNGttag9q9yAOTz
dbsqvIa7aNm5WffBZFpWYr2aWrklWAw==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">= 6"
      }
    },
    "node_modules/optionator": {
      "version": "0.9.4",
      "resolved": "https://registry.npmjs.org/optionator/-/optionator-0.9.4.tgz",
      "integrity":
"sha512-6IpQ7mKUxRcZNLIObR0hz7lxsapSSIYNZJwXPGeF0mTVqGKFIXj1DQcMoT22S3ROcLyY/
rz0PWaWZ9ayWmad9g==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "deep-is": "^0.1.3",
        "fast-levenshtein": "^2.0.6",
        "levn": "^0.4.1",
        "prelude-ls": "^1.2.1",
        "type-check": "^0.4.0",
        "word-wrap": "^1.2.5"
      },
      "engines": {
        "node": ">= 0.8.0"
      }
    },
    "node_modules/package-json-from-dist": {
```

```
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/package-json-from-dist/-/package-
json-from-dist-1.0.1.tgz",
      "integrity": "sha512-UEZIS3/by4OC8vL3P2dTXRETpebLI2NiI5vIrjaD/5UtrkFX/
tNbwjTSRAGC/+7CAo2pIcBaRgWmcBBHcsaCIw==",
      "dev": true,
      "license": "BlueOak-1.0.0"
    },
    "node_modules/parent-module": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/parent-module/-/parent-
module-1.0.1.tgz",
      "integrity": "sha512-GQ2EWRpQV8/o+Aw8YqtfZZPfNRWZYkbidE9k5rpl/
hC3vtHHBfGm2Ifi6qWV+coDGkrUKZAxE3Lot5kcsRlh+g==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "callsites": "^3.0.0"
      },
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/path-exists": {
      "version": "4.0.0",
      "resolved": "https://registry.npmjs.org/path-exists/-/path-
exists-4.0.0.tgz",
      "integrity": "sha512-ak9Qy5Q7jYb2Wwcey5Fpvg2KoAc/
ZIhLSLOSBmRmygPsGwkVVt0fZa0qrtMz+m6tJTAHfZQ8FnmB4MG4LWy7/w==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/path-key": {
      "version": "3.1.1",
      "resolved": "https://registry.npmjs.org/path-key/-/path-key-3.1.1.tgz",
      "integrity": "sha512-ojmeN0qd+y0jszEtoY48r0Peq5dwMEkIlCOu6Q5f41lfkswXuKtYrhg
oTpLnyIcHm24Uhqx+5Tqm2InSwLhE6Q==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/path-parse": {
      "version": "1.0.7",
      "resolved": "https://registry.npmjs.org/path-parse/-/path-parse-1.0.7.tgz",
      "integrity": "sha512-LDJzPVEEEPR+y48z93A0Ed0yXb8pAByGWo/
k5YYdYgpY2/2EsOsksJrq7lOHxryrVOn1ejG6oAp8ahvOIQD8sw==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/path-scurry": {
      "version": "1.11.1",
      "resolved": "https://registry.npmjs.org/path-scurry/-/path-
scurry-1.11.1.tgz",
      "integrity": "sha512-
```

```
Xa4Nw17FS9ApQFJ9umLiJS4orGjm7ZzwUrwamcGQuHSzDyth9boKDaycYdDcZDuqYATXw4HFXgaqWTctW/
v1HA==",
      "dev": true,
      "license": "BlueOak-1.0.0",
      "dependencies": {
        "lru-cache": "^10.2.0",
        "minipass": "^5.0.0 || ^6.0.2 || ^7.0.0"
      },
      "engines": {
        "node": ">=16 || 14 >=14.18"
      },
      "funding": {
        "url": "https://github.com/sponsors/isaacs"
      }
    },
    "node_modules/path-scurry/node_modules/lru-cache": {
      "version": "10.4.3",
      "resolved": "https://registry.npmjs.org/lru-cache/-/lru-cache-10.4.3.tgz",
      "integrity": "sha512-JNAzZcXrCt42VGLuYz0zfAzDfAvJWW6AfYlDBQyDV5DClI2m5sAmK+O
IO7s59XfsRsWHp02jAJrRadPRGTt6SQ==",
      "dev": true,
      "license": "ISC"
    },
    "node_modules/picocolors": {
      "version": "1.1.1",
      "resolved": "https://registry.npmjs.org/picocolors/-/picocolors-1.1.1.tgz",
      "integrity": "sha512-xceH2snhtb5M9liqDsmEw56le376mTZkEX/jEb/
RxNFyegNul7eNslCXP9FDj/Lcu0X8KEyMceP2ntpaHrDEVA==",
      "dev": true,
      "license": "ISC"
    },
    "node_modules/picomatch": {
      "version": "2.3.1",
      "resolved": "https://registry.npmjs.org/picomatch/-/picomatch-2.3.1.tgz",
      "integrity": "sha512-JU3teHTNjmE2VCGFzuY8EXzCDVwEqB2a8fsIvwaStHhAWJEeVd1o1QD
80CU6+ZdEXXSLbSsuLwJjkCBWqRQUVA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8.6"
      },
      "funding": {
        "url": "https://github.com/sponsors/jonschlinkert"
      }
    },
    "node_modules/pify": {
      "version": "2.3.0",
      "resolved": "https://registry.npmjs.org/pify/-/pify-2.3.0.tgz",
      "integrity": "sha512-
udgsAY+fTnvv7kI7aaxbqwWNb0AHiB0qBO89PZKPkoTmGOgdbrHDKD+0B2X4uTfJ/
FT1R09r9gTsjUjNJotuog==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/pirates": {
      "version": "4.0.7",
```

```
      "resolved": "https://registry.npmjs.org/pirates/-/pirates-4.0.7.tgz",
      "integrity": "sha512-TfySrs/5nm8fQJDcBDuUng3VOUKsd7S+zqvbOTiGXHfxX4wK31ard+h
oNuvkicM/2YFzlpDgABOevKSsB4G/FA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">= 6"
      }
    },
    "node_modules/postcss": {
      "version": "8.5.3",
      "resolved": "https://registry.npmjs.org/postcss/-/postcss-8.5.3.tgz",
      "integrity": "sha512-dle9A3yYxlBSrt8Fu+IpjGT8SY8hN0mlaA6GY8t0P5PjIOZemULz/
E2Bnm/2dcUOena75OTNkHI76uZBNUUq3A==",
      "dev": true,
      "funding": [
        {
          "type": "opencollective",
          "url": "https://opencollective.com/postcss/"
        },
        {
          "type": "tidelift",
          "url": "https://tidelift.com/funding/github/npm/postcss"
        },
        {
          "type": "github",
          "url": "https://github.com/sponsors/ai"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "nanoid": "^3.3.8",
        "picocolors": "^1.1.1",
        "source-map-js": "^1.2.1"
      },
      "engines": {
        "node": "^10 || ^12 || >=14"
      }
    },
    "node_modules/postcss-import": {
      "version": "15.1.0",
      "resolved": "https://registry.npmjs.org/postcss-import/-/postcss-
import-15.1.0.tgz",
      "integrity": "sha512-hpr+J05B2FVYUAXHeK1YyI267J/
dDDhMU6B6civm8hSY1jYJnBXxzKDKDswzJmtLHryrjhnDjqqp/49t8FALew==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "postcss-value-parser": "^4.0.0",
        "read-cache": "^1.0.0",
        "resolve": "^1.1.7"
      },
      "engines": {
        "node": ">=14.0.0"
      },
      "peerDependencies": {
        "postcss": "^8.0.0"
      }
    },
```

```
    "node_modules/postcss-js": {
      "version": "4.0.1",
      "resolved": "https://registry.npmjs.org/postcss-js/-/postcss-js-4.0.1.tgz",
      "integrity": "sha512-dDLF8pEO191hJMtlHFPRa8xsizHaM82MLfNkUHdUtVEV3tgTp5oj+8q
bEqYM57SLfc74KSbw//4SeJma2LRVIw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "camelcase-css": "^2.0.1"
      },
      "engines": {
        "node": "^12 || ^14 || >= 16"
      },
      "funding": {
        "type": "opencollective",
        "url": "https://opencollective.com/postcss/"
      },
      "peerDependencies": {
        "postcss": "^8.4.21"
      }
    },
    "node_modules/postcss-load-config": {
      "version": "4.0.2",
      "resolved": "https://registry.npmjs.org/postcss-load-config/-/postcss-load-
config-4.0.2.tgz",
      "integrity": "sha512-bSVhyJGL00wMVoPUzAVAnbEoWyqRxkjv64tUl427SKnPrENtq6hJwUo
jroMz2VB+Q1edmi4IfrAPpami5VVgMQ==",
      "dev": true,
      "funding": [
        {
          "type": "opencollective",
          "url": "https://opencollective.com/postcss/"
        },
        {
          "type": "github",
          "url": "https://github.com/sponsors/ai"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "lilconfig": "^3.0.0",
        "yaml": "^2.3.4"
      },
      "engines": {
        "node": ">= 14"
      },
      "peerDependencies": {
        "postcss": ">=8.0.9",
        "ts-node": ">=9.0.0"
      },
      "peerDependenciesMeta": {
        "postcss": {
          "optional": true
        },
        "ts-node": {
          "optional": true
        }
      }
    },
```

```
    "node_modules/postcss-nested": {
      "version": "6.2.0",
      "resolved": "https://registry.npmjs.org/postcss-nested/-/postcss-
nested-6.2.0.tgz",
      "integrity": "sha512-HQbt28KulC5AJzG+cZtj9kvKB93CFCdLvog1WFLf1D+xmMvPGlBstkp
TEZfK5+AN9hfJocyBFCNiqyS48bpgzQ==",
      "dev": true,
      "funding": [
        {
          "type": "opencollective",
          "url": "https://opencollective.com/postcss/"
        },
        {
          "type": "github",
          "url": "https://github.com/sponsors/ai"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "postcss-selector-parser": "^6.1.1"
      },
      "engines": {
        "node": ">=12.0"
      },
      "peerDependencies": {
        "postcss": "^8.2.14"
      }
    },
    "node_modules/postcss-selector-parser": {
      "version": "6.1.2",
      "resolved": "https://registry.npmjs.org/postcss-selector-parser/-/postcss-
selector-parser-6.1.2.tgz",
      "integrity": "sha512-Q8qQfPiZ+THO/3ZrOrO0cJJKfpYCagtMUkXbnEfmgUjwXg6z/
WBeOyS9APBBPCTSiDV+s4SwQGu8yFsiMRIudg==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "cssesc": "^3.0.0",
        "util-deprecate": "^1.0.2"
      },
      "engines": {
        "node": ">=4"
      }
    },
    "node_modules/postcss-value-parser": {
      "version": "4.2.0",
      "resolved": "https://registry.npmjs.org/postcss-value-parser/-/postcss-
value-parser-4.2.0.tgz",
      "integrity": "sha512-1NNCs6uurfkVbeXG4S8JFT9t19m45ICnif8zWLd5oPSZ50QnwMfK+H3
jv408d4jw/7Bttv5axS5IiHoLaVNHeQ==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/prelude-ls": {
      "version": "1.2.1",
      "resolved": "https://registry.npmjs.org/prelude-ls/-/prelude-ls-1.2.1.tgz",
      "integrity": "sha512-vkcDPrRZo1QZLbn5RLGPpg/WmIQ65qoWWhcGKf/
b5eplkkarX0m9z8ppCat4mlOqUsWpyNuYgO3VRyrYHSzX5g==",
      "dev": true,
```

```
      "license": "MIT",
      "engines": {
        "node": ">= 0.8.0"
      }
    },
    "node_modules/proxy-from-env": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/proxy-from-env/-/proxy-from-
env-1.1.0.tgz",
      "integrity": "sha512-D+zkORCbA9f1tdWRK0RaCR3GPv50cMxcrz4X8k5LTSUD1Dkw47mKJEZ
QNunItRTkWwgtaUSo1RVFRIG9ZXiFYg==",
      "license": "MIT"
    },
    "node_modules/punycode": {
      "version": "2.3.1",
      "resolved": "https://registry.npmjs.org/punycode/-/punycode-2.3.1.tgz",
      "integrity": "sha512-vYt7UD1U9Wg6138shLtLOvdAu+8DsC/
ilFtEVHcH+wydcSpNE20AfSOduf6MkRFahL5FY7X1oU7nKVZFtfq8Fg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=6"
      }
    },
    "node_modules/queue-microtask": {
      "version": "1.2.3",
      "resolved": "https://registry.npmjs.org/queue-microtask/-/queue-
microtask-1.2.3.tgz",
      "integrity": "sha512-NuaNSa6flKT5JaSYQzJok04JzTL1CA6aGhv5rfLW3PgqA+M2ChpZQnA
C8h8i4ZFkBS8X5RqkDBHA7r4hej3K9A==",
      "dev": true,
      "funding": [
        {
          "type": "github",
          "url": "https://github.com/sponsors/feross"
        },
        {
          "type": "patreon",
          "url": "https://www.patreon.com/feross"
        },
        {
          "type": "consulting",
          "url": "https://feross.org/support"
        }
      ],
      "license": "MIT"
    },
    "node_modules/react": {
      "version": "18.3.1",
      "resolved": "https://registry.npmjs.org/react/-/react-18.3.1.tgz",
      "integrity": "sha512-
wS+hAgJShR0KhEvPJArfuPVN1+Hz1t0Y6n5jLrGQbkb4urgPE/0Rve+1kMB1v/
oWgHgm4WIcV+i7F2pTVj+2iQ==",
      "license": "MIT",
      "dependencies": {
        "loose-envify": "^1.1.0"
      },
      "engines": {
        "node": ">=0.10.0"
```

```
      }
    },
    "node_modules/react-chartjs-2": {
      "version": "5.3.0",
      "resolved": "https://registry.npmjs.org/react-chartjs-2/-/react-
chartjs-2-5.3.0.tgz",
      "integrity": "sha512-UfZZFnDsERI3c3CZGxzvNJd02SHjaSJ8kgW1djn65H1KK8rehwTjyrR
KOG3VTMG8wtHZ5rgAO5oTHtHi9GCCmw==",
      "license": "MIT",
      "peerDependencies": {
        "chart.js": "^4.1.1",
        "react": "^16.8.0 || ^17.0.0 || ^18.0.0 || ^19.0.0"
      }
    },
    "node_modules/react-dom": {
      "version": "18.3.1",
      "resolved": "https://registry.npmjs.org/react-dom/-/react-dom-18.3.1.tgz",
      "integrity": "sha512-5m4nQKp+rZRb09LNH59GM4BxTh9251/
ylbKIbpe7TpGxfJ+9kv6BLkLBXIjjspbgbnIBNqlI23tRnTWT0snUIw==",
      "license": "MIT",
      "dependencies": {
        "loose-envify": "^1.1.0",
        "scheduler": "^0.23.2"
      },
      "peerDependencies": {
        "react": "^18.3.1"
      }
    },
    "node_modules/react-hot-toast": {
      "version": "2.5.2",
      "resolved": "https://registry.npmjs.org/react-hot-toast/-/react-hot-
toast-2.5.2.tgz",
      "integrity": "sha512-Tun3BbCxzmXXM7C+NI4qiv6lT0uwGh4oAfeJyNOjYUejTsm35mK9iCa
YLGv8cBz9L5YxZLx/2ii7zsIwPtPUdw==",
      "license": "MIT",
      "dependencies": {
        "csstype": "^3.1.3",
        "goober": "^2.1.16"
      },
      "engines": {
        "node": ">=10"
      },
      "peerDependencies": {
        "react": ">=16",
        "react-dom": ">=16"
      }
    },
    "node_modules/react-icons": {
      "version": "4.12.0",
      "resolved": "https://registry.npmjs.org/react-icons/-/react-
icons-4.12.0.tgz",
      "integrity": "sha512-IBaDuHiShdZqmfc/
TwHu6+d6k2ltNCf3AszxNmjJc1KUfXdEeRJOKyNvLmAHaarhzGmTSVygNdyu8/opXv2gaw==",
      "license": "MIT",
      "peerDependencies": {
        "react": "*"
      }
    },
    "node_modules/react-image-crop": {
```

```
      "version": "11.0.10",
      "resolved": "https://registry.npmjs.org/react-image-crop/-/react-image-
crop-11.0.10.tgz",
      "integrity": "sha512-+5FfDXUgYLLqBh1Y/
uQhIycpHCbXkI50a+nbfkB1C0xXXUTwkisHDo2QCB1SQJyHCqIuia4FeyReqXuMDKWQTQ==",
      "license": "ISC",
      "peerDependencies": {
        "react": ">=16.13.1"
      }
    },
    "node_modules/react-redux": {
      "version": "9.2.0",
      "resolved": "https://registry.npmjs.org/react-redux/-/react-
redux-9.2.0.tgz",
      "integrity": "sha512-ROY9fvHhwOD9ySfrF0wmvu//bKCQ6AeZZq1nJNtbDC+kk5DuSuNX/
n6YWYF/SYy7bSba4D4FSz8DJeKY/S/r+g==",
      "license": "MIT",
      "dependencies": {
        "@types/use-sync-external-store": "^0.0.6",
        "use-sync-external-store": "^1.4.0"
      },
      "peerDependencies": {
        "@types/react": "^18.2.25 || ^19",
        "react": "^18.0 || ^19",
        "redux": "^5.0.0"
      },
      "peerDependenciesMeta": {
        "@types/react": {
          "optional": true
        },
        "redux": {
          "optional": true
        }
      }
    },
    "node_modules/react-router": {
      "version": "6.30.1",
      "resolved": "https://registry.npmjs.org/react-router/-/react-
router-6.30.1.tgz",
      "integrity": "sha512-X1m21aEmxGXqENEPG3T6u0Th7g0aS4ZmoNynhbs+Cn+q+QGTLt+d5IQ
2bHAXKzKcxGJjxACpVbnYQSCRcfxHlQ==",
      "license": "MIT",
      "dependencies": {
        "@remix-run/router": "1.23.0"
      },
      "engines": {
        "node": ">=14.0.0"
      },
      "peerDependencies": {
        "react": ">=16.8"
      }
    },
    "node_modules/react-router-dom": {
      "version": "6.30.1",
      "resolved": "https://registry.npmjs.org/react-router-dom/-/react-router-
dom-6.30.1.tgz",
      "integrity": "sha512-
llKsgOkZdbPU1Eg3zK8lCn+sjD9wMRZZPuzmdWWX5SUs8OFkN5HnFVC0u5KMeMaC9aoancFI/
KoLuKPqN+hxHw==",
```

```
      "license": "MIT",
      "dependencies": {
        "@remix-run/router": "1.23.0",
        "react-router": "6.30.1"
      },
      "engines": {
        "node": ">=14.0.0"
      },
      "peerDependencies": {
        "react": ">=16.8",
        "react-dom": ">=16.8"
      }
    },
    "node_modules/read-cache": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/read-cache/-/read-cache-1.0.0.tgz",
      "integrity": "sha512-Owdv/Ft7IjOgm/
i0xvNDZ1LrRANRfew4b2prF3OWMQLxLfu3bS8FVhCsrSCMK4lR56Y9ya+AThoTpDCTxCmpRA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "pify": "^2.3.0"
      }
    },
    "node_modules/readdirp": {
      "version": "3.6.0",
      "resolved": "https://registry.npmjs.org/readdirp/-/readdirp-3.6.0.tgz",
      "integrity": "sha512-hOS089on8RduqdbhvQ5Z37A0ESjsqz6qnRcffsMU3495FuTdqSm+7bh
J29JvIOsBDEEnan5DPu9t3To9VRlMzA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "picomatch": "^2.2.1"
      },
      "engines": {
        "node": ">=8.10.0"
      }
    },
    "node_modules/resolve": {
      "version": "1.22.10",
      "resolved": "https://registry.npmjs.org/resolve/-/resolve-1.22.10.tgz",
      "integrity": "sha512-NPRy+/
ncIMeDlTAsuqwKIiferiawhefFJtkNSW0qZJEqMEb+qBt/77B/
jGeeek+F0uOeN05CDa6HXbbIgtVX4w==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "is-core-module": "^2.16.0",
        "path-parse": "^1.0.7",
        "supports-preserve-symlinks-flag": "^1.0.0"
      },
      "bin": {
        "resolve": "bin/resolve"
      },
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
```

```
      }
    },
    "node_modules/reusify": {
      "version": "1.1.0",
      "resolved": "https://registry.npmjs.org/reusify/-/reusify-1.1.0.tgz",
      "integrity": "sha512-g6QUff04oZpHs0eG5p83rFLhHeV00ug/
Yf9nZM6fLeUrPguBTkTQOdpAWWspMh55TZfVQDPaN3NQJfbVRAxdIw==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "iojs": ">=1.0.0",
        "node": ">=0.10.0"
      }
    },
    "node_modules/run-parallel": {
      "version": "1.2.0",
      "resolved": "https://registry.npmjs.org/run-parallel/-/run-
parallel-1.2.0.tgz",
      "integrity": "sha512-5l4VyZR86LZ/lDxZTR6jqL8AFE2S0IFLMP26AbjsLVADxHdhB/
c0GUsH+y39UfCi3dzz8OlQuPmnaJOMoDHQBA==",
      "dev": true,
      "funding": [
        {
          "type": "github",
          "url": "https://github.com/sponsors/feross"
        },
        {
          "type": "patreon",
          "url": "https://www.patreon.com/feross"
        },
        {
          "type": "consulting",
          "url": "https://feross.org/support"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "queue-microtask": "^1.2.2"
      }
    },
    "node_modules/scheduler": {
      "version": "0.23.2",
      "resolved": "https://registry.npmjs.org/scheduler/-/scheduler-0.23.2.tgz",
      "integrity": "sha512-UOShsPwz7NrMUqhR6t0hWjFduvOzbtv7toDH1/
hIrfRNIDBnnBWd0CwJTGvTpngVlmwGCdP9/Zl/tVrDqcuYzQ==",
      "license": "MIT",
      "dependencies": {
        "loose-envify": "^1.1.0"
      }
    },
    "node_modules/shebang-command": {
      "version": "2.0.0",
      "resolved": "https://registry.npmjs.org/shebang-command/-/shebang-
command-2.0.0.tgz",
      "integrity": "sha512-kHxr2zZpYtdmrN1qDjrrX/
Z1rR1kG8Dx+gkpK1G4eXmvXswmcE1hTWBWYUzlraYw1/yZp6YuDY77YtvbN0dmDA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
```

```
        "shebang-regex": "^3.0.0"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/shebang-regex": {
      "version": "3.0.0",
      "resolved": "https://registry.npmjs.org/shebang-regex/-/shebang-
regex-3.0.0.tgz",
      "integrity": "sha512-7+
+dFhtcx3353uBaq8DDR4NuxBetBzC7ZQOhmTQInHEd6bSrXdiEyzCvG07Z44UYdLShWUyXt5M/
yhz8ekcb1A==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/signal-exit": {
      "version": "4.1.0",
      "resolved": "https://registry.npmjs.org/signal-exit/-/signal-
exit-4.1.0.tgz",
      "integrity": "sha512-bzyZ1e88w9O1iNJbKnOlvYTrWPDl46O1bG0D3XInv+9tkPrxrN8jUUT
iFlDkkmKWgn1M6CfIA13SuGqOa9Korw==",
      "dev": true,
      "license": "ISC",
      "engines": {
        "node": ">=14"
      },
      "funding": {
        "url": "https://github.com/sponsors/isaacs"
      }
    },
    "node_modules/socket.io-client": {
      "version": "4.8.1",
      "resolved": "https://registry.npmjs.org/socket.io-client/-/socket.io-
client-4.8.1.tgz",
      "integrity": "sha512-hJVXfu3E28NmzGk8o1sHhN3om52tRvwYeidbj7xKy2eIIse5IoKX3US
lS6Tqt3BHAtflLIkCQBkzVrEEfWUyYQ==",
      "license": "MIT",
      "dependencies": {
        "@socket.io/component-emitter": "~3.1.0",
        "debug": "~4.3.2",
        "engine.io-client": "~6.6.1",
        "socket.io-parser": "~4.2.4"
      },
      "engines": {
        "node": ">=10.0.0"
      }
    },
    "node_modules/socket.io-client/node_modules/debug": {
      "version": "4.3.7",
      "resolved": "https://registry.npmjs.org/debug/-/debug-4.3.7.tgz",
      "integrity": "sha512-Er2nc/
H7RrMXZBFCEim6TCmMk02Z8vLC2Rbi1KEBggpo0fS6l0S1nnapwmIi3yW/
+GOJap1Krg4w0Hg80oCqgQ==",
      "license": "MIT",
      "dependencies": {
```

```
        "ms": "^2.1.3"
      },
      "engines": {
        "node": ">=6.0"
      },
      "peerDependenciesMeta": {
        "supports-color": {
          "optional": true
        }
      }
    },
    "node_modules/socket.io-parser": {
      "version": "4.2.4",
      "resolved": "https://registry.npmjs.org/socket.io-parser/-/socket.io-
parser-4.2.4.tgz",
      "integrity": "sha512-/GbIKmo8ioc+NIWIhwdecY0ge+qVBSMdgxGygevmdHj24bsfgtCmcUU
cQ5ZzcylGFHsN3k4HB4Cgkl96KVnuew==",
      "license": "MIT",
      "dependencies": {
        "@socket.io/component-emitter": "~3.1.0",
        "debug": "~4.3.1"
      },
      "engines": {
        "node": ">=10.0.0"
      }
    },
    "node_modules/socket.io-parser/node_modules/debug": {
      "version": "4.3.7",
      "resolved": "https://registry.npmjs.org/debug/-/debug-4.3.7.tgz",
      "integrity": "sha512-Er2nc/
H7RrMXZBFCEim6TCmMk02Z8vLC2Rbi1KEBggpo0fS6l0S1nnapwmIi3yW/
+GOJap1Krg4w0Hg80oCqgQ==",
      "license": "MIT",
      "dependencies": {
        "ms": "^2.1.3"
      },
      "engines": {
        "node": ">=6.0"
      },
      "peerDependenciesMeta": {
        "supports-color": {
          "optional": true
        }
      }
    },
    "node_modules/source-map-js": {
      "version": "1.2.1",
      "resolved": "https://registry.npmjs.org/source-map-js/-/source-map-
js-1.2.1.tgz",
      "integrity": "sha512-UXWMKhLOwVKb728IUtQPXxfYU+usdybtUrK/8uGE8CQMvrhOpwvzDBw
j0QhSL7MQc7vIsISBG8VQ8+IDQxpfQA==",
      "dev": true,
      "license": "BSD-3-Clause",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/source-map-support": {
      "version": "0.5.21",
```

```
      "resolved": "https://registry.npmjs.org/source-map-support/-/source-map-
support-0.5.21.tgz",
      "integrity": "sha512-uBHU3L3czsIyYXKX88fdrGovxdSCoTGDRZ6SYXtSRxLZUzHg5P/66Ht
6uoUlHu9EZod+inXhKo3qQgwXUT/y1w==",
      "dev": true,
      "license": "MIT",
      "optional": true,
      "peer": true,
      "dependencies": {
        "buffer-from": "^1.0.0",
        "source-map": "^0.6.0"
      }
    },
    "node_modules/source-map-support/node_modules/source-map": {
      "version": "0.6.1",
      "resolved": "https://registry.npmjs.org/source-map/-/source-map-0.6.1.tgz",
      "integrity": "sha512-UjgapumWlbMhkBgzT7Ykc5YXUT46F0iKu8SGXq0bcwP5dz/
h0Plj6enJqjz1Zbq2l5WaqYnrVbwWOWMyF3F47g==",
      "dev": true,
      "license": "BSD-3-Clause",
      "optional": true,
      "peer": true,
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/ssf": {
      "version": "0.11.2",
      "resolved": "https://registry.npmjs.org/ssf/-/ssf-0.11.2.tgz",
      "integrity": "sha512-+idbmIXoYET47hH+d7dfm2epdOMUDjqcB4648sTZ+t2JwoyBFL/
insLfB/racrDmsKB3diwsDA696pZMieAC5g==",
      "license": "Apache-2.0",
      "dependencies": {
        "frac": "~1.1.2"
      },
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/string-width": {
      "version": "5.1.2",
      "resolved": "https://registry.npmjs.org/string-width/-/string-
width-5.1.2.tgz",
      "integrity": "sha512-HnLOCR3vjcY8beoNLtcjZ5/
nxn2afmME6lhrDrebokqMap+XbeW8n9TXpPDOqdGK5qcI3oT0GKTW6wC7EMiVqA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "eastasianwidth": "^0.2.0",
        "emoji-regex": "^9.2.2",
        "strip-ansi": "^7.0.1"
      },
      "engines": {
        "node": ">=12"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
```

```
    "node_modules/string-width-cjs": {
      "name": "string-width",
      "version": "4.2.3",
      "resolved": "https://registry.npmjs.org/string-width/-/string-
width-4.2.3.tgz",
      "integrity": "sha512-
wKyQRQpjJ0sIp62ErSZdGsjMJWsap5oRNihHhu6G7JVO/9jIB6UyevL+tXuOqrng8j/
cxKTWyWUwvSTriiZz/g==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "emoji-regex": "^8.0.0",
        "is-fullwidth-code-point": "^3.0.0",
        "strip-ansi": "^6.0.1"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/string-width-cjs/node_modules/ansi-regex": {
      "version": "5.0.1",
      "resolved": "https://registry.npmjs.org/ansi-regex/-/ansi-regex-5.0.1.tgz",
      "integrity": "sha512-
quJQXlTSUGL2LH9SUXo8VwsY4soanhgo6LNSm84E1LBcE8s3O0wpdiRzyR9z/
ZZJMlMWv37qOOb9pdJlMUEKFQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/string-width-cjs/node_modules/emoji-regex": {
      "version": "8.0.0",
      "resolved": "https://registry.npmjs.org/emoji-regex/-/emoji-
regex-8.0.0.tgz",
      "integrity": "sha512-MSjYzcWNOA0ewAHpz0MxpYFvwg6yjy1NG3xteoqz644VCo/RPgnr1/
GGt+ic3iJTzQ8Eu3TdM14SawnVUmGE6A==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/string-width-cjs/node_modules/strip-ansi": {
      "version": "6.0.1",
      "resolved": "https://registry.npmjs.org/strip-ansi/-/strip-ansi-6.0.1.tgz",
      "integrity": "sha512-Y38VPSHcqkFrCpFnQ9vuSXmquuv5oXOKpGeT6aGrr3o3Gc9AlVa6JBf
USOCnbxGGZF+/0ooI7KrPuUSztUdU5A==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ansi-regex": "^5.0.1"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/strip-ansi": {
      "version": "7.1.0",
      "resolved": "https://registry.npmjs.org/strip-ansi/-/strip-ansi-7.1.0.tgz",
      "integrity": "sha512-iq6eVVI64nQQTRYq2KtEg2d2uU7LElhTJwsH4YzIHZshxlgZms/
wIc4VoDQTlG/IvVIrBKG06CrZnp0qv7hkcQ==",
```

```
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ansi-regex": "^6.0.1"
      },
      "engines": {
        "node": ">=12"
      },
      "funding": {
        "url": "https://github.com/chalk/strip-ansi?sponsor=1"
      }
    },
    "node_modules/strip-ansi-cjs": {
      "name": "strip-ansi",
      "version": "6.0.1",
      "resolved": "https://registry.npmjs.org/strip-ansi/-/strip-ansi-6.0.1.tgz",
      "integrity": "sha512-Y38VPSHcqkFrCpFnQ9vuSXmquuv5oXOKpGeT6aGrr3o3Gc9AlVa6JBf
USOCnbxGGZF+/0ooI7KrPuUSztUdU5A==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ansi-regex": "^5.0.1"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/strip-ansi-cjs/node_modules/ansi-regex": {
      "version": "5.0.1",
      "resolved": "https://registry.npmjs.org/ansi-regex/-/ansi-regex-5.0.1.tgz",
      "integrity": "sha512-
quJQXlTSUGL2LH9SUXo8VwsY4soanhgo6LNSm84E1LBcE8s3O0wpdiRzyR9z/
ZZJMlMWv37qOOb9pdJlMUEKFQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/strip-json-comments": {
      "version": "3.1.1",
      "resolved": "https://registry.npmjs.org/strip-json-comments/-/strip-json-
comments-3.1.1.tgz",
      "integrity": "sha512-6fPc+R4ihwqP6N/
aIv2f1gMH8lOVtWQHoqC4yK6oSDVVocumAsfCqjkXnqiYMhmMwS/mEHLp7Vehlt3ql6lEig==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    },
    "node_modules/sucrase": {
      "version": "3.35.0",
      "resolved": "https://registry.npmjs.org/sucrase/-/sucrase-3.35.0.tgz",
      "integrity": "sha512-8EbVDiu9iN/nESwxeSxDKe0dunta1GOlHufmSSXxMD2z2/
tMZpDMpvXQGsc+ajGo8y2uYUmixaSRUc/QPoQ0GA==",
```

```
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@jridgewell/gen-mapping": "^0.3.2",
        "commander": "^4.0.0",
        "glob": "^10.3.10",
        "lines-and-columns": "^1.1.6",
        "mz": "^2.7.0",
        "pirates": "^4.0.1",
        "ts-interface-checker": "^0.1.9"
      },
      "bin": {
        "sucrase": "bin/sucrase",
        "sucrase-node": "bin/sucrase-node"
      },
      "engines": {
        "node": ">=16 || 14 >=14.17"
      }
    },
    "node_modules/supports-color": {
      "version": "7.2.0",
      "resolved": "https://registry.npmjs.org/supports-color/-/supports-
color-7.2.0.tgz",
      "integrity": "sha512-qpCAvRl9stuOHveKsn7HncJRvv501qIacKzQlO/
+Lwxc9+0q2wLyv4Dfvt80/DPn2pqOBsJdDiogXGR9+OvwRw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "has-flag": "^4.0.0"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/supports-preserve-symlinks-flag": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/supports-preserve-symlinks-flag/-/
supports-preserve-symlinks-flag-1.0.0.tgz",
      "integrity": "sha512-ot0WnXS9fgdkgIcePe6RHNk1WA8+muPa6cSjeR3V8K27q9BB1rTE3R1
p7Hv0z1ZyAc8s6Vvv8DIyWf681MAt0w==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">= 0.4"
      },
      "funding": {
        "url": "https://github.com/sponsors/ljharb"
      }
    },
    "node_modules/tailwindcss": {
      "version": "3.4.17",
      "resolved": "https://registry.npmjs.org/tailwindcss/-/
tailwindcss-3.4.17.tgz",
      "integrity": "sha512-w33E2aCvSDP0tW9RZuNXadXlkHXqFzSkQew/aIa2i/
Sj8fThxwovwlXHSPXTbAHwEIhBFXAedUhP2tueAKP8Og==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@alloc/quick-lru": "^5.2.0",
```

```
        "arg": "^5.0.2",
        "chokidar": "^3.6.0",
        "didyoumean": "^1.2.2",
        "dlv": "^1.1.3",
        "fast-glob": "^3.3.2",
        "glob-parent": "^6.0.2",
        "is-glob": "^4.0.3",
        "jiti": "^1.21.6",
        "lilconfig": "^3.1.3",
        "micromatch": "^4.0.8",
        "normalize-path": "^3.0.0",
        "object-hash": "^3.0.0",
        "picocolors": "^1.1.1",
        "postcss": "^8.4.47",
        "postcss-import": "^15.1.0",
        "postcss-js": "^4.0.1",
        "postcss-load-config": "^4.0.2",
        "postcss-nested": "^6.2.0",
        "postcss-selector-parser": "^6.1.2",
        "resolve": "^1.22.8",
        "sucrase": "^3.35.0"
      },
      "bin": {
        "tailwind": "lib/cli.js",
        "tailwindcss": "lib/cli.js"
      },
      "engines": {
        "node": ">=14.0.0"
      }
    },
    "node_modules/tailwindcss/node_modules/jiti": {
      "version": "1.21.7",
      "resolved": "https://registry.npmjs.org/jiti/-/jiti-1.21.7.tgz",
      "integrity": "sha512-/imKNG4EbWNrVjoNC/1H5/9GFy+tqjGBHCaSsN+P2RnPqjsLmv6UD3E
j+Kj8nBWaRAwyk7kK5ZUc+OEatnTR3A==",
      "dev": true,
      "license": "MIT",
      "bin": {
        "jiti": "bin/jiti.js"
      }
    },
    "node_modules/terser": {
      "version": "5.39.0",
      "resolved": "https://registry.npmjs.org/terser/-/terser-5.39.0.tgz",
      "integrity": "sha512-LBAhFyLho16harJoWMg/
nZsQYgTrg5jXOn2nCYjRUcZZEdE3qa2zb8QEDRUGVZBW4rlazf2fxkg8tztybTaqWw==",
      "dev": true,
      "license": "BSD-2-Clause",
      "optional": true,
      "peer": true,
      "dependencies": {
        "@jridgewell/source-map": "^0.3.3",
        "acorn": "^8.8.2",
        "commander": "^2.20.0",
        "source-map-support": "~0.5.20"
      },
      "bin": {
        "terser": "bin/terser"
      },
```

```
      "engines": {
        "node": ">=10"
      }
    },
    "node_modules/terser/node_modules/commander": {
      "version": "2.20.3",
      "resolved": "https://registry.npmjs.org/commander/-/commander-2.20.3.tgz",
      "integrity": "sha512-
GpVkmM8vF2vQUkj2LvZmD35JxeJOLCwJ9cUkugyk2nuhbv3+mJvpLYYt+0+USMxE+oj+ey/lJEnhZw75x/
OMcQ==",
      "dev": true,
      "license": "MIT",
      "optional": true,
      "peer": true
    },
    "node_modules/thenify": {
      "version": "3.3.1",
      "resolved": "https://registry.npmjs.org/thenify/-/thenify-3.3.1.tgz",
      "integrity": "sha512-RVZSIV5IG10Hk3enotrhvz0T9em6cyHBLkH/
YAZuKqd8hRkKhSfCGIcP2KUY0EPxndzANBmNllzWPwak+bheSw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "any-promise": "^1.0.0"
      }
    },
    "node_modules/thenify-all": {
      "version": "1.6.0",
      "resolved": "https://registry.npmjs.org/thenify-all/-/thenify-
all-1.6.0.tgz",
      "integrity": "sha512-RNxQH/qI8/t3thXJDwcstUO4zeqo64+Uy/
+sNVRBx4Xn2OX+OZ9oP+iJnNFqplFra2ZUVeKCSa2oVWi3T4uVmA==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "thenify": ">= 3.1.0 < 4"
      },
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/tinyglobby": {
      "version": "0.2.13",
      "resolved": "https://registry.npmjs.org/tinyglobby/-/tinyglobby-0.2.13.tgz",
      "integrity": "sha512-mEwzpUgrLySlveBwEVDMKk5B57bhLPYovRfPAXD5gA/98Opn0rCDj3G
tLwFvCvH5RK9uPCExUROW5NjDwvqkxw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "fdir": "^6.4.4",
        "picomatch": "^4.0.2"
      },
      "engines": {
        "node": ">=12.0.0"
      },
      "funding": {
        "url": "https://github.com/sponsors/SuperchupuDev"
      }
    },
```

```
    "node_modules/tinyglobby/node_modules/fdir": {
      "version": "6.4.4",
      "resolved": "https://registry.npmjs.org/fdir/-/fdir-6.4.4.tgz",
      "integrity":
"sha512-1NZP+GK4GfuAv3PqKvxQRDMjdSRZjnkq7KfhlNrCNNlZ0ygQFpebfrnfnq/
W7fpUnAv9aGWmY1zKx7FYL3gwhg==",
      "dev": true,
      "license": "MIT",
      "peerDependencies": {
        "picomatch": "^3 || ^4"
      },
      "peerDependenciesMeta": {
        "picomatch": {
          "optional": true
        }
      }
    },
    "node_modules/tinyglobby/node_modules/picomatch": {
      "version": "4.0.2",
      "resolved": "https://registry.npmjs.org/picomatch/-/picomatch-4.0.2.tgz",
      "integrity": "sha512-M7BAV6Rlcy5u+m6oPhAPFgJTzAioX/6B0DxyvDlo9l8+T3nLKbrczg2
WLUyzd45L8RqfUMyGPzekbMvX2Ldkwg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=12"
      },
      "funding": {
        "url": "https://github.com/sponsors/jonschlinkert"
      }
    },
    "node_modules/to-regex-range": {
      "version": "5.0.1",
      "resolved": "https://registry.npmjs.org/to-regex-range/-/to-regex-
range-5.0.1.tgz",
      "integrity": "sha512-65P7iz6X5yEr1cwcgvQxbbIw7Uk3gOy5dIdtZ4rDveLqhrdJP+Li/
Hx6tyK0NEb+2GCyneCMJiGqrADCSNk8sQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "is-number": "^7.0.0"
      },
      "engines": {
        "node": ">=8.0"
      }
    },
    "node_modules/ts-interface-checker": {
      "version": "0.1.13",
      "resolved": "https://registry.npmjs.org/ts-interface-checker/-/ts-interface-
checker-0.1.13.tgz",
      "integrity": "sha512-Y/arvbn+rrz3JCKl9C4kVNfTfSm2/
mEp5FSz5EsZSANGPSlQrpRI5M4PKF+mJnE52jOO90PnPSc3Ur3bTQw0gA==",
      "dev": true,
      "license": "Apache-2.0"
    },
    "node_modules/type-check": {
      "version": "0.4.0",
      "resolved": "https://registry.npmjs.org/type-check/-/type-check-0.4.0.tgz",
      "integrity": "sha512-XleUoc9uwGXqjWwXaUTZAmzMcFZ5858QA2vvx1Ur5xIcixXIP+8LnFD
```

```
gRplU30us6teqdlskFfu+ae4K79Ooew==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "prelude-ls": "^1.2.1"
      },
      "engines": {
        "node": ">= 0.8.0"
      }
    },
    "node_modules/undici-types": {
      "version": "6.21.0",
      "resolved": "https://registry.npmjs.org/undici-types/-/undici-
types-6.21.0.tgz",
      "integrity": "sha512-iwDZqg0QAGrg9Rav5H4n0M64c3mkR59cJ6wQp+7C4nI0gsmExaedaYL
NO44eT4AtBBwjbTiGPMlt2Md0T9H9JQ==",
      "dev": true,
      "license": "MIT",
      "optional": true,
      "peer": true
    },
    "node_modules/update-browserslist-db": {
      "version": "1.1.3",
      "resolved": "https://registry.npmjs.org/update-browserslist-db/-/update-
browserslist-db-1.1.3.tgz",
      "integrity": "sha512-UxhIZQ+QInVdunkDAaiazvvT/
+fXL5Osr0JZlJulepYu6Jd7qJtDZjlur0emRlT71EN3ScPoE7gvsuIKKNavKw==",
      "dev": true,
      "funding": [
        {
          "type": "opencollective",
          "url": "https://opencollective.com/browserslist"
        },
        {
          "type": "tidelift",
          "url": "https://tidelift.com/funding/github/npm/browserslist"
        },
        {
          "type": "github",
          "url": "https://github.com/sponsors/ai"
        }
      ],
      "license": "MIT",
      "dependencies": {
        "escalade": "^3.2.0",
        "picocolors": "^1.1.1"
      },
      "bin": {
        "update-browserslist-db": "cli.js"
      },
      "peerDependencies": {
        "browserslist": ">= 4.21.0"
      }
    },
    "node_modules/uri-js": {
      "version": "4.4.1",
      "resolved": "https://registry.npmjs.org/uri-js/-/uri-js-4.4.1.tgz",
      "integrity": "sha512-7rKUyy33Q1yc98pQ1DAmLtwX109F7TIfWlW1Ydo8Wl1ii1SeHieeh0H
HfPeL2fMXK6z0s8ecKs9frCuLJvndBg==",
```

```
      "dev": true,
      "license": "BSD-2-Clause",
      "dependencies": {
        "punycode": "^2.1.0"
      }
    },
    "node_modules/use-sync-external-store": {
      "version": "1.5.0",
      "resolved": "https://registry.npmjs.org/use-sync-external-store/-/use-sync-
external-store-1.5.0.tgz",
      "integrity": "sha512-Rb46I4cGGVBmjamjphe8L/
UnvJD+uPPtTkNvX5mZgqdbavhI4EbgIWJiIHXJ8bc/i9EQGPRh4DwEURJ552Do0A==",
      "license": "MIT",
      "peerDependencies": {
        "react": "^16.8.0 || ^17.0.0 || ^18.0.0 || ^19.0.0"
      }
    },
    "node_modules/util-deprecate": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/util-deprecate/-/util-
deprecate-1.0.2.tgz",
      "integrity": "sha512-
EPD5q1uXyFxJpCrLnCc1nHnq3gOa6DZBocAIiI2TaSCA7VCJ1UJDMagCzIkXNsUYfD1daK//
LTEQ8xiIbrHtcw==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/vite": {
      "version": "6.3.3",
      "resolved": "https://registry.npmjs.org/vite/-/vite-6.3.3.tgz",
      "integrity": "sha512-5nXH+QsELbFKhsEfWLkHrvgRpTdGJzqOZ+utSdmPTvwHmvU6ITTm3xx
+mRusihkcI8GeC7lCDyn3kDtiki9scw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "esbuild": "^0.25.0",
        "fdir": "^6.4.4",
        "picomatch": "^4.0.2",
        "postcss": "^8.5.3",
        "rollup": "^4.34.9",
        "tinyglobby": "^0.2.13"
      },
      "bin": {
        "vite": "bin/vite.js"
      },
      "engines": {
        "node": "^18.0.0 || ^20.0.0 || >=22.0.0"
      },
      "funding": {
        "url": "https://github.com/vitejs/vite?sponsor=1"
      },
      "optionalDependencies": {
        "fsevents": "~2.3.3"
      },
      "peerDependencies": {
        "@types/node": "^18.0.0 || ^20.0.0 || >=22.0.0",
        "jiti": ">=1.21.0",
        "less": "*",
        "lightningcss": "^1.21.0",
```

```
        "sass": "*",
        "sass-embedded": "*",
        "stylus": "*",
        "sugarss": "*",
        "terser": "^5.16.0",
        "tsx": "^4.8.1",
        "yaml": "^2.4.2"
      },
      "peerDependenciesMeta": {
        "@types/node": {
          "optional": true
        },
        "jiti": {
          "optional": true
        },
        "less": {
          "optional": true
        },
        "lightningcss": {
          "optional": true
        },
        "sass": {
          "optional": true
        },
        "sass-embedded": {
          "optional": true
        },
        "stylus": {
          "optional": true
        },
        "sugarss": {
          "optional": true
        },
        "terser": {
          "optional": true
        },
        "tsx": {
          "optional": true
        },
        "yaml": {
          "optional": true
        }
      }
    },
    "node_modules/vite/node_modules/fdir": {
      "version": "6.4.4",
      "resolved": "https://registry.npmjs.org/fdir/-/fdir-6.4.4.tgz",
      "integrity":
"sha512-1NZP+GK4GfuAv3PqKvxQRDMjdSRZjnkq7KfhlNrCNNlZ0ygQFpebfrnfnq/
W7fpUnAv9aGWmY1zKx7FYL3gwhg==",
      "dev": true,
      "license": "MIT",
      "peerDependencies": {
        "picomatch": "^3 || ^4"
      },
      "peerDependenciesMeta": {
        "picomatch": {
          "optional": true
        }
```

```
      }
    },
    "node_modules/vite/node_modules/picomatch": {
      "version": "4.0.2",
      "resolved": "https://registry.npmjs.org/picomatch/-/picomatch-4.0.2.tgz",
      "integrity": "sha512-M7BAV6Rlcy5u+m6oPhAPFgJTzAioX/6B0DxyvDlo9l8+T3nLKbrczg2
WLUyzd45L8RqfUMyGPzekbMvX2Ldkwg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=12"
      },
      "funding": {
        "url": "https://github.com/sponsors/jonschlinkert"
      }
    },
    "node_modules/vite/node_modules/rollup": {
      "version": "4.40.1",
      "resolved": "https://registry.npmjs.org/rollup/-/rollup-4.40.1.tgz",
      "integrity": "sha512-C5VvvgCCyfyotVITIAv+4efVytl5F7wt+/
I2i9q9GZcEXW9BP52YYOXC58igUi+LFZVHukErIIqQSWwv/M3WRw==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "@types/estree": "1.0.7"
      },
      "bin": {
        "rollup": "dist/bin/rollup"
      },
      "engines": {
        "node": ">=18.0.0",
        "npm": ">=8.0.0"
      },
      "optionalDependencies": {
        "@rollup/rollup-android-arm-eabi": "4.40.1",
        "@rollup/rollup-android-arm64": "4.40.1",
        "@rollup/rollup-darwin-arm64": "4.40.1",
        "@rollup/rollup-darwin-x64": "4.40.1",
        "@rollup/rollup-freebsd-arm64": "4.40.1",
        "@rollup/rollup-freebsd-x64": "4.40.1",
        "@rollup/rollup-linux-arm-gnueabihf": "4.40.1",
        "@rollup/rollup-linux-arm-musleabihf": "4.40.1",
        "@rollup/rollup-linux-arm64-gnu": "4.40.1",
        "@rollup/rollup-linux-arm64-musl": "4.40.1",
        "@rollup/rollup-linux-loongarch64-gnu": "4.40.1",
        "@rollup/rollup-linux-powerpc64le-gnu": "4.40.1",
        "@rollup/rollup-linux-riscv64-gnu": "4.40.1",
        "@rollup/rollup-linux-riscv64-musl": "4.40.1",
        "@rollup/rollup-linux-s390x-gnu": "4.40.1",
        "@rollup/rollup-linux-x64-gnu": "4.40.1",
        "@rollup/rollup-linux-x64-musl": "4.40.1",
        "@rollup/rollup-win32-arm64-msvc": "4.40.1",
        "@rollup/rollup-win32-ia32-msvc": "4.40.1",
        "@rollup/rollup-win32-x64-msvc": "4.40.1",
        "fsevents": "~2.3.2"
      }
    },
    "node_modules/which": {
      "version": "2.0.2",
```

```
      "resolved": "https://registry.npmjs.org/which/-/which-2.0.2.tgz",
      "integrity": "sha512-BLI3Tl1TW3Pvl70l3yq3Y64i+awpwXqsGBYWkkqMtnbXgrMD+yj7rhW
0kuEDxzJaYXGjEW5ogapKNMEKNMjibA==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "isexe": "^2.0.0"
      },
      "bin": {
        "node-which": "bin/node-which"
      },
      "engines": {
        "node": ">= 8"
      }
    },
    "node_modules/wmf": {
      "version": "1.0.2",
      "resolved": "https://registry.npmjs.org/wmf/-/wmf-1.0.2.tgz",
      "integrity": "sha512-/
p9K7bEh0Dj6WbXg4JG0xvLQmIadrner1bi45VMJTfnbVHsc7yIajZyoSoK60/
dtVBs12Fm6WkUI5/3WAVsNMw==",
      "license": "Apache-2.0",
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/word": {
      "version": "0.3.0",
      "resolved": "https://registry.npmjs.org/word/-/word-0.3.0.tgz",
      "integrity": "sha512-OELeY0Q61OXpdUfTp+oweA/vtLVg5VDOXh+3he3PNzLGG/
y0oylSOC1xRVj0+l4vQ3tj/bB1HVHv1ocXkQceFA==",
      "license": "Apache-2.0",
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/word-wrap": {
      "version": "1.2.5",
      "resolved": "https://registry.npmjs.org/word-wrap/-/word-wrap-1.2.5.tgz",
      "integrity": "sha512-BN22B5eaMMI9UMtjrGd5g5eCYPpCPDUy0FJXbYsaT5zYxjFOckS53SQ
DE3pWkVoWpHXVb3BrYcEN4Twa55B5cA==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=0.10.0"
      }
    },
    "node_modules/wrap-ansi": {
      "version": "8.1.0",
      "resolved": "https://registry.npmjs.org/wrap-ansi/-/wrap-ansi-8.1.0.tgz",
      "integrity": "sha512-si7QWI6zUMq56bESFvagtmzMdGOtoxfR+Sez11Mobfc7tm+VkUckk9b
W2UeffTGVUbOksxmSw0AA2gs8g71NCQ==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ansi-styles": "^6.1.0",
        "string-width": "^5.0.1",
        "strip-ansi": "^7.0.1"
      },
```

```
      "engines": {
        "node": ">=12"
      },
      "funding": {
        "url": "https://github.com/chalk/wrap-ansi?sponsor=1"
      }
    },
    "node_modules/wrap-ansi-cjs": {
      "name": "wrap-ansi",
      "version": "7.0.0",
      "resolved": "https://registry.npmjs.org/wrap-ansi/-/wrap-ansi-7.0.0.tgz",
      "integrity": "sha512-
YVGIj2kamLSTxw6NsZjoBxfSwsn0ycdesmc4p+Q21c5zPuZ1pl+NfxVdxPtdHvmNVOQ6XSYG4AUtyt/
Fi7D16Q==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ansi-styles": "^4.0.0",
        "string-width": "^4.1.0",
        "strip-ansi": "^6.0.0"
      },
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/chalk/wrap-ansi?sponsor=1"
      }
    },
    "node_modules/wrap-ansi-cjs/node_modules/ansi-regex": {
      "version": "5.0.1",
      "resolved": "https://registry.npmjs.org/ansi-regex/-/ansi-regex-5.0.1.tgz",
      "integrity": "sha512-
quJQXlTSUGL2LH9SUXo8VwsY4soanhgo6LNSm84E1LBcE8s3O0wpdiRzyR9z/
ZZJMlMWv37qOOb9pdJlMUEKFQ==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/wrap-ansi-cjs/node_modules/emoji-regex": {
      "version": "8.0.0",
      "resolved": "https://registry.npmjs.org/emoji-regex/-/emoji-
regex-8.0.0.tgz",
      "integrity": "sha512-MSjYzcWNOA0ewAHpz0MxpYFvwg6yjy1NG3xteoqz644VCo/RPgnr1/
GGt+ic3iJTzQ8Eu3TdM14SawnVUmGE6A==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/wrap-ansi-cjs/node_modules/string-width": {
      "version": "4.2.3",
      "resolved": "https://registry.npmjs.org/string-width/-/string-
width-4.2.3.tgz",
      "integrity": "sha512-
wKyQRQpjJ0sIp62ErSZdGsjMJWsap5oRNihHhu6G7JVO/9jIB6UyevL+tXuOqrng8j/
cxKTWyWUwvSTriiZz/g==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
```

```
        "emoji-regex": "^8.0.0",
        "is-fullwidth-code-point": "^3.0.0",
        "strip-ansi": "^6.0.1"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/wrap-ansi-cjs/node_modules/strip-ansi": {
      "version": "6.0.1",
      "resolved": "https://registry.npmjs.org/strip-ansi/-/strip-ansi-6.0.1.tgz",
      "integrity": "sha512-Y38VPSHcqkFrCpFnQ9vuSXmquuv5oXOKpGeT6aGrr3o3Gc9AlVa6JBf
USOCnbxGGZF+/0ooI7KrPuUSztUdU5A==",
      "dev": true,
      "license": "MIT",
      "dependencies": {
        "ansi-regex": "^5.0.1"
      },
      "engines": {
        "node": ">=8"
      }
    },
    "node_modules/wrap-ansi/node_modules/ansi-styles": {
      "version": "6.2.1",
      "resolved": "https://registry.npmjs.org/ansi-styles/-/ansi-
styles-6.2.1.tgz",
      "integrity": "sha512-
bN798gFfQX+viw3R7yrGWRqnrN2oRkEkUjjl4JNn4E8GxxbjtG3FbrEIIY3l8/
hrwUwIeCZvi4QuOTP4MErVug==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=12"
      },
      "funding": {
        "url": "https://github.com/chalk/ansi-styles?sponsor=1"
      }
    },
    "node_modules/ws": {
      "version": "8.17.1",
      "resolved": "https://registry.npmjs.org/ws/-/ws-8.17.1.tgz",
      "integrity": "sha512-6XQFvXTkbfUOZOKKILFG1PDK2NDQs4azKQl26T0YS5CxqWLgXajbPZ+
h4gZekJyRqFU8pvnbAbbs/3TgRPy+GQ==",
      "license": "MIT",
      "engines": {
        "node": ">=10.0.0"
      },
      "peerDependencies": {
        "bufferutil": "^4.0.1",
        "utf-8-validate": ">=5.0.2"
      },
      "peerDependenciesMeta": {
        "bufferutil": {
          "optional": true
        },
        "utf-8-validate": {
          "optional": true
        }
      }
```

```
    },
    "node_modules/xlsx": {
      "version": "0.18.5",
      "resolved": "https://registry.npmjs.org/xlsx/-/xlsx-0.18.5.tgz",
      "integrity": "sha512-dmg3LCjBPHZnQp5/F/
+nnTa+miPJxUXB6vtk42YjBBKayDNagxGEeIdWApkYPOf3Z3pm3k62Knjzp7lMeTEtFQ==",
      "license": "Apache-2.0",
      "dependencies": {
        "adler-32": "~1.3.0",
        "cfb": "~1.2.1",
        "codepage": "~1.15.0",
        "crc-32": "~1.2.1",
        "ssf": "~0.11.2",
        "wmf": "~1.0.1",
        "word": "~0.3.0"
      },
      "bin": {
        "xlsx": "bin/xlsx.njs"
      },
      "engines": {
        "node": ">=0.8"
      }
    },
    "node_modules/xmlhttprequest-ssl": {
      "version": "2.1.2",
      "resolved": "https://registry.npmjs.org/xmlhttprequest-ssl/-/xmlhttprequest-
ssl-2.1.2.tgz",
      "integrity": "sha512-
TEU+nJVUUnA4CYJFLvK5X9AOeH4KvDvhIfm0vV1GaQRtchnG0hgK5p8hw/
xjv8cunWYCsiPCSDzObPyhEwq3KQ==",
      "engines": {
        "node": ">=0.4.0"
      }
    },
    "node_modules/yallist": {
      "version": "3.1.1",
      "resolved": "https://registry.npmjs.org/yallist/-/yallist-3.1.1.tgz",
      "integrity": "sha512-a4UGQaWPH59mOXUYnAG2ewncQS4i4F43Tv3JoAM+s2VDAmS9NsK8GpD
MLrCHPksFT7h3K6TOoUNn2pb7RoXx4g==",
      "dev": true,
      "license": "ISC"
    },
    "node_modules/yaml": {
      "version": "2.7.1",
      "resolved": "https://registry.npmjs.org/yaml/-/yaml-2.7.1.tgz",
      "integrity": "sha512-10ULxpnOCQXxJvBgxsn9ptjq6uviG/htZKk9veJGhlqn3w/
DxQ631zFF+nlQXLwmImeS5amR2dl2U8sg6U9jsQ==",
      "dev": true,
      "license": "ISC",
      "bin": {
        "yaml": "bin.mjs"
      },
      "engines": {
        "node": ">= 14"
      }
    },
    "node_modules/yocto-queue": {
      "version": "0.1.0",
      "resolved": "https://registry.npmjs.org/yocto-queue/-/yocto-
queue-0.1.0.tgz",
```

```
      "integrity": "sha512-rVksvsnNCdJ/ohGc6xgPwyN8eheCxsiLM8mxuE/t/
mOVqJewPuO1miLpTHQiRgTKCLexL4MeAFVagts7HmNZ2Q==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=10"
      },
      "funding": {
        "url": "https://github.com/sponsors/sindresorhus"
      }
    }
  }
}
```

package.json

```
{
  "name": "client",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@heroicons/react": "^2.0.18",
    "aos": "^2.3.4",
    "axios": "^1.4.0",
    "chart.js": "^4.4.9",
    "date-fns": "^4.1.0",
    "lottie-react": "^2.4.1",
    "lucide-react": "^0.511.0",
    "react": "^18.2.0",
    "react-chartjs-2": "^5.3.0",
    "react-dom": "^18.2.0",
    "react-hot-toast": "^2.4.1",
    "react-icons": "^4.10.1",
    "react-image-crop": "^11.0.10",
    "react-redux": "^9.2.0",
    "react-router-dom": "^6.14.1",
    "socket.io-client": "^4.8.1",
    "xlsx": "^0.18.5"
  },
  "devDependencies": {
    "@eslint/js": "^9.22.0",
    "@types/react": "^19.0.10",
    "@types/react-dom": "^19.0.4",
    "@vitejs/plugin-react": "^4.3.4",
    "autoprefixer": "^10.4.21",
    "eslint": "^9.22.0",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.19",
    "globals": "^16.0.0",
    "postcss": "^8.5.3",
    "tailwindcss": "^3.1.0",
```

```
    "vite": "^6.3.1"
  }
}
```

postcss.config.js

```
import tailwindcss from 'tailwindcss';Ð
import autoprefixer from 'autoprefixer';Ð
Ð
export default {Ð
  plugins: [Ð
    tailwindcss,Ð
    autoprefixer,Ð
  ],Ð
};Ð
```

public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/
fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/css/
bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-SgOJa3DmI69IUzQ2PVdRZhwQ+dy64/
BUtbMJw1MZ8t5HZApcHrRKUc4W0kG879m7"
      crossorigin="anonymous"
    />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
```

```
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->

    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/js/
bootstrap.bundle.min.js"
      integrity="sha384-k6d4wzSIapyDyv1kpU366/
PK5hCdSbCRGRCMv+eplOQJWyd1fbcAu9OCUj5zNLiq"
      crossorigin="anonymous"
    ></script>
  </body>
</html>
```

public/manifest.json

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

README.md

```
# React + Vite

This template provides a minimal setup to get React working in Vite with HMR and
some ESLint rules.

Currently, two official plugins are available:
```

```
- [@vitejs/plugin-react](https://github.com/vitejs/vite-plugin-react/blob/main/
packages/plugin-react) uses [Babel](https://babeljs.io/) for Fast Refresh
- [@vitejs/plugin-react-swc](https://github.com/vitejs/vite-plugin-react/blob/
main/packages/plugin-react-swc) uses [SWC](https://swc.rs/) for Fast Refresh

## Expanding the ESLint configuration

If you are developing a production application, we recommend using TypeScript
with type-aware lint rules enabled. Check out the [TS template](https://
github.com/vitejs/vite/tree/main/packages/create-vite/template-react-ts) for
information on how to integrate TypeScript and [`typescript-eslint`](https://
typescript-eslint.io) in your project.
```

scripts/generateCodePDF.js

```
import PDFDocument from 'pdfkit';
import fs from 'fs';
import path from 'path';
import { promisify } from 'util';
import { fileURLToPath } from 'url';

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

const readdir = promisify(fs.readdir);
const readFile = promisify(fs.readFile);
const stat = promisify(fs.stat);

// Function to get all files recursively
async function getFiles(dir) {
  const files = [];
  const items = await readdir(dir);

  for (const item of items) {
    if (item.startsWith('.') || item === 'node_modules' || item === 'dist' ||
item === 'build') continue;

    const fullPath = path.join(dir, item);
    const stats = await stat(fullPath);

    if (stats.isDirectory()) {
      const subFiles = await getFiles(fullPath);
      files.push(...subFiles);
    } else {
      // Only include code files
      const ext = path.extname(item).toLowerCase();
      if (['.js', '.jsx', '.ts', '.tsx', '.css', '.html', '.json',
'.md'].includes(ext)) {
        files.push(fullPath);
      }
    }
  }

  return files;
}

async function generatePDF() {
```

```
try {
  // Create PDF document
  const doc = new PDFDocument({
    size: 'A4',
    margins: {
      top: 50,
      bottom: 50,
      left: 50,
      right: 50
    }
  });

  // Pipe output to file
  doc.pipe(fs.createWriteStream('client_code_documentation.pdf'));

  // Set font
  doc.font('Helvetica');

  // Get all files
  const rootDir = path.join(__dirname, '..');
  const files = await getFiles(rootDir);

  // Sort files by directory and name
  files.sort((a, b) => a.localeCompare(b));

  // Process each file
  for (const file of files) {
    // Get relative path for display
    const relativePath = path.relative(rootDir, file);

    // Add page break except for first page
    if (doc.page.pageNumber > 1) {
      doc.addPage();
    }

    // Add file path as header
    doc.fontSize(16)
       .fillColor('#2563eb')
       .text(relativePath, { underline: true })
       .moveDown();

    // Read and add file content
    const content = await readFile(file, 'utf8');

    // Add file content with monospace font and smaller size
    doc.font('Courier')
       .fontSize(10)
       .fillColor('#000000')
       .text(content, {
         lineGap: 2,
         align: 'left'
       })
       .moveDown();
  }

  // Finalize PDF
  doc.end();
  console.log('PDF generated successfully: client_code_documentation.pdf');
} catch (error) {
```

```
    console.error('Error generating PDF:', error);
  }
}

// Run the script
generatePDF();
```

src/App.css

```
#root {
  width: 100%;
  height: 100vh;
  margin: 0;
  padding: 0;
  text-align: center;
}

.logo {
  height: 6em;
  padding: 1.5em;
  will-change: filter;
  transition: filter 300ms;
}
.logo:hover {
  filter: drop-shadow(0 0 2em #646cffaa);
}
.logo.react:hover {
  filter: drop-shadow(0 0 2em #61dafbaa);
}

@keyframes logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

@media (prefers-reduced-motion: no-preference) {
  a:nth-of-type(2) .logo {
    animation: logo-spin infinite 20s linear;
  }
}

.card {
  padding: 2em;
}

.read-the-docs {
  color: #888;
}

/* Floating AI Button Styles */
.ai-floating-button {
  position: fixed;
  bottom: 20px;
  right: 20px;
```

```
  z-index: 1000;
}

.ai-modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1001;
}

.ai-modal-content {
  background: white;
  border-radius: 8px;
  max-width: 900px;
  width: 90%;
  max-height: 80vh;
  overflow-y: auto;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
```

src/App.jsx

```
// src/App.jsx
import React, { useEffect } from "react";
import { HashRouter as Router } from "react-router-dom";
import AllRoutes from "./routes/AllRoutes";
import "./App.css";
import { AuthProvider, useAuth } from "./context/AuthContext";
import { ChatProvider } from "./context/ChatContext";
import { Toaster } from 'react-hot-toast';
import ChatWindow from "./components/Chat/ChatWindow";
import ChatDebug from "./components/Chat/ChatDebug";
import notificationService from "./services/notificationService";
import DocumentationPage from './pages/DocumentationPage';

// Component to handle notification service initialization
const NotificationHandler = () => {
  const { user } = useAuth();

  useEffect(() => {
    if (user && user._id) {
      console.log('Ø=Ý  Initializing notifications for user:', user.fullName);

      // Connect to notification service
      notificationService.connect(user._id);

      // Request notification permission
      notificationService.requestNotificationPermission();

      // Cleanup on unmount
      return () => {
        notificationService.disconnect();
```

```
      };
    }
  }, [user]);

  return null; // This component doesn't render anything
};

const App = () => {
  useEffect(() => {
    document.title = 'Traincape Technology CRM';
  }, []);
  return (
    <Router>
      <AuthProvider>
        <ChatProvider>
          <div className="min-h-screen bg-white dark:bg-slate-900 text-slate-900
dark:text-slate-100 transition-all duration-200 ease-out">
            <NotificationHandler />
            <AllRoutes />
            <ChatWindow />
            <ChatDebug />
            <Toaster
              position="top-right"
              toastOptions={{
                duration: 4000,
                style: {
                  background: '#363636',
                  color: '#fff',
                },
                success: {
                  duration: 3000,
                  iconTheme: {
                    primary: '#4ade80',
                    secondary: '#fff',
                  },
                },
                error: {
                  duration: 5000,
                  iconTheme: {
                    primary: '#ef4444',
                    secondary: '#fff',
                  },
                },
              }}
            />
          </div>
        </ChatProvider>
      </AuthProvider>
    </Router>
  );
};

export default App;
```

src/App.test.js

```
import { render, screen } from '@testing-library/react';
import App from './App';
```

```
test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

src/assets/sounds/notification-sounds.js

```
// WhatsApp-style notification sounds using Web Audio API
class NotificationSounds {
  constructor() {
    this.audioContext = null;
    this.initializeAudio();
  }

  // Initialize audio context
  initializeAudio() {
    try {
      this.audioContext = new (window.AudioContext || window.webkitAudioContext)
();
    } catch (error) {
      console.warn('Audio context not supported:', error);
    }
  }

  // Resume audio context if suspended
  async resumeAudioContext() {
    if (this.audioContext && this.audioContext.state === 'suspended') {
      await this.audioContext.resume();
    }
  }

  // Create and play a tone
  playTone(frequency, duration, volume = 1.3, type = 'sine') {
    if (!this.audioContext) return;

    const oscillator = this.audioContext.createOscillator();
    const gainNode = this.audioContext.createGain();

    oscillator.connect(gainNode);
    gainNode.connect(this.audioContext.destination);

    oscillator.frequency.value = frequency;
    oscillator.type = type;

    gainNode.gain.setValueAtTime(volume, this.audioContext.currentTime);
    gainNode.gain.exponentialRampToValueAtTime(0.01,
this.audioContext.currentTime + duration / 1000);

    oscillator.start(this.audioContext.currentTime);
    oscillator.stop(this.audioContext.currentTime + duration / 1000);
  }

  // WhatsApp-style message sound (ascending tone)
  async playMessageSound() {
    try {
      await this.resumeAudioContext();
```

```
    // Two-tone notification similar to WhatsApp
    this.playTone(800, 150, 0.3, 'sine');
    setTimeout(() => {
      this.playTone(1000, 150, 0.3, 'sine');
    }, 100);
  } catch (error) {
    console.error('Error playing message sound:', error);
  }
}

// Different sound for group messages (triple tone)
async playGroupMessageSound() {
  try {
    await this.resumeAudioContext();

    // Three-tone notification for group messages
    this.playTone(650, 120, 0.25, 'sine');
    setTimeout(() => {
      this.playTone(800, 120, 0.25, 'sine');
    }, 150);
    setTimeout(() => {
      this.playTone(1000, 120, 0.25, 'sine');
    }, 300);
  } catch (error) {
    console.error('Error playing group message sound:', error);
  }
}

// Urgent notification sound (rapid beeps)
async playUrgentSound() {
  try {
    await this.resumeAudioContext();

    // Rapid beeps for urgent notifications
    for (let i = 0; i < 3; i++) {
      setTimeout(() => {
        this.playTone(1200, 100, 0.4, 'square');
      }, i * 200);
    }
  } catch (error) {
    console.error('Error playing urgent sound:', error);
  }
}

// Soft notification sound (gentle tone)
async playSoftSound() {
  try {
    await this.resumeAudioContext();

    // Gentle single tone
    this.playTone(600, 300, 0.2, 'sine');
  } catch (error) {
    console.error('Error playing soft sound:', error);
  }
}

// Success sound (ascending chord)
async playSuccessSound() {
```

```
    try {
      await this.resumeAudioContext();

      // Ascending chord
      this.playTone(523, 200, 0.15, 'sine'); // C
      setTimeout(() => {
        this.playTone(659, 200, 0.15, 'sine'); // E
      }, 100);
      setTimeout(() => {
        this.playTone(784, 200, 0.15, 'sine'); // G
      }, 200);
    } catch (error) {
      console.error('Error playing success sound:', error);
    }
  }

  // Error sound (descending tone)
  async playErrorSound() {
    try {
      await this.resumeAudioContext();

      // Descending tone
      this.playTone(400, 300, 0.3, 'sawtooth');
      setTimeout(() => {
        this.playTone(300, 300, 0.3, 'sawtooth');
      }, 200);
    } catch (error) {
      console.error('Error playing error sound:', error);
    }
  }

  // Test if audio is supported
  isAudioSupported() {
    return !!this.audioContext;
  }

  // Get audio context state
  getAudioState() {
    return this.audioContext ? this.audioContext.state : 'not-supported';
  }
}

// Export singleton instance
export default new NotificationSounds();
```

src/components/ActivityTimer/ActivityTimer.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../../context/AuthContext';
import useActivityTracker from '../../hooks/useActivityTracker';

const ActivityTimer = () => {
  const { user } = useAuth();
  const [sessionTime, setSessionTime] = useState(0);
  const [showControls, setShowControls] = useState(false);

  // Get activity tracker functions and state
  const {
```

```
    isSessionActive,
    sessionStartTime,
    isManuallyPaused,
    startSession,
    pauseSession
  } = useActivityTracker();

  useEffect(() => {
    if (!user || user.role === 'Customer') return;

    // Update timer every second
    const interval = setInterval(() => {
      if (sessionStartTime && !isManuallyPaused) {
        const currentTime = Date.now();
        const elapsed = Math.floor((currentTime - sessionStartTime) / 1000);
        setSessionTime(elapsed);
      } else {
        setSessionTime(0);
      }
    }, 1000);

    // Cleanup on unmount
    return () => {
      clearInterval(interval);
    };
  }, [user, sessionStartTime, isManuallyPaused]);

  // Format time as HH:MM:SS
  const formatTime = (seconds) => {
    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);
    const secs = seconds % 60;

    if (hours > 0) {
      return `${hours.toString().padStart(2, '0')}:
${minutes.toString().padStart(2, '0')}:${secs.toString().padStart(2, '0')}`;
    } else {
      return `${minutes.toString().padStart(2, '0')}:
${secs.toString().padStart(2, '0')}`;
    }
  };

  // Handle start/pause button click
  const handleToggle = () => {
    if (isManuallyPaused) {
      startSession();
    } else {
      pauseSession();
    }
  };

  // Don't show for customers or if user is not logged in
  if (!user || user.role === 'Customer') {
    return null;
  }

  return (
    <div
      className="relative"
```

```
      onMouseEnter={() => setShowControls(true)}
      onMouseLeave={() => setShowControls(false)}
    >
      {/* Main Timer Display */}
      <div className="flex items-center space-x-2 bg-gray-100 dark:bg-gray-800
px-3 py-2 rounded-lg cursor-pointer">
        <div className={`w-2 h-2 rounded-full ${
          !isManuallyPaused && sessionStartTime ? 'bg-green-500 animate-pulse' :
'bg-gray-400'
        }`}></div>
        <div className="text-sm font-medium text-gray-700 dark:text-gray-500">
          <span className="hidden sm:inline">Session: </span>
          <span className="font-mono">{formatTime(sessionTime)}</span>
        </div>

        {/* Status indicator */}
        <div className="text-xs text-gray-500 dark:text-gray-500">
          {isManuallyPaused ? '#øþ ' : sessionStartTime ? '%¶þ ' : '#ùþ '}
        </div>
      </div>

      {/* Control Panel (shows on hover) */}
      {showControls && (
        <div className="absolute top-full left-0 mt-1 bg-white dark:bg-gray-700
border border-gray-200 dark:border-gray-600 rounded-lg shadow-lg dark:shadow-
black/25 p-3 z-50 min-w-[200px]">
          <div className="text-xs font-semibold text-gray-600 dark:text-gray-500
mb-2">
            Activity Timer Controls
          </div>

          {/* Current Status */}
          <div className="text-xs text-gray-500 dark:text-gray-500 mb-3">
            Status: {
              isManuallyPaused ? 'Manually Paused' :
              sessionStartTime ? 'Active' : 'Stopped'
            }
          </div>

          {/* Control Buttons */}
          <div className="flex space-x-2">
            <button
              onClick={handleToggle}
              className={`px-3 py-1 text-xs rounded-md font-medium transition-
colors ${
                isManuallyPaused
                  ? 'bg-green-600 hover:bg-green-700 text-white'
                  : 'bg-orange-600 hover:bg-orange-700 text-white'
              }`}
            >
              {isManuallyPaused ? '%¶þ  Start' : '#øþ  Pause'}
            </button>
          </div>

          {/* Help Text */}
          <div className="text-xs text-gray-400 dark:text-gray-400 mt-2 leading-
relaxed">
            Timer automatically pauses when system locks or tab is hidden.
          </div>
```

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

export default ActivityTimer;
```

src/components/Admin/LeaveApproval.jsx

```
import React, { useState, useEffect } from 'react';
import { FaCheck, FaTimes, FaEye, FaFilter, FaSearch, FaClock, FaCalendarCheck,
FaExclamationTriangle, FaCheckCircle } from 'react-icons/fa';
import leaveAPI from '../../services/leaveAPI';

const LeaveApproval = () => {
  const [leaves, setLeaves] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState('');
  const [success, setSuccess] = useState('');
  const [selectedLeave, setSelectedLeave] = useState(null);
  const [showApprovalModal, setShowApprovalModal] = useState(false);
  const [rejectionReason, setRejectionReason] = useState('');
  const [actionType, setActionType] = useState(''); // 'approve' or 'reject'
  const [processing, setProcessing] = useState(false);

  // Filters
  const [filters, setFilters] = useState({
    status: '',
    leaveType: '',
    startDate: '',
    endDate: '',
    search: ''
  });

  const [pagination, setPagination] = useState({
    page: 1,
    limit: 10,
    total: 0,
    pages: 0
  });

  const leaveTypes = [
    { value: 'casual', label: 'Casual Leave', icon: 'Ø<ßÖþ ' },
    { value: 'sick', label: 'Sick Leave', icon: 'Ø>Ý ' },
    { value: 'annual', label: 'Annual Leave', icon: '' þ ' },
    { value: 'emergency', label: 'Emergency Leave', icon: 'Ø=Þ¨' },
    { value: 'personal', label: 'Personal Leave', icon: 'Ø=Üd' },
    { value: 'maternity', label: 'Maternity Leave', icon: 'Ø>Ý1' },
    { value: 'paternity', label: 'Paternity Leave', icon: 'Ø=Üh  Ø=Üv' },
    { value: 'bereavement', label: 'Bereavement Leave', icon: 'Ø=ÞO' }
  ];

  const statusColors = {
    pending: 'bg-yellow-100 text-yellow-800 border-yellow-200',
    approved: 'bg-green-100 text-green-800 border-green-200',
    rejected: 'bg-red-100 text-red-800 border-red-200',
    cancelled: 'bg-gray-100 text-gray-800 border-gray-200'
```

```
};

const statusIcons = {
  pending: <FaClock className="w-4 h-4" />,
  approved: <FaCheckCircle className="w-4 h-4" />,
  rejected: <FaTimes className="w-4 h-4" />,
  cancelled: <FaTimes className="w-4 h-4" />
};

useEffect(() => {
  fetchLeaves();
}, [filters, pagination.page]);

const fetchLeaves = async () => {
  try {
    setLoading(true);
    const params = {
      ...filters,
      page: pagination.page,
      limit: pagination.limit
    };

    // Remove empty filter values
    Object.keys(params).forEach(key => {
      if (params[key] === '') delete params[key];
    });

    const response = await leaveAPI.getAllLeaves(params);
    setLeaves(response.data.data || []);
    setPagination(prev => ({
      ...prev,
      ...response.data.pagination
    }));
  } catch (error) {
    setError('Failed to fetch leave applications');
    console.error('Error fetching leaves:', error);
  } finally {
    setLoading(false);
  }
};

const handleFilterChange = (e) => {
  const { name, value } = e.target;
  setFilters(prev => ({
    ...prev,
    [name]: value
  }));
  setPagination(prev => ({ ...prev, page: 1 })); // Reset to first page
};

const clearFilters = () => {
  setFilters({
    status: '',
    leaveType: '',
    startDate: '',
    endDate: '',
    search: ''
  });
  setPagination(prev => ({ ...prev, page: 1 }));
```

```
  };

  const handleApprovalAction = (leave, action) => {
    setSelectedLeave(leave);
    setActionType(action);
    setRejectionReason('');
    setShowApprovalModal(true);
  };

  const confirmApprovalAction = async () => {
    if (actionType === 'reject' && !rejectionReason.trim()) {
      setError('Please provide a reason for rejection');
      return;
    }

    try {
      setProcessing(true);
      const data = {
        status: actionType === 'approve' ? 'approved' : 'rejected'
      };

      if (actionType === 'reject') {
        data.rejectionReason = rejectionReason;
      }

      await leaveAPI.updateLeaveStatus(selectedLeave._id, data);

      setSuccess(`Leave ${actionType === 'approve' ? 'approved' : 'rejected'}
successfully`);
      setShowApprovalModal(false);
      fetchLeaves();

      setTimeout(() => setSuccess(''), 3000);
    } catch (error) {
      setError(error.response?.data?.message || `Failed to ${actionType} leave`);
    } finally {
      setProcessing(false);
    }
  };

  const formatDate = (dateString) => {
    return new Date(dateString).toLocaleDateString('en-US', {
      year: 'numeric',
      month: 'short',
      day: 'numeric'
    });
  };

  const getLeaveTypeInfo = (type) => {
    return leaveTypes.find(t => t.value === type) || { label: type, icon: 'Ø=ÜÄ' };
  };

  const handlePageChange = (newPage) => {
    setPagination(prev => ({ ...prev, page: newPage }));
  };

  return (
    <div className="max-w-7xl mx-auto p-6">
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md border
```

```
border-slate-200 dark:border-slate-700">
        <div className="p-6 border-b border-gray-200 dark:border-gray-700">
          <div className="flex items-center justify-between mb-6">
            <h2 className="text-2xl font-bold text-gray-900 dark:text-white flex
items-center">
              <FaCalendarCheck className="mr-3 text-blue-600" />
              Leave Approval
            </h2>
            <div className="text-sm text-gray-500 dark:text-gray-400">
              Total: {pagination.total} applications
            </div>
          </div>

          {/* Success/Error Messages */}
          {error && (
            <div className="mb-4 p-3 bg-red-50 border border-red-200 text-red-600
rounded-md flex items-center">
              <FaExclamationTriangle className="mr-2" />
              {error}
              <button onClick={() => setError('')} className="ml-auto">
                <FaTimes className="w-4 h-4" />
              </button>
            </div>
          )}

          {success && (
            <div className="mb-4 p-3 bg-green-50 border border-green-200 text-
green-600 rounded-md flex items-center">
              <FaCheckCircle className="mr-2" />
              {success}
              <button onClick={() => setSuccess('')} className="ml-auto">
                <FaTimes className="w-4 h-4" />
              </button>
            </div>
          )}

          {/* Filters */}
          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 xl:grid-
cols-6 gap-4 mb-6">
            <div>
              <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
                Status
              </label>
              <select
                name="status"
                value={filters.status}
                onChange={handleFilterChange}
                className="w-full border border-gray-300 dark:border-gray-600
rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
text-sm"
              >
                <option value="">All Status</option>
                <option value="pending">Pending</option>
                <option value="approved">Approved</option>
                <option value="rejected">Rejected</option>
                <option value="cancelled">Cancelled</option>
              </select>
            </div>
```

```
            <div>
              <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
                Leave Type
              </label>
              <select
                name="leaveType"
                value={filters.leaveType}
                onChange={handleFilterChange}
                className="w-full border border-gray-300 dark:border-gray-600
rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
text-sm"
              >
                <option value="">All Types</option>
                {leaveTypes.map(type => (
                  <option key={type.value} value={type.value}>
                    {type.label}
                  </option>
                ))}
              </select>
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
                From Date
              </label>
              <input
                type="date"
                name="startDate"
                value={filters.startDate}
                onChange={handleFilterChange}
                className="w-full border border-gray-300 dark:border-gray-600
rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
text-sm"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
                To Date
              </label>
              <input
                type="date"
                name="endDate"
                value={filters.endDate}
                onChange={handleFilterChange}
                className="w-full border border-gray-300 dark:border-gray-600
rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
text-sm"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
                Search
```

```
              </label>
              <div className="relative">
                <FaSearch className="absolute left-3 top-1/2 transform -translate-
y-1/2 text-gray-400 w-4 h-4" />
                <input
                  type="text"
                  name="search"
                  value={filters.search}
                  onChange={handleFilterChange}
                  placeholder="Employee name..."
                  className="w-full border border-gray-300 dark:border-gray-600
rounded-md pl-10 pr-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100 text-sm"
                />
              </div>
            </div>

            <div className="flex items-end">
              <button
                onClick={clearFilters}
                className="w-full bg-gray-100 hover:bg-gray-200 dark:bg-gray-700
dark:hover:bg-gray-600 text-gray-700 dark:text-gray-300 px-4 py-2 rounded-md text-
sm font-medium transition duration-200 flex items-center justify-center"
              >
                <FaFilter className="mr-2" />
                Clear
              </button>
            </div>
          </div>
        </div>

        {/* Leave Applications List */}
        <div className="p-6">
          {loading ? (
            <div className="flex justify-center py-8">
              <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-
blue-600"></div>
            </div>
          ) : leaves.length === 0 ? (
            <div className="text-center py-8 text-gray-500 dark:text-gray-400">
              <FaCalendarCheck className="mx-auto h-12 w-12 mb-4 opacity-50" />
              <p>No leave applications found</p>
            </div>
          ) : (
            <div className="space-y-4">
              {leaves.map((leave) => {
                const leaveTypeInfo = getLeaveTypeInfo(leave.leaveType);
                return (
                  <div key={leave._id} className="border border-gray-200
dark:border-gray-700 rounded-lg p-4 hover:shadow-md transition-shadow">
                    <div className="flex items-start justify-between">
                      <div className="flex-1">
                        <div className="flex items-center space-x-3 mb-3">
                          <div className="flex items-center space-x-2">
                            <span className="text-lg">{leaveTypeInfo.icon}</span>
                            <h4 className="font-medium text-gray-900 dark:text-
white">
                              {leaveTypeInfo.label}
                            </h4>
```

```
                        </div>
                        <span className={`inline-flex items-center px-2.5
py-0.5 rounded-full text-xs font-medium border ${statusColors[leave.status]}`}>
                          {statusIcons[leave.status]}
                          <span className="ml-1 capitalize">{leave.status}</
span>
                        </span>
                      </div>

                      <div className="grid grid-cols-1 md:grid-cols-4 gap-4
text-sm text-gray-600 dark:text-gray-400 mb-3">
                        <div>
                          <span className="font-medium">Employee:</span>
                          <div className="text-gray-900 dark:text-gray-100">
                            {leave.employeeId?.fullName ||
leave.userId?.fullName}
                          </div>
                          <div className="text-xs">
                            {leave.employeeId?.email || leave.userId?.email}
                          </div>
                        </div>
                        <div>
                          <span className="font-medium">Duration:</span>
                          <div className="text-gray-900 dark:text-gray-100">
                            {formatDate(leave.startDate)}
                            {leave.startDate !== leave.endDate && ` -
${formatDate(leave.endDate)}`}
                            {leave.isHalfDay && (
                              <span className="text-xs bg-blue-100 text-
blue-800 px-1 rounded ml-1">
                                {leave.halfDaySession} half
                              </span>
                            )}
                          </div>
                        </div>
                        <div>
                          <span className="font-medium">Days:</span>
                          <span className="text-gray-900 dark:text-gray-100
ml-1">{leave.totalDays}</span>
                        </div>
                        <div>
                          <span className="font-medium">Applied:</span>
                          <div className="text-gray-900 dark:text-
gray-100">{formatDate(leave.appliedDate)}</div>
                        </div>
                      </div>

                      <div className="mb-3">
                        <span className="font-medium text-sm text-gray-600
dark:text-gray-400">Reason:</span>
                        <p className="text-sm text-gray-800 dark:text-gray-200
mt-1">{leave.reason}</p>
                      </div>

                      {leave.rejectionReason && (
                        <div className="mb-3 p-2 bg-red-50 dark:bg-red-900/20
border border-red-200 dark:border-red-800 rounded">
                          <span className="font-medium text-sm text-red-800
dark:text-red-200">Rejection Reason:</span>
```

```
                            <p className="text-sm text-red-700 dark:text-red-300
mt-1">{leave.rejectionReason}</p>
                          </div>
                        )}

                        {leave.approvedBy && leave.status === 'approved' && (
                          <div className="text-sm text-green-600 dark:text-
green-400">
                            <span className="font-medium">Approved by:</span>
{leave.approvedBy.fullName} on {formatDate(leave.approvedDate)}
                          </div>
                        )}
                      </div>

                      {leave.status === 'pending' && (
                        <div className="ml-4 flex space-x-2">
                          <button
                            onClick={() => handleApprovalAction(leave, 'approve')}
                            className="bg-green-600 hover:bg-green-700 text-white
px-3 py-1 rounded-md text-sm font-medium transition duration-200 flex items-
center"
                          >
                            <FaCheck className="mr-1" />
                            Approve
                          </button>
                          <button
                            onClick={() => handleApprovalAction(leave, 'reject')}
                            className="bg-red-600 hover:bg-red-700 text-white
px-3 py-1 rounded-md text-sm font-medium transition duration-200 flex items-
center"
                          >
                            <FaTimes className="mr-1" />
                            Reject
                          </button>
                        </div>
                      )}
                    </div>
                  </div>
                );
              })}
            </div>
          )}

          {/* Pagination */}
          {pagination.pages > 1 && (
            <div className="mt-6 flex items-center justify-between">
              <div className="text-sm text-gray-500 dark:text-gray-400">
                Showing {((pagination.page - 1) * pagination.limit) + 1} to
{Math.min(pagination.page * pagination.limit, pagination.total)} of
{pagination.total} results
              </div>
              <div className="flex space-x-2">
                <button
                  onClick={() => handlePageChange(pagination.page - 1)}
                  disabled={pagination.page === 1}
                  className="px-3 py-1 border border-gray-300 dark:border-
gray-600 rounded-md text-sm disabled:opacity-50 disabled:cursor-not-allowed
hover:bg-gray-50 dark:hover:bg-gray-700"
                >
```

```
                Previous
              </button>
              <span className="px-3 py-1 text-sm text-gray-600 dark:text-
gray-400">
                Page {pagination.page} of {pagination.pages}
              </span>
              <button
                onClick={() => handlePageChange(pagination.page + 1)}
                disabled={pagination.page === pagination.pages}
                className="px-3 py-1 border border-gray-300 dark:border-
gray-600 rounded-md text-sm disabled:opacity-50 disabled:cursor-not-allowed
hover:bg-gray-50 dark:hover:bg-gray-700"
              >
                Next
              </button>
            </div>
          </div>
        )}
      </div>
    </div>

    {/* Approval/Rejection Modal */}
    {showApprovalModal && (
      <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
        <div className="bg-white dark:bg-slate-900 rounded-lg shadow-lg max-w-
md w-full mx-4">
          <div className="p-6">
            <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-4">
              {actionType === 'approve' ? 'Approve Leave' : 'Reject Leave'}
            </h3>

            <div className="mb-4">
              <p className="text-sm text-gray-600 dark:text-gray-400">
                Employee: <span className="font-medium text-gray-900 dark:text-
white">
                  {selectedLeave?.employeeId?.fullName ||
selectedLeave?.userId?.fullName}
                </span>
              </p>
              <p className="text-sm text-gray-600 dark:text-gray-400">
                Leave Type: <span className="font-medium text-gray-900
dark:text-white">
                  {getLeaveTypeInfo(selectedLeave?.leaveType).label}
                </span>
              </p>
              <p className="text-sm text-gray-600 dark:text-gray-400">
                Duration: <span className="font-medium text-gray-900 dark:text-
white">
                  {selectedLeave && formatDate(selectedLeave.startDate)} -
{selectedLeave && formatDate(selectedLeave.endDate)} ({selectedLeave?.totalDays}
days)
                </span>
              </p>
            </div>

            {actionType === 'reject' && (
              <div className="mb-4">
```

```
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                    Reason for Rejection *
                  </label>
                  <textarea
                    value={rejectionReason}
                    onChange={(e) => setRejectionReason(e.target.value)}
                    rows={3}
                    placeholder="Please provide a reason for rejecting this
leave..."
                    className="w-full border border-gray-300 dark:border-gray-600
rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
                    required
                  />
                </div>
              )}

              <div className="flex justify-end space-x-3">
                <button
                  onClick={() => setShowApprovalModal(false)}
                  className="px-4 py-2 border border-gray-300 dark:border-
gray-600 text-gray-700 dark:text-gray-300 rounded-md hover:bg-gray-50
dark:hover:bg-gray-700 transition duration-200"
                >
                  Cancel
                </button>
                <button
                  onClick={confirmApprovalAction}
                  disabled={processing || (actionType === 'reject' && !
rejectionReason.trim())}
                  className={`px-4 py-2 rounded-md font-medium transition
duration-200 flex items-center ${
                    actionType === 'approve'
                      ? 'bg-green-600 hover:bg-green-700 text-white'
                      : 'bg-red-600 hover:bg-red-700 text-white'
                  } disabled:opacity-50 disabled:cursor-not-allowed`}
                >
                  {processing ? (
                    <>
                      <div className="animate-spin rounded-full h-4 w-4 border-
b-2 border-white mr-2"></div>
                      Processing...
                    </>
                  ) : (
                    <>
                      {actionType === 'approve' ? <FaCheck className="mr-2" /> :
<FaTimes className="mr-2" />}
                      {actionType === 'approve' ? 'Approve' : 'Reject'}
                    </>
                  )}
                </button>
              </div>
            </div>
          </div>
        </div>
      )}
    </div>
  );
};
```

```
export default LeaveApproval;
```

src/components/AttendanceWidget.jsx

```
import React, { useState, useEffect } from 'react';
import api from '../services/api';
import { toast } from 'react-toastify';
import { FaSpinner, FaMapMarkerAlt } from 'react-icons/fa';

const AttendanceWidget = () => {
  const [todayAttendance, setTodayAttendance] = useState(null);
  const [loading, setLoading] = useState(true);
  const [checkingIn, setCheckingIn] = useState(false);
  const [checkingOut, setCheckingOut] = useState(false);
  const [notes, setNotes] = useState('');
  const [location, setLocation] = useState(null);
  const [gettingLocation, setGettingLocation] = useState(false);
  const [locationError, setLocationError] = useState('');

  // Office location coordinates
  const OFFICE_LOCATION = {
    latitude: 28.607217,   // Replace with your actual office latitude
    longitude: 77.081655,  // Replace with your actual office longitude
    allowedRadius: 20    // 20 meters radius
  };

  // Add auto-refresh effect
  useEffect(() => {
    // Initial fetch
    fetchTodayAttendance();

    // Set up interval to refresh data every minute
    const interval = setInterval(fetchTodayAttendance, 60000); // 60000 ms = 1
minute

    return () => clearInterval(interval);
  }, []);

  const fetchTodayAttendance = async () => {
    try {
      const response = await api.get('/attendance/today');
      if (response.data.success) {
        // Format the attendance data to match the expected structure
        const attendanceData = response.data.data ? {
          data: {
            checkIn: response.data.data.checkIn ? new
Date(response.data.data.checkIn) : null,
            checkOut: response.data.data.checkOut ? new
Date(response.data.data.checkOut) : null,
            status: response.data.data.status || 'PENDING',
            notes: response.data.data.notes || ''
          },
          hasCheckedIn: response.data.hasCheckedIn,
          hasCheckedOut: response.data.hasCheckedOut
        } : null;

        setTodayAttendance(attendanceData);
```

```
      // If we have attendance data, also get the location to show status
      if (attendanceData?.data?.checkIn && !attendanceData?.data?.checkOut) {
        try {
          await getCurrentLocation();
        } catch (error) {
          // Ignore location errors during auto-refresh
          console.warn('Could not get location during refresh:', error);
        }
      }
    } else {
      toast.error('Failed to fetch attendance status');
    }
    setLoading(false);
  } catch (error) {
    console.error('Error fetching today attendance:', error);
    toast.error('Failed to fetch attendance status');
    setLoading(false);
  }
};

// Calculate distance between two coordinates using Haversine formula
const calculateDistance = (lat1, lon1, lat2, lon2) => {
  const R = 6371e3; // Earth's radius in meters
  const <c  Ò Æ C  ¢ Ö F,å ’ ò  ƒ °
  const <c" Ò Æ C" ¢ Ö F,å ’ ò  ƒ °
  const 9CÆ = (lat2 - lat1) * Math.PI / 180;
  const 9C» = (lon2 - lon1) * Math.PI / 180;

  const a = Math.sin(9CÆ/2) * Math.sin(9CÆ/2) +
            Math.cos(<c ’ ¢ Ö F,æ6÷2ƒÆ2) *
            Math.sin(9C»/2) * Math.sin(9C»/2);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

  return R * c; // Distance in meters
};

const getCurrentLocation = () => {
  return new Promise((resolve, reject) => {
    if (!navigator.geolocation) {
      reject(new Error('Geolocation is not supported by this browser.'));
      return;
    }

    setGettingLocation(true);
    setLocationError('');

    navigator.geolocation.getCurrentPosition(
      (position) => {
        const userLocation = {
          latitude: position.coords.latitude,
          longitude: position.coords.longitude,
          accuracy: position.coords.accuracy
        };

        // Check if user is within office range
        const distance = calculateDistance(
          OFFICE_LOCATION.latitude,
          OFFICE_LOCATION.longitude,
```

```
            userLocation.latitude,
            userLocation.longitude
          );

          const isInRange = distance <= OFFICE_LOCATION.allowedRadius;

          setLocation({ ...userLocation, distance, isInRange });
          setGettingLocation(false);

          if (!isInRange) {
            setLocationError(`You are ${distance.toFixed(1)} meters away from
office. Please move within ${OFFICE_LOCATION.allowedRadius} meters to mark
attendance.`);
            reject(new Error('Not in office range'));
          } else {
            resolve(userLocation);
          }
        },
        (error) => {
          setGettingLocation(false);
          let errorMessage = 'Unable to get your location.';

          switch (error.code) {
            case error.PERMISSION_DENIED:
              errorMessage = 'Location access denied. Please enable location
services.';
              break;
            case error.POSITION_UNAVAILABLE:
              errorMessage = 'Location information is unavailable.';
              break;
            case error.TIMEOUT:
              errorMessage = 'Location request timed out.';
              break;
            default:
              errorMessage = 'An unknown error occurred while getting location.';
              break;
          }

          setLocationError(errorMessage);
          reject(new Error(errorMessage));
        },
        { enableHighAccuracy: true, timeout: 10000, maximumAge: 0 }
      );
    });
  };

  const handleCheckIn = async () => {
    try {
      setCheckingIn(true);

      // Get location first
      const locationData = await getCurrentLocation();

      const response = await api.post('/attendance/checkin', {
        notes,
        location: locationData
      });

      if (response.data.success) {
```

```
      // Format the attendance data
      const attendanceData = {
        data: {
          checkIn: new Date(response.data.data.checkIn),
          checkOut: null,
          status: response.data.data.status || 'PRESENT',
          notes: response.data.data.notes || ''
        },
        hasCheckedIn: true,
        hasCheckedOut: false
      };

      setTodayAttendance(attendanceData);
      setNotes('');
      toast.success('Check-in successful!');
    } else {
      toast.error(response.data.message || 'Failed to check in');
    }
  } catch (error) {
    console.error('Error checking in:', error);
    if (error.message === 'Not in office range') {
      toast.error('You must be within office premises to check in');
    } else {
      toast.error(error.response?.data?.message || 'Failed to check in');
    }
  } finally {
    setCheckingIn(false);
  }
};

const handleCheckOut = async () => {
  try {
    setCheckingOut(true);

    // Get location first
    const locationData = await getCurrentLocation();

    const response = await api.put('/attendance/checkout', {
      notes,
      location: locationData
    });

    if (response.data.success) {
      // Format the attendance data
      const attendanceData = {
        data: {
          checkIn: new Date(response.data.data.checkIn),
          checkOut: new Date(response.data.data.checkOut),
          status: response.data.data.status || 'PRESENT',
          notes: response.data.data.notes || ''
        },
        hasCheckedIn: true,
        hasCheckedOut: true
      };

      setTodayAttendance(attendanceData);
      setNotes('');
      toast.success('Check-out successful!');
    } else {
```

```
        toast.error(response.data.message || 'Failed to check out');
      }
    } catch (error) {
      console.error('Error checking out:', error);
      if (error.message === 'Not in office range') {
        toast.error('You must be within office premises to check out');
      } else {
        toast.error(error.response?.data?.message || 'Failed to check out');
      }
    } finally {
      setCheckingOut(false);
    }
  };

  const formatTime = (dateString) => {
    return new Date(dateString).toLocaleTimeString('en-US', {
      hour: '2-digit',
      minute: '2-digit',
      hour12: true
    });
  };

  if (loading) {
    return (
      <div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">
        <div className="animate-pulse">
          <div className="h-4 bg-gray-200 dark:bg-gray-700 rounded w-3/4 mb-4"></
div>
          <div className="h-8 bg-gray-200 dark:bg-gray-700 rounded w-1/2"></div>
        </div>
      </div>
    );
  }

  return (
    <div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">
      <div className="flex items-center justify-between mb-6">
        <h3 className="text-lg font-semibold text-gray-900 dark:text-white">
          Today's Attendance
        </h3>
        <div className="text-sm text-gray-500 dark:text-gray-400">
          {new Date().toLocaleDateString()}
        </div>
      </div>

      {todayAttendance?.data ? (
        <div>
          <div className="grid grid-cols-2 gap-4">
            <div className="bg-green-50 dark:bg-green-900/20 p-4 rounded-lg">
              <div className="text-green-800 dark:text-green-300 text-sm font-
medium">
                Check-in Time
              </div>
              <div className="text-green-900 dark:text-green-100 text-lg font-
bold">
                {formatTime(todayAttendance.data.checkIn)}
              </div>
            </div>
```

```
            {todayAttendance.data.checkOut ? (
              <div className="bg-red-50 dark:bg-red-900/20 p-4 rounded-lg">
                <div className="text-red-800 dark:text-red-300 text-sm font-
medium">
                  Check-out Time
                </div>
                <div className="text-red-900 dark:text-red-100 text-lg font-bold">
                  {formatTime(todayAttendance.data.checkOut)}
                </div>
              </div>
            ) : (
              <div className="bg-gray-50 dark:bg-gray-700 p-4 rounded-lg">
                <div className="text-gray-600 dark:text-gray-400 text-sm font-
medium">
                  Check-out Time
                </div>
                <div className="text-gray-500 dark:text-gray-400 text-lg">
                  Not checked out
                </div>
              </div>
            )}
          </div>

          {/* Location Status */}
          {location && (
            <div className={`mt-4 p-3 rounded-lg ${
              location.isInRange
                ? 'bg-green-50 dark:bg-green-900/20 border border-green-200
dark:border-green-800'
                : 'bg-red-50 dark:bg-red-900/20 border border-red-200 dark:border-
red-800'
            }`}>
              <div className={`flex items-center text-sm ${
                location.isInRange
                  ? 'text-green-600 dark:text-green-400'
                  : 'text-red-600 dark:text-red-400'
              }`}>
                <FaMapMarkerAlt className="mr-2" />
                <div>
                  <div className="font-medium">
                    {location.isInRange ? ''  Within Office Range' : ''L Outside
Office Range'}
                  </div>
                  <div className="text-xs mt-1">
                    Distance: {location.distance ?
`${location.distance.toFixed(1)}m` : 'Calculating...'}
                    (Required: "d {OFFICE_LOCATION.allowedRadius}m)
                  </div>
                </div>
              </div>
            </div>
          )}

          {!todayAttendance.data.checkOut && (
            <div className="mt-4 space-y-3">
              <textarea
                value={notes}
                onChange={(e) => setNotes(e.target.value)}
                placeholder="Add notes for check-out (optional)"
```

```
                classNameName="w-full p-2 border border-gray-300 dark:border-gray-600
rounded-md bg-white dark:bg-slate-700 text-gray-900 dark:text-white"
                rows="2"
              />
              <button
                onClick={handleCheckOut}
                disabled={checkingOut || gettingLocation}
                classNameName="w-full bg-red-600 hover:bg-red-700 disabled:bg-red-400
text-white py-2 px-4 rounded-md transition-colors flex items-center justify-
center"
              >
                {gettingLocation ? (
                  <>
                    <FaSpinner classNameName="animate-spin mr-2" />
                    Getting Location...
                  </>
                ) : checkingOut ? (
                  <>
                    <FaSpinner classNameName="animate-spin mr-2" />
                    Checking Out...
                  </>
                ) : (
                  'Check Out'
                )}
              </button>
            </div>
          )}
        </div>
      ) : (
        <div classNameName="text-center space-y-4">
          <div classNameName="text-gray-500 dark:text-gray-400">
            You haven't checked in today
          </div>

          <div classNameName="space-y-3">
            <textarea
              value={notes}
              onChange={(e) => setNotes(e.target.value)}
              placeholder="Add notes for check-in (optional)"
              classNameName="w-full p-2 border border-gray-300 dark:border-gray-600
rounded-md bg-white dark:bg-slate-700 text-gray-900 dark:text-white"
              rows="2"
            />
            <button
              onClick={handleCheckIn}
              disabled={checkingIn || gettingLocation}
              classNameName="w-full bg-green-600 hover:bg-green-700 disabled:bg-
green-400 text-white py-2 px-4 rounded-md transition-colors flex items-center
justify-center"
            >
              {gettingLocation ? (
                <>
                  <FaSpinner classNameName="animate-spin mr-2" />
                  Getting Location...
                </>
              ) : checkingIn ? (
                <>
                  <FaSpinner classNameName="animate-spin mr-2" />
                  Checking In...
```

```
                </>
              ) : (
                'Check In'
              )}
            </button>
          </div>
        </div>
      )}

      {/* Location Error */}
      {locationError && (
        <div className="mt-4 p-3 bg-red-50 dark:bg-red-900/20 border border-
red-200 dark:border-red-800 rounded-lg">
          <div className="text-sm text-red-600 dark:text-red-400">
            {locationError}
          </div>
        </div>
      )}
    </div>
  );
};

export default AttendanceWidget;
```

src/components/Auth/CustomerSignUp.jsx

```
import React, { useState } from "react";
import { useNavigate, Link } from "react-router-dom";
import Layout from "../Layout/Layout";
import { useAuth } from "../../context/AuthContext";

const CustomerSignUp = () => {
  const navigate = useNavigate();
  const { register, error } = useAuth();
  const [formData, setFormData] = useState({
    fullName: "",
    email: "",
    password: "",
    confirmPassword: "",
    role: "Customer" // Fixed role for customers
  });
  const [formError, setFormError] = useState("");
  const [isLoading, setIsLoading] = useState(false);

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

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

    // Validate passwords match
    if (formData.password !== formData.confirmPassword) {
      setFormError("Passwords do not match");
```

```
      return;
    }

    // Validate password length
    if (formData.password.length < 6) {
      setFormError("Password must be at least 6 characters");
      return;
    }

    try {
      setIsLoading(true);
      // Send registration data to backend
      const userData = {
        fullName: formData.fullName,
        email: formData.email,
        password: formData.password,
        role: "Customer" // Always Customer for this form
      };

      await register(userData);

      // Redirect to login page with success message
      navigate("/login", {
        state: {
          message: "Welcome! Your customer account has been created successfully.
Please login to start chatting with our team.",
          email: formData.email
        }
      });
    } catch (err) {
      setFormError(err.response?.data?.message || "Registration failed. Please
try again.");
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <Layout>
      <div className="flex-grow flex items-center justify-center py-12 px-4
sm:px-6 lg:px-8 bg-gray-50 dark:bg-slate-800">
        <div className="max-w-md w-full space-y-8 bg-white dark:bg-slate-900
transition-all duration-200 ease-out p-8 rounded-lg shadow-md dark:shadow-
black/25">
          <div>
            <h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900
dark:text-white">
              Create Customer Account
            </h2>
            <p className="mt-2 text-center text-sm text-gray-600 dark:text-
gray-400">
              Join our platform to chat with our team and get support
            </p>
            <p className="mt-1 text-center text-sm text-gray-600 dark:text-
gray-400">
              Already have an account?{" "}
              <Link to="/login" className="font-medium text-blue-600 hover:text-
blue-500">
                Sign in here
```

```
              </Link>
            </p>
          </div>

          {(formError || error) && (
            <div className="p-3 bg-red-50 text-red-700 border border-red-200
rounded-md">
              {formError || error}
            </div>
          )}

          <form className="mt-8 space-y-6" onSubmit={handleSubmit}>
            <div className="space-y-4">
              <div>
                <label htmlFor="fullName" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                  Full Name
                </label>
                <input
                  id="fullName"
                  name="fullName"
                  type="text"
                  autoComplete="name"
                  required
                  className="appearance-none relative block w-full border border-
gray-300 dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900
dark:text-white placeholder-gray-500 dark:placeholder-gray-500 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"
                  placeholder="Enter your full name"
                  value={formData.fullName}
                  onChange={handleChange}
                />
              </div>

              <div>
                <label htmlFor="email" className="block text-sm font-medium text-
gray-700 dark:text-gray-400 mb-1">
                  Email Address
                </label>
                <input
                  id="email"
                  name="email"
                  type="email"
                  autoComplete="email"
                  required
                  className="appearance-none relative block w-full border border-
gray-300 dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900
dark:text-white placeholder-gray-500 dark:placeholder-gray-500 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"
                  placeholder="Enter your email address"
                  value={formData.email}
                  onChange={handleChange}
                />
              </div>

              <div>
                <label htmlFor="password" className="block text-sm font-medium
```

```
text-gray-700 dark:text-gray-400 mb-1">
                  Password
                </label>
                <input
                  id="password"
                  name="password"
                  type="password"
                  autoComplete="new-password"
                  required
                  className="appearance-none relative block w-full border border-
gray-300 dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900
dark:text-white placeholder-gray-500 dark:placeholder-gray-500 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"
                  placeholder="Create a password (min 6 characters)"
                  value={formData.password}
                  onChange={handleChange}
                  minLength={6}
                />
              </div>

              <div>
                <label htmlFor="confirmPassword" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                  Confirm Password
                </label>
                <input
                  id="confirmPassword"
                  name="confirmPassword"
                  type="password"
                  autoComplete="new-password"
                  required
                  className="appearance-none relative block w-full border border-
gray-300 dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900
dark:text-white placeholder-gray-500 dark:placeholder-gray-500 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"
                  placeholder="Confirm your password"
                  value={formData.confirmPassword}
                  onChange={handleChange}
                  minLength={6}
                />
              </div>
            </div>

            <div>
              <button
                type="submit"
                disabled={isLoading}
                className={`group relative w-full flex justify-center py-2 px-4
border border-transparent rounded-md text-white ${
                  isLoading ? 'bg-blue-400' : 'bg-blue-600 hover:bg-blue-700'
                } focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-
blue-500 dark:focus:ring-blue-400 transition`}
              >
                {isLoading ? 'Creating Account...' : 'Create Customer Account'}
              </button>
            </div>
```

```
            <div className="text-center">
              <p className="text-xs text-gray-500 dark:text-gray-500">
                By creating an account, you agree to our terms of service and can
chat with our support team.
              </p>
            </div>
          </form>
        </div>
      </div>
    </Layout>
  );
};

export default CustomerSignUp;
```

src/components/Auth/ForgotPassword.jsx

```
import React, { useEffect, useState } from "react";
import banner from "../../assets/ForgotPassword.jpg";
import logo from "../../assets/ForgotPassword.jpg";
import { motion } from "framer-motion";
import { FaRegEyeSlash, FaEye } from "react-icons/fa6";
import { useNavigate } from "react-router-dom";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import { authAPI } from "../../services/api"; // Import the authAPI service

const ForgotPassword = () => {
  const navigate = useNavigate();

  const [error, setError] = useState(false);
  const [showEmail, setShowEmail] = useState(true);
  const [successMessage, setSuccessMessage] = useState(false);
  const [loading, setLoading] = useState(false);
  const [payload, setPayload] = useState({
    email: "",
    otp: "",
    newPassword: "",
    confirmPassword: "",
  });
  const [passwordVisible, setPasswordVisible] = useState(false);

  const inputRefs = React.useRef([]);

  const handleOtpChange = (e, index) => {
    const newOtpArray = [...inputRefs.current.map((input) => input?.value || "")];
    newOtpArray[index] = e.target.value;

    // Auto-advance to next field
    if (e.target.value.length > 0 && index < inputRefs.current.length - 1) {
      inputRefs.current[index + 1].focus();
    }
  };

  const handleKeyDown = (e, index) => {
    if (e.key === "Backspace" && e.target.value === "" && index > 0) {
      inputRefs.current[index - 1].focus();
    }
```

```
};

const handlePaste = (e) => {
  e.preventDefault();
  const paste = e.clipboardData.getData("text");
  const pasteArray = paste.split("");

  inputRefs.current.forEach((input, index) => {
    if (index < pasteArray.length && input) {
      input.value = pasteArray[index];
    }
  });

  if (inputRefs.current[pasteArray.length - 1]) {
    inputRefs.current[pasteArray.length - 1].focus();
  }
};

const [showOtp, setShowOtp] = useState(false);
const [showNewPassword, setShowNewPassword] = useState(false);

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

const handleEmailSubmit = async (e) => {
  e.preventDefault();
  try {
    setLoading(true);
    if (!payload.email.trim()) {
      setError("Email is required");
      toast.error("Email is required");
      setLoading(false);
      return;
    }

    const response = await authAPI.sendOTP(payload.email);
    const result = response.data;

    if (result.success) {
      setError(false);
      setShowOtp(true);
      setShowEmail(false);
      setLoading(false);
      setSuccessMessage("OTP sent successfully");
      toast.success("OTP sent successfully to your email!");
    } else {
      console.log("result", result);
      toast.error(result.message || "Failed to send OTP");
      setError(result.message || "Failed to send OTP");
      setLoading(false);
    }
  } catch (e) {
    setLoading(false);
    console.error(e);
    if (e.message === "Network Error") {
```

```
        toast.error("Network error. Please check your internet connection or
contact support.");
        setError("Network error. Please check your internet connection or contact
support.");
      } else {
        toast.error(e.response?.data?.message || "Failed to send OTP. Please try
again later.");
        setError(e.response?.data?.message || "Failed to send OTP. Please try
again later.");
      }
    }
  };

  const handleOtpSubmit = async (e) => {
    e.preventDefault();
    const otpArray = inputRefs.current.map((input) => input?.value || "");
    const otp = otpArray.join("");

    if (otp.length !== 6) {
      toast.error("Please enter the complete 6-digit OTP");
      setError("Please enter the complete 6-digit OTP");
      return;
    }

    try {
      setLoading(true);
      console.log("payload.email", payload.email);
      console.log("otp", otp);
      const response = await authAPI.verifyOTP({ otp, email: payload.email });
      const result = response.data;

      if (result.success) {
        setError(false);
        setShowNewPassword(true);
        setShowOtp(false);
        setLoading(false);
        setSuccessMessage("OTP verified successfully. Enter your new password");
        toast.success("OTP verified successfully!");
      } else {
        setSuccessMessage(false);
        toast.error(result.message || "Invalid OTP");
        setError(result.message || "Invalid OTP");
        setLoading(false);
      }
    } catch (e) {
      setLoading(false);
      console.error(e);
      if (e.message === "Network Error") {
        toast.error("Network error. Please check your internet connection or
contact support.");
        setError("Network error. Please check your internet connection or contact
support.");
      } else {
        toast.error(e.response?.data?.message || "Failed to verify OTP. Please
try again later.");
        setError(e.response?.data?.message || "Failed to verify OTP. Please try
again later.");
      }
    }
```

```
  };

  const handleResetPassword = async (e) => {
    setSuccessMessage(false);
    e.preventDefault();

    if (payload.newPassword !== payload.confirmPassword) {
      toast.error("Passwords do not match");
      setError("Passwords do not match");
      return;
    }

    if (payload.newPassword.length < 8) {
      toast.error("Password must be at least 8 characters long");
      setError("Password must be at least 8 characters long");
      return;
    }

    try {
      setLoading(true);
      console.log("newPassword sending to backend is ", payload.newPassword);
      const response = await authAPI.resetPassword({
        email: payload.email,
        newPassword: payload.newPassword
      });
      const result = response.data;

      if (result.success) {
        setError(false);
        setLoading(false);
        toast.success("Password changed successfully!");
        setTimeout(() => {
          navigate("/login");
        }, 2000);
      } else {
        toast.error(result.message || "Failed to reset password");
        setError(result.message || "Failed to reset password");
        setLoading(false);
      }
    } catch (e) {
      setLoading(false);
      console.error(e);
      if (e.message === "Network Error") {
        toast.error("Network error. Please check your internet connection or
contact support.");
        setError("Network error. Please check your internet connection or contact
support.");
      } else {
        toast.error(e.response?.data?.message || "Failed to reset password.
Please try again later.");
        setError(e.response?.data?.message || "Failed to reset password. Please
try again later.");
      }
    }
  };

  // Animation variants
  const containerVariants = {
    hidden: { opacity: 0 },
```

```
    visible: { opacity: 1, transition: { duration: 0.5 } },
    exit: { opacity: 0, transition: { duration: 0.5 } }
  };

  const formVariants = {
    hidden: { opacity: 0, y: 20 },
    visible: {
      opacity: 1,
      y: 0,
      transition: {
        duration: 0.6,
        when: "beforeChildren",
        staggerChildren: 0.1
      }
    },
    exit: {
      opacity: 0,
      y: -20,
      transition: { duration: 0.3 }
    }
  };

  const itemVariants = {
    hidden: { opacity: 0, y: 10 },
    visible: { opacity: 1, y: 0, transition: { duration: 0.5 } }
  };

  return (
    <div
      className="w-full min-h-screen bg-cover bg-center relative"
      style={{ backgroundImage: `url(${banner})` }}
    >
      <ToastContainer position="top-right" autoClose={3000} />
      <div className="absolute inset-0 bg-black opacity-50"></div>

      <motion.div
        className="absolute inset-0 flex justify-center items-center p-4"
        initial="hidden"
        animate="visible"
        exit="exit"
        variants={containerVariants}
      >
        <div className="grid grid-cols-1 md:flex max-w-5xl w-full">
          <motion.div
            className="bg-[#152B54] w-full md:w-[40%] p-6 hidden md:flex flex-col
justify-center items-center"
            initial={{ x: -50, opacity: 0 }}
            animate={{ x: 0, opacity: 1 }}
            transition={{ duration: 0.8 }}
          >
            <div className="relative">
              <img
                src={logo}
                alt="Traincape Technology"
                className="w-[130px] h-[130px] mb-8"
              />
            </div>
            <motion.div
              initial={{ opacity: 0 }}
```

```
              animate={{ opacity: 1 }}
              transition={{ delay: 0.5, duration: 1 }}
              className="text-white text-center"
            >
              <h2 className="text-2xl font-bold mb-4">Password Recovery</h2>
              <p className="mb-6">Recover your account access quickly and
securely</p>
              <div className="w-16 h-1 bg-white dark:bg-slate-900 transition-all
duration-200 ease-out mx-auto"></div>
            </motion.div>
          </motion.div>

          <motion.div
            className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out bg-opacity-95 p-8 md:p-12 shadow-xl w-full md:w-[60%] rounded-lg"
            variants={formVariants}
            initial="hidden"
            animate="visible"
            exit="exit"
          >
            <motion.h1
              className="text-3xl font-semibold text-center text-[#152B54] mb-6"
              variants={itemVariants}
            >
              Forgot Password
            </motion.h1>

            {error && (
              <motion.div
                className="mb-4 p-3 bg-red-100 text-red-800 rounded-md"
                initial={{ opacity: 0, y: -10 }}
                animate={{ opacity: 1, y: 0 }}
                transition={{ duration: 0.3 }}
              >
                {error}
              </motion.div>
            )}

            {successMessage && (
              <motion.div
                className="mb-4 p-3 bg-green-100 text-green-800 rounded-md"
                initial={{ opacity: 0, y: -10 }}
                animate={{ opacity: 1, y: 0 }}
                transition={{ duration: 0.3 }}
              >
                {successMessage}
              </motion.div>
            )}

            <form>
              {/* Email Form */}
              {showEmail && (
                <motion.div variants={formVariants}>
                  <motion.div className="mb-6" variants={itemVariants}>
                    <label
                      htmlFor="email"
                      className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-2"
                    >
```

```
                    Email Address
                  </label>
                  <input
                    type="email"
                    id="email"
                    name="email"
                    required
                    value={payload.email}
                    onChange={handleChange}
                    placeholder="Enter your email address"
                    className="w-full px-4 py-3 mt-1 border border-gray-300
dark:border-slate-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-
[#152B54] transition-all duration-200"
                  />
                </motion.div>
                <motion.button
                  type="button"
                  onClick={handleEmailSubmit}
                  disabled={loading}
                  variants={itemVariants}
                  whileHover={{ scale: 1.02 }}
                  whileTap={{ scale: 0.98 }}
                  className="w-full py-3 bg-[#152B54] text-white rounded-lg
hover:bg-sky-950 transition duration-200 flex justify-center items-center"
                >
                  {loading ? (
                    <div className="flex items-center">
                      <div className="animate-spin mr-2 h-5 w-5 border-t-2
border-b-2 border-white rounded-full"></div>
                      <span>Sending...</span>
                    </div>
                  ) : (
                    "Send Reset OTP"
                  )}
                </motion.button>
                <motion.div
                  className="mt-4 text-center"
                  variants={itemVariants}
                >
                  <button
                    type="button"
                    onClick={() => navigate('/login')}
                    className="text-[#152B54] hover:underline"
                  >
                    Back to Login
                  </button>
                </motion.div>
              </motion.div>
            )}

            {/* OTP Form */}
            {showOtp && (
              <motion.div
                className="mt-4"
                variants={formVariants}
                initial="hidden"
                animate="visible"
                exit="exit"
              >
```

```
                  <motion.h2
                    className="text-xl font-semibold text-center text-[#152B54]
mb-4"
                    variants={itemVariants}
                  >
                    Enter Verification Code
                  </motion.h2>
                  <motion.p
                    className="text-center text-gray-600 dark:text-gray-400 mb-6"
                    variants={itemVariants}
                  >
                    We've sent a 6-digit code to your email
                  </motion.p>
                  <motion.div
                    onPaste={handlePaste}
                    className="flex justify-between mb-6"
                    variants={itemVariants}
                  >
                    {Array(6)
                      .fill(0)
                      .map((_, index) => (
                        <motion.input
                          type="text"
                          maxLength="1"
                          key={index}
                          required
                          onChange={(e) => handleOtpChange(e, index)}
                          onKeyDown={(e) => handleKeyDown(e, index)}
                          ref={(el) => (inputRefs.current[index] = el)}
                          whileFocus={{ scale: 1.1 }}
                          className="w-12 h-12 text-3xl text-center border-2
border-gray-300 dark:border-slate-600 rounded-lg focus:outline-none focus:ring-2
focus:ring-[#152B54] focus:border-[#152B54] transition-all duration-200"
                        />
                      ))}
                  </motion.div>
                  <motion.button
                    type="button"
                    onClick={handleOtpSubmit}
                    disabled={loading}
                    variants={itemVariants}
                    whileHover={{ scale: 1.02 }}
                    whileTap={{ scale: 0.98 }}
                    className="w-full py-3 mt-2 bg-[#152B54] text-white rounded-
lg hover:bg-sky-950 transition duration-200 flex justify-center items-center"
                  >
                    {loading ? (
                      <div className="flex items-center">
                        <div className="animate-spin mr-2 h-5 w-5 border-t-2
border-b-2 border-white rounded-full"></div>
                        <span>Verifying...</span>
                      </div>
                    ) : (
                      "Verify OTP"
                    )}
                  </motion.button>
                  <motion.div
                    className="mt-4 text-center"
                    variants={itemVariants}
```

```
                >
                  <button
                    type="button"
                    onClick={handleEmailSubmit}
                    className="text-[#152B54] hover:underline"
                  >
                    Didn't receive code? Resend
                  </button>
                </motion.div>
              </motion.div>
            )}

            {/* New Password Form */}
            {showNewPassword && (
              <motion.div
                className="mt-4"
                variants={formVariants}
                initial="hidden"
                animate="visible"
                exit="exit"
              >
                <motion.h2
                  className="text-xl font-semibold text-center text-[#152B54]
mb-4"
                  variants={itemVariants}
                >
                  Create New Password
                </motion.h2>
                <motion.p
                  className="text-center text-gray-600 dark:text-gray-400 mb-6"
                  variants={itemVariants}
                >
                  Your new password must be at least 8 characters
                </motion.p>
                <motion.div className="mb-5 relative" variants={itemVariants}>
                  <label
                    htmlFor="newPassword"
                    className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-2"
                  >
                    New Password
                  </label>
                  <div className="relative">
                    <input
                      type={passwordVisible ? "text" : "password"}
                      id="newPassword"
                      name="newPassword"
                      value={payload.newPassword}
                      onChange={handleChange}
                      required
                      placeholder="Enter new password"
                      className="w-full px-4 py-3 border border-gray-300
dark:border-slate-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-
[#152B54] transition-all duration-200"
                    />
                    <button
                      type="button"
                      className="absolute inset-y-0 right-0 pr-3 flex items-
center text-gray-500 dark:text-gray-300 hover:text-gray-700"
```

```
                        onClick={() => setPasswordVisible(!passwordVisible)}
                      >
                        {passwordVisible ? <FaRegEyeSlash size={18} /> : <FaEye
size={18} />}
                      </button>
                    </div>
                  </motion.div>
                  <motion.div className="mb-6" variants={itemVariants}>
                    <label
                      htmlFor="confirmPassword"
                      className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-2"
                    >
                      Confirm Password
                    </label>
                    <input
                      type="password"
                      id="confirmPassword"
                      name="confirmPassword"
                      value={payload.confirmPassword}
                      onChange={handleChange}
                      required
                      placeholder="Confirm new password"
                      className="w-full px-4 py-3 border border-gray-300
dark:border-slate-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-
[#152B54] transition-all duration-200"
                    />
                  </motion.div>
                  <motion.button
                    type="button"
                    onClick={handleResetPassword}
                    disabled={loading}
                    variants={itemVariants}
                    whileHover={{ scale: 1.02 }}
                    whileTap={{ scale: 0.98 }}
                    className="w-full py-3 bg-[#152B54] text-white rounded-lg
hover:bg-sky-950 transition duration-200 flex justify-center items-center"
                  >
                    {loading ? (
                      <div className="flex items-center">
                        <div className="animate-spin mr-2 h-5 w-5 border-t-2
border-b-2 border-white rounded-full"></div>
                        <span>Resetting Password...</span>
                      </div>
                    ) : (
                      "Reset Password"
                    )}
                  </motion.button>
                </motion.div>
              )}
            </form>
          </motion.div>
        </div>
      </motion.div>
    </div>
  );
};

export default ForgotPassword;
```

src/components/Auth/Login.jsx

```
import React, { useState, useEffect } from "react";Ð
import { useNavigate, useLocation, Link } from "react-router-dom";Ð
import Layout from "../Layout/Layout";Ð
import { useAuth } from "../../context/AuthContext";Ð
Ð
const Login = () => {Ð
  const navigate = useNavigate();Ð
  const location = useLocation();Ð
  const { login, error } = useAuth();Ð
  const [formData, setFormData] = useState({Ð
    email: "",Ð
    password: ""Ð
  });Ð
  const [formError, setFormError] = useState("");Ð
  const [successMessage, setSuccessMessage] = useState("");Ð
  const [isLoading, setIsLoading] = useState(false);Ð
  const [showPassword, setShowPassword] = useState(false);Ð
Ð
  // Check for success message from signup pageÐ
  useEffect(() => {Ð
    if (location.state?.message) {Ð
      setSuccessMessage(location.state.message);Ð
      Ð
      // Pre-fill email if providedÐ
      if (location.state.email) {Ð
        setFormData(prev => ({Ð
          ...prev,Ð
          email: location.state.emailÐ
        }));Ð
      }Ð
    }Ð
  }, [location]);Ð
Ð
  const handleChange = (e) => {Ð
    setFormData({Ð
      ...formData,Ð
      [e.target.name]: e.target.valueÐ
    });Ð
  };Ð
Ð
  const togglePasswordVisibility = () => {Ð
    setShowPassword(!showPassword);Ð
  };Ð
Ð
  const handleSubmit = async (e) => {Ð
    e.preventDefault();Ð
    setFormError("");Ð
    setSuccessMessage("");Ð
Ð
    try {Ð
      setIsLoading(true);Ð
      const response = await login(formData);Ð
      Ð
      if (response.success) {Ð
        // Redirect based on user roleÐ
        const userData = response.data;Ð
```

```
        switch(userData.role) {Đ
          case "Lead Person":Đ
          case "Manager":Đ
          case "Admin":Đ
            navigate("/leads");Đ
            break;Đ
          case "Sales Person":Đ
            navigate("/sales");Đ
            break;Đ
          case "Customer":Đ
            navigate("/customer");Đ
            break;Đ
          default:Đ
            navigate("/");Đ
        }Đ
      } else {Đ
        setFormError(response.message || "Invalid credentials. Please try
again.");Đ
      }Đ
    } catch (err) {Đ
      setFormError(err.response?.data?.message || "Invalid credentials. Please
try again.");Đ
    } finally {Đ
      setIsLoading(false);Đ
    }Đ
  };Đ
Đ
  return (Đ
    <Layout>Đ
      <div className="flex-grow flex items-center justify-center py-12 px-4
sm:px-6 lg:px-8 bg-gray-50 dark:bg-slate-800">Đ
        <div className="max-w-md w-full space-y-8 bg-white dark:bg-slate-900
transition-all duration-200 ease-out p-8 rounded-lg shadow-md dark:shadow-
black/25">Đ
          <div>Đ
            <h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900
dark:text-white">Đ
              Sign in to your accountĐ
            </h2>Đ
            {/* <p className="mt-2 text-center text-sm text-gray-600 dark:text-
gray-500">Đ
              Or{" "}Đ
              <Link to="/signup" className="font-medium text-blue-600 hover:text-
blue-500">Đ
                create a new accountĐ
              </Link>Đ
            </p> */}Đ
          </div>Đ
          Đ
          {successMessage && (Đ
            <div className="p-3 bg-green-50 text-green-700 border border-
green-200 rounded-md">Đ
              {successMessage}Đ
            </div>Đ
          )}Đ
          Đ
          {(formError || error) && (Đ
            <div className="p-3 bg-red-50 text-red-700 border border-red-200
rounded-md">Đ
```

```
              {formError || error}Ð
            </div>Ð
          )}Ð
          Ð
          <form className="mt-8 space-y-6" onSubmit={handleSubmit}>Ð
            <div className="rounded-md -space-y-px">Ð
              <div className="mb-4">Ð
                <label htmlFor="email" className="block text-sm font-medium text-
gray-700 dark:text-gray-400 mb-1">Ð
                  Email addressÐ
                </label>Ð
                <inputÐ
                  id="email"Ð
                  name="email"Ð
                  type="email"Ð
                  autoComplete="email"Ð
                  requiredÐ
                  className="appearance-none relative block w-full px-3 py-2
border border-gray-300 dark:border-gray-600 rounded bg-white dark:bg-gray-700
text-gray-900 dark:text-gray-100 placeholder-gray-500 dark:placeholder-gray-400
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500 focus:z-10 sm:text-sm"Ð
                  placeholder="Email address"Ð
                  value={formData.email}Ð
                  onChange={handleChange}Ð
                />Ð
              </div>Ð
              Ð
              <div className="mb-4">Ð
                <label htmlFor="password" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">Ð
                  PasswordÐ
                </label>Ð
                <div className="relative">Ð
                  <inputÐ
                    id="password"Ð
                    name="password"Ð
                    type={showPassword ? "text" : "password"}Ð
                    autoComplete="current-password"Ð
                    requiredÐ
                    className="appearance-none relative block w-full px-3 py-2
border border-gray-300 dark:border-gray-600 rounded bg-white dark:bg-gray-700
text-gray-900 dark:text-gray-100 placeholder-gray-500 dark:placeholder-gray-400
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500 focus:z-10 sm:text-sm pr-10"Ð
                    placeholder="Password"Ð
                    value={formData.password}Ð
                    onChange={handleChange}Ð
                  />Ð
                  <buttonÐ
                    type="button"Ð
                    className="absolute inset-y-0 right-0 pr-3 flex items-center
text-sm text-gray-500 dark:text-gray-200 hover:text-gray-800"Ð
                    onClick={togglePasswordVisibility}Ð
                  >Ð
                    {showPassword ? (Ð
                      <svg xmlns="http://www.w3.org/2000/svg" className="h-5 w-5"
viewBox="0 0 20 20" fill="currentColor">Ð
                        <path d="M10 12a2 2 0 100-4 2 2 0 000 4z" />Ð
```

```
                        <path fillRule="evenodd" d="M.458 10C1.732 5.943 5.522 3
10 3s8.268 2.943 9.542 7c-1.274 4.057-5.064 7-9.542 7S1.732 14.057.458 10zM14
10a4 4 0 11-8 0 4 4 0 018 0z" clipRule="evenodd" />Ð
                      </svg>Ð
                    ) : (Ð
                      <svg xmlns="http://www.w3.org/2000/svg" className="h-5 w-5"
viewBox="0 0 20 20" fill="currentColor">Ð
                        <path fillRule="evenodd" d="M3.707 2.293a1 1 0 00-1.414
1.414l14 14a1 1 0 001.414-1.414l-1.473-1.473A10.014 10.014 0 0019.542 10C18.268
5.943 14.478 3 10 3a9.958 9.958 0 00-4.512 1.074l-1.78-1.781zm4.261 4.26l1.514
1.515a2.003 2.003 0 012.45 2.45l1.514 1.514a4 4 0 00-5.478-5.478z"
clipRule="evenodd" />Ð
                        <path d="M12.454 16.697L9.75 13.992a4 4 0
01-3.742-3.741L2.335 6.578A9.98 9.98 0 00.458 10c1.274 4.057 5.065 7 9.542 7 .847
0 1.669-.105 2.454-.303z" />Ð
                      </svg>Ð
                    )}Ð
                  </button>Ð
                </div>Ð
              </div>Ð
            </div>Ð
Ð
            <div className="flex items-center justify-between">Ð
              <div className="flex items-center">Ð
                <inputÐ
                  id="remember-me"Ð
                  name="remember-me"Ð
                  type="checkbox"Ð
                  className="h-4 w-4 text-blue-600 focus:ring-blue-500
dark:focus:ring-blue-400 border-gray-300 dark:border-slate-600 rounded"Ð
                />Ð
                <label htmlFor="remember-me" className="ml-2 block text-sm text-
gray-900 dark:text-white">Ð
                  Remember meÐ
                </label>Ð
              </div>Ð
Ð
              <div className="text-sm">Ð
                <Link to="/forgot-password" className="font-medium text-blue-600
dark:text-blue-400 hover:text-blue-500 dark:hover:text-blue-300">Ð
                  Forgot your password?Ð
                </Link>Ð
              </div>Ð
            </div>Ð
Ð
            <div>Ð
              <buttonÐ
                type="submit"Ð
                disabled={isLoading}Ð
                className={`group relative w-full flex justify-center py-2 px-4
border border-transparent rounded-md text-white ${Ð
                  isLoading ? 'bg-blue-400' : 'bg-blue-600 hover:bg-blue-700'Ð
                } focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-
blue-500 dark:focus:ring-blue-400 transition`}Ð
              >Ð
                {isLoading ? 'Signing in...' : 'Sign in'}Ð
              </button>Ð
            </div>Ð
          </form>Ð
```

```
        </div>Đ
      </div>Đ
    </Layout>Đ
  );Đ
};Đ
Đ
export default Login;
```

src/components/Auth/SignUp.jsx

```
import React, { useState } from "react";Đ
import { useNavigate, Link } from "react-router-dom";Đ
import Layout from "../Layout/Layout";Đ
import { useAuth } from "../../context/AuthContext";Đ
Đ
const SignUp = () => {Đ
  const navigate = useNavigate();Đ
  const { register, error } = useAuth();Đ
  const [formData, setFormData] = useState({Đ
    fullName: "",Đ
    email: "",Đ
    password: "",Đ
    confirmPassword: "",Đ
    role: "Customer" // Default role for public registrationĐ
  });Đ
  const [formError, setFormError] = useState("");Đ
  const [isLoading, setIsLoading] = useState(false);Đ
Đ
  const handleChange = (e) => {Đ
    setFormData({Đ
      ...formData,Đ
      [e.target.name]: e.target.valueĐ
    });Đ
  };Đ
Đ
  const handleSubmit = async (e) => {Đ
    e.preventDefault();Đ
    setFormError("");Đ
Đ
    // Validate passwords matchĐ
    if (formData.password !== formData.confirmPassword) {Đ
      setFormError("Passwords do not match");Đ
      return;Đ
    }Đ
Đ
    // Validate password lengthĐ
    if (formData.password.length < 6) {Đ
      setFormError("Password must be at least 6 characters");Đ
      return;Đ
    }Đ
Đ
    try {Đ
      setIsLoading(true);Đ
      // Send registration data to backendĐ
      const userData = {Đ
        fullName: formData.fullName,Đ
        email: formData.email,Đ
        password: formData.password,Đ
```

```
        role: formData.roleÐ
      };Ð
Ð
      await register(userData);Ð
      Ð
      // Instead of redirecting based on role, redirect to login page with
success messageÐ
      navigate("/login", { Ð
        state: { Ð
          message: "Registration successful! Please login with your credentials.",Ð
          email: formData.emailÐ
        } Ð
      });Ð
    } catch (err) {Ð
      setFormError(err.response?.data?.message || "Registration failed. Please
try again.");Ð
    } finally {Ð
      setIsLoading(false);Ð
    }Ð
  };Ð
Ð
  return (Ð
    <Layout>Ð
      <div className="flex-grow flex items-center justify-center py-12 px-4
sm:px-6 lg:px-8 bg-gray-50 dark:bg-slate-800">Ð
        <div className="max-w-md w-full space-y-8 bg-white dark:bg-slate-900
transition-all duration-200 ease-out p-8 rounded-lg shadow-md dark:shadow-
black/25">Ð
          <div>Ð
            <h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900
dark:text-white">Ð
              Create a new accountÐ
            </h2>Ð
            <p className="mt-2 text-center text-sm text-gray-600 dark:text-
gray-400">Ð
              Or{" "}Ð
              <Link to="/login" className="font-medium text-blue-600 hover:text-
blue-500">Ð
                sign in to your existing accountÐ
              </Link>Ð
            </p>Ð
          </div>Ð
          Ð
          {(formError || error) && (Ð
            <div className="p-3 bg-red-50 text-red-700 border border-red-200
rounded-md">Ð
              {formError || error}Ð
            </div>Ð
          )}Ð
          Ð
          <form className="mt-8 space-y-6" onSubmit={handleSubmit}>Ð
            <div className="rounded-md -space-y-px">Ð
              <div className="mb-4">Ð
                <label htmlFor="fullName" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">Ð
                  Full NameÐ
                </label>Ð
                <inputÐ
                  id="fullName"Ð
```

```
                  name="fullName"Đ
                  type="text"Đ
                  autoComplete="name"Đ
                  requiredĐ
                  className="appearance-none relative block w-full px-3 py-2
border border-gray-300 dark:border-slate-600 rounded-md focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"Đ
                  placeholder="Full Name"Đ
                  value={formData.fullName}Đ
                  onChange={handleChange}Đ
                />Đ
              </div>Đ
              Đ
              <div className="mb-4">Đ
                <label htmlFor="email" className="block text-sm font-medium text-
gray-700 dark:text-gray-400 mb-1">Đ
                  Email addressĐ
                </label>Đ
                <inputĐ
                  id="email"Đ
                  name="email"Đ
                  type="email"Đ
                  autoComplete="email"Đ
                  requiredĐ
                  className="appearance-none relative block w-full px-3 py-2
border border-gray-300 dark:border-slate-600 rounded-md focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"Đ
                  placeholder="Email address"Đ
                  value={formData.email}Đ
                  onChange={handleChange}Đ
                />Đ
              </div>Đ
              Đ
              <div className="mb-4">Đ
                <label htmlFor="password" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">Đ
                  PasswordĐ
                </label>Đ
                <inputĐ
                  id="password"Đ
                  name="password"Đ
                  type="password"Đ
                  autoComplete="new-password"Đ
                  requiredĐ
                  className="appearance-none relative block w-full px-3 py-2
border border-gray-300 dark:border-slate-600 rounded-md focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"Đ
                  placeholder="Password"Đ
                  value={formData.password}Đ
                  onChange={handleChange}Đ
                />Đ
              </div>Đ
              Đ
              <div className="mb-4">Đ
                <label htmlFor="confirmPassword" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">Đ
```

```
                  Confirm PasswordÐ
                </label>Ð
                <inputÐ
                  id="confirmPassword"Ð
                  name="confirmPassword"Ð
                  type="password"Ð
                  autoComplete="new-password"Ð
                  requiredÐ
                  className="appearance-none relative block w-full px-3 py-2
border border-gray-300 dark:border-slate-600 rounded-md focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"Ð
                  placeholder="Confirm Password"Ð
                  value={formData.confirmPassword}Ð
                  onChange={handleChange}Ð
                />Ð
              </div>Ð
              Ð
              <div>Ð
                <label htmlFor="role" className="block text-sm font-medium text-
gray-700 dark:text-gray-400 mb-1">Ð
                  RoleÐ
                </label>Ð
                <selectÐ
                  id="role"Ð
                  name="role"Ð
                  className="appearance-none relative block w-full px-3 py-2
border border-gray-300 dark:border-slate-600 rounded-md focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500 focus:z-10
sm:text-sm"Ð
                  value={formData.role}Ð
                  onChange={handleChange}Ð
                >Ð
                  <option value="Lead Person">Lead Person</option>Ð
                  <option value="Sales Person">Sales Person</option>Ð
                  <option value="Manager">Manager</option>Ð
                  <option value="Admin">Admin</option>Ð
                  <option value="Customer">Customer</option>Ð
                </select>Ð
                <p className="text-xs text-gray-500 dark:text-gray-500 mt-1">For
initial setup, you can create an Admin account. This option will be removed in
production.</p>Ð
              </div>Ð
            </div>Ð
Ð
            <div>Ð
              <buttonÐ
                type="submit"Ð
                disabled={isLoading}Ð
                className={`group relative w-full flex justify-center py-2 px-4
border border-transparent rounded-md text-white ${Ð
                  isLoading ? 'bg-blue-400' : 'bg-blue-600 hover:bg-blue-700'Ð
                } focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-
blue-500 dark:focus:ring-blue-400 transition`}Ð
              >Ð
                {isLoading ? 'Creating Account...' : 'Create Account'}Ð
              </button>Ð
            </div>Ð
          </form>Ð
```

```
        </div>Đ
      </div>Đ
    </Layout>Đ
  );Đ
};Đ
Đ
export default SignUp;
```

src/components/Chat/ChatDebug.jsx

```
import React, { useState } from 'react';
import { useAuth } from '../../context/AuthContext';
import { useChat } from '../../context/ChatContext';

const ChatDebug = () => {
  const { user, token } = useAuth();
  const { socket, isConnected } = useChat();
  const [showDebug, setShowDebug] = useState(false);

  if (!showDebug) {
    return (
      <button
        onClick={() => setShowDebug(true)}
        style={{
          position: 'fixed',
          top: '10px',
          right: '10px',
          // background: '#f59e0b',
          // color: 'white',
          // border: 'none',
          // padding: '8px 12px',
          // borderRadius: '4px',
          // fontSize: '12px',
          // cursor: 'pointer',
          // zIndex: 9999
        }}
      >
        {/* Ø=Ü  Debug Chat */}
      </button>
    );
  }

  const testConnection = async () => {
    try {
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const serverUrl = isDevelopment ? 'http://localhost:8080' : 'https://crm-
backend-o36v.onrender.com';
      const response = await fetch(`${serverUrl}/api/chat/users`, {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      });

      if (response.ok) {
        const data = await response.json();
        alert(`'  API Connection OK! Found ${data.data.length} users`);
```

```
    } else {
      alert(`'L API Error: ${response.status} ${response.statusText}`);
    }
  } catch (error) {
    alert(`'L Connection Error: ${error.message}`);
  }
};

const testSocketConnection = () => {
  if (socket) {
    socket.emit('test-connection', { message: 'Hello from debug' });
    alert('  Socket test message sent');
  } else {
    alert(''L No socket connection available');
  }
};

const requestNotificationPermission = async () => {
  if ('Notification' in window) {
    const permission = await Notification.requestPermission();
    if (permission === 'granted') {
      new Notification('Chat Notifications Enabled!', {
        body: 'You will now receive chat notifications.',
        icon: '/favicon.ico'
      });
      alert('  Notifications enabled!');
    } else {
      alert(''L Notifications denied');
    }
  } else {
    alert(''L Notifications not supported');
  }
};

return (
  <div style={{
    position: 'fixed',
    top: '10px',
    right: '10px',
    background: 'white',
    border: '2px solid #ccc',
    borderRadius: '8px',
    padding: '15px',
    zIndex: 9999,
    minWidth: '300px',
    boxShadow: '0 4px 12px rgba(0,0,0,0.15)'
  }}>
    <h3 style={{ margin: '0 0 10px 0', color: '#333' }}>Ø=Ü  Chat Debug</h3>

    <div style={{ marginBottom: '8px' }}>
      <strong>User:</strong> {user?.fullName || 'Not logged in'}
    </div>

    <div style={{ marginBottom: '8px' }}>
      <strong>Token:</strong> {token ? ''  Present' : ''L Missing'}
      {token && (
        <div style={{ fontSize: '10px', color: '#666', marginTop: '2px' }}>
          {token.substring(0, 20)}...
        </div>
```

```
        )}
      </div>

      <div style={{ marginBottom: '8px' }}>
        <strong>Socket:</strong> {socket ? ''  Created' : ''L Not created'}
      </div>

      <div style={{ marginBottom: '8px' }}>
        <strong>Connected:</strong> {isConnected ? ''  Yes' : ''L No'}
      </div>

      <div style={{ marginBottom: '12px' }}>
        <strong>Server URL:</strong> {import.meta.env.DEV &&
import.meta.env.MODE !== 'production' ? 'http://localhost:8080' : 'https://crm-
backend-o36v.onrender.com'}
      </div>

      <div style={{ display: 'flex', flexDirection: 'column', gap: '8px' }}>
        <button
          onClick={testConnection}
          style={{
            padding: '8px 12px',
            background: '#007bff',
            color: 'white',
            border: 'none',
            borderRadius: '4px',
            cursor: 'pointer'
          }}
        >
          Test API Connection
        </button>

        <button
          onClick={testSocketConnection}
          style={{
            padding: '8px 12px',
            background: '#28a745',
            color: 'white',
            border: 'none',
            borderRadius: '4px',
            cursor: 'pointer'
          }}
        >
          Test Socket Connection
        </button>

        <button
          onClick={requestNotificationPermission}
          style={{
            padding: '8px 12px',
            background: '#ffc107',
            color: 'black',
            border: 'none',
            borderRadius: '4px',
            cursor: 'pointer'
          }}
        >
          Request Permission
        </button>
```

```
        <button
          onClick={() => window.location.reload()}
          style={{
            padding: '8px 12px',
            background: '#6c757d',
            color: 'white',
            border: 'none',
            borderRadius: '4px',
            cursor: 'pointer'
          }}
        >
          Reload Page
        </button>
      </div>

      <div style={{ marginTop: '12px', fontSize: '11px', color: '#6b7280' }}>
        Ø=Ü¡ Check browser console for detailed logs
      </div>
    </div>
  );
};

export default ChatDebug;
```

src/components/Chat/ChatSoundTest.jsx

```
import React, { useState } from 'react';
import { useChat } from '../../context/ChatContext';
import notificationService from '../../services/notificationService';
import {
  FaPlay,
  FaVolumeUp,
  FaComment,
  FaUsers,
  FaExclamationTriangle,
  FaMusic,
  FaCheckCircle,
  FaTimesCircle
} from 'react-icons/fa';

const ChatSoundTest = ({ isOpen, onClose }) => {
  const { testNotificationSound } = useChat();
  const [lastPlayed, setLastPlayed] = useState('');

  const soundTests = [
    {
      type: 'message',
      name: 'Message Sound',
      icon: FaComment,
      color: '#10b981',
      description: 'Standard WhatsApp-style message notification'
    },
    {
      type: 'group',
      name: 'Group Message',
      icon: FaUsers,
      color: '#3b82f6',
```

```
    description: 'Triple tone for group conversations'
  },
  {
    type: 'urgent',
    name: 'Urgent Alert',
    icon: FaExclamationTriangle,
    color: '#ef4444',
    description: 'Rapid beeps for important messages'
  },
  {
    type: 'soft',
    name: 'Soft Notification',
    icon: FaMusic,
    color: '#8b5cf6',
    description: 'Gentle tone for quiet environments'
  },
  {
    type: 'success',
    name: 'Success Sound',
    icon: FaCheckCircle,
    color: '#06d6a0',
    description: 'Message sent confirmation'
  },
  {
    type: 'error',
    name: 'Error Sound',
    icon: FaTimesCircle,
    color: '#f72585',
    description: 'Error or failure notification'
  }
];

const handleTestSound = async (soundType, soundName) => {
  try {
    setLastPlayed(`Testing ${soundName}...`);
    await testNotificationSound(soundType);
    setLastPlayed(`'  Played: ${soundName}`);

    setTimeout(() => {
      setLastPlayed('');
    }, 3000);
  } catch (error) {
    setLastPlayed(`'L Error playing ${soundName}`);
    console.error('Error testing sound:', error);
  }
};

const testBrowserNotification = () => {
  if ('Notification' in window) {
    if (Notification.permission === 'granted') {
      new Notification('Ø=Ý  CRM Chat Test', {
        body: 'This is a test browser notification with sound!',
        icon: '/favicon.ico',
        requireInteraction: false
      });
      setLastPlayed('  Browser notification sent');
    } else if (Notification.permission !== 'denied') {
      Notification.requestPermission().then(permission => {
        if (permission === 'granted') {
```

```
            new Notification('Ø=Ý  Permission Granted!', {
              body: 'Browser notifications are now enabled',
              icon: '/favicon.ico'
            });
            setLastPlayed(''  Permission granted & notification sent');
          } else {
            setLastPlayed(''L Permission denied');
          }
        });
      } else {
        setLastPlayed(''L Notifications are blocked');
      }
    } else {
      setLastPlayed(''L Browser does not support notifications');
    }
  };

  const simulateNewMessage = () => {
    // Simulate receiving a message notification
    const mockNotification = {
      senderId: 'test-user-123',
      senderName: 'John Doe (Test)',
      content: 'Hey! This is a test message to demonstrate WhatsApp-style
notifications Ø<ß‰',
      timestamp: new Date().toISOString(),
      isGuest: false
    };

    // Trigger the notification handler directly
    notificationService.handleChatMessageNotification(mockNotification);

    setLastPlayed(''  Simulated incoming message with sound & notification');
  };

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-[9999]">
      <div className="bg-white dark:bg-gray-800 rounded-lg shadow-2xl w-full max-
w-lg mx-4 max-h-[90vh] overflow-y-auto">
        {/* Header */}
        <div className="flex items-center justify-between p-6 border-b
dark:border-gray-700">
          <h2 className="text-xl font-bold text-gray-900 dark:text-white flex
items-center gap-3">
            <FaVolumeUp className="text-blue-500" />
            Test Chat Notifications
          </h2>
          <button
            onClick={onClose}
            className="text-gray-500 hover:text-gray-700 dark:text-gray-400
dark:hover:text-gray-200 text-xl"
          >
            '
          </button>
        </div>

        {/* Content */}
```

```
        <div className="p-6 space-y-6">

          {/* Status Display */}
          {lastPlayed && (
            <div className="bg-blue-50 dark:bg-blue-900/20 border border-blue-200
dark:border-blue-800 rounded-lg p-3">
              <p className="text-blue-800 dark:text-blue-200 text-sm font-medium">
                {lastPlayed}
              </p>
            </div>
          )}

          {/* Audio Status */}
          <div className="bg-gray-50 dark:bg-gray-700 rounded-lg p-4">
            <h3 className="font-semibold text-gray-900 dark:text-white
mb-2">Audio System Status</h3>
            <div className="space-y-1 text-sm">
              <p className="text-gray-600 dark:text-gray-300">
                Ø=Ý  Audio Support: <span className="font-medium text-green-600">
                  {notificationService.sounds?.isAudioSupported() ? 'Enabled' :
'Not Available'}
                </span>
              </p>
              <p className="text-gray-600 dark:text-gray-300">
                Ø<ßµ Audio State: <span className="font-medium text-blue-600">
                  {notificationService.sounds?.getAudioState() || 'Unknown'}
                </span>
              </p>
              <p className="text-gray-600 dark:text-gray-300">
                Ø=Ý  Notifications: <span className="font-medium text-green-600">
                  {Notification.permission === 'granted' ? 'Allowed' :
                   Notification.permission === 'denied' ? 'Blocked' : 'Not
Requested'}
                </span>
              </p>
            </div>
          </div>

          {/* Sound Tests */}
          <div>
            <h3 className="font-semibold text-gray-900 dark:text-white mb-4">Test
Notification Sounds</h3>
            <div className="grid grid-cols-1 gap-3">
              {soundTests.map((sound) => {
                const IconComponent = sound.icon;
                return (
                  <button
                    key={sound.type}
                    onClick={() => handleTestSound(sound.type, sound.name)}
                    className="flex items-center gap-4 p-4 bg-gray-50 dark:bg-
gray-700 hover:bg-gray-100 dark:hover:bg-gray-600 rounded-lg transition-all group"
                  >
                    <div
                      className="w-12 h-12 rounded-full flex items-center justify-
center text-white"
                      style={{ backgroundColor: sound.color }}
                    >
                      <IconComponent className="text-lg" />
                    </div>
```

```
                  <div className="flex-1 text-left">
                    <div className="font-medium text-gray-900 dark:text-white
group-hover:text-blue-600 dark:group-hover:text-blue-400">
                      {sound.name}
                    </div>
                    <div className="text-sm text-gray-500 dark:text-gray-400">
                      {sound.description}
                    </div>
                  </div>
                  <FaPlay className="text-gray-400 group-hover:text-blue-500
transition-colors" />
                </button>
              );
            })}
          </div>
        </div>

        {/* Full Experience Tests */}
        <div>
          <h3 className="font-semibold text-gray-900 dark:text-white
mb-4">Complete Experience Tests</h3>
          <div className="space-y-3">

            {/* Browser Notification Test */}
            <button
              onClick={testBrowserNotification}
              className="w-full flex items-center gap-3 p-4 bg-gradient-to-r
from-purple-500 to-pink-500 text-white rounded-lg hover:from-purple-600 hover:to-
pink-600 transition-all"
            >
              <div className="w-10 h-10 bg-white bg-opacity-20 rounded-full
flex items-center justify-center">
                Ø=Ý
              </div>
              <div className="flex-1 text-left">
                <div className="font-medium">Test Browser Notification</div>
                <div className="text-sm opacity-90">Check desktop notification
permissions</div>
              </div>
            </button>

            {/* Simulate Message */}
            <button
              onClick={simulateNewMessage}
              className="w-full flex items-center gap-3 p-4 bg-gradient-to-r
from-green-500 to-blue-500 text-white rounded-lg hover:from-green-600 hover:to-
blue-600 transition-all"
            >
              <div className="w-10 h-10 bg-white bg-opacity-20 rounded-full
flex items-center justify-center">
                Ø=Ü¬
              </div>
              <div className="flex-1 text-left">
                <div className="font-medium">Simulate Incoming Message</div>
                <div className="text-sm opacity-90">Full notification
experience with sound + popup</div>
              </div>
            </button>
          </div>
```

```
          </div>

          {/* Instructions */}
          <div className="bg-yellow-50 dark:bg-yellow-900/20 border border-
yellow-200 dark:border-yellow-800 rounded-lg p-4">
            <h4 className="font-medium text-yellow-800 dark:text-yellow-200 mb-2">
Ø=Ü¡ How to Test</h4>
            <ol className="text-sm text-yellow-700 dark:text-yellow-300 space-y-1
list-decimal list-inside">
              <li>Click any sound button to test different notification tones</li>
              <li>Test browser notifications to enable desktop alerts</li>
              <li>Simulate an incoming message for the full experience</li>
              <li>Try sending real messages to hear the sounds in action</li>
            </ol>
          </div>
        </div>

        {/* Footer */}
        <div className="flex justify-end p-6 border-t dark:border-gray-700">
          <button
            onClick={onClose}
            className="px-6 py-2 bg-gray-600 text-white rounded-lg hover:bg-
gray-700 transition-colors"
          >
            Close
          </button>
        </div>
      </div>
    </div>
  );
};

export default ChatSoundTest;
```

src/components/Chat/ChatWindow.css

```
/* Chat Toggle Button */
.chat-toggle-button {
  position: fixed;
  bottom: 20px;
  right: 20px;
  width: 60px;
  height: 60px;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  cursor: pointer;
  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.15);
  transition: all 0.3s ease;
  z-index: 1000;
}

.chat-toggle-button:hover {
  transform: scale(1.1);
  box-shadow: 0 6px 25px rgba(0, 0, 0, 0.2);
}
```

```
.chat-toggle-button i {
  color: white;
  font-size: 24px;
}

.unread-badge {
  position: absolute;
  top: -5px;
  right: -5px;
  background: #ef4444;
  color: white;
  border-radius: 50%;
  width: 24px;
  height: 24px;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 12px;
  font-weight: bold;
  border: 2px solid white;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.15);
}

/* Chat Window */
.chat-window {
  position: fixed;
  bottom: 20px;
  right: 20px;
  width: 800px;
  height: 600px;
  background: white;
  border-radius: 12px;
  box-shadow: 0 10px 40px rgba(0, 0, 0, 0.15);
  display: flex;
  flex-direction: column;
  z-index: 1000;
  overflow: hidden;
}

/* Chat Header */
.chat-header {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  padding: 16px 20px;
  display: flex;
  justify-content: space-between;
  align-items: center;
  position: relative;
}

.chat-header-left h3 {
  margin: 0;
  font-size: 18px;
  display: flex;
  align-items: center;
  gap: 8px;
  position: relative;
}
```

```
/* Header Unread Count */
.header-unread-count {
  background: #ef4444;
  color: white;
  border-radius: 12px;
  padding: 2px 8px;
  font-size: 11px;
  font-weight: bold;
  margin-left: 8px;
  border: 1px solid rgba(255, 255, 255, 0.3);
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.2);
}

.connection-status {
  font-size: 12px;
  display: flex;
  align-items: center;
  gap: 6px;
  opacity: 0.9;
}

.status-indicator {
  width: 8px;
  height: 8px;
  border-radius: 50%;
}

.status-indicator.connected {
  background: #10b981;
  box-shadow: 0 0 0 2px rgba(16, 185, 129, 0.3);
}

.status-indicator.disconnected {
  background: #ef4444;
  box-shadow: 0 0 0 2px rgba(239, 68, 68, 0.3);
}

.retry-connection {
  background: rgba(255, 255, 255, 0.2);
  border: 1px solid rgba(255, 255, 255, 0.3);
  color: white;
  cursor: pointer;
  padding: 4px 6px;
  border-radius: 4px;
  font-size: 12px;
  transition: all 0.2s;
  margin-left: 4px;
}

.retry-connection:hover {
  background: rgba(255, 255, 255, 0.3);
  border-color: rgba(255, 255, 255, 0.5);
  transform: scale(1.05);
}

/* Chat header buttons - FIXED for visibility */
.chat-header-actions {
  display: flex;
```

```
  gap: 8px;
  align-items: center;
}

.btn-icon {
  background: #ffffff !important;
  border: 2px solid #007bff !important;
  border-radius: 8px !important;
  width: 40px !important;
  height: 40px !important;
  display: flex !important;
  align-items: center !important;
  justify-content: center !important;
  cursor: pointer !important;
  transition: all 0.2s ease !important;
  position: relative !important;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1) !important;
}

.btn-icon:hover {
  background: #007bff !important;
  color: white !important;
  transform: scale(1.05) !important;
  box-shadow: 0 4px 8px rgba(0, 123, 255, 0.3) !important;
}

.btn-icon:active {
  transform: scale(0.95) !important;
}

/* Hide Font Awesome icons and show emoji fallbacks */
.btn-icon i {
  display: none !important;
}

.btn-icon .btn-text {
  display: block !important;
  font-size: 18px !important;
  color: #007bff !important;
  transition: color 0.2s ease !important;
}

.btn-icon:hover .btn-text {
  color: white !important;
}

/* Specific button styles for better identification */
.btn-icon:nth-child(1) {
  border-color: #28a745 !important;
}

.btn-icon:nth-child(1):hover {
  background: #28a745 !important;
}

.btn-icon:nth-child(1) .btn-text {
  color: #28a745 !important;
}
```

```
.btn-icon:nth-child(2) {
  border-color: #ffc107 !important;
}

.btn-icon:nth-child(2):hover {
  background: #ffc107 !important;
}

.btn-icon:nth-child(2) .btn-text {
  color: #ffc107 !important;
}

.btn-icon:nth-child(3) {
  border-color: #17a2b8 !important;
}

.btn-icon:nth-child(3):hover {
  background: #17a2b8 !important;
}

.btn-icon:nth-child(3) .btn-text {
  color: #17a2b8 !important;
}

.btn-icon:nth-child(4) {
  border-color: #dc3545 !important;
}

.btn-icon:nth-child(4):hover {
  background: #dc3545 !important;
}

.btn-icon:nth-child(4) .btn-text {
  color: #dc3545 !important;
}

/* Chat Body */
.chat-body {
  flex: 1;
  display: flex;
  overflow: hidden;
}

/* User List Sidebar */
.user-list-sidebar {
  width: 300px;
  border-right: 1px solid #e5e7eb;
  display: flex;
  flex-direction: column;
  background: #f9fafb;
}

.user-list-header {
  padding: 16px;
  border-bottom: 1px solid #e5e7eb;
  display: flex;
  justify-content: space-between;
  align-items: center;
  background: white;
```

```
}

.user-list-header h4 {
  margin: 0;
  font-size: 16px;
  color: #374151;
  font-weight: 600;
}

.btn-close {
  background: none;
  border: none;
  color: #6b7280;
  cursor: pointer;
  padding: 6px;
  border-radius: 6px;
  transition: all 0.2s;
  font-size: 14px;
}

.btn-close:hover {
  background: #e5e7eb;
  color: #374151;
}

.user-search {
  padding: 16px;
  border-bottom: 1px solid #e5e7eb;
  background: white;
}

.search-input {
  width: 100%;
  padding: 10px 12px;
  border: 1px solid #d1d5db;
  border-radius: 8px;
  font-size: 14px;
  transition: border-color 0.2s;
}

.search-input:focus {
  outline: none;
  border-color: #667eea;
  box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.user-list {
  flex: 1;
  overflow-y: auto;
  padding: 8px;
}

.user-item {
  display: flex;
  align-items: center;
  padding: 12px;
  cursor: pointer;
  border-radius: 8px;
  transition: all 0.2s ease;
```

```
  margin-bottom: 4px;
  position: relative;
}

.user-item:hover {
  background: white;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  transform: translateY(-1px);
}

.user-avatar {
  position: relative;
  margin-right: 12px;
}

.user-avatar img {
  width: 40px;
  height: 40px;
  border-radius: 50%;
  object-fit: cover;
}

.avatar-placeholder {
  width: 40px;
  height: 40px;
  border-radius: 50%;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  display: flex;
  align-items: center;
  justify-content: center;
  color: white;
  font-weight: bold;
  font-size: 16px;
}

.status-dot {
  position: absolute;
  bottom: 2px;
  right: 2px;
  width: 12px;
  height: 12px;
  border-radius: 50%;
  border: 2px solid white;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
}

.user-info {
  flex: 1;
  min-width: 0;
}

.user-name {
  font-weight: 600;
  color: #374151;
  font-size: 14px;
  margin-bottom: 2px;
}

.user-role {
```

```
  font-size: 12px;
  color: #667eea;
  font-weight: 500;
  margin-bottom: 2px;
}

.user-status {
  font-size: 12px;
  color: #6b7280;
}

/* User Unread Count */
.user-unread-count {
  background: #ef4444;
  color: white;
  border-radius: 10px;
  padding: 2px 6px;
  font-size: 11px;
  font-weight: bold;
  min-width: 18px;
  text-align: center;
  margin-left: 8px;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.2);
}

/* No Users Message */
.no-users {
  text-align: center;
  padding: 40px 20px;
  color: #6b7280;
  font-style: italic;
}

/* Chat Content */
.chat-content {
  flex: 1;
  display: flex;
  flex-direction: column;
  background: white;
}

/* Active Chat Header */
.active-chat-header {
  padding: 16px 20px;
  border-bottom: 1px solid #e5e7eb;
  display: flex;
  justify-content: space-between;
  align-items: center;
  background: white;
}

.chat-user-info {
  display: flex;
  align-items: center;
  gap: 12px;
}

.chat-user-avatar {
  position: relative;
```

```
}

.chat-user-avatar img {
  width: 48px;
  height: 48px;
  border-radius: 50%;
  object-fit: cover;
}

.chat-user-avatar .avatar-placeholder {
  width: 48px;
  height: 48px;
  font-size: 18px;
}

.chat-user-details {
  flex: 1;
}

.chat-user-name {
  font-weight: 600;
  color: #374151;
  font-size: 16px;
  margin-bottom: 2px;
}

.chat-user-status {
  font-size: 13px;
  color: #6b7280;
}

.typing-indicator {
  color: #667eea;
  font-style: italic;
}

.typing-indicator i {
  margin-right: 4px;
  animation: pulse 1.5s ease-in-out infinite;
}

@keyframes pulse {
  0%, 100% { opacity: 1; }
  50% { opacity: 0.5; }
}

.btn-close-chat {
  background: none;
  border: none;
  color: #6b7280;
  cursor: pointer;
  padding: 8px;
  border-radius: 6px;
  transition: all 0.2s;
  font-size: 16px;
}

.btn-close-chat:hover {
  background: #f3f4f6;
```

```
  color: #374151;
}

/* Messages Container */
.messages-container {
  flex: 1;
  overflow-y: auto;
  padding: 20px;
  background: #f9fafb;
}

.no-messages {
  text-align: center;
  padding: 60px 20px;
  color: #6b7280;
}

.no-messages i {
  font-size: 48px;
  margin-bottom: 16px;
  opacity: 0.5;
}

.no-messages p {
  margin: 0;
  font-size: 16px;
}

.message {
  margin-bottom: 16px;
  display: flex;
  animation: fadeInUp 0.3s ease;
}

@keyframes fadeInUp {
  from {
    opacity: 0;
    transform: translateY(10px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.message.sent {
  justify-content: flex-end;
}

.message.received {
  justify-content: flex-start;
}

.message.optimistic {
  opacity: 0.7;
}

.message-content {
  max-width: 70%;
```

```
  padding: 12px 16px;
  border-radius: 18px;
  position: relative;
}

.message.sent .message-content {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  border-bottom-right-radius: 4px;
}

.message.received .message-content {
  background: white;
  color: #374151;
  border: 1px solid #e5e7eb;
  border-bottom-left-radius: 4px;
}

.message-text {
  margin-bottom: 4px;
  line-height: 1.4;
}

.message-meta {
  display: flex;
  align-items: center;
  justify-content: flex-end;
  gap: 4px;
  margin-top: 4px;
}

.message-time {
  font-size: 11px;
  opacity: 0.7;
}

.message.sent .message-time {
  color: rgba(255, 255, 255, 0.8);
}

.message.received .message-time {
  color: #6b7280;
}

.message-status {
  font-size: 11px;
  opacity: 0.8;
}

.message-status i {
  color: rgba(255, 255, 255, 0.8);
}

/* Message Input */
.message-input-form {
  padding: 16px 20px;
  border-top: 1px solid #e5e7eb;
  background: white;
}
```

```
.message-input-container {
  display: flex;
  gap: 12px;
  align-items: flex-end;
}

.message-input {
  flex: 1;
  border: 1px solid #d1d5db;
  border-radius: 24px;
  padding: 12px 16px;
  font-size: 14px;
  resize: none;
  outline: none;
  transition: border-color 0.2s;
}

.message-input:focus {
  border-color: #667eea;
  box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.send-button {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  border: none;
  color: white;
  width: 44px;
  height: 44px;
  border-radius: 50%;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: all 0.2s ease;
  font-size: 16px;
  box-shadow: 0 2px 8px rgba(102, 126, 234, 0.3);
}

.send-button:hover:not(:disabled) {
  transform: scale(1.05);
  box-shadow: 0 4px 12px rgba(102, 126, 234, 0.4);
}

.send-button:disabled {
  opacity: 0.5;
  cursor: not-allowed;
  transform: none;
}

/* Chat Welcome */
.chat-welcome {
  flex: 1;
  display: flex;
  align-items: center;
  justify-content: center;
  background: #f9fafb;
}
```

```
.welcome-content {
  text-align: center;
  max-width: 400px;
  padding: 40px;
}

.welcome-content i {
  font-size: 64px;
  color: #667eea;
  margin-bottom: 20px;
  opacity: 0.8;
}

.welcome-content h3 {
  color: #374151;
  margin-bottom: 12px;
  font-size: 24px;
  font-weight: 600;
}

.welcome-content p {
  color: #6b7280;
  margin-bottom: 30px;
  line-height: 1.5;
}

.welcome-stats {
  display: flex;
  justify-content: space-around;
  margin-top: 30px;
}

.stat {
  text-align: center;
}

.stat-number {
  display: block;
  font-size: 24px;
  font-weight: bold;
  color: #667eea;
  margin-bottom: 4px;
}

.stat-label {
  font-size: 12px;
  color: #6b7280;
  text-transform: uppercase;
  letter-spacing: 0.5px;
}

/* Responsive Design */
@media (max-width: 1024px) {
  .chat-window {
    width: 600px;
    height: 500px;
  }

  .user-list-sidebar {
```

```
    width: 250px;
  }
}

@media (max-width: 768px) {
  .chat-window {
    width: 100vw;
    height: 100vh;
    border-radius: 0;
    bottom: 0;
    right: 0;
  }

  .chat-body {
    flex-direction: column;
  }

  .user-list-sidebar {
    width: 100%;
    height: 50%;
  }
}

/* Dark mode support */
@media (prefers-color-scheme: dark) {
  .chat-window {
    background: #1f2937;
    color: #f9fafb;
  }

  .user-list-sidebar {
    background: #111827;
    border-right-color: #374151;
  }

  .user-list-header {
    background: #1f2937;
    border-bottom-color: #374151;
  }

  .user-search {
    background: #1f2937;
    border-bottom-color: #374151;
  }

  .search-input {
    background: #374151;
    border-color: #4b5563;
    color: #f9fafb;
  }

  .user-item:hover {
    background: #374151;
  }

  .messages-container {
    background: #111827;
  }
```

```
  .message.received .message-content {
    background: #374151;
    border-color: #4b5563;
    color: #f9fafb;
  }

  .chat-welcome {
    background: #111827;
  }
}
```

src/components/Chat/ChatWindow.jsx

```
import React, { useState, useEffect, useRef } from 'react';
import { useChat } from '../../context/ChatContext';
import { useAuth } from '../../context/AuthContext';
import NotificationSettings from './NotificationSettings';
import ChatSoundTest from './ChatSoundTest';
import './ChatWindow.css';

const ChatWindow = () => {
  const { user } = useAuth();
  const {
    isConnected,
    onlineUsers,
    chatRooms,
    activeChat,
    messages,
    unreadCounts,
    typingUsers,
    isChatOpen,
    sendMessage,
    sendTypingIndicator,
    fetchChatRooms,
    fetchAllUsers,
    startChat,
    closeChat,
    setIsChatOpen,
    getTotalUnreadCount
  } = useChat();

  const [messageInput, setMessageInput] = useState('');
  const [showUserList, setShowUserList] = useState(false);
  const [showNotificationSettings, setShowNotificationSettings] = useState(false);
  const [showSoundTest, setShowSoundTest] = useState(false);
  const [searchTerm, setSearchTerm] = useState('');
  const messagesEndRef = useRef(null);
  const typingTimeoutRef = useRef(null);

  // Scroll to bottom when new messages arrive
  useEffect(() => {
    scrollToBottom();
  }, [messages, activeChat]);

  // Fetch initial data
  useEffect(() => {
    if (isConnected) {
      fetchChatRooms();
```

```
    fetchAllUsers();
  }
}, [isConnected]);

const scrollToBottom = () => {
  messagesEndRef.current?.scrollIntoView({ behavior: 'smooth' });
};

const handleSendMessage = (e) => {
  e.preventDefault();
  if (messageInput.trim() && activeChat) {
    sendMessage(activeChat.otherUser._id, messageInput.trim());
    setMessageInput('');
  }
};

const handleInputChange = (e) => {
  setMessageInput(e.target.value);

  // Send typing indicator
  if (activeChat) {
    sendTypingIndicator(activeChat.otherUser._id, true);

    // Clear previous timeout
    if (typingTimeoutRef.current) {
      clearTimeout(typingTimeoutRef.current);
    }

    // Set timeout to stop typing indicator
    typingTimeoutRef.current = setTimeout(() => {
      sendTypingIndicator(activeChat.otherUser._id, false);
    }, 1000);
  }
};

const formatTime = (timestamp) => {
  return new Date(timestamp).toLocaleTimeString([], {
    hour: '2-digit',
    minute: '2-digit'
  });
};

const formatLastSeen = (lastSeen) => {
  const now = new Date();
  const lastSeenDate = new Date(lastSeen);
  const diffInMinutes = Math.floor((now - lastSeenDate) / (1000 * 60));

  if (diffInMinutes < 1) return 'Just now';
  if (diffInMinutes < 60) return `${diffInMinutes}m ago`;
  if (diffInMinutes < 1440) return `${Math.floor(diffInMinutes / 60)}h ago`;
  return lastSeenDate.toLocaleDateString();
};

const getStatusColor = (status) => {
  switch (status) {
    case 'ONLINE': return '#10b981';
    case 'AWAY': return '#f59e0b';
    case 'OFFLINE': return '#6b7280';
    default: return '#6b7280';
```

```
    }
  };

  const filteredUsers = onlineUsers.filter(u =>
    u.fullName.toLowerCase().includes(searchTerm.toLowerCase()) ||
    u.email.toLowerCase().includes(searchTerm.toLowerCase())
  );

  const currentMessages = activeChat ? messages[activeChat.chatId] || [] : [];
  const totalUnreadCount = getTotalUnreadCount();

  if (!isChatOpen) {
    return (
      <div className="chat-toggle-button" onClick={() => setIsChatOpen(true)}>
        <i className="fas fa-comments"></i>
        {totalUnreadCount > 0 && (
          <span className="unread-badge">
            {totalUnreadCount > 99 ? '99+' : totalUnreadCount}
          </span>
        )}
      </div>
    );
  }

  return (
    <div className="chat-window">
      <div className="chat-header">
        <div className="chat-header-left">
          <h3>
            <i className="fas fa-comments"></i>
            Chat
            {totalUnreadCount > 0 && (
              <span className="header-unread-count">
                {totalUnreadCount > 99 ? '99+' : totalUnreadCount}
              </span>
            )}
          </h3>
          <div className="connection-status">
            <span className={`status-indicator ${isConnected ? 'connected' :
'disconnected'}`}></span>
            {isConnected ? 'Connected' : 'Connecting...'}
            {!isConnected && (
              <button
                className="retry-connection"
                onClick={() => window.location.reload()}
                title="Retry connection"
              >
                Ø=Ý
              </button>
            )}
          </div>
        </div>
        <div className="chat-header-actions">
          <button
            className="btn-icon"
            onClick={() => setShowSoundTest(true)}
            title="Test notification sounds"
          >
            <i className="fas fa-volume-up"></i>
```

```
      <span className="btn-text">Ø=Ý </span>
    </button>
    <button
      className="btn-icon"
      onClick={() => setShowNotificationSettings(true)}
      title="Notification settings"
    >
      <i className="fas fa-bell"></i>
      <span className="btn-text">Ø=Ý </span>
    </button>
    <button
      className="btn-icon"
      onClick={() => setShowUserList(!showUserList)}
      title="Start new chat"
    >
      <i className="fas fa-plus"></i>
      <span className="btn-text">'•</span>
    </button>
    <button
      className="btn-icon"
      onClick={() => setIsChatOpen(false)}
      title="Minimize chat"
    >
      <i className="fas fa-minus"></i>
      <span className="btn-text">'–</span>
    </button>
  </div>
</div>

<div className="chat-body">
  {/* User List Sidebar */}
  {showUserList && (
    <div className="user-list-sidebar">
      <div className="user-list-header">
        <h4>Start New Chat</h4>
        <button
          className="btn-close"
          onClick={() => setShowUserList(false)}
        >
          <i className="fas fa-times"></i>
        </button>
      </div>
      <div className="user-search">
        <input
          type="text"
          placeholder="Search users..."
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          className="search-input"
        />
      </div>
      <div className="user-list">
        {filteredUsers.map(u => (
          <div
            key={u._id}
            className="user-item"
            onClick={() => {
              startChat(u);
              setShowUserList(false);
```

```
                  }}
                >
                  <div className="user-avatar">
                    {u.profilePicture ? (
                      <img src={u.profilePicture} alt={u.fullName} />
                    ) : (
                      <div className="avatar-placeholder">
                        {u.fullName.charAt(0).toUpperCase()}
                      </div>
                    )}
                    <div
                      className="status-dot"
                      style={{ backgroundColor: getStatusColor(u.status ||
u.chatStatus) }}
                    ></div>
                  </div>
                  <div className="user-info">
                    <div className="user-name">{u.fullName}</div>
                    <div className="user-role">{u.role}</div>
                    <div className="user-status">
                      {u.status === 'ONLINE' || u.chatStatus === 'ONLINE'
                        ? 'Online'
                        : u.lastSeen
                          ? `Last seen ${formatLastSeen(u.lastSeen)}`
                          : 'Offline'
                      }
                    </div>
                  </div>
                  {unreadCounts[u._id] > 0 && (
                    <div className="user-unread-count">
                      {unreadCounts[u._id] > 99 ? '99+' : unreadCounts[u._id]}
                    </div>
                  )}
                </div>
              ))}
              {filteredUsers.length === 0 && (
                <div className="no-users">
                  {searchTerm ? 'No users found' : 'Loading users...'}
                </div>
              )}
            </div>
          </div>
        )}

        {/* Chat Content */}
        <div className="chat-content">
          {activeChat ? (
            <>
              {/* Chat Header */}
              <div className="active-chat-header">
                <div className="chat-user-info">
                  <div className="chat-user-avatar">
                    {activeChat.otherUser.profilePicture ? (
                      <img src={activeChat.otherUser.profilePicture}
alt={activeChat.otherUser.fullName} />
                    ) : (
                      <div className="avatar-placeholder">
                        {activeChat.otherUser.fullName.charAt(0).toUpperCase()}
                      </div>
```

```
                    )}
                    <div
                      className="status-dot"
                      style={{ backgroundColor:
getStatusColor(activeChat.otherUser.status || activeChat.otherUser.chatStatus) }}
                    ></div>
                  </div>
                  <div className="chat-user-details">
                    <div className="chat-user-
name">{activeChat.otherUser.fullName}</div>
                    <div className="chat-user-status">
                      {typingUsers[activeChat.otherUser._id] ? (
                        <span className="typing-indicator">
                          <i className="fas fa-ellipsis-h"></i>
                          Typing...
                        </span>
                      ) : (
                        <span>
                          {activeChat.otherUser.status === 'ONLINE' ||
activeChat.otherUser.chatStatus === 'ONLINE'
                            ? 'Online'
                            : activeChat.otherUser.lastSeen
                              ? `Last seen
${formatLastSeen(activeChat.otherUser.lastSeen)}`
                              : 'Offline'
                          }
                        </span>
                      )}
                    </div>
                  </div>
                </div>
                <button
                  className="btn-close-chat"
                  onClick={closeChat}
                  title="Close chat"
                >
                  <i className="fas fa-times"></i>
                </button>
              </div>

              {/* Messages */}
              <div className="messages-container">
                {currentMessages.length === 0 ? (
                  <div className="no-messages">
                    <i className="fas fa-comments"></i>
                    <p>No messages yet. Start the conversation!</p>
                  </div>
                ) : (
                  currentMessages.map((message) => (
                    <div
                      key={message._id}
                      className={`message ${message.senderId._id === user._id ?
'sent' : 'received'} ${message.isOptimistic ? 'optimistic' : ''}`}
                    >
                      <div className="message-content">
                        <div className="message-text">{message.content}</div>
                        <div className="message-meta">
                          <span className="message-
time">{formatTime(message.timestamp)}</span>
```

```
                        {message.senderId._id === user._id && (
                          <span className="message-status">
                            {message.isOptimistic ? (
                              <i className="fas fa-clock" title="Sending..."></
i>
                            ) : message.delivered ? (
                              <i className="fas fa-check-double"
title="Delivered"></i>
                            ) : (
                              <i className="fas fa-check" title="Sent"></i>
                            )}
                          </span>
                        )}
                      </div>
                    </div>
                  </div>
                ))
              )}
              <div ref={messagesEndRef} />
            </div>

            {/* Message Input */}
            <form onSubmit={handleSendMessage} className="message-input-form">
              <div className="message-input-container">
                <input
                  type="text"
                  value={messageInput}
                  onChange={handleInputChange}
                  placeholder="Type a message..."
                  className="message-input"
                  disabled={!isConnected}
                />
                <button
                  type="submit"
                  className="send-button"
                  disabled={!messageInput.trim() || !isConnected}
                >
                  <i className="fas fa-paper-plane"></i>
                </button>
              </div>
            </form>
          </>
        ) : (
          <div className="chat-welcome">
            <div className="welcome-content">
              <i className="fas fa-comments"></i>
              <h3>Welcome to Chat</h3>
              <p>Select a user to start chatting, or click the + button to
browse available users.</p>
              <div className="welcome-stats">
                <div className="stat">
                  <span className="stat-number">{onlineUsers.length}</span>
                  <span className="stat-label">Users Online</span>
                </div>
                <div className="stat">
                  <span className="stat-number">{chatRooms.length}</span>
                  <span className="stat-label">Active Chats</span>
                </div>
                <div className="stat">
```

```
                    <span className="stat-number">{totalUnreadCount}</span>
                    <span className="stat-label">Unread Messages</span>
                  </div>
                </div>

                {/* Quick Test Button */}
                <div className="mt-6">
                  <button
                    onClick={() => setShowSoundTest(true)}
                    className="px-6 py-3 bg-gradient-to-r from-blue-500 to-
purple-500 text-white rounded-lg hover:from-blue-600 hover:to-purple-600
transition-all transform hover:scale-105 shadow-lg"
                  >
                    Ø=Ý  Test WhatsApp-Style Notifications
                  </button>
                </div>
              </div>
            </div>
          )}
        </div>
      </div>

      {/* Notification Settings Modal */}
      <NotificationSettings
        isOpen={showNotificationSettings}
        onClose={() => setShowNotificationSettings(false)}
      />

      {/* Sound Test Modal */}
      <ChatSoundTest
        isOpen={showSoundTest}
        onClose={() => setShowSoundTest(false)}
      />
    </div>
  );
};

export default ChatWindow;
```

src/components/Chat/GuestChat.css

```
/* Guest Chat Button */
.guest-chat-button {
  position: fixed;
  bottom: 20px;
  right: 20px;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  padding: 15px 20px;
  border-radius: 50px;
  cursor: pointer;
  box-shadow: 0 4px 20px rgba(102, 126, 234, 0.4);
  display: flex;
  align-items: center;
  gap: 10px;
  font-weight: 600;
  transition: all 0.3s ease;
  z-index: 1000;
```

```
  max-width: 200px;
  animation: slideInUp 0.5s ease-out;
}

.guest-chat-button:hover {
  transform: translateY(-2px);
  box-shadow: 0 6px 25px rgba(102, 126, 234, 0.5);
}

.guest-chat-button .chat-icon {
  font-size: 20px;
}

.guest-chat-button .chat-text {
  font-size: 14px;
  white-space: nowrap;
}

.chat-pulse {
  position: absolute;
  top: -5px;
  right: -5px;
  width: 12px;
  height: 12px;
  background: #10b981;
  border-radius: 50%;
  animation: pulse 2s infinite;
}

@keyframes pulse {
  0% {
    transform: scale(0.95);
    box-shadow: 0 0 0 0 rgba(16, 185, 129, 0.7);
  }
  70% {
    transform: scale(1);
    box-shadow: 0 0 0 10px rgba(16, 185, 129, 0);
  }
  100% {
    transform: scale(0.95);
    box-shadow: 0 0 0 0 rgba(16, 185, 129, 0);
  }
}

@keyframes slideInUp {
  from {
    transform: translateY(100px);
    opacity: 0;
  }
  to {
    transform: translateY(0);
    opacity: 1;
  }
}

/* Guest Chat Window */
.guest-chat-window {
  position: fixed;
  bottom: 20px;
```

```
  right: 20px;
  width: 380px;
  height: 500px;
  background: white;
  border-radius: 12px;
  box-shadow: 0 10px 40px rgba(0, 0, 0, 0.15);
  display: flex;
  flex-direction: column;
  z-index: 1000;
  animation: slideInUp 0.3s ease-out;
  overflow: hidden;
}

/* Header */
.guest-chat-header {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  padding: 15px 20px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.header-content {
  display: flex;
  align-items: center;
  gap: 12px;
}

.header-icon {
  font-size: 20px;
}

.guest-chat-header h3 {
  margin: 0;
  font-size: 16px;
  font-weight: 600;
}

.status {
  font-size: 12px;
  opacity: 0.9;
}

.status.online {
  color: #10b981;
}

.status.offline {
  color: #f59e0b;
}

.close-button {
  background: none;
  border: none;
  color: white;
  font-size: 16px;
  cursor: pointer;
  padding: 5px;
```

```
  border-radius: 4px;
  transition: background-color 0.2s;
}

.close-button:hover {
  background-color: rgba(255, 255, 255, 0.1);
}

/* Body */
.guest-chat-body {
  flex: 1;
  overflow-y: auto;
  padding: 20px;
  background: #f8fafc;
}

/* Guest Info Form */
.guest-info-form {
  text-align: center;
}

.guest-info-form h4 {
  color: #1f2937;
  margin-bottom: 8px;
  font-size: 18px;
}

.guest-info-form p {
  color: #6b7280;
  margin-bottom: 20px;
  font-size: 14px;
}

.guest-info-form form {
  display: flex;
  flex-direction: column;
  gap: 12px;
}

.guest-info-form input {
  padding: 12px;
  border: 1px solid #d1d5db;
  border-radius: 8px;
  font-size: 14px;
  transition: border-color 0.2s;
}

.guest-info-form input:focus {
  outline: none;
  border-color: #667eea;
  box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.start-chat-btn {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  border: none;
  padding: 12px 24px;
  border-radius: 8px;
```

```
  font-weight: 600;
  cursor: pointer;
  transition: transform 0.2s;
}

.start-chat-btn:hover {
  transform: translateY(-1px);
}

/* Support Team Selection */
.support-team-selection h4 {
  color: #1f2937;
  margin-bottom: 15px;
  font-size: 16px;
}

.support-list {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.support-member {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 12px;
  background: white;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.2s;
  border: 1px solid #e5e7eb;
}

.support-member:hover {
  background: #f3f4f6;
  border-color: #667eea;
}

.member-info {
  flex: 1;
}

.member-name {
  font-weight: 600;
  color: #1f2937;
  font-size: 14px;
}

.member-role {
  font-size: 12px;
  color: #6b7280;
}

.online-indicator {
  width: 8px;
  height: 8px;
  background: #10b981;
  border-radius: 50%;
```

```
}

.no-support {
  text-align: center;
  padding: 20px;
}

.no-support p {
  color: #6b7280;
  margin-bottom: 10px;
  font-size: 14px;
}

.continue-anyway-btn {
  background: #6b7280;
  color: white;
  border: none;
  padding: 10px 20px;
  border-radius: 6px;
  cursor: pointer;
  font-size: 14px;
  transition: background-color 0.2s;
}

.continue-anyway-btn:hover {
  background: #4b5563;
}

/* Chat Messages */
.chat-messages {
  display: flex;
  flex-direction: column;
  gap: 12px;
  max-height: 300px;
  overflow-y: auto;
  padding-right: 5px;
}

.message {
  display: flex;
  flex-direction: column;
  max-width: 80%;
}

.guest-message {
  align-self: flex-end;
}

.support-message {
  align-self: flex-start;
}

.system-message {
  align-self: center;
  max-width: 90%;
}

.message-content {
  padding: 10px 14px;
```

```
  border-radius: 12px;
  font-size: 14px;
  line-height: 1.4;
  word-wrap: break-word;
}

.guest-message .message-content {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  border-bottom-right-radius: 4px;
}

.support-message .message-content {
  background: white;
  color: #1f2937;
  border: 1px solid #e5e7eb;
  border-bottom-left-radius: 4px;
}

.system-message .message-content {
  background: #f3f4f6;
  color: #6b7280;
  text-align: center;
  font-style: italic;
  border-radius: 20px;
}

.message-time {
  font-size: 11px;
  color: #9ca3af;
  margin-top: 4px;
  text-align: right;
}

.support-message .message-time {
  text-align: left;
}

.system-message .message-time {
  text-align: center;
}

/* Footer */
.guest-chat-footer {
  background: white;
  border-top: 1px solid #e5e7eb;
  padding: 15px;
}

.message-input-container {
  display: flex;
  gap: 10px;
  align-items: center;
}

.message-input {
  flex: 1;
  padding: 10px 12px;
  border: 1px solid #d1d5db;
```

```
  border-radius: 20px;
  font-size: 14px;
  outline: none;
  transition: border-color 0.2s;
}

.message-input:focus {
  border-color: #667eea;
}

.send-button {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  border: none;
  width: 40px;
  height: 40px;
  border-radius: 50%;
  cursor: pointer;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: transform 0.2s;
}

.send-button:hover:not(:disabled) {
  transform: scale(1.05);
}

.send-button:disabled {
  opacity: 0.5;
  cursor: not-allowed;
}

.chat-footer-info {
  margin-top: 8px;
  text-align: center;
}

.chat-footer-info small {
  color: #6b7280;
  font-size: 12px;
}

/* Responsive Design */
@media (max-width: 480px) {
  .guest-chat-window {
    width: calc(100vw - 20px);
    height: calc(100vh - 40px);
    bottom: 10px;
    right: 10px;
  }

  .guest-chat-button {
    bottom: 15px;
    right: 15px;
    padding: 12px 16px;
  }

  .guest-chat-button .chat-text {
```

```
    display: none;
  }
}

/* Scrollbar Styling */
.chat-messages::-webkit-scrollbar {
  width: 4px;
}

.chat-messages::-webkit-scrollbar-track {
  background: #f1f1f1;
  border-radius: 2px;
}

.chat-messages::-webkit-scrollbar-thumb {
  background: #c1c1c1;
  border-radius: 2px;
}

.chat-messages::-webkit-scrollbar-thumb:hover {
  background: #a1a1a1;
}
```

src/components/Chat/GuestChat.jsx

```
import React, { useState, useEffect, useRef } from 'react';
import { io } from 'socket.io-client';
import { FaPaperPlane, FaTimes, FaComments } from 'react-icons/fa';
import './GuestChat.css';

const GuestChat = () => {
  const [isOpen, setIsOpen] = useState(false);
  const [messages, setMessages] = useState([]);
  const [newMessage, setNewMessage] = useState('');
  const [guestInfo, setGuestInfo] = useState({
    name: '',
    email: '',
    phone: '',
    isInfoProvided: false
  });
  const [socket, setSocket] = useState(null);
  const [isConnected, setIsConnected] = useState(false);
  const [supportTeam, setSupportTeam] = useState([]);
  const [selectedSupport, setSelectedSupport] = useState(null);
  const messagesEndRef = useRef(null);
  const guestId = useRef(`guest_${Date.now()}
_${Math.random().toString(36).substr(2, 9)}`);

  // Auto-scroll to bottom
  const scrollToBottom = () => {
    messagesEndRef.current?.scrollIntoView({ behavior: "smooth" });
  };

  useEffect(() => {
    scrollToBottom();
  }, [messages]);

  // Initialize Socket.IO connection for guest
```

```
  useEffect(() => {
    if (isOpen && !socket) {
      const serverUrl = import.meta.env?.VITE_API_URL ||
process.env.REACT_APP_API_URL || 'http://localhost:8080';

      const newSocket = io(serverUrl, {
        transports: ['websocket', 'polling'],
        query: {
          isGuest: true,
          guestId: guestId.current
        }
      });

      newSocket.on('connect', () => {
        console.log('  Guest connected to chat server');
        setIsConnected(true);

        // Join guest room
        newSocket.emit('join-guest-room', guestId.current);

        // Request available support team
        newSocket.emit('get-support-team');
      });

      newSocket.on('support-team-list', (team) => {
        setSupportTeam(team.filter(member => member.chatStatus === 'ONLINE'));
      });

      newSocket.on('guest-message-received', (message) => {
        setMessages(prev => [...prev, {
          ...message,
          timestamp: new Date(message.timestamp)
        }]);
      });

      newSocket.on('connect_error', (error) => {
        console.error(''L Guest connection error:', error);
        setIsConnected(false);
      });

      setSocket(newSocket);

      return () => {
        newSocket.close();
        setSocket(null);
        setIsConnected(false);
      };
    }
  }, [isOpen]);

  const handleInfoSubmit = (e) => {
    e.preventDefault();
    if (guestInfo.name && guestInfo.email) {
      setGuestInfo(prev => ({ ...prev, isInfoProvided: true }));

      // Send welcome message
      const welcomeMessage = {
        id: Date.now(),
        content: `Hi ${guestInfo.name}! Thanks for reaching out. How can we help
```

```
you today?`,
        sender: 'system',
        timestamp: new Date(),
        isSystem: true
      };
      setMessages([welcomeMessage]);
    }
  };

  const sendMessage = () => {
    if (newMessage.trim() && socket && selectedSupport) {
      const messageData = {
        guestId: guestId.current,
        guestInfo,
        recipientId: selectedSupport._id,
        content: newMessage.trim(),
        timestamp: new Date()
      };

      // Add message to local state immediately
      const localMessage = {
        id: Date.now(),
        content: newMessage.trim(),
        sender: 'guest',
        timestamp: new Date()
      };
      setMessages(prev => [...prev, localMessage]);

      // Send via socket
      socket.emit('guest-message', messageData);
      setNewMessage('');
    }
  };

  const handleKeyPress = (e) => {
    if (e.key === 'Enter' && !e.shiftKey) {
      e.preventDefault();
      sendMessage();
    }
  };

  const selectSupport = (supportMember) => {
    setSelectedSupport(supportMember);
    const selectionMessage = {
      id: Date.now(),
      content: `You're now connected with ${supportMember.fullName}
(${supportMember.role}). They'll respond shortly.`,
      sender: 'system',
      timestamp: new Date(),
      isSystem: true
    };
    setMessages(prev => [...prev, selectionMessage]);
  };

  if (!isOpen) {
    return (
      <div className="guest-chat-button" onClick={() => setIsOpen(true)}>
        <FaComments className="chat-icon" />
        <span className="chat-text">Need Help? Chat Now!</span>
```

```
        <div className="chat-pulse"></div>
      </div>
    );
  }

  return (
    <div className="guest-chat-window">
      <div className="guest-chat-header">
        <div className="header-content">
          <FaComments className="header-icon" />
          <div>
            <h3>Live Support Chat</h3>
            <span className={`status ${isConnected ? 'online' : 'offline'}`}>
              {isConnected ? 'Connected' : 'Connecting...'}
            </span>
          </div>
        </div>
        <button className="close-button" onClick={() => setIsOpen(false)}>
          <FaTimes />
        </button>
      </div>

      <div className="guest-chat-body">
        {!guestInfo.isInfoProvided ? (
          <div className="guest-info-form">
            <h4>Welcome! Let's get started</h4>
            <p>Please provide your details to begin chatting with our support
team:</p>
            <form onSubmit={handleInfoSubmit}>
              <input
                type="text"
                placeholder="Your Name *"
                value={guestInfo.name}
                onChange={(e) => setGuestInfo(prev => ({ ...prev, name:
e.target.value }))}
                required
              />
              <input
                type="email"
                placeholder="Your Email *"
                value={guestInfo.email}
                onChange={(e) => setGuestInfo(prev => ({ ...prev, email:
e.target.value }))}
                required
              />
              <input
                type="tel"
                placeholder="Your Phone (Optional)"
                value={guestInfo.phone}
                onChange={(e) => setGuestInfo(prev => ({ ...prev, phone:
e.target.value }))}
              />
              <button type="submit" className="start-chat-btn">
                Start Chat
              </button>
            </form>
          </div>
        ) : !selectedSupport ? (
          <div className="support-team-selection">
```

```
            <h4>Choose a support team member:</h4>
            {supportTeam.length > 0 ? (
              <div className="support-list">
                {supportTeam.map(member => (
                  <div
                    key={member._id}
                    className="support-member"
                    onClick={() => selectSupport(member)}
                  >
                    <div className="member-info">
                      <div className="member-name">{member.fullName}</div>
                      <div className="member-role">{member.role}</div>
                    </div>
                    <div className="online-indicator"></div>
                  </div>
                ))}
              </div>
            ) : (
              <div className="no-support">
                <p>No support team members are currently online.</p>
                <p>Please leave your message and we'll get back to you soon!</p>
                <button
                  className="continue-anyway-btn"
                  onClick={() => setSelectedSupport({ _id: 'offline', fullName:
'Support Team', role: 'Support' })}
                >
                  Continue Anyway
                </button>
              </div>
            )}
          </div>
        ) : (
          <div className="chat-messages">
            {messages.map((message) => (
              <div
                key={message.id}
                className={`message ${message.sender === 'guest' ? 'guest-
message' : 'support-message'} ${message.isSystem ? 'system-message' : ''}`}
              >
                <div className="message-content">
                  {message.content}
                </div>
                <div className="message-time">
                  {message.timestamp.toLocaleTimeString([], { hour: '2-digit',
minute: '2-digit' })}
                </div>
              </div>
            ))}
            <div ref={messagesEndRef} />
          </div>
        )}
      </div>

      {guestInfo.isInfoProvided && selectedSupport && (
        <div className="guest-chat-footer">
          <div className="message-input-container">
            <input
              type="text"
              value={newMessage}
```

```
              onChange={(e) => setNewMessage(e.target.value)}
              onKeyPress={handleKeyPress}
              placeholder="Type your message..."
              className="message-input"
            />
            <button
              onClick={sendMessage}
              disabled={!newMessage.trim() || !isConnected}
              className="send-button"
            >
              <FaPaperPlane />
            </button>
          </div>
          <div className="chat-footer-info">
            <small>Chatting with {selectedSupport.fullName} • {isConnected ?
'Online' : 'Offline'}</small>
          </div>
        </div>
      )}
    </div>
  );
};

export default GuestChat;
```

src/components/Chat/NotificationSettings.jsx

```
import React, { useState, useEffect } from 'react';
import { useChat } from '../../context/ChatContext';
import notificationService from '../../services/notificationService';
import {
  FaVolumeDown,
  FaVolumeUp,
  FaVolumeOff,
  FaPlay,
  FaBell,
  FaBellSlash,
  FaCog,
  FaCheck,
  FaTimes,
  FaDesktop,
  FaComment,
  FaExclamationTriangle,
  FaMusic
} from 'react-icons/fa';

const NotificationSettings = ({ isOpen, onClose }) => {
  const {
    notificationPreferences,
    saveNotificationPreferences,
    testNotificationSound
  } = useChat();

  const [localPreferences, setLocalPreferences] = useState({
    enableSounds: true,
    messageSound: 'message',
    volume: 0.3,
    enableBrowserNotifications: true,
```

```
    enableToastNotifications: true,
    enableStatusSounds: false
  });

  // Load current preferences when component opens
  useEffect(() => {
    if (isOpen) {
      setLocalPreferences(prev => ({
        ...prev,
        ...notificationPreferences
      }));
    }
  }, [isOpen, notificationPreferences]);

  // Handle preference changes
  const handlePreferenceChange = (key, value) => {
    setLocalPreferences(prev => ({
      ...prev,
      [key]: value
    }));
  };

  // Save preferences
  const handleSave = () => {
    saveNotificationPreferences(localPreferences);
    onClose();
  };

  // Test notification sound
  const handleTestSound = (soundType) => {
    testNotificationSound(soundType);
  };

  // Volume level descriptions
  const getVolumeDescription = (volume) => {
    if (volume === 0) return 'Muted';
    if (volume <= 0.2) return 'Very Low';
    if (volume <= 0.4) return 'Low';
    if (volume <= 0.6) return 'Medium';
    if (volume <= 0.8) return 'High';
    return 'Very High';
  };

  // Sound type options
  const soundOptions = [
    { value: 'message', label: 'Message', icon: FaComment, description: 'Standard
message tone' },
    { value: 'group', label: 'Group Chat', icon: FaComment, description: 'Triple
tone for group messages' },
    { value: 'urgent', label: 'Urgent', icon: FaExclamationTriangle, description:
'Rapid beeps for urgent messages' },
    { value: 'soft', label: 'Soft', icon: FaMusic, description: 'Gentle tone for
quiet environments' }
  ];

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
```

```
justify-center z-50">
      <div className="bg-white dark:bg-gray-800 rounded-lg shadow-xl w-full max-w-
md mx-4 max-h-[90vh] overflow-y-auto">
        {/* Header */}
        <div className="flex items-center justify-between p-4 border-b
dark:border-gray-700">
          <h2 className="text-lg font-semibold text-gray-900 dark:text-white flex
items-center gap-2">
            <FaBell className="text-blue-500" />
            Notification Settings
          </h2>
          <button
            onClick={onClose}
            className="text-gray-500 hover:text-gray-700 dark:text-gray-400
dark:hover:text-gray-200"
          >
            <FaTimes />
          </button>
        </div>

        {/* Settings Content */}
        <div className="p-4 space-y-6">

          {/* Sound Settings */}
          <div className="space-y-4">
            <h3 className="text-md font-medium text-gray-900 dark:text-white flex
items-center gap-2">
              <FaVolumeUp className="text-green-500" />
              Sound Settings
            </h3>

            {/* Enable Sounds Toggle */}
            <div className="flex items-center justify-between">
              <div className="flex items-center gap-2">
                <FaVolume className="text-gray-500" />
                <span className="text-sm text-gray-700 dark:text-gray-300">Enable
Sounds</span>
              </div>
              <label className="relative inline-flex items-center cursor-pointer">
                <input
                  type="checkbox"
                  checked={localPreferences.enableSounds}
                  onChange={(e) => handlePreferenceChange('enableSounds',
e.target.checked)}
                  className="sr-only peer"
                />
                <div className="w-11 h-6 bg-gray-200 peer-focus:outline-none peer-
focus:ring-4 peer-focus:ring-blue-300 dark:peer-focus:ring-blue-800 rounded-full
peer dark:bg-gray-700 peer-checked:after:translate-x-full peer-
checked:after:border-white after:content-[''] after:absolute after:top-[2px]
after:left-[2px] after:bg-white after:border-gray-300 after:border after:rounded-
full after:h-5 after:w-5 after:transition-all dark:border-gray-600 peer-
checked:bg-blue-600"></div>
              </label>
            </div>

            {/* Volume Control */}
            {localPreferences.enableSounds && (
              <div className="space-y-2">
```

```
                <div className="flex items-center justify-between">
                  <span className="text-sm text-gray-700 dark:text-
gray-300">Volume</span>
                  <span className="text-sm text-blue-600 dark:text-blue-400">
                    {getVolumeDescription(localPreferences.volume)}
                  </span>
                </div>
                <div className="flex items-center gap-2">
                  <FaVolumeOff className="text-gray-400" />
                  <input
                    type="range"
                    min="0"
                    max="1"
                    step="0.1"
                    value={localPreferences.volume}
                    onChange={(e) => handlePreferenceChange('volume',
parseFloat(e.target.value))}
                    className="flex-1 h-2 bg-gray-200 rounded-lg appearance-none
cursor-pointer dark:bg-gray-700"
                  />
                  <FaVolumeUp className="text-gray-400" />
                </div>
              </div>
            )}

            {/* Sound Type Selection */}
            {localPreferences.enableSounds && (
              <div className="space-y-2">
                <span className="text-sm text-gray-700 dark:text-
gray-300">Notification Sound</span>
                <div className="grid grid-cols-2 gap-2">
                  {soundOptions.map((option) => (
                    <div key={option.value} className="space-y-1">
                      <label
                        className={`flex items-center gap-2 p-2 rounded-lg border
cursor-pointer transition-colors ${
                          localPreferences.messageSound === option.value
                            ? 'border-blue-500 bg-blue-50 dark:bg-blue-900/20'
                            : 'border-gray-200 dark:border-gray-600 hover:bg-
gray-50 dark:hover:bg-gray-700'
                        }`}
                      >
                        <input
                          type="radio"
                          name="messageSound"
                          value={option.value}
                          checked={localPreferences.messageSound === option.value}
                          onChange={(e) => handlePreferenceChange('messageSound',
e.target.value)}
                          className="sr-only"
                        />
                        <option.icon className={`text-sm ${
                          localPreferences.messageSound === option.value ? 'text-
blue-600' : 'text-gray-400'
                        }`} />
                        <span className={`text-sm ${
                          localPreferences.messageSound === option.value
                            ? 'text-blue-600 dark:text-blue-400 font-medium'
                            : 'text-gray-700 dark:text-gray-300'
```

```
                      }`}>
                        {option.label}
                      </span>
                    </label>
                    <button
                      onClick={() => handleTestSound(option.value)}
                      className="w-full flex items-center justify-center gap-1
py-1 px-2 text-xs text-gray-500 hover:text-blue-600 dark:text-gray-400
dark:hover:text-blue-400 transition-colors"
                    >
                      <FaPlay className="text-xs" />
                      Test
                    </button>
                  </div>
                ))}
              </div>
            </div>
          )}

          {/* Status Sound Toggle */}
          <div className="flex items-center justify-between">
            <div className="flex items-center gap-2">
              <FaMusic className="text-gray-500" />
              <span className="text-sm text-gray-700 dark:text-gray-300">Status
Change Sounds</span>
            </div>
            <label className="relative inline-flex items-center cursor-pointer">
              <input
                type="checkbox"
                checked={localPreferences.enableStatusSounds}
                onChange={(e) => handlePreferenceChange('enableStatusSounds',
e.target.checked)}
                className="sr-only peer"
              />
              <div className="w-11 h-6 bg-gray-200 peer-focus:outline-none peer-
focus:ring-4 peer-focus:ring-blue-300 dark:peer-focus:ring-blue-800 rounded-full
peer dark:bg-gray-700 peer-checked:after:translate-x-full peer-
checked:after:border-white after:content-[''] after:absolute after:top-[2px]
after:left-[2px] after:bg-white after:border-gray-300 after:border after:rounded-
full after:h-5 after:w-5 after:transition-all dark:border-gray-600 peer-
checked:bg-blue-600"></div>
            </label>
          </div>
        </div>

        {/* Notification Types */}
        <div className="space-y-4">
          <h3 className="text-md font-medium text-gray-900 dark:text-white flex
items-center gap-2">
            <FaBell className="text-blue-500" />
            Notification Types
          </h3>

          {/* Browser Notifications */}
          <div className="flex items-center justify-between">
            <div className="flex items-center gap-2">
              <FaDesktop className="text-gray-500" />
              <div>
                <span className="text-sm text-gray-700 dark:text-
```

```
gray-300">Browser Notifications</span>
                  <p className="text-xs text-gray-500 dark:text-gray-400">Show
notifications outside the browser</p>
                </div>
              </div>
              <label className="relative inline-flex items-center cursor-pointer">
                <input
                  type="checkbox"
                  checked={localPreferences.enableBrowserNotifications}
                  onChange={(e) =>
handlePreferenceChange('enableBrowserNotifications', e.target.checked)}
                  className="sr-only peer"
                />
                <div className="w-11 h-6 bg-gray-200 peer-focus:outline-none peer-
focus:ring-4 peer-focus:ring-blue-300 dark:peer-focus:ring-blue-800 rounded-full
peer dark:bg-gray-700 peer-checked:after:translate-x-full peer-
checked:after:border-white after:content-[''] after:absolute after:top-[2px]
after:left-[2px] after:bg-white after:border-gray-300 after:border after:rounded-
full after:h-5 after:w-5 after:transition-all dark:border-gray-600 peer-
checked:bg-blue-600"></div>
              </label>
            </div>

            {/* Toast Notifications */}
            <div className="flex items-center justify-between">
              <div className="flex items-center gap-2">
                <FaComment className="text-gray-500" />
                <div>
                  <span className="text-sm text-gray-700 dark:text-gray-300">In-
App Notifications</span>
                  <p className="text-xs text-gray-500 dark:text-gray-400">Show
toast notifications within the app</p>
                </div>
              </div>
              <label className="relative inline-flex items-center cursor-pointer">
                <input
                  type="checkbox"
                  checked={localPreferences.enableToastNotifications}
                  onChange={(e) =>
handlePreferenceChange('enableToastNotifications', e.target.checked)}
                  className="sr-only peer"
                />
                <div className="w-11 h-6 bg-gray-200 peer-focus:outline-none peer-
focus:ring-4 peer-focus:ring-blue-300 dark:peer-focus:ring-blue-800 rounded-full
peer dark:bg-gray-700 peer-checked:after:translate-x-full peer-
checked:after:border-white after:content-[''] after:absolute after:top-[2px]
after:left-[2px] after:bg-white after:border-gray-300 after:border after:rounded-
full after:h-5 after:w-5 after:transition-all dark:border-gray-600 peer-
checked:bg-blue-600"></div>
              </label>
            </div>
          </div>

          {/* Test Section */}
          <div className="space-y-4">
            <h3 className="text-md font-medium text-gray-900 dark:text-white flex
items-center gap-2">
              <FaPlay className="text-purple-500" />
              Test Notifications
```

```
            </h3>

            <div className="grid grid-cols-2 gap-2">
              <button
                onClick={() => handleTestSound('message')}
                className="flex items-center justify-center gap-2 py-2 px-3 bg-
blue-500 text-white rounded-lg hover:bg-blue-600 transition-colors"
              >
                <FaPlay className="text-sm" />
                <span className="text-sm">Message</span>
              </button>
              <button
                onClick={() => handleTestSound('urgent')}
                className="flex items-center justify-center gap-2 py-2 px-3 bg-
red-500 text-white rounded-lg hover:bg-red-600 transition-colors"
              >
                <FaPlay className="text-sm" />
                <span className="text-sm">Urgent</span>
              </button>
            </div>
          </div>
        </div>

        {/* Footer */}
        <div className="flex items-center justify-between p-4 border-t
dark:border-gray-700">
          <button
            onClick={onClose}
            className="px-4 py-2 text-gray-600 dark:text-gray-400 hover:text-
gray-800 dark:hover:text-gray-200 transition-colors"
          >
            Cancel
          </button>
          <button
            onClick={handleSave}
            className="flex items-center gap-2 px-4 py-2 bg-blue-600 text-white
rounded-lg hover:bg-blue-700 transition-colors"
          >
            <FaCheck />
            Save Changes
          </button>
        </div>
      </div>
    </div>
  );
};

export default NotificationSettings;
```

src/components/CurrencySelector.jsx

```
import React, { useState, useEffect } from 'react';
import { getCurrencySettings, setCurrencySettings } from '../utils/helpers';
import axios from 'axios';
import { currencyAPI } from '../services/api';

// Expanded currency list with more options
const currencyInfo = {
```

```
  USD: { locale: 'en-US', label: 'USD ($)', symbol: '$' },
  INR: { locale: 'en-IN', label: 'INR ( ¹)', symbol: ' ¹' },
  EUR: { locale: 'en-EU', label: 'EUR (€)', symbol: '€' },
  GBP: { locale: 'en-GB', label: 'GBP (£)', symbol: '£' },
  JPY: { locale: 'ja-JP', label: 'JPY (¥)', symbol: '¥' },
  CAD: { locale: 'en-CA', label: 'CAD ($)', symbol: 'C$' },
  AUD: { locale: 'en-AU', label: 'AUD ($)', symbol: 'A$' },
  CNY: { locale: 'zh-CN', label: 'CNY (¥)', symbol: '¥' },
  SGD: { locale: 'en-SG', label: 'SGD ($)', symbol: 'S$' },
  CHF: { locale: 'de-CH', label: 'CHF (Fr)', symbol: 'Fr' },
  AED: { locale: 'ar-AE', label: 'AED (bòæ%)', symbol: 'bòæ%' }
};

// Fallback rates if API fails
const initialRates = {
  USD: 1,
  INR: 84.62,
  EUR: 0.92,
  GBP: 0.79,
  JPY: 151.15,
  CAD: 1.37,
  AUD: 1.52,
  CNY: 7.24,
  SGD: 1.35,
  CHF: 0.90,
  AED: 3.67
};

/**
 * Currency selector component that allows users to change the currency format
globally
 * @param {Object} props - Component props
 * @param {boolean} props.darkMode - Whether to use dark mode styling (white text)
 */
const CurrencySelector = ({ darkMode = true }) => {
  const [currentCurrency, setCurrentCurrency] = useState(() =>
getCurrencySettings().currency);
  const [exchangeRates, setExchangeRates] = useState(initialRates);
  const [availableCurrencies, setAvailableCurrencies] = useState([]);
  const [fetchError, setFetchError] = useState(null);
  const [lastUpdated, setLastUpdated] = useState(null);

  // Text colors based on mode
  const textColor = darkMode ? 'text-white' : 'text-gray-700 dark:text-gray-300
dark:text-gray-400';
  const subTextColor = darkMode ? 'text-blue-200' : 'text-gray-500 dark:text-
gray-400';
  const errorTextColor = darkMode ? 'text-yellow-200' : 'text-yellow-600';
  const bgColor = darkMode ? 'bg-blue-700' : 'bg-white dark:bg-slate-900';
  const borderColor = darkMode ? 'border-blue-600' : 'border-gray-300 dark:border-
slate-600';

  // Fetch latest exchange rates from our backend API
  useEffect(() => {
    fetchExchangeRates();

    // Refresh rates periodically if user stays on the page
    const intervalId = setInterval(fetchExchangeRates, 60 * 60 * 1000); // Every
hour
```

```
    return () => clearInterval(intervalId);
  }, []);

  const fetchExchangeRates = async () => {
    try {
      // Use our server-side endpoint which implements caching and fallbacks
      const response = await currencyAPI.getRates();

      if (response.data && response.data.rates) {
        const rates = response.data.rates;

        // Update with the latest rates
        setExchangeRates(rates);

        // Generate available currencies based on the rates we received
        const currencies = Object.keys(rates)
          .filter(code => currencyInfo[code]) // Only use currencies we have info
for
          .map(code => ({
            code,
            ...currencyInfo[code],
            rate: rates[code]
          }))
          .sort((a, b) => {
            // Sort with USD first, then alphabetically
            if (a.code === 'USD') return -1;
            if (b.code === 'USD') return 1;
            return a.code.localeCompare(b.code);
          });

        setAvailableCurrencies(currencies);
        setLastUpdated(new Date(response.data.date || new Date()));
        setFetchError(response.data.source === 'fallback' ? 'Using backup
rates' : null);

        // Update global currency settings
        setCurrencySettings({
          ...getCurrencySettings(),
          exchangeRates: rates,
          lastUpdated: response.data.date || new Date().toISOString()
        });



        // Store in localStorage for additional persistence
        localStorage.setItem('exchange_rates', JSON.stringify(rates));
        localStorage.setItem('rates_last_updated', response.data.date || new
Date().toISOString());

        // Show notification if rates are fallback rates
        if (response.data.source === 'fallback') {
          // Using fallback exchange rates - all APIs failed
        }
      } else {
        throw new Error('Invalid response format from currency API');
      }
    } catch (error) {
      setFetchError('Using last known rates');
```

```
    // Try to load from localStorage as a fallback
    const storedRates = localStorage.getItem('exchange_rates');
    const storedLastUpdated = localStorage.getItem('rates_last_updated');

    if (storedRates) {
      try {
        const parsedRates = JSON.parse(storedRates);
        setExchangeRates(parsedRates);

        // Generate available currencies from fallback
        const currencies = Object.keys(parsedRates)
          .filter(code => currencyInfo[code])
          .map(code => ({
            code,
            ...currencyInfo[code],
            rate: parsedRates[code]
          }))
          .sort((a, b) => {
            if (a.code === 'USD') return -1;
            if (b.code === 'USD') return 1;
            return a.code.localeCompare(b.code);
          });

        setAvailableCurrencies(currencies);
        setLastUpdated(new Date(storedLastUpdated));

        setCurrencySettings({
          ...getCurrencySettings(),
          exchangeRates: parsedRates,
          lastUpdated: storedLastUpdated
        });
      } catch (e) {
        // If parsing fails, use initialRates
        setExchangeRates(initialRates);

        // Create currencies from initialRates
        const currencies = Object.keys(initialRates)
          .filter(code => currencyInfo[code])
          .map(code => ({
            code,
            ...currencyInfo[code],
            rate: initialRates[code]
          }))
          .sort((a, b) => {
            if (a.code === 'USD') return -1;
            if (b.code === 'USD') return 1;
            return a.code.localeCompare(b.code);
          });

        setAvailableCurrencies(currencies);
      }
    }
  }
};

useEffect(() => {
  if (availableCurrencies.length === 0) return;
```

```
    // Initialize with stored settings if available
    const storedCurrency = localStorage.getItem('preferred_currency');
    if (storedCurrency) {
      const currency = availableCurrencies.find(c => c.code === storedCurrency);
      if (currency) {
        setCurrencySettings({
          currency: currency.code,
          locale: currency.locale,
          rate: exchangeRates[currency.code] || currency.rate,
          exchangeRates
        });
        setCurrentCurrency(currency.code);
      }
    } else {
      // Set default currency settings with exchange rates
      const defaultCurrency = availableCurrencies.find(c => c.code === 'USD') ||
availableCurrencies[0];
      setCurrencySettings({
        currency: defaultCurrency.code,
        locale: defaultCurrency.locale,
        rate: exchangeRates[defaultCurrency.code] || defaultCurrency.rate,
        exchangeRates
      });
    }
  }, [availableCurrencies, exchangeRates]);

  const handleCurrencyChange = (e) => {
    const selectedCode = e.target.value;
    const currency = availableCurrencies.find(c => c.code === selectedCode);

    if (currency) {
      // Update global currency settings
      setCurrencySettings({
        currency: currency.code,
        locale: currency.locale,
        rate: exchangeRates[currency.code] || currency.rate,
        exchangeRates
      });
      setCurrentCurrency(currency.code);

      // Store preference
      localStorage.setItem('preferred_currency', currency.code);

      // Reload the page to apply changes everywhere
      window.location.reload();
    }
  };

  const formatLastUpdated = (date) => {
    if (!date) return '';

    const now = new Date();
    const diffMs = now - date;
    const diffMins = Math.floor(diffMs / (1000 * 60));

    if (diffMins < 1) return 'just now';
    if (diffMins < 60) return `${diffMins} mins ago`;

    const diffHours = Math.floor(diffMins / 60);
```

```
    if (diffHours < 24) return `${diffHours} hours ago`;

    const diffDays = Math.floor(diffHours / 24);
    return `${diffDays} days ago`;
  };

  // If we have no currencies yet, show a loading state
  if (availableCurrencies.length === 0) {
    return (
      <div className={`flex items-center ${textColor} text-sm`}>
        <span className="mr-2">Loading currencies...</span>
        <div className={`animate-spin h-4 w-4 border-2 ${textColor} rounded-full
border-t-transparent`}></div>
      </div>
    );
  }

  return (
    <div className="flex flex-col items-end">
      <div className="flex items-center">
        <label htmlFor="currency-selector" className={`text-sm ${textColor} mr-2`}
>
          Currency:
        </label>
        <select
          id="currency-selector"
          value={currentCurrency}
          onChange={handleCurrencyChange}
          className={`text-sm border ${borderColor} rounded p-1 focus:outline-
none focus:ring-blue-500 dark:focus:ring-blue-400 focus:border-blue-500
dark:focus:border-blue-400 ${bgColor} ${textColor} font-medium`}
          style={{
            WebkitAppearance: 'menulist', // For Safari
            MozAppearance: 'menulist',    // For Firefox
            appearance: 'menulist'        // Standard
          }}
        >
          {availableCurrencies.map((currency) => (
            <option
              key={currency.code}
              value={currency.code}
              style={{ backgroundColor: '#fff', color: '#333', padding: '6px' }}
            >
              {currency.label} {currency.code !== 'USD' && `(1 USD =
${currency.rate.toFixed(2)})`}
            </option>
          ))}
        </select>
      </div>
      <div className={`text-xs ${subTextColor} mt-1`}>
        {fetchError && <span className={`${errorTextColor} mr-2`}>{fetchError}</
span>}
        {lastUpdated && `Rates updated: ${formatLastUpdated(lastUpdated)}`}
      </div>
    </div>
  );
};

export default CurrencySelector;
```

src/components/DarkModeProvider.jsx

```
import React from 'react';
import { useTheme } from '../hooks/useTheme';

// Higher-order component to automatically apply dark mode classes
const DarkModeProvider = ({ children, className = "", ...props }) => {
  const { theme } = useTheme();

  // Automatically add dark mode classes based on existing classes
  const enhanceClassName = (originalClassName) => {
    if (!originalClassName) return "transition-colors duration-300";

    let enhanced = originalClassName;

    // Add transition if not present
    if (!enhanced.includes('transition')) {
      enhanced += ' transition-colors duration-300';
    }

    // Auto-enhance common patterns
    const enhancements = {
      'bg-white dark:bg-slate-900': 'bg-white dark:bg-gray-800',
      'bg-gray-50 dark:bg-slate-800': 'bg-gray-50 dark:bg-gray-700',
      'text-gray-900 dark:text-white': 'text-gray-900 dark:text-white',
      'text-gray-700 dark:text-gray-300 dark:text-gray-400': 'text-gray-700
dark:text-gray-300',
      'text-gray-500 dark:text-gray-400': 'text-gray-500 dark:text-gray-400',
      'border-gray-200 dark:border-slate-700': 'border-gray-200 dark:border-
gray-600',
      'border-gray-300 dark:border-slate-600': 'border-gray-300 dark:border-
gray-600',
      'divide-gray-200 dark:divide-slate-700': 'divide-gray-200 dark:divide-
gray-700'
    };

    Object.entries(enhancements).forEach(([pattern, replacement]) => {
      if (enhanced.includes(pattern) && !enhanced.includes('dark:')) {
        enhanced = enhanced.replace(pattern, replacement);
      }
    });

    return enhanced;
  };

  const enhancedClassName = enhanceClassName(className);

  return (
    <div className={enhancedClassName} {...props}>
      {children}
    </div>
  );
};

// Hook to get enhanced classes
export const useDarkModeClasses = (baseClasses) => {
  const { theme } = useTheme();
```

```
  const enhanceClasses = (classes) => {
    if (!classes) return "transition-colors duration-300";

    let enhanced = classes;

    // Add transition
    if (!enhanced.includes('transition')) {
      enhanced += ' transition-colors duration-300';
    }

    // Common dark mode patterns
    const patterns = [
      { from: /bg-white dark:bg-slate-900(?!\s+dark:)/g, to: 'bg-white dark:bg-
gray-800' },
      { from: /bg-gray-50 dark:bg-slate-800(?!\s+dark:)/g, to: 'bg-gray-50
dark:bg-gray-700' },
      { from: /text-gray-900 dark:text-white(?!\s+dark:)/g, to: 'text-gray-900
dark:text-white' },
      { from: /text-gray-700 dark:text-gray-300 dark:text-gray-400(?!\s+dark:)/g,
to: 'text-gray-700 dark:text-gray-300' },
      { from: /text-gray-500 dark:text-gray-400(?!\s+dark:)/g, to: 'text-gray-500
dark:text-gray-400' },
      { from: /border-gray-200 dark:border-slate-700(?!\s+dark:)/g, to: 'border-
gray-200 dark:border-gray-600' },
      { from: /border-gray-300 dark:border-slate-600(?!\s+dark:)/g, to: 'border-
gray-300 dark:border-gray-600' },
      { from: /divide-gray-200 dark:divide-slate-700(?!\s+dark:)/g, to: 'divide-
gray-200 dark:divide-gray-700' }
    ];

    patterns.forEach(({ from, to }) => {
      enhanced = enhanced.replace(from, to);
    });

    return enhanced;
  };

  return enhanceClasses(baseClasses);
};

export default DarkModeProvider;
```

src/components/Employee/AddEmployeeDialog.jsx

```
import React, { useState } from 'react';
import { useAuth } from '../../context/AuthContext';
import employeeAPI from '../../services/employeeAPI';
import { toast } from 'react-toastify';

const AddEmployeeDialog = ({ open, onOpenChange, onEmployeeAdded, departments,
roles }) => {
  const { user } = useAuth();
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [formData, setFormData] = useState({
    fullName: '',
    email: '',
    phoneNumber: '',
    whatsappNumber: '',
```

```
  linkedInUrl: '',
  currentAddress: '',
  permanentAddress: '',
  collegeName: '',
  role: '',
  department: '',
  joiningDate: '',
  internshipDuration: '',
  salary: '',
  username: '',
  password: '',
});

const [files, setFiles] = useState({
  photograph: null,
  tenthMarksheet: null,
  twelfthMarksheet: null,
  bachelorDegree: null,
  postgraduateDegree: null,
  aadharCard: null,
  panCard: null,
  pcc: null,
  resume: null,
  offerLetter: null,
});

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

const handleFileChange = (e) => {
  const { name, files: selectedFiles } = e.target;
  if (selectedFiles && selectedFiles[0]) {
    const file = selectedFiles[0];
    const fileSizeKB = file.size / 1024;

    if (fileSizeKB < 10 || fileSizeKB > 20) {
      toast.error(`${name} file size must be between 10KB and 20KB`);
      e.target.value = ''; // Clear the file input
      return;
    }

    setFiles(prev => ({
      ...prev,
      [name]: file
    }));
  }
};

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

  try {
    const form = new FormData();
```

```
      // Create employee JSON payload
      const employeeJsonPayload = {
        fullName: formData.fullName,
        email: formData.email,
        phoneNumber: formData.phoneNumber,
        whatsappNumber: formData.whatsappNumber,
        linkedInUrl: formData.linkedInUrl,
        currentAddress: formData.currentAddress,
        permanentAddress: formData.permanentAddress,
        collegeName: formData.collegeName,
        role: formData.role,
        department: formData.department,
        joiningDate: new Date(formData.joiningDate).toISOString(),
        internshipDuration: formData.internshipDuration ?
parseInt(formData.internshipDuration) : undefined,
        salary: parseFloat(formData.salary),
        status: 'ACTIVE',
        hrId: user?.id || '',
      };

      form.append('employee', JSON.stringify(employeeJsonPayload));

      if (formData.username) form.append('username', formData.username);
      if (formData.password) form.append('password', formData.password);

      // Append files
      Object.keys(files).forEach(fieldName => {
        if (files[fieldName]) {
          form.append(fieldName, files[fieldName]);
        }
      });

      const response = await employeeAPI.create(form);

      if (response.data.success) {
        // Reset form
        setFormData({
          fullName: '',
          email: '',
          phoneNumber: '',
          whatsappNumber: '',
          linkedInUrl: '',
          currentAddress: '',
          permanentAddress: '',
          collegeName: '',
          role: '',
          department: '',
          joiningDate: '',
          internshipDuration: '',
          salary: '',
          username: '',
          password: '',
        });
        setFiles({
          photograph: null,
          tenthMarksheet: null,
          twelfthMarksheet: null,
          bachelorDegree: null,
```

```
          postgraduateDegree: null,
          aadharCard: null,
          panCard: null,
          pcc: null,
          resume: null,
          offerLetter: null,
        });
        onEmployeeAdded();
        onOpenChange(false);
      } else {
        alert('Failed to add employee. Please try again.');
      }
    } catch (error) {
      console.error('Error adding employee:', error);
      alert('An error occurred while adding the employee.');
    } finally {
      setIsSubmitting(false);
    }
  };

  const renderFileInput = (name, label) => (
    <div className="mb-4">
      <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
        {label}
      </label>
      <div className="flex items-center space-x-2">
        <input
          type="file"
          name={name}
          onChange={handleFileChange}
          accept=".pdf,.jpg,.jpeg,.png"
          className="w-full px-3 py-2 border border-gray-300 dark:border-gray-600
rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-gray-700
dark:text-white"
        />
        <span className="text-xs text-gray-500">(10KB-20KB)</span>
      </div>
    </div>
  );

  if (!open) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
      <div className="bg-white dark:bg-gray-800 rounded-lg shadow-xl max-w-2xl w-
full mx-4 max-h-[90vh] overflow-y-auto">
        <div className="p-6">
          <div className="flex justify-between items-center mb-6">
            <h2 className="text-xl font-bold text-gray-900 dark:text-white">Add
New Employee</h2>
            <button
              onClick={() => onOpenChange(false)}
              className="text-gray-400 hover:text-gray-600 dark:hover:text-
gray-300"
            >
              <svg className="w-6 h-6" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
```

```
                <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M6 18L18 6M6 6l12 12" />
              </svg>
            </button>
          </div>

          <form onSubmit={handleSubmit} className="space-y-4">
            {/* Basic Information */}
            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Full Name *
                </label>
                <input
                  type="text"
                  name="fullName"
                  value={formData.fullName}
                  onChange={handleInputChange}
                  required
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Email *
                </label>
                <input
                  type="email"
                  name="email"
                  value={formData.email}
                  onChange={handleInputChange}
                  required
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Phone Number
                </label>
                <input
                  type="tel"
                  name="phoneNumber"
                  value={formData.phoneNumber}
                  onChange={handleInputChange}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>
```

```
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  WhatsApp Number
                </label>
                <input
                  type="tel"
                  name="whatsappNumber"
                  value={formData.whatsappNumber}
                  onChange={handleInputChange}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Department *
                </label>
                <select
                  name="department"
                  value={formData.department}
                  onChange={handleInputChange}
                  required
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                >
                  <option value="">Select Department</option>
                  {departments.map((dept) => (
                    <option key={dept._id} value={dept._id}>
                      {dept.name}
                    </option>
                  ))}
                </select>
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Role *
                </label>
                <select
                  name="role"
                  value={formData.role}
                  onChange={handleInputChange}
                  required
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                >
                  <option value="">Select Role</option>
                  {roles.map((role) => (
                    <option key={role._id} value={role._id}>
                      {role.name}
                    </option>
                  ))}
```

```
                </select>
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Joining Date *
                </label>
                <input
                  type="date"
                  name="joiningDate"
                  value={formData.joiningDate}
                  onChange={handleInputChange}
                  required
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Salary *
                </label>
                <input
                  type="number"
                  name="salary"
                  value={formData.salary}
                  onChange={handleInputChange}
                  required
                  min="0"
                  step="0.01"
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>
            </div>

            {/* Additional Information */}
            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Current Address
                </label>
                <textarea
                  name="currentAddress"
                  value={formData.currentAddress}
                  onChange={handleInputChange}
                  rows="2"
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>

              <div>
```

```
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Permanent Address
                </label>
                <textarea
                  name="permanentAddress"
                  value={formData.permanentAddress}
                  onChange={handleInputChange}
                  rows="2"
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  College Name
                </label>
                <input
                  type="text"
                  name="collegeName"
                  value={formData.collegeName}
                  onChange={handleInputChange}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  LinkedIn URL
                </label>
                <input
                  type="url"
                  name="linkedInUrl"
                  value={formData.linkedInUrl}
                  onChange={handleInputChange}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
                />
              </div>
            </div>

            {/* User Account Information */}
            <div className="border-t pt-4">
              <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-4">User Account (Optional)</h3>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                    Username
                  </label>
                  <input
```

```
                    type="text"
                    name="username"
                    value={formData.username}
                    onChange={handleInputChange}
                    className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                    Password
                  </label>
                  <input
                    type="password"
                    name="password"
                    value={formData.password}
                    onChange={handleInputChange}
                    className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                  />
                </div>
              </div>
            </div>

            {/* File Uploads */}
            <div className="border-t pt-4">
              <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-4">Documents (Optional)</h3>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                {renderFileInput("photograph", "Photograph")}
                {renderFileInput("tenthMarksheet", "10th Marksheet")}
                {renderFileInput("twelfthMarksheet", "12th Marksheet")}
                {renderFileInput("bachelorDegree", "Bachelor's Degree")}
                {renderFileInput("postgraduateDegree", "Postgraduate Degree")}
                {renderFileInput("aadharCard", "Aadhar Card")}
                {renderFileInput("panCard", "PAN Card")}
                {renderFileInput("pcc", "Police Clearance Certificate")}
                {renderFileInput("resume", "Resume")}
                {renderFileInput("offerLetter", "Offer Letter")}
              </div>
            </div>

            {/* Submit Buttons */}
            <div className="flex justify-end space-x-3 pt-6 border-t">
              <button
                type="button"
                onClick={() => onOpenChange(false)}
                className="px-4 py-2 text-gray-700 dark:text-gray-300 bg-gray-100
dark:bg-gray-700 hover:bg-gray-200 dark:hover:bg-gray-600 rounded-lg transition-
colors"
              >
                Cancel
              </button>
              <button
                type="submit"
```

```
                disabled={isSubmitting}
                className="px-4 py-2 bg-blue-600 hover:bg-blue-700 text-white
rounded-lg transition-colors disabled:opacity-50 disabled:cursor-not-allowed flex
items-center space-x-2"
              >
                {isSubmitting && (
                  <div className="animate-spin rounded-full h-4 w-4 border-t-2
border-b-2 border-white"></div>
                )}
                <span>{isSubmitting ? 'Adding...' : 'Add Employee'}</span>
              </button>
            </div>
          </form>
        </div>
      </div>
    </div>
  );
};

export default AddEmployeeDialog;
```

src/components/Employee/AttendanceManagement.jsx

```
import React, { useState, useEffect } from 'react';
import { FaClock, FaCalendarCheck, FaSignInAlt, FaSignOutAlt, FaChartBar,
FaMapMarkerAlt, FaDownload, FaFilter, FaSpinner } from 'react-icons/fa';
import employeeAPI from '../../services/employeeAPI';

// Office location coordinates (update these with your actual office coordinates)
const OFFICE_LOCATION = {
  latitude: 28.607407 ,   // UPDATE: Replace with your actual office latitude
  longitude: 77.081754,  // UPDATE: Replace with your actual office longitude
  allowedRadius: 20    // 20 meters radius (you can adjust this)
};

const AttendanceManagement = ({ employeeId, userRole }) => {
  const [attendance, setAttendance] = useState([]);
  const [todayAttendance, setTodayAttendance] = useState(null);
  const [loading, setLoading] = useState(false);
  const [currentTime, setCurrentTime] = useState(new Date());
  const [selectedMonth, setSelectedMonth] = useState(new Date().getMonth());
  const [selectedYear, setSelectedYear] = useState(new Date().getFullYear());
  const [location, setLocation] = useState(null);
  const [gettingLocation, setGettingLocation] = useState(false);
  const [locationError, setLocationError] = useState('');
  const [isInOfficeRange, setIsInOfficeRange] = useState(false);
  const [filterStatus, setFilterStatus] = useState('all');
  const [showReports, setShowReports] = useState(false);

  // Calculate distance between two coordinates using Haversine formula
  const calculateDistance = (lat1, lon1, lat2, lon2) => {
    const R = 6371e3; // Earth's radius in meters
    const <c  Ò Æ C  ¢ Ö F,å ’ ò  ƒ °
    const <c" Ò Æ C" ¢ Ö F,å ’ ò  ƒ °
    const 9CÆ = (lat2 - lat1) * Math.PI / 180;
    const 9C» = (lon2 - lon1) * Math.PI / 180;

    const a = Math.sin(9CÆ/2) * Math.sin(9CÆ/2) +
```

```
              Math.cos(<c ’ ¢ Ö F,æ6÷2ƒÆ2) *
              Math.sin(9C»/2) * Math.sin(9C»/2);
    const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

    return R * c; // Distance in meters
  };

  // Check if user is within office range
  const checkLocationValidity = (userLat, userLng) => {
    const distance = calculateDistance(
      OFFICE_LOCATION.latitude,
      OFFICE_LOCATION.longitude,
      userLat,
      userLng
    );
    return distance <= OFFICE_LOCATION.allowedRadius;
  };

  useEffect(() => {
    const timer = setInterval(() => {
      setCurrentTime(new Date());
    }, 1000);

    return () => clearInterval(timer);
  }, []);

  useEffect(() => {
    // Initial fetch
    if (employeeId) {
      fetchAttendance();
      fetchTodayAttendance();
    }

    // Set up interval to refresh data every minute
    const interval = setInterval(() => {
      if (employeeId) {
        fetchAttendance();
        fetchTodayAttendance();
      }
    }, 60000); // 60000 ms = 1 minute

    return () => clearInterval(interval);
  }, [employeeId]);

  const fetchAttendance = async () => {
    try {
      setLoading(true);
      // This would be implemented in the API
      // const response = await employeeAPI.getAttendance(employeeId,
selectedMonth, selectedYear);
      // setAttendance(response.data || []);

      // Mock data for now
      const mockAttendance = [];
      const daysInMonth = new Date(selectedYear, selectedMonth + 1, 0).getDate();

      for (let day = 1; day <= daysInMonth; day++) {
        const date = new Date(selectedYear, selectedMonth, day);
        const isWeekend = date.getDay() === 0 || date.getDay() === 6;
```

```
        const isFuture = date > new Date();

        if (!isWeekend && !isFuture && Math.random() > 0.1) { // 90% attendance
          mockAttendance.push({
            _id: `${selectedYear}-${selectedMonth}-${day}`,
            date: date.toISOString().split('T')[0],
            checkIn: `09:${Math.floor(Math.random() * 30).toString().padStart(2,
'0')}`,
            checkOut: `18:${Math.floor(Math.random() * 30).toString().padStart(2,
'0')}`,
            workingHours: 8 + Math.random() * 2,
            status: 'Present'
          });
        }
      }

      setAttendance(mockAttendance);
    } catch (error) {
      console.error('Error fetching attendance:', error);
    } finally {
      setLoading(false);
    }
  };

  const fetchTodayAttendance = async () => {
    try {
      // This would be implemented in the API
      // const response = await employeeAPI.getTodayAttendance(employeeId);
      // setTodayAttendance(response.data);

      // Get today's date
      const today = new Date().toISOString().split('T')[0];
      const existingToday = attendance.find(att => att.date === today);

      if (!existingToday) {
        setTodayAttendance({
          date: today,
          checkIn: null,
          checkOut: null,
          workingHours: 0,
          status: 'Absent'
        });
      } else {
        setTodayAttendance(existingToday);
      }
    } catch (error) {
      console.error('Error fetching today attendance:', error);
    }
  };

  const getCurrentLocation = () => {
    return new Promise((resolve, reject) => {
      if (!navigator.geolocation) {
        reject(new Error('Geolocation is not supported by this browser.'));
        return;
      }

      setGettingLocation(true);
      setLocationError('');
```

```
      navigator.geolocation.getCurrentPosition(
        (position) => {
          setGettingLocation(false);
          const userLocation = {
            latitude: position.coords.latitude,
            longitude: position.coords.longitude,
            accuracy: position.coords.accuracy
          };

          // Check if user is within office range
          const isInRange = checkLocationValidity(userLocation.latitude,
userLocation.longitude);
          const distance = calculateDistance(
            OFFICE_LOCATION.latitude,
            OFFICE_LOCATION.longitude,
            userLocation.latitude,
            userLocation.longitude
          );

          userLocation.distance = distance;
          userLocation.isInOfficeRange = isInRange;

          setLocation(userLocation);
          setIsInOfficeRange(isInRange);

          if (!isInRange) {
            setLocationError(`You are ${distance.toFixed(1)} meters away from
office. Please move within ${OFFICE_LOCATION.allowedRadius} meters to mark
attendance.`);
          } else {
            setLocationError('');
          }

          resolve(userLocation);
        },
        (error) => {
          setGettingLocation(false);
          let errorMessage = 'Unable to get your location.';

          switch (error.code) {
            case error.PERMISSION_DENIED:
              errorMessage = 'Location access denied. Please enable location
services.';
              break;
            case error.POSITION_UNAVAILABLE:
              errorMessage = 'Location information is unavailable.';
              break;
            case error.TIMEOUT:
              errorMessage = 'Location request timed out.';
              break;
            default:
              errorMessage = 'An unknown error occurred while getting location.';
              break;
          }

          setLocationError(errorMessage);
          reject(error);
        },
```

```
        { enableHighAccuracy: true, timeout: 10000, maximumAge: 0 }
      );
    });
  };

  const handleCheckIn = async () => {
    try {
      const now = new Date();
      const timeString = now.toTimeString().slice(0, 5);

      // Get location for check-in
      let locationData = null;
      try {
        locationData = await getCurrentLocation();

        // Check if user is within office range
        if (!locationData.isInOfficeRange) {
          alert(`Cannot check in: You are ${locationData.distance.toFixed(1)}
meters away from office. Please move within ${OFFICE_LOCATION.allowedRadius}
meters of the office location.`);
          return;
        }
      } catch (error) {
        console.warn('Could not get location:', error);
        alert('Location access is required for attendance marking. Please enable
location services and try again.');
        return;
      }

      // This would be implemented in the API
      // await employeeAPI.checkIn(employeeId, { location: locationData });

      const updatedAttendance = {
        ...todayAttendance,
        checkIn: timeString,
        status: 'Present',
        location: locationData
      };

      setTodayAttendance(updatedAttendance);

      // Update attendance list
      const today = new Date().toISOString().split('T')[0];
      setAttendance(prev => {
        const filtered = prev.filter(att => att.date !== today);
        return [...filtered, updatedAttendance];
      });

      alert('Check-in successful! You are within the office premises.');
    } catch (error) {
      console.error('Error checking in:', error);
      alert('Failed to check in. Please try again.');
    }
  };

  const handleCheckOut = async () => {
    try {
      const now = new Date();
      const timeString = now.toTimeString().slice(0, 5);
```

```
      // Get location for check-out
      let locationData = null;
      try {
        locationData = await getCurrentLocation();

        // Check if user is within office range
        if (!locationData.isInOfficeRange) {
          alert(`Cannot check out: You are ${locationData.distance.toFixed(1)}
meters away from office. Please move within ${OFFICE_LOCATION.allowedRadius}
meters of the office location.`);
          return;
        }
      } catch (error) {
        console.warn('Could not get location:', error);
        alert('Location access is required for attendance marking. Please enable
location services and try again.');
        return;
      }

      // Calculate working hours
      const checkInTime = new Date(`2000-01-01 ${todayAttendance.checkIn}`);
      const checkOutTime = new Date(`2000-01-01 ${timeString}`);
      const workingHours = (checkOutTime - checkInTime) / (1000 * 60 * 60);

      // This would be implemented in the API
      // await employeeAPI.checkOut(employeeId);

      const updatedAttendance = {
        ...todayAttendance,
        checkOut: timeString,
        workingHours: workingHours,
        status: 'Present'
      };

      setTodayAttendance(updatedAttendance);

      // Update attendance list
      const today = new Date().toISOString().split('T')[0];
      setAttendance(prev => {
        const filtered = prev.filter(att => att.date !== today);
        return [...filtered, updatedAttendance];
      });

      alert('Check-out successful! Have a great day.');
    } catch (error) {
      console.error('Error checking out:', error);
      alert('Failed to check out. Please try again.');
    }
  };

  const calculateStats = () => {
    const totalDays = attendance.length;
    const presentDays = attendance.filter(att => att.status === 'Present').length;
    const totalHours = attendance.reduce((sum, att) => sum + (att.workingHours ||
0), 0);
    const avgHours = totalDays > 0 ? totalHours / totalDays : 0;

    return {
```

```
      totalDays,
      presentDays,
      totalHours: totalHours.toFixed(1),
      avgHours: avgHours.toFixed(1),
      attendanceRate: totalDays > 0 ? ((presentDays / totalDays) *
100).toFixed(1) : 0
    };
  };

  const exportAttendanceData = () => {
    const filteredData = getFilteredAttendance();
    const csvContent = [
      ['Date', 'Check In', 'Check Out', 'Hours', 'Status', 'Location'],
      ...filteredData.map(record => [
        new Date(record.date).toLocaleDateString(),
        record.checkIn || '--',
        record.checkOut || '--',
        record.workingHours ? `${record.workingHours.toFixed(1)}h` : '--',
        record.status,
        record.location ? 'Yes' : 'No'
      ])
    ].map(row => row.join(',')).join('\n');

    const blob = new Blob([csvContent], { type: 'text/csv' });
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `attendance_${months[selectedMonth]}_${selectedYear}.csv`;
    a.click();
    window.URL.revokeObjectURL(url);
  };

  const getFilteredAttendance = () => {
    return attendance.filter(record => {
      if (filterStatus === 'all') return true;
      return record.status === filterStatus;
    });
  };

  const stats = calculateStats();

  const months = [
    'January', 'February', 'March', 'April', 'May', 'June',
    'July', 'August', 'September', 'October', 'November', 'December'
  ];

  return (
    <div className="space-y-6">
      {/* Today's Attendance */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <h3 className="text-lg font-semibold text-gray-900 dark:text-white flex
items-center mb-4">
          <FaClock className="mr-2 text-blue-600" />
          Today's Attendance
        </h3>

        <div className="grid grid-cols-1 md:grid-cols-3 gap-4 mb-6">
          <div className="text-center">
            <div className="text-2xl font-bold text-blue-600">
```

```
              {currentTime.toTimeString().slice(0, 8)}
            </div>
            <div className="text-sm text-gray-600 dark:text-gray-400">Current
Time</div>
          </div>

          <div className="text-center">
            <div className="text-2xl font-bold text-green-600">
              {todayAttendance?.checkIn || '--:--'}
            </div>
            <div className="text-sm text-gray-600 dark:text-gray-400">Check In</
div>
          </div>

          <div className="text-center">
            <div className="text-2xl font-bold text-red-600">
              {todayAttendance?.checkOut || '--:--'}
            </div>
            <div className="text-sm text-gray-600 dark:text-gray-400">Check Out</
div>
          </div>
        </div>

        <div className="flex justify-center gap-4">
          {!todayAttendance?.checkIn ? (
            <button
              onClick={handleCheckIn}
              disabled={gettingLocation}
              className="px-6 py-3 bg-green-600 text-white rounded-lg hover:bg-
green-700 transition-colors flex items-center disabled:opacity-50"
            >
              {gettingLocation ? (
                <FaSpinner className="animate-spin mr-2" />
              ) : (
                <FaSignInAlt className="mr-2" />
              )}
              {gettingLocation ? 'Getting Location...' : 'Check In'}
            </button>
          ) : !todayAttendance?.checkOut ? (
            <button
              onClick={handleCheckOut}
              disabled={gettingLocation}
              className="px-6 py-3 bg-red-600 text-white rounded-lg hover:bg-
red-700 transition-colors flex items-center disabled:opacity-50"
            >
              {gettingLocation ? (
                <FaSpinner className="animate-spin mr-2" />
              ) : (
                <FaSignOutAlt className="mr-2" />
              )}
              {gettingLocation ? 'Getting Location...' : 'Check Out'}
            </button>
          ) : (
            <div className="text-center">
              <div className="text-green-600 font-medium">
                '  Attendance marked for today
              </div>
              <div className="text-sm text-gray-600 dark:text-gray-400">
                Working Hours: {todayAttendance.workingHours?.toFixed(1)} hours
```

```
              </div>
              {todayAttendance?.location && (
                <div className="text-xs text-gray-500 dark:text-gray-400 mt-1
flex items-center justify-center">
                  <FaMapMarkerAlt className="mr-1" />
                  Location tracked
                </div>
              )}
            </div>
          )}
        </div>

        {/* Location Status */}
        {location && (
          <div className={`mt-4 p-3 rounded-lg ${
            isInOfficeRange
              ? 'bg-green-50 dark:bg-green-900/20 border border-green-200
dark:border-green-800'
              : 'bg-red-50 dark:bg-red-900/20 border border-red-200 dark:border-
red-800'
          }`}>
            <div className={`flex items-center text-sm ${
              isInOfficeRange
                ? 'text-green-600 dark:text-green-400'
                : 'text-red-600 dark:text-red-400'
            }`}>
              <FaMapMarkerAlt className="mr-2" />
              <div>
                <div className="font-medium">
                  {isInOfficeRange ? ''  Within Office Range' : ''L Outside Office
Range'}
                </div>
                <div className="text-xs mt-1">
                  Your Location: {location.latitude.toFixed(6)},
{location.longitude.toFixed(6)}
                </div>
                <div className="text-xs">
                  Office Location: {OFFICE_LOCATION.latitude.toFixed(6)},
{OFFICE_LOCATION.longitude.toFixed(6)}
                </div>
                <div className="text-xs">
                  Distance: {location.distance ? `${location.distance.toFixed(1)}
m` : 'Calculating...'}
                  (Required: "d {OFFICE_LOCATION.allowedRadius}m)
                </div>
                <div className="text-xs">
                  GPS Accuracy: ±{location.accuracy.toFixed(0)}m
                </div>
              </div>
            </div>
          </div>
        )}

        {/* Location Error */}
        {locationError && (
          <div className="mt-4 p-3 bg-red-50 dark:bg-red-900/20 border border-
red-200 dark:border-red-800 rounded-lg">
            <div className="flex items-center text-sm text-red-600 dark:text-
red-400">
```

```
              <FaMapMarkerAlt className="mr-2" />
              <div>
                <div className="font-medium">Location Error</div>
                <div className="text-xs mt-1">{locationError}</div>
              </div>
            </div>
          </div>
        )}

        {/* Office Location Info */}
        <div className="mt-4 p-3 bg-blue-50 dark:bg-blue-900/20 border border-
blue-200 dark:border-blue-800 rounded-lg">
          <div className="flex items-center text-sm text-blue-600 dark:text-
blue-400">
            <FaMapMarkerAlt className="mr-2" />
            <div>
              <div className="font-medium">Office Location</div>
              <div className="text-xs mt-1">
                {OFFICE_LOCATION.latitude.toFixed(6)},
{OFFICE_LOCATION.longitude.toFixed(6)}
              </div>
              <div className="text-xs">
                Attendance allowed within {OFFICE_LOCATION.allowedRadius} meters
radius
              </div>
            </div>
          </div>
        </div>
      </div>

      {/* Monthly Statistics */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <h3 className="text-lg font-semibold text-gray-900 dark:text-white flex
items-center mb-4">
          <FaChartBar className="mr-2 text-blue-600" />
          Monthly Statistics
        </h3>

        <div className="grid grid-cols-2 md:grid-cols-5 gap-4 mb-6">
          <div className="text-center">
            <div className="text-2xl font-bold text-blue-600">{stats.totalDays}</
div>
            <div className="text-sm text-gray-600 dark:text-gray-400">Total Days</
div>
          </div>
          <div className="text-center">
            <div className="text-2xl font-bold text-green-600">{stats.presentDays}
</div>
            <div className="text-sm text-gray-600 dark:text-gray-400">Present</
div>
          </div>
          <div className="text-center">
            <div className="text-2xl font-bold text-purple-600">{stats.totalHours}
h</div>
            <div className="text-sm text-gray-600 dark:text-gray-400">Total
Hours</div>
          </div>
          <div className="text-center">
            <div className="text-2xl font-bold text-orange-600">{stats.avgHours}
```

```
h</div>
            <div className="text-sm text-gray-600 dark:text-gray-400">Avg Hours</
div>
          </div>
          <div className="text-center">
            <div className="text-2xl font-bold text-
indigo-600">{stats.attendanceRate}%</div>
            <div className="text-sm text-gray-600 dark:text-gray-400">Attendance</
div>
          </div>
        </div>
      </div>

      {/* Attendance History */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <div className="flex flex-col md:flex-row justify-between items-start
md:items-center mb-4 gap-4">
          <h3 className="text-lg font-semibold text-gray-900 dark:text-white flex
items-center">
            <FaCalendarCheck className="mr-2 text-blue-600" />
            Attendance History
          </h3>

          <div className="flex flex-wrap gap-2">
            <select
              value={filterStatus}
              onChange={(e) => setFilterStatus(e.target.value)}
              className="border border-gray-300 dark:border-gray-600 rounded px-3
py-1 bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
            >
              <option value="all">All Status</option>
              <option value="Present">Present</option>
              <option value="Absent">Absent</option>
              <option value="Late">Late</option>
              <option value="Half Day">Half Day</option>
            </select>
            <select
              value={selectedMonth}
              onChange={(e) => setSelectedMonth(parseInt(e.target.value))}
              className="border border-gray-300 dark:border-gray-600 rounded px-3
py-1 bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
            >
              {months.map((month, index) => (
                <option key={index} value={index}>{month}</option>
              ))}
            </select>
            <select
              value={selectedYear}
              onChange={(e) => setSelectedYear(parseInt(e.target.value))}
              className="border border-gray-300 dark:border-gray-600 rounded px-3
py-1 bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
            >
              {[2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030].map(year => (
                <option key={year} value={year}>{year}</option>
              ))}
            </select>
            <button
              onClick={() => exportAttendanceData()}
              className="flex items-center px-3 py-1 bg-green-600 text-white
```

```
rounded hover:bg-green-700 transition-colors"
            >
              <FaDownload className="mr-1" />
              Export
            </button>
            <button
              onClick={() => setShowReports(!showReports)}
              className="flex items-center px-3 py-1 bg-purple-600 text-white
rounded hover:bg-purple-700 transition-colors"
            >
              <FaChartBar className="mr-1" />
              Reports
            </button>
          </div>
        </div>

        {loading ? (
          <div className="flex justify-center py-8">
            <div className="animate-spin rounded-full h-8 w-8 border-t-2 border-
b-2 border-blue-500"></div>
          </div>
        ) : (
          <div className="overflow-x-auto">
            <table className="min-w-full">
              <thead className="bg-gray-50 dark:bg-slate-800">
                <tr>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase">Date</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase">Check In</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase">Check Out</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase">Hours</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase">Status</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase">Location</th>
                </tr>
              </thead>
              <tbody className="divide-y divide-gray-200 dark:divide-gray-700">
                {getFilteredAttendance().length === 0 ? (
                  <tr>
                    <td colSpan="6" className="px-4 py-8 text-center text-
gray-500 dark:text-gray-400">
                      No attendance records found for selected filters
                    </td>
                  </tr>
                ) : (
                  getFilteredAttendance().map((record) => (
                    <tr key={`${record.date}-${record._id || Date.now()}`}
className="hover:bg-gray-50 dark:hover:bg-slate-800">
                      <td className="px-4 py-2 text-sm text-gray-900 dark:text-
white">
                        {new Date(record.date).toLocaleDateString()}
                      </td>
                      <td className="px-4 py-2 text-sm text-gray-900 dark:text-
white">
                        {record.checkIn || '--'}
```

```
                    </td>
                    <td className="px-4 py-2 text-sm text-gray-900 dark:text-
white">
                      {record.checkOut || '--'}
                    </td>
                    <td className="px-4 py-2 text-sm text-gray-900 dark:text-
white">
                      {record.workingHours ? `${record.workingHours.toFixed(1)}
h` : '--'}
                    </td>
                    <td className="px-4 py-2">
                      <span className={`px-2 py-1 rounded-full text-xs font-
medium ${
                        record.status === 'Present'
                          ? 'bg-green-100 text-green-800 dark:bg-green-900
dark:text-green-200'
                          : record.status === 'Late'
                          ? 'bg-yellow-100 text-yellow-800 dark:bg-yellow-900
dark:text-yellow-200'
                          : record.status === 'Half Day'
                          ? 'bg-orange-100 text-orange-800 dark:bg-orange-900
dark:text-orange-200'
                          : 'bg-red-100 text-red-800 dark:bg-red-900 dark:text-
red-200'
                      }`}>
                        {record.status}
                      </span>
                    </td>
                    <td className="px-4 py-2 text-sm text-gray-900 dark:text-
white">
                      {record.location ? (
                        <FaMapMarkerAlt className="text-green-600"
title="Location tracked" />
                      ) : (
                        <span className="text-gray-400">--</span>
                      )}
                    </td>
                  </tr>
                ))
              )}
            </tbody>
          </table>
        </div>
      )}

      {/* Enhanced Reports Section */}
      {showReports && (
        <div className="mt-6 bg-gray-50 dark:bg-slate-800 rounded-lg p-6">
          <h4 className="text-lg font-semibold text-gray-900 dark:text-white
mb-4">
            Detailed Analytics
          </h4>
          <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
            <div className="bg-white dark:bg-slate-900 p-4 rounded-lg">
              <h5 className="font-medium text-gray-700 dark:text-gray-300
mb-2">Weekly Pattern</h5>
              <div className="text-sm text-gray-600 dark:text-gray-400">
                <p>Monday: {attendance.filter(a => new Date(a.date).getDay()
=== 1).length} days</p>
```

```
                  <p>Tuesday: {attendance.filter(a => new Date(a.date).getDay()
=== 2).length} days</p>
                  <p>Wednesday: {attendance.filter(a => new Date(a.date).getDay()
=== 3).length} days</p>
                  <p>Thursday: {attendance.filter(a => new Date(a.date).getDay()
=== 4).length} days</p>
                  <p>Friday: {attendance.filter(a => new Date(a.date).getDay()
=== 5).length} days</p>
                </div>
              </div>

              <div className="bg-white dark:bg-slate-900 p-4 rounded-lg">
                <h5 className="font-medium text-gray-700 dark:text-gray-300
mb-2">Time Patterns</h5>
                <div className="text-sm text-gray-600 dark:text-gray-400">
                  <p>Early Check-ins: {attendance.filter(a => a.checkIn &&
a.checkIn < '09:00').length}</p>
                  <p>Late Check-ins: {attendance.filter(a => a.checkIn &&
a.checkIn > '09:15').length}</p>
                  <p>Early Check-outs: {attendance.filter(a => a.checkOut &&
a.checkOut < '17:30').length}</p>
                  <p>Late Check-outs: {attendance.filter(a => a.checkOut &&
a.checkOut > '18:30').length}</p>
                </div>
              </div>

              <div className="bg-white dark:bg-slate-900 p-4 rounded-lg">
                <h5 className="font-medium text-gray-700 dark:text-gray-300
mb-2">Performance</h5>
                <div className="text-sm text-gray-600 dark:text-gray-400">
                  <p>Perfect Days: {attendance.filter(a => a.workingHours >=
8).length}</p>
                  <p>Short Days: {attendance.filter(a => a.workingHours < 8 &&
a.workingHours > 0).length}</p>
                  <p>Location Tracked: {attendance.filter(a =>
a.location).length} days</p>
                  <p>Punctuality Score: {attendance.length > 0 ?
((attendance.filter(a => a.checkIn && a.checkIn <= '09:00').length /
attendance.length) * 100).toFixed(1) : 0}%</p>
                </div>
              </div>
            </div>
          </div>
        )}
      </div>
    </div>
  );
};

export default AttendanceManagement;
```

src/components/Employee/BulkOperations.jsx

```
import React, { useState, useEffect } from 'react';
import { FaCheck, FaDownload, FaUpload, FaEdit, FaTrash, FaUsers, FaFilter,
FaSpinner, FaCog } from 'react-icons/fa';
import employeeAPI from '../../services/employeeAPI';
import { useAuth } from '../../context/AuthContext';
```

```
const BulkOperations = () => {
  const { user } = useAuth();
  const [employees, setEmployees] = useState([]);
  const [departments, setDepartments] = useState([]);
  const [roles, setRoles] = useState([]);
  const [selectedEmployees, setSelectedEmployees] = useState([]);
  const [loading, setLoading] = useState(true);
  const [processing, setProcessing] = useState(false);
  const [activeOperation, setActiveOperation] = useState('update');
  const [bulkData, setBulkData] = useState({
    department: '',
    role: '',
    status: '',
    salary: ''
  });
  const [filterStatus, setFilterStatus] = useState('all');
  const [filterDepartment, setFilterDepartment] = useState('all');

  useEffect(() => {
    fetchData();
  }, []);

  const fetchData = async () => {
    try {
      setLoading(true);
      const [employeesRes, departmentsRes, rolesRes] = await Promise.all([
        employeeAPI.getAll(),
        employeeAPI.getDepartments(),
        employeeAPI.getRoles()
      ]);

      if (employeesRes.data.success) {
        setEmployees(employeesRes.data.data);
      }
      if (departmentsRes.data.success) {
        setDepartments(departmentsRes.data.data);
      }
      if (rolesRes.data.success) {
        setRoles(rolesRes.data.data);
      }
    } catch (error) {
      console.error('Error fetching data:', error);
    } finally {
      setLoading(false);
    }
  };

  const getFilteredEmployees = () => {
    return employees.filter(emp => {
      if (filterStatus !== 'all' && emp.status !== filterStatus) return false;
      if (filterDepartment !== 'all') {
        const deptId = emp.department?._id || emp.department;
        if (deptId !== filterDepartment) return false;
      }
      return true;
    });
  };
```

```
  const handleSelectEmployee = (employeeId) => {
    setSelectedEmployees(prev => {
      if (prev.includes(employeeId)) {
        return prev.filter(id => id !== employeeId);
      } else {
        return [...prev, employeeId];
      }
    });
  };

  const handleSelectAll = () => {
    const filteredEmployees = getFilteredEmployees();
    if (selectedEmployees.length === filteredEmployees.length) {
      setSelectedEmployees([]);
    } else {
      setSelectedEmployees(filteredEmployees.map(emp => emp._id));
    }
  };

  const handleBulkUpdate = async () => {
    if (selectedEmployees.length === 0) {
      alert('Please select employees to update');
      return;
    }

    const updateData = Object.entries(bulkData)
      .filter(([key, value]) => value !== '')
      .reduce((acc, [key, value]) => {
        acc[key] = value;
        return acc;
      }, {});

    if (Object.keys(updateData).length === 0) {
      alert('Please specify at least one field to update');
      return;
    }

    try {
      setProcessing(true);

      // This would be implemented in the API
      // await employeeAPI.bulkUpdate(selectedEmployees, updateData);

      // Mock implementation - in real scenario, this would be a single API call
      for (const employeeId of selectedEmployees) {
        console.log(`Updating employee ${employeeId} with:`, updateData);
        // await employeeAPI.update(employeeId, { employee:
JSON.stringify(updateData) });
      }

      alert(`Successfully updated ${selectedEmployees.length} employees`);
      setSelectedEmployees([]);
      setBulkData({ department: '', role: '', status: '', salary: '' });
      fetchData();
    } catch (error) {
      console.error('Error in bulk update:', error);
      alert('Failed to update employees. Please try again.');
    } finally {
      setProcessing(false);
```

```
    }
  };

  const exportEmployeeData = () => {
    const dataToExport = selectedEmployees.length > 0
      ? employees.filter(emp => selectedEmployees.includes(emp._id))
      : getFilteredEmployees();

    const csvContent = [
      ['Name', 'Email', 'Phone', 'Department', 'Role', 'Status', 'Salary',
'Joining Date'],
      ...dataToExport.map(emp => [
        emp.fullName || '',
        emp.email || '',
        emp.phoneNumber || '',
        emp.department?.name || '',
        emp.role?.name || '',
        emp.status || '',
        emp.salary || '',
        emp.joiningDate ? new Date(emp.joiningDate).toLocaleDateString() : ''
      ])
    ].map(row => row.map(cell => `"${cell}"`).join(',')).join('\n');

    const blob = new Blob([csvContent], { type: 'text/csv' });
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `employees_export_${new Date().toISOString().split('T')[0]}.csv`;
    a.click();
    window.URL.revokeObjectURL(url);
  };

  const generateReport = () => {
    const filteredEmployees = getFilteredEmployees();

    const report = {
      totalEmployees: filteredEmployees.length,
      activeEmployees: filteredEmployees.filter(emp => emp.status ===
'ACTIVE').length,
      departmentBreakdown: departments.map(dept => ({
        department: dept.name,
        count: filteredEmployees.filter(emp => {
          const deptId = emp.department?._id || emp.department;
          return deptId === dept._id;
        }).length
      })),
      roleBreakdown: roles.map(role => ({
        role: role.name,
        count: filteredEmployees.filter(emp => {
          const roleId = emp.role?._id || emp.role;
          return roleId === role._id;
        }).length
      })),
      averageSalary: filteredEmployees.reduce((sum, emp) => sum + (emp.salary ||
0), 0) / filteredEmployees.length || 0,
      newHires: filteredEmployees.filter(emp => {
        const joiningDate = new Date(emp.joiningDate);
        const threeMonthsAgo = new Date();
        threeMonthsAgo.setMonth(threeMonthsAgo.getMonth() - 3);
```

```
        return joiningDate > threeMonthsAgo;
      }).length
    };

    const reportContent = `
Employee Report - Generated on ${new Date().toLocaleDateString()}

SUMMARY
=======
Total Employees: ${report.totalEmployees}
Active Employees: ${report.activeEmployees}
Average Salary: $${report.averageSalary.toFixed(2)}
New Hires (Last 3 months): ${report.newHires}

DEPARTMENT BREAKDOWN
==================
${report.departmentBreakdown.map(d => `${d.department}: ${d.count}
employees`).join('\n')}

ROLE BREAKDOWN
==============
${report.roleBreakdown.map(r => `${r.role}: ${r.count} employees`).join('\n')}
    `.trim();

    const blob = new Blob([reportContent], { type: 'text/plain' });
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `employee_report_${new Date().toISOString().split('T')[0]}.txt`;
    a.click();
    window.URL.revokeObjectURL(url);
  };

  if (loading) {
    return (
      <div className="flex justify-center items-center h-64">
        <div className="animate-spin rounded-full h-8 w-8 border-t-2 border-b-2
border-blue-500"></div>
        <span className="ml-2">Loading employees...</span>
      </div>
    );
  }

  const filteredEmployees = getFilteredEmployees();

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <div className="flex justify-between items-center">
          <div>
            <h2 className="text-xl font-bold text-gray-900 dark:text-white flex
items-center">
              <FaCog className="mr-2 text-blue-600" />
              Bulk Operations
            </h2>
            <p className="text-gray-600 dark:text-gray-400 mt-1">
              Manage multiple employees efficiently
            </p>
```

```
          </div>
          <div className="text-sm text-gray-500 dark:text-gray-400">
            {selectedEmployees.length} of {filteredEmployees.length} selected
          </div>
        </div>
      </div>

      {/* Operation Tabs */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <div className="flex space-x-4 mb-6">
          <button
            onClick={() => setActiveOperation('update')}
            className={`px-4 py-2 rounded-md ${activeOperation === 'update' ? 'bg-
blue-600 text-white' : 'bg-gray-200 text-gray-700'}`}
          >
            Bulk Update
          </button>
          <button
            onClick={() => setActiveOperation('export')}
            className={`px-4 py-2 rounded-md ${activeOperation === 'export' ? 'bg-
blue-600 text-white' : 'bg-gray-200 text-gray-700'}`}
          >
            Export Data
          </button>
          <button
            onClick={() => setActiveOperation('reports')}
            className={`px-4 py-2 rounded-md ${activeOperation === 'reports' ?
'bg-blue-600 text-white' : 'bg-gray-200 text-gray-700'}`}
          >
            Reports
          </button>
        </div>

        {/* Bulk Update Tab */}
        {activeOperation === 'update' && (
          <div className="space-y-4">
            <h3 className="text-lg font-semibold text-gray-900 dark:text-white">
              Update Selected Employees
            </h3>
            <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Department
                </label>
                <select
                  value={bulkData.department}
                  onChange={(e) => setBulkData(prev => ({ ...prev, department:
e.target.value }))}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md dark:bg-gray-700 dark:text-white"
                >
                  <option value="">No Change</option>
                  {departments.map(dept => (
                    <option key={dept._id} value={dept._id}>{dept.name}</option>
                  ))}
                </select>
              </div>
```

```
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Role
                </label>
                <select
                  value={bulkData.role}
                  onChange={(e) => setBulkData(prev => ({ ...prev, role:
e.target.value }))}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md dark:bg-gray-700 dark:text-white"
                >
                  <option value="">No Change</option>
                  {roles.map(role => (
                    <option key={role._id} value={role._id}>{role.name}</option>
                  ))}
                </select>
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Status
                </label>
                <select
                  value={bulkData.status}
                  onChange={(e) => setBulkData(prev => ({ ...prev, status:
e.target.value }))}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md dark:bg-gray-700 dark:text-white"
                >
                  <option value="">No Change</option>
                  <option value="ACTIVE">Active</option>
                  <option value="INACTIVE">Inactive</option>
                  <option value="TERMINATED">Terminated</option>
                </select>
              </div>

              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Salary Adjustment (%)
                </label>
                <input
                  type="number"
                  value={bulkData.salary}
                  onChange={(e) => setBulkData(prev => ({ ...prev, salary:
e.target.value }))}
                  placeholder="e.g., 10 for 10% increase"
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md dark:bg-gray-700 dark:text-white"
                />
              </div>
            </div>

            <button
              onClick={handleBulkUpdate}
              disabled={processing || selectedEmployees.length === 0}
              className="px-6 py-2 bg-blue-600 text-white rounded-md hover:bg-
```

```
blue-700 disabled:opacity-50 disabled:cursor-not-allowed flex items-center"
            >
              {processing ? <FaSpinner className="animate-spin mr-2" /> : <FaEdit
className="mr-2" />}
              {processing ? 'Updating...' : `Update ${selectedEmployees.length}
Employees`}
            </button>
          </div>
        )}

        {/* Export Tab */}
        {activeOperation === 'export' && (
          <div className="space-y-4">
            <h3 className="text-lg font-semibold text-gray-900 dark:text-white">
              Export Employee Data
            </h3>
            <p className="text-gray-600 dark:text-gray-400">
              Export {selectedEmployees.length > 0 ? 'selected' : 'filtered'}
employee data to CSV format.
            </p>
            <button
              onClick={exportEmployeeData}
              className="px-6 py-2 bg-green-600 text-white rounded-md hover:bg-
green-700 flex items-center"
            >
              <FaDownload className="mr-2" />
              Export {selectedEmployees.length > 0 ? `${selectedEmployees.length}
Selected` : `${filteredEmployees.length} Filtered`} Employees
            </button>
          </div>
        )}

        {/* Reports Tab */}
        {activeOperation === 'reports' && (
          <div className="space-y-4">
            <h3 className="text-lg font-semibold text-gray-900 dark:text-white">
              Generate Reports
            </h3>
            <p className="text-gray-600 dark:text-gray-400">
              Generate comprehensive reports on employee statistics and
demographics.
            </p>
            <button
              onClick={generateReport}
              className="px-6 py-2 bg-purple-600 text-white rounded-md hover:bg-
purple-700 flex items-center"
            >
              <FaDownload className="mr-2" />
              Generate Employee Report
            </button>
          </div>
        )}
      </div>

      {/* Filters */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <div className="flex flex-wrap gap-4 items-center">
          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
```

```
gray-300 mb-1">
              Filter by Status
            </label>
            <select
              value={filterStatus}
              onChange={(e) => setFilterStatus(e.target.value)}
              className="px-3 py-2 border border-gray-300 dark:border-gray-600
rounded-md dark:bg-gray-700 dark:text-white"
            >
              <option value="all">All Status</option>
              <option value="ACTIVE">Active</option>
              <option value="INACTIVE">Inactive</option>
              <option value="TERMINATED">Terminated</option>
            </select>
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              Filter by Department
            </label>
            <select
              value={filterDepartment}
              onChange={(e) => setFilterDepartment(e.target.value)}
              className="px-3 py-2 border border-gray-300 dark:border-gray-600
rounded-md dark:bg-gray-700 dark:text-white"
            >
              <option value="all">All Departments</option>
              {departments.map(dept => (
                <option key={dept._id} value={dept._id}>{dept.name}</option>
              ))}
            </select>
          </div>

          <div className="flex items-end">
            <button
              onClick={handleSelectAll}
              className="px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-
blue-700"
            >
              {selectedEmployees.length === filteredEmployees.length ? 'Deselect
All' : 'Select All'}
            </button>
          </div>
        </div>
      </div>

      {/* Employee List */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <div className="overflow-x-auto">
          <table className="min-w-full divide-y divide-gray-200 dark:divide-
gray-700">
            <thead className="bg-gray-50 dark:bg-slate-800">
              <tr>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  <input
                    type="checkbox"
                    checked={selectedEmployees.length ===
```

```
filteredEmployees.length && filteredEmployees.length > 0}
                    onChange={handleSelectAll}
                    className="h-4 w-4 text-blue-600"
                  />
                </th>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Employee
                </th>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Department
                </th>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Role
                </th>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Status
                </th>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Salary
                </th>
              </tr>
            </thead>
            <tbody className="bg-white dark:bg-slate-900 divide-y divide-gray-200
dark:divide-gray-700">
              {filteredEmployees.map((employee) => (
                <tr key={employee._id} className="hover:bg-gray-50 dark:hover:bg-
slate-800">
                  <td className="px-6 py-4 whitespace-nowrap">
                    <input
                      type="checkbox"
                      checked={selectedEmployees.includes(employee._id)}
                      onChange={() => handleSelectEmployee(employee._id)}
                      className="h-4 w-4 text-blue-600"
                    />
                  </td>
                  <td className="px-6 py-4 whitespace-nowrap">
                    <div className="flex items-center">
                      <div className="w-10 h-10 bg-blue-100 dark:bg-blue-900
rounded-full flex items-center justify-center mr-3">
                        <span className="text-blue-600 dark:text-blue-400 font-
semibold text-sm">
                          {employee.fullName ? employee.fullName[0] : 'E'}
                        </span>
                      </div>
                      <div>
                        <div className="text-sm font-medium text-gray-900
dark:text-white">
                          {employee.fullName || 'Unknown'}
                        </div>
                        <div className="text-sm text-gray-500 dark:text-gray-400">
                          {employee.email || 'No email'}
                        </div>
                      </div>
                    </div>
```

```
                  </td>
                  <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                    {employee.department?.name || 'No Department'}
                  </td>
                  <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                    {employee.role?.name || 'No Role'}
                  </td>
                  <td className="px-6 py-4 whitespace-nowrap">
                    <span className={`px-2 py-1 rounded-full text-xs font-medium
${
                      employee.status === 'ACTIVE'
                        ? 'bg-green-100 text-green-800 dark:bg-green-900
dark:text-green-200'
                        : employee.status === 'INACTIVE'
                        ? 'bg-yellow-100 text-yellow-800 dark:bg-yellow-900
dark:text-yellow-200'
                        : 'bg-red-100 text-red-800 dark:bg-red-900 dark:text-
red-200'
                    }`}>
                      {employee.status || 'Unknown'}
                    </span>
                  </td>
                  <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                    {employee.salary ? `$${employee.salary.toLocaleString()}` :
'Not set'}
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </div>
    </div>
  );
};

export default BulkOperations;
```

src/components/Employee/EditEmployeeDialog.jsx

```
import React, { useState, useEffect } from 'react';
import employeeAPI from '../../services/employeeAPI';
import { FaUser, FaBriefcase, FaGraduationCap, FaFileUpload, FaSpinner } from
'react-icons/fa';

const EditEmployeeDialog = ({ employeeId, isOpen, onOpenChange,
onEmployeeUpdated, departments, roles }) => {
  const [employee, setEmployee] = useState(null);
  const [loading, setLoading] = useState(false);
  const [saving, setSaving] = useState(false);
  const [activeTab, setActiveTab] = useState('personal');
  const [formData, setFormData] = useState({
    fullName: '',
    email: '',
    phoneNumber: '',
```

```
    whatsappNumber: '',
    linkedInUrl: '',
    currentAddress: '',
    permanentAddress: '',
    dateOfBirth: '',
    joiningDate: '',
    salary: '',
    status: 'ACTIVE',
    department: '',
    role: '',
    collegeName: '',
    internshipDuration: ''
  });
  const [files, setFiles] = useState({});
  const [errors, setErrors] = useState({});

  useEffect(() => {
    if (isOpen && employeeId) {
      fetchEmployeeDetails();
    }
  }, [isOpen, employeeId]);

  const fetchEmployeeDetails = async () => {
    try {
      setLoading(true);
      const response = await employeeAPI.getById(employeeId);
      if (response.data.success) {
        const emp = response.data.data;
        setEmployee(emp);
        setFormData({
          fullName: emp.fullName || '',
          email: emp.email || '',
          phoneNumber: emp.phoneNumber || '',
          whatsappNumber: emp.whatsappNumber || '',
          linkedInUrl: emp.linkedInUrl || '',
          currentAddress: emp.currentAddress || '',
          permanentAddress: emp.permanentAddress || '',
          dateOfBirth: emp.dateOfBirth ? new
Date(emp.dateOfBirth).toISOString().split('T')[0] : '',
          joiningDate: emp.joiningDate ? new
Date(emp.joiningDate).toISOString().split('T')[0] : '',
          salary: emp.salary || '',
          status: emp.status || 'ACTIVE',
          department: emp.department?._id || emp.department || '',
          role: emp.role?._id || emp.role || '',
          collegeName: emp.collegeName || '',
          internshipDuration: emp.internshipDuration || ''
        });
      }
    } catch (err) {
      console.error('Error fetching employee details:', err);
    } finally {
      setLoading(false);
    }
  };

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({
```

```
    ...prev,
    [name]: value
  }));
  // Clear error when user starts typing
  if (errors[name]) {
    setErrors(prev => ({
      ...prev,
      [name]: ''
    }));
  }
};

const handleFileChange = (e) => {
  const { name, files: fileList } = e.target;
  if (fileList && fileList[0]) {
    setFiles(prev => ({
      ...prev,
      [name]: fileList[0]
    }));
  }
};

const validateForm = () => {
  const newErrors = {};

  if (!formData.fullName.trim()) {
    newErrors.fullName = 'Full name is required';
  }

  if (!formData.email.trim()) {
    newErrors.email = 'Email is required';
  } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
    newErrors.email = 'Email is invalid';
  }

  if (!formData.department) {
    newErrors.department = 'Department is required';
  }

  if (!formData.role) {
    newErrors.role = 'Role is required';
  }

  if (formData.salary && isNaN(formData.salary)) {
    newErrors.salary = 'Salary must be a number';
  }

  if (formData.internshipDuration && isNaN(formData.internshipDuration)) {
    newErrors.internshipDuration = 'Internship duration must be a number';
  }

  setErrors(newErrors);
  return Object.keys(newErrors).length === 0;
};

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

  if (!validateForm()) {
```

```
      return;
    }

    try {
      setSaving(true);

      const formDataToSend = new FormData();

      // Add employee data
      formDataToSend.append('employee', JSON.stringify(formData));

      // Add files
      Object.keys(files).forEach(key => {
        if (files[key]) {
          formDataToSend.append(key, files[key]);
        }
      });

      const response = await employeeAPI.update(employeeId, formDataToSend);

      if (response.data.success) {
        onEmployeeUpdated();
        onOpenChange(false);
      } else {
        alert('Failed to update employee. Please try again.');
      }
    } catch (error) {
      console.error('Error updating employee:', error);
      alert('Failed to update employee. Please try again.');
    } finally {
      setSaving(false);
    }
  };

  const tabs = [
    { id: 'personal', label: 'Personal Info', icon: FaUser },
    { id: 'professional', label: 'Professional Info', icon: FaBriefcase },
    { id: 'education', label: 'Education', icon: FaGraduationCap },
    { id: 'documents', label: 'Documents', icon: FaFileUpload }
  ];

  const documentFields = [
    { key: 'photograph', label: 'Photograph', accept: 'image/*' },
    { key: 'resume', label: 'Resume', accept: '.pdf,.doc,.docx' },
    { key: 'tenthMarksheet', label: '10th Marksheet', accept:
'.pdf,.jpg,.jpeg,.png' },
    { key: 'twelfthMarksheet', label: '12th Marksheet', accept:
'.pdf,.jpg,.jpeg,.png' },
    { key: 'bachelorDegree', label: 'Bachelor Degree', accept:
'.pdf,.jpg,.jpeg,.png' },
    { key: 'postgraduateDegree', label: 'Postgraduate Degree', accept:
'.pdf,.jpg,.jpeg,.png' },
    { key: 'aadharCard', label: 'Aadhar Card', accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'panCard', label: 'PAN Card', accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'pcc', label: 'PCC', accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'offerLetter', label: 'Offer Letter', accept: '.pdf,.doc,.docx' }
  ];

  if (!isOpen) return null;
```

```
  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
      <div className="bg-white dark:bg-gray-800 rounded-lg shadow-xl max-w-4xl w-
full mx-4 max-h-[90vh] overflow-y-auto">
        <div className="p-6">
          <div className="flex justify-between items-center mb-6">
            <h2 className="text-xl font-bold text-gray-900 dark:text-white">Edit
Employee</h2>
            <button
              onClick={() => onOpenChange(false)}
              className="text-gray-400 hover:text-gray-600 dark:hover:text-
gray-300"
            >
              <svg className="w-6 h-6" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M6 18L18 6M6 6l12 12" />
              </svg>
            </button>
          </div>

          {loading ? (
            <div className="flex justify-center items-center py-8">
              <FaSpinner className="animate-spin text-2xl text-blue-600" />
              <span className="ml-2">Loading employee details...</span>
            </div>
          ) : (
            <>
              {/* Tab Navigation */}
              <div className="border-b border-gray-200 dark:border-gray-700 mb-6">
                <nav className="flex space-x-8">
                  {tabs.map((tab) => (
                    <button
                      key={tab.id}
                      onClick={() => setActiveTab(tab.id)}
                      className={`${
                        activeTab === tab.id
                          ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                          : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
                      } whitespace-nowrap py-2 px-1 border-b-2 font-medium text-
sm flex items-center`}
                    >
                      <tab.icon className="mr-2 h-4 w-4" />
                      {tab.label}
                    </button>
                  ))}
                </nav>
              </div>

              <form onSubmit={handleSubmit}>
                {/* Personal Information Tab */}
                {activeTab === 'personal' && (
                  <div className="space-y-4">
                    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                      <div>
                        <label className="block text-sm font-medium text-gray-700
```

```
dark:text-gray-300 mb-1">
                          Full Name *
                        </label>
                        <input
                          type="text"
                          name="fullName"
                          value={formData.fullName}
                          onChange={handleInputChange}
                          className={`w-full px-3 py-2 border rounded-md dark:bg-
gray-700 dark:border-gray-600 dark:text-white ${
                            errors.fullName ? 'border-red-500' : 'border-gray-300'
                          }`}
                        />
                        {errors.fullName && <p className="text-red-500 text-xs
mt-1">{errors.fullName}</p>}
                      </div>

                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          Email *
                        </label>
                        <input
                          type="email"
                          name="email"
                          value={formData.email}
                          onChange={handleInputChange}
                          className={`w-full px-3 py-2 border rounded-md dark:bg-
gray-700 dark:border-gray-600 dark:text-white ${
                            errors.email ? 'border-red-500' : 'border-gray-300'
                          }`}
                        />
                        {errors.email && <p className="text-red-500 text-xs
mt-1">{errors.email}</p>}
                      </div>

                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          Phone Number
                        </label>
                        <input
                          type="tel"
                          name="phoneNumber"
                          value={formData.phoneNumber}
                          onChange={handleInputChange}
                          className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                        />
                      </div>

                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          WhatsApp Number
                        </label>
                        <input
                          type="tel"
                          name="whatsappNumber"
```

```
                        value={formData.whatsappNumber}
                        onChange={handleInputChange}
                        className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                      />
                    </div>

                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                        LinkedIn URL
                      </label>
                      <input
                        type="url"
                        name="linkedInUrl"
                        value={formData.linkedInUrl}
                        onChange={handleInputChange}
                        className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                      />
                    </div>

                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                        Date of Birth
                      </label>
                      <input
                        type="date"
                        name="dateOfBirth"
                        value={formData.dateOfBirth}
                        onChange={handleInputChange}
                        className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                      />
                    </div>
                  </div>

                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                      Current Address
                    </label>
                    <textarea
                      name="currentAddress"
                      value={formData.currentAddress}
                      onChange={handleInputChange}
                      rows="3"
                      className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                    />
                  </div>

                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                      Permanent Address
                    </label>
                    <textarea
```

```
                        name="permanentAddress"
                        value={formData.permanentAddress}
                        onChange={handleInputChange}
                        rows="3"
                        className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                      />
                    </div>
                  </div>
                )}

                {/* Professional Information Tab */}
                {activeTab === 'professional' && (
                  <div className="space-y-4">
                    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          Department *
                        </label>
                        <select
                          name="department"
                          value={formData.department}
                          onChange={handleInputChange}
                          className={`w-full px-3 py-2 border rounded-md dark:bg-
gray-700 dark:border-gray-600 dark:text-white ${
                            errors.department ? 'border-red-500' : 'border-
gray-300'
                          }`}
                        >
                          <option value="">Select Department</option>
                          {departments.map(dept => (
                            <option key={dept._id} value={dept._id}>{dept.name}</
option>
                          ))}
                        </select>
                        {errors.department && <p className="text-red-500 text-xs
mt-1">{errors.department}</p>}
                      </div>

                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          Role *
                        </label>
                        <select
                          name="role"
                          value={formData.role}
                          onChange={handleInputChange}
                          className={`w-full px-3 py-2 border rounded-md dark:bg-
gray-700 dark:border-gray-600 dark:text-white ${
                            errors.role ? 'border-red-500' : 'border-gray-300'
                          }`}
                        >
                          <option value="">Select Role</option>
                          {roles.map(role => (
                            <option key={role._id} value={role._id}>{role.name}</
option>
                          ))}
```

```
                        </select>
                        {errors.role && <p className="text-red-500 text-xs
mt-1">{errors.role}</p>}
                      </div>

                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          Joining Date
                        </label>
                        <input
                          type="date"
                          name="joiningDate"
                          value={formData.joiningDate}
                          onChange={handleInputChange}
                          className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                        />
                      </div>

                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          Salary
                        </label>
                        <input
                          type="number"
                          name="salary"
                          value={formData.salary}
                          onChange={handleInputChange}
                          className={`w-full px-3 py-2 border rounded-md dark:bg-
gray-700 dark:border-gray-600 dark:text-white ${
                            errors.salary ? 'border-red-500' : 'border-gray-300'
                          }`}
                        />
                        {errors.salary && <p className="text-red-500 text-xs
mt-1">{errors.salary}</p>}
                      </div>

                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          Status
                        </label>
                        <select
                          name="status"
                          value={formData.status}
                          onChange={handleInputChange}
                          className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                        >
                          <option value="ACTIVE">Active</option>
                          <option value="INACTIVE">Inactive</option>
                          <option value="TERMINATED">Terminated</option>
                        </select>
                      </div>
                    </div>
                  </div>
                )}
```

```
                {/* Education Tab */}
                {activeTab === 'education' && (
                  <div className="space-y-4">
                    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          College Name
                        </label>
                        <input
                          type="text"
                          name="collegeName"
                          value={formData.collegeName}
                          onChange={handleInputChange}
                          className="w-full px-3 py-2 border border-gray-300
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                        />
                      </div>

                      <div>
                        <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                          Internship Duration (months)
                        </label>
                        <input
                          type="number"
                          name="internshipDuration"
                          value={formData.internshipDuration}
                          onChange={handleInputChange}
                          className={`w-full px-3 py-2 border rounded-md dark:bg-
gray-700 dark:border-gray-600 dark:text-white ${
                            errors.internshipDuration ? 'border-red-500' :
'border-gray-300'
                          }`}
                        />
                        {errors.internshipDuration && <p className="text-red-500
text-xs mt-1">{errors.internshipDuration}</p>}
                      </div>
                    </div>
                  </div>
                )}

                {/* Documents Tab */}
                {activeTab === 'documents' && (
                  <div className="space-y-4">
                    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                      {documentFields.map((field) => (
                        <div key={field.key}>
                          <label className="block text-sm font-medium text-
gray-700 dark:text-gray-300 mb-1">
                            {field.label}
                          </label>
                          <input
                            type="file"
                            name={field.key}
                            accept={field.accept}
                            onChange={handleFileChange}
                            className="w-full px-3 py-2 border border-gray-300
```

```
rounded-md dark:bg-gray-700 dark:border-gray-600 dark:text-white"
                          />
                          {employee && employee[field.key] && (
                            <p className="text-xs text-gray-500 mt-1">
                              Current: {employee[field.key].split('/').pop()}
                            </p>
                          )}
                        </div>
                      ))}
                    </div>
                  </div>
                )}

                <div className="flex justify-end space-x-3 pt-6 border-t">
                  <button
                    type="button"
                    onClick={() => onOpenChange(false)}
                    className="px-4 py-2 bg-gray-100 dark:bg-gray-700 hover:bg-
gray-200 dark:hover:bg-gray-600 text-gray-700 dark:text-gray-300 rounded-lg
transition-colors"
                  >
                    Cancel
                  </button>
                  <button
                    type="submit"
                    disabled={saving}
                    className="px-4 py-2 bg-blue-600 hover:bg-blue-700 text-white
rounded-lg transition-colors disabled:opacity-50 disabled:cursor-not-allowed flex
items-center"
                  >
                    {saving ? (
                      <>
                        <FaSpinner className="animate-spin mr-2" />
                        Saving...
                      </>
                    ) : (
                      'Save Changes'
                    )}
                  </button>
                </div>
              </form>
            </>
          )}
        </div>
      </div>
    </div>
  );
};

export default EditEmployeeDialog;
```

src/components/Employee/EmployeeDetailsDialog.jsx

```
import React, { useState, useEffect } from 'react';
import { FaUser, FaEnvelope, FaPhone, FaMapMarkerAlt, FaCalendarAlt,
FaDollarSign, FaLinkedin, FaWhatsapp, FaBuilding, FaUserTie, FaFileAlt,
FaDownload, FaEye, FaUpload } from 'react-icons/fa';
import employeeAPI from '../../services/employeeAPI';
```

```
import { toast } from 'react-toastify';

const EmployeeDetailsDialog = ({ employeeId, isOpen, onOpenChange }) => {
  const [employee, setEmployee] = useState(null);
  const [isLoading, setIsLoading] = useState(false);

  useEffect(() => {
    if (isOpen && employeeId) {
      fetchEmployeeDetails();
    }
  }, [isOpen, employeeId]);

  const fetchEmployeeDetails = async () => {
    try {
      setIsLoading(true);
      const response = await employeeAPI.getById(employeeId);
      if (response.data.success) {
        setEmployee(response.data.data);
      }
    } catch (err) {
      console.error('Error fetching employee details:', err);
    } finally {
      setIsLoading(false);
    }
  };

  // Document types with their display names
  const documentTypes = {
    photograph: 'Photograph',
    tenthMarksheet: '10th Marksheet',
    twelfthMarksheet: '12th Marksheet',
    bachelorDegree: 'Bachelor Degree',
    postgraduateDegree: 'Postgraduate Degree',
    aadharCard: 'Aadhar Card',
    panCard: 'PAN Card',
    pcc: 'Police Clearance Certificate',
    resume: 'Resume',
    offerLetter: 'Offer Letter'
  };

  const downloadDocument = async (documentType) => {
    try {
      const documents = employee.documents || {};
      const docInfo = documents[documentType];

      if (!docInfo || (!docInfo.filename && !docInfo.path)) {
        toast.error('Document not found');
        return;
      }

      // Get the document path from either filename or path
      const documentPath = docInfo.path ? docInfo.path.split('/').pop() :
docInfo.filename;

      // Use the API endpoint for downloading documents
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const baseUrl = isDevelopment
        ? 'http://localhost:8080/api'
```

```
        : (import.meta.env.VITE_API_URL || 'https://crm-backend-o36v.onrender.com/
api');
      const fileUrl = `${baseUrl}/employees/documents/${documentPath}`;
      const token = localStorage.getItem('token');

      // Fetch the file with authentication
      const response = await fetch(fileUrl, {
        headers: {
          'Authorization': `Bearer ${token}`
        }
      });

      if (!response.ok) {
        const errorText = await response.text();
        console.error('Failed to download document:', errorText);
        throw new Error(`Failed to download document: ${response.status}`);
      }

      // Get the file blob
      const blob = await response.blob();

      // Create download link
      const link = document.createElement('a');
      link.href = window.URL.createObjectURL(blob);
      link.download = `${employee.fullName || employee.email}
_${documentTypes[documentType]}_${docInfo.filename}`;
      document.body.appendChild(link);
      link.click();
      document.body.removeChild(link);

      // Clean up the blob URL
      window.URL.revokeObjectURL(link.href);

      toast.success(`Downloaded ${documentTypes[documentType]}`);
    } catch (err) {
      console.error('Error downloading document:', err);
      toast.error('Failed to download document');
    }
  };

  const viewDocument = (documentType) => {
    try {
      const documents = employee.documents || {};
      const docInfo = documents[documentType];

      if (!docInfo || (!docInfo.filename && !docInfo.path)) {
        toast.error('Document not found');
        return;
      }

      // Get the document path from either filename or path
      const documentPath = docInfo.path ? docInfo.path.split('/').pop() :
docInfo.filename;

      // Use the API endpoint for viewing documents
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const baseUrl = isDevelopment
        ? 'http://localhost:8080/api'
```

```
        : (import.meta.env.VITE_API_URL || 'https://crm-backend-o36v.onrender.com/
api');
      const fileUrl = `${baseUrl}/employees/documents/${documentPath}`;
      const token = localStorage.getItem('token');

      // Create a new window/tab with the authenticated URL
      const newWindow = window.open();

      // Fetch the file with authentication
      fetch(fileUrl, {
        headers: {
          'Authorization': `Bearer ${token}`
        }
      })
      .then(response => {
        if (!response.ok) {
          return response.text().then(errorText => {
            console.error('Failed to load document:', errorText);
            throw new Error(`Failed to load document: ${response.status}`);
          });
        }
        return response.blob();
      })
      .then(blob => {
        const blobUrl = window.URL.createObjectURL(blob);
        newWindow.location.href = blobUrl;
      })
      .catch(err => {
        console.error('Error viewing document:', err);
        toast.error('Failed to view document');
        newWindow.close();
      });
    } catch (err) {
      console.error('Error viewing document:', err);
      toast.error('Failed to view document');
    }
  };

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50 p-4">
      <div className="bg-white dark:bg-gray-800 rounded-lg shadow-xl max-w-4xl w-
full max-h-[90vh] overflow-y-auto">
        <div className="p-6">
          <div className="flex justify-between items-center mb-6">
            <h2 className="text-2xl font-bold text-gray-900 dark:text-
white">Employee Details</h2>
            <button
              onClick={() => onOpenChange(false)}
              className="text-gray-400 hover:text-gray-600 dark:hover:text-
gray-300 p-2"
            >
              <svg className="w-6 h-6" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M6 18L18 6M6 6l12 12" />
              </svg>
```

```
            </button>
          </div>

          {isLoading ? (
            <div className="flex items-center justify-center h-32">
              <div className="animate-spin rounded-full h-8 w-8 border-t-2 border-
b-2 border-blue-500"></div>
            </div>
          ) : employee ? (
            <div className="space-y-6">
              {/* Employee Header */}
              <div className="text-center bg-gradient-to-r from-blue-50 to-
indigo-50 dark:from-blue-900/20 dark:to-indigo-900/20 rounded-lg p-6">
                <div className="w-32 h-32 bg-blue-100 dark:bg-blue-900 rounded-
full flex items-center justify-center mx-auto mb-4">
                  {employee.photograph ? (
                    <img
                      src={`/api/uploads/${employee.photograph}`}
                      alt={employee.fullName}
                      className="w-32 h-32 rounded-full object-cover"
                    />
                  ) : (
                    <span className="text-blue-600 dark:text-blue-400 font-
semibold text-4xl">
                      {employee.fullName ? employee.fullName[0] : 'E'}
                    </span>
                  )}
                </div>
                <h3 className="text-2xl font-bold text-gray-900 dark:text-white">
                  {employee?.fullName || 'N/A'}
                </h3>
                <p className="text-lg text-gray-600 dark:text-gray-400">
                  {employee?.role ? (typeof employee.role === 'object' ?
employee.role?.name : employee.role) : 'N/A'}
                </p>
                <div className="flex justify-center items-center space-x-4 mt-3">
                  <span className={`px-3 py-1 text-sm rounded-full ${
                    employee?.status === 'ACTIVE'
                      ? 'bg-green-100 text-green-800 dark:bg-green-900 dark:text-
green-200'
                      : 'bg-gray-100 text-gray-800 dark:bg-gray-700 dark:text-
gray-300'
                  }`}>
                    {employee?.status || 'N/A'}
                  </span>
                  <span className="text-sm text-gray-500 dark:text-gray-400">
                    Employee ID: {employee?.employeeId || 'N/A'}
                  </span>
                </div>
              </div>

              {/* Employee Information Grid */}
              <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">

                {/* Personal Information */}
                <div className="bg-gray-50 dark:bg-gray-900 rounded-lg p-6">
                  <h4 className="text-lg font-semibold text-gray-900 dark:text-
white mb-4 flex items-center">
                    <FaUser className="mr-2 text-blue-600" />
```

```
                    Personal Information
                  </h4>
                  <div className="space-y-3">
                    <div className="flex items-center">
                      <FaEnvelope className="w-4 h-4 text-gray-400 mr-3" />
                      <div>
                        <p className="text-sm text-gray-600 dark:text-
gray-400">Email</p>
                        <p className="text-gray-900 dark:text-
white">{employee?.email || 'N/A'}</p>
                      </div>
                    </div>
                    <div className="flex items-center">
                      <FaPhone className="w-4 h-4 text-gray-400 mr-3" />
                      <div>
                        <p className="text-sm text-gray-600 dark:text-
gray-400">Phone</p>
                        <p className="text-gray-900 dark:text-
white">{employee?.phoneNumber || 'N/A'}</p>
                      </div>
                    </div>
                    {employee?.whatsappNumber && (
                      <div className="flex items-center">
                        <FaWhatsapp className="w-4 h-4 text-gray-400 mr-3" />
                        <div>
                          <p className="text-sm text-gray-600 dark:text-
gray-400">WhatsApp</p>
                          <p className="text-gray-900 dark:text-
white">{employee.whatsappNumber}</p>
                        </div>
                      </div>
                    )}
                    {employee?.linkedInUrl && (
                      <div className="flex items-center">
                        <FaLinkedin className="w-4 h-4 text-gray-400 mr-3" />
                        <div>
                          <p className="text-sm text-gray-600 dark:text-
gray-400">LinkedIn</p>
                          <a href={employee.linkedInUrl} target="_blank"
rel="noopener noreferrer" className="text-blue-600 hover:underline">
                            View Profile
                          </a>
                        </div>
                      </div>
                    )}
                    {employee?.dateOfBirth && (
                      <div className="flex items-center">
                        <FaCalendarAlt className="w-4 h-4 text-gray-400 mr-3" />
                        <div>
                          <p className="text-sm text-gray-600 dark:text-
gray-400">Date of Birth</p>
                          <p className="text-gray-900 dark:text-white">{new
Date(employee.dateOfBirth).toLocaleDateString()}</p>
                        </div>
                      </div>
                    )}
                  </div>
                </div>
```

```
                {/* Professional Information */}
                <div className="bg-gray-50 dark:bg-gray-900 rounded-lg p-6">
                  <h4 className="text-lg font-semibold text-gray-900 dark:text-
white mb-4 flex items-center">
                    <FaUserTie className="mr-2 text-blue-600" />
                    Professional Information
                  </h4>
                  <div className="space-y-3">
                    <div className="flex items-center">
                      <FaBuilding className="w-4 h-4 text-gray-400 mr-3" />
                      <div>
                        <p className="text-sm text-gray-600 dark:text-
gray-400">Department</p>
                        <p className="text-gray-900 dark:text-white">
                          {employee?.department ? (typeof employee.department ===
'object' ? employee.department?.name : employee.department) : 'N/A'}
                        </p>
                      </div>
                    </div>
                    <div className="flex items-center">
                      <FaCalendarAlt className="w-4 h-4 text-gray-400 mr-3" />
                      <div>
                        <p className="text-sm text-gray-600 dark:text-
gray-400">Joining Date</p>
                        <p className="text-gray-900 dark:text-white">
                          {employee?.joiningDate ? new
Date(employee.joiningDate).toLocaleDateString() : 'N/A'}
                        </p>
                      </div>
                    </div>
                    {employee?.salary && (
                      <div className="flex items-center">
                        <FaDollarSign className="w-4 h-4 text-gray-400 mr-3" />
                        <div>
                          <p className="text-sm text-gray-600 dark:text-
gray-400">Salary</p>
                          <p className="text-gray-900 dark:text-white font-
semibold">
                             ₹{employee.salary.toLocaleString()}
                          </p>
                        </div>
                      </div>
                    )}
                    {employee?.collegeName && (
                      <div className="flex items-center">
                        <FaFileAlt className="w-4 h-4 text-gray-400 mr-3" />
                        <div>
                          <p className="text-sm text-gray-600 dark:text-
gray-400">College</p>
                          <p className="text-gray-900 dark:text-
white">{employee.collegeName}</p>
                        </div>
                      </div>
                    )}
                    {employee?.internshipDuration && (
                      <div className="flex items-center">
                        <FaCalendarAlt className="w-4 h-4 text-gray-400 mr-3" />
                        <div>
                          <p className="text-sm text-gray-600 dark:text-
```

```
gray-400">Internship Duration</p>
                          <p className="text-gray-900 dark:text-
white">{employee.internshipDuration} months</p>
                        </div>
                      </div>
                    )}
                  </div>
                </div>

                {/* Address Information */}
                {(employee?.currentAddress || employee?.permanentAddress) && (
                  <div className="bg-gray-50 dark:bg-gray-900 rounded-lg p-6">
                    <h4 className="text-lg font-semibold text-gray-900 dark:text-
white mb-4 flex items-center">
                      <FaMapMarkerAlt className="mr-2 text-blue-600" />
                      Address Information
                    </h4>
                    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                      {employee?.currentAddress && (
                        <div>
                          <p className="text-sm text-gray-600 dark:text-gray-400
mb-1">Current Address</p>
                          <p className="text-gray-900 dark:text-
white">{employee.currentAddress}</p>
                        </div>
                      )}
                      {employee?.permanentAddress && (
                        <div>
                          <p className="text-sm text-gray-600 dark:text-gray-400
mb-1">Permanent Address</p>
                          <p className="text-gray-900 dark:text-
white">{employee.permanentAddress}</p>
                        </div>
                      )}
                    </div>
                  </div>
                )}

                {/* Document Status */}
                {employee?.documents && Object.keys(employee.documents).length > 0
&& (
                  <div className="bg-gray-50 dark:bg-gray-900 rounded-lg p-6">
                    <h4 className="text-lg font-semibold text-gray-900 dark:text-
white mb-4 flex items-center">
                      <FaFileAlt className="mr-2 text-blue-600" />
                      Document Status
                    </h4>
                    <div className="space-y-3">
                      {Object.entries(employee.documents).map(([key, value]) => {
                        const hasDocument = value && value.filename;
                        const displayName = documentTypes[key] || key.replace(/([A-
Z])/g, ' $1').trim();

                        return (
                          <div key={key} className="flex items-center justify-
between p-3 bg-white dark:bg-gray-800 rounded-lg border border-gray-200
dark:border-gray-700">
                            <div className="flex items-center">
```

```
                            <span className={`w-3 h-3 rounded-full mr-3
${hasDocument ? 'bg-green-500' : 'bg-red-500'}`}></span>
                            <div>
                              <span className="text-sm font-medium text-gray-900
dark:text-white">
                                {displayName}
                              </span>
                              <p className="text-xs text-gray-500 dark:text-
gray-400">
                                {hasDocument ? `Uploaded: ${value.originalName ||
value.filename}` : 'Not uploaded'}
                              </p>
                            </div>
                          </div>

                          <div className="flex items-center space-x-2">
                            {hasDocument && (
                              <>
                                <button
                                  onClick={() => viewDocument(key)}
                                  className="p-2 text-blue-600 hover:bg-blue-50
dark:hover:bg-blue-900/20 rounded-full transition-colors"
                                  title="View Document"
                                >
                                  <FaEye className="w-4 h-4" />
                                </button>
                                <button
                                  onClick={() => downloadDocument(key)}
                                  className="p-2 text-green-600 hover:bg-green-50
dark:hover:bg-green-900/20 rounded-full transition-colors"
                                  title="Download Document"
                                >
                                  <FaDownload className="w-4 h-4" />
                                </button>
                              </>
                            )}
                          </div>
                        </div>
                      );
                    })}
                  </div>
                </div>
              )}
            </div>
          ) : (
            <div className="text-center py-8">
              <p className="text-gray-500 dark:text-gray-400">Employee not found</
p>
            </div>
          )}

          <div className="flex justify-end pt-6 border-t border-gray-200
dark:border-gray-700">
            <button
              onClick={() => onOpenChange(false)}
              className="px-6 py-2 bg-gray-100 dark:bg-gray-700 hover:bg-gray-200
dark:hover:bg-gray-600 text-gray-700 dark:text-gray-300 rounded-lg transition-
colors"
            >
```

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

export default EmployeeDetailsDialog;
```

src/components/Employee/EmployeeDirectory.jsx

```
import React, { useState, useEffect } from 'react';
import { FaSearch, FaUser, FaPhone, FaEnvelope, FaLinkedin, FaFilter, FaUsers }
from 'react-icons/fa';
import employeeAPI from '../../services/employeeAPI';

const EmployeeDirectory = () => {
  const [employees, setEmployees] = useState([]);
  const [filteredEmployees, setFilteredEmployees] = useState([]);
  const [departments, setDepartments] = useState([]);
  const [roles, setRoles] = useState([]);
  const [loading, setLoading] = useState(true);
  const [searchTerm, setSearchTerm] = useState('');
  const [selectedDepartment, setSelectedDepartment] = useState('all');
  const [selectedRole, setSelectedRole] = useState('all');
  const [viewMode, setViewMode] = useState('grid'); // grid or list

  useEffect(() => {
    fetchData();
  }, []);

  useEffect(() => {
    filterEmployees();
  }, [employees, searchTerm, selectedDepartment, selectedRole]);

  const fetchData = async () => {
    try {
      setLoading(true);
      const [employeesRes, departmentsRes, rolesRes] = await Promise.all([
        employeeAPI.getAll(),
        employeeAPI.getDepartments(),
        employeeAPI.getRoles()
      ]);

      if (employeesRes.data.success) {
        setEmployees(employeesRes.data.data);
      }
      if (departmentsRes.data.success) {
        setDepartments(departmentsRes.data.data);
      }
      if (rolesRes.data.success) {
        setRoles(rolesRes.data.data);
      }
    } catch (error) {
      console.error('Error fetching directory data:', error);
    } finally {
```

```
      setLoading(false);
    }
  };

  const filterEmployees = () => {
    let filtered = employees.filter(emp => emp.status === 'ACTIVE');

    // Search filter
    if (searchTerm) {
      filtered = filtered.filter(emp =>
        emp.fullName?.toLowerCase().includes(searchTerm.toLowerCase()) ||
        emp.email?.toLowerCase().includes(searchTerm.toLowerCase()) ||
        emp.phoneNumber?.includes(searchTerm)
      );
    }

    // Department filter
    if (selectedDepartment !== 'all') {
      filtered = filtered.filter(emp => {
        const deptId = emp.department?._id || emp.department;
        return deptId === selectedDepartment;
      });
    }

    // Role filter
    if (selectedRole !== 'all') {
      filtered = filtered.filter(emp => {
        const roleId = emp.role?._id || emp.role;
        return roleId === selectedRole;
      });
    }

    setFilteredEmployees(filtered);
  };

  const getEmployeeInitials = (name) => {
    if (!name) return 'E';
    return name.split(' ').map(n => n[0]).join('').toUpperCase().slice(0, 2);
  };

  const getDepartmentStats = () => {
    const stats = {};
    departments.forEach(dept => {
      const count = employees.filter(emp => {
        const deptId = emp.department?._id || emp.department;
        return deptId === dept._id && emp.status === 'ACTIVE';
      }).length;
      stats[dept.name] = count;
    });
    return stats;
  };

  if (loading) {
    return (
      <div className="flex justify-center items-center h-64">
        <div className="animate-spin rounded-full h-8 w-8 border-t-2 border-b-2
border-blue-500"></div>
        <span className="ml-2">Loading directory...</span>
      </div>
```

```
    );
  }

  const departmentStats = getDepartmentStats();

  return (
    <div className="space-y-6">
      {/* Header */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <div className="flex flex-col md:flex-row justify-between items-start
md:items-center gap-4">
          <div>
            <h2 className="text-xl font-bold text-gray-900 dark:text-white flex
items-center">
              <FaUsers className="mr-2 text-blue-600" />
              Employee Directory
            </h2>
            <p className="text-gray-600 dark:text-gray-400 mt-1">
              {filteredEmployees.length} of {employees.filter(e => e.status ===
'ACTIVE').length} employees
            </p>
          </div>

          <div className="flex items-center gap-2">
            <button
              onClick={() => setViewMode('grid')}
              className={`px-3 py-1 rounded ${viewMode === 'grid' ? 'bg-blue-600
text-white' : 'bg-gray-200 text-gray-700'}`}
            >
              Grid
            </button>
            <button
              onClick={() => setViewMode('list')}
              className={`px-3 py-1 rounded ${viewMode === 'list' ? 'bg-blue-600
text-white' : 'bg-gray-200 text-gray-700'}`}
            >
              List
            </button>
          </div>
        </div>
      </div>

      {/* Filters */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              Search
            </label>
            <div className="relative">
              <FaSearch className="absolute left-3 top-1/2 transform -translate-
y-1/2 text-gray-400" />
              <input
                type="text"
                placeholder="Search employees..."
                value={searchTerm}
                onChange={(e) => setSearchTerm(e.target.value)}
                className="w-full pl-10 pr-3 py-2 border border-gray-300
```

```
dark:border-gray-600 rounded-md dark:bg-gray-700 dark:text-white"
              />
            </div>
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              Department
            </label>
            <select
              value={selectedDepartment}
              onChange={(e) => setSelectedDepartment(e.target.value)}
              className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md dark:bg-gray-700 dark:text-white"
            >
              <option value="all">All Departments</option>
              {departments.map(dept => (
                <option key={dept._id} value={dept._id}>
                  {dept.name} ({departmentStats[dept.name] || 0})
                </option>
              ))}
            </select>
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              Role
            </label>
            <select
              value={selectedRole}
              onChange={(e) => setSelectedRole(e.target.value)}
              className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md dark:bg-gray-700 dark:text-white"
            >
              <option value="all">All Roles</option>
              {roles.map(role => (
                <option key={role._id} value={role._id}>{role.name}</option>
              ))}
            </select>
          </div>

          <div className="flex items-end">
            <button
              onClick={() => {
                setSearchTerm('');
                setSelectedDepartment('all');
                setSelectedRole('all');
              }}
              className="w-full px-3 py-2 bg-gray-600 text-white rounded-md
hover:bg-gray-700 transition-colors"
            >
              Clear Filters
            </button>
          </div>
        </div>
      </div>
```

```
      {/* Employee List/Grid */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
        {filteredEmployees.length === 0 ? (
          <div className="text-center py-12">
            <FaUser className="mx-auto text-4xl text-gray-400 mb-4" />
            <h3 className="text-lg font-medium text-gray-600 dark:text-gray-400
mb-2">
              No employees found
            </h3>
            <p className="text-gray-500 dark:text-gray-500">
              Try adjusting your search criteria or filters.
            </p>
          </div>
        ) : viewMode === 'grid' ? (
          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-
cols-4 gap-4">
            {filteredEmployees.map((employee) => (
              <div key={employee._id} className="border border-gray-200
dark:border-gray-700 rounded-lg p-4 hover:shadow-md transition-shadow">
                <div className="flex items-center space-x-3 mb-3">
                  <div className="w-12 h-12 bg-blue-100 dark:bg-blue-900 rounded-
full flex items-center justify-center">
                    {employee.photograph ? (
                      <img
                        src={`/api/uploads/${employee.photograph}`}
                        alt={employee.fullName}
                        className="w-12 h-12 rounded-full object-cover"
                      />
                    ) : (
                      <span className="text-blue-600 dark:text-blue-400 font-
semibold">
                        {getEmployeeInitials(employee.fullName)}
                      </span>
                    )}
                  </div>
                  <div className="flex-1 min-w-0">
                    <h3 className="text-sm font-semibold text-gray-900 dark:text-
white truncate">
                      {employee.fullName || 'Unknown'}
                    </h3>
                    <p className="text-xs text-gray-600 dark:text-gray-400
truncate">
                      {employee.role?.name || 'No Role'}
                    </p>
                  </div>
                </div>

                <div className="space-y-2 text-xs">
                  <div className="flex items-center text-gray-600 dark:text-
gray-400">
                    <FaEnvelope className="mr-2 w-3 h-3" />
                    <span className="truncate">{employee.email || 'No email'}</
span>
                  </div>

                  {employee.phoneNumber && (
                    <div className="flex items-center text-gray-600 dark:text-
gray-400">
                      <FaPhone className="mr-2 w-3 h-3" />
```

```
                    <span>{employee.phoneNumber}</span>
                  </div>
                )}

                {employee.linkedInUrl && (
                  <div className="flex items-center text-gray-600 dark:text-
gray-400">
                    <FaLinkedin className="mr-2 w-3 h-3" />
                    <a
                      href={employee.linkedInUrl}
                      target="_blank"
                      rel="noopener noreferrer"
                      className="text-blue-600 hover:underline truncate"
                    >
                      LinkedIn
                    </a>
                  </div>
                )}

                <div className="pt-2 border-t border-gray-200 dark:border-
gray-700">
                  <span className="text-xs text-gray-500 dark:text-gray-500">
                    {employee.department?.name || 'No Department'}
                  </span>
                </div>
              </div>
            </div>
          ))}
        </div>
      ) : (
        <div className="overflow-x-auto">
          <table className="min-w-full divide-y divide-gray-200 dark:divide-
gray-700">
            <thead className="bg-gray-50 dark:bg-slate-800">
              <tr>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Employee
                </th>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Department
                </th>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Role
                </th>
                <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                  Contact
                </th>
              </tr>
            </thead>
            <tbody className="bg-white dark:bg-slate-900 divide-y divide-
gray-200 dark:divide-gray-700">
              {filteredEmployees.map((employee) => (
                <tr key={employee._id} className="hover:bg-gray-50
dark:hover:bg-slate-800">
                  <td className="px-6 py-4 whitespace-nowrap">
```

```
                    <div className="flex items-center">
                      <div className="w-10 h-10 bg-blue-100 dark:bg-blue-900
rounded-full flex items-center justify-center mr-3">
                        {employee.photograph ? (
                          <img
                            src={`/api/uploads/${employee.photograph}`}
                            alt={employee.fullName}
                            className="w-10 h-10 rounded-full object-cover"
                          />
                        ) : (
                          <span className="text-blue-600 dark:text-blue-400
font-semibold text-sm">
                            {getEmployeeInitials(employee.fullName)}
                          </span>
                        )}
                      </div>
                      <div>
                        <div className="text-sm font-medium text-gray-900
dark:text-white">
                          {employee.fullName || 'Unknown'}
                        </div>
                        <div className="text-sm text-gray-500 dark:text-
gray-400">
                          {employee.email || 'No email'}
                        </div>
                      </div>
                    </div>
                  </td>
                  <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                    {employee.department?.name || 'No Department'}
                  </td>
                  <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                    {employee.role?.name || 'No Role'}
                  </td>
                  <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-400">
                    <div className="space-y-1">
                      {employee.phoneNumber && (
                        <div className="flex items-center">
                          <FaPhone className="mr-2 w-3 h-3" />
                          <span>{employee.phoneNumber}</span>
                        </div>
                      )}
                      {employee.linkedInUrl && (
                        <div className="flex items-center">
                          <FaLinkedin className="mr-2 w-3 h-3" />
                          <a
                            href={employee.linkedInUrl}
                            target="_blank"
                            rel="noopener noreferrer"
                            className="text-blue-600 hover:underline"
                          >
                            LinkedIn
                          </a>
                        </div>
                      )}
                    </div>
```

```
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      )}
    </div>
  </div>
  );
};

export default EmployeeDirectory;
```

src/components/Employee/EmployeeList.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../../context/AuthContext';
import employeeAPI from '../../services/employeeAPI';
import AddEmployeeDialog from './AddEmployeeDialog';
import EmployeeDetailsDialog from './EmployeeDetailsDialog';
import EditEmployeeDialog from './EditEmployeeDialog';

const EmployeeList = () => {
  const { user } = useAuth();
  const [employees, setEmployees] = useState([]);
  const [departments, setDepartments] = useState([]);
  const [roles, setRoles] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
  const [error, setError] = useState(null);
  const [searchTerm, setSearchTerm] = useState('');
  const [showAddDialog, setShowAddDialog] = useState(false);
  const [showDetailsDialog, setShowDetailsDialog] = useState(false);
  const [showEditDialog, setShowEditDialog] = useState(false);
  const [selectedEmployeeId, setSelectedEmployeeId] = useState(null);
  const [deletingId, setDeletingId] = useState(null);

  useEffect(() => {
    fetchEmployees();
    fetchDepartments();
    fetchRoles();
  }, []);

  const fetchEmployees = async () => {
    try {
      setIsLoading(true);
      const response = await employeeAPI.getAll();
      if (response.data.success) {
        setEmployees(response.data.data);
      } else {
        setError(response.data.message || 'Failed to fetch employees');
      }
    } catch (err) {
      setError(err.response?.data?.message || 'An error occurred while fetching
employees');
    } finally {
      setIsLoading(false);
    }
```

```
  };

  const fetchDepartments = async () => {
    try {
      const response = await employeeAPI.getDepartments();
      if (response.data.success) {
        setDepartments(response.data.data);
      }
    } catch (err) {
      console.error('Error fetching departments:', err);
    }
  };

  const fetchRoles = async () => {
    try {
      const response = await employeeAPI.getRoles();
      if (response.data.success) {
        setRoles(response.data.data);
      }
    } catch (err) {
      console.error('Error fetching roles:', err);
    }
  };

  const handleDelete = async (id) => {
    if (window.confirm('Are you sure you want to delete this employee?')) {
      setDeletingId(id);
      try {
        const response = await employeeAPI.delete(id);
        if (response.data.success) {
          setEmployees(employees.filter(emp => emp._id !== id));
        } else {
          alert('Failed to delete employee. Please try again.');
        }
      } catch (err) {
        alert('Failed to delete employee. Please try again.');
      } finally {
        setDeletingId(null);
      }
    }
  };

  const handleEdit = (employeeId) => {
    setSelectedEmployeeId(employeeId);
    setShowEditDialog(true);
  };

  const handleViewDetails = (employeeId) => {
    setSelectedEmployeeId(employeeId);
    setShowDetailsDialog(true);
  };

  const filteredEmployees = employees.filter((employee) => {
    const departmentName = typeof employee.department === 'object' &&
employee.department !== null
      ? employee.department.name
      : employee.department;
    const roleName = typeof employee.role === 'object' && employee.role !== null
      ? employee.role.name
```

```
      : employee.role;

    return (
      employee.fullName?.toLowerCase().includes(searchTerm.toLowerCase()) ||
      employee.email?.toLowerCase().includes(searchTerm.toLowerCase()) ||
      departmentName?.toLowerCase().includes(searchTerm.toLowerCase()) ||
      roleName?.toLowerCase().includes(searchTerm.toLowerCase())
    );
  });

  if (isLoading && employees.length === 0) {
    return (
      <div className="flex items-center justify-center h-64">
        <div className="text-center">
          <div className="animate-spin rounded-full h-8 w-8 border-t-2 border-b-2
border-blue-500 mx-auto mb-4"></div>
          <p className="text-gray-600 dark:text-gray-400">Loading employees...</p>
        </div>
      </div>
    );
  }

  return (
    <div className="space-y-6">
      <div className="flex justify-between items-center">
        <div>
          <h2 className="text-2xl font-bold tracking-tight text-gray-900
dark:text-white">Employee Management</h2>
          <p className="text-gray-600 dark:text-gray-400">Manage your
organization's employees</p>
        </div>
        <button
          onClick={() => setShowAddDialog(true)}
          className="bg-blue-600 hover:bg-blue-700 text-white px-4 py-2 rounded-
lg flex items-center space-x-2 transition-colors"
        >
          <svg className="w-4 h-4" fill="none" stroke="currentColor" viewBox="0 0
24 24">
            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2}
d="M12 4v16m8-8H4" />
          </svg>
          <span>Add Employee</span>
        </button>
      </div>

      {error && (
        <div className="flex items-center space-x-2 text-red-600 text-sm bg-
red-50 dark:bg-red-900/20 p-3 rounded-md">
          <svg className="w-4 h-4" fill="none" stroke="currentColor" viewBox="0 0
24 24">
            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2}
d="M12 9v2m0 4h.01m-6.938 4h13.856c1.54 0 2.502-1.667 1.732-2.5L13.732
4c-.77-.833-1.964-.833-2.732 0L3.732 16.5c-.77.833.192 2.5 1.732 2.5z" />
          </svg>
          <span>{error}</span>
          <button
            onClick={fetchEmployees}
            className="ml-auto text-sm bg-red-100 hover:bg-red-200 dark:bg-
red-800 dark:hover:bg-red-700 px-2 py-1 rounded"
```

```
          >
            Retry
          </button>
        </div>
      )}

      <div className="flex items-center space-x-2">
        <svg className="w-4 h-4 text-gray-400" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
          <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2}
d="M21 21l-6-6m2-5a7 7 0 11-14 0 7 7 0 0114 0z" />
        </svg>
        <input
          type="text"
          placeholder="Search employees..."
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          className="max-w-sm px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 dark:bg-
gray-700 dark:text-white"
        />
      </div>

      {filteredEmployees.length === 0 ? (
        <div className="text-center py-12">
          <p className="text-gray-500 dark:text-gray-400">No employees found.</p>
        </div>
      ) : (
        <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3">
          {filteredEmployees.map((employee) => (
            <div key={employee._id} className="bg-white dark:bg-gray-800 rounded-
lg shadow-md hover:shadow-lg transition-shadow p-6">
              <div className="flex items-center space-x-3 mb-4">
                <div className="w-12 h-12 bg-blue-100 dark:bg-blue-900 rounded-
full flex items-center justify-center">
                  {employee.photograph ? (
                    <img
                      src={`/api/uploads/${employee.photograph}`}
                      alt={employee.fullName}
                      className="w-12 h-12 rounded-full object-cover"
                    />
                  ) : (
                    <span className="text-blue-600 dark:text-blue-400 font-
semibold text-lg">
                      {employee.fullName ? employee.fullName[0] : 'E'}
                    </span>
                  )}
                </div>
                <div className="flex-1">
                  <h3
                    className="text-lg font-semibold text-gray-900 dark:text-
white cursor-pointer hover:underline"
                    onClick={() => handleViewDetails(employee._id)}
                  >
                    {employee.fullName || 'No Name'}
                  </h3>
                  <p className="text-sm text-gray-600 dark:text-gray-400">
                    {typeof employee.role === 'object' && employee.role !==
null ? employee.role.name : employee.role || 'No Role'}
```

```
                  </p>
                </div>
                <span className={`px-2 py-1 text-xs rounded-full ${
                  employee.status === 'ACTIVE'
                    ? 'bg-green-100 text-green-800 dark:bg-green-900 dark:text-
green-200'
                    : 'bg-gray-100 text-gray-800 dark:bg-gray-700 dark:text-
gray-300'
                }`}>
                  {employee.status?.toLowerCase() || 'unknown'}
                </span>
              </div>

              <div className="space-y-2 mb-4">
                <div className="flex items-center text-sm text-gray-600 dark:text-
gray-400">
                  <svg className="w-4 h-4 mr-2" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M3 8l7.89 4.26a2 2 0 002.22 0L21 8M5 19h14a2 2 0 002-2V7a2 2 0
00-2-2H5a2 2 0 00-2 2v10a2 2 0 002 2z" />
                  </svg>
                  {employee.email || 'No Email'}
                </div>
                <div className="text-sm">
                  <span className="font-medium text-gray-900 dark:text-
white">Department:</span>{' '}
                  <span className="text-gray-600 dark:text-gray-400">
                    {typeof employee.department === 'object' &&
employee.department !== null ? employee.department.name : employee.department ||
'N/A'}
                  </span>
                </div>
                <div className="text-sm">
                  <span className="font-medium text-gray-900 dark:text-
white">Salary:</span>{' '}
                  <span className="text-gray-600 dark:text-gray-400">
                    {employee.salary ? ` ¹${employee.salary.toLocaleString()}` : 'N/
A'}
                  </span>
                </div>
                <div className="text-sm">
                  <span className="font-medium text-gray-900 dark:text-
white">Joining Date:</span>{' '}
                  <span className="text-gray-600 dark:text-gray-400">
                    {employee.joiningDate ? new
Date(employee.joiningDate).toLocaleDateString() : 'N/A'}
                  </span>
                </div>
                {employee.phoneNumber && (
                  <div className="text-sm">
                    <span className="font-medium text-gray-900 dark:text-
white">Phone:</span>{' '}
                    <span className="text-gray-600 dark:text-
gray-400">{employee.phoneNumber}</span>
                  </div>
                )}
              </div>
```

```
              <div className="flex space-x-2 pt-2">
                <button
                  onClick={() => handleEdit(employee._id)}
                  className="flex-1 bg-blue-50 hover:bg-blue-100 dark:bg-blue-900
dark:hover:bg-blue-800 text-blue-600 dark:text-blue-400 px-3 py-2 rounded-lg text-
sm font-medium transition-colors flex items-center justify-center space-x-1"
                >
                  <svg className="w-4 h-4" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M11 5H6a2 2 0 00-2 2v11a2 2 0 002 2h11a2 2 0
002-2v-5m-1.414-9.414a2 2 0 112.828 2.828L11.828 15H9v-2.828l8.586-8.586z" />
                  </svg>
                  <span>Edit</span>
                </button>
                <button
                  onClick={() => handleDelete(employee._id)}
                  disabled={deletingId === employee._id}
                  className="bg-red-50 hover:bg-red-100 dark:bg-red-900
dark:hover:bg-red-800 text-red-600 dark:text-red-400 px-3 py-2 rounded-lg text-sm
font-medium transition-colors disabled:opacity-50"
                >
                  {deletingId === employee._id ? (
                    <div className="animate-spin rounded-full h-4 w-4 border-t-2
border-b-2 border-red-600"></div>
                  ) : (
                    <svg className="w-4 h-4" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M19 7l-.867 12.142A2 2 0 0116.138 21H7.862a2 2 0
01-1.995-1.858L5 7m5 4v6m4-6v6m1-10V4a1 1 0 00-1-1h-4a1 1 0 00-1 1v3M4 7h16" />
                    </svg>
                  )}
                </button>
              </div>
            </div>
          ))}
        </div>
      )}

      <AddEmployeeDialog
        open={showAddDialog}
        onOpenChange={setShowAddDialog}
        onEmployeeAdded={fetchEmployees}
        departments={departments}
        roles={roles}
      />

      <EmployeeDetailsDialog
        employeeId={selectedEmployeeId}
        isOpen={showDetailsDialog}
        onOpenChange={setShowDetailsDialog}
      />

      <EditEmployeeDialog
        employeeId={selectedEmployeeId}
        isOpen={showEditDialog}
        onOpenChange={setShowEditDialog}
        onEmployeeUpdated={fetchEmployees}
```

```
        departments={departments}
        roles={roles}
      />
    </div>
  );
};

export default EmployeeList;
```

src/components/Employee/EmployeeSelfService.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../../context/AuthContext';
import employeeAPI from '../../services/employeeAPI';
import { FaUser, FaPhone, FaEnvelope, FaMapMarkerAlt, FaLinkedin, FaFileUpload,
FaEdit, FaSave, FaTimes, FaSpinner, FaDollarSign } from 'react-icons/fa';

const EmployeeSelfService = ({ employeeData, onDataUpdated }) => {
  const { user } = useAuth();
  const [editing, setEditing] = useState(false);
  const [loading, setLoading] = useState(false);
  const [formData, setFormData] = useState({
    phoneNumber: '',
    whatsappNumber: '',
    linkedInUrl: '',
    currentAddress: '',
    permanentAddress: ''
  });
  const [files, setFiles] = useState({});
  const [uploadingFiles, setUploadingFiles] = useState({});

  useEffect(() => {
    if (employeeData) {
      setFormData({
        phoneNumber: employeeData.phoneNumber || '',
        whatsappNumber: employeeData.whatsappNumber || '',
        linkedInUrl: employeeData.linkedInUrl || '',
        currentAddress: employeeData.currentAddress || '',
        permanentAddress: employeeData.permanentAddress || ''
      });
    }
  }, [employeeData]);

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

  const handleFileChange = (e) => {
    const { name, files: fileList } = e.target;
    if (fileList && fileList[0]) {
      setFiles(prev => ({
        ...prev,
        [name]: fileList[0]
      }));
```

```
    }
  };

  const handleSavePersonalInfo = async () => {
    if (!employeeData) return;

    try {
      setLoading(true);
      const response = await employeeAPI.update(employeeData._id, {
        employee: JSON.stringify(formData)
      });

      if (response.data.success) {
        setEditing(false);
        onDataUpdated?.();
      }
    } catch (error) {
      console.error('Error updating personal info:', error);
      alert('Failed to update personal information. Please try again.');
    } finally {
      setLoading(false);
    }
  };

  const handleFileUpload = async (fieldName) => {
    if (!files[fieldName] || !employeeData) return;

    try {
      setUploadingFiles(prev => ({ ...prev, [fieldName]: true }));

      const formDataToSend = new FormData();
      formDataToSend.append(fieldName, files[fieldName]);

      const response = await employeeAPI.update(employeeData._id, formDataToSend);

      if (response.data.success) {
        setFiles(prev => ({ ...prev, [fieldName]: null }));
        onDataUpdated?.();
        // Reset file input
        const fileInput = document.querySelector(`input[name="${fieldName}"]`);
        if (fileInput) fileInput.value = '';
      }
    } catch (error) {
      console.error('Error uploading file:', error);
      alert('Failed to upload file. Please try again.');
    } finally {
      setUploadingFiles(prev => ({ ...prev, [fieldName]: false }));
    }
  };

  const documentFields = [
    { key: 'photograph', label: 'Photograph', accept: 'image/*', icon: 'Ø=Ü÷' },
    { key: 'resume', label: 'Resume', accept: '.pdf,.doc,.docx', icon: 'Ø=ÜÄ' },
    { key: 'tenthMarksheet', label: '10th Marksheet', accept:
'.pdf,.jpg,.jpeg,.png', icon: 'Ø=ÜÜ' },
    { key: 'twelfthMarksheet', label: '12th Marksheet', accept:
'.pdf,.jpg,.jpeg,.png', icon: 'Ø=ÜÜ' },
    { key: 'bachelorDegree', label: 'Bachelor Degree', accept:
'.pdf,.jpg,.jpeg,.png', icon: 'Ø<ß“' },
```

```
    { key: 'postgraduateDegree', label: 'Postgraduate Degree', accept:
'.pdf,.jpg,.jpeg,.png', icon: 'Ø<ß“' },
    { key: 'aadharCard', label: 'Aadhar Card', accept: '.pdf,.jpg,.jpeg,.png',
icon: 'Ø<Ý”' },
    { key: 'panCard', label: 'PAN Card', accept: '.pdf,.jpg,.jpeg,.png', icon:
'Ø<Ý”' },
    { key: 'pcc', label: 'PCC', accept: '.pdf,.jpg,.jpeg,.png', icon: 'Ø=Þáþ ' },
    { key: 'offerLetter', label: 'Offer Letter', accept: '.pdf,.doc,.docx', icon:
'Ø=Üè' }
  ];

  return (
    <div className="space-y-6 max-w-full overflow-hidden">
      {/* Personal Information Section */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-4 sm:p-6">
        <div className="flex flex-col sm:flex-row justify-between items-start
sm:items-center gap-4 mb-6">
          <h3 className="text-lg font-semibold text-gray-900 dark:text-white flex
items-center">
            <FaUser className="mr-2 text-blue-600" />
            Personal Information
          </h3>
          {!editing ? (
            <button
              onClick={() => setEditing(true)}
              className="flex items-center px-3 py-1 bg-blue-600 text-white
rounded-md hover:bg-blue-700 transition-colors text-sm"
            >
              <FaEdit className="mr-1" />
              Edit
            </button>
          ) : (
            <div className="flex space-x-2">
              <button
                onClick={handleSavePersonalInfo}
                disabled={loading}
                className="flex items-center px-3 py-1 bg-green-600 text-white
rounded-md hover:bg-green-700 transition-colors disabled:opacity-50 text-sm"
              >
                {loading ? <FaSpinner className="animate-spin mr-1" /> : <FaSave
className="mr-1" />}
                Save
              </button>
              <button
                onClick={() => setEditing(false)}
                className="flex items-center px-3 py-1 bg-gray-600 text-white
rounded-md hover:bg-gray-700 transition-colors text-sm"
              >
                <FaTimes className="mr-1" />
                Cancel
              </button>
            </div>
          )}
        </div>

        <div className="grid grid-cols-1 lg:grid-cols-2 gap-4">
          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
```

```
              <FaPhone className="inline mr-2" />
              Phone Number
            </label>
            {editing ? (
              <input
                type="tel"
                name="phoneNumber"
                value={formData.phoneNumber}
                onChange={handleInputChange}
                className="w-full px-3 py-2 border border-gray-300 rounded-md
dark:bg-gray-700 dark:border-gray-600 dark:text-white focus:ring-2 focus:ring-
blue-500 focus:border-transparent"
                placeholder="Enter phone number"
              />
            ) : (
              <p className="text-gray-900 dark:text-white
py-2">{formData.phoneNumber || 'Not provided'}</p>
            )}
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              <FaPhone className="inline mr-2" />
              WhatsApp Number
            </label>
            {editing ? (
              <input
                type="tel"
                name="whatsappNumber"
                value={formData.whatsappNumber}
                onChange={handleInputChange}
                className="w-full px-3 py-2 border border-gray-300 rounded-md
dark:bg-gray-700 dark:border-gray-600 dark:text-white focus:ring-2 focus:ring-
blue-500 focus:border-transparent"
                placeholder="Enter WhatsApp number"
              />
            ) : (
              <p className="text-gray-900 dark:text-white
py-2">{formData.whatsappNumber || 'Not provided'}</p>
            )}
          </div>

          <div className="lg:col-span-2">
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              <FaLinkedin className="inline mr-2" />
              LinkedIn URL
            </label>
            {editing ? (
              <input
                type="url"
                name="linkedInUrl"
                value={formData.linkedInUrl}
                onChange={handleInputChange}
                className="w-full px-3 py-2 border border-gray-300 rounded-md
dark:bg-gray-700 dark:border-gray-600 dark:text-white focus:ring-2 focus:ring-
blue-500 focus:border-transparent"
                placeholder="Enter LinkedIn profile URL"
```

```
              />
            ) : (
              <p className="text-gray-900 dark:text-white py-2">
                {formData.linkedInUrl ? (
                  <a href={formData.linkedInUrl} target="_blank" rel="noopener
noreferrer" className="text-blue-600 hover:underline break-all">
                    {formData.linkedInUrl}
                  </a>
                ) : (
                  'Not provided'
                )}
              </p>
            )}
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              <FaMapMarkerAlt className="inline mr-2" />
              Current Address
            </label>
            {editing ? (
              <textarea
                name="currentAddress"
                value={formData.currentAddress}
                onChange={handleInputChange}
                rows="3"
                className="w-full px-3 py-2 border border-gray-300 rounded-md
dark:bg-gray-700 dark:border-gray-600 dark:text-white focus:ring-2 focus:ring-
blue-500 focus:border-transparent resize-none"
                placeholder="Enter current address"
              />
            ) : (
              <p className="text-gray-900 dark:text-white
py-2">{formData.currentAddress || 'Not provided'}</p>
            )}
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              <FaMapMarkerAlt className="inline mr-2" />
              Permanent Address
            </label>
            {editing ? (
              <textarea
                name="permanentAddress"
                value={formData.permanentAddress}
                onChange={handleInputChange}
                rows="3"
                className="w-full px-3 py-2 border border-gray-300 rounded-md
dark:bg-gray-700 dark:border-gray-600 dark:text-white focus:ring-2 focus:ring-
blue-500 focus:border-transparent resize-none"
                placeholder="Enter permanent address"
              />
            ) : (
              <p className="text-gray-900 dark:text-white
py-2">{formData.permanentAddress || 'Not provided'}</p>
            )}
```

```
          </div>
        </div>
      </div>

      {/* Document Upload Section */}
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-4 sm:p-6">
        <h3 className="text-lg font-semibold text-gray-900 dark:text-white flex
items-center mb-6">
          <FaFileUpload className="mr-2 text-blue-600" />
          Document Management
        </h3>

        <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
          {documentFields.map((field) => (
            <div key={field.key} className="border border-gray-200 dark:border-
gray-700 rounded-lg p-4">
              <div className="flex items-center justify-between mb-2">
                <label className="text-sm font-medium text-gray-700 dark:text-
gray-300">
                  {field.icon} {field.label}
                </label>
                {employeeData && employeeData[field.key] && (
                  <span className="text-xs text-green-600 bg-green-100 px-2 py-1
rounded">
                    '  Uploaded
                  </span>
                )}
              </div>

              {employeeData && employeeData[field.key] && (
                <p className="text-xs text-gray-500 mb-2 truncate">
                  Current: {employeeData[field.key].split('/').pop()}
                </p>
              )}

              <div className="flex flex-col sm:flex-row items-stretch sm:items-
center gap-2">
                <input
                  type="file"
                  name={field.key}
                  accept={field.accept}
                  onChange={handleFileChange}
                  className="flex-1 text-sm text-gray-500 file:mr-2 file:py-1
file:px-2 file:rounded file:border-0 file:bg-blue-50 file:text-blue-700
hover:file:bg-blue-100 file:cursor-pointer cursor-pointer"
                />
                {files[field.key] && (
                  <button
                    onClick={() => handleFileUpload(field.key)}
                    disabled={uploadingFiles[field.key]}
                    className="px-3 py-1 bg-blue-600 text-white rounded text-sm
hover:bg-blue-700 disabled:opacity-50 flex items-center justify-center whitespace-
nowrap"
                  >
                    {uploadingFiles[field.key] ? (
                      <FaSpinner className="animate-spin mr-1" />
                    ) : (
                      <FaFileUpload className="mr-1" />
                    )}
```

```
                  Upload
                </button>
              )}
            </div>
          </div>
        ))}
      </div>
    </div>

    {/* Read-only Information */}
    <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-4 sm:p-6">
      <h3 className="text-lg font-semibold text-gray-900 dark:text-white flex
items-center mb-6">
        <FaUser className="mr-2 text-blue-600" />
        Professional Information
      </h3>
      <div className="grid grid-cols-1 md:grid-cols-2 xl:grid-cols-3 gap-4 text-
sm">
        <div>
          <p className="text-gray-600 dark:text-gray-400">Email</p>
          <p className="font-medium text-gray-900 dark:text-white flex items-
center">
            <FaEnvelope className="mr-2 text-gray-500" />
            <span className="truncate">{employeeData?.email || 'N/A'}</span>
          </p>
        </div>
        <div>
          <p className="text-gray-600 dark:text-gray-400">Employee ID</p>
          <p className="font-medium text-gray-900 dark:text-
white">{employeeData?.employeeId || 'N/A'}</p>
        </div>
        <div>
          <p className="text-gray-600 dark:text-gray-400">Department</p>
          <p className="font-medium text-gray-900 dark:text-white">
            {employeeData?.department?.name || 'N/A'}
          </p>
        </div>
        <div>
          <p className="text-gray-600 dark:text-gray-400">Role</p>
          <p className="font-medium text-gray-900 dark:text-white">
            {employeeData?.role?.name || 'N/A'}
          </p>
        </div>
        <div>
          <p className="text-gray-600 dark:text-gray-400">Salary</p>
          <p className="font-medium text-gray-900 dark:text-white flex items-
center">
            <FaDollarSign className="mr-1 text-gray-500" />
            {employeeData?.salary ? ` ¹${employeeData.salary.toLocaleString()}` :
'Not disclosed'}
          </p>
        </div>
        <div>
          <p className="text-gray-600 dark:text-gray-400">Status</p>
          <span className={`inline-flex px-2 py-1 text-xs rounded-full ${
            employeeData?.status === 'ACTIVE'
              ? 'bg-green-100 text-green-800 dark:bg-green-900 dark:text-
green-200'
              : 'bg-gray-100 text-gray-800 dark:bg-gray-700 dark:text-gray-300'
```

```
            }`}>
              {employeeData?.status || 'N/A'}
            </span>
          </div>
          <div>
            <p className="text-gray-600 dark:text-gray-400">Joining Date</p>
            <p className="font-medium text-gray-900 dark:text-white">
              {employeeData?.joiningDate ? new
Date(employeeData.joiningDate).toLocaleDateString() : 'N/A'}
            </p>
          </div>
        </div>
      </div>
    </div>
  );
};

export default EmployeeSelfService;
```

src/components/Employee/LeaveApplication.jsx

```
import React, { useState, useEffect } from 'react';
import { FaCalendarPlus, FaCalendarCheck, FaClock, FaExclamationTriangle,
FaCheckCircle, FaTimes } from 'react-icons/fa';
import leaveAPI from '../../services/leaveAPI';

const LeaveApplication = () => {
  const [leaveData, setLeaveData] = useState({
    leaveType: 'casual',
    startDate: '',
    endDate: '',
    reason: '',
    isHalfDay: false,
    halfDaySession: 'morning'
  });

  const [myLeaves, setMyLeaves] = useState([]);
  const [leaveBalance, setLeaveBalance] = useState({});
  const [loading, setLoading] = useState(false);
  const [submitting, setSubmitting] = useState(false);
  const [error, setError] = useState('');
  const [success, setSuccess] = useState('');
  const [activeTab, setActiveTab] = useState('apply');

  const leaveTypes = [
    { value: 'casual', label: 'Casual Leave', icon: 'Ø<ßÖþ ' },
    { value: 'sick', label: 'Sick Leave', icon: 'Ø>Ý ' },
    { value: 'annual', label: 'Annual Leave', icon: '' þ ' },
    { value: 'emergency', label: 'Emergency Leave', icon: 'Ø=Þ¨' },
    { value: 'personal', label: 'Personal Leave', icon: 'Ø=Üd' },
    { value: 'maternity', label: 'Maternity Leave', icon: 'Ø>Ý1' },
    { value: 'paternity', label: 'Paternity Leave', icon: 'Ø=Üh  Ø=Üv' },
    { value: 'bereavement', label: 'Bereavement Leave', icon: 'Ø=ÞO' }
  ];

  const statusColors = {
    pending: 'bg-yellow-100 text-yellow-800 border-yellow-200',
    approved: 'bg-green-100 text-green-800 border-green-200',
```

```
    rejected: 'bg-red-100 text-red-800 border-red-200',
    cancelled: 'bg-gray-100 text-gray-800 border-gray-200'
  };

  const statusIcons = {
    pending: <FaClock className="w-4 h-4" />,
    approved: <FaCheckCircle className="w-4 h-4" />,
    rejected: <FaTimes className="w-4 h-4" />,
    cancelled: <FaTimes className="w-4 h-4" />
  };

  useEffect(() => {
    fetchMyLeaves();
    fetchLeaveBalance();
  }, []);

  const fetchMyLeaves = async () => {
    try {
      setLoading(true);
      const response = await leaveAPI.getMyLeaves();
      setMyLeaves(response.data.data || []);
    } catch (error) {
      setError('Failed to fetch leave history');
      console.error('Error fetching leaves:', error);
    } finally {
      setLoading(false);
    }
  };

  const fetchLeaveBalance = async () => {
    try {
      const response = await leaveAPI.getLeaveBalance();
      setLeaveBalance(response.data.data || {});
    } catch (error) {
      console.error('Error fetching leave balance:', error);
    }
  };

  const handleInputChange = (e) => {
    const { name, value, type, checked } = e.target;
    setLeaveData(prev => ({
      ...prev,
      [name]: type === 'checkbox' ? checked : value
    }));

    // Clear error when user starts typing
    if (error) setError('');
  };

  const validateForm = () => {
    if (!leaveData.leaveType || !leaveData.startDate || !leaveData.endDate || !
leaveData.reason.trim()) {
      setError('Please fill in all required fields');
      return false;
    }

    if (leaveData.reason.trim().length < 10) {
      setError('Reason must be at least 10 characters long');
      return false;
```

```
    }

    const startDate = new Date(leaveData.startDate);
    const endDate = new Date(leaveData.endDate);
    const today = new Date();
    today.setHours(0, 0, 0, 0);

    if (startDate < today) {
      setError('Start date cannot be in the past');
      return false;
    }

    if (startDate > endDate) {
      setError('End date must be after start date');
      return false;
    }

    if (leaveData.isHalfDay && !leaveData.halfDaySession) {
      setError('Please select half day session');
      return false;
    }

    return true;
  };

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

    if (!validateForm()) return;

    try {
      setSubmitting(true);
      setError('');

      await leaveAPI.applyLeave(leaveData);

      setSuccess('Leave application submitted successfully!');
      setLeaveData({
        leaveType: 'casual',
        startDate: '',
        endDate: '',
        reason: '',
        isHalfDay: false,
        halfDaySession: 'morning'
      });

      // Refresh data
      fetchMyLeaves();
      fetchLeaveBalance();

      // Switch to history tab to see the new application
      setTimeout(() => {
        setActiveTab('history');
        setSuccess('');
      }, 2000);

    } catch (error) {
      setError(error.response?.data?.message || 'Failed to submit leave
application');
```

```
    } finally {
      setSubmitting(false);
    }
  };

  const handleCancelLeave = async (leaveId) => {
    if (!window.confirm('Are you sure you want to cancel this leave
application?')) {
      return;
    }

    try {
      await leaveAPI.cancelLeave(leaveId);
      setSuccess('Leave cancelled successfully');
      fetchMyLeaves();
      fetchLeaveBalance();
      setTimeout(() => setSuccess(''), 3000);
    } catch (error) {
      setError(error.response?.data?.message || 'Failed to cancel leave');
      setTimeout(() => setError(''), 3000);
    }
  };

  const formatDate = (dateString) => {
    return new Date(dateString).toLocaleDateString('en-US', {
      year: 'numeric',
      month: 'short',
      day: 'numeric'
    });
  };

  const calculateDays = () => {
    if (!leaveData.startDate || !leaveData.endDate) return 0;

    if (leaveData.isHalfDay) return 0.5;

    const start = new Date(leaveData.startDate);
    const end = new Date(leaveData.endDate);
    const timeDiff = end.getTime() - start.getTime();
    return Math.ceil(timeDiff / (1000 * 3600 * 24)) + 1;
  };

  return (
    <div className="max-w-6xl mx-auto p-6">
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md border
border-slate-200 dark:border-slate-700">
        <div className="border-b border-gray-200 dark:border-gray-700">
          <nav className="flex space-x-8 px-6">
            <button
              onClick={() => setActiveTab('apply')}
              className={`${
                activeTab === 'apply'
                  ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                  : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
              } whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm flex
items-center`}
            >
              <FaCalendarPlus className="mr-2" />
```

```
              Apply Leave
            </button>
            <button
              onClick={() => setActiveTab('history')}
              className={`${
                activeTab === 'history'
                  ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                  : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
              } whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm flex
items-center`}
            >
              <FaCalendarCheck className="mr-2" />
              Leave History
            </button>
            <button
              onClick={() => setActiveTab('balance')}
              className={`${
                activeTab === 'balance'
                  ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                  : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
              } whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm flex
items-center`}
            >
              <FaClock className="mr-2" />
              Leave Balance
            </button>
          </nav>
        </div>

        <div className="p-6">
          {/* Success/Error Messages */}
          {error && (
            <div className="mb-4 p-3 bg-red-50 border border-red-200 text-red-600
rounded-md flex items-center">
              <FaExclamationTriangle className="mr-2" />
              {error}
            </div>
          )}

          {success && (
            <div className="mb-4 p-3 bg-green-50 border border-green-200 text-
green-600 rounded-md flex items-center">
              <FaCheckCircle className="mr-2" />
              {success}
            </div>
          )}

          {/* Apply Leave Tab */}
          {activeTab === 'apply' && (
            <div>
              <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-6">Apply for Leave</h3>

              <form onSubmit={handleSubmit} className="space-y-6">
                <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
                  {/* Leave Type */}
                  <div>
```

```
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      Leave Type *
                    </label>
                    <select
                      name="leaveType"
                      value={leaveData.leaveType}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100 focus:outline-none focus:ring-2 focus:ring-blue-500"
                      required
                    >
                      {leaveTypes.map(type => (
                        <option key={type.value} value={type.value}>
                          {type.icon} {type.label}
                        </option>
                      ))}
                    </select>
                  </div>

                  {/* Half Day Option */}
                  <div className="flex items-center space-x-4">
                    <label className="flex items-center">
                      <input
                        type="checkbox"
                        name="isHalfDay"
                        checked={leaveData.isHalfDay}
                        onChange={handleInputChange}
                        className="mr-2 h-4 w-4 text-blue-600 focus:ring-blue-500
border-gray-300 rounded"
                      />
                      <span className="text-sm font-medium text-gray-700
dark:text-gray-300">Half Day</span>
                    </label>

                    {leaveData.isHalfDay && (
                      <select
                        name="halfDaySession"
                        value={leaveData.halfDaySession}
                        onChange={handleInputChange}
                        className="border border-gray-300 dark:border-gray-600
rounded-md px-3 py-1 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
text-sm"
                      >
                        <option value="morning">Morning</option>
                        <option value="afternoon">Afternoon</option>
                      </select>
                    )}
                  </div>
                </div>

                <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
                  {/* Start Date */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      Start Date *
                    </label>
```

```
                    <input
                      type="date"
                      name="startDate"
                      value={leaveData.startDate}
                      onChange={handleInputChange}
                      min={new Date().toISOString().split('T')[0]}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100 focus:outline-none focus:ring-2 focus:ring-blue-500"
                      required
                    />
                  </div>

                  {/* End Date */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      End Date *
                    </label>
                    <input
                      type="date"
                      name="endDate"
                      value={leaveData.endDate}
                      onChange={handleInputChange}
                      min={leaveData.startDate || new
Date().toISOString().split('T')[0]}
                      disabled={leaveData.isHalfDay}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100 focus:outline-none focus:ring-2 focus:ring-blue-500 disabled:bg-gray-100
disabled:cursor-not-allowed"
                      required={!leaveData.isHalfDay}
                    />
                    {leaveData.isHalfDay && (
                      <input type="hidden" name="endDate"
value={leaveData.startDate} />
                    )}
                  </div>
                </div>

                {/* Total Days Display */}
                {(leaveData.startDate && (leaveData.endDate ||
leaveData.isHalfDay)) && (
                  <div className="bg-blue-50 dark:bg-blue-900/20 border border-
blue-200 dark:border-blue-800 rounded-md p-3">
                    <p className="text-sm text-blue-800 dark:text-blue-200">
                      <span className="font-medium">Total Days:</span>
{calculateDays()} day(s)
                    </p>
                  </div>
                )}

                {/* Reason */}
                <div>
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                    Reason for Leave *
                  </label>
                  <textarea
```

```
                    name="reason"
                    value={leaveData.reason}
                    onChange={handleInputChange}
                    rows={4}
                    placeholder="Please provide a detailed reason for your leave
(minimum 10 characters)"
                    className="w-full border border-gray-300 dark:border-gray-600
rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-2 focus:ring-blue-500"
                    required
                    minLength={10}
                  />
                  <p className="text-xs text-gray-500 mt-1">
                    {leaveData.reason.length}/500 characters (minimum 10)
                  </p>
                </div>

                {/* Submit Button */}
                <div className="flex justify-end">
                  <button
                    type="submit"
                    disabled={submitting}
                    className="bg-blue-600 hover:bg-blue-700 disabled:bg-blue-400
text-white px-6 py-2 rounded-md font-medium transition duration-200 flex items-
center"
                  >
                    {submitting ? (
                      <>
                        <div className="animate-spin rounded-full h-4 w-4 border-
b-2 border-white mr-2"></div>
                        Submitting...
                      </>
                    ) : (
                      <>
                        <FaCalendarPlus className="mr-2" />
                        Submit Application
                      </>
                    )}
                  </button>
                </div>
              </form>
            </div>
          )}

          {/* Leave History Tab */}
          {activeTab === 'history' && (
            <div>
              <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-6">Leave History</h3>

              {loading ? (
                <div className="flex justify-center py-8">
                  <div className="animate-spin rounded-full h-8 w-8 border-b-2
border-blue-600"></div>
                </div>
              ) : myLeaves.length === 0 ? (
                <div className="text-center py-8 text-gray-500 dark:text-
gray-400">
                  <FaCalendarCheck className="mx-auto h-12 w-12 mb-4 opacity-50" /
>
```

```
                  <p>No leave applications found</p>
                </div>
              ) : (
                <div className="space-y-4">
                  {myLeaves.map((leave) => (
                    <div key={leave._id} className="border border-gray-200
dark:border-gray-700 rounded-lg p-4">
                      <div className="flex items-start justify-between">
                        <div className="flex-1">
                          <div className="flex items-center space-x-3 mb-2">
                            <h4 className="font-medium text-gray-900 dark:text-
white">
                              {leaveTypes.find(t => t.value ===
leave.leaveType)?.icon} {' '}
                              {leaveTypes.find(t => t.value ===
leave.leaveType)?.label}
                            </h4>
                            <span className={`inline-flex items-center px-2.5
py-0.5 rounded-full text-xs font-medium border ${statusColors[leave.status]}`}>
                              {statusIcons[leave.status]}
                              <span className="ml-1 capitalize">{leave.status}</
span>
                            </span>
                          </div>

                          <div className="grid grid-cols-1 md:grid-cols-3 gap-4
text-sm text-gray-600 dark:text-gray-400">
                            <div>
                              <span className="font-medium">Duration:</span>
{formatDate(leave.startDate)}
                              {leave.startDate !== leave.endDate && ` -
${formatDate(leave.endDate)}`}
                              {leave.isHalfDay && ` (${leave.halfDaySession}
half)`}
                            </div>
                            <div>
                              <span className="font-medium">Days:</span>
{leave.totalDays}
                            </div>
                            <div>
                              <span className="font-medium">Applied:</span>
{formatDate(leave.appliedDate)}
                            </div>
                          </div>

                          <div className="mt-2">
                            <span className="font-medium text-sm text-gray-600
dark:text-gray-400">Reason:</span>
                            <p className="text-sm text-gray-800 dark:text-
gray-200 mt-1">{leave.reason}</p>
                          </div>

                          {leave.rejectionReason && (
                            <div className="mt-2 p-2 bg-red-50 dark:bg-red-900/20
border border-red-200 dark:border-red-800 rounded">
                              <span className="font-medium text-sm text-red-800
dark:text-red-200">Rejection Reason:</span>
                              <p className="text-sm text-red-700 dark:text-
red-300 mt-1">{leave.rejectionReason}</p>
```

```
                          </div>
                        )}

                        {leave.approvedBy && leave.status === 'approved' && (
                          <div className="mt-2 text-sm text-green-600 dark:text-
green-400">
                            <span className="font-medium">Approved by:</span>
{leave.approvedBy.fullName} on {formatDate(leave.approvedDate)}
                          </div>
                        )}
                      </div>

                      {leave.status === 'pending' && (
                        <button
                          onClick={() => handleCancelLeave(leave._id)}
                          className="ml-4 text-red-600 hover:text-red-800 text-
sm font-medium"
                        >
                          Cancel
                        </button>
                      )}
                    </div>
                  </div>
                ))}
              </div>
            )}
          </div>
        )}

        {/* Leave Balance Tab */}
        {activeTab === 'balance' && (
          <div>
            <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-6">Leave Balance ({new Date().getFullYear()})</h3>

            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3
gap-6">
              {Object.entries(leaveBalance).map(([type, balance]) => {
                const leaveTypeInfo = leaveTypes.find(t => t.value === type);
                return (
                  <div key={type} className="bg-gradient-to-r from-blue-50 to-
indigo-50 dark:from-blue-900/20 dark:to-indigo-900/20 border border-blue-200
dark:border-blue-800 rounded-lg p-4">
                    <div className="flex items-center justify-between mb-3">
                      <h4 className="font-medium text-gray-900 dark:text-white
flex items-center">
                        <span className="mr-2">{leaveTypeInfo?.icon}</span>
                        {leaveTypeInfo?.label}
                      </h4>
                    </div>

                    <div className="space-y-2">
                      <div className="flex justify-between text-sm">
                        <span className="text-gray-600 dark:text-
gray-400">Total:</span>
                        <span className="font-medium text-gray-900 dark:text-
white">{balance.total} days</span>
                      </div>
                      <div className="flex justify-between text-sm">
```

```
                          <span className="text-gray-600 dark:text-
gray-400">Used:</span>
                          <span className="font-medium text-red-600 dark:text-
red-400">{balance.used} days</span>
                        </div>
                        <div className="flex justify-between text-sm">
                          <span className="text-gray-600 dark:text-
gray-400">Remaining:</span>
                          <span className="font-medium text-green-600 dark:text-
green-400">{balance.remaining} days</span>
                        </div>

                        {/* Progress bar */}
                        <div className="mt-3">
                          <div className="bg-gray-200 dark:bg-gray-700 rounded-
full h-2">
                            <div
                              className="bg-blue-600 h-2 rounded-full transition-
all duration-300"
                              style={{ width: `${(balance.used / balance.total) *
100}%` }}
                            ></div>
                          </div>
                        </div>
                      </div>
                    </div>
                  );
                })}
              </div>
            </div>
          )}
        </div>
      </div>
    </div>
  );
};

export default LeaveApplication;
```

src/components/Employee/LeaveManagement.jsx

```
import React, { useState, useEffect } from 'react';
import { FaCalendarPlus, FaCalendarCheck, FaClock, FaExclamationTriangle,
FaCheckCircle, FaTimes } from 'react-icons/fa';
import leaveAPI from '../../services/leaveAPI';

const LeaveManagement = ({ employeeId, userRole }) => {
  const [leaveData, setLeaveData] = useState({
    leaveType: 'casual',
    startDate: '',
    endDate: '',
    reason: '',
    isHalfDay: false,
    halfDaySession: 'morning'
  });

  const [myLeaves, setMyLeaves] = useState([]);
  const [leaveBalance, setLeaveBalance] = useState({});
```

```
const [loading, setLoading] = useState(false);
const [submitting, setSubmitting] = useState(false);
const [error, setError] = useState('');
const [success, setSuccess] = useState('');
const [activeTab, setActiveTab] = useState('apply');

const leaveTypes = [
  { value: 'casual', label: 'Casual Leave', icon: 'Ø<ßÖþ ' },
  { value: 'sick', label: 'Sick Leave', icon: 'Ø>Ý ' },
  { value: 'annual', label: 'Annual Leave', icon: '' þ ' },
  { value: 'emergency', label: 'Emergency Leave', icon: 'Ø=Þ¨' },
  { value: 'personal', label: 'Personal Leave', icon: 'Ø=Üd' },
  { value: 'maternity', label: 'Maternity Leave', icon: 'Ø>Ý1' },
  { value: 'paternity', label: 'Paternity Leave', icon: 'Ø=Üh  Ø=Üv' },
  { value: 'bereavement', label: 'Bereavement Leave', icon: 'Ø=ÞO' }
];

const statusColors = {
  pending: 'bg-yellow-100 text-yellow-800 border-yellow-200',
  approved: 'bg-green-100 text-green-800 border-green-200',
  rejected: 'bg-red-100 text-red-800 border-red-200',
  cancelled: 'bg-gray-100 text-gray-800 border-gray-200'
};

const statusIcons = {
  pending: <FaClock className="w-4 h-4" />,
  approved: <FaCheckCircle className="w-4 h-4" />,
  rejected: <FaTimes className="w-4 h-4" />,
  cancelled: <FaTimes className="w-4 h-4" />
};

useEffect(() => {
  fetchMyLeaves();
  fetchLeaveBalance();
}, []);

const fetchMyLeaves = async () => {
  try {
    setLoading(true);
    const response = await leaveAPI.getMyLeaves();
    setMyLeaves(response.data.data || []);
  } catch (error) {
    setError('Failed to fetch leave history');
    console.error('Error fetching leaves:', error);
  } finally {
    setLoading(false);
  }
};

const fetchLeaveBalance = async () => {
  try {
    const response = await leaveAPI.getLeaveBalance();
    setLeaveBalance(response.data.data || {});
  } catch (error) {
    console.error('Error fetching leave balance:', error);
  }
};

const handleInputChange = (e) => {
```

```
    const { name, value, type, checked } = e.target;
    setLeaveData(prev => ({
      ...prev,
      [name]: type === 'checkbox' ? checked : value
    }));

    // Clear error when user starts typing
    if (error) setError('');
  };

  const validateForm = () => {
    if (!leaveData.leaveType || !leaveData.startDate || !leaveData.endDate || !
leaveData.reason.trim()) {
      setError('Please fill in all required fields');
      return false;
    }

    if (leaveData.reason.trim().length < 10) {
      setError('Reason must be at least 10 characters long');
      return false;
    }

    const startDate = new Date(leaveData.startDate);
    const endDate = new Date(leaveData.endDate);
    const today = new Date();
    today.setHours(0, 0, 0, 0);

    if (startDate < today) {
      setError('Start date cannot be in the past');
      return false;
    }

    if (startDate > endDate) {
      setError('End date must be after start date');
      return false;
    }

    if (leaveData.isHalfDay && !leaveData.halfDaySession) {
      setError('Please select half day session');
      return false;
    }

    return true;
  };

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

    if (!validateForm()) return;

    try {
      setSubmitting(true);
      setError('');

      await leaveAPI.applyLeave(leaveData);

      setSuccess('Leave application submitted successfully!');
      setLeaveData({
        leaveType: 'casual',
```

```
        startDate: '',
        endDate: '',
        reason: '',
        isHalfDay: false,
        halfDaySession: 'morning'
      });

      // Refresh data
      fetchMyLeaves();
      fetchLeaveBalance();

      // Switch to history tab to see the new application
      setTimeout(() => {
        setActiveTab('history');
        setSuccess('');
      }, 2000);

    } catch (error) {
      setError(error.response?.data?.message || 'Failed to submit leave
application');
    } finally {
      setSubmitting(false);
    }
  };

  const handleCancelLeave = async (leaveId) => {
    if (!window.confirm('Are you sure you want to cancel this leave
application?')) {
      return;
    }

    try {
      await leaveAPI.cancelLeave(leaveId);
      setSuccess('Leave cancelled successfully');
      fetchMyLeaves();
      fetchLeaveBalance();
      setTimeout(() => setSuccess(''), 3000);
    } catch (error) {
      setError(error.response?.data?.message || 'Failed to cancel leave');
      setTimeout(() => setError(''), 3000);
    }
  };

  const formatDate = (dateString) => {
    return new Date(dateString).toLocaleDateString('en-US', {
      year: 'numeric',
      month: 'short',
      day: 'numeric'
    });
  };

  const calculateDays = () => {
    if (!leaveData.startDate || !leaveData.endDate) return 0;

    if (leaveData.isHalfDay) return 0.5;

    const start = new Date(leaveData.startDate);
    const end = new Date(leaveData.endDate);
    const timeDiff = end.getTime() - start.getTime();
```

```
    return Math.ceil(timeDiff / (1000 * 3600 * 24)) + 1;
  };

  return (
    <div className="max-w-6xl mx-auto p-6">
      <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md border
border-slate-200 dark:border-slate-700">
        <div className="border-b border-gray-200 dark:border-gray-700">
          <nav className="flex space-x-8 px-6">
            <button
              onClick={() => setActiveTab('apply')}
              className={`${
                activeTab === 'apply'
                  ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                  : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
              } whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm flex
items-center`}
            >
              <FaCalendarPlus className="mr-2" />
              Apply Leave
            </button>
            <button
              onClick={() => setActiveTab('history')}
              className={`${
                activeTab === 'history'
                  ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                  : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
              } whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm flex
items-center`}
            >
              <FaCalendarCheck className="mr-2" />
              Leave History
            </button>
            <button
              onClick={() => setActiveTab('balance')}
              className={`${
                activeTab === 'balance'
                  ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                  : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
              } whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm flex
items-center`}
            >
              <FaClock className="mr-2" />
              Leave Balance
            </button>
          </nav>
        </div>

        <div className="p-6">
          {/* Success/Error Messages */}
          {error && (
            <div className="mb-4 p-3 bg-red-50 border border-red-200 text-red-600
rounded-md flex items-center">
              <FaExclamationTriangle className="mr-2" />
              {error}
            </div>
```

```
          )}

          {success && (
            <div className="mb-4 p-3 bg-green-50 border border-green-200 text-
green-600 rounded-md flex items-center">
              <FaCheckCircle className="mr-2" />
              {success}
            </div>
          )}

          {/* Apply Leave Tab */}
          {activeTab === 'apply' && (
            <div>
              <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-6">Apply for Leave</h3>

              <form onSubmit={handleSubmit} className="space-y-6">
                <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
                  {/* Leave Type */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      Leave Type *
                    </label>
                    <select
                      name="leaveType"
                      value={leaveData.leaveType}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100 focus:outline-none focus:ring-2 focus:ring-blue-500"
                      required
                    >
                      {leaveTypes.map(type => (
                        <option key={type.value} value={type.value}>
                          {type.icon} {type.label}
                        </option>
                      ))}
                    </select>
                  </div>

                  {/* Half Day Option */}
                  <div className="flex items-center space-x-4">
                    <label className="flex items-center">
                      <input
                        type="checkbox"
                        name="isHalfDay"
                        checked={leaveData.isHalfDay}
                        onChange={handleInputChange}
                        className="mr-2 h-4 w-4 text-blue-600 focus:ring-blue-500
border-gray-300 rounded"
                      />
                      <span className="text-sm font-medium text-gray-700
dark:text-gray-300">Half Day</span>
                    </label>

                    {leaveData.isHalfDay && (
                      <select
                        name="halfDaySession"
```

```
                        value={leaveData.halfDaySession}
                        onChange={handleInputChange}
                        className="border border-gray-300 dark:border-gray-600
rounded-md px-3 py-1 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
text-sm"
                      >
                        <option value="morning">Morning</option>
                        <option value="afternoon">Afternoon</option>
                      </select>
                    )}
                  </div>
                </div>

                <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
                  {/* Start Date */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      Start Date *
                    </label>
                    <input
                      type="date"
                      name="startDate"
                      value={leaveData.startDate}
                      onChange={handleInputChange}
                      min={new Date().toISOString().split('T')[0]}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100 focus:outline-none focus:ring-2 focus:ring-blue-500"
                      required
                    />
                  </div>

                  {/* End Date */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      End Date *
                    </label>
                    <input
                      type="date"
                      name="endDate"
                      value={leaveData.endDate}
                      onChange={handleInputChange}
                      min={leaveData.startDate || new
Date().toISOString().split('T')[0]}
                      disabled={leaveData.isHalfDay}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100 focus:outline-none focus:ring-2 focus:ring-blue-500 disabled:bg-gray-100
disabled:cursor-not-allowed"
                      required={!leaveData.isHalfDay}
                    />
                    {leaveData.isHalfDay && (
                      <input type="hidden" name="endDate"
value={leaveData.startDate} />
                    )}
                  </div>
                </div>
```

```
                {/* Total Days Display */}
                {(leaveData.startDate && (leaveData.endDate ||
leaveData.isHalfDay)) && (
                  <div className="bg-blue-50 dark:bg-blue-900/20 border border-
blue-200 dark:border-blue-800 rounded-md p-3">
                    <p className="text-sm text-blue-800 dark:text-blue-200">
                      <span className="font-medium">Total Days:</span>
{calculateDays()} day(s)
                    </p>
                  </div>
                )}

                {/* Reason */}
                <div>
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                    Reason for Leave *
                  </label>
                  <textarea
                    name="reason"
                    value={leaveData.reason}
                    onChange={handleInputChange}
                    rows={4}
                    placeholder="Please provide a detailed reason for your leave
(minimum 10 characters)"
                    className="w-full border border-gray-300 dark:border-gray-600
rounded-md px-3 py-2 bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-2 focus:ring-blue-500"
                    required
                    minLength={10}
                  />
                  <p className="text-xs text-gray-500 mt-1">
                    {leaveData.reason.length}/500 characters (minimum 10)
                  </p>
                </div>

                {/* Submit Button */}
                <div className="flex justify-end">
                  <button
                    type="submit"
                    disabled={submitting}
                    className="bg-blue-600 hover:bg-blue-700 disabled:bg-blue-400
text-white px-6 py-2 rounded-md font-medium transition duration-200 flex items-
center"
                  >
                    {submitting ? (
                      <>
                        <div className="animate-spin rounded-full h-4 w-4 border-
b-2 border-white mr-2"></div>
                        Submitting...
                      </>
                    ) : (
                      <>
                        <FaCalendarPlus className="mr-2" />
                        Submit Application
                      </>
                    )}
                  </button>
```

```
                </div>
              </form>
            </div>
          )}

          {/* Leave History Tab */}
          {activeTab === 'history' && (
            <div>
              <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-6">Leave History</h3>

              {loading ? (
                <div className="flex justify-center py-8">
                  <div className="animate-spin rounded-full h-8 w-8 border-b-2
border-blue-600"></div>
                </div>
              ) : myLeaves.length === 0 ? (
                <div className="text-center py-8 text-gray-500 dark:text-
gray-400">
                  <FaCalendarCheck className="mx-auto h-12 w-12 mb-4 opacity-50" /
>
                  <p>No leave applications found</p>
                </div>
              ) : (
                <div className="space-y-4">
                  {myLeaves.map((leave) => (
                    <div key={leave._id} className="border border-gray-200
dark:border-gray-700 rounded-lg p-4">
                      <div className="flex items-start justify-between">
                        <div className="flex-1">
                          <div className="flex items-center space-x-3 mb-2">
                            <h4 className="font-medium text-gray-900 dark:text-
white">
                              {leaveTypes.find(t => t.value ===
leave.leaveType)?.icon} {' '}
                              {leaveTypes.find(t => t.value ===
leave.leaveType)?.label}
                            </h4>
                            <span className={`inline-flex items-center px-2.5
py-0.5 rounded-full text-xs font-medium border ${statusColors[leave.status]}`}>
                              {statusIcons[leave.status]}
                              <span className="ml-1 capitalize">{leave.status}</
span>
                            </span>
                          </div>

                          <div className="grid grid-cols-1 md:grid-cols-3 gap-4
text-sm text-gray-600 dark:text-gray-400">
                            <div>
                              <span className="font-medium">Duration:</span>
{formatDate(leave.startDate)}
                              {leave.startDate !== leave.endDate && ` -
${formatDate(leave.endDate)}`}
                              {leave.isHalfDay && ` (${leave.halfDaySession}
half)`}
                            </div>
                            <div>
                              <span className="font-medium">Days:</span>
{leave.totalDays}
```

```
                            </div>
                            <div>
                              <span className="font-medium">Applied:</span>
{formatDate(leave.appliedDate)}
                            </div>
                          </div>

                          <div className="mt-2">
                            <span className="font-medium text-sm text-gray-600
dark:text-gray-400">Reason:</span>
                            <p className="text-sm text-gray-800 dark:text-
gray-200 mt-1">{leave.reason}</p>
                          </div>

                          {leave.rejectionReason && (
                            <div className="mt-2 p-2 bg-red-50 dark:bg-red-900/20
border border-red-200 dark:border-red-800 rounded">
                              <span className="font-medium text-sm text-red-800
dark:text-red-200">Rejection Reason:</span>
                              <p className="text-sm text-red-700 dark:text-
red-300 mt-1">{leave.rejectionReason}</p>
                            </div>
                          )}

                          {leave.approvedBy && leave.status === 'approved' && (
                            <div className="mt-2 text-sm text-green-600 dark:text-
green-400">
                              <span className="font-medium">Approved by:</span>
{leave.approvedBy.fullName} on {formatDate(leave.approvedDate)}
                            </div>
                          )}
                        </div>

                        {leave.status === 'pending' && (
                          <button
                            onClick={() => handleCancelLeave(leave._id)}
                            className="ml-4 text-red-600 hover:text-red-800 text-
sm font-medium"
                          >
                            Cancel
                          </button>
                        )}
                      </div>
                    </div>
                  ))}
                </div>
              )}
            </div>
          )}

          {/* Leave Balance Tab */}
          {activeTab === 'balance' && (
            <div>
              <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-6">Leave Balance ({new Date().getFullYear()})</h3>

              <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3
gap-6">
                {Object.entries(leaveBalance).map(([type, balance]) => {
```

```
                  const leaveTypeInfo = leaveTypes.find(t => t.value === type);
                  return (
                    <div key={type} className="bg-gradient-to-r from-blue-50 to-
indigo-50 dark:from-blue-900/20 dark:to-indigo-900/20 border border-blue-200
dark:border-blue-800 rounded-lg p-4">
                      <div className="flex items-center justify-between mb-3">
                        <h4 className="font-medium text-gray-900 dark:text-white
flex items-center">
                          <span className="mr-2">{leaveTypeInfo?.icon}</span>
                          {leaveTypeInfo?.label}
                        </h4>
                      </div>

                      <div className="space-y-2">
                        <div className="flex justify-between text-sm">
                          <span className="text-gray-600 dark:text-
gray-400">Total:</span>
                          <span className="font-medium text-gray-900 dark:text-
white">{balance.total} days</span>
                        </div>
                        <div className="flex justify-between text-sm">
                          <span className="text-gray-600 dark:text-
gray-400">Used:</span>
                          <span className="font-medium text-red-600 dark:text-
red-400">{balance.used} days</span>
                        </div>
                        <div className="flex justify-between text-sm">
                          <span className="text-gray-600 dark:text-
gray-400">Remaining:</span>
                          <span className="font-medium text-green-600 dark:text-
green-400">{balance.remaining} days</span>
                        </div>

                        {/* Progress bar */}
                        <div className="mt-3">
                          <div className="bg-gray-200 dark:bg-gray-700 rounded-
full h-2">
                            <div
                              className="bg-blue-600 h-2 rounded-full transition-
all duration-300"
                              style={{ width: `${(balance.used / balance.total) *
100}%` }}
                            ></div>
                          </div>
                        </div>
                      </div>
                    );
                  })}
                </div>
              </div>
            )}
          </div>
        </div>
      </div>
    );
};

export default LeaveManagement;
```

src/components/Employee/PayrollComponent.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../../context/AuthContext';
import api, { payrollAPI } from '../../services/api';
import { toast } from 'react-hot-toast';
import PayrollSummaryTable from './PayrollSummaryTable';
import {
  FaPlus,
  FaEye,
  FaCheck,
  FaTimes,
  FaDownload,
  FaEdit,
  FaMoneyBillWave,
  FaSearch,
  FaFilter,
  FaFileAlt,
  FaSpinner,
  FaTrash,
  FaCalculator,
  FaTable,
  FaList
} from 'react-icons/fa';

const PayrollComponent = () => {
  const { user } = useAuth();
  const [payrollRecords, setPayrollRecords] = useState([]);
  const [employees, setEmployees] = useState([]);
  const [loading, setLoading] = useState(true);
  const [selectedMonth, setSelectedMonth] = useState(new Date().getMonth() + 1);
  const [selectedYear, setSelectedYear] = useState(new Date().getFullYear());
  const [selectedEmployee, setSelectedEmployee] = useState('');
  const [showGenerateModal, setShowGenerateModal] = useState(false);
  const [showEditModal, setShowEditModal] = useState(false);
  const [showDeleteModal, setShowDeleteModal] = useState(false);
  const [editingPayroll, setEditingPayroll] = useState(null);
  const [deletingPayroll, setDeletingPayroll] = useState(null);
  const [processingPayroll, setProcessingPayroll] = useState(null);
  const [downloadingSlip, setDownloadingSlip] = useState(null);
  const [viewMode, setViewMode] = useState('detailed');
  const [statusFilter, setStatusFilter] = useState('');
  const [searchTerm, setSearchTerm] = useState('');

  const months = [
    { value: 1, label: 'January' },
    { value: 2, label: 'February' },
    { value: 3, label: 'March' },
    { value: 4, label: 'April' },
    { value: 5, label: 'May' },
    { value: 6, label: 'June' },
    { value: 7, label: 'July' },
    { value: 8, label: 'August' },
    { value: 9, label: 'September' },
    { value: 10, label: 'October' },
    { value: 11, label: 'November' },
    { value: 12, label: 'December' }
  ];
```

```
  const statusOptions = [
    { value: 'DRAFT', label: 'Draft', color: 'bg-gray-100 text-gray-800' },
    { value: 'APPROVED', label: 'Approved', color: 'bg-green-100 text-
green-800' },
    { value: 'PAID', label: 'Paid', color: 'bg-blue-100 text-blue-800' },
    { value: 'CANCELLED', label: 'Cancelled', color: 'bg-red-100 text-red-800' }
  ];

  useEffect(() => {
    fetchInitialData();
  }, [selectedMonth, selectedYear, selectedEmployee]);

  const fetchInitialData = async () => {
    try {
      setLoading(true);
      await Promise.all([
        fetchPayrollRecords(),
        fetchEmployees()
      ]);
    } catch (error) {
      console.error('Error fetching initial data:', error);
      toast.error('Failed to fetch data');
    } finally {
      setLoading(false);
    }
  };

  const fetchPayrollRecords = async () => {
    try {
      const filters = {};
      if (selectedMonth) filters.month = selectedMonth;
      if (selectedYear) filters.year = selectedYear;
      if (selectedEmployee) filters.employeeId = selectedEmployee;

      const response = await payrollAPI.getAll(filters);

      if (response.data.success) {
        setPayrollRecords(response.data.data);
      } else {
        toast.error('Failed to fetch payroll records');
      }
    } catch (error) {
      console.error('Error fetching payroll records:', error);
      toast.error('Failed to fetch payroll records');
    }
  };

  const fetchEmployees = async () => {
    try {
      const response = await api.get('/employees');
      if (response.data && Array.isArray(response.data)) {
        setEmployees(response.data);
      } else if (response.data.data && Array.isArray(response.data.data)) {
        setEmployees(response.data.data);
      }
    } catch (error) {
      console.error('Error fetching employees:', error);
      toast.error('Failed to fetch employees');
```

```
  }
};

const handleDownloadSalarySlip = async (payrollId) => {
  try {
    setDownloadingSlip(payrollId);
    const response = await payrollAPI.downloadSalarySlip(payrollId);

    // Create a blob from the PDF stream
    const blob = new Blob([response.data], { type: 'application/pdf' });
    const url = window.URL.createObjectURL(blob);

    // Create a link and trigger download
    const link = document.createElement('a');
    link.href = url;
    link.setAttribute('download', `salary-slip-${payrollId}.pdf`);
    document.body.appendChild(link);
    link.click();
    link.remove();

    // Cleanup
    window.URL.revokeObjectURL(url);
    toast.success('Salary slip downloaded successfully');
  } catch (error) {
    console.error('Error downloading salary slip:', error);
    toast.error('Failed to download salary slip');
  } finally {
    setDownloadingSlip(null);
  }
};

const handleDeletePayroll = async (payroll) => {
  try {
    setDeletingPayroll(payroll._id);
    const response = await payrollAPI.delete(payroll._id);
    if (response.data.success) {
      toast.success('Payroll deleted successfully');
      fetchPayrollRecords();
    } else {
      toast.error(response.data.message || 'Failed to delete payroll');
    }
  } catch (error) {
    console.error('Error deleting payroll:', error);
    toast.error(error.response?.data?.message || 'Failed to delete payroll');
  } finally {
    setDeletingPayroll(null);
    setShowDeleteModal(false);
  }
};

const handleEditPayroll = async (payroll) => {
  setEditingPayroll(payroll._id);
  setShowGenerateModal(true);
  setGenerateForm({
    employeeId: payroll.employeeId?._id || payroll.userId?._id,
    month: payroll.month,
    year: payroll.year,
    baseSalary: payroll.baseSalary,
    daysPresent: payroll.daysPresent,
```

```
    workingDays: payroll.workingDays,
    calculatedSalary: payroll.calculatedSalary,
    // Manual Allowances
    hra: payroll.hra,
    da: payroll.da,
    conveyanceAllowance: payroll.conveyanceAllowance,
    medicalAllowance: payroll.medicalAllowance,
    specialAllowance: payroll.specialAllowance,
    overtimeAmount: payroll.overtimeAmount,
    // Bonuses
    performanceBonus: payroll.performanceBonus,
    projectBonus: payroll.projectBonus,
    attendanceBonus: payroll.attendanceBonus,
    festivalBonus: payroll.festivalBonus,
    // Manual Deductions
    pf: payroll.pf,
    esi: payroll.esi,
    tax: payroll.tax,
    loan: payroll.loan,
    other: payroll.other,
    notes: payroll.notes
  });
};

const handleUpdatePayroll = async (e) => {
  e.preventDefault();
  try {
    setProcessingPayroll(editingPayroll);

    const updateData = {
      ...generateForm,
      baseSalary: parseFloat(generateForm.baseSalary) || 0,
      daysPresent: parseFloat(generateForm.daysPresent) || 0,
      workingDays: parseFloat(generateForm.workingDays) || 0,
      // Manual Allowances
      hra: parseFloat(generateForm.hra) || 0,
      da: parseFloat(generateForm.da) || 0,
      conveyanceAllowance: parseFloat(generateForm.conveyanceAllowance) || 0,
      medicalAllowance: parseFloat(generateForm.medicalAllowance) || 0,
      specialAllowance: parseFloat(generateForm.specialAllowance) || 0,
      overtimeAmount: parseFloat(generateForm.overtimeAmount) || 0,
      // Bonuses
      performanceBonus: parseFloat(generateForm.performanceBonus) || 0,
      projectBonus: parseFloat(generateForm.projectBonus) || 0,
      attendanceBonus: parseFloat(generateForm.attendanceBonus) || 0,
      festivalBonus: parseFloat(generateForm.festivalBonus) || 0,
      // Manual Deductions
      pf: parseFloat(generateForm.pf) || 0,
      esi: parseFloat(generateForm.esi) || 0,
      tax: parseFloat(generateForm.tax) || 0,
      loan: parseFloat(generateForm.loan) || 0,
      other: parseFloat(generateForm.other) || 0
    };

    const response = await payrollAPI.update(editingPayroll, updateData);

    if (response.data.success) {
      toast.success('Payroll updated successfully');
      setShowGenerateModal(false);
```

```
        resetForm();
        fetchPayrollRecords();
      } else {
        toast.error(response.data.message || 'Failed to update payroll');
      }
    } catch (error) {
      console.error('Error updating payroll:', error);
      toast.error(error.response?.data?.message || 'Failed to update payroll');
    } finally {
      setProcessingPayroll(null);
      setEditingPayroll(null);
    }
  };

  const resetForm = () => {
    setGenerateForm({
      employeeId: '',
      month: new Date().getMonth() + 1,
      year: new Date().getFullYear(),
      baseSalary: '',
      daysPresent: '',
      workingDays: '',
      calculatedSalary: '',
      hra: '',
      da: '',
      conveyanceAllowance: '',
      medicalAllowance: '',
      specialAllowance: '',
      overtimeAmount: '',
      performanceBonus: '',
      projectBonus: '',
      attendanceBonus: '',
      festivalBonus: '',
      pf: '',
      esi: '',
      tax: '',
      loan: '',
      other: '',
      notes: ''
    });
  };

  // Update the form submit handler
  const handleFormSubmit = (e) => {
    if (editingPayroll) {
      handleUpdatePayroll(e);
    } else {
      handleGeneratePayroll(e);
    }
  };

  const filteredPayrollRecords = payrollRecords.filter(payroll => {
    const matchesStatus = !statusFilter || payroll.status === statusFilter;
    const matchesSearch = !searchTerm ||
      (payroll.employeeId?.fullName ||
''').toLowerCase().includes(searchTerm.toLowerCase()) ||
      (payroll.userId?.fullName ||
''').toLowerCase().includes(searchTerm.toLowerCase());
```

```
    return matchesStatus && matchesSearch;
  });

  // Add this form state if not already present
  const [generateForm, setGenerateForm] = useState({
    employeeId: '',
    month: new Date().getMonth() + 1,
    year: new Date().getFullYear(),
    baseSalary: '',
    daysPresent: '',
    workingDays: '',
    calculatedSalary: '',
    // Manual Allowances
    hra: '',
    da: '',
    conveyanceAllowance: '',
    medicalAllowance: '',
    specialAllowance: '',
    overtimeAmount: '',
    // Bonuses
    performanceBonus: '',
    projectBonus: '',
    attendanceBonus: '',
    festivalBonus: '',
    // Manual Deductions
    pf: '',
    esi: '',
    tax: '',
    loan: '',
    other: '',
    notes: ''
  });

  // Add the salary calculation function
  const calculateSalary = (baseSalary, daysPresent) => {
    if (!baseSalary || !daysPresent) return 0;
    const calculatedAmount = (parseFloat(baseSalary) / 30) *
parseFloat(daysPresent);
    return calculatedAmount;
  };

  // Update calculated salary when base salary or days present changes
  useEffect(() => {
    if (generateForm.baseSalary && generateForm.daysPresent) {
      const calculated = calculateSalary(generateForm.baseSalary,
generateForm.daysPresent);
      setGenerateForm(prev => ({ ...prev, calculatedSalary:
calculated.toFixed(2) }));
    }
  }, [generateForm.baseSalary, generateForm.daysPresent]);

  const handleGeneratePayroll = async (e) => {
    e.preventDefault();
    try {
      setProcessingPayroll(generateForm.employeeId);

      // Convert empty strings to 0 for numeric fields
      const generateData = {
        ...generateForm,
```

```
      baseSalary: parseFloat(generateForm.baseSalary) || 0,
      daysPresent: parseFloat(generateForm.daysPresent) || 0,
      workingDays: parseFloat(generateForm.workingDays) || 0,
      // Manual Allowances
      hra: parseFloat(generateForm.hra) || 0,
      da: parseFloat(generateForm.da) || 0,
      conveyanceAllowance: parseFloat(generateForm.conveyanceAllowance) || 0,
      medicalAllowance: parseFloat(generateForm.medicalAllowance) || 0,
      specialAllowance: parseFloat(generateForm.specialAllowance) || 0,
      overtimeAmount: parseFloat(generateForm.overtimeAmount) || 0,
      // Bonuses
      performanceBonus: parseFloat(generateForm.performanceBonus) || 0,
      projectBonus: parseFloat(generateForm.projectBonus) || 0,
      attendanceBonus: parseFloat(generateForm.attendanceBonus) || 0,
      festivalBonus: parseFloat(generateForm.festivalBonus) || 0,
      // Manual Deductions
      pf: parseFloat(generateForm.pf) || 0,
      esi: parseFloat(generateForm.esi) || 0,
      tax: parseFloat(generateForm.tax) || 0,
      loan: parseFloat(generateForm.loan) || 0,
      other: parseFloat(generateForm.other) || 0
    };

    const response = await payrollAPI.generate(generateData);

    if (response.data.success) {
      toast.success('Payroll generated successfully');
      setShowGenerateModal(false);
      setGenerateForm({
        employeeId: '',
        month: new Date().getMonth() + 1,
        year: new Date().getFullYear(),
        baseSalary: '',
        daysPresent: '',
        workingDays: '',
        calculatedSalary: '',
        hra: '',
        da: '',
        conveyanceAllowance: '',
        medicalAllowance: '',
        specialAllowance: '',
        overtimeAmount: '',
        performanceBonus: '',
        projectBonus: '',
        attendanceBonus: '',
        festivalBonus: '',
        pf: '',
        esi: '',
        tax: '',
        loan: '',
        other: '',
        notes: ''
      });
      fetchPayrollRecords();
    } else {
      toast.error(response.data.message || 'Failed to generate payroll');
    }
  } catch (error) {
    console.error('Error generating payroll:', error);
```

```
      toast.error(error.response?.data?.message || 'Failed to generate payroll');
    } finally {
      setProcessingPayroll(null);
    }
  };

  if (loading) {
    return (
      <div className="flex justify-center items-center h-64">
        <FaSpinner className="animate-spin text-4xl text-blue-500" />
      </div>
    );
  }

  return (
    <div className="container mx-auto px-4 py-8">
      <div className="flex justify-between items-center mb-6">
        <h2 className="text-2xl font-bold text-gray-900 dark:text-white">
          Payroll Management
        </h2>
        <button
          onClick={() => setShowGenerateModal(true)}
          className="bg-indigo-600 text-white px-4 py-2 rounded-md hover:bg-
indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-
indigo-500"
        >
          <FaPlus className="inline-block mr-2" />
          Generate Payroll
        </button>
      </div>

      {/* Filters Section */}
      <div className="bg-white dark:bg-gray-800 rounded-lg shadow p-6 mb-6">
        <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              Month
            </label>
            <select
              value={selectedMonth}
              onChange={(e) => setSelectedMonth(parseInt(e.target.value))}
              className="block w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-500 focus:ring-indigo-500 dark:bg-gray-700 dark:border-
gray-600"
            >
              {months.map((month) => (
                <option key={month.value} value={month.value}>
                  {month.label}
                </option>
              ))}
            </select>
          </div>
          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              Year
            </label>
            <select
```

```
              value={selectedYear}
              onChange={(e) => setSelectedYear(parseInt(e.target.value))}
              className="block w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-500 focus:ring-indigo-500 dark:bg-gray-700 dark:border-
gray-600"
            >
              {Array.from({ length: 5 }, (_, i) => new Date().getFullYear() - 2 +
i).map((year) => (
                <option key={year} value={year}>
                  {year}
                </option>
              ))}
            </select>
          </div>
          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              Employee
            </label>
            <select
              value={selectedEmployee}
              onChange={(e) => setSelectedEmployee(e.target.value)}
              className="block w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-500 focus:ring-indigo-500 dark:bg-gray-700 dark:border-
gray-600"
            >
              <option value="">All Employees</option>
              {employees.map((employee) => (
                <option key={employee._id} value={employee._id}>
                  {employee.fullName}
                </option>
              ))}
            </select>
          </div>
          <div>
            <label className="block text-sm font-medium text-gray-700 dark:text-
gray-300 mb-1">
              Status
            </label>
            <select
              value={statusFilter}
              onChange={(e) => setStatusFilter(e.target.value)}
              className="block w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-500 focus:ring-indigo-500 dark:bg-gray-700 dark:border-
gray-600"
            >
              <option value="">All Status</option>
              {statusOptions.map((option) => (
                <option key={option.value} value={option.value}>
                  {option.label}
                </option>
              ))}
            </select>
          </div>
        </div>
      </div>

      {/* Generate Payroll Modal */}
      {showGenerateModal && (
```

```
        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center p-4 z-50">
          <div className="bg-white dark:bg-gray-800 rounded-lg p-6 w-full max-
w-2xl max-h-[80vh] overflow-y-auto">
            <h2 className="text-xl font-bold mb-4 text-gray-900 dark:text-white">
              {editingPayroll ? 'Edit Payroll' : 'Generate Payroll'}
            </h2>
            <form onSubmit={handleFormSubmit}>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div className="mb-4 col-span-2">
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                    Employee *
                  </label>
                  <select
                    value={generateForm.employeeId}
                    onChange={(e) => setGenerateForm({...generateForm,
employeeId: e.target.value})}
                    className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                    required
                  >
                    <option value="">Select Employee</option>
                    {employees.map(employee => (
                      <option key={employee._id} value={employee._id}>
                        {employee.fullName}
                      </option>
                    ))}
                  </select>
                </div>

                <div>
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                    Month *
                  </label>
                  <select
                    value={generateForm.month}
                    onChange={(e) => setGenerateForm({...generateForm, month:
parseInt(e.target.value)})}
                    className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                    required
                  >
                    {months.map(month => (
                      <option key={month.value} value={month.value}>
                        {month.label}
                      </option>
                    ))}
                  </select>
                </div>

                <div>
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                    Year *
                  </label>
```

```
                  <select
                    value={generateForm.year}
                    onChange={(e) => setGenerateForm({...generateForm, year:
parseInt(e.target.value)})}
                    className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                    required
                  >
                    {Array.from({ length: 5 }, (_, i) => new Date().getFullYear()
- 2 + i).map(year => (
                      <option key={year} value={year}>
                        {year}
                      </option>
                    ))}
                  </select>
                </div>

                <div className="col-span-2 mb-4">
                  <h3 className="text-lg font-semibold text-gray-900 dark:text-
white mb-3 flex items-center gap-2">
                    <FaCalculator />
                    Salary Calculation
                  </h3>
                  <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                        Base Salary (Monthly) *
                      </label>
                      <input
                        type="number"
                        value={generateForm.baseSalary}
                        onChange={(e) => setGenerateForm({...generateForm,
baseSalary: e.target.value})}
                        className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                        min="0"
                        step="0.01"
                        placeholder="Enter base salary"
                        required
                      />
                    </div>

                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                        Days Present *
                      </label>
                      <input
                        type="number"
                        value={generateForm.daysPresent}
                        onChange={(e) => setGenerateForm({...generateForm,
daysPresent: e.target.value})}
                        className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                        min="0"
```

```
                      max="31"
                      step="1"
                      placeholder="Enter days present"
                      required
                    />
                  </div>

                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      Working Days *
                    </label>
                    <input
                      type="number"
                      value={generateForm.workingDays}
                      onChange={(e) => setGenerateForm({...generateForm,
workingDays: e.target.value})}
                      className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                      min="0"
                      max="31"
                      step="1"
                      placeholder="Enter working days"
                      required
                    />
                  </div>
                </div>
              </div>

              {/* Manual Allowances */}
              <div className="col-span-2 mb-4">
                <h3 className="text-lg font-semibold text-gray-900 dark:text-
white mb-3">Manual Allowances</h3>
                <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">HRA</label>
                    <input
                      type="number"
                      value={generateForm.hra}
                      onChange={(e) => setGenerateForm({...generateForm, hra:
e.target.value})}
                      className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                      min="0"
                      step="0.01"
                      placeholder="Enter HRA"
                    />
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">DA</label>
                    <input
                      type="number"
                      value={generateForm.da}
                      onChange={(e) => setGenerateForm({...generateForm, da:
e.target.value})}
```

```
                        className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                        min="0"
                        step="0.01"
                        placeholder="Enter DA"
                      />
                    </div>
                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                        Conveyance Allowance
                      </label>
                      <input
                        type="number"
                        value={generateForm.conveyanceAllowance}
                        onChange={(e) => setGenerateForm({...generateForm,
conveyanceAllowance: e.target.value})}
                        className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                        min="0"
                        step="0.01"
                        placeholder="Enter conveyance allowance"
                      />
                    </div>
                  </div>
                </div>

                {/* Bonuses */}
                <div className="col-span-2 mb-4">
                  <h3 className="text-lg font-semibold text-gray-900 dark:text-
white mb-3 flex items-center gap-2">
                    <FaMoneyBillWave />
                    Bonuses
                  </h3>
                  <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                        Performance Bonus
                      </label>
                      <input
                        type="number"
                        value={generateForm.performanceBonus}
                        onChange={(e) => setGenerateForm({...generateForm,
performanceBonus: e.target.value})}
                        className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                        min="0"
                        step="0.01"
                        placeholder="Enter performance bonus"
                      />
                    </div>
                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                        Project Bonus
```

```
                    </label>
                    <input
                      type="number"
                      value={generateForm.projectBonus}
                      onChange={(e) => setGenerateForm({...generateForm,
projectBonus: e.target.value})}
                      className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                      min="0"
                      step="0.01"
                      placeholder="Enter project bonus"
                    />
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      Attendance Bonus
                    </label>
                    <input
                      type="number"
                      value={generateForm.attendanceBonus}
                      onChange={(e) => setGenerateForm({...generateForm,
attendanceBonus: e.target.value})}
                      className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                      min="0"
                      step="0.01"
                      placeholder="Enter attendance bonus"
                    />
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      Festival Bonus
                    </label>
                    <input
                      type="number"
                      value={generateForm.festivalBonus}
                      onChange={(e) => setGenerateForm({...generateForm,
festivalBonus: e.target.value})}
                      className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                      min="0"
                      step="0.01"
                      placeholder="Enter festival bonus"
                    />
                  </div>
                </div>
              </div>

              {/* Deductions */}
              <div className="col-span-2 mb-4">
                <h3 className="text-lg font-semibold text-gray-900 dark:text-
white mb-3 flex items-center gap-2">
                  <FaCalculator />
                  Deductions
```

```
                  </h3>
                  <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                        PF
                      </label>
                      <input
                        type="number"
                        value={generateForm.pf}
                        onChange={(e) => setGenerateForm({...generateForm, pf:
e.target.value})}
                        className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                        min="0"
                        step="0.01"
                        placeholder="Enter PF amount"
                      />
                    </div>
                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                        ESI
                      </label>
                      <input
                        type="number"
                        value={generateForm.esi}
                        onChange={(e) => setGenerateForm({...generateForm, esi:
e.target.value})}
                        className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                        min="0"
                        step="0.01"
                        placeholder="Enter ESI amount"
                      />
                    </div>
                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                        Tax
                      </label>
                      <input
                        type="number"
                        value={generateForm.tax}
                        onChange={(e) => setGenerateForm({...generateForm, tax:
e.target.value})}
                        className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                        min="0"
                        step="0.01"
                        placeholder="Enter tax amount"
                      />
                    </div>
                    <div>
                      <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
```

```
                      Loan
                    </label>
                    <input
                      type="number"
                      value={generateForm.loan}
                      onChange={(e) => setGenerateForm({...generateForm, loan:
e.target.value})}
                      className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                      min="0"
                      step="0.01"
                      placeholder="Enter loan deduction"
                    />
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                      Other Deductions
                    </label>
                    <input
                      type="number"
                      value={generateForm.other}
                      onChange={(e) => setGenerateForm({...generateForm, other:
e.target.value})}
                      className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                      min="0"
                      step="0.01"
                      placeholder="Enter other deductions"
                    />
                  </div>
                </div>
              </div>

              {/* Notes */}
              <div className="col-span-2 mb-4">
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                  Notes
                </label>
                <textarea
                  value={generateForm.notes}
                  onChange={(e) => setGenerateForm({...generateForm, notes:
e.target.value})}
                  className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:bg-gray-700 dark:text-white"
                  rows="3"
                  placeholder="Enter any additional notes"
                />
              </div>

              {/* Form Actions */}
              <div className="col-span-2 flex justify-end space-x-2 mt-4">
                <button
                  type="button"
                  onClick={() => {
```

```
                    setShowGenerateModal(false);
                    setEditingPayroll(null);
                    resetForm();
                  }}
                  className="px-4 py-2 text-sm font-medium text-gray-700 bg-
white border border-gray-300 rounded-md hover:bg-gray-50 focus:outline-none
focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"
                >
                  Cancel
                </button>
                <button
                  type="submit"
                  disabled={processingPayroll}
                  className="px-4 py-2 text-sm font-medium text-white bg-
indigo-600 border border-transparent rounded-md hover:bg-indigo-700 focus:outline-
none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"
                >
                  {processingPayroll ? (
                    <>
                      <FaSpinner className="inline-block animate-spin mr-2" />
                      {editingPayroll ? 'Updating...' : 'Generating...'}
                    </>
                  ) : (
                    editingPayroll ? 'Update Payroll' : 'Generate Payroll'
                  )}
                </button>
              </div>
            </div>
          </form>
        </div>
      </div>
    )}

    {/* Delete Confirmation Modal */}
    {showDeleteModal && deletingPayroll && (
      <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
        <div className="bg-white dark:bg-gray-800 rounded-lg p-6 w-full max-w-
md">
          <h2 className="text-xl font-bold mb-4 text-gray-900 dark:text-white">
            Confirm Delete
          </h2>
          <p className="text-gray-600 dark:text-gray-400 mb-4">
            Are you sure you want to delete this payroll record? This action
cannot be undone.
          </p>
          <div className="flex justify-end space-x-2">
            <button
              onClick={() => {
                setShowDeleteModal(false);
                setDeletingPayroll(null);
              }}
              className="px-4 py-2 text-sm font-medium text-gray-700 bg-white
border border-gray-300 rounded-md hover:bg-gray-50 focus:outline-none
focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500"
            >
              Cancel
            </button>
            <button
```

```
                onClick={() => handleDeletePayroll(deletingPayroll)}
                className="px-4 py-2 text-sm font-medium text-white bg-red-600
border border-transparent rounded-md hover:bg-red-700 focus:outline-none
focus:ring-2 focus:ring-offset-2 focus:ring-red-500"
              >
                {deletingPayroll === deletingPayroll._id ? (
                  <>
                    <FaSpinner className="inline-block animate-spin mr-2" />
                    Deleting...
                  </>
                ) : (
                  'Delete'
                )}
              </button>
            </div>
          </div>
        </div>
      )}

      {/* Content Section */}
      {loading ? (
        <div className="flex justify-center items-center py-12">
          <FaSpinner className="animate-spin text-4xl text-indigo-600" />
        </div>
      ) : filteredPayrollRecords.length === 0 ? (
        <div className="text-center py-12">
          <p className="text-gray-500 dark:text-gray-400">No payroll records
found.</p>
        </div>
      ) : (
        <PayrollSummaryTable
          payrollRecords={filteredPayrollRecords}
          onDownloadSlip={handleDownloadSalarySlip}
          downloadingSlip={downloadingSlip}
          onDelete={handleDeletePayroll}
          onEdit={handleEditPayroll}
          deletingPayroll={deletingPayroll}
          editingPayroll={editingPayroll}
          userRole={user.role}
        />
      )}
    </div>
  );
};

export default PayrollComponent;
```

src/components/Employee/PayrollSummaryTable.jsx

```
import React, { useMemo } from 'react';
import { FaDownload, FaTrash, FaEdit, FaSpinner } from 'react-icons/fa';

const PayrollSummaryTable = ({
  payrollRecords,
  onDownloadSlip,
  downloadingSlip,
  onDelete,
  onEdit,
```

```
  deletingPayroll,
  editingPayroll,
  userRole = 'Employee' // Default to Employee if not provided
}) => {
  // Calculate totals for all numeric columns
  const totals = useMemo(() => {
    return payrollRecords.reduce((acc, record) => {
      acc.baseSalary += parseFloat(record.baseSalary) || 0;
      acc.calculatedSalary += parseFloat(record.calculatedSalary) || 0;
      acc.hra += parseFloat(record.hra) || 0;
      acc.da += parseFloat(record.da) || 0;
      acc.conveyanceAllowance += parseFloat(record.conveyanceAllowance) || 0;
      acc.medicalAllowance += parseFloat(record.medicalAllowance) || 0;
      acc.specialAllowance += parseFloat(record.specialAllowance) || 0;
      acc.overtimeAmount += parseFloat(record.overtimeAmount) || 0;
      acc.performanceBonus += parseFloat(record.performanceBonus) || 0;
      acc.projectBonus += parseFloat(record.projectBonus) || 0;
      acc.attendanceBonus += parseFloat(record.attendanceBonus) || 0;
      acc.festivalBonus += parseFloat(record.festivalBonus) || 0;
      acc.pf += parseFloat(record.pf) || 0;
      acc.esi += parseFloat(record.esi) || 0;
      acc.tax += parseFloat(record.tax) || 0;
      acc.loan += parseFloat(record.loan) || 0;
      acc.other += parseFloat(record.other) || 0;
      acc.netSalary += parseFloat(record.netSalary) || 0;
      return acc;
    }, {
      baseSalary: 0,
      calculatedSalary: 0,
      hra: 0,
      da: 0,
      conveyanceAllowance: 0,
      medicalAllowance: 0,
      specialAllowance: 0,
      overtimeAmount: 0,
      performanceBonus: 0,
      projectBonus: 0,
      attendanceBonus: 0,
      festivalBonus: 0,
      pf: 0,
      esi: 0,
      tax: 0,
      loan: 0,
      other: 0,
      netSalary: 0
    });
  }, [payrollRecords]);

  const formatCurrency = (amount) => {
    return new Intl.NumberFormat('en-IN', {
      style: 'currency',
      currency: 'INR',
      minimumFractionDigits: 0,
      maximumFractionDigits: 0
    }).format(amount);
  };

  const getStatusColor = (status) => {
    switch (status) {
```

```
      case 'APPROVED':
        return 'text-green-600 bg-green-100';
      case 'PAID':
        return 'text-blue-600 bg-blue-100';
      case 'CANCELLED':
        return 'text-red-600 bg-red-100';
      default:
        return 'text-gray-600 bg-gray-100';
    }
  };

  return (
    <div className="overflow-x-auto">
      <table className="min-w-full bg-white dark:bg-gray-800 rounded-lg overflow-
hidden">
        <thead className="bg-gray-50 dark:bg-gray-700">
          <tr>
            <th className="px-4 py-3 text-left text-xs font-medium text-gray-500
dark:text-gray-300 uppercase tracking-wider">Employee</th>
            <th className="px-4 py-3 text-left text-xs font-medium text-gray-500
dark:text-gray-300 uppercase tracking-wider">Status</th>
            <th className="px-4 py-3 text-right text-xs font-medium text-gray-500
dark:text-gray-300 uppercase tracking-wider">Base Salary</th>
            <th className="px-4 py-3 text-right text-xs font-medium text-gray-500
dark:text-gray-300 uppercase tracking-wider">Days Present</th>
            <th className="px-4 py-3 text-right text-xs font-medium text-gray-500
dark:text-gray-300 uppercase tracking-wider">Allowances</th>
            <th className="px-4 py-3 text-right text-xs font-medium text-gray-500
dark:text-gray-300 uppercase tracking-wider">Bonuses</th>
            <th className="px-4 py-3 text-right text-xs font-medium text-gray-500
dark:text-gray-300 uppercase tracking-wider">Deductions</th>
            <th className="px-4 py-3 text-right text-xs font-medium text-gray-500
dark:text-gray-300 uppercase tracking-wider">Net Salary</th>
            <th className="px-4 py-3 text-center text-xs font-medium text-
gray-500 dark:text-gray-300 uppercase tracking-wider">
              Actions
            </th>
          </tr>
        </thead>
        <tbody className="divide-y divide-gray-200 dark:divide-gray-600">
          {payrollRecords.map((record) => {
            const totalAllowances = (
              parseFloat(record.hra) +
              parseFloat(record.da) +
              parseFloat(record.conveyanceAllowance) +
              parseFloat(record.medicalAllowance) +
              parseFloat(record.specialAllowance) +
              parseFloat(record.overtimeAmount)
            ) || 0;

            const totalBonuses = (
              parseFloat(record.performanceBonus) +
              parseFloat(record.projectBonus) +
              parseFloat(record.attendanceBonus) +
              parseFloat(record.festivalBonus)
            ) || 0;

            const totalDeductions = (
              parseFloat(record.pf) +
```

```
              parseFloat(record.esi) +
              parseFloat(record.tax) +
              parseFloat(record.loan) +
              parseFloat(record.other)
            ) || 0;

            // Get employee name from either employeeId object or userId object
            const employeeName = record.employeeId?.fullName ||
record.userId?.fullName || 'Unknown';
            const employeeId = record.employeeId?._id || record.userId?._id || 'N/
A';

            return (
              <tr key={record._id} className="hover:bg-gray-50 dark:hover:bg-
gray-600">
                <td className="px-4 py-3 whitespace-nowrap">
                  <div className="text-sm font-medium text-gray-900 dark:text-
white">{employeeName}</div>
                  <div className="text-sm text-gray-500 dark:text-
gray-400">{employeeId}</div>
                </td>
                <td className="px-4 py-3 whitespace-nowrap">
                  <span className={`px-2 inline-flex text-xs leading-5 font-
semibold rounded-full ${getStatusColor(record.status)}`}>
                    {record.status}
                  </span>
                </td>
                <td className="px-4 py-3 text-right whitespace-nowrap text-sm
text-gray-900 dark:text-white">
                  {formatCurrency(record.baseSalary)}
                </td>
                <td className="px-4 py-3 text-right whitespace-nowrap text-sm
text-gray-900 dark:text-white">
                  {record.daysPresent}/{record.workingDays}
                </td>
                <td className="px-4 py-3 text-right whitespace-nowrap text-sm
text-gray-900 dark:text-white">
                  {formatCurrency(totalAllowances)}
                </td>
                <td className="px-4 py-3 text-right whitespace-nowrap text-sm
text-gray-900 dark:text-white">
                  {formatCurrency(totalBonuses)}
                </td>
                <td className="px-4 py-3 text-right whitespace-nowrap text-sm
text-gray-900 dark:text-white">
                  {formatCurrency(totalDeductions)}
                </td>
                <td className="px-4 py-3 text-right whitespace-nowrap text-sm
font-medium text-gray-900 dark:text-white">
                  {formatCurrency(record.netSalary)}
                </td>
                <td className="px-4 py-3 text-center whitespace-nowrap text-sm
text-gray-500 dark:text-gray-400">
                  <div className="flex justify-center space-x-2">
                    {/* Download Button */}
                    <button
                      onClick={() => onDownloadSlip(record._id)}
                      disabled={downloadingSlip === record._id}
                      className="text-blue-600 hover:text-blue-800 dark:text-
```

```
blue-400 dark:hover:text-blue-300"
                      title="Download Salary Slip"
                    >
                      {downloadingSlip === record._id ? (
                        <FaSpinner className="animate-spin h-4 w-4" />
                      ) : (
                        <FaDownload className="h-4 w-4" />
                      )}
                    </button>

                    {/* Edit Button - Only show for Admin/HR */}
                    {(userRole === 'Admin' || userRole === 'HR') && (
                      <button
                        onClick={() => onEdit(record)}
                        disabled={editingPayroll === record._id}
                        className="text-yellow-600 hover:text-yellow-800
dark:text-yellow-400 dark:hover:text-yellow-300"
                        title="Edit Payroll"
                      >
                        {editingPayroll === record._id ? (
                          <FaSpinner className="animate-spin h-4 w-4" />
                        ) : (
                          <FaEdit className="h-4 w-4" />
                        )}
                      </button>
                    )}

                    {/* Delete Button - Only show for Admin/HR */}
                    {(userRole === 'Admin' || userRole === 'HR') && (
                      <button
                        onClick={() => onDelete(record)}
                        disabled={deletingPayroll === record._id}
                        className="text-red-600 hover:text-red-800 dark:text-
red-400 dark:hover:text-red-300"
                        title="Delete Payroll"
                      >
                        {deletingPayroll === record._id ? (
                          <FaSpinner className="animate-spin h-4 w-4" />
                        ) : (
                          <FaTrash className="h-4 w-4" />
                        )}
                      </button>
                    )}
                  </div>
                </td>
              </tr>
            );
          })}

          {/* Totals row */}
          <tr className="bg-gray-50 dark:bg-gray-700 font-semibold">
            <td className="px-4 py-3 whitespace-nowrap text-sm text-gray-900
dark:text-white" colSpan={2}>Total</td>
            <td className="px-4 py-3 text-right whitespace-nowrap text-sm text-
gray-900 dark:text-white">
              {formatCurrency(totals.baseSalary)}
            </td>
            <td className="px-4 py-3 text-right whitespace-nowrap text-sm text-
gray-900 dark:text-white">-</td>
```

```
            <td className="px-4 py-3 text-right whitespace-nowrap text-sm text-
gray-900 dark:text-white">
              {formatCurrency(totals.hra + totals.da + totals.conveyanceAllowance
+ totals.medicalAllowance + totals.specialAllowance + totals.overtimeAmount)}
            </td>
            <td className="px-4 py-3 text-right whitespace-nowrap text-sm text-
gray-900 dark:text-white">
              {formatCurrency(totals.performanceBonus + totals.projectBonus +
totals.attendanceBonus + totals.festivalBonus)}
            </td>
            <td className="px-4 py-3 text-right whitespace-nowrap text-sm text-
gray-900 dark:text-white">
              {formatCurrency(totals.pf + totals.esi + totals.tax + totals.loan +
totals.other)}
            </td>
            <td className="px-4 py-3 text-right whitespace-nowrap text-sm text-
gray-900 dark:text-white font-bold">
              {formatCurrency(totals.netSalary)}
            </td>
            <td className="px-4 py-3 text-center whitespace-nowrap text-sm text-
gray-500">-</td>
          </tr>
        </tbody>
      </table>
    </div>
  );
};

export default PayrollSummaryTable;
```

src/components/Footer.jsx

```
import React from "react";Ð
Ð
const Footer = () => {Ð
  return (Ð
    <footer className="bg-gray-800 text-white py-6">Ð
      <div className="container mx-auto px-4">Ð
        <div className="flex flex-col md:flex-row justify-between items-center">Ð
          <div className="mb-4 md:mb-0">Ð
            <h3 className="text-xl font-bold">CRM Platform</h3>Ð
            <p className="text-gray-400 dark:text-gray-500 mt-1">Streamlining
your business relationships</p>Ð
          </div>Ð
          Ð
          <div className="flex space-x-8">Ð
            <div>Ð
              <h4 className="font-semibold mb-2">Resources</h4>Ð
              <ul className="space-y-1">Ð
                <li><a href="#" className="text-gray-400 dark:text-gray-500
hover:text-white transition">Documentation</a></li>Ð
                <li><a href="#" className="text-gray-400 dark:text-gray-500
hover:text-white transition">Help Center</a></li>Ð
                <li><a href="#tutorials" className="text-gray-400 dark:text-
gray-500 hover:text-white transition">Tutorials</a></li>Ð
                <li><a href="#countries" className="text-gray-400 dark:text-
gray-500 hover:text-white transition">Countries</a></li>Ð
              </ul>Ð
```

```
            </div>Đ
            Đ
            <div>Đ
              <h4 className="font-semibold mb-2">Company</h4>Đ
              <ul className="space-y-1">Đ
                <li><a href="#" className="text-gray-400 dark:text-gray-500
hover:text-white transition">About Us</a></li>Đ
                <li><a href="#" className="text-gray-400 dark:text-gray-500
hover:text-white transition">Careers</a></li>Đ
                <li><a href="#management-contacts" className="text-gray-400
dark:text-gray-500 hover:text-white transition">Contact</a></li>Đ
              </ul>Đ
            </div>Đ
          </div>Đ
        </div>Đ
        Đ
        <div className="border-t border-gray-700 mt-6 pt-6 text-center md:text-
left">Đ
          <p>&copy; {new Date().getFullYear()} CRM Platform. All Rights
Reserved.</p>Đ
        </div>Đ
      </div>Đ
    </footer>Đ
  );Đ
};Đ
Đ
export default Footer;Đ
```

src/components/Layout/Layout.jsx

```
import React from "react";Đ
import Navbar from "../Navbar";Đ
import Footer from "../Footer";Đ
Đ
const Layout = ({ children }) => {Đ
  return (Đ
    <div className="min-h-screen flex flex-col">Đ
      <Navbar />Đ
      <main className="flex-grow">Đ
        {children}Đ
      </main>Đ
      <Footer />Đ
    </div>Đ
  );Đ
};Đ
Đ
export default Layout;
```

src/components/Layout/Sidebar.jsx

```
import React from 'react';
import { Link, useLocation } from 'react-router-dom';
import { useAuth } from '../../context/AuthContext';
import { FaHome, FaUser, FaUsers, FaChartLine, FaClipboardList, FaChartBar,
FaCog, FaFileImport, FaCalendarCheck, FaUserCog, FaClock, FaUserTie, FaFileAlt,
FaHistory } from 'react-icons/fa';
import ThemeToggle from '../ThemeToggle';
```

```
const Sidebar = () => {
  const location = useLocation();
  const { user, logout } = useAuth();

  return (
    <div className="h-screen bg-slate-800 dark:bg-slate-900 text-white w-64 flex-
shrink-0 hidden md:block transition-all duration-200 ease-out border-r border-
slate-700 dark:border-slate-700">
      <div className="flex flex-col h-full">
        <div className="px-4 py-6">
          <h1 className="text-2xl font-bold">CRM Dashboard</h1>
          <p className="text-sm text-gray-400 dark:text-gray-500 mt-1">Welcome,
{user?.name}</p>
        </div>

        <nav className="flex-1 px-2 py-4">
          <ul className="space-y-2">
            <li>
              <Link
                to="/"
                className={`${
                  location.pathname === '/'
                    ? 'bg-gray-900 text-white'
                    : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                } group flex items-center px-2 py-2 text-base font-medium rounded-
md`}
              >
                <FaHome
                  className={`${
                    location.pathname === '/'
                      ? 'text-gray-300 dark:text-gray-400'
                      : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                  } mr-3 flex-shrink-0 h-6 w-6`}
                />
                Home
              </Link>
            </li>

            {/* Manager Dashboard - Only visible to Manager and Admin */}
            {(user?.role === 'Manager' || user?.role === 'Admin') && (
              <li>
                <Link
                  to="/manager"
                  className={`${
                    location.pathname === '/manager'
                      ? 'bg-gray-900 text-white'
                      : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                  } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                >
                  <FaUserCog
                    className={`${
                      location.pathname === '/manager'
                        ? 'text-gray-300 dark:text-gray-400'
                        : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
```

```
                  } mr-3 flex-shrink-0 h-6 w-6`}
                />
                Manager Dashboard
              </Link>
            </li>
          )}

          {/* Lead Management - Only visible to Lead Person, Manager, and Admin
*/}
          {(user?.role === 'Lead Person' || user?.role === 'Manager' ||
user?.role === 'Admin') && (
            <li>
              <Link
                to="/leads"
                className={`${
                  location.pathname === '/leads'
                    ? 'bg-gray-900 text-white'
                    : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
              >
                <FaClipboardList
                  className={`${
                    location.pathname === '/leads'
                      ? 'text-gray-300 dark:text-gray-400'
                      : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                  } mr-3 flex-shrink-0 h-6 w-6`}
                />
                Leads
              </Link>
            </li>
          )}

          {/* Sales Management - Only visible to Sales Person, Manager, and
Admin */}
          {(user?.role === 'Sales Person' || user?.role === 'Manager' ||
user?.role === 'Admin') && (
            <li>
              <Link
                to="/sales"
                className={`${
                  location.pathname === '/sales'
                    ? 'bg-gray-900 text-white'
                    : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
              >
                <FaChartBar
                  className={`${
                    location.pathname === '/sales'
                      ? 'text-gray-300 dark:text-gray-400'
                      : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                  } mr-3 flex-shrink-0 h-6 w-6`}
                />
                Sales
```

```
              </Link>
            </li>
          )}

          {/* Create Sale for Lead Person - Only visible to Sales Person,
Manager, and Admin */}
          {(user?.role === 'Sales Person' || user?.role === 'Manager' ||
user?.role === 'Admin') && (
            <li>
              <Link
                to="/create-sale"
                className={`${
                  location.pathname === '/create-sale'
                    ? 'bg-gray-900 text-white'
                    : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
              >
                <FaChartLine
                  className={`${
                    location.pathname === '/create-sale'
                      ? 'text-gray-300 dark:text-gray-400'
                      : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                  } mr-3 flex-shrink-0 h-6 w-6`}
                />
                Create Sale for Lead
              </Link>
            </li>
          )}

          {/* Task Management - Visible to all authenticated users */}
          {user && (
            <li>
              <Link
                to="/tasks"
                className={`${
                  location.pathname === '/tasks'
                    ? 'bg-gray-900 text-white'
                    : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
              >
                <FaCalendarCheck
                  className={`${
                    location.pathname === '/tasks'
                      ? 'text-gray-300 dark:text-gray-400'
                      : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                  } mr-3 flex-shrink-0 h-6 w-6`}
                />
                Task Management
              </Link>
            </li>
          )}

          {/* Lead Sales Sheet - Only visible to Lead Person, Manager, and
```

```
Admin */}
            {(user?.role === 'Lead Person' || user?.role === 'Manager' ||
user?.role === 'Admin') && (
              <li>
                <Link
                  to="/lead-sales-sheet"
                  className={`${
                    location.pathname === '/lead-sales-sheet'
                      ? 'bg-gray-900 text-white'
                      : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                  } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                >
                  <FaChartLine
                    className={`${
                      location.pathname === '/lead-sales-sheet'
                        ? 'text-gray-300 dark:text-gray-400'
                        : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                    } mr-3 flex-shrink-0 h-6 w-6`}
                  />
                  My Sales Sheet
                </Link>
              </li>
            )}

            {/* Update Sales - Only visible to Manager and Admin, removed for
Lead Person */}
            {(user?.role === 'Manager' || user?.role === 'Admin') && (
              <li>
                <Link
                  to="/update-sales"
                  className={`${
                    location.pathname === '/update-sales'
                      ? 'bg-gray-900 text-white'
                      : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                  } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                >
                  <FaChartBar
                    className={`${
                      location.pathname === '/update-sales'
                        ? 'text-gray-300 dark:text-gray-400'
                        : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                    } mr-3 flex-shrink-0 h-6 w-6`}
                  />
                  Update Sales
                </Link>
              </li>
            )}

            {/* Employee Management - Only visible to HR, Manager, and Admin */}
            {(user?.role === 'HR' || user?.role === 'Manager' || user?.role ===
'Admin') && (
              <li>
                <Link
```

```
                  to="/employees"
                  className={`${
                    location.pathname === '/employees'
                      ? 'bg-gray-900 text-white'
                      : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                  } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                >
                  <FaUserTie
                    className={`${
                      location.pathname === '/employees'
                        ? 'text-gray-300 dark:text-gray-400'
                        : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                    } mr-3 flex-shrink-0 h-6 w-6`}
                  />
                  Employee Management
                </Link>
              </li>
            )}

            {/* Document Management - Visible to Employees only (Admin/Manager
can use Employee Management) */}
            {user && user.role === 'Employee' && (
              <li>
                <Link
                  to="/documents"
                  className={`${
                    location.pathname === '/documents'
                      ? 'bg-gray-900 text-white'
                      : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                  } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                >
                  <FaFileAlt
                    className={`${
                      location.pathname === '/documents'
                        ? 'text-gray-300 dark:text-gray-400'
                        : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                    } mr-3 flex-shrink-0 h-6 w-6`}
                  />
                  My Documents
                </Link>
              </li>
            )}

            {/* Profile - Visible to all */}
            <li>
              <Link
                to="/profile"
                className={`${
                  location.pathname === '/profile'
                    ? 'bg-gray-900 text-white'
                    : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                } group flex items-center px-2 py-2 text-base font-medium rounded-
md`}
```

```
              >
                <FaUser
                  className={`${
                    location.pathname === '/profile'
                      ? 'text-gray-300 dark:text-gray-400'
                      : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                  } mr-3 flex-shrink-0 h-6 w-6`}
                />
                Profile
              </Link>
            </li>

            {/* Admin Section - Only visible to Admin */}
            {user?.role === 'Admin' && (
              <>
                <li className="mt-8 mb-2">
                  <h3 className="px-3 text-xs font-semibold text-gray-400
dark:text-gray-500 uppercase tracking-wider">
                    Admin
                  </h3>
                </li>
                <li>
                  <Link
                    to="/admin/dashboard"
                    className={`${
                      location.pathname === '/admin/dashboard'
                        ? 'bg-gray-900 text-white'
                        : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                    } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                  >
                    <FaChartLine
                      className={`${
                        location.pathname === '/admin/dashboard'
                          ? 'text-gray-300 dark:text-gray-400'
                          : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                      } mr-3 flex-shrink-0 h-6 w-6`}
                    />
                    Dashboard
                  </Link>
                </li>
                <li>
                  <Link
                    to="/admin/users"
                    className={`${
                      location.pathname === '/admin/users'
                        ? 'bg-gray-900 text-white'
                        : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                    } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                  >
                    <FaUsers
                      className={`${
                        location.pathname === '/admin/users'
                          ? 'text-gray-300 dark:text-gray-400'
```

```
                        : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                      } mr-3 flex-shrink-0 h-6 w-6`}
                    />
                    Users
                  </Link>
                </li>
                <li>
                  <Link
                    to="/admin/leads"
                    className={`${
                      location.pathname === '/admin/leads'
                        ? 'bg-gray-900 text-white'
                        : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                    } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                  >
                    <FaClipboardList
                      className={`${
                        location.pathname === '/admin/leads'
                          ? 'text-gray-300 dark:text-gray-400'
                          : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                      } mr-3 flex-shrink-0 h-6 w-6`}
                    />
                    Leads
                  </Link>
                </li>
                <li>
                  <Link
                    to="/admin/activity-logs"
                    className={`${
                      location.pathname === '/admin/activity-logs'
                        ? 'bg-gray-900 text-white'
                        : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                    } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                  >
                    <FaHistory
                      className={`${
                        location.pathname === '/admin/activity-logs'
                          ? 'text-gray-300 dark:text-gray-400'
                          : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                      } mr-3 flex-shrink-0 h-6 w-6`}
                    />
                    Activity Logs
                  </Link>
                </li>
                <li>
                  <Link
                    to="/admin/import"
                    className={`${
                      location.pathname === '/admin/import'
                        ? 'bg-gray-900 text-white'
                        : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
```

```
                    } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                  >
                    <FaFileImport
                      className={`${
                        location.pathname === '/admin/import'
                          ? 'text-gray-300 dark:text-gray-400'
                          : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                      } mr-3 flex-shrink-0 h-6 w-6`}
                    />
                    Import Data
                  </Link>
                </li>
                <li>
                  <Link
                    to="/admin/activity"
                    className={`${
                      location.pathname === '/admin/activity'
                        ? 'bg-gray-900 text-white'
                        : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                    } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                  >
                    <FaClock
                      className={`${
                        location.pathname === '/admin/activity'
                          ? 'text-gray-300 dark:text-gray-400'
                          : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                      } mr-3 flex-shrink-0 h-6 w-6`}
                    />
                    Activity Dashboard
                  </Link>
                </li>
                <li>
                  <Link
                    to="/admin"
                    className={`${
                      location.pathname === '/admin'
                        ? 'bg-gray-900 text-white'
                        : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                    } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                  >
                    <FaCog
                      className={`${
                        location.pathname === '/admin'
                          ? 'text-gray-300 dark:text-gray-400'
                          : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                      } mr-3 flex-shrink-0 h-6 w-6`}
                    />
                    Settings
                  </Link>
                </li>
              </>
```

```
            )}

            {/* Activity Dashboard for Managers - Only visible to Manager (not
Admin) */}
            {user?.role === 'Manager' && (
              <>
                <li className="mt-8 mb-2">
                  <h3 className="px-3 text-xs font-semibold text-gray-400
dark:text-gray-500 uppercase tracking-wider">
                    Management
                  </h3>
                </li>
                <li>
                  <Link
                    to="/admin/activity"
                    className={`${
                      location.pathname === '/admin/activity'
                        ? 'bg-gray-900 text-white'
                        : 'text-gray-300 dark:text-gray-400 hover:bg-gray-700
hover:text-white'
                    } group flex items-center px-2 py-2 text-base font-medium
rounded-md`}
                  >
                    <FaClock
                      className={`${
                        location.pathname === '/admin/activity'
                          ? 'text-gray-300 dark:text-gray-400'
                          : 'text-gray-400 dark:text-gray-500 group-hover:text-
gray-300 dark:text-gray-400'
                      } mr-3 flex-shrink-0 h-6 w-6`}
                    />
                    Activity Dashboard
                  </Link>
                </li>
              </>
            )}
          </ul>
        </nav>

        <div className="p-4 border-t border-gray-700 space-y-3">
          {/* Theme Toggle */}
          <div className="flex justify-center">
            <ThemeToggle className="w-full justify-center" />
          </div>

          <button
            onClick={logout}
            className="w-full flex items-center px-4 py-2 text-sm text-gray-300
dark:text-gray-500 rounded-md hover:bg-gray-700"
          >
            <svg
              className="mr-3 h-4 w-4"
              fill="none"
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth="2"
              viewBox="0 0 24 24"
              stroke="currentColor"
            >
```

```
              <path d="M11 16l-4-4m0 0l4-4m-4 4h14m-5 4v1a3 3 0 01-3 3H6a3 3 0
01-3-3V7a3 3 0 013-3h7a3 3 0 013 3v1"></path>
            </svg>
            Logout
          </button>
        </div>
      </div>
    </div>
  );
};

export default Sidebar;
```

src/components/LeadAssignment/LeadPolling.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../../context/AuthContext';
import { FaVoteYea, FaClock, FaUsers, FaCheckCircle } from 'react-icons/fa';
import './LeadPolling.css';

const LeadPolling = () => {
  const { user, token } = useAuth();
  const [activePolls, setActivePolls] = useState([]);
  const [myVotes, setMyVotes] = useState({});
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    fetchActivePolls();
    // Poll for updates every 30 seconds
    const interval = setInterval(fetchActivePolls, 30000);
    return () => clearInterval(interval);
  }, []);

  const fetchActivePolls = async () => {
    try {
      const serverUrl = import.meta.env?.VITE_API_URL ||
process.env.REACT_APP_API_URL || 'http://localhost:8080';
      const response = await fetch(`${serverUrl}/api/lead-polling/active`, {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      });

      if (response.ok) {
        const data = await response.json();
        setActivePolls(data.data || []);

        // Get user's votes
        const votesResponse = await fetch(`${serverUrl}/api/lead-polling/my-
votes`, {
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json'
          }
        });

        if (votesResponse.ok) {
```

```
          const votesData = await votesResponse.json();
          setMyVotes(votesData.data || {});
        }
      }
    } catch (error) {
      console.error('Error fetching polls:', error);
    } finally {
      setIsLoading(false);
    }
  };

  const submitVote = async (pollId, interested) => {
    try {
      const serverUrl = import.meta.env?.VITE_API_URL ||
process.env.REACT_APP_API_URL || 'http://localhost:8080';
      const response = await fetch(`${serverUrl}/api/lead-polling/vote`, {
        method: 'POST',
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({
          pollId,
          interested
        })
      });

      if (response.ok) {
        // Update local state
        setMyVotes(prev => ({
          ...prev,
          [pollId]: interested
        }));

        // Refresh polls to get updated vote counts
        fetchActivePolls();
      }
    } catch (error) {
      console.error('Error submitting vote:', error);
    }
  };

  const getTimeRemaining = (expiresAt) => {
    const now = new Date();
    const expiry = new Date(expiresAt);
    const diff = expiry - now;

    if (diff <= 0) return 'Expired';

    const hours = Math.floor(diff / (1000 * 60 * 60));
    const minutes = Math.floor((diff % (1000 * 60 * 60)) / (1000 * 60));

    if (hours > 0) {
      return `${hours}h ${minutes}m remaining`;
    }
    return `${minutes}m remaining`;
  };

  if (isLoading) {
```

```
    return (
      <div className="lead-polling-container">
        <div className="loading-state">
          <div className="spinner"></div>
          <p>Loading active polls...</p>
        </div>
      </div>
    );
  }

  return (
    <div className="lead-polling-container">
      <div className="polling-header">
        <h2>
          <FaVoteYea className="header-icon" />
          Lead Assignment Polls
        </h2>
        <p>Vote on leads you're interested in taking</p>
      </div>

      {activePolls.length === 0 ? (
        <div className="no-polls">
          <FaVoteYea className="no-polls-icon" />
          <h3>No Active Polls</h3>
          <p>There are currently no leads available for polling.</p>
        </div>
      ) : (
        <div className="polls-grid">
          {activePolls.map(poll => (
            <div key={poll._id} className="poll-card">
              <div className="poll-header">
                <div className="lead-info">
                  <h3>{poll.leadData.name}</h3>
                  <p className="lead-details">
                    {poll.leadData.email} • {poll.leadData.phone}
                  </p>
                  <div className="lead-meta">
                    <span className="lead-source">{poll.leadData.source}</span>
                    <span className="lead-budget">Budget: {poll.leadData.budget}</
span>
                  </div>
                </div>
                <div className="poll-status">
                  <div className="time-remaining">
                    <FaClock className="clock-icon" />
                    {getTimeRemaining(poll.expiresAt)}
                  </div>
                </div>
              </div>

              <div className="poll-description">
                <h4>Lead Requirements:</h4>
                <p>{poll.leadData.requirements || 'No specific requirements
mentioned.'}</p>
              </div>

              <div className="poll-stats">
                <div className="vote-counts">
                  <div className="interested-count">
```

```
                    <FaCheckCircle className="vote-icon interested" />
                    <span>{poll.interestedCount} Interested</span>
                  </div>
                  <div className="not-interested-count">
                    <FaUsers className="vote-icon not-interested" />
                    <span>{poll.notInterestedCount} Not Interested</span>
                  </div>
                </div>
                <div className="total-votes">
                  Total Votes: {poll.totalVotes}
                </div>
              </div>

              <div className="voting-section">
                {myVotes[poll._id] !== undefined ? (
                  <div className="vote-submitted">
                    <FaCheckCircle className="submitted-icon" />
                    <span>
                      You voted: {myVotes[poll._id] ? 'Interested' : 'Not
Interested'}
                    </span>
                  </div>
                ) : (
                  <div className="vote-buttons">
                    <button
                      className="vote-btn interested"
                      onClick={() => submitVote(poll._id, true)}
                    >
                      <FaCheckCircle />
                      I'm Interested
                    </button>
                    <button
                      className="vote-btn not-interested"
                      onClick={() => submitVote(poll._id, false)}
                    >
                      <FaUsers />
                      Not for Me
                    </button>
                  </div>
                )}
              </div>

              {poll.assignedTo && (
                <div className="assignment-result">
                  <div className="assigned-banner">
                    <FaCheckCircle className="assigned-icon" />
                    <span>Assigned to: {poll.assignedTo.fullName}</span>
                  </div>
                </div>
              )}
            </div>
          ))}
        </div>
      )}
    </div>
  );
};

export default LeadPolling;
```

src/components/Leads/LeadForm.jsx

```
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import { leadsAPI } from '../../services/api';
import { useAuth } from '../../context/AuthContext';
import { authAPI } from '../../services/api';
import LoggingService from '../../services/loggingService'; // Add LoggingService
import

const LeadForm = ({ lead = null, onSuccess }) => {
  const navigate = useNavigate();
  const { user } = useAuth();
  const [salesPersons, setSalesPersons] = useState([]);
  const [leadPersons, setLeadPersons] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    course: '',
    countryCode: '',
    phone: '',
    country: '',
    pseudoId: '',
    company: '',
    client: '',
    status: 'Introduction',
    source: '',
    sourceLink: '',
    assignedTo: '',
    leadPerson: '',
    remarks: '',
    feedback: '',
    customCreatedAt: new Date().toISOString().split('T')[0] // Add default date
as today
  });

  // Fetch users for dropdown
  useEffect(() => {
    const fetchUsers = async () => {
      try {
        console.log('Fetching users for dropdowns');

        // Fetch sales persons
        const salesRes = await authAPI.getUsers('Sales Person');
        console.log('Sales persons fetched:', salesRes.data.data);
        setSalesPersons(salesRes.data.data || []);

        // Fetch lead persons
        const leadRes = await authAPI.getUsers('Lead Person');
        console.log('Lead persons fetched:', leadRes.data.data);
        setLeadPersons(leadRes.data.data || []);
      } catch (err) {
        console.error('Failed to fetch users', err);
      }
    };
```

```
    fetchUsers();
  }, []);

  // If editing, populate form with lead data
  useEffect(() => {
    console.log('========= FORM DATA INITIALIZATION =========');
    console.log('Current user:', user);
    console.log('User ID formats:', {
      _id: user?._id,
      id: user?.id
    });
    console.log('Lead data:', lead);

    if (lead) {
      // Editing an existing lead
      console.log('Editing existing lead, populating form with lead data');

      // Format the created date to YYYY-MM-DD format for date input
      const createdDate = lead.createdAt
        ? new Date(lead.createdAt).toISOString().split('T')[0]
        : new Date().toISOString().split('T')[0];

      // Ensure country code doesn't have a plus sign to match server expectations
      let countryCode = lead.countryCode || lead.CODE || '';
      countryCode = countryCode.replace(/^\+/, '');

      const formValues = {
        name: lead.name || lead.NAME || '',
        email: lead.email || lead['E-MAIL'] || '',
        course: lead.course || lead.COURSE || '',
        countryCode: countryCode,
        phone: lead.phone || lead.NUMBER || '',
        country: lead.country || lead.COUNTRY || '',
        pseudoId: lead.pseudoId || lead['PSUDO ID'] || '',
        company: lead.company || lead.COMPANY || '',
        client: lead.client || lead['CLIENT REMARK'] || '',
        status: lead.status || 'Introduction',
        source: lead.source || lead.SOURSE || '',
        sourceLink: lead.sourceLink || lead['SOURCE LINK'] || '',
        assignedTo: lead.assignedTo?._id || lead['SALE PERSON']?._id ||
lead['SALE PERSON'] || '',
        leadPerson: lead.leadPerson?._id || lead['LEAD PERSON']?._id ||
lead['LEAD PERSON'] || '',
        remarks: lead.remarks || '',
        feedback: lead.feedback || lead.FEEDBACK || '',
        customCreatedAt: createdDate
      };

      setFormData(formValues);
      console.log('Populated form data:', formValues);
    } else {
      // Creating a new lead
      console.log('Creating new lead, setting default values');

      // For consistent ID handling, use both _id and id fields from user object
      const userId = (user?._id || user?.id || '').toString();
      console.log('Using user ID for defaults:', userId);

      // Set defaults for new leads including the current date
```

```
      const today = new Date().toISOString().split('T')[0];

      const formValues = {
        name: '',
        email: '',
        course: '',
        countryCode: '1',  // Default country code without + sign to match server
expectations
        phone: '',
        country: '',
        pseudoId: '',
        company: '',
        client: '',
        status: 'Introduction',
        source: '',
        sourceLink: '',
        assignedTo: userId,
        leadPerson: user?.role === 'Lead Person' ? userId : '',
        remarks: '',
        feedback: '',
        customCreatedAt: today
      };

      setFormData(formValues);
      console.log('Default form data:', formValues);
    }
    console.log('========================================');
  }, [lead, user]);

  // Handle form input changes
  const handleChange = (e) => {
    const { name, value } = e.target;
    console.log(`Field changed: ${name}, value: ${value}`);
    setFormData(prev => ({
      ...prev,
      [name]: value
    }));
  };

  // Handle form submission
  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    setError(null);

    console.log('============= LEAD FORM SUBMIT =============');
    console.log('Form submission started');
    console.log('Is updating existing lead:', !!lead);
    console.log('Lead ID:', lead?._id);
    console.log('Form data:', formData);

    try {
      let response;

      // Prepare data for submission with proper field mapping
      const dataToSubmit = {
        NAME: formData.name,
        'E-MAIL': formData.email || '',
        COURSE: formData.course,
```

```
        CODE: formData.countryCode,
        NUMBER: formData.phone,
        COUNTRY: formData.country,
        'PSUDO ID': formData.pseudoId || '',
        'CLIENT REMARK': formData.client || '',
        status: formData.status || 'Introduction',
        SOURSE: formData.source || '',
        'SOURCE LINK': formData.sourceLink || '',
        'SALE PERSON': formData.assignedTo,
        leadPerson: formData.leadPerson || '',
        'LEAD PERSON': formData.leadPerson || '',
        feedback: formData.feedback || '',
        FEEDBACK: formData.feedback || '',
        ...(lead ? {
          // For existing leads, preserve the original createdAt date
          // Don't send DATE or createdAt fields to avoid changing the lead's
month/year
        } : {
          // For new leads, use the custom date or current date
          DATE: formData.customCreatedAt ? new
Date(formData.customCreatedAt).toISOString() : new Date().toISOString(),
          createdAt: formData.customCreatedAt ? new
Date(formData.customCreatedAt).toISOString() : new Date().toISOString()
        })
      };

      // Double-check all required fields are present and valid
      const requiredFields = ['name', 'course', 'phone', 'country', 'assignedTo'];
      const missingFields = requiredFields.filter(field =>
        !formData[field] || formData[field].trim() === ''
      );

      if (missingFields.length > 0) {
        console.error('Missing required fields:', missingFields);
        setError(`Missing required fields: ${missingFields.join(', ')}`);
        setLoading(false);
        return;
      }

      // Validate email only if provided and not empty (email is optional)
      if (formData.email && formData.email.trim() !== '') {
        // Simple check for @ symbol - don't be too strict on format to allow
international emails
        if (!formData.email.includes('@')) {
          console.error('Invalid email format:', formData.email);
          setError('Please enter a valid email address or leave it blank');
          setLoading(false);
          return;
        }
      } else {
        // Ensure empty email is properly handled by setting to empty string
        dataToSubmit.email = '';
        dataToSubmit['E-MAIL'] = '';
        console.log('Email field is blank, setting to empty string');
      }

      // Log the final data being sent for debugging
      console.log('Final data being sent to API:', dataToSubmit);
```

```
      if (lead) {
        // Update existing lead
        console.log('Calling API to update existing lead:', lead._id);
        response = await leadsAPI.update(lead._id, dataToSubmit);
        console.log('Update API response received:', response);

        // Log the lead update
        try {
          await LoggingService.logLeadUpdate(lead._id, dataToSubmit);
        } catch (logError) {
          console.error('Error logging lead update:', logError);
        }
      } else {
        // Create new lead
        console.log('Calling API to create new lead');
        response = await leadsAPI.create(dataToSubmit);
        console.log('Create API response received:', response);

        // Log the lead creation
        try {
          await LoggingService.logLeadCreate(response.data.data);
        } catch (logError) {
          console.error('Error logging lead creation:', logError);
        }
      }

      console.log('Full API response data:', response.data);
      console.log('API response success:', response.data?.success);
      console.log('API response lead data:', response.data?.data);

      if (response.data && response.data.success) {
        console.log('Lead saved successfully, calling onSuccess callback');
        console.log('Lead data being passed to onSuccess:', response.data.data);
        onSuccess(response.data.data);
        console.log('onSuccess callback completed');
      } else {
        console.error('API returned success: false', response);
        setError('Failed to save lead. Please check your input and try again.');
      }
    } catch (err) {
      console.error('Error saving lead:', err);
      console.error('Error response:', err.response?.data);
      setError(err.response?.data?.message || 'Failed to save lead. Please try
again later.');
    } finally {
      setLoading(false);
      console.log('============= LEAD FORM SUBMIT END =============');
    }
  };

  return (
    <form onSubmit={handleSubmit} className="space-y-8">
      {error && (
        <div className="bg-red-50 border border-red-200 text-red-700 px-4 py-3
rounded">
          {error}
        </div>
      )}
```

```
      {/* Contact Information */}
      <div>
        <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-4">Contact Information</h3>
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div>
            <label htmlFor="name" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Full Name <span className="text-red-500">*</span>
            </label>
            <input
              type="text"
              name="name"
              id="name"
              value={formData.name}
              onChange={handleChange}
              required
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            />
          </div>

          <div>
            <label htmlFor="email" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Email
            </label>
            <input
              type="email"
              name="email"
              id="email"
              value={formData.email}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            />
          </div>

          <div>
            <label htmlFor="phone" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Phone <span className="text-red-500">*</span>
            </label>
            <div className="mt-1 flex rounded-md shadow-sm dark:shadow-black/25">
              <div className="flex-shrink-0">
                <input
                  type="text"
                  name="countryCode"
                  id="countryCode"
                  value={formData.countryCode}
                  onChange={handleChange}
                  placeholder="Code"
                  className="block w-16 border border-gray-300 dark:border-
slate-600 rounded-md rounded-r-none shadow-sm dark:shadow-black/25 py-2 px-3
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                />
```

```
              </div>
              <input
                type="text"
                name="phone"
                id="phone"
                value={formData.phone}
                onChange={handleChange}
                required
                className="flex-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md rounded-l-none shadow-sm dark:shadow-black/25 py-2 px-3
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
              />
            </div>
          </div>

          <div>
            <label htmlFor="country" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Country <span className="text-red-500">*</span>
            </label>
            <input
              type="text"
              name="country"
              id="country"
              value={formData.country}
              onChange={handleChange}
              required
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            />
          </div>
        </div>
      </div>

      {/* Course Information */}
      <div>
        <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-4">Course Information</h3>
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div>
            <label htmlFor="course" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Course <span className="text-red-500">*</span>
            </label>
            <input
              type="text"
              name="course"
              id="course"
              value={formData.course}
              onChange={handleChange}
              required
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            />
          </div>
```

```
          <div>
            <label htmlFor="source" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Source
            </label>
            <input
              type="text"
              name="source"
              id="source"
              value={formData.source}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            />
          </div>

          <div>
            <label htmlFor="sourceLink" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Source Link
            </label>
            <input
              type="text"
              name="sourceLink"
              id="sourceLink"
              value={formData.sourceLink}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            />
          </div>

          <div>
            <label htmlFor="pseudoId" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Pseudo ID
            </label>
            <input
              type="text"
              name="pseudoId"
              id="pseudoId"
              value={formData.pseudoId}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            />
          </div>
        </div>
      </div>

      {/* Assignment & Status */}
      <div>
        <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-4">Assignment & Status</h3>
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div>
```

```
            <label htmlFor="assignedTo" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Assigned Sales Person <span className="text-red-500">*</span>
            </label>
            <select
              id="assignedTo"
              name="assignedTo"
              value={formData.assignedTo}
              onChange={handleChange}
              required
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            >
              <option value="">Select Sales Person</option>
              {salesPersons.map(person => (
                <option key={person._id} value={person._id}>
                  {person.fullName || person.email}
                </option>
              ))}
            </select>
          </div>

          <div>
            <label htmlFor="leadPerson" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Lead Person
            </label>
            <select
              id="leadPerson"
              name="leadPerson"
              value={formData.leadPerson}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            >
              <option value="">Select Lead Person</option>
              {leadPersons.map(person => (
                <option key={person._id} value={person._id}>
                  {person.fullName || person.email}
                </option>
              ))}
            </select>
          </div>

          <div>
            <label htmlFor="status" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Status
            </label>
            <select
              id="status"
              name="status"
              value={formData.status}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
```

```
            >
              <option value="Introduction">Introduction</option>
              <option value="Acknowledgement">Acknowledgement</option>
              <option value="Question">Question</option>
              <option value="Future Promise">Future Promise</option>
              <option value="Payment">Payment</option>
              <option value="Analysis">Analysis</option>
            </select>
          </div>

          <div>
            <label htmlFor="feedback" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Feedback
            </label>
            <select
              id="feedback"
              name="feedback"
              value={formData.feedback}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            >
              <option value="">Select Feedback</option>
              <option value="Pending">Pending</option>
              <option value="Converted">Converted</option>
              <option value="Not Interested">Not Interested</option>
              <option value="Follow Up">Follow Up</option>
            </select>
          </div>
        </div>
      </div>

      {/* Additional Information */}
      <div>
        <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-4">Additional Information</h3>
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div className="md:col-span-2">
            <label htmlFor="client" className="block text-sm font-medium text-
gray-700 dark:text-gray-400">
              Client Remarks
            </label>
            <textarea
              id="client"
              name="client"
              rows="3"
              value={formData.client}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            ></textarea>
          </div>

          <div>
            <label htmlFor="customCreatedAt" className="block text-sm font-medium
text-gray-700 dark:text-gray-400">
```

```
              Date Added
            </label>
            <input
              type="date"
              name="customCreatedAt"
              id="customCreatedAt"
              value={formData.customCreatedAt}
              onChange={handleChange}
              className="mt-1 block w-full border border-gray-300 dark:border-
slate-600 rounded-md shadow-sm dark:shadow-black/25 py-2 px-3 focus:outline-none
focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
            />
          </div>
        </div>
      </div>

      <div className="pt-5 flex justify-end">
        <button
          type="submit"
          disabled={loading}
          className="ml-3 inline-flex justify-center py-2 px-4 border border-
transparent shadow-sm dark:shadow-black/25 text-sm font-medium rounded-md text-
white bg-blue-600 hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-
offset-2 focus:ring-blue-500 dark:focus:ring-blue-400"
        >
          {loading ? 'Saving...' : lead ? 'Update Lead' : 'Add Lead'}
        </button>
      </div>
    </form>
  );
};

export default LeadForm;
```

src/components/Navbar.jsx

```
// src/components/Navbar.jsxÐ
import React, { useState, useRef, useEffect } from "react";Ð
import { Link, useNavigate } from "react-router-dom";Ð
import { useAuth } from "../context/AuthContext";Ð
import CurrencySelector from "./CurrencySelector";Ð
import ThemeToggle from "./ThemeToggle";Ð
import ActivityTimer from "./ActivityTimer/ActivityTimer";Ð
import logo from '../assets/traincape-logo.jpg';Ð
Ð
const Navbar = () => {Ð
  const { user, logout } = useAuth();Ð
  const navigate = useNavigate();Ð
  const [isMenuOpen, setIsMenuOpen] = useState(false);Ð
  const [isAdminDropdownOpen, setIsAdminDropdownOpen] = useState(false);Ð
  const [isSalesDropdownOpen, setIsSalesDropdownOpen] = useState(false);Ð
  const [isLeadsDropdownOpen, setIsLeadsDropdownOpen] = useState(false);Ð
  const [isProfileDropdownOpen, setIsProfileDropdownOpen] = useState(false);Ð
  Ð
  const adminDropdownRef = useRef(null);Ð
  const salesDropdownRef = useRef(null);Ð
  const leadsDropdownRef = useRef(null);Ð
  const profileDropdownRef = useRef(null);Ð
```

```
Ð
  // Close dropdowns when clicking outsideÐ
  useEffect(() => {Ð
    const handleClickOutside = (event) => {Ð
      if (adminDropdownRef.current && !
adminDropdownRef.current.contains(event.target)) {Ð
        setIsAdminDropdownOpen(false);Ð
      }Ð
      if (salesDropdownRef.current && !
salesDropdownRef.current.contains(event.target)) {Ð
        setIsSalesDropdownOpen(false);Ð
      }Ð
      if (leadsDropdownRef.current && !
leadsDropdownRef.current.contains(event.target)) {Ð
        setIsLeadsDropdownOpen(false);Ð
      }Ð
      if (profileDropdownRef.current && !
profileDropdownRef.current.contains(event.target)) {Ð
        setIsProfileDropdownOpen(false);Ð
      }Ð
    };Ð
Ð
    document.addEventListener('mousedown', handleClickOutside);Ð
    return () => document.removeEventListener('mousedown', handleClickOutside);Ð
  }, []);Ð
Ð
  const handleLogout = () => {Ð
    logout();Ð
    navigate("/login");Ð
  };Ð
Ð
  const toggleMenu = () => {Ð
    setIsMenuOpen(!isMenuOpen);Ð
  };Ð
Ð
  // Dropdown menu componentÐ
  const DropdownMenu = ({ isOpen, items, title, reference }) => (Ð
    <div ref={reference} className="relative">Ð
      <buttonÐ
        onClick={() => reference === adminDropdownRef ? setIsAdminDropdownOpen(!
isOpen) :Ð
                     reference === salesDropdownRef ? setIsSalesDropdownOpen(!
isOpen) :Ð
                     reference === leadsDropdownRef ? setIsLeadsDropdownOpen(!
isOpen) :Ð
                     setIsProfileDropdownOpen(!isOpen)}Ð
        className="flex items-center hover:text-green-300 focus:outline-none"Ð
      >Ð
        {title}Ð
        <svg className={`ml-1 h-4 w-4 transition-transform ${isOpen ?
'rotate-180' : ''}`} fill="none" stroke="currentColor" viewBox="0 0 24 24">Ð
          <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2"
d="M19 9l-7 7-7-7" />Ð
        </svg>Ð
      </button>Ð
      {isOpen && (Ð
        <div className="absolute right-0 mt-2 py-2 w-48 bg-white dark:bg-
slate-800 rounded-md shadow-lg z-50 border dark:border-slate-700">Ð
          {items.map((item, index) => (Ð
```

```
            item.onClick ? (Đ
              <buttonĐ
                key={index}Đ
                onClick={() => {Đ
                  item.onClick();Đ
                  reference === adminDropdownRef ? setIsAdminDropdownOpen(false) :Đ
                  reference === salesDropdownRef ? setIsSalesDropdownOpen(false) :Đ
                  reference === leadsDropdownRef ? setIsLeadsDropdownOpen(false) :Đ
                  setIsProfileDropdownOpen(false);Đ
                }}Đ
                className="w-full text-left px-4 py-2 text-sm text-gray-700
dark:text-gray-200 hover:bg-gray-100 dark:hover:bg-slate-700"Đ
              >Đ
                {item.label}Đ
              </button>Đ
            ) : (Đ
              <LinkĐ
                key={index}Đ
                to={item.path}Đ
                className="block px-4 py-2 text-sm text-gray-700 dark:text-
gray-200 hover:bg-gray-100 dark:hover:bg-slate-700"Đ
                onClick={() => reference === adminDropdownRef ?
setIsAdminDropdownOpen(false) :Đ
                             reference === salesDropdownRef ?
setIsSalesDropdownOpen(false) :Đ
                             reference === leadsDropdownRef ?
setIsLeadsDropdownOpen(false) :Đ
                             setIsProfileDropdownOpen(false)}Đ
              >Đ
                {item.label}Đ
              </Link>Đ
            )Đ
          ))}Đ
        </div>Đ
      )}Đ
    </div>Đ
  );Đ
Đ
  // Define dropdown itemsĐ
  const adminItems = [Đ
    { label: 'Dashboard', path: '/admin' },Đ
    { label: 'Manage Users', path: '/admin/users' },Đ
    { label: 'Manage Leads', path: '/admin/leads' },Đ
    { label: 'Import Data', path: '/admin/import' },Đ
    { label: 'Activity Logs', path: '/admin/activity-logs' }Đ
  ];Đ
Đ
  const salesItems = [Đ
    { label: 'Sales Tracking', path: '/sales-tracking' },Đ
    { label: 'Prospects', path: '/prospects' }Đ
  ];Đ
Đ
  const leadsItems = [Đ
    { label: 'Leads', path: '/leads' },Đ
    { label: 'Lead Sales Sheet', path: '/lead-sales-sheet' }Đ
  ];Đ
Đ
  const profileItems = [Đ
    { label: 'Profile', path: '/profile' },Đ
```

```
    { label: 'Settings', path: '/settings' }Đ
  ];Đ
Đ
  return (Đ
    <header className="bg-blue-600 dark:bg-slate-900 text-white py-4 shadow-md
dark:shadow-black/25 transition-all duration-200 ease-out border-b border-
transparent dark:border-slate-700">Đ
      <nav className="container mx-auto flex justify-between items-center px-4">Đ
        <a href="/" className="flex items-center space-x-3">Đ
          <img src={logo} alt="Traincape Technology Logo" className="h-12 w-12
rounded" />Đ
          <span className="text-xl font-bold tracking-wide hidden
md:block">Traincape Technology</span>Đ
        </a>Đ
        Đ
        <button Đ
          className="text-white md:hidden" Đ
          onClick={toggleMenu}Đ
        >Đ
          <svg className="w-6 h-6" fill="none" stroke="currentColor" viewBox="0 0
24 24">Đ
            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2"
d="M4 6h16M4 12h16m-7 6h7"></path>Đ
          </svg>Đ
        </button>Đ
Đ
        <div className={`${isMenuOpen ? 'flex' : 'hidden'} md:flex flex-col
md:flex-row items-center space-y-4 md:space-y-0 md:space-x-6 absolute md:relative
top-16 md:top-0 left-0 md:left-auto w-full md:w-auto bg-blue-600 dark:bg-
slate-900 md:bg-transparent p-4 md:p-0 z-50`}>Đ
          {/* Home link accessible to all */}Đ
          <Link to="/" className="hover:text-green-300">Home</Link>Đ
Đ
          {/* Customer Dashboard */}Đ
          {user && user.role === "Customer" && (Đ
            <Link to="/customer" className="hover:text-green-300">Dashboard</Link>Đ
          )}Đ
Đ
          {/* Sales Person Links */}Đ
          {user && (user.role === "Sales Person" || user.role === "Manager" ||
user.role === "Admin") && (Đ
            <DropdownMenuĐ
              isOpen={isSalesDropdownOpen}Đ
              items={salesItems}Đ
              title="Sales"Đ
              reference={salesDropdownRef}Đ
            />Đ
          )}Đ
Đ
          {/* Lead Person Links */}Đ
          {user && (user.role === "Lead Person" || user.role === "Manager" ||
user.role === "Admin") && (Đ
            <DropdownMenuĐ
              isOpen={isLeadsDropdownOpen}Đ
              items={leadsItems}Đ
              title="Leads"Đ
              reference={leadsDropdownRef}Đ
            />Đ
          )}Đ
```

```
Đ
          {/* Admin Links */}Đ
          {user && user.role === "Admin" && (Đ
            <DropdownMenuĐ
              isOpen={isAdminDropdownOpen}Đ
              items={adminItems}Đ
              title="Admin"Đ
              reference={adminDropdownRef}Đ
            />Đ
          )}Đ
Đ
          {/* Right side items */}Đ
          <div className="flex items-center space-x-4">Đ
            <CurrencySelector />Đ
            <ThemeToggle />Đ
            <ActivityTimer />Đ
            Đ
            {user ? (Đ
              <DropdownMenuĐ
                isOpen={isProfileDropdownOpen}Đ
                items={[Đ
                  ...profileItems,Đ
                  { Đ
                    label: 'Logout',Đ
                    path: '#',Đ
                    onClick: handleLogoutĐ
                  }Đ
                ]}Đ
                title={user.fullName || 'Profile'}Đ
                reference={profileDropdownRef}Đ
              />Đ
            ) : (Đ
              <Link to="/login" className="hover:text-green-300">Login</Link>Đ
            )}Đ
          </div>Đ
        </div>Đ
      </nav>Đ
    </header>Đ
  );Đ
};Đ
Đ
export default Navbar;Đ
```

src/components/SalarySlipWidget.jsx

```
import React, { useState, useEffect } from 'react';
import api, { payrollAPI } from '../services/api';

const SalarySlipWidget = () => {
  const [payrollData, setPayrollData] = useState([]);
  const [loading, setLoading] = useState(false);
  const [downloadingSlip, setDownloadingSlip] = useState(null);
  const [selectedMonth, setSelectedMonth] = useState(new Date().getMonth() + 1);
  const [selectedYear, setSelectedYear] = useState(new Date().getFullYear());

  // Remove automatic data fetching on component mount
  // useEffect(() => {
  //   fetchPayrollData();
```

```
  // }, []);

  const fetchPayrollData = async () => {
    try {
      setLoading(true);
      const response = await payrollAPI.getAll({ month: selectedMonth, year:
selectedYear });
      setPayrollData(response.data.data);
    } catch (error) {
      console.error('Error fetching payroll data:', error);
      alert('Error fetching payroll data: ' + (error.response?.data?.message ||
'Unknown error'));
    } finally {
      setLoading(false);
    }
  };

  const handleManualRefresh = () => {
    fetchPayrollData();
  };

  const handleDownloadSlip = async (payrollId) => {
    setDownloadingSlip(payrollId);
    try {
      const response = await payrollAPI.generateSalarySlip(payrollId);

      // Create blob link to download
      const url = window.URL.createObjectURL(new Blob([response.data]));
      const link = document.createElement('a');
      link.href = url;

      // Get filename from response headers or create default
      const contentDisposition = response.headers['content-disposition'];
      let filename = 'salary-slip.pdf';
      if (contentDisposition) {
        const filenameMatch = contentDisposition.match(/filename="(.+)"/);
        if (filenameMatch) {
          filename = filenameMatch[1];
        }
      }

      link.setAttribute('download', filename);
      document.body.appendChild(link);
      link.click();
      link.remove();
      window.URL.revokeObjectURL(url);

    } catch (error) {
      console.error('Error downloading salary slip:', error);
      alert('Error downloading salary slip: ' + (error.response?.data?.message ||
'Unknown error'));
    } finally {
      setDownloadingSlip(null);
    }
  };

  const formatCurrency = (amount) => {
    return new Intl.NumberFormat('en-IN', {
      style: 'currency',
```

```
      currency: 'INR'
    }).format(amount);
  };

  const getStatusBadge = (status) => {
    const statusClasses = {
      'DRAFT': 'bg-gray-100 text-gray-800 dark:bg-gray-700 dark:text-gray-300',
      'APPROVED': 'bg-green-100 text-green-800 dark:bg-green-700 dark:text-
green-300',
      'PAID': 'bg-blue-100 text-blue-800 dark:bg-blue-700 dark:text-blue-300',
      'CANCELLED': 'bg-red-100 text-red-800 dark:bg-red-700 dark:text-red-300'
    };

    return (
      <span className={`inline-flex items-center px-2.5 py-0.5 rounded-full text-
xs font-medium ${statusClasses[status] || statusClasses['DRAFT']}`}>
        {status}
      </span>
    );
  };

  const currentYear = new Date().getFullYear();
  const years = Array.from({length: 5}, (_, i) => currentYear - i);
  const months = [
    { value: 1, label: 'January' },
    { value: 2, label: 'February' },
    { value: 3, label: 'March' },
    { value: 4, label: 'April' },
    { value: 5, label: 'May' },
    { value: 6, label: 'June' },
    { value: 7, label: 'July' },
    { value: 8, label: 'August' },
    { value: 9, label: 'September' },
    { value: 10, label: 'October' },
    { value: 11, label: 'November' },
    { value: 12, label: 'December' }
  ];

  return (
    <div className="bg-white dark:bg-slate-800 rounded-lg shadow-md p-6">
      <div className="flex items-center justify-between mb-6">
        <h3 className="text-lg font-semibold text-gray-900 dark:text-white">
          Salary Slips
        </h3>
        <div className="text-sm text-gray-500 dark:text-gray-400">
          Manual Update Required
        </div>
      </div>

      {/* Manual Controls */}
      <div className="bg-gray-50 dark:bg-gray-700 p-4 rounded-lg mb-6">
        <div className="flex flex-wrap items-center gap-4">
          <div className="flex items-center space-x-2">
            <label className="text-sm font-medium text-gray-700 dark:text-
gray-300">
              Month:
            </label>
            <select
              value={selectedMonth}
```

```
              onChange={(e) => setSelectedMonth(parseInt(e.target.value))}
              className="px-3 py-2 border border-gray-300 dark:border-gray-600
rounded-md bg-white dark:bg-gray-800 text-gray-900 dark:text-white text-sm"
            >
              {months.map(month => (
                <option key={month.value} value={month.value}>
                  {month.label}
                </option>
              ))}
            </select>
          </div>

          <div className="flex items-center space-x-2">
            <label className="text-sm font-medium text-gray-700 dark:text-
gray-300">
              Year:
            </label>
            <select
              value={selectedYear}
              onChange={(e) => setSelectedYear(parseInt(e.target.value))}
              className="px-3 py-2 border border-gray-300 dark:border-gray-600
rounded-md bg-white dark:bg-gray-800 text-gray-900 dark:text-white text-sm"
            >
              {years.map(year => (
                <option key={year} value={year}>
                  {year}
                </option>
              ))}
            </select>
          </div>

          <button
            onClick={handleManualRefresh}
            disabled={loading}
            className="bg-blue-600 hover:bg-blue-700 disabled:bg-blue-400 text-
white px-4 py-2 rounded-md text-sm font-medium transition-colors flex items-
center space-x-2"
          >
            {loading ? (
              <>
                <svg className="animate-spin h-4 w-4" fill="none" viewBox="0 0 24
24">
                  <circle className="opacity-25" cx="12" cy="12" r="10"
stroke="currentColor" strokeWidth="4"></circle>
                  <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0
018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135
5.824 3 7.938l3-2.647z"></path>
                </svg>
                <span>Loading...</span>
              </>
            ) : (
              <>
                <svg className="h-4 w-4" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth="2" d="M4 4v5h.582m15.356 2A8.001 8.001 0 004.582 9m0 0H9m11
11v-5h-.581m0 0a8.003 8.003 0 01-15.357-2m15.357 2H15"></path>
                </svg>
                <span>Load Salary Slips</span>
```

```
              </>
            )}
          </button>
        </div>
      </div>

      {/* Results */}
      {loading ? (
        <div className="animate-pulse space-y-4">
          <div className="h-4 bg-gray-200 dark:bg-gray-700 rounded w-3/4"></div>
          <div className="h-8 bg-gray-200 dark:bg-gray-700 rounded w-1/2"></div>
          <div className="h-4 bg-gray-200 dark:bg-gray-700 rounded w-full"></div>
        </div>
      ) : payrollData.length === 0 ? (
        <div className="text-center py-8">
          <div className="text-gray-500 dark:text-gray-400 mb-4">
            No salary slip found for the selected month and year
          </div>
          <p className="text-sm text-gray-400 dark:text-gray-500">
            Click "Load Salary Slips" to fetch data or select a different month/
year
          </p>
        </div>
      ) : (
        <div className="space-y-4">
          {payrollData.map((payroll) => (
            <div key={payroll._id} className="border border-gray-200 dark:border-
gray-700 rounded-lg p-4">
              <div className="flex items-center justify-between mb-4">
                <div>
                  <h4 className="font-medium text-gray-900 dark:text-white">
                    {payroll.monthName} {payroll.year}
                  </h4>
                  <p className="text-sm text-gray-500 dark:text-gray-400">
                    Generated on {new
Date(payroll.createdAt).toLocaleDateString()}
                  </p>
                </div>
                <div>
                  {getStatusBadge(payroll.status)}
                </div>
              </div>

              <div className="grid grid-cols-2 md:grid-cols-4 gap-4 mb-4">
                <div className="bg-gray-50 dark:bg-gray-700 p-3 rounded-lg">
                  <div className="text-xs text-gray-500 dark:text-
gray-400">Working Days</div>
                  <div className="text-lg font-semibold text-gray-900 dark:text-
white">
                    {payroll.workingDays}
                  </div>
                </div>
                <div className="bg-green-50 dark:bg-green-900/20 p-3 rounded-lg">
                  <div className="text-xs text-green-700 dark:text-
green-400">Present Days</div>
                  <div className="text-lg font-semibold text-green-900 dark:text-
green-100">
                    {payroll.presentDays}
                  </div>
```

```
                </div>
                <div className="bg-blue-50 dark:bg-blue-900/20 p-3 rounded-lg">
                  <div className="text-xs text-blue-700 dark:text-blue-400">Gross
Salary</div>
                  <div className="text-lg font-semibold text-blue-900 dark:text-
blue-100">
                    {formatCurrency(payroll.grossSalary)}
                  </div>
                </div>
                <div className="bg-purple-50 dark:bg-purple-900/20 p-3 rounded-
lg">
                  <div className="text-xs text-purple-700 dark:text-
purple-400">Net Salary</div>
                  <div className="text-lg font-semibold text-purple-900 dark:text-
purple-100">
                    {formatCurrency(payroll.netSalary)}
                  </div>
                </div>
              </div>

              {payroll.status === 'APPROVED' || payroll.status === 'PAID' ? (
                <button
                  onClick={() => handleDownloadSlip(payroll._id)}
                  disabled={downloadingSlip === payroll._id}
                  className="w-full bg-blue-600 hover:bg-blue-700 disabled:bg-
blue-400 text-white py-2 px-4 rounded-md transition-colors flex items-center
justify-center space-x-2"
                >
                  {downloadingSlip === payroll._id ? (
                    <>
                      <svg className="animate-spin h-4 w-4" fill="none"
viewBox="0 0 24 24">
                        <circle className="opacity-25" cx="12" cy="12" r="10"
stroke="currentColor" strokeWidth="4"></circle>
                        <path className="opacity-75" fill="currentColor" d="M4
12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042
1.135 5.824 3 7.938l3-2.647z"></path>
                      </svg>
                      <span>Downloading...</span>
                    </>
                  ) : (
                    <>
                      <svg className="h-4 w-4" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                        <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth="2" d="M12 10v6m0 0l-3-3m3 3l3-3m2 8H7a2 2 0 01-2-2V5a2 2 0
012-2h5.586a1 1 0 01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z"></
path>
                      </svg>
                      <span>Download Salary Slip</span>
                    </>
                  )}
                </button>
              ) : (
                <div className="w-full bg-gray-100 dark:bg-gray-700 text-gray-500
dark:text-gray-400 py-2 px-4 rounded-md text-center">
                  Salary slip not available (Status: {payroll.status})
                </div>
              )}
```

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

export default SalarySlipWidget;
```

src/components/Sidebar.jsx

```
import { FaBook } from 'react-icons/fa';
```

src/components/ThemeToggle.jsx

```
import { FaSun, FaMoon } from 'react-icons/fa';
import { useTheme } from '../hooks/useTheme';

const ThemeToggle = ({ className = "" }) => {
  const { theme, toggleTheme } = useTheme();

  return (
    <button
      onClick={toggleTheme}
      className={`relative inline-flex items-center justify-center p-2 rounded-lg
bg-slate-200 dark:bg-slate-700 hover:bg-slate-300 dark:hover:bg-slate-600 border
border-slate-300 dark:border-slate-600 shadow-sm dark:shadow-lg dark:shadow-2xl
dark:shadow-black/25 hover:shadow-md dark:hover:shadow-xl transition-all
duration-200 ease-out ${className}`}
      title={`Switch to ${theme === 'dark' ? 'light' : 'dark'} mode`}
    >
      <div className="relative w-5 h-5">
        {theme === 'dark' ? (
          <FaSun className="w-5 h-5 text-yellow-400 animate-pulse drop-shadow-sm
dark:shadow-black/25" />
        ) : (
          <FaMoon className="w-5 h-5 text-slate-600 dark:text-slate-300 drop-
shadow-sm dark:shadow-black/25" />
        )}
      </div>
    </button>
  );
};

export default ThemeToggle;
```

src/components/ui/card.jsx

```
import React from "react";

export function Card({ children, className = "" }) {
  return (
    <div className={`bg-white dark:bg-slate-900 border border-gray-200
dark:border-slate-700 rounded-xl shadow-sm dark:shadow-lg dark:shadow-black/25
```

```
${className}`}>
      {children}
    </div>
  );
}

export function CardContent({ children, className = "" }) {
  return <div className={`p-4 ${className}`}>{children}</div>;
}
```

src/context/AuthContext.jsx

```
import React, { createContext, useState, useEffect, useContext } from 'react';
import { authAPI } from '../services/api';
import LoggingService from '../services/loggingService';

// Create auth context
const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [token, setToken] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  // Initialize auth state
  useEffect(() => {
    // Check if user is already logged in (token exists)
    const storedToken = localStorage.getItem('token');
    if (storedToken) {
      setToken(storedToken);
      loadUser();
    } else {
      setLoading(false);
    }
  }, []);

  // Load user data
  const loadUser = async () => {
    try {
      setLoading(true);
      const storedToken = localStorage.getItem('token');

      if (!storedToken) {
        setToken(null);
        setUser(null);
        setLoading(false);
        return;
      }

      setToken(storedToken);
      const res = await authAPI.getProfile();
      setUser(res.data.data);
      setError(null);
    } catch (err) {
      localStorage.removeItem('token');
      setToken(null);
      setUser(null);
```

```
      setError('Authentication failed. Please login again.');
    } finally {
      setLoading(false);
    }
  };

  // Register user
  const register = async (userData) => {
    try {
      setLoading(true);
      const res = await authAPI.register(userData);
      setError(null);
      return res.data;
    } catch (err) {
      const errorMessage = err.response?.data?.message || 'Registration failed.
Please try again.';
      setError(errorMessage);
      throw new Error(errorMessage);
    } finally {
      setLoading(false);
    }
  };

  const login = async (formData) => {
    try {
      const response = await authAPI.login(formData);

      if (response.data.success && response.data.token) {
        // Save token first
        localStorage.setItem('token', response.data.token);

        // Set user data
        const userData = response.data.data;
        setUser(userData);

        // Log the login action after user is set
        try {
          await LoggingService.logLogin(userData._id, true);
        } catch (logError) {
          console.error('Error logging login:', logError);
          // Don't fail login if logging fails
        }

        return {
          success: true,
          data: userData
        };
      }

      return {
        success: false,
        message: response.data.message || 'Invalid credentials'
      };
    } catch (error) {
      // Log failed login attempt
      try {
        await LoggingService.logLogin(null, false);
      } catch (logError) {
        console.error('Error logging failed login:', logError);
```

```
      }

      return {
        success: false,
        message: error.response?.data?.message || 'Login failed'
      };
    }
  };

  const logout = async () => {
    try {
      const userId = user?._id;
      if (userId) {
        try {
          await LoggingService.logLogout(userId);
        } catch (logError) {
          console.error('Error logging logout:', logError);
        }
      }

      // Clear user data and token
      setUser(null);
      localStorage.removeItem('token');

      // Clear any other session data
      localStorage.removeItem('activitySessionStart');
      localStorage.removeItem('activitySessionActive');
      localStorage.removeItem('activityManuallyPaused');
      localStorage.removeItem('activityPageHiddenTime');
    } catch (error) {
      console.error('Error during logout:', error);
    }
  };

  return (
    <AuthContext.Provider
      value={{
        user,
        token,
        loading,
        error,
        register,
        login,
        logout,
        setUser,
        loadUser
      }}
    >
      {children}
    </AuthContext.Provider>
  );
};

// Custom hook to use auth context
export const useAuth = () => useContext(AuthContext);

export default AuthContext;
```

src/context/ChatContext.jsx

```
import React, { createContext, useContext, useEffect, useState, useRef } from
'react';
import { io } from 'socket.io-client';
import { useAuth } from './AuthContext';
import toast from 'react-hot-toast';
import notificationService from '../services/notificationService';

const ChatContext = createContext();

export const useChat = () => {
  const context = useContext(ChatContext);
  if (!context) {
    throw new Error('useChat must be used within a ChatProvider');
  }
  return context;
};

export const ChatProvider = ({ children }) => {
  const { user, token } = useAuth();
  const [socket, setSocket] = useState(null);
  const [isConnected, setIsConnected] = useState(false);
  const [onlineUsers, setOnlineUsers] = useState([]);
  const [chatRooms, setChatRooms] = useState([]);
  const [activeChat, setActiveChat] = useState(null);
  const [messages, setMessages] = useState({});
  const [unreadCounts, setUnreadCounts] = useState({});
  const [typingUsers, setTypingUsers] = useState({});
  const [isChatOpen, setIsChatOpen] = useState(false);
  const [notificationPreferences, setNotificationPreferences] = useState({
    enableSounds: true,
    messageSound: 'message',
    volume: 0.3,
    enableBrowserNotifications: true,
    enableToastNotifications: true
  });

  const socketRef = useRef(null);
  const typingTimeoutRef = useRef({});

  // Initialize Socket.IO connection
  useEffect(() => {
    if (user && token) {
      // Support both Vite and Create React App environment variables
      // Remove '/api' from the URL for socket connection
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const apiUrl = isDevelopment ? 'http://localhost:8080/api' : 'https://crm-
backend-o36v.onrender.com/api';
      const serverUrl = apiUrl.replace('/api', ''); // Remove /api for socket
connection
      console.log('Ø=Ý  Connecting to chat server:', serverUrl);
      console.log('Ø=Üd User:', user.fullName, 'Role:', user.role);
      console.log('Ø=Ý  Token present:', !!token);

      const newSocket = io(serverUrl, {
        transports: ['websocket', 'polling'],
```

```
  timeout: 20000,
  forceNew: true,
  reconnection: true,
  reconnectionDelay: 1000,
  reconnectionAttempts: 5,
  maxReconnectionAttempts: 5,
  auth: {
    token: token
  },
  query: {
    userId: user._id,
    userRole: user.role
  }
});

newSocket.on('connect', () => {
  console.log('  Connected to chat server');
  setIsConnected(true);

  // Join user room for targeted messages
  newSocket.emit('join-user-room', user._id);
  console.log('Ø=Üá Joined user room:', user._id);

  // Update user status to online
  updateUserStatus('ONLINE');

  // Fetch initial data
  fetchAllUsers();
  fetchChatRooms();
});

newSocket.on('connect_error', (error) => {
  console.error(''L Socket connection error:', error);
  setIsConnected(false);

  // Show user-friendly error message
  toast.error('Failed to connect to chat server. Please refresh the page.');
});

newSocket.on('disconnect', (reason) => {
  console.log('Ø=Ý  Disconnected from chat server:', reason);
  setIsConnected(false);

  if (reason === 'io server disconnect') {
    // Server disconnected, try to reconnect
    newSocket.connect();
  }
});

newSocket.on('reconnect', (attemptNumber) => {
  console.log('Ø=Ý  Reconnected to chat server, attempt:', attemptNumber);
  setIsConnected(true);
  // Rejoin user room after reconnection
  newSocket.emit('join-user-room', user._id);
  updateUserStatus('ONLINE');

  toast.success('Reconnected to chat server');
});
```

```
      // Handle new messages
      newSocket.on('newMessage', (message) => {
        console.log('Ø=Üè New message received:', message);

        setMessages(prev => ({
          ...prev,
          [message.chatId]: [...(prev[message.chatId] || []), message]
        }));

        // Update unread count if not in active chat
        if (!activeChat || activeChat.chatId !== message.chatId) {
          setUnreadCounts(prev => ({
            ...prev,
            [message.senderId._id]: (prev[message.senderId._id] || 0) + 1
          }));
        }

        // Update chat rooms with latest message
        setChatRooms(prev => prev.map(room =>
          room.chatId === message.chatId
            ? { ...room, lastMessage: message.content, lastMessageTime:
message.timestamp }
            : room
        ));
      });

      // Handle message notifications - integrated with notification service
      newSocket.on('messageNotification', (notification) => {
        console.log('Ø=Ý  Message notification received:', notification);

        // The notification service will handle this automatically
        // But we can also add local logic here if needed

        // Only show local toast if chat window is not open or not focused on
sender
        if (!isChatOpen || (activeChat && activeChat.otherUser._id !==
notification.senderId)) {

          // Check user preferences before showing notifications
          if (notificationPreferences.enableToastNotifications) {
            const message = notification.isGuest
              ? `Ø=Üd ${notification.senderName}: ${notification.content.substring(0,
50)}${notification.content.length > 50 ? '...' : ''}`
              : `Ø=Ü¬ ${notification.senderName}: ${notification.content.substring(0,
50)}${notification.content.length > 50 ? '...' : ''}`;

            toast(message, {
              icon: notification.isGuest ? 'Ø=Üd' : 'Ø=Ü¬',
              duration: 5000,
              position: 'top-right',
              style: {
                background: notification.isGuest ? '#fef3c7' : '#f3f4f6',
                color: notification.isGuest ? '#92400e' : '#1f2937',
                border: notification.isGuest ? '1px solid #d97706' : '1px solid
#d1d5db'
              },
              onClick: () => {
                // Find and start chat with sender
                if (notification.senderId) {
```

```
                const sender = onlineUsers.find(u => u._id ===
notification.senderId);
                if (sender) {
                  startChat(sender);
                }
              }
            }
          });
        }
      }
    });

    // Handle user status updates
    newSocket.on('userStatusUpdate', (statusUpdate) => {
      console.log('Ø=Üd User status update received:', statusUpdate);

      setOnlineUsers(prev => prev.map(user =>
        user._id === statusUpdate.userId
          ? { ...user, status: statusUpdate.status, lastSeen:
statusUpdate.lastSeen }
          : user
      ));
    });

    // Handle typing indicators
    newSocket.on('typing', (typingData) => {
      console.log('' þ  Typing indicator received:', typingData);

      setTypingUsers(prev => ({
        ...prev,
        [typingData.senderId]: typingData.isTyping
      }));

      // Clear typing indicator after timeout
      if (typingData.isTyping) {
        setTimeout(() => {
          setTypingUsers(prev => ({
            ...prev,
            [typingData.senderId]: false
          }));
        }, 3000);
      }
    });

    // Handle message delivery confirmations
    newSocket.on('messageDelivered', (deliveryData) => {
      console.log('' Message delivered:', deliveryData);

      // Update message status in local state
      setMessages(prev => {
        const updated = { ...prev };
        Object.keys(updated).forEach(chatId => {
          updated[chatId] = updated[chatId].map(msg =>
            msg._id === deliveryData._id
              ? { ...msg, delivered: true, isOptimistic: false }
              : msg
          );
        });
        return updated;
```

```
      });
    });

    // Handle message errors
    newSocket.on('messageError', (errorData) => {
      console.error(''L Message error:', errorData);

      toast.error(`Failed to send message: ${errorData.error}`);

      // Remove optimistic message from state
      setMessages(prev => {
        const updated = { ...prev };
        Object.keys(updated).forEach(chatId => {
          updated[chatId] = updated[chatId].filter(msg => !msg.isOptimistic);
        });
        return updated;
      });
    });

    setSocket(newSocket);
    socketRef.current = newSocket;

    return () => {
      console.log('Ø=Ý  Cleaning up socket connection');
      newSocket.close();
      setSocket(null);
      socketRef.current = null;
    };
  }
}, [user, token]);

// Clean up typing timeouts on unmount
useEffect(() => {
  return () => {
    Object.values(typingTimeoutRef.current).forEach(clearTimeout);
  };
}, []);

// Listen for focus chat window events from notification service
useEffect(() => {
  const handleFocusChatWindow = () => {
    setIsChatOpen(true);
  };

  window.addEventListener('focusChatWindow', handleFocusChatWindow);

  return () => {
    window.removeEventListener('focusChatWindow', handleFocusChatWindow);
  };
}, []);

// Load notification preferences from localStorage
useEffect(() => {
  const loadPreferences = () => {
    try {
      const saved = localStorage.getItem('chatNotificationPreferences');
      if (saved) {
        const preferences = JSON.parse(saved);
        setNotificationPreferences(prev => ({ ...prev, ...preferences }));
```

```
        }
      } catch (error) {
        console.warn('Error loading chat notification preferences:', error);
      }
    };

    loadPreferences();
  }, []);

  // Save notification preferences to localStorage
  const saveNotificationPreferences = (preferences) => {
    try {
      localStorage.setItem('chatNotificationPreferences',
JSON.stringify(preferences));
      setNotificationPreferences(prev => ({ ...prev, ...preferences }));

      // Update notification service preferences
      notificationService.updatePreferences(preferences);

      toast.success('Notification preferences saved');
    } catch (error) {
      console.error('Error saving chat notification preferences:', error);
      toast.error('Failed to save notification preferences');
    }
  };

  // Send message with enhanced feedback
  const sendMessage = (recipientId, content, messageType = 'text') => {
    if (socket && user) {
      const messageData = {
        senderId: user._id,
        recipientId,
        content,
        messageType
      };

      console.log('Ø=Üä Sending message:', messageData);
      socket.emit('sendMessage', messageData);

      // Optimistically add message to local state
      const chatId = [user._id, recipientId].sort().join('_');
      const optimisticMessage = {
        _id: Date.now().toString(), // Temporary ID
        chatId,
        senderId: user,
        recipientId: { _id: recipientId },
        content,
        messageType,
        timestamp: new Date(),
        isRead: false,
        isOptimistic: true,
        delivered: false
      };

      setMessages(prev => ({
        ...prev,
        [chatId]: [...(prev[chatId] || []), optimisticMessage]
      }));
```

```
      // Play success sound for sent messages
      if (notificationPreferences.enableSounds) {
        notificationService.playNotificationSound('success');
      }
    }
  };

  // Send typing indicator
  const sendTypingIndicator = (recipientId, isTyping) => {
    if (socket && user) {
      socket.emit('typing', {
        senderId: user._id,
        recipientId,
        isTyping
      });
    }
  };

  // Update user status
  const updateUserStatus = (status) => {
    if (socket && user) {
      socket.emit('updateStatus', {
        userId: user._id,
        status
      });
    }
  };

  // Fetch chat rooms
  const fetchChatRooms = async () => {
    try {
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const apiUrl = isDevelopment ? 'http://localhost:8080/api' : 'https://crm-
backend-o36v.onrender.com/api';
      const response = await fetch(`${apiUrl}/chat/rooms`, {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      });

      if (response.ok) {
        const data = await response.json();
        setChatRooms(data.data);

        // Set unread counts
        const counts = {};
        data.data.forEach(room => {
          if (room.unreadCount > 0) {
            counts[room.otherUser._id] = room.unreadCount;
          }
        });
        setUnreadCounts(counts);
      }
    } catch (error) {
      console.error('Error fetching chat rooms:', error);
    }
  };
```

```
  // Fetch messages for a specific chat
  const fetchMessages = async (roomId) => {
    try {
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const apiUrl = isDevelopment ? 'http://localhost:8080/api' : 'https://crm-
backend-o36v.onrender.com/api';
      const response = await fetch(`${apiUrl}/chat/messages/${roomId}`, {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      });

      if (response.ok) {
        const data = await response.json();
        const chatId = [user._id, roomId].sort().join('_');
        setMessages(prev => ({
          ...prev,
          [chatId]: data.data
        }));

        // Clear unread count for this user
        setUnreadCounts(prev => ({
          ...prev,
          [roomId]: 0
        }));
      }
    } catch (error) {
      console.error('Error fetching messages:', error);
    }
  };

  // Fetch all users for chat
  const fetchAllUsers = async () => {
    try {
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const apiUrl = isDevelopment ? 'http://localhost:8080/api' : 'https://crm-
backend-o36v.onrender.com/api';
      const response = await fetch(`${apiUrl}/chat/users`, {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      });

      if (response.ok) {
        const data = await response.json();
        setOnlineUsers(data.data);
      }
    } catch (error) {
      console.error('Error fetching users:', error);
    }
  };

  // Start chat with user
  const startChat = async (otherUser) => {
```

```
  console.log('Ø=Þ€ Starting chat with:', otherUser.fullName);

  setActiveChat({
    chatId: [user._id, otherUser._id].sort().join('_'),
    otherUser
  });

  await fetchMessages(otherUser._id);
  setIsChatOpen(true);

  // Clear unread count for this user
  setUnreadCounts(prev => ({
    ...prev,
    [otherUser._id]: 0
  }));
};

// Close chat
const closeChat = () => {
  setActiveChat(null);
  setIsChatOpen(false);
};

// Test notification sound
const testNotificationSound = (type = 'message') => {
  notificationService.testNotificationSound(type);
};

// Get total unread count
const getTotalUnreadCount = () => {
  return Object.values(unreadCounts).reduce((sum, count) => sum + count, 0);
};

const value = {
  socket,
  isConnected,
  onlineUsers,
  chatRooms,
  activeChat,
  messages,
  unreadCounts,
  typingUsers,
  isChatOpen,
  notificationPreferences,
  sendMessage,
  sendTypingIndicator,
  updateUserStatus,
  fetchChatRooms,
  fetchMessages,
  fetchAllUsers,
  startChat,
  closeChat,
  setIsChatOpen,
  saveNotificationPreferences,
  testNotificationSound,
  getTotalUnreadCount
};

return (
```

```
    <ChatContext.Provider value={value}>
      {children}
    </ChatContext.Provider>
  );
};
```

src/context/ThemeContext.jsx

```
import { createContext, useEffect, useState } from 'react';

export const ThemeContext = createContext();

export const ThemeProvider = ({ children }) => {
  const getInitialTheme = () => {
    if (typeof window !== 'undefined' && window.localStorage) {
      const storedPrefs = window.localStorage.getItem('theme');
      if (typeof storedPrefs === 'string') return storedPrefs;
      const userMedia = window.matchMedia('(prefers-color-scheme: dark)');
      if (userMedia.matches) return 'dark';
    }
    return 'light';
  };

  const [theme, setTheme] = useState(getInitialTheme);

  useEffect(() => {
    const root = window.document.documentElement;
    const isDark = theme === 'dark';

    root.classList.remove(isDark ? 'light' : 'dark');
    root.classList.add(theme);

    localStorage.setItem('theme', theme);
  }, [theme]);

  const toggleTheme = () =>
    setTheme((prev) => (prev === 'dark' ? 'light' : 'dark'));

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};
```

src/hooks/useActivityTracker.js

```
import { useEffect, useRef, useCallback } from 'react';
import { useAuth } from '../context/AuthContext';
import axios from 'axios';

// API configuration
const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
const API_BASE_URL = isDevelopment ? 'http://localhost:8080/api' :
(import.meta.env.VITE_API_URL || 'https://crm-backend-o36v.onrender.com/api');

const useActivityTracker = () => {
```

```
  const { user, token } = useAuth();
  const sessionStartTime = useRef(null);
  const lastActivityTime = useRef(Date.now());
  const activityInterval = useRef(null);
  const inactivityTimeout = useRef(null);
  const isSessionActive = useRef(false);
  const hasStartedSession = useRef(false);
  const isManuallyPaused = useRef(false);
  const visibilityChangeListener = useRef(null);

  // Configuration
  const ACTIVITY_CHECK_INTERVAL = 30000; // Check every 30 seconds
  const INACTIVITY_THRESHOLD = 5 * 60 * 1000; // 5 minutes of inactivity
  const TRACK_INTERVAL = 60000; // Send tracking data every minute

  // API helper with auth headers
  const apiCall = useCallback(async (endpoint, method = 'GET', data = null) => {
    if (!token) return null;

    try {
      const config = {
        method,
        url: `${API_BASE_URL}${endpoint}`,
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      };

      if (data) {
        config.data = data;
      }

      const response = await axios(config);
      return response.data;
    } catch (error) {
      console.error('Activity API error:', error);
      return null;
    }
  }, [token]);

  // Start activity session
  const startSession = useCallback(async () => {
    if (!user || !token || hasStartedSession.current || isManuallyPaused.current)
return;

    try {
      const result = await apiCall('/activity/start-session', 'POST');
      if (result?.success) {
        sessionStartTime.current = Date.now();
        lastActivityTime.current = Date.now();
        isSessionActive.current = true;
        hasStartedSession.current = true;

        // Store session start time in localStorage
        localStorage.setItem('activitySessionStart',
sessionStartTime.current.toString());
        localStorage.setItem('activitySessionActive', 'true');
```

```
        console.log('Activity session started');
      }
    } catch (error) {
      console.error('Failed to start activity session:', error);
    }
  }, [user, token, apiCall]);

  // End activity session
  const endSession = useCallback(async (duration = null) => {
    if (!hasStartedSession.current || !sessionStartTime.current) return;

    try {
      const sessionDuration = duration || Math.floor((Date.now() -
sessionStartTime.current) / 1000);

      // For page unload, use sendBeacon with special endpoint
      if (navigator.sendBeacon && token) {
        const data = JSON.stringify({
          duration: sessionDuration,
          token // Include token in body for beacon endpoint
        });
        const url = `${API_BASE_URL}/activity/end-session-beacon`;

        const blob = new Blob([data], { type: 'application/json' });
        const sent = navigator.sendBeacon(url, blob);

        if (sent) {
          console.log('Activity session ended via beacon');
        } else {
          // Fallback to regular API call
          await apiCall('/activity/end-session', 'POST', { duration:
sessionDuration });
        }
      } else {
        // Fallback for browsers without sendBeacon
        await apiCall('/activity/end-session', 'POST', { duration:
sessionDuration });
      }

      // Reset session state
      sessionStartTime.current = null;
      isSessionActive.current = false;
      hasStartedSession.current = false;

      // Clear localStorage
      localStorage.removeItem('activitySessionStart');
      localStorage.removeItem('activitySessionActive');

      console.log('Activity session ended');
    } catch (error) {
      console.error('Failed to end activity session:', error);
    }
  }, [apiCall, token]);

  // Manual start function for employees
  const manualStart = useCallback(async () => {
    isManuallyPaused.current = false;
    localStorage.setItem('activityManuallyPaused', 'false');
    await startSession();
```

```
  }, [startSession]);

  // Manual pause function for employees
  const manualPause = useCallback(async () => {
    isManuallyPaused.current = true;
    localStorage.setItem('activityManuallyPaused', 'true');
    await endSession();
  }, [endSession]);

  // Track activity (periodic updates)
  const trackActivity = useCallback(async () => {
    if (!hasStartedSession.current || !sessionStartTime.current ||
isManuallyPaused.current) return;

    try {
      const currentTime = Date.now();
      const timeSinceLastActivity = currentTime - lastActivityTime.current;
      const isCurrentlyActive = timeSinceLastActivity < INACTIVITY_THRESHOLD;

      if (isCurrentlyActive) {
        const sessionDuration = Math.floor((currentTime -
sessionStartTime.current) / 1000);
        await apiCall('/activity/track', 'POST', {
          duration: sessionDuration,
          isActive: true
        });
      }
    } catch (error) {
      console.error('Failed to track activity:', error);
    }
  }, [apiCall, INACTIVITY_THRESHOLD]);

  // Update last activity time
  const updateActivity = useCallback(() => {
    if (isManuallyPaused.current) return;

    lastActivityTime.current = Date.now();

    // Clear existing inactivity timeout
    if (inactivityTimeout.current) {
      clearTimeout(inactivityTimeout.current);
    }

    // Set new inactivity timeout
    inactivityTimeout.current = setTimeout(() => {
      if (isSessionActive.current && !isManuallyPaused.current) {
        console.log('User inactive, pausing session tracking');
        isSessionActive.current = false;
      }
    }, INACTIVITY_THRESHOLD);
  }, [INACTIVITY_THRESHOLD]);

  // Handle page visibility change (tab switching, system lock, etc.)
  const handleVisibilityChange = useCallback(async () => {
    if (document.hidden || document.visibilityState === 'hidden') {
      // Page is hidden - could be tab switch or system lock
      console.log('Page hidden - continuing activity tracking (tab switch or
system lock)');
      // Store the time when page became hidden
```

```
      localStorage.setItem('activityPageHiddenTime', Date.now().toString());

      // DON'T pause the session - let it continue running
      // The timer should keep running even when tab is switched
    } else if (document.visibilityState === 'visible') {
      // Page is visible again
      const hiddenTime = localStorage.getItem('activityPageHiddenTime');
      if (hiddenTime) {
        const timeHidden = Date.now() - parseInt(hiddenTime);
        console.log(`Page visible again after ${Math.floor(timeHidden / 1000)}
seconds`);

        // If hidden for more than 10 minutes (600000ms), likely system was locked
        if (timeHidden > 600000) {
          console.log('System was likely locked - restarting session');
          if (user && token && !isManuallyPaused.current) {
            // End previous session and start new one
            if (hasStartedSession.current) {
              await endSession();
            }
            setTimeout(() => startSession(), 1000);
          }
        } else {
          console.log('Tab switch detected - continuing existing session');
          // Just a tab switch - continue existing session
          if (user && token && !hasStartedSession.current && !
isManuallyPaused.current) {
            startSession();
          }
        }

        localStorage.removeItem('activityPageHiddenTime');
      }
    }
  }, [user, token, startSession, endSession]);

  // Handle page unload
  const handleBeforeUnload = useCallback(() => {
    if (hasStartedSession.current && sessionStartTime.current) {
      const duration = Math.floor((Date.now() - sessionStartTime.current) / 1000);

      // Use sendBeacon for reliable tracking
      if (navigator.sendBeacon && token) {
        const data = JSON.stringify({
          duration,
          token // Include token in body for beacon endpoint
        });
        const url = `${API_BASE_URL}/activity/end-session-beacon`;

        const blob = new Blob([data], { type: 'application/json' });
        navigator.sendBeacon(url, blob);
      }
    }
  }, [token]);

  // Set up activity tracking
  useEffect(() => {
    if (!user || !token) return;
```

```
    // Check if manually paused from localStorage
    const manuallyPaused = localStorage.getItem('activityManuallyPaused') ===
'true';
    isManuallyPaused.current = manuallyPaused;

    // Check for existing session from localStorage
    const existingSessionStart = localStorage.getItem('activitySessionStart');
    const existingSessionActive = localStorage.getItem('activitySessionActive')
=== 'true';

    if (existingSessionStart && existingSessionActive && !manuallyPaused) {
      // Resume existing session
      console.log('Resuming existing activity session');
      sessionStartTime.current = parseInt(existingSessionStart);
      isSessionActive.current = true;
      hasStartedSession.current = true;
      lastActivityTime.current = Date.now();
    } else if (!manuallyPaused) {
      // Start new session
      startSession();
    }

    // Set up activity listeners
    const events = ['mousedown', 'mousemove', 'keypress', 'scroll', 'touchstart',
'click'];
    events.forEach(event => {
      document.addEventListener(event, updateActivity, true);
    });

    // Set up periodic tracking
    activityInterval.current = setInterval(trackActivity, TRACK_INTERVAL);

    // Set up visibility change listener (handles system lock/unlock)
    visibilityChangeListener.current = handleVisibilityChange;
    document.addEventListener('visibilitychange', handleVisibilityChange);

    // Set up beforeunload listener
    window.addEventListener('beforeunload', handleBeforeUnload);

    // Cleanup function
    return () => {
      // Remove event listeners
      events.forEach(event => {
        document.removeEventListener(event, updateActivity, true);
      });
      document.removeEventListener('visibilitychange', handleVisibilityChange);
      window.removeEventListener('beforeunload', handleBeforeUnload);

      // Clear intervals and timeouts
      if (activityInterval.current) {
        clearInterval(activityInterval.current);
      }
      if (inactivityTimeout.current) {
        clearTimeout(inactivityTimeout.current);
      }

      // Don't end session on component unmount - let it persist
      // Session will be ended manually or on actual logout
    };
```

```
  }, [user, token, startSession, endSession, updateActivity, trackActivity,
handleVisibilityChange, handleBeforeUnload, TRACK_INTERVAL]);

  // Return session info and controls
  return {
    isSessionActive: isSessionActive.current,
    sessionStartTime: sessionStartTime.current,
    isManuallyPaused: isManuallyPaused.current,
    startSession: manualStart,
    pauseSession: manualPause,
    updateActivity
  };
};

export default useActivityTracker;
```

src/hooks/useTheme.js

```
import { useContext } from 'react';
import { ThemeContext } from '../context/ThemeContext';

export const useTheme = () => {
  const context = useContext(ThemeContext);

  if (context === undefined) {
    throw new Error('useTheme must be used within a ThemeProvider');
  }

  return context;
};
```

src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* Force text visibility in dark mode */
.dark {
  color-scheme: dark;
}

.dark * {
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

.dark input,
.dark textarea,
.dark select {
  background-color: rgb(51 65 85) !important; /* slate-700 */
  color: rgb(248 250 252) !important; /* slate-50 */
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

.dark input::placeholder,
.dark textarea::placeholder {
  color: rgb(148 163 184) !important; /* slate-400 */
}
```

```
/* Force text colors in dark mode */
.dark h1,
.dark h2,
.dark h3,
.dark h4,
.dark h5,
.dark h6 {
  color: rgb(248 250 252) !important; /* slate-50 */
}

.dark p,
.dark span,
.dark div,
.dark label,
.dark li {
  color: rgb(203 213 225) !important; /* slate-300 */
}

.dark a {
  color: rgb(96 165 250) !important; /* blue-400 */
}

.dark a:hover {
  color: rgb(147 197 253) !important; /* blue-300 */
}

/* Force background colors */
.dark .bg-white {
  background-color: rgb(15 23 42) !important; /* slate-900 */
}

.dark .bg-gray-50 {
  background-color: rgb(30 41 59) !important; /* slate-800 */
}

.dark .bg-gray-100 {
  background-color: rgb(51 65 85) !important; /* slate-700 */
}

/* Force card styling */
.dark [class*="bg-white"] {
  background-color: rgb(15 23 42) !important; /* slate-900 */
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

/* Force button text visibility */
.dark button {
  color: rgb(248 250 252) !important; /* slate-50 */
}

.dark button:not(.bg-blue-600):not(.bg-green-600):not(.bg-red-600):not(.bg-
purple-600) {
  background-color: rgb(51 65 85) !important; /* slate-700 */
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

/* Force table text visibility */
.dark table,
```

```
.dark th,
.dark td {
  color: rgb(203 213 225) !important; /* slate-300 */
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

.dark th {
  background-color: rgb(30 41 59) !important; /* slate-800 */
  color: rgb(248 250 252) !important; /* slate-50 */
}

/* Force form styling */
.dark .form-control,
.dark .form-input,
.dark input[type="text"],
.dark input[type="email"],
.dark input[type="password"],
.dark input[type="number"],
.dark input[type="tel"],
.dark input[type="url"],
.dark input[type="search"],
.dark input[type="date"],
.dark input[type="datetime-local"],
.dark input[type="month"],
.dark input[type="week"],
.dark input[type="time"] {
  background-color: rgb(51 65 85) !important; /* slate-700 */
  color: rgb(248 250 252) !important; /* slate-50 */
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

/* Force select styling */
.dark select {
  background-color: rgb(51 65 85) !important; /* slate-700 */
  color: rgb(248 250 252) !important; /* slate-50 */
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

/* Force checkbox and radio styling */
.dark input[type="checkbox"],
.dark input[type="radio"] {
  background-color: rgb(51 65 85) !important; /* slate-700 */
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

/* Force modal and dropdown styling */
.dark .modal,
.dark .dropdown-menu {
  background-color: rgb(15 23 42) !important; /* slate-900 */
  border-color: rgb(71 85 105) !important; /* slate-600 */
  color: rgb(203 213 225) !important; /* slate-300 */
}

/* Force alert styling */
.dark .alert {
  border-color: rgb(71 85 105) !important; /* slate-600 */
}

.dark .alert-info {
```

```
  background-color: rgb(30 58 138) !important; /* blue-800 */
  color: rgb(219 234 254) !important; /* blue-100 */
}

.dark .alert-success {
  background-color: rgb(22 101 52) !important; /* green-800 */
  color: rgb(220 252 231) !important; /* green-100 */
}

.dark .alert-warning {
  background-color: rgb(146 64 14) !important; /* amber-800 */
  color: rgb(254 243 199) !important; /* amber-100 */
}

.dark .alert-error,
.dark .alert-danger {
  background-color: rgb(153 27 27) !important; /* red-800 */
  color: rgb(254 226 226) !important; /* red-100 */
}

html, body {
  height: 100%;
  width: 100%;
  margin: 0;
  padding: 0;
  overflow-x: hidden;
}

body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
```

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

src/main.jsx

```
import React, { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import { ThemeProvider } from './context/ThemeContext.jsx'
import App from './App.jsx'

// This is the main entry point for Vite builds
const root = createRoot(document.getElementById('root'))
root.render(
  <StrictMode>
    <ThemeProvider>
      <App />
    </ThemeProvider>
  </StrictMode>,
)
```

src/pages/AdminActivityLogsPage.jsx

```
import React, { useState, useEffect } from 'react';
import { toast } from 'react-toastify';
import Layout from '../components/Layout/Layout';
import { FaFilter, FaSync, FaCalendar, FaUser, FaTag, FaCheck, FaTimes,
FaExclamationTriangle } from 'react-icons/fa';
import api from '../services/api';

const AdminActivityLogsPage = () => {
  const [logs, setLogs] = useState([]);
  const [loading, setLoading] = useState(true);
  const [stats, setStats] = useState(null);
  const [filters, setFilters] = useState({
    action: '',
    performedBy: '',
    status: '',
    startDate: '',
    endDate: '',
    affectedResource: ''
  });
  const [pagination, setPagination] = useState({
    page: 1,
    limit: 50,
    total: 0,
    totalPages: 0
  });
  const [showFilters, setShowFilters] = useState(false);

  // Fetch logs on mount and when filters change
  useEffect(() => {
    fetchLogs();
    fetchStats();
  }, [filters, pagination.page]);

  const fetchLogs = async () => {
    try {
      setLoading(true);
      const queryParams = new URLSearchParams({
        page: pagination.page,
        limit: pagination.limit,
        ...filters
```

```
      });

      const response = await api.get(`/logs?${queryParams}`);

      if (response.data.success) {
        setLogs(response.data.data);
        setPagination(prev => ({
          ...prev,
          total: response.data.total,
          totalPages: response.data.pagination.totalPages
        }));
      }
    } catch (error) {
      toast.error('Failed to fetch logs');
      console.error('Error fetching logs:', error);
    } finally {
      setLoading(false);
    }
  };

  const fetchStats = async () => {
    try {
      const response = await api.get('/logs/stats');
      if (response.data.success) {
        setStats(response.data.data);
      }
    } catch (error) {
      console.error('Error fetching stats:', error);
    }
  };

  const handleFilterChange = (e) => {
    const { name, value } = e.target;
    setFilters(prev => ({ ...prev, [name]: value }));
    setPagination(prev => ({ ...prev, page: 1 })); // Reset to first page on
filter change
  };

  const resetFilters = () => {
    setFilters({
      action: '',
      performedBy: '',
      status: '',
      startDate: '',
      endDate: '',
      affectedResource: ''
    });
    setPagination(prev => ({ ...prev, page: 1 }));
  };

  const formatDate = (dateString) => {
    return new Date(dateString).toLocaleString();
  };

  const getStatusIcon = (status) => {
    switch (status) {
      case 'SUCCESS':
        return <FaCheck className="text-green-500" />;
      case 'FAILURE':
```

```
        return <FaTimes className="text-red-500" />;
      case 'WARNING':
        return <FaExclamationTriangle className="text-yellow-500" />;
      default:
        return null;
    }
  };

  const renderActionStats = () => {
    if (!stats?.actionStats) return null;

    return (
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4 mb-6">
        {stats.actionStats.map(stat => (
          <div key={stat._id} className="bg-white dark:bg-gray-800 p-4 rounded-lg
shadow">
            <h3 className="font-semibold text-gray-700 dark:text-
gray-300">{stat._id}</h3>
            <div className="mt-2 text-sm">
              <div className="flex justify-between">
                <span>Total:</span>
                <span className="font-medium">{stat.count}</span>
              </div>
              <div className="flex justify-between text-green-600">
                <span>Success:</span>
                <span>{stat.successCount}</span>
              </div>
              <div className="flex justify-between text-red-600">
                <span>Failure:</span>
                <span>{stat.failureCount}</span>
              </div>
            </div>
          </div>
        ))}
      </div>
    );
  };

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-2xl font-bold">Activity Logs</h1>
          <div className="flex space-x-2">
            <button
              onClick={() => setShowFilters(!showFilters)}
              className="flex items-center px-4 py-2 bg-blue-600 text-white
rounded hover:bg-blue-700"
            >
              <FaFilter className="mr-2" />
              {showFilters ? 'Hide Filters' : 'Show Filters'}
            </button>
            <button
              onClick={fetchLogs}
              className="flex items-center px-4 py-2 bg-gray-600 text-white
rounded hover:bg-gray-700"
            >
              <FaSync className="mr-2" />
              Refresh
```

```
            </button>
          </div>
        </div>

        {/* Stats Summary */}
        {stats && (
          <div className="mb-6 p-4 bg-white dark:bg-gray-800 rounded-lg shadow">
            <h2 className="text-xl font-semibold mb-4">Activity Summary</h2>
            <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
              <div className="p-4 bg-blue-50 dark:bg-blue-900 rounded-lg">
                <div className="text-blue-600 dark:text-blue-300 text-lg font-
semibold">
                  Today's Activities
                </div>
                <div className="text-2xl font-bold">{stats.todayCount}</div>
              </div>
              <div className="p-4 bg-green-50 dark:bg-green-900 rounded-lg">
                <div className="text-green-600 dark:text-green-300 text-lg font-
semibold">
                  Total Users Active
                </div>
                <div className="text-2xl font-bold">{stats.topUsers?.length || 0}
</div>
              </div>
              <div className="p-4 bg-purple-50 dark:bg-purple-900 rounded-lg">
                <div className="text-purple-600 dark:text-purple-300 text-lg font-
semibold">
                  Total Actions
                </div>
                <div className="text-2xl font-bold">
                  {stats.actionStats?.reduce((sum, stat) => sum + stat.count, 0)
|| 0}
                </div>
              </div>
            </div>
          </div>
        )}

        {/* Action Stats */}
        {renderActionStats()}

        {/* Filters */}
        {showFilters && (
          <div className="mb-6 p-4 bg-white dark:bg-gray-800 rounded-lg shadow">
            <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Action Type
                </label>
                <select
                  name="action"
                  value={filters.action}
                  onChange={handleFilterChange}
                  className="w-full border border-gray-300 rounded-md shadow-sm
p-2"
                >
                  <option value="">All Actions</option>
                  <option value="LOGIN">Login</option>
```

```
                  <option value="LOGOUT">Logout</option>
                  <option value="LEAD_CREATE">Lead Created</option>
                  <option value="LEAD_UPDATE">Lead Updated</option>
                  <option value="SALE_CREATE">Sale Created</option>
                  <option value="SALE_UPDATE">Sale Updated</option>
                  {/* Add other action types */}
                </select>
              </div>
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Status
                </label>
                <select
                  name="status"
                  value={filters.status}
                  onChange={handleFilterChange}
                  className="w-full border border-gray-300 rounded-md shadow-sm
p-2"
                >
                  <option value="">All Status</option>
                  <option value="SUCCESS">Success</option>
                  <option value="FAILURE">Failure</option>
                  <option value="WARNING">Warning</option>
                </select>
              </div>
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Resource Type
                </label>
                <select
                  name="affectedResource"
                  value={filters.affectedResource}
                  onChange={handleFilterChange}
                  className="w-full border border-gray-300 rounded-md shadow-sm
p-2"
                >
                  <option value="">All Resources</option>
                  <option value="USER">User</option>
                  <option value="LEAD">Lead</option>
                  <option value="SALE">Sale</option>
                  <option value="EMPLOYEE">Employee</option>
                </select>
              </div>
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Start Date
                </label>
                <input
                  type="date"
                  name="startDate"
                  value={filters.startDate}
                  onChange={handleFilterChange}
                  className="w-full border border-gray-300 rounded-md shadow-sm
p-2"
                />
              </div>
```

```
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  End Date
                </label>
                <input
                  type="date"
                  name="endDate"
                  value={filters.endDate}
                  onChange={handleFilterChange}
                  className="w-full border border-gray-300 rounded-md shadow-sm
p-2"
                />
              </div>
              <div className="flex items-end">
                <button
                  onClick={resetFilters}
                  className="px-4 py-2 bg-gray-200 text-gray-700 rounded hover:bg-
gray-300"
                >
                  Reset Filters
                </button>
              </div>
            </div>
          </div>
        )}

        {/* Logs Table */}
        <div className="bg-white dark:bg-gray-800 shadow rounded-lg overflow-
hidden">
          <div className="overflow-x-auto">
            <table className="min-w-full divide-y divide-gray-200 dark:divide-
gray-700">
              <thead className="bg-gray-50 dark:bg-gray-900">
                <tr>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                    Timestamp
                  </th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                    Action
                  </th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                    User
                  </th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                    Resource
                  </th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                    Status
                  </th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                    Details
                  </th>
```

```
                </tr>
              </thead>
              <tbody className="bg-white dark:bg-gray-800 divide-y divide-
gray-200 dark:divide-gray-700">
                {loading ? (
                  <tr>
                    <td colSpan="6" className="px-6 py-4 text-center">
                      Loading...
                    </td>
                  </tr>
                ) : logs.length === 0 ? (
                  <tr>
                    <td colSpan="6" className="px-6 py-4 text-center">
                      No logs found
                    </td>
                  </tr>
                ) : (
                  logs.map((log) => (
                    <tr key={log._id} className="hover:bg-gray-50 dark:hover:bg-
gray-700">
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-400">
                        {formatDate(log.timestamp)}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm font-
medium text-gray-900 dark:text-white">
                        {log.action}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-400">
                        {log.performedBy?.fullName || log.performedBy?.email ||
'Unknown'}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-400">
                        {log.affectedResource}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap">
                        <div className="flex items-center">
                          {getStatusIcon(log.status)}
                          <span className={`ml-2 text-sm ${
                            log.status === 'SUCCESS' ? 'text-green-600 dark:text-
green-400' :
                            log.status === 'FAILURE' ? 'text-red-600 dark:text-
red-400' :
                            'text-yellow-600 dark:text-yellow-400'
                          }`}>
                            {log.status}
                          </span>
                        </div>
                      </td>
                      <td className="px-6 py-4 text-sm text-gray-500 dark:text-
gray-400">
                        <div className="max-w-xs overflow-hidden text-ellipsis">
                          {typeof log.details === 'string' ? log.details :
JSON.stringify(log.details)}
                        </div>
                      </td>
                    </tr>
```

```
                  ))
                )}
              </tbody>
            </table>
          </div>

          {/* Pagination */}
          <div className="bg-white dark:bg-gray-800 px-4 py-3 border-t border-
gray-200 dark:border-gray-700 sm:px-6">
            <div className="flex items-center justify-between">
              <div className="flex-1 flex justify-between sm:hidden">
                <button
                  onClick={() => setPagination(prev => ({ ...prev, page:
prev.page - 1 }))}
                  disabled={pagination.page === 1}
                  className="relative inline-flex items-center px-4 py-2 border
border-gray-300 text-sm font-medium rounded-md text-gray-700 bg-white hover:bg-
gray-50"
                >
                  Previous
                </button>
                <button
                  onClick={() => setPagination(prev => ({ ...prev, page:
prev.page + 1 }))}
                  disabled={pagination.page === pagination.totalPages}
                  className="ml-3 relative inline-flex items-center px-4 py-2
border border-gray-300 text-sm font-medium rounded-md text-gray-700 bg-white
hover:bg-gray-50"
                >
                  Next
                </button>
              </div>
              <div className="hidden sm:flex-1 sm:flex sm:items-center sm:justify-
between">
                <div>
                  <p className="text-sm text-gray-700 dark:text-gray-300">
                    Showing{' '}
                    <span className="font-medium">
                      {((pagination.page - 1) * pagination.limit) + 1}
                    </span>
                    {' '}to{' '}
                    <span className="font-medium">
                      {Math.min(pagination.page * pagination.limit,
pagination.total)}
                    </span>
                    {' '}of{' '}
                    <span className="font-medium">{pagination.total}</span>
                    {' '}results
                  </p>
                </div>
                <div>
                  <nav className="relative z-0 inline-flex rounded-md shadow-sm -
space-x-px" aria-label="Pagination">
                    <button
                      onClick={() => setPagination(prev => ({ ...prev, page:
prev.page - 1 }))}
                      disabled={pagination.page === 1}
                      className="relative inline-flex items-center px-2 py-2
rounded-l-md border border-gray-300 bg-white text-sm font-medium text-gray-500
```

```
hover:bg-gray-50"
                    >
                      Previous
                    </button>
                    {/* Add page numbers here if needed */}
                    <button
                      onClick={() => setPagination(prev => ({ ...prev, page:
prev.page + 1 }))}
                      disabled={pagination.page === pagination.totalPages}
                      className="relative inline-flex items-center px-2 py-2
rounded-r-md border border-gray-300 bg-white text-sm font-medium text-gray-500
hover:bg-gray-50"
                    >
                      Next
                    </button>
                  </nav>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </Layout>
  );
};

export default AdminActivityLogsPage;
```

src/pages/AdminActivityPage.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../context/AuthContext';
import Layout from '../components/Layout/Layout';
import { activityAPI } from '../services/api';
import toast from 'react-hot-toast';

const AdminActivityPage = () => {
  const { user } = useAuth();
  const [loading, setLoading] = useState(true);
  const [activeTab, setActiveTab] = useState('today');
  const [selectedDate, setSelectedDate] = useState(new
Date().toISOString().split('T')[0]);
  const [dateRange, setDateRange] = useState({
    startDate: new Date(Date.now() - 6 * 24 * 60 * 60 *
1000).toISOString().split('T')[0],
    endDate: new Date().toISOString().split('T')[0]
  });
  const [statisticsPeriod, setStatisticsPeriod] = useState(7);
  const [selectedEmployee, setSelectedEmployee] = useState(null);
  const [employeeDetailDate, setEmployeeDetailDate] = useState(new
Date().toISOString().split('T')[0]);

  // State for different data types
  const [todayActivity, setTodayActivity] = useState({
    data: [],
    summary: {
      totalUsers: 0,
      activeUsers: 0,
```

```
    totalActiveTime: 0,
    averageActiveTime: 0
  }
});
const [dateRangeActivity, setDateRangeActivity] = useState([]);
const [statistics, setStatistics] = useState({
  dailyStats: [],
  userStats: [],
  period: { startDate: '', endDate: '', days: 0 }
});
const [employeeDetails, setEmployeeDetails] = useState(null);

// Fetch today's activity
const fetchTodayActivity = async () => {
  try {
    const response = await activityAPI.getAllUsersActivity(selectedDate);
    if (response.data.success) {
      setTodayActivity({
        data: response.data.data || [],
        summary: response.data.summary || {
          totalUsers: 0,
          activeUsers: 0,
          totalActiveTime: 0,
          averageActiveTime: 0
        }
      });
    }
  } catch (error) {
    console.error('Error fetching today activity:', error);
    toast.error('Failed to fetch today\'s activity data');
  }
};

// Fetch date range activity
const fetchDateRangeActivity = async () => {
  try {
    const response = await activityAPI.getAllUsersActivity(
      null,
      dateRange.startDate,
      dateRange.endDate
    );
    if (response.data.success) {
      setDateRangeActivity(response.data.data || []);
    }
  } catch (error) {
    console.error('Error fetching date range activity:', error);
    toast.error('Failed to fetch date range activity data');
  }
};

// Fetch statistics
const fetchStatistics = async () => {
  try {
    const response = await activityAPI.getStatistics(statisticsPeriod);
    if (response.data.success) {
      setStatistics(response.data.data);
    }
  } catch (error) {
    console.error('Error fetching statistics:', error);
```

```
      toast.error('Failed to fetch activity statistics');
    }
  };

  // Fetch individual employee details
  const fetchEmployeeDetails = async (userId, date) => {
    try {
      const response = await activityAPI.getAllUsersActivity(date);
      if (response.data.success) {
        const employeeData = response.data.data.find(emp => emp.userId ===
userId);
        setEmployeeDetails(employeeData);
      }
    } catch (error) {
      console.error('Error fetching employee details:', error);
      toast.error('Failed to fetch employee details');
    }
  };

  // Load data based on active tab
  const loadData = async () => {
    setLoading(true);
    try {
      switch (activeTab) {
        case 'today':
          await fetchTodayActivity();
          break;
        case 'range':
          await fetchDateRangeActivity();
          break;
        case 'statistics':
          await fetchStatistics();
          break;
        case 'employee':
          if (selectedEmployee) {
            await fetchEmployeeDetails(selectedEmployee.userId,
employeeDetailDate);
          }
          break;
        default:
          await fetchTodayActivity();
      }
    } finally {
      setLoading(false);
    }
  };

  // Initial load and refresh
  useEffect(() => {
    loadData();

    // Auto-refresh every 30 seconds for today's data
    const interval = setInterval(() => {
      if (activeTab === 'today') {
        fetchTodayActivity();
      }
    }, 30000);

    return () => clearInterval(interval);
```

```
  }, [activeTab, selectedDate, dateRange, statisticsPeriod, selectedEmployee,
employeeDetailDate]);

  // Enhanced format duration helper with hours and minutes
  const formatDuration = (seconds) => {
    if (!seconds || seconds < 0) return '0m';

    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);

    if (hours > 0) {
      return `${hours}h ${minutes}m`;
    } else {
      return `${minutes}m`;
    }
  };

  // Format duration with seconds for detailed view
  const formatDetailedDuration = (seconds) => {
    if (!seconds || seconds < 0) return '0h 0m 0s';

    const hours = Math.floor(seconds / 3600);
    const minutes = Math.floor((seconds % 3600) / 60);
    const secs = seconds % 60;

    return `${hours}h ${minutes}m ${secs}s`;
  };

  // Format date helper
  const formatDate = (dateString) => {
    return new Date(dateString).toLocaleDateString('en-US', {
      year: 'numeric',
      month: 'short',
      day: 'numeric'
    });
  };

  // Format time helper
  const formatTime = (dateString) => {
    return new Date(dateString).toLocaleTimeString('en-US', {
      hour: '2-digit',
      minute: '2-digit'
    });
  };

  // Get status badge color
  const getStatusBadge = (isActive) => {
    return isActive
      ? 'bg-green-100 text-green-800 border-green-200'
      : 'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-gray-200 border-
gray-200 dark:border-slate-700';
  };

  // Get role badge color
  const getRoleBadge = (role) => {
    const colors = {
      'Admin': 'bg-red-100 text-red-800 border-red-200',
      'Manager': 'bg-blue-100 text-blue-800 border-blue-200',
      'Sales Person': 'bg-green-100 text-green-800 border-green-200',
```

```
      'Lead Person': 'bg-yellow-100 text-yellow-800 border-yellow-200'
    };
    return colors[role] || 'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-
gray-200 border-gray-200 dark:border-slate-700';
  };

  // Handle employee selection for detailed view
  const handleEmployeeSelect = (employee) => {
    setSelectedEmployee(employee);
    setActiveTab('employee');
  };

  if (!user || (user.role !== 'Admin' && user.role !== 'Manager')) {
    return (
      <Layout>
        <div className="container mx-auto p-6">
          <div className="bg-red-50 border border-red-200 rounded-lg p-4">
            <h2 className="text-red-800 font-semibold">Access Denied</h2>
            <p className="text-red-600">You don't have permission to view this
page.</p>
          </div>
        </div>
      </Layout>
    );
  }

  return (
    <Layout>
      <div className="container mx-auto p-6">
        {/* Header */}
        <div className="flex justify-between items-center mb-6">
          <div>
            <h1 className="text-3xl font-bold text-gray-900 dark:text-white">User
Activity Dashboard</h1>
            <p className="text-gray-600 dark:text-gray-500 mt-1">Monitor employee
CRM usage and productivity</p>
          </div>
          <div className="flex items-center space-x-2">
            <button
              onClick={loadData}
              disabled={loading}
              className="bg-blue-600 hover:bg-blue-700 text-white px-4 py-2
rounded-lg flex items-center space-x-2 disabled:opacity-50"
            >
              <svg className={`w-4 h-4 ${loading ? 'animate-spin' : ''}`}
fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M4 4v5h.582m15.356 2A8.001 8.001 0 004.582 9m0 0H9m11
11v-5h-.581m0 0a8.003 8.003 0 01-15.357-2m15.357 2H15" />
              </svg>
              <span>Refresh</span>
            </button>
          </div>
        </div>

        {/* Tabs */}
        <div className="border-b border-gray-200 dark:border-slate-700 mb-6">
          <nav className="-mb-px flex space-x-8">
            {[
```

```
              { id: 'today', label: 'Today\'s Activity', icon: 'Ø=ÜÊ' },
              { id: 'range', label: 'Date Range', icon: 'Ø=ÜÅ' },
              { id: 'statistics', label: 'Statistics', icon: 'Ø=ÜÈ' },
              { id: 'employee', label: 'Employee Details', icon: 'Ø=Üd' }
            ].map((tab) => (
              <button
                key={tab.id}
                onClick={() => setActiveTab(tab.id)}
                className={`py-2 px-1 border-b-2 font-medium text-sm flex items-
center space-x-2 ${
                  activeTab === tab.id
                    ? 'border-blue-500 text-blue-600'
                    : 'border-transparent text-gray-500 dark:text-gray-400
dark:text-gray-400 hover:text-gray-700 dark:text-gray-300 dark:text-gray-400
hover:border-gray-300 dark:border-slate-600'
                }`}
              >
                <span>{tab.icon}</span>
                <span>{tab.label}</span>
              </button>
            ))}
          </nav>
        </div>

        {/* Today's Activity Tab */}
        {activeTab === 'today' && (
          <div className="space-y-6">
            {/* Date Selector */}
            <div className="flex items-center space-x-4">
              <label className="text-sm font-medium text-gray-700 dark:text-
gray-400">Select Date:</label>
              <input
                type="date"
                value={selectedDate}
                onChange={(e) => setSelectedDate(e.target.value)}
                className="border border-gray-300 dark:border-slate-600 rounded-
lg px-3 py-2 text-sm"
              />
            </div>

            {/* Summary Cards */}
            <div className="grid grid-cols-1 md:grid-cols-4 gap-6">
              <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow p-6 border">
                <div className="flex items-center">
                  <div className="p-2 bg-blue-100 rounded-lg">
                    <svg className="w-6 h-6 text-blue-600" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 4.354a4 4 0 110 5.292M15 21H3v-1a6 6 0 0112 0v1zm0
0h6v-1a6 6 0 00-9-5.197m13.5-9a2.5 2.5 0 11-5 0 2.5 2.5 0 015 0z" />
                    </svg>
                  </div>
                  <div className="ml-4">
                    <p className="text-sm font-medium text-gray-600 dark:text-
gray-500">Total Users</p>
                    <p className="text-2xl font-bold text-gray-900 dark:text-
white">{todayActivity.summary.totalUsers}</p>
                  </div>
```

```
              </div>
            </div>

            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow p-6 border">
              <div className="flex items-center">
                <div className="p-2 bg-green-100 rounded-lg">
                  <svg className="w-6 h-6 text-green-600" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 12l2 2 4-4m6 2a9 9 0 11-18 0 9 9 0 0118 0z" />
                  </svg>
                </div>
                <div className="ml-4">
                  <p className="text-sm font-medium text-gray-600 dark:text-
gray-500">Currently Active</p>
                  <p className="text-2xl font-bold text-gray-900 dark:text-
white">{todayActivity.summary.activeUsers}</p>
                </div>
              </div>
            </div>

            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow p-6 border">
              <div className="flex items-center">
                <div className="p-2 bg-purple-100 rounded-lg">
                  <svg className="w-6 h-6 text-purple-600" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 8v4l3 3m6-3a9 9 0 11-18 0 9 9 0 0118 0z" />
                  </svg>
                </div>
                <div className="ml-4">
                  <p className="text-sm font-medium text-gray-600 dark:text-
gray-500">Total Active Time</p>
                  <p className="text-2xl font-bold text-gray-900 dark:text-
white">{formatDuration(todayActivity.summary.totalActiveTime)}</p>
                </div>
              </div>
            </div>

            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow p-6 border">
              <div className="flex items-center">
                <div className="p-2 bg-orange-100 rounded-lg">
                  <svg className="w-6 h-6 text-orange-600" fill="none"
stroke="currentColor" viewBox="0 0 24 24">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0
002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2
2 0 012-2h2a2 2 0 012 2v4a2 2 0 01-2 2h-2a2 2 0 01-2-2z" />
                  </svg>
                </div>
                <div className="ml-4">
                  <p className="text-sm font-medium text-gray-600 dark:text-
gray-500">Average Time</p>
                  <p className="text-2xl font-bold text-gray-900 dark:text-
white">{formatDuration(todayActivity.summary.averageActiveTime)}</p>
                </div>
```

```
              </div>
            </div>
          </div>

          {/* User Activity Table */}
          <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow border">
            <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
              <h3 className="text-lg font-medium text-gray-900 dark:text-
white">User Activity Details</h3>
              <p className="text-sm text-gray-600 dark:text-gray-500">Activity
for {formatDate(selectedDate)}</p>
            </div>

            {loading ? (
              <div className="p-8 text-center">
                <div className="animate-spin rounded-full h-8 w-8 border-b-2
border-blue-600 mx-auto"></div>
                <p className="text-gray-600 dark:text-gray-500 mt-2">Loading
activity data...</p>
              </div>
            ) : todayActivity.data.length === 0 ? (
              <div className="p-8 text-center">
                <svg className="w-12 h-12 text-gray-400 dark:text-gray-400 mx-
auto mb-4" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M20 13V6a2 2 0 00-2-2H6a2 2 0 00-2 2v7m16 0v5a2 2 0 01-2 2H6a2
2 0 01-2-2v-5m16 0h-2.586a1 1 0 00-.707.293l-2.414 2.414a1 1 0
01-.707.293h-3.172a1 1 0 01-.707-.293l-2.414-2.414A1 1 0 006.586 13H4" />
                </svg>
                <p className="text-gray-600 dark:text-gray-500">No activity
data found for this date</p>
              </div>
            ) : (
              <div className="overflow-x-auto">
                <table className="min-w-full divide-y divide-gray-200
dark:divide-slate-700">
                  <thead className="bg-gray-50 dark:bg-slate-800">
                    <tr>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">User</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Role</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Status</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Active Time</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Sessions</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Last Activity</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Actions</th>
                    </tr>
                  </thead>
                  <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-gray-200 dark:divide-slate-700">
                    {todayActivity.data.map((activity) => (
```

```
                        <tr key={activity.userId} className="hover:bg-gray-50
dark:bg-slate-800">
                          <td className="px-6 py-4 whitespace-nowrap">
                            <div>
                              <div className="text-sm font-medium text-gray-900
dark:text-white">{activity.userName}</div>
                              <div className="text-sm text-gray-500 dark:text-
gray-500">{activity.userEmail}</div>
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap">
                            <span className={`inline-flex px-2 py-1 text-xs font-
semibold rounded-full border ${getRoleBadge(activity.userRole)}`}>
                              {activity.userRole}
                            </span>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap">
                            <span className={`inline-flex px-2 py-1 text-xs font-
semibold rounded-full border ${getStatusBadge(activity.isCurrentlyActive)}`}>
                              {activity.isCurrentlyActive ? 'Ø=ßâ Active' : '&«
Offline'}
                            </span>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            <div>
                              <div className="font-
medium">{formatDuration(activity.totalActiveTime)}</div>
                              <div className="text-xs text-gray-500 dark:text-
gray-500">{formatDetailedDuration(activity.totalActiveTime)}</div>
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            {activity.sessionsCount}
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-500">
                            {activity.lastActivity ?
formatTime(activity.lastActivity) : 'N/A'}
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-500">
                            <button
                              onClick={() => handleEmployeeSelect(activity)}
                              className="bg-blue-600 hover:bg-blue-700 text-white
px-3 py-1 rounded text-xs"
                            >
                              View Details
                            </button>
                          </td>
                        </tr>
                      ))}
                    </tbody>
                  </table>
                </div>
              )}
            </div>
          </div>
```

```
        )}

        {/* Date Range Tab */}
        {activeTab === 'range' && (
          <div className="space-y-6">
            {/* Date Range Selector */}
            <div className="flex items-center space-x-4">
              <label className="text-sm font-medium text-gray-700 dark:text-
gray-400">From:</label>
              <input
                type="date"
                value={dateRange.startDate}
                onChange={(e) => setDateRange(prev => ({ ...prev, startDate:
e.target.value }))}
                className="border border-gray-300 dark:border-slate-600 rounded-
lg px-3 py-2 text-sm"
              />
              <label className="text-sm font-medium text-gray-700 dark:text-
gray-400">To:</label>
              <input
                type="date"
                value={dateRange.endDate}
                onChange={(e) => setDateRange(prev => ({ ...prev, endDate:
e.target.value }))}
                className="border border-gray-300 dark:border-slate-600 rounded-
lg px-3 py-2 text-sm"
              />
            </div>

            {/* Date Range Activity Table */}
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow border">
              <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
                <h3 className="text-lg font-medium text-gray-900 dark:text-
white">Activity by Date Range</h3>
                <p className="text-sm text-gray-600 dark:text-gray-500">
                  {formatDate(dateRange.startDate)} to
{formatDate(dateRange.endDate)}
                </p>
              </div>

              {loading ? (
                <div className="p-8 text-center">
                  <div className="animate-spin rounded-full h-8 w-8 border-b-2
border-blue-600 mx-auto"></div>
                  <p className="text-gray-600 dark:text-gray-500 mt-2">Loading
date range data...</p>
                </div>
              ) : dateRangeActivity.length === 0 ? (
                <div className="p-8 text-center">
                  <svg className="w-12 h-12 text-gray-400 dark:text-gray-400 mx-
auto mb-4" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M20 13V6a2 2 0 00-2-2H6a2 2 0 00-2 2v7m16 0v5a2 2 0 01-2 2H6a2
2 0 01-2-2v-5m16 0h-2.586a1 1 0 00-.707.293l-2.414 2.414a1 1 0
01-.707.293h-3.172a1 1 0 01-.707-.293l-2.414-2.414A1 1 0 006.586 13H4" />
                  </svg>
                  <p className="text-gray-600 dark:text-gray-500">No activity
```

```
data found for this date range</p>
                </div>
              ) : (
                <div className="overflow-x-auto">
                  <table className="min-w-full divide-y divide-gray-200
dark:divide-slate-700">
                    <thead className="bg-gray-50 dark:bg-slate-800">
                      <tr>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">User</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Role</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Date</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Active Time</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Sessions</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Last Activity</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Actions</th>
                      </tr>
                    </thead>
                    <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-gray-200 dark:divide-slate-700">
                      {dateRangeActivity.map((activity, index) => (
                        <tr key={`${activity.userId}-${activity.date}-${index}`}
className="hover:bg-gray-50 dark:bg-slate-800">
                          <td className="px-6 py-4 whitespace-nowrap">
                            <div>
                              <div className="text-sm font-medium text-gray-900
dark:text-white">{activity.userName}</div>
                              <div className="text-sm text-gray-500 dark:text-
gray-500">{activity.userEmail}</div>
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap">
                            <span className={`inline-flex px-2 py-1 text-xs font-
semibold rounded-full border ${getRoleBadge(activity.userRole)}`}>
                              {activity.userRole}
                            </span>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            {formatDate(activity.date)}
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            <div>
                              <div className="font-
medium">{formatDuration(activity.totalActiveTime)}</div>
                              <div className="text-xs text-gray-500 dark:text-
gray-500">{formatDetailedDuration(activity.totalActiveTime)}</div>
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            {activity.sessionsCount}
```

```
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-500">
                            {activity.lastActivity ?
formatTime(activity.lastActivity) : 'N/A'}
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-500">
                            <button
                              onClick={() => {
                                setEmployeeDetailDate(activity.date);
                                handleEmployeeSelect(activity);
                              }}
                              className="bg-blue-600 hover:bg-blue-700 text-white
px-3 py-1 rounded text-xs"
                            >
                              View Details
                            </button>
                          </td>
                        </tr>
                      ))}
                    </tbody>
                  </table>
                </div>
              )}
            </div>
          </div>
        )}

        {/* Statistics Tab */}
        {activeTab === 'statistics' && (
          <div className="space-y-6">
            {/* Period Selector */}
            <div className="flex items-center space-x-4">
              <label className="text-sm font-medium text-gray-700 dark:text-
gray-400">Period:</label>
              {[7, 14, 30].map((days) => (
                <button
                  key={days}
                  onClick={() => setStatisticsPeriod(days)}
                  className={`px-4 py-2 rounded-lg text-sm font-medium ${
                    statisticsPeriod === days
                      ? 'bg-blue-600 text-white'
                      : 'bg-gray-100 dark:bg-slate-700 text-gray-700 dark:text-
gray-300 dark:text-gray-400 hover:bg-gray-200 dark:bg-slate-600'
                  }`}
                >
                  {days} days
                </button>
              ))}
            </div>

            {/* Daily Statistics */}
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow border">
              <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
                <h3 className="text-lg font-medium text-gray-900 dark:text-
white">Daily Activity Trends</h3>
```

```
                <p className="text-sm text-gray-600 dark:text-gray-500">Last
{statisticsPeriod} days</p>
              </div>

              {loading ? (
                <div className="p-8 text-center">
                  <div className="animate-spin rounded-full h-8 w-8 border-b-2
border-blue-600 mx-auto"></div>
                </div>
              ) : (
                <div className="overflow-x-auto">
                  <table className="min-w-full divide-y divide-gray-200
dark:divide-slate-700">
                    <thead className="bg-gray-50 dark:bg-slate-800">
                      <tr>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Date</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Active Users</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Total Time</th>
                        <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Average Time</th>
                      </tr>
                    </thead>
                    <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-gray-200 dark:divide-slate-700">
                      {statistics.dailyStats.map((day) => (
                        <tr key={day.date} className="hover:bg-gray-50 dark:bg-
slate-800">
                          <td className="px-6 py-4 whitespace-nowrap text-sm font-
medium text-gray-900 dark:text-white">
                            {formatDate(day.date)}
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            {day.totalUsers}
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            <div>
                              <div className="font-
medium">{formatDuration(day.totalActiveTime)}</div>
                              <div className="text-xs text-gray-500 dark:text-
gray-500">{formatDetailedDuration(day.totalActiveTime)}</div>
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            <div>
                              <div className="font-
medium">{formatDuration(day.averageActiveTime)}</div>
                              <div className="text-xs text-gray-500 dark:text-
gray-500">{formatDetailedDuration(day.averageActiveTime)}</div>
                            </div>
                          </td>
                        </tr>
                      ))}
                    </tbody>
```

```
                </table>
              </div>
            )}
          </div>

          {/* User Statistics */}
          <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow border">
            <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
              <h3 className="text-lg font-medium text-gray-900 dark:text-
white">Top Active Users</h3>
              <p className="text-sm text-gray-600 dark:text-gray-500">Last
{statisticsPeriod} days</p>
            </div>

            {loading ? (
              <div className="p-8 text-center">
                <div className="animate-spin rounded-full h-8 w-8 border-b-2
border-blue-600 mx-auto"></div>
              </div>
            ) : (
              <div className="overflow-x-auto">
                <table className="min-w-full divide-y divide-gray-200
dark:divide-slate-700">
                  <thead className="bg-gray-50 dark:bg-slate-800">
                    <tr>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">User</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Role</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Active Days</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Total Time</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Daily Average</th>
                      <th className="px-6 py-3 text-left text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">Actions</th>
                    </tr>
                  </thead>
                  <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-gray-200 dark:divide-slate-700">
                    {statistics.userStats.map((user) => (
                      <tr key={user.userId} className="hover:bg-gray-50 dark:bg-
slate-800">
                        <td className="px-6 py-4 whitespace-nowrap">
                          <div>
                            <div className="text-sm font-medium text-gray-900
dark:text-white">{user.userName}</div>
                              <div className="text-sm text-gray-500 dark:text-
gray-500">{user.userEmail}</div>
                          </div>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap">
                          <span className={`inline-flex px-2 py-1 text-xs font-
semibold rounded-full border ${getRoleBadge(user.userRole)}`}>
                            {user.userRole}
                          </span>
```

```
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            {user.totalDays}
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            <div>
                              <div className="font-
medium">{formatDuration(user.totalActiveTime)}</div>
                              <div className="text-xs text-gray-500 dark:text-
gray-500">{formatDetailedDuration(user.totalActiveTime)}</div>
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-900 dark:text-white">
                            <div>
                              <div className="font-
medium">{formatDuration(user.averageActiveTime)}</div>
                              <div className="text-xs text-gray-500 dark:text-
gray-500">{formatDetailedDuration(user.averageActiveTime)}</div>
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-500">
                            <button
                              onClick={() => handleEmployeeSelect(user)}
                              className="bg-blue-600 hover:bg-blue-700 text-white
px-3 py-1 rounded text-xs"
                            >
                              View Details
                            </button>
                          </td>
                        </tr>
                      ))}
                    </tbody>
                  </table>
                </div>
              )}
            </div>
          </div>
        )}

        {/* Employee Details Tab */}
        {activeTab === 'employee' && (
          <div className="space-y-6">
            {/* Employee and Date Selector */}
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow p-6 border">
              <h3 className="text-lg font-medium text-gray-900 dark:text-white
mb-4">Employee Activity Details</h3>

              {/* Date Selector */}
              <div className="mb-4">
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-2">Select Date:</label>
                <input
                  type="date"
                  value={employeeDetailDate}
```

```
                  onChange={(e) => setEmployeeDetailDate(e.target.value)}
                  className="border border-gray-300 dark:border-slate-600 rounded-
lg px-3 py-2 text-sm"
                />
              </div>

              {/* Employee Selector */}
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-2">Select Employee:</label>
                <div className="grid grid-cols-2 md:grid-cols-3 lg:grid-cols-4
gap-2">
                  {todayActivity.data.map((employee) => (
                    <button
                      key={employee.userId}
                      onClick={() => handleEmployeeSelect(employee)}
                      className={`px-3 py-2 rounded-lg text-sm font-medium border
${
                        selectedEmployee?.userId === employee.userId
                          ? 'bg-blue-600 text-white border-blue-600'
                          : 'bg-gray-100 dark:bg-slate-700 text-gray-700
dark:text-gray-300 dark:text-gray-400 border-gray-300 dark:border-slate-600
hover:bg-gray-200 dark:bg-slate-600'
                      }`}
                    >
                      {employee.userName}
                    </button>
                  ))}
                </div>
              </div>
            </div>

            {/* Employee Details Display */}
            {selectedEmployee && (
              <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow border">
                <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
                  <h3 className="text-lg font-medium text-gray-900 dark:text-
white">
                    {selectedEmployee.userName} - Activity Details
                  </h3>
                  <p className="text-sm text-gray-600 dark:text-gray-500">
                    Activity for {formatDate(employeeDetailDate)}
                  </p>
                </div>

                {loading ? (
                  <div className="p-8 text-center">
                    <div className="animate-spin rounded-full h-8 w-8 border-b-2
border-blue-600 mx-auto"></div>
                    <p className="text-gray-600 dark:text-gray-500 mt-2">Loading
employee details...</p>
                  </div>
                ) : employeeDetails ? (
                  <div className="p-6">
                    {/* Employee Info Cards */}
                    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-
cols-4 gap-4 mb-6">
```

```
                        <div className="bg-blue-50 rounded-lg p-4 border border-
blue-200">
                          <div className="text-sm font-medium text-blue-600">Total
Active Time</div>
                          <div className="text-2xl font-bold text-blue-900">
                            {formatDuration(employeeDetails.totalActiveTime)}
                          </div>
                          <div className="text-xs text-blue-600">

{formatDetailedDuration(employeeDetails.totalActiveTime)}
                          </div>
                        </div>

                        <div className="bg-green-50 rounded-lg p-4 border border-
green-200">
                          <div className="text-sm font-medium text-
green-600">Sessions</div>
                          <div className="text-2xl font-bold text-green-900">
                            {employeeDetails.sessionsCount}
                          </div>
                          <div className="text-xs text-green-600">
                            Total sessions today
                          </div>
                        </div>

                        <div className="bg-purple-50 rounded-lg p-4 border border-
purple-200">
                          <div className="text-sm font-medium text-
purple-600">Status</div>
                          <div className="text-lg font-bold text-purple-900">
                            {employeeDetails.isCurrentlyActive ? 'Ø=ßâ Active' : '&«
Offline'}
                          </div>
                          <div className="text-xs text-purple-600">
                            Current status
                          </div>
                        </div>

                        <div className="bg-orange-50 rounded-lg p-4 border border-
orange-200">
                          <div className="text-sm font-medium text-orange-600">Last
Activity</div>
                          <div className="text-lg font-bold text-orange-900">
                            {employeeDetails.lastActivity ?
formatTime(employeeDetails.lastActivity) : 'N/A'}
                          </div>
                          <div className="text-xs text-orange-600">
                            Last seen
                          </div>
                        </div>
                      </div>

                      {/* Employee Information */}
                      <div className="bg-gray-50 dark:bg-slate-800 rounded-lg p-4
mb-6">
                        <h4 className="text-md font-semibold text-gray-900
dark:text-white mb-3">Employee Information</h4>
                        <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                          <div>
```

```
                          <span className="text-sm font-medium text-gray-600
dark:text-gray-500">Name:</span>
                          <span className="ml-2 text-sm text-gray-900 dark:text-
white">{employeeDetails.userName}</span>
                        </div>
                        <div>
                          <span className="text-sm font-medium text-gray-600
dark:text-gray-500">Email:</span>
                          <span className="ml-2 text-sm text-gray-900 dark:text-
white">{employeeDetails.userEmail}</span>
                        </div>
                        <div>
                          <span className="text-sm font-medium text-gray-600
dark:text-gray-500">Role:</span>
                          <span className={`ml-2 inline-flex px-2 py-1 text-xs
font-semibold rounded-full border ${getRoleBadge(employeeDetails.userRole)}`}>
                            {employeeDetails.userRole}
                          </span>
                        </div>
                        <div>
                          <span className="text-sm font-medium text-gray-600
dark:text-gray-500">Date:</span>
                          <span className="ml-2 text-sm text-gray-900 dark:text-
white">{formatDate(employeeDetails.date)}</span>
                        </div>
                      </div>
                    </div>

                    {/* Activity Summary */}
                    <div className="bg-gray-50 dark:bg-slate-800 rounded-lg p-4">
                      <h4 className="text-md font-semibold text-gray-900
dark:text-white mb-3">Activity Summary</h4>
                      <div className="space-y-2">
                        <div className="flex justify-between">
                          <span className="text-sm text-gray-600 dark:text-
gray-500">Total Active Time:</span>
                          <span className="text-sm font-medium text-gray-900
dark:text-white">

{formatDetailedDuration(employeeDetails.totalActiveTime)}
                          </span>
                        </div>
                        <div className="flex justify-between">
                          <span className="text-sm text-gray-600 dark:text-
gray-500">Number of Sessions:</span>
                          <span className="text-sm font-medium text-gray-900
dark:text-white">
                            {employeeDetails.sessionsCount}
                          </span>
                        </div>
                        <div className="flex justify-between">
                          <span className="text-sm text-gray-600 dark:text-
gray-500">Average Session Time:</span>
                          <span className="text-sm font-medium text-gray-900
dark:text-white">
                            {employeeDetails.sessionsCount > 0
                              ?
formatDuration(Math.floor(employeeDetails.totalActiveTime /
employeeDetails.sessionsCount))
```

```
                              : '0m'
                            }
                          </span>
                        </div>
                        <div className="flex justify-between">
                          <span className="text-sm text-gray-600 dark:text-
gray-500">Current Status:</span>
                          <span className={`text-sm font-medium
${employeeDetails.isCurrentlyActive ? 'text-green-600' : 'text-gray-600 dark:text-
gray-300'}`}>
                            {employeeDetails.isCurrentlyActive ? 'Currently
Active' : 'Offline'}
                          </span>
                        </div>
                      </div>
                    </div>
                  </div>
                ) : (
                  <div className="p-8 text-center">
                    <svg className="w-12 h-12 text-gray-400 dark:text-gray-400 mx-
auto mb-4" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M20 13V6a2 2 0 00-2-2H6a2 2 0 00-2 2v7m16 0v5a2 2 0 01-2 2H6a2
2 0 01-2-2v-5m16 0h-2.586a1 1 0 00-.707.293l-2.414 2.414a1 1 0
01-.707.293h-3.172a1 1 0 01-.707-.293l-2.414-2.414A1 1 0 006.586 13H4" />
                    </svg>
                    <p className="text-gray-600 dark:text-gray-500">No activity
data found for this employee on {formatDate(employeeDetailDate)}</p>
                  </div>
                )}
              </div>
            )}
          </div>
        )}
      </div>
    </Layout>
  );
};

export default AdminActivityPage;
```

src/pages/AdminDashboardPage.jsx

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import Layout from "../components/Layout/Layout";
import { useAuth } from "../context/AuthContext";
import { leadsAPI, salesAPI, authAPI } from "../services/api";
import { formatCurrency, getDirectSalesCount } from "../utils/helpers";
import axios from "axios";
import { componentClasses, darkModeClasses } from '../utils/darkModeClasses';

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
import { FaHistory } from 'react-icons/fa';
const AdminDashboardPage = () => {
  const { user } = useAuth();
  const [stats, setStats] = useState({
```

```
  totalLeads: 0,
  totalSales: 0,
  totalRevenue: 0,
  totalUsers: 0,
  recentLeads: [],
  recentSales: [],
  userCounts: {
    salesPerson: 0,
    leadPerson: 0,
    manager: 0,
    admin: 0
  },
  leadStages: {
    'Introduction': 0,
    'Acknowledgement': 0,
    'Question': 0,
    'Future Promise': 0,
    'Payment': 0,
    'Analysis': 0
  }
});
const [selectedStage, setSelectedStage] = useState('Acknowledgement');
const [stageLeads, setStageLeads] = useState([]);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);

useEffect(() => {
  fetchDashboardData();
}, []);

useEffect(() => {
  // Update stage leads when selectedStage changes
  if (stats.totalLeads > 0) {
    updateStageLeads();
  }
}, [selectedStage]);

const fetchDashboardData = async () => {
  try {
    setLoading(true);

    // Fetch leads data based on user role
    let leads = [];
    if (user?.role === 'Sales Person') {
      // For Sales Persons, fetch only their assigned leads
      const leadsResponse = await leadsAPI.getAssigned();
      leads = leadsResponse.data.success ? leadsResponse.data.data : [];
      console.log(`Sales Person assigned leads:`, leads.length);
    } else {
      // For other roles, fetch all leads
      const leadsResponse = await leadsAPI.getAll();
      leads = leadsResponse.data.success ? leadsResponse.data.data : [];
      console.log(`All leads fetched:`, leads.length);
    }

    // Initialize variables
    let salesCount = 0;
    let sales = [];
```

```
      // NEW APPROACH: Use direct sales count utility
      try {
        salesCount = await getDirectSalesCount();
      } catch (directCountError) {
        console.error("Error getting direct sales count:", directCountError);
      }

      // Still need to fetch sales data for other info
      try {
        const salesResponse = await salesAPI.getAll();
        if (salesResponse.data && salesResponse.data.success) {
          sales = salesResponse.data.data;
        }
      } catch (salesError) {
        console.error("Error fetching sales data:", salesError);
      }

      // Debug logging for sales data structure
      if (sales.length > 0) {
        console.log("Sales Data Sample:", sales[0]);
        console.log("Sales Data Fields:", Object.keys(sales[0]));
      }

      // Fetch users data
      const usersResponse = await authAPI.getUsers();
      const users = usersResponse.data.success ? usersResponse.data.data : [];

      // Calculate statistics - use totalCost if amount is not available
      const totalRevenue = sales.reduce((sum, sale) =>
        sum + parseFloat(sale.amount || sale.totalCost || 0), 0);

      // Get recent leads (last 5) - sorted by creation date DESCENDING for most
recent first
      const recentLeads = [...leads]
        .sort((a, b) => new Date(b.createdAt) - new Date(a.createdAt))
        .slice(0, 5);

      // Debug logging for leads dates - More detailed debugging
      if (recentLeads.length > 0) {
        console.log("Recent leads date analysis:");
        recentLeads.forEach((lead, index) => {
          const createdDate = new Date(lead.createdAt);
          console.log(`Lead ${index + 1}: ${lead.name}`);
          console.log(`  - Raw createdAt: ${lead.createdAt}`);
          console.log(`  - Parsed Date: ${createdDate}`);
          console.log(`  - Formatted: ${createdDate.toLocaleDateString()}`);
          console.log(`  - Is valid date: ${!isNaN(createdDate.getTime())}`);
          console.log(`  - Lead Person: ${lead.leadPerson?.fullName ||
lead.leadPerson?.name || 'N/A'}`);
          console.log('---');
        });
      }

      // Get recent sales (last 5) - sorted by creation date DESCENDING for most
recent first
      const recentSales = [...sales]
        .sort((a, b) => new Date(b.createdAt) - new Date(a.createdAt))
        .slice(0, 5);
```

```
      // Debug logging for sales data
      if (recentSales.length > 0) {
        console.log("Recent sales sample:", recentSales[0]);
      }

      // Count users by role
      const userCounts = {
        salesPerson: users.filter(u => u.role === "Sales Person").length,
        leadPerson: users.filter(u => u.role === "Lead Person").length,
        manager: users.filter(u => u.role === "Manager").length,
        admin: users.filter(u => u.role === "Admin").length
      };

      // Calculate lead stage statistics
      const leadStages = {
        'Introduction': leads.filter(lead => lead.status ===
'Introduction').length,
        'Acknowledgement': leads.filter(lead => lead.status ===
'Acknowledgement').length,
        'Question': leads.filter(lead => lead.status === 'Question').length,
        'Future Promise': leads.filter(lead => lead.status === 'Future
Promise').length,
        'Payment': leads.filter(lead => lead.status === 'Payment').length,
        'Analysis': leads.filter(lead => lead.status === 'Analysis').length
      };

      // Use our direct count instead of sales.length
      setStats({
        totalLeads: leads.length,
        totalSales: salesCount || sales.length,
        totalRevenue,
        totalUsers: users.length,
        recentLeads,
        recentSales,
        userCounts,
        leadStages
      });

      // Update stage leads based on selected stage
      updateStageLeads(leads, selectedStage);

      setError(null);
    } catch (err) {
      console.error("Error fetching dashboard data:", err);
      setError("Failed to load dashboard data. Please try again.");
    } finally {
      setLoading(false);
    }
  };

  // Update leads for selected stage
  const updateStageLeads = async (allLeads = null, stage = selectedStage) => {
    try {
      let leads = allLeads;
      if (!leads) {
        if (user?.role === 'Sales Person') {
          const leadsResponse = await leadsAPI.getAssigned();
          leads = leadsResponse.data.success ? leadsResponse.data.data : [];
        } else {
```

```
          const leadsResponse = await leadsAPI.getAll();
          leads = leadsResponse.data.success ? leadsResponse.data.data : [];
        }
      }

      const filteredLeads = leads.filter(lead => lead.status === stage);
      setStageLeads(filteredLeads);
    } catch (error) {
      console.error('Error filtering leads by stage:', error);
      setStageLeads([]);
    }
  };

  // Handle stage selection change
  const handleStageChange = (stage) => {
    setSelectedStage(stage);
    updateStageLeads(null, stage);
  };

  // Format date
  const formatDate = (dateString) => {
    const date = new Date(dateString);
    return date.toLocaleDateString();
  };

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-3xl font-bold">
            {user?.role === 'Lead Person' ? 'Lead Person Dashboard' :
             user?.role === 'Sales Person' ? 'Sales Person Dashboard' : 'Admin
Dashboard'}
          </h1>
          <div className="text-sm text-gray-600 dark:text-gray-500">Last updated:
{new Date().toLocaleString()}</div>
        </div>

        {error && (
          <div className="bg-red-50 text-red-600 p-4 rounded-lg mb-6">
            {error}
          </div>
        )}

        {loading ? (
          <div className="flex justify-center items-center h-64">
            <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-
b-2 border-blue-500"></div>
          </div>
        ) : (
          <>
            {/* Stats Overview Cards */}
            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6
mb-8">
              <div className={`${componentClasses.card} rounded-lg p-6`}>
                <div className="flex justify-between items-start">
                  <div>
                    <h3 className={`${darkModeClasses.text.muted} text-sm font-
medium`}>Total Leads</h3>
```

```
                  <p className="text-3xl font-bold mt-1">{stats.totalLeads}</p>
                </div>
                <div className="bg-purple-100 dark:bg-purple-900 p-3 rounded-
full">
                  <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6
text-purple-600 dark:text-purple-400" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M17 20h5v-2a3 3 0 00-5.356-1.857M17 20H7m10
0v-2c0-.656-.126-1.283-.356-1.857M7 20H2v-2a3 3 0 015.356-1.857M7
20v-2c0-.656.126-1.283.356-1.857m0 0a5.002 5.002 0 019.288 0M15 7a3 3 0 11-6 0 3
3 0 016 0zm6 3a2 2 0 11-4 0 2 2 0 014 0zM7 10a2 2 0 11-4 0 2 2 0 014 0z" />
                  </svg>
                </div>
              </div>
              <div className="mt-4">
                <Link to="/admin/leads" className="text-sm text-blue-600
dark:text-blue-400 hover:underline">Manage leads</Link>
              </div>
            </div>

            <div className={`${componentClasses.card} rounded-lg p-6`}>
              <div className="flex justify-between items-start">
                <div>
                  <h3 className={`${darkModeClasses.text.muted} text-sm font-
medium`}>Total Sales</h3>
                  <p className="text-3xl font-bold mt-1">{stats.totalSales}</p>
                </div>
                <div className="bg-green-100 dark:bg-green-900 p-3 rounded-
full">
                  <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6
text-green-600 dark:text-green-400" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 5H7a2 2 0 00-2 2v12a2 2 0 002 2h10a2 2 0 002-2V7a2 2 0
00-2-2h-2M9 5a2 2 0 002 2h2a2 2 0 002-2M9 5a2 2 0 012-2h2a2 2 0 012 2m-6 9l2 2
4-4" />
                  </svg>
                </div>
              </div>
              <div className="mt-4">
                <Link to="/sales-tracking" className="text-sm text-blue-600
dark:text-blue-400 hover:underline">View all sales</Link>
              </div>
            </div>

            <div className={`${componentClasses.card} rounded-lg p-6`}>
              <div className="flex justify-between items-start">
                <div>
                  <h3 className={`${darkModeClasses.text.muted} text-sm font-
medium`}>Revenue</h3>
                  <p className="text-3xl font-bold
mt-1">${stats.totalRevenue?.toLocaleString() || '0'}</p>
                </div>
                <div className="bg-yellow-100 dark:bg-yellow-900 p-3 rounded-
full">
                  <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6
text-yellow-600 dark:text-yellow-400" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
```

```
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 8c-1.657 0-3 .895-3 2s1.343 2 3 2 3 .895 3 2-1.343 2-3
2m0-8c1.11 0 2.08.402 2.599 1M12 8V7m0 1v8m0 0v1m0-1c-1.11 0-2.08-.402-2.599-1M21
12a9 9 0 11-18 0 9 9 0 0118 0z" />
                  </svg>
                </div>
              </div>

              <div className={`${componentClasses.card} rounded-lg p-6`}>
                <div className="flex justify-between items-start">
                  <div>
                    <h3 className={`${darkModeClasses.text.muted} text-sm font-
medium`}>Total Users</h3>
                    <p className="text-3xl font-bold mt-1">{stats.totalUsers}</p>
                  </div>
                  <div className="bg-blue-100 dark:bg-blue-900 p-3 rounded-full">
                    <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6
text-blue-600 dark:text-blue-400" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 4.354a4 4 0 110 5.292M15 21H3v-1a6 6 0 0112 0v1zm0
0h6v-1a6 6 0 00-9-2.239" />
                    </svg>
                  </div>
                </div>
                <div className="mt-4">
                  <Link to="/admin/users" className="text-sm text-blue-600
dark:text-blue-400 hover:underline">Manage users</Link>
                </div>
              </div>
            </div>

            {/* Quick Access Cards */}
            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4
mb-8">
              {/* Existing quick access cards */}

              {/* Activity Logs Card */}
              <Link
                to="/admin/activity-logs"
                className="bg-white dark:bg-gray-800 p-6 rounded-lg shadow-md
hover:shadow-lg transition-shadow duration-200"
              >
                <div className="flex items-center">
                  <div className="p-3 bg-blue-100 dark:bg-blue-900 rounded-full">
                    <FaHistory className="h-6 w-6 text-blue-600 dark:text-
blue-400" />
                  </div>
                  <div className="ml-4">
                    <h3 className="text-lg font-semibold text-gray-900 dark:text-
white">Activity Logs</h3>
                    <p className="text-sm text-gray-500 dark:text-gray-400">Track
all system activities</p>
                  </div>
                </div>
              </Link>
            </div>
```

```
            {/* Activity Dashboard Card - Only show for Admins and Managers */}
            {(user?.role === 'Admin' || user?.role === 'Manager') && (
              <div className={`${componentClasses.card} rounded-lg mb-8`}>
                <div className={`px-6 py-4 ${darkModeClasses.border.primary}
border-b`}>
                  <h2 className={`text-lg font-medium
${darkModeClasses.text.heading}`}>Employee Activity Monitoring</h2>
                </div>
                <div className="p-6">
                  <div className="flex items-center justify-between">
                    <div className="flex items-center space-x-4">
                      <div className="bg-indigo-100 p-3 rounded-full">
                        <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-indigo-600" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                          <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 8v4l3 3m6-3a9 9 0 11-18 0 9 9 0 0118 0z" />
                        </svg>
                      </div>
                      <div>
                        <h3 className={`text-xl font-semibold
${darkModeClasses.text.heading}`}>Activity Dashboard</h3>
                        <p className={`${darkModeClasses.text.secondary} mt-1`}
>Monitor employee CRM usage time and productivity</p>
                        <div className={`mt-2 text-sm
${darkModeClasses.text.muted}`}>
                          • View daily usage time in hours and minutes<br/>
                          • Track individual employee activity<br/>
                          • Generate activity reports and analytics
                        </div>
                      </div>
                    </div>
                    <div className="flex flex-col space-y-3">
                      <Link
                        to="/admin/activity"
                        className="bg-indigo-600 hover:bg-indigo-700 text-white
px-6 py-3 rounded-lg font-medium transition-colors duration-200 text-center"
                      >
                        Ø=ÝP Open Activity Dashboard
                      </Link>
                      <div className="text-xs text-gray-500 dark:text-gray-500
text-center">
                        View employee time tracking
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            )}

            {/* Users Overview - Only show for Admins */}
            {user?.role === 'Admin' && (
              <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700 rounded-lg
shadow mb-8 shadow-sm dark:shadow-black/25">
                <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
                  <h2 className="text-lg font-medium">Team Overview</h2>
                </div>
                <div className="p-6">
```

```
                <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
                  <div className="border rounded-lg p-4 text-center">
                    <h3 className="text-sm text-gray-500 dark:text-
gray-500">Sales Team</h3>
                    <p className="text-2xl font-bold
mt-1">{stats.userCounts.salesPerson}</p>
                  </div>
                  <div className="border rounded-lg p-4 text-center">
                    <h3 className="text-sm text-gray-500 dark:text-
gray-500">Lead Team</h3>
                    <p className="text-2xl font-bold
mt-1">{stats.userCounts.leadPerson}</p>
                  </div>
                  <div className="border rounded-lg p-4 text-center">
                    <h3 className="text-sm text-gray-500 dark:text-
gray-500">Managers</h3>
                    <p className="text-2xl font-bold
mt-1">{stats.userCounts.manager}</p>
                  </div>
                  <div className="border rounded-lg p-4 text-center">
                    <h3 className="text-sm text-gray-500 dark:text-
gray-500">Admins</h3>
                    <p className="text-2xl font-bold
mt-1">{stats.userCounts.admin}</p>
                  </div>
                </div>
              </div>
            </div>
          )}

          {/* Lead Stage Analytics */}
          <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700 rounded-lg
shadow mb-8 shadow-sm dark:shadow-black/25">
            <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
              <h2 className="text-lg font-medium">Lead Stage Analytics</h2>
            </div>
            <div className="p-6">
              {/* Stage Overview Cards */}
              <div className="grid grid-cols-2 md:grid-cols-3 lg:grid-cols-6
gap-4 mb-6">
                {Object.entries(stats.leadStages).map(([stage, count]) => (
                  <div
                    key={stage}
                    className={`border rounded-lg p-4 text-center cursor-
pointer transition-all duration-200 ${
                      selectedStage === stage
                        ? 'border-blue-500 bg-blue-50 shadow-md dark:shadow-xl
dark:shadow-black/25'
                        : 'border-gray-200 dark:border-slate-700 hover:border-
gray-300 dark:border-slate-600 hover:shadow-sm dark:shadow-lg dark:shadow-
black/25'
                    }`}
                    onClick={() => handleStageChange(stage)}
                  >
                    <h3 className={`text-sm font-medium ${
                      selectedStage === stage ? 'text-blue-700' : 'text-
gray-500 dark:text-gray-400 dark:text-gray-400'
```

```
                    }`}>
                      {stage}
                    </h3>
                    <p className={`text-2xl font-bold mt-1 ${
                      selectedStage === stage ? 'text-blue-900' : 'text-
gray-900 dark:text-white'
                    }`}>
                      {count}
                    </p>
                    {selectedStage === stage && (
                      <div className="mt-2">
                        <span className="inline-block w-2 h-2 bg-blue-500
rounded-full"></span>
                      </div>
                    )}
                  </div>
                ))}
              </div>

              {/* Stage Selector and Details */}
              <div className="border-t pt-6">
                <div className="flex flex-col sm:flex-row sm:items-center
sm:justify-between mb-4">
                  <h3 className="text-lg font-medium text-gray-900 dark:text-
white mb-2 sm:mb-0">
                    {user?.role === 'Lead Person' ? `My leads in
"${selectedStage}" Stage` : `Leads in "${selectedStage}" Stage`}
                  </h3>
                  <div className="flex items-center space-x-2">
                    <label htmlFor="stage-select" className="text-sm text-
gray-600 dark:text-gray-500">
                      Select Stage:
                    </label>
                    <select
                      id="stage-select"
                      value={selectedStage}
                      onChange={(e) => handleStageChange(e.target.value)}
                      className="border border-gray-300 dark:border-slate-600
rounded-md px-3 py-2 text-sm focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                    >
                      <option value="Introduction">Introduction</option>
                      <option value="Acknowledgement">Acknowledgement</option>
                      <option value="Question">Question</option>
                      <option value="Future Promise">Future Promise</option>
                      <option value="Payment">Payment</option>
                      <option value="Analysis">Analysis</option>
                    </select>
                  </div>
                </div>

                {/* Stage Details */}
                <div className="bg-gray-50 dark:bg-slate-800 rounded-lg p-4">
                  <div className="flex items-center justify-between mb-3">
                    <span className="text-sm font-medium text-gray-700
dark:text-gray-400">
                      {user?.role === 'Lead Person' ? `My leads in
${selectedStage}:` : `Total leads in ${selectedStage}:`}
                    </span>
```

```
                    <span className="text-lg font-bold text-blue-600">
                      {stats.leadStages[selectedStage]} leads
                    </span>
                  </div>

                  {stageLeads.length > 0 ? (
                    <div className="overflow-x-auto">
                      <table className="min-w-full">
                        <thead>
                          <tr className="border-b border-gray-200 dark:border-
slate-700">
                            <th className="px-4 py-2 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Name</th>
                            <th className="px-4 py-2 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Course</th>
                            <th className="px-4 py-2 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Country</th>
                            <th className="px-4 py-2 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Contact</th>
                            <th className="px-4 py-2 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Date</th>
                          </tr>
                        </thead>
                        <tbody className="divide-y divide-gray-200 dark:divide-
slate-700">
                          {stageLeads.slice(0, 5).map((lead) => (
                            <tr key={lead._id} className="hover:bg-slate-50
dark:hover:bg-slate-800">
                              <td className="px-4 py-2 whitespace-nowrap text-
sm font-medium text-gray-900 dark:text-white">
                                {lead.name}
                              </td>
                              <td className="px-4 py-2 whitespace-nowrap text-
sm text-gray-500 dark:text-gray-500">
                                {lead.course}
                              </td>
                              <td className="px-4 py-2 whitespace-nowrap text-
sm text-gray-500 dark:text-gray-500">
                                {lead.country}
                              </td>
                              <td className="px-4 py-2 whitespace-nowrap text-
sm text-gray-500 dark:text-gray-500">
                                {lead.contactNumber}
                              </td>
                              <td className="px-4 py-2 whitespace-nowrap text-
sm text-gray-500 dark:text-gray-500">
                                {formatDate(lead.createdAt)}
                              </td>
                            </tr>
                          ))}
                        </tbody>
                      </table>
                      {stageLeads.length > 5 && (
                        <div className="mt-3 text-center">
                          <Link
                            to={`/leads?status=${selectedStage}`}
                            className="text-sm text-blue-600 hover:underline"
                          >
                            View all {stageLeads.length} leads in
```

```
{selectedStage} stage
                            </Link>
                          </div>
                        )}
                      </div>
                    ) : (
                      <div className="text-center py-4">
                        <p className="text-gray-500 dark:text-gray-500">No leads
found in {selectedStage} stage</p>
                      </div>
                    )}
                  </div>
                </div>
              </div>

              {/* Recent Activity */}
              <div className="grid grid-cols-1 lg:grid-cols-2 gap-6 mb-8">
                {/* Recent Leads */}
                <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700 rounded-lg
shadow shadow-sm dark:shadow-black/25">
                  <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
                    <h2 className="text-lg font-medium">
                      {user?.role === 'Sales Person' ? 'My Assigned Leads' :
'Recent Leads'}
                    </h2>
                  </div>
                  <div className="p-6">
                    {stats.recentLeads.length === 0 ? (
                      <p className="text-gray-500 dark:text-gray-500 text-center
py-4">No recent leads found</p>
                    ) : (
                      <div className="overflow-x-auto">
                        <table className="min-w-full">
                          <thead>
                            <tr className="border-b border-gray-200 dark:border-
slate-700">
                              <th className="px-4 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Name</th>
                              <th className="px-4 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Course</th>
                              <th className="px-4 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Date</th>
                              {user?.role === 'Sales Person' && (
                                <th className="px-4 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Lead Person</th>
                              )}
                            </tr>
                          </thead>
                          <tbody>
                            {stats.recentLeads.map((lead) => (
                              <tr key={lead._id} className="hover:bg-slate-50
dark:hover:bg-slate-800">
                                <td className="px-4 py-3 whitespace-
nowrap">{lead.name}</td>
                                <td className="px-4 py-3 whitespace-
nowrap">{lead.course}</td>
```

```
                              <td className="px-4 py-3 whitespace-
nowrap">{formatDate(lead.createdAt)}</td>
                              {user?.role === 'Sales Person' && (
                                <td className="px-4 py-3 whitespace-nowrap text-
sm text-gray-500 dark:text-gray-500">
                                  {lead.leadPerson?.fullName ||
lead.leadPerson?.name || 'N/A'}
                                </td>
                              )}
                            </tr>
                          ))}
                        </tbody>
                      </table>
                    </div>
                  )}
                  <div className="mt-4 text-right">
                    <Link to="/leads" className="text-sm text-blue-600
hover:underline">
                      {user?.role === 'Sales Person' ? 'View all my leads' :
'View all leads'}
                    </Link>
                  </div>
                </div>
              </div>

              {/* Recent Sales */}
              <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700 rounded-lg
shadow shadow-sm dark:shadow-black/25">
                <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
                  <h2 className="text-lg font-medium">Recent Sales</h2>
                </div>
                <div className="p-6">
                  {stats.recentSales.length === 0 ? (
                    <p className="text-gray-500 dark:text-gray-500 text-center
py-4">No recent sales found</p>
                  ) : (
                    <div className="overflow-x-auto">
                      <table className="min-w-full">
                        <thead>
                          <tr className="border-b border-gray-200 dark:border-
slate-700">
                            <th className="px-4 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Customer</th>
                            <th className="px-4 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Product</th>
                            <th className="px-4 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">Amount</th>
                          </tr>
                        </thead>
                        <tbody>
                          {stats.recentSales.map((sale) => (
                            <tr key={sale._id} className="hover:bg-slate-50
dark:hover:bg-slate-800">
                              <td className="px-4 py-3 whitespace-nowrap">
                                {sale.leadId?.name || sale.leadId?.NAME ||
sale.customerName || sale.leadName || 'N/A'}
                              </td>
```

```
                            <td className="px-4 py-3 whitespace-nowrap">
                              {sale.product || sale.course || sale.productName
|| 'N/A'}
                            </td>
                            <td className="px-4 py-3 whitespace-nowrap">
                              {formatCurrency(sale.amount || sale.totalCost ||
sale.price || 0)}
                            </td>
                          </tr>
                        ))}
                      </tbody>
                    </table>
                  </div>
                )}
                <div className="mt-4 text-right">
                  <Link to="/sales-tracking" className="text-sm text-blue-600
hover:underline">
                    View all sales
                  </Link>
                </div>
              </div>
            </div>
          </div>

          {/* Quick Actions */}
          <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700 rounded-lg
shadow shadow-sm dark:shadow-black/25">
            <div className="px-6 py-4 border-b border-gray-200 dark:border-
slate-700">
              <h2 className="text-lg font-medium">Quick Actions</h2>
            </div>
            <div className="p-6">
              <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
                {user?.role === 'Admin' ? (
                  <>
                    <Link to="/admin/users" className="bg-blue-50 hover:bg-
blue-100 transition p-4 rounded-lg text-center">
                      <div className="flex flex-col items-center">
                        <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-blue-600 mb-2" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                          <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 4.354a4 4 0 110 5.292M15 21H3v-1a6 6 0 0112 0v1zm0
0h6v-1a6 6 0 00-9-5.197M13 7a4 4 0 11-8 0 4 4 0 018 0z" />
                        </svg>
                        <span className="text-gray-800 dark:text-gray-200 font-
medium">Manage Users</span>
                      </div>
                    </Link>
                    <Link to="/admin/leads" className="bg-green-50 hover:bg-
green-100 transition p-4 rounded-lg text-center">
                      <div className="flex flex-col items-center">
                        <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-green-600 mb-2" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                          <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 12h6m-6 4h6m2 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0
01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z" />
                        </svg>
                        <span className="text-gray-800 dark:text-gray-200 font-
```

```
medium">Manage Leads</span>
                        </div>
                      </Link>
                      <Link to="/admin/reports" className="bg-purple-50 hover:bg-
purple-100 transition p-4 rounded-lg text-center">
                        <div className="flex flex-col items-center">
                          <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-purple-600 mb-2" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0
002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2
2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z" />
                          </svg>
                          <span className="text-gray-800 dark:text-gray-200 font-
medium">View Reports</span>
                        </div>
                      </Link>
                      <Link to="/sales-tracking" className="bg-yellow-50 hover:bg-
yellow-100 transition p-4 rounded-lg text-center">
                        <div className="flex flex-col items-center">
                          <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-yellow-600 mb-2" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 8c-1.657 0-3 .895-3 2s1.343 2 3 2 3 .895 3 2-1.343 2-3
2m0-8c1.11 0 2.08.402 2.599 1M12 8V7m0 1v8m0 0v1m0-1c-1.11 0-2.08-.402-2.599-1M21
12a9 9 0 11-18 0 9 9 0 0118 0z" />
                          </svg>
                          <span className="text-gray-800 dark:text-gray-200 font-
medium">Sales Tracking</span>
                        </div>
                      </Link>
                    </>
                  ) : (
                    <>
                      <Link to="/leads" className="bg-blue-50 hover:bg-blue-100
transition p-4 rounded-lg text-center">
                        <div className="flex flex-col items-center">
                          <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-blue-600 mb-2" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M17 20h5v-2a3 3 0 00-5.356-1.857M17 20H7m10
0v-2c0-.656-.126-1.283-.356-1.857M7 20H2v-2a3 3 0 015.356-1.857M7
20v-2c0-.656.126-1.283.356-1.857m0 0a5.002 5.002 0 019.288 0M15 7a3 3 0 11-6 0 3
3 0 016 0zm6 3a2 2 0 11-4 0 2 2 0 014 0zM7 10a2 2 0 11-4 0 2 2 0 014 0z" />
                          </svg>
                          <span className="text-gray-800 dark:text-gray-200 font-
medium">Manage Leads</span>
                        </div>
                      </Link>
                      <Link to="/admin/import" className="bg-green-50 hover:bg-
green-100 transition p-4 rounded-lg text-center">
                        <div className="flex flex-col items-center">
                          <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-green-600 mb-2" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M7 16a4 4 0 01-.88-7.903A5 5 0 1115.9 6L16 6a5 5 0 011 9.9M15
13l-3-3m0 0l-3 3m3-3v12" />
                          </svg>
                          <span className="text-gray-800 dark:text-gray-200 font-
```

```
medium">Import Data</span>
                        </div>
                      </Link>
                      <Link to="/lead-sales-sheet" className="bg-purple-50
hover:bg-purple-100 transition p-4 rounded-lg text-center">
                        <div className="flex flex-col items-center">
                          <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-purple-600 mb-2" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0
002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2
2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z" />
                          </svg>
                          <span className="text-gray-800 dark:text-gray-200 font-
medium">Sales Sheet</span>
                        </div>
                      </Link>
                      <Link to="/tasks" className="bg-yellow-50 hover:bg-
yellow-100 transition p-4 rounded-lg text-center">
                        <div className="flex flex-col items-center">
                          <svg xmlns="http://www.w3.org/2000/svg" className="h-8
w-8 text-yellow-600 mb-2" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                            <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 5H7a2 2 0 00-2 2v12a2 2 0 002 2h10a2 2 0 002-2V7a2 2 0
00-2-2h-2M9 5a2 2 0 002 2h2a2 2 0 002-2M9 5a2 2 0 012-2h2a2 2 0 012 2m-6 9l2 2
4-4" />
                          </svg>
                          <span className="text-gray-800 dark:text-gray-200 font-
medium">Tasks</span>
                        </div>
                      </Link>
                    </>
                  )}
                </div>
              </div>
            </div>
          </>
        )}
      </div>
    </Layout>
  );
};

export default AdminDashboardPage;
```

src/pages/AdminImportPage.jsx

```
import React, { useState } from "react";
import { toast } from "react-hot-toast";
import axios from "axios";
import Layout from "../components/Layout/Layout";
import { useAuth } from "../context/AuthContext";
import { leadsAPI, salesAPI } from "../services/api";

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const AdminImportPage = () => {
  const { user } = useAuth();
```

```
const [activeTab, setActiveTab] = useState("leads");
const [loading, setLoading] = useState(false);
const [uploadProgress, setUploadProgress] = useState(0);
const [importSummary, setImportSummary] = useState(null);
const [csvFileLeads, setCsvFileLeads] = useState(null);
const [csvFileSales, setCsvFileSales] = useState(null);
const [csvFileGoogleForms, setCsvFileGoogleForms] = useState(null);

// Ensure user is admin
if (user?.role !== "Admin") {
  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="bg-red-50 text-red-600 p-4 rounded-lg mb-6">
          Access denied. Only administrators can access this page.
        </div>
      </div>
    </Layout>
  );
}

const handleFileChangeLeads = (e) => {
  const file = e.target.files[0];
  if (file && file.type === "text/csv") {
    setCsvFileLeads(file);
  } else {
    toast.error("Please upload a valid CSV file");
    e.target.value = null;
  }
};

const handleFileChangeSales = (e) => {
  const file = e.target.files[0];
  if (file && file.type === "text/csv") {
    setCsvFileSales(file);
  } else {
    toast.error("Please upload a valid CSV file");
    e.target.value = null;
  }
};

const handleFileChangeGoogleForms = (e) => {
  const file = e.target.files[0];
  if (file && file.type === "text/csv") {
    setCsvFileGoogleForms(file);
  } else {
    toast.error("Please upload a valid CSV file");
    e.target.value = null;
  }
};

const parseCsvFile = async (file) => {
  return new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.onload = (event) => {
      try {
        const text = event.target.result;
        const lines = text.split("\n");
        const headers = lines[0].split(",").map(h => h.trim());
```

```
        const data = [];
        for (let i = 1; i < lines.length; i++) {
          if (lines[i].trim() === "") continue;

          // Handle quoted fields properly
          const values = [];
          let currentValue = "";
          let insideQuotes = false;

          for (let char of lines[i]) {
            if (char === '"') {
              insideQuotes = !insideQuotes;
            } else if (char === ',' && !insideQuotes) {
              values.push(currentValue);
              currentValue = "";
            } else {
              currentValue += char;
            }
          }
          values.push(currentValue); // Add the last value

          const row = {};
          headers.forEach((header, index) => {
            if (index < values.length) {
              row[header] = values[index].replace(/^"|"$/g, "").trim();
            }
          });

          data.push(row);
        }

        resolve(data);
      } catch (error) {
        reject(error);
      }
    };
    reader.onerror = (error) => reject(error);
    reader.readAsText(file);
  });
};

const importLeads = async () => {
  if (!csvFileLeads) {
    toast.error("Please select a CSV file to import");
    return;
  }

  setLoading(true);
  setUploadProgress(10);
  setImportSummary(null);

  try {
    // Parse the CSV file client-side
    const leads = await parseCsvFile(csvFileLeads);
    setUploadProgress(50);

    if (leads.length === 0) {
      toast.error("No data found in the CSV file");
```

```
      setLoading(false);
      setUploadProgress(0);
      return;
    }

    // Send the parsed data to the server
    const response = await leadsAPI.importLeads(leads);

    setUploadProgress(100);
    setImportSummary({
      total: leads.length,
      imported: response.data.count,
      errors: leads.length - response.data.count
    });

    toast.success(`Successfully imported ${response.data.count} leads`);
  } catch (error) {
    console.error("Import error:", error);
    toast.error(error.response?.data?.message || "Failed to import leads");
  } finally {
    setLoading(false);
  }
};

const importSales = async () => {
  if (!csvFileSales) {
    toast.error("Please select a CSV file to import");
    return;
  }

  setLoading(true);
  setUploadProgress(10);
  setImportSummary(null);

  try {
    // Parse the CSV file client-side
    const sales = await parseCsvFile(csvFileSales);
    setUploadProgress(50);

    if (sales.length === 0) {
      toast.error("No data found in the CSV file");
      setLoading(false);
      setUploadProgress(0);
      return;
    }

    // Send the parsed data to the server
    const response = await salesAPI.importSales(sales);

    setUploadProgress(100);
    setImportSummary({
      total: sales.length,
      imported: response.data.count,
      errors: response.data.errorCount || 0,
      errorDetails: response.data.errors
    });

    toast.success(`Successfully imported ${response.data.count} sales`);
  } catch (error) {
```

```
      console.error("Import error:", error);
      toast.error(error.response?.data?.message || "Failed to import sales");
    } finally {
      setLoading(false);
    }
  };

  const importGoogleForms = async () => {
    if (!csvFileGoogleForms) {
      toast.error("Please select a CSV file to import");
      return;
    }

    setLoading(true);
    setUploadProgress(10);
    setImportSummary(null);

    try {
      // Parse the CSV file client-side
      const leads = await parseCsvFile(csvFileGoogleForms);
      setUploadProgress(50);

      if (leads.length === 0) {
        toast.error("No data found in the CSV file");
        setLoading(false);
        setUploadProgress(0);
        return;
      }

      // Map Google Forms fields to our lead structure
      const mappedLeads = leads.map(lead => ({
        name: lead["Name"] || lead["Full Name"] || lead["Your name"],
        email: lead["Email"] || lead["Email Address"] || lead["Your email"],
        phone: lead["Phone"] || lead["Phone Number"] || lead["Mobile"],
        countryCode: lead["Country Code"] || "+1",
        country: lead["Country"] || "Unknown",
        course: lead["Course"] || lead["Program"] || lead["Interest"] || "General
Inquiry",
        source: "Google Forms",
        sourceLink: lead["Form URL"] || "",
        remarks: lead["Comments"] || lead["Message"] || "",
      }));

      // Send the parsed data to the server
      const response = await leadsAPI.importLeads(mappedLeads);

      setUploadProgress(100);
      setImportSummary({
        total: mappedLeads.length,
        imported: response.data.count,
        errors: mappedLeads.length - response.data.count
      });

      toast.success(`Successfully imported ${response.data.count} leads from
Google Forms`);
    } catch (error) {
      console.error("Import error:", error);
      toast.error(error.response?.data?.message || "Failed to import Google Forms
data");
```

```
    } finally {
      setLoading(false);
    }
  };

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-3xl font-bold">Data Import</h1>
        </div>

        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl overflow-hidden mb-6 shadow-sm">
          <div className="border-b border-slate-200 dark:border-slate-700">
            <nav className="flex -mb-px">
              <button
                onClick={() => setActiveTab("leads")}
                className={`py-4 px-6 text-center border-b-2 font-medium text-sm
${
                  activeTab === "leads"
                    ? "border-blue-500 text-blue-600"
                    : "border-transparent text-gray-500 dark:text-gray-400
dark:text-gray-400 hover:text-gray-700 dark:text-gray-300 dark:text-gray-400
hover:border-gray-300 dark:border-slate-600"
                }`}
              >
                Import Leads
              </button>
              <button
                onClick={() => setActiveTab("sales")}
                className={`py-4 px-6 text-center border-b-2 font-medium text-sm
${
                  activeTab === "sales"
                    ? "border-blue-500 text-blue-600"
                    : "border-transparent text-gray-500 dark:text-gray-400
dark:text-gray-400 hover:text-gray-700 dark:text-gray-300 dark:text-gray-400
hover:border-gray-300 dark:border-slate-600"
                }`}
              >
                Import Sales
              </button>
              <button
                onClick={() => setActiveTab("google-forms")}
                className={`py-4 px-6 text-center border-b-2 font-medium text-sm
${
                  activeTab === "google-forms"
                    ? "border-blue-500 text-blue-600"
                    : "border-transparent text-gray-500 dark:text-gray-400
dark:text-gray-400 hover:text-gray-700 dark:text-gray-300 dark:text-gray-400
hover:border-gray-300 dark:border-slate-600"
                }`}
              >
                Import Google Forms
              </button>
            </nav>
          </div>
```

```
          <div className="p-6">
            {activeTab === "leads" && (
              <div>
                <h2 className="text-lg font-medium mb-4">Import Leads from CSV</
h2>
                <p className="mb-4 text-gray-600 dark:text-gray-500">
                  Upload a CSV file exported from Google Sheets containing lead
information.
                  Make sure your file has headers for: Name, Email, Phone,
Country, Course,
                  and other lead properties. For dates, you can use either YYYY-
MM-DD (e.g., 2025-04-15)
                  or DD-MM-YYYY (e.g., 15-04-2025) format.
                </p>

                <div className="mb-6">
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Upload CSV File
                  </label>
                  <input
                    type="file"
                    accept=".csv"
                    onChange={handleFileChangeLeads}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                    disabled={loading}
                  />
                  {csvFileLeads && (
                    <p className="mt-1 text-sm text-slate-500 dark:text-gray-400">
                      Selected file: {csvFileLeads.name}
                    </p>
                  )}
                </div>

                <button
                  onClick={importLeads}
                  disabled={loading || !csvFileLeads}
                  className={`px-4 py-2 rounded-md text-white ${
                    loading || !csvFileLeads
                      ? "bg-blue-400 cursor-not-allowed"
                      : "bg-blue-600 hover:bg-blue-700"
                  } transition`}
                >
                  {loading ? "Importing..." : "Import Leads"}
                </button>
              </div>
            )}

            {activeTab === "sales" && (
              <div>
                <h2 className="text-lg font-medium mb-4">Import Sales from CSV</
h2>
                <p className="mb-4 text-gray-600 dark:text-gray-500">
                  Upload a CSV file exported from Google Sheets containing sales
information.
                  Make sure your file has headers for: Email or Phone (to
identify the lead),
```

```
                  Amount, Product, Status, and other sales properties.
                </p>

                <div className="mb-6">
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Upload CSV File
                  </label>
                  <input
                    type="file"
                    accept=".csv"
                    onChange={handleFileChangeSales}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                    disabled={loading}
                  />
                  {csvFileSales && (
                    <p className="mt-1 text-sm text-slate-500 dark:text-gray-400">
                      Selected file: {csvFileSales.name}
                    </p>
                  )}
                </div>

                <button
                  onClick={importSales}
                  disabled={loading || !csvFileSales}
                  className={`px-4 py-2 rounded-md text-white ${
                    loading || !csvFileSales
                      ? "bg-blue-400 cursor-not-allowed"
                      : "bg-blue-600 hover:bg-blue-700"
                  } transition`}
                >
                  {loading ? "Importing..." : "Import Sales"}
                </button>
              </div>
            )}

            {activeTab === "google-forms" && (
              <div>
                <h2 className="text-lg font-medium mb-4">Import Google Forms
Responses</h2>
                <p className="mb-4 text-gray-600 dark:text-gray-500">
                  Upload a CSV file exported from Google Forms containing lead
information.
                  The system will try to map common Google Forms fields to the
appropriate lead properties.
                  For dates, you can use either YYYY-MM-DD (e.g., 2025-04-15) or
DD-MM-YYYY (e.g., 15-04-2025) format.
                </p>

                <div className="mb-6">
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Upload CSV File
                  </label>
                  <input
                    type="file"
                    accept=".csv"
```

```
                    onChange={handleFileChangeGoogleForms}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                    disabled={loading}
                  />
                  {csvFileGoogleForms && (
                    <p className="mt-1 text-sm text-slate-500 dark:text-gray-400">
                      Selected file: {csvFileGoogleForms.name}
                    </p>
                  )}
                </div>

                <button
                  onClick={importGoogleForms}
                  disabled={loading || !csvFileGoogleForms}
                  className={`px-4 py-2 rounded-md text-white ${
                    loading || !csvFileGoogleForms
                      ? "bg-blue-400 cursor-not-allowed"
                      : "bg-blue-600 hover:bg-blue-700"
                  } transition`}
                >
                  {loading ? "Importing..." : "Import Google Forms Data"}
                </button>
              </div>
            )}

            {loading && (
              <div className="mt-6">
                <div className="relative pt-1">
                  <div className="flex mb-2 items-center justify-between">
                    <div>
                      <span className="text-xs font-semibold inline-block py-1
px-2 uppercase rounded-full text-blue-600 bg-blue-200">
                        Progress
                      </span>
                    </div>
                    <div className="text-right">
                      <span className="text-xs font-semibold inline-block text-
blue-600">
                        {uploadProgress}%
                      </span>
                    </div>
                  </div>
                  <div className="overflow-hidden h-2 mb-4 text-xs flex rounded
bg-blue-200">
                    <div
                      style={{ width: `${uploadProgress}%` }}
                      className="shadow-none flex flex-col text-center whitespace-
nowrap text-white justify-center bg-blue-500 transition-all duration-500"
                    ></div>
                  </div>
                </div>
              </div>
            )}

            {importSummary && (
              <div className="mt-6 p-4 bg-gray-50 dark:bg-slate-800 transition-
all duration-200 ease-out rounded-lg">
```

```
                <h3 className="text-lg font-medium mb-2">Import Summary</h3>
                <div className="grid grid-cols-3 gap-4 mb-4">
                  <div className="bg-blue-50 p-3 rounded-md">
                    <p className="text-sm text-gray-600 dark:text-gray-500">Total
Records</p>
                    <p className="text-xl font-bold">{importSummary.total}</p>
                  </div>
                  <div className="bg-green-50 p-3 rounded-md">
                    <p className="text-sm text-gray-600 dark:text-
gray-500">Successfully Imported</p>
                    <p className="text-xl font-bold text-
green-600">{importSummary.imported}</p>
                  </div>
                  <div className="bg-red-50 p-3 rounded-md">
                    <p className="text-sm text-gray-600 dark:text-
gray-500">Failed Records</p>
                    <p className="text-xl font-bold text-
red-600">{importSummary.errors}</p>
                  </div>
                </div>

                {importSummary.errorDetails && importSummary.errorDetails.length
> 0 && (
                  <div className="mt-4">
                    <h4 className="text-md font-medium mb-2">Error Details</h4>
                    <div className="max-h-40 overflow-y-auto bg-white dark:bg-
slate-900 transition-all duration-200 ease-out p-2 rounded border border-
slate-300 dark:border-slate-600">
                      <ul className="text-sm text-slate-700 dark:text-slate-300">
                        {importSummary.errorDetails.slice(0, 10).map((error,
index) => (
                          <li key={index} className="mb-1 pb-1 border-b border-
slate-200 dark:border-slate-700">
                            {error.message}
                          </li>
                        ))}
                        {importSummary.errorDetails.length > 10 && (
                          <li className="text-slate-500 dark:text-gray-400
italic">
                            ...and {importSummary.errorDetails.length - 10} more
errors
                          </li>
                        )}
                      </ul>
                    </div>
                  </div>
                )}
              </div>
            )}
          </div>
        </div>

        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl overflow-hidden p-6 shadow-sm">
          <h2 className="text-lg font-medium mb-4">Import Instructions</h2>

          <div className="mb-4">
            <h3 className="text-md font-semibold mb-2">Required Fields</h3>
```

```
            <div className="mb-4">
              <h4 className="font-medium text-blue-600">Leads Sheet:</h4>
              <ul className="list-disc ml-5 text-sm text-gray-600 dark:text-
gray-500">
                <li><strong>Name</strong> - Lead's full name</li>
                <li><strong>Email</strong> - Lead's email address</li>
                <li><strong>Phone</strong> - Lead's phone number</li>
                <li><strong>CountryCode</strong> - Phone country code (e.g., +1,
+91)</li>
                <li><strong>Country</strong> - Lead's country</li>
                <li><strong>Course</strong> - Course/product of interest</li>
                <li>Optional: PseudoId, Company, Client, Status, Source,
SourceLink, Remarks, Feedback</li>
              </ul>
            </div>

            <div className="mb-4">
              <h4 className="font-medium text-blue-600">Sales Sheet:</h4>
              <ul className="list-disc ml-5 text-sm text-gray-600 dark:text-
gray-500">
                <li><strong>Email</strong> or <strong>Phone</strong> - To
identify the associated lead</li>
                <li><strong>Amount</strong> - Sale amount</li>
                <li><strong>Product</strong> - Product/service sold</li>
                <li>Optional: Token, Status (Pending/Closed), Notes</li>
              </ul>
            </div>

            <div>
              <h4 className="font-medium text-blue-600">Google Forms:</h4>
              <ul className="list-disc ml-5 text-sm text-gray-600 dark:text-
gray-500">
                <li>The system automatically maps common Google Forms field
names</li>
                <li>Ensure your form includes name, email, phone, and course
interest fields</li>
              </ul>
            </div>
          </div>

          <div>
            <h3 className="text-md font-semibold mb-2">Export Instructions</h3>
            <ol className="list-decimal ml-5 text-sm text-gray-600 dark:text-
gray-500">
              <li>Open your Google Sheet</li>
              <li>Go to <strong>File</strong> &gt; <strong>Download</strong> &gt;
<strong>Comma-separated values (.csv)</strong></li>
              <li>Save the file to your computer</li>
              <li>Upload the CSV file using the form above</li>
            </ol>
          </div>
        </div>
      </div>
    </Layout>
  );
};

export default AdminImportPage;
```

src/pages/AdminLeadsPage.jsx

```
import React, { useState, useEffect } from "react";
import Layout from "../components/Layout/Layout";
import { leadsAPI, authAPI } from "../services/api";
import { useAuth } from "../context/AuthContext";
import { Link } from "react-router-dom";

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const AdminLeadsPage = () => {
  const { user } = useAuth();
  const [leads, setLeads] = useState([]);
  const [filteredLeads, setFilteredLeads] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  // Filter state
  const [filters, setFilters] = useState({
    search: "",
    status: "",
    country: "",
    course: "",
    source: "",
    dateFrom: "",
    dateTo: "",
    assignedTo: "",
    leadPerson: ""
  });

  // Options for filters
  const [filterOptions, setFilterOptions] = useState({
    countries: [],
    courses: [],
    sources: [],
    salesPersons: [],
    leadPersons: []
  });

  // Lead statuses
  const statusOptions = [
    'New',
    'Contacted',
    'Qualified',
    'Lost',
    'Converted',
    'Introduction',
    'Acknowledgement',
    'Question',
    'Future Promise',
    'Payment',
    'Analysis'
  ];

  useEffect(() => {
    fetchLeads();
    fetchUserOptions();
  }, []);
```

```
  useEffect(() => {
    if (leads.length > 0) {
      applyFilters();
      extractFilterOptions();
    }
  }, [leads, filters]);

  const fetchLeads = async () => {
    try {
      setLoading(true);
      const response = await leadsAPI.getAll();

      if (response.data.success) {
        setLeads(response.data.data);
        setFilteredLeads(response.data.data);
        setError(null);
      } else {
        setError("Failed to load leads");
      }
    } catch (err) {
      console.error("Error fetching leads:", err);
      setError("Failed to load leads. Please try again.");
    } finally {
      setLoading(false);
    }
  };

  const fetchUserOptions = async () => {
    try {
      const salesPersonsResponse = await authAPI.getUsers("Sales Person");
      const leadPersonsResponse = await authAPI.getUsers("Lead Person");

      setFilterOptions(prev => ({
        ...prev,
        salesPersons: salesPersonsResponse.data.data || [],
        leadPersons: leadPersonsResponse.data.data || []
      }));
    } catch (err) {
      console.error("Error fetching user options:", err);
    }
  };

  const extractFilterOptions = () => {
    // Extract unique values for filter options
    const countries = [...new Set(leads.map(lead =>
lead.country).filter(Boolean))];
    const courses = [...new Set(leads.map(lead => lead.course).filter(Boolean))];
    const sources = [...new Set(leads.map(lead => lead.source).filter(Boolean))];

    setFilterOptions(prev => ({
      ...prev,
      countries,
      courses,
      sources
    }));
  };

  const handleFilterChange = (e) => {
```

```
  const { name, value } = e.target;
  setFilters(prev => ({ ...prev, [name]: value }));
};

const resetFilters = () => {
  setFilters({
    search: "",
    status: "",
    country: "",
    course: "",
    source: "",
    dateFrom: "",
    dateTo: "",
    assignedTo: "",
    leadPerson: ""
  });
};

const applyFilters = () => {
  let filtered = [...leads];

  // Text search (name, email, phone, pseudoId)
  if (filters.search) {
    const searchTerm = filters.search.toLowerCase();
    filtered = filtered.filter(lead =>
      (lead.name && lead.name.toLowerCase().includes(searchTerm)) ||
      (lead.email && lead.email.toLowerCase().includes(searchTerm)) ||
      (lead.phone && lead.phone.includes(searchTerm)) ||
      (lead.pseudoId && lead.pseudoId.toLowerCase().includes(searchTerm))
    );
  }

  // Status filter
  if (filters.status) {
    filtered = filtered.filter(lead => lead.status === filters.status);
  }

  // Country filter
  if (filters.country) {
    filtered = filtered.filter(lead => lead.country === filters.country);
  }

  // Course filter
  if (filters.course) {
    filtered = filtered.filter(lead => lead.course === filters.course);
  }

  // Source filter
  if (filters.source) {
    filtered = filtered.filter(lead => lead.source === filters.source);
  }

  // Date range filter
  if (filters.dateFrom) {
    const fromDate = new Date(filters.dateFrom);
    filtered = filtered.filter(lead => new Date(lead.createdAt) >= fromDate);
  }

  if (filters.dateTo) {
```

```
      const toDate = new Date(filters.dateTo);
      toDate.setHours(23, 59, 59, 999); // End of the day
      filtered = filtered.filter(lead => new Date(lead.createdAt) <= toDate);
    }

    // Assigned To filter
    if (filters.assignedTo) {
      filtered = filtered.filter(lead =>
        lead.assignedTo && lead.assignedTo._id === filters.assignedTo
      );
    }

    // Lead Person filter
    if (filters.leadPerson) {
      filtered = filtered.filter(lead =>
        lead.leadPerson && lead.leadPerson._id === filters.leadPerson
      );
    }

    setFilteredLeads(filtered);
  };

  // Format date for display
  const formatDate = (dateString) => {
    const date = new Date(dateString);
    return date.toLocaleDateString();
  };

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-3xl font-bold">Admin Leads Management</h1>
          <Link
            to="/leads"
            className="px-4 py-2 bg-blue-600 hover:bg-blue-700 dark:bg-blue-500
dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md transition-all
duration-200 text-white rounded-md transition"
          >
            Standard Leads View
          </Link>
        </div>

        {error && (
          <div className="bg-red-50 text-red-600 p-4 rounded-lg mb-6">
            {error}
          </div>
        )}

        {/* Advanced Filters */}
        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 mb-6 shadow-sm">
          <div className="flex justify-between items-center mb-4">
            <h2 className="text-lg font-medium">Advanced Filters</h2>
            <button
              onClick={resetFilters}
              className="text-sm text-blue-600 hover:underline"
            >
```

```
              Reset Filters
            </button>
          </div>

          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
            {/* Search Field */}
            <div>
              <label htmlFor="search" className="block text-sm font-medium text-
slate-700 dark:text-slate-300 mb-1">
                Search
              </label>
              <input
                type="text"
                id="search"
                name="search"
                placeholder="Search name, email, phone..."
                value={filters.search}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              />
            </div>

            {/* Status Filter */}
            <div>
              <label htmlFor="status" className="block text-sm font-medium text-
slate-700 dark:text-slate-300 mb-1">
                Status
              </label>
              <select
                id="status"
                name="status"
                value={filters.status}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              >
                <option value="">All Statuses</option>
                {statusOptions.map(status => (
                  <option key={status} value={status}>{status}</option>
                ))}
              </select>
            </div>

            {/* Country Filter */}
            <div>
              <label htmlFor="country" className="block text-sm font-medium text-
slate-700 dark:text-slate-300 mb-1">
                Country
              </label>
              <select
                id="country"
                name="country"
                value={filters.country}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
```

```
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              >
                <option value="">All Countries</option>
                {filterOptions.countries.map(country => (
                  <option key={country} value={country}>{country}</option>
                ))}
              </select>
            </div>

            {/* Course Filter */}
            <div>
              <label htmlFor="course" className="block text-sm font-medium text-
slate-700 dark:text-slate-300 mb-1">
                Course
              </label>
              <select
                id="course"
                name="course"
                value={filters.course}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              >
                <option value="">All Courses</option>
                {filterOptions.courses.map(course => (
                  <option key={course} value={course}>{course}</option>
                ))}
              </select>
            </div>

            {/* Source Filter */}
            <div>
              <label htmlFor="source" className="block text-sm font-medium text-
slate-700 dark:text-slate-300 mb-1">
                Source
              </label>
              <select
                id="source"
                name="source"
                value={filters.source}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              >
                <option value="">All Sources</option>
                {filterOptions.sources.map(source => (
                  <option key={source} value={source}>{source}</option>
                ))}
              </select>
            </div>

            {/* Date Range - From */}
            <div>
              <label htmlFor="dateFrom" className="block text-sm font-medium text-
slate-700 dark:text-slate-300 mb-1">
                Date From
              </label>
```

```
              <input
                type="date"
                id="dateFrom"
                name="dateFrom"
                value={filters.dateFrom}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              />
            </div>

            {/* Date Range - To */}
            <div>
              <label htmlFor="dateTo" className="block text-sm font-medium text-
slate-700 dark:text-slate-300 mb-1">
                Date To
              </label>
              <input
                type="date"
                id="dateTo"
                name="dateTo"
                value={filters.dateTo}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              />
            </div>

            {/* Assigned To Filter */}
            <div>
              <label htmlFor="assignedTo" className="block text-sm font-medium
text-slate-700 dark:text-slate-300 mb-1">
                Assigned To
              </label>
              <select
                id="assignedTo"
                name="assignedTo"
                value={filters.assignedTo}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              >
                <option value="">All Sales Persons</option>
                {filterOptions.salesPersons.map(salesPerson => (
                  <option key={salesPerson._id} value={salesPerson._id}>
                    {salesPerson.fullName}
                  </option>
                ))}
              </select>
            </div>

            {/* Lead Person Filter */}
            <div>
              <label htmlFor="leadPerson" className="block text-sm font-medium
text-slate-700 dark:text-slate-300 mb-1">
                Lead Person
```

```
              </label>
              <select
                id="leadPerson"
                name="leadPerson"
                value={filters.leadPerson}
                onChange={handleFilterChange}
                className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
              >
                <option value="">All Lead Persons</option>
                {filterOptions.leadPersons.map(leadPerson => (
                  <option key={leadPerson._id} value={leadPerson._id}>
                    {leadPerson.fullName}
                  </option>
                ))}
              </select>
            </div>
          </div>
        </div>

        {/* Leads Table */}
        {loading ? (
          <div className="flex justify-center items-center h-64">
            <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-
b-2 border-blue-500"></div>
          </div>
        ) : (
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl overflow-hidden shadow-sm">
            <div className="p-4 bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out border-b border-slate-200 dark:border-slate-700 flex
justify-between items-center">
              <h3 className="text-lg font-medium">{filteredLeads.length} Leads
Found</h3>
              <div className="text-sm text-slate-500 dark:text-gray-400">
                Showing filtered results from a total of {leads.length} leads
              </div>
            </div>

            {filteredLeads.length === 0 ? (
              <div className="p-6 text-center text-slate-500 dark:text-gray-400">
                No leads found matching your filters. Try adjusting your criteria.
              </div>
            ) : (
              <div className="overflow-x-auto">
                <table className="min-w-full divide-y divide-slate-200
dark:divide-slate-700">
                  <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
                    <tr>
                      <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                        #
                      </th>
                      <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                        Name
```

```
                    </th>
                    <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      Contact
                    </th>
                    <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      Course
                    </th>
                    <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      Status
                    </th>
                    <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      Country
                    </th>
                    <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      Assigned To
                    </th>
                    <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      Lead By
                    </th>
                    <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      Date
                    </th>
                    <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      Actions
                    </th>
                  </tr>
                </thead>
                <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-slate-200 dark:divide-slate-700">
                  {filteredLeads.map((lead, index) => (
                    <tr key={lead._id} className="hover:bg-slate-50
dark:hover:bg-slate-800 dark:bg-slate-800 transition-all duration-200 ease-out">
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                        {index + 1}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap">
                        <div className="text-sm font-medium text-slate-900
dark:text-slate-100">
                          {lead.name}
                          {lead.isRepeatCustomer && (
                            <span
                              className="ml-2 inline-flex items-center px-2.5
py-0.5 rounded-full text-xs font-medium bg-purple-100 text-purple-800"
                              title={`Repeat customer! Previous courses:
${lead.previousCourses?.join(', ') || 'None'}`}
                            >
                              Repeat
                            </span>
                          )}
                        </div>
```

```
                          <div className="text-sm text-slate-500 dark:text-
gray-400">{lead.pseudoId}</div>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap">
                          <div className="text-sm text-slate-900 dark:text-
slate-100">{lead.email}</div>
                          <div className="text-sm text-slate-500 dark:text-
gray-400">
                            {lead.countryCode} {lead.phone}
                          </div>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap">
                          <div className="text-sm text-slate-900 dark:text-
slate-100">{lead.course}</div>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap">
                          <span className={`px-2 py-1 inline-flex text-xs
leading-5 font-semibold rounded-full
                            ${lead.status === 'Introduction' ? 'bg-blue-100 text-
blue-800' :
                            lead.status === 'Acknowledgement' ? 'bg-yellow-100
text-yellow-800' :
                            lead.status === 'Question' ? 'bg-purple-100 text-
purple-800' :
                            lead.status === 'Future Promise' ? 'bg-red-100 text-
red-800' :
                            lead.status === 'Payment' ? 'bg-green-100 text-
green-800' :
                            lead.status === 'Analysis' ? 'bg-gray-100 dark:bg-
slate-700 text-gray-800 dark:text-gray-200' :
                            'bg-gray-100 dark:bg-slate-700 text-gray-800
dark:text-gray-200'}
                          `}>
                            {lead.status}
                          </span>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          {lead.country}
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          {lead.assignedTo ? lead.assignedTo.fullName : 'N/A'}
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          {lead.leadPerson ? lead.leadPerson.fullName : 'N/A'}
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          {formatDate(lead.createdAt)}
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          <Link
                            to={`/leads#${lead._id}`}
                            className="text-blue-600 hover:text-blue-900 mr-3"
                          >
                            View
```

```
                        </Link>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          )}
        </div>
      )}
    </div>
  </Layout>
 );
};

export default AdminLeadsPage;
```

src/pages/AdminReportsPage.jsx

```
import React, { useState, useEffect, useMemo } from 'react';
import { Link } from 'react-router-dom';
import Layout from '../components/Layout/Layout';
import { salesAPI, leadsAPI, authAPI, currencyAPI } from '../services/api';
import { FaFilter, FaCalendar, FaChartBar, FaChartPie, FaChartLine, FaFileExport,
FaTable, FaSortDown, FaSortUp, FaSort, FaDollarSign, FaGraduationCap, FaDownload,
FaCalendarAlt } from 'react-icons/fa';
import { formatCurrency, convertCurrency, getCurrencySettings,
setCurrencySettings, BASE_CURRENCY } from '../utils/helpers';
import * as XLSX from 'xlsx';
import CurrencySelector from '../components/CurrencySelector';
import axios from 'axios';
import { toast } from 'react-toastify';

// Chart.js components
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  BarElement,
  ArcElement,
  Title,
  Tooltip,
  Legend,
} from 'chart.js';
import { Line, Bar, Pie } from 'react-chartjs-2';

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
// Register Chart.js components
ChartJS.register(
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  BarElement,
  ArcElement,
```

```
  Title,
  Tooltip,
  Legend
);

const AdminReportsPage = () => {
  // State for loading status and errors
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  // State for sales and lead data
  const [sales, setSales] = useState([]);
  const [leads, setLeads] = useState([]);
  const [filteredSales, setFilteredSales] = useState([]);

  // State for exchange rates
  const [exchangeRates, setExchangeRates] = useState({});
  const [selectedCurrency, setSelectedCurrency] = useState('USD');

  // State for time-based filters
  const [timeFrame, setTimeFrame] = useState('monthly'); // 'monthly',
'quarterly', 'yearly'
  const [currentYear, setCurrentYear] = useState(new Date().getFullYear());
  const [currentMonth, setCurrentMonth] = useState(new Date().getMonth() + 1);
  const [currentQuarter, setCurrentQuarter] = useState(Math.ceil((new
Date().getMonth() + 1) / 3));

  // State for active tab
  const [activeTab, setActiveTab] = useState('overview'); // 'overview', 'sales',
'courses', 'detailed'

  // State for table sorting
  const [sortField, setSortField] = useState('date');
  const [sortDirection, setSortDirection] = useState('desc');

  // Years for year selection
  const years = useMemo(() => {
    const currentYear = new Date().getFullYear();
    return Array.from({ length: 5 }, (_, i) => currentYear - i);
  }, []);

  // Months for month selection
  const months = useMemo(() => [
    { value: 1, label: 'January' },
    { value: 2, label: 'February' },
    { value: 3, label: 'March' },
    { value: 4, label: 'April' },
    { value: 5, label: 'May' },
    { value: 6, label: 'June' },
    { value: 7, label: 'July' },
    { value: 8, label: 'August' },
    { value: 9, label: 'September' },
    { value: 10, label: 'October' },
    { value: 11, label: 'November' },
    { value: 12, label: 'December' }
  ], []);

  // Quarters for quarter selection
  const quarters = useMemo(() => [
```

```
    { value: 1, label: 'Q1 (Jan-Mar)' },
    { value: 2, label: 'Q2 (Apr-Jun)' },
    { value: 3, label: 'Q3 (Jul-Sep)' },
    { value: 4, label: 'Q4 (Oct-Dec)' }
  ], []);

  // Course analysis filter options
  const courseFilterOptions = [
    { value: 'monthly', label: 'Month wise (Last 12 months)' },
    { value: 'quarterly', label: '3 Months (Quarterly)' },
    { value: 'half-yearly', label: 'Half Yearly (6 months)' },
    { value: 'yearly', label: 'Yearly (Last 3 years)' }
  ];

  // Revenue analysis filter options
  const revenueFilterOptions = [
    { value: '1month', label: '1 Month' },
    { value: '3month', label: '3 Months' },
    { value: '6month', label: '6 Months' },
    { value: '1year', label: '1 Year' }
  ];

  // Listen for currency setting changes
  useEffect(() => {
    const settings = getCurrencySettings();
    setSelectedCurrency(settings.currency);
    setExchangeRates(settings.exchangeRates);

    // Define a handler function for storage events
    const handleStorageChange = () => {
      const newSettings = getCurrencySettings();
      setSelectedCurrency(newSettings.currency);
      setExchangeRates(newSettings.exchangeRates);
    };

    // Add event listener for changes in localStorage
    window.addEventListener('storage', handleStorageChange);

    // Cleanup
    return () => {
      window.removeEventListener('storage', handleStorageChange);
    };
  }, []);

  // Fetch all data on component mount
  useEffect(() => {
    fetchAllData();
  }, []);

  // Update filtered sales when timeframe or filters change
  useEffect(() => {
    if (sales.length > 0) {
      filterSalesByTimeFrame();
    }
  }, [sales, timeFrame, currentYear, currentMonth, currentQuarter,
selectedCurrency]);

  // Function to fetch all required data
  const fetchAllData = async () => {
```

```
    try {
      setLoading(true);
      setError(null);

      // Fetch sales data with robust error handling
      let salesData = [];
      try {
        // Use direct axios for more reliable data fetching with full=true to get
all sales
        const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
        const apiUrl = isDevelopment ? 'http://localhost:8080' : 'https://crm-
backend-o36v.onrender.com/api';
        const token = localStorage.getItem('token');
        const salesResponse = await axios.get(`${apiUrl}${isDevelopment ? '/
api' : ''}/sales?full=true&nocache=${new Date().getTime()}`, {
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json'
          }
        });

        // Log the full response for debugging
        console.log("AdminReportsPage - Full sales response:", salesResponse);

        // Check if we have data in the expected format
        if (salesResponse.data && salesResponse.data.success &&
salesResponse.data.data && Array.isArray(salesResponse.data.data)) {
          salesData = salesResponse.data.data;
          console.log("AdminReportsPage - Sales count from direct axios call:",
salesData.length);
        } else if (salesResponse.data && Array.isArray(salesResponse.data)) {
          // Handle case where data might be directly in the response
          salesData = salesResponse.data;
          console.log("AdminReportsPage - Sales count from direct array
response:", salesData.length);
        } else {
          console.warn("AdminReportsPage - Unexpected sales response format:",
salesResponse.data);
          // Fallback to empty array
          salesData = [];
        }
      } catch (salesError) {
        console.error('AdminReportsPage - Error fetching sales with direct
axios:', salesError);

        // Fallback to the API service as a second attempt
        try {
          const fallbackResponse = await salesAPI.getAllForced();
          if (fallbackResponse.data && fallbackResponse.data.success) {
            salesData = fallbackResponse.data.data;
            console.log("AdminReportsPage - Sales count from fallback API:",
salesData.length);
          }
        } catch (fallbackError) {
          console.error("AdminReportsPage - Fallback sales fetch also failed:",
fallbackError);
          salesData = []; // Ensure we have an empty array as fallback
        }
```

```
      }

      // Fetch leads data
      const leadsResponse = await leadsAPI.getAll();
      const leadsData = leadsResponse.data.success ? leadsResponse.data.data : [];

      // Fetch exchange rates
      const ratesResponse = await currencyAPI.getRates();
      let ratesData = {};
      if (ratesResponse.data.success) {
        ratesData = ratesResponse.data.data;
        // Update currency settings with latest rates
        setCurrencySettings({ exchangeRates: ratesData });
      } else {
        // Use default rates if API fails
        ratesData = getCurrencySettings().exchangeRates;
      }

      // Process the sales data to ensure consistency
      const processedSales = salesData.map(sale => {
        return {
          ...sale,
          // Ensure we have standard field names
          id: sale._id,
          date: new Date(sale.date || sale.createdAt),
          customerName: sale.customerName || (sale.leadId && (sale.leadId.name ||
sale.leadId.NAME)) || 'Unknown',
          course: sale.course || sale.product || 'Unknown',
          // Ensure financial fields are consistent and numbers
          totalCost: parseFloat(sale.totalCost || sale.amount || 0),
          tokenAmount: parseFloat(sale.tokenAmount || sale.token || 0),
          pendingAmount: parseFloat(sale.status === 'Completed' ? 0 :
            (sale.pendingAmount || (sale.totalCost || sale.amount || 0) -
(sale.tokenAmount || sale.token || 0))),
          // Ensure we have the currency
          currency: sale.currency || 'USD'
        };
      });

      // Set state with the data
      setSales(processedSales);
      setLeads(leadsData);
      setExchangeRates(ratesData);

      // Initial filtering
      filterSalesByTimeFrame(processedSales);

      setError(null);
    } catch (err) {
      console.error('Error fetching report data:', err);
      setError('Failed to load report data. Please try again.');
    } finally {
      setLoading(false);
    }
  };

  // Function to filter sales by the selected time frame
  const filterSalesByTimeFrame = (salesData = sales) => {
    // Use passed salesData or fall back to state
```

```
  const data = salesData || [];

  let filteredData = [];

  switch (timeFrame) {
    case 'monthly':
      // Filter by selected month and year
      filteredData = data.filter(sale => {
        const saleDate = new Date(sale.date);
        return saleDate.getMonth() + 1 === currentMonth &&
               saleDate.getFullYear() === currentYear;
      });
      break;

    case 'quarterly':
      // Filter by selected quarter and year
      filteredData = data.filter(sale => {
        const saleDate = new Date(sale.date);
        const saleMonth = saleDate.getMonth() + 1;
        const saleQuarter = Math.ceil(saleMonth / 3);
        return saleQuarter === currentQuarter &&
               saleDate.getFullYear() === currentYear;
      });
      break;

    case 'yearly':
      // Filter by selected year
      filteredData = data.filter(sale => {
        const saleDate = new Date(sale.date);
        return saleDate.getFullYear() === currentYear;
      });
      break;

    default:
      filteredData = data;
  }

  // Apply sorting
  filteredData = sortSalesData(filteredData, sortField, sortDirection);

  // Update state
  setFilteredSales(filteredData);
};

// Function to sort sales data
const sortSalesData = (data, field, direction) => {
  return [...data].sort((a, b) => {
    let valueA, valueB;

    // Handle different field types
    switch (field) {
      case 'date':
        valueA = new Date(a.date).getTime();
        valueB = new Date(b.date).getTime();
        break;
      case 'totalCost':
      case 'tokenAmount':
      case 'pendingAmount':
        // Convert all to a common currency for sorting
```

```
          valueA = convertCurrency(a[field], 'USD');
          valueB = convertCurrency(b[field], 'USD');
          break;
        default:
          valueA = a[field] || '';
          valueB = b[field] || '';
      }

      // String comparison for non-numeric fields
      if (typeof valueA === 'string' && typeof valueB === 'string') {
        return direction === 'asc'
          ? valueA.localeCompare(valueB)
          : valueB.localeCompare(valueA);
      }

      // Numeric comparison
      return direction === 'asc' ? valueA - valueB : valueB - valueA;
    });
  };

  // Handle sort column change
  const handleSort = (field) => {
    const newDirection =
      field === sortField && sortDirection === 'asc' ? 'desc' : 'asc';

    setSortField(field);
    setSortDirection(newDirection);

    // Re-sort the data
    const sorted = sortSalesData(filteredSales, field, newDirection);
    setFilteredSales(sorted);
  };

  // Get sort icon for a column
  const getSortIcon = (field) => {
    if (field !== sortField) return <FaSort className="ml-1" />;
    return sortDirection === 'asc' ? <FaSortUp className="ml-1" /> : <FaSortDown
className="ml-1" />;
  };

  // Calculate summary statistics from filtered sales
  const calculateSummaryStats = () => {
    if (!filteredSales.length) return { totalSales: 0, totalRevenue: 0,
totalToken: 0, totalPending: 0, averageSaleValue: 0 };

    // Initialize values
    let totalRevenue = 0;
    let totalToken = 0;
    let totalPending = 0;

    // Process each sale
    filteredSales.forEach(sale => {
      // Convert amounts to the selected currency
      const totalCostInSelectedCurrency =
convertAmountToSelectedCurrency(sale.totalCost, sale.currency);
      const tokenAmountInSelectedCurrency =
convertAmountToSelectedCurrency(sale.tokenAmount, sale.currency);
      const pendingAmountInSelectedCurrency =
convertAmountToSelectedCurrency(sale.pendingAmount, sale.currency);
```

```
      // Add to totals
      totalRevenue += totalCostInSelectedCurrency;
      totalToken += tokenAmountInSelectedCurrency;
      totalPending += pendingAmountInSelectedCurrency;
    });

    // Calculate average sale value
    const averageSaleValue = filteredSales.length ? totalRevenue /
filteredSales.length : 0;

    return {
      totalSales: filteredSales.length,
      totalRevenue,
      totalToken,
      totalPending,
      averageSaleValue
    };
  };

  // Helper function to convert amount to selected currency
  const convertAmountToSelectedCurrency = (amount, fromCurrency) => {
    if (!amount) return 0;

    // If currencies match, no conversion needed
    if (fromCurrency === selectedCurrency) return amount;

    // Get exchange rates
    const rates = exchangeRates || getCurrencySettings().exchangeRates;

    // Convert to USD first (if not already USD)
    let amountInUSD;
    if (fromCurrency === 'USD') {
      amountInUSD = amount;
    } else {
      // If rate is available, use it, otherwise use a default 1:1 rate
      const fromRate = rates[fromCurrency] || 1;
      amountInUSD = amount / fromRate;
    }

    // Then convert from USD to selected currency
    const toRate = rates[selectedCurrency] || 1;
    return amountInUSD * toRate;
  };

  // Group sales by course and calculate metrics
  const calculateCourseMetrics = () => {
    if (!filteredSales.length) return [];

    const courseData = {};

    // Process each sale
    filteredSales.forEach(sale => {
      const course = sale.course || 'Unknown';
      const totalCostInSelectedCurrency =
convertAmountToSelectedCurrency(sale.totalCost, sale.currency);

      // Initialize course data if it doesn't exist
      if (!courseData[course]) {
```

```
        courseData[course] = {
          name: course,
          count: 0,
          totalRevenue: 0
        };
      }

      // Update course metrics
      courseData[course].count += 1;
      courseData[course].totalRevenue += totalCostInSelectedCurrency;
    });

    // Convert to array and sort by count (most popular first)
    return Object.values(courseData).sort((a, b) => b.count - a.count);
  };

  // Group sales by month/quarter/year for trend analysis
  const calculateTimeTrends = () => {
    if (!filteredSales.length) return [];

    const timeData = {};
    let timeFormat = '';

    // Determine format based on time frame
    switch (timeFrame) {
      case 'monthly':
        timeFormat = 'day'; // Group by day within month
        break;
      case 'quarterly':
        timeFormat = 'month'; // Group by month within quarter
        break;
      case 'yearly':
        timeFormat = 'month'; // Group by month within year
        break;
      default:
        timeFormat = 'month';
    }

    // Process each sale
    filteredSales.forEach(sale => {
      const saleDate = new Date(sale.date);
      let timeKey;

      // Format the time key based on time frame
      if (timeFormat === 'day') {
        timeKey = saleDate.getDate().toString(); // Day of month (1-31)
      } else if (timeFormat === 'month') {
        timeKey = (saleDate.getMonth() + 1).toString(); // Month (1-12)
      } else {
        timeKey = saleDate.getFullYear().toString(); // Year
      }

      const totalCostInSelectedCurrency =
convertAmountToSelectedCurrency(sale.totalCost, sale.currency);

      // Initialize time data if it doesn't exist
      if (!timeData[timeKey]) {
        timeData[timeKey] = {
          timeKey,
```

```
          count: 0,
          totalRevenue: 0
        };
      }

      // Update time metrics
      timeData[timeKey].count += 1;
      timeData[timeKey].totalRevenue += totalCostInSelectedCurrency;
    });

    // Convert to array and sort by time key
    return Object.values(timeData).sort((a, b) => {
      return parseInt(a.timeKey) - parseInt(b.timeKey);
    });
  };

  // Format trend data for chart display
  const formatTrendDataForChart = () => {
    const trendData = calculateTimeTrends();

    // Format labels based on time frame
    let labels = [];
    if (timeFrame === 'monthly') {
      // Days in month
      labels = trendData.map(item => `Day ${item.timeKey}`);
    } else if (timeFrame === 'quarterly') {
      // Months in quarter
      labels = trendData.map(item => months.find(m => m.value ===
parseInt(item.timeKey))?.label || item.timeKey);
    } else {
      // Months in year
      labels = trendData.map(item => months.find(m => m.value ===
parseInt(item.timeKey))?.label || item.timeKey);
    }

    // Dataset for revenue
    const revenueData = trendData.map(item => item.totalRevenue);

    // Dataset for count
    const countData = trendData.map(item => item.count);

    return {
      labels,
      datasets: [
        {
          label: 'Revenue',
          data: revenueData,
          borderColor: 'rgb(75, 192, 192)',
          backgroundColor: 'rgba(75, 192, 192, 0.5)',
          yAxisID: 'y',
        },
        {
          label: 'Number of Sales',
          data: countData,
          borderColor: 'rgb(53, 162, 235)',
          backgroundColor: 'rgba(53, 162, 235, 0.5)',
          yAxisID: 'y1',
        },
      ],
```

```
    };
  };

  // Format course data for chart display
  const formatCourseDataForChart = () => {
    const courseData = calculateCourseMetrics();

    // Only show top 10 courses for readability
    const topCourses = courseData.slice(0, 10);

    // Labels (course names)
    const labels = topCourses.map(course => course.name);

    // Dataset for count
    const countData = topCourses.map(course => course.count);

    // Dataset for revenue
    const revenueData = topCourses.map(course => course.totalRevenue);

    return {
      labels,
      countData,
      revenueData,
    };
  };

  // Function to handle exporting data to Excel
  const exportToExcel = () => {
    // Prepare data for export
    const exportData = filteredSales.map(sale => ({
      Date: new Date(sale.date).toLocaleDateString(),
      Customer: sale.customerName,
      Course: sale.course,
      'Total Cost':
formatCurrency(convertAmountToSelectedCurrency(sale.totalCost, sale.currency)),
      'Token Amount':
formatCurrency(convertAmountToSelectedCurrency(sale.tokenAmount, sale.currency)),
      'Pending Amount':
formatCurrency(convertAmountToSelectedCurrency(sale.pendingAmount,
sale.currency)),
      Status: sale.status,
      'Original Currency': sale.currency
    }));

    // Create worksheet
    const worksheet = XLSX.utils.json_to_sheet(exportData);

    // Create workbook
    const workbook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(workbook, worksheet, 'Sales Report');

    // Generate filename based on current filter
    let filename;
    if (timeFrame === 'monthly') {
      filename = `Sales_Report_${months.find(m => m.value ===
currentMonth)?.label}_${currentYear}.xlsx`;
    } else if (timeFrame === 'quarterly') {
      filename = `Sales_Report_Q${currentQuarter}_${currentYear}.xlsx`;
    } else {
```

```
    filename = `Sales_Report_${currentYear}.xlsx`;
  }

  // Export to file
  XLSX.writeFile(workbook, filename);
};

const [courseAnalysisData, setCourseAnalysisData] = useState(null);
const [revenueAnalysisData, setRevenueAnalysisData] = useState(null);
const [topCoursesData, setTopCoursesData] = useState(null);
const [statusAnalysisData, setStatusAnalysisData] = useState(null);
const [selectedCourseFilter, setSelectedCourseFilter] = useState('monthly');
const [selectedRevenueFilter, setSelectedRevenueFilter] = useState('1month');
const [selectedStatusFilter, setSelectedStatusFilter] = useState('1month');
const [selectedStatus, setSelectedStatus] = useState(null);

useEffect(() => {
  loadAllReports();
  loadExchangeRates();
}, []);

useEffect(() => {
  loadCourseAnalysis();
}, [selectedCourseFilter]);

useEffect(() => {
  loadRevenueAnalysis();
}, [selectedRevenueFilter]);

useEffect(() => {
  loadStatusAnalysis();
}, [selectedStatusFilter, selectedStatus]);

const loadExchangeRates = async () => {
  try {
    const response = await currencyAPI.getRates();
    if (response.data && response.data.rates) {
      setExchangeRates(response.data.rates);
    }
  } catch (error) {
    console.error('Error loading exchange rates:', error);
    // Use default rates as fallback
    setExchangeRates({
      'USD': 1,
      'EUR': 0.85,
      'GBP': 0.73,
      'INR': 83.12,
      'CAD': 1.36,
      'AUD': 1.52,
      'JPY': 149.50,
      'CNY': 7.24
    });
  }
};

const loadCourseAnalysis = async () => {
  try {
    setLoading(true);
    const response = await salesAPI.getCourseAnalysis(selectedCourseFilter);
```

```
      if (response.data && response.data.success) {
        setCourseAnalysisData(response.data.data);
      }
    } catch (error) {
      console.error('Error loading course analysis:', error);
      toast.error('Failed to load course analysis');
    } finally {
      setLoading(false);
    }
  };

  const loadRevenueAnalysis = async () => {
    try {
      setLoading(true);
      const response = await salesAPI.getRevenueAnalysis(selectedRevenueFilter);
      if (response.data && response.data.success) {
        setRevenueAnalysisData(response.data.data);
      }
    } catch (error) {
      console.error('Error loading revenue analysis:', error);
      toast.error('Failed to load revenue analysis');
    } finally {
      setLoading(false);
    }
  };

  const loadTopCourses = async () => {
    try {
      const response = await salesAPI.getTopCourses('all', 10);
      if (response.data && response.data.success) {
        setTopCoursesData(response.data.data);
      }
    } catch (error) {
      console.error('Error loading top courses:', error);
      toast.error('Failed to load top courses');
    }
  };

  const loadStatusAnalysis = async () => {
    try {
      setLoading(true);
      const response = await salesAPI.getStatusAnalysis(selectedStatusFilter,
selectedStatus);
      if (response.data && response.data.success) {
        setStatusAnalysisData(response.data.data);
      }
    } catch (error) {
      console.error('Error loading status analysis:', error);
      toast.error('Failed to load status analysis');
    } finally {
      setLoading(false);
    }
  };

  const loadAllReports = async () => {
    await Promise.all([
      loadCourseAnalysis(),
      loadRevenueAnalysis(),
      loadTopCourses(),
```

```
      loadStatusAnalysis()
    ]);
  };

  const renderCourseAnalysisTable = () => {
    if (!courseAnalysisData || !courseAnalysisData.courseAnalysis) {
      return <div className="text-center py-4">No course analysis data available</
div>;
    }

    const courses = Object.keys(courseAnalysisData.courseAnalysis);
    const periods = new Set();

    // Collect all periods
    courses.forEach(course => {
      Object.keys(courseAnalysisData.courseAnalysis[course]).forEach(period => {
        periods.add(period);
      });
    });

    const sortedPeriods = Array.from(periods).sort().reverse();

    return (
      <div className="overflow-x-auto">
        <table className="min-w-full bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700">
          <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
            <tr>
              <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                Course
              </th>
              {sortedPeriods.map(period => (
                <th key={period} className="px-6 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                  {period}
                </th>
              ))}
              <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                Total Sales
              </th>
            </tr>
          </thead>
          <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-slate-200 dark:divide-slate-700">
            {courses.map(course => {
              const courseData = courseAnalysisData.courseAnalysis[course];
              const totalSales = Object.values(courseData).reduce((sum, period)
=> sum + period.totalSales, 0);

              return (
                <tr key={course} className="hover:bg-slate-50 dark:hover:bg-
slate-800 dark:bg-slate-800 transition-all duration-200 ease-out">
                  <td className="px-6 py-4 whitespace-nowrap text-sm font-medium
text-slate-900 dark:text-slate-100">
                    {course}
                  </td>
```

```
                  {sortedPeriods.map(period => {
                    const periodData = courseData[period];
                    return (
                      <td key={period} className="px-6 py-4 whitespace-nowrap
text-sm text-slate-500 dark:text-gray-400">
                        {periodData ? (
                          <div>
                            <div className="font-
semibold">{periodData.totalSales} sales</div>
                            <div className="text-xs text-gray-400 dark:text-
gray-400">
                              ${periodData.totalRevenue?.toFixed(2) || '0.00'}
                            </div>
                          </div>
                        ) : (
                          <span className="text-gray-300 dark:text-gray-500">-</
span>
                        )}
                      </td>
                    );
                  })}
                  <td className="px-6 py-4 whitespace-nowrap text-sm font-
semibold text-blue-600">
                    {totalSales}
                  </td>
                </tr>
              );
            })}
          </tbody>
        </table>
      </div>
    );
  };

  const renderRevenueAnalysis = () => {
    if (!revenueAnalysisData) {
      return <div className="text-center py-4">No revenue analysis data
available</div>;
    }

    const { summary, currencyBreakdown, dailyBreakdown } = revenueAnalysisData;

    return (
      <div className="space-y-6">
        {/* Summary Cards */}
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
          <div className="bg-blue-50 p-6 rounded-lg border border-blue-200">
            <div className="flex items-center">
              <FaChartBar className="text-blue-600 text-2xl mr-3" />
              <div>
                <p className="text-sm font-medium text-blue-600">Total Sales</p>
                <p className="text-2xl font-bold text-
blue-900">{summary.totalSales}</p>
              </div>
            </div>
          </div>

          <div className="bg-green-50 p-6 rounded-lg border border-green-200">
            <div className="flex items-center">
```

```
              <FaDollarSign className="text-green-600 text-2xl mr-3" />
              <div>
                <p className="text-sm font-medium text-green-600">Total Revenue
(USD)</p>
                <p className="text-2xl font-bold text-
green-900">{formatCurrency(summary.totalRevenueUSD)}</p>
              </div>
            </div>
          </div>

          <div className="bg-yellow-50 p-6 rounded-lg border border-yellow-200">
            <div className="flex items-center">
              <FaChartLine className="text-yellow-600 text-2xl mr-3" />
              <div>
                <p className="text-sm font-medium text-yellow-600">Tokens
Received (USD)</p>
                <p className="text-2xl font-bold text-
yellow-900">{formatCurrency(summary.totalTokensUSD)}</p>
              </div>
            </div>
          </div>

          <div className="bg-red-50 p-6 rounded-lg border border-red-200">
            <div className="flex items-center">
              <FaCalendarAlt className="text-red-600 text-2xl mr-3" />
              <div>
                <p className="text-sm font-medium text-red-600">Pending Amount
(USD)</p>
                <p className="text-2xl font-bold text-
red-900">{formatCurrency(summary.pendingAmountUSD)}</p>
              </div>
            </div>
          </div>
        </div>

        {/* Currency Breakdown */}
        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-6 rounded-lg shadow-
md dark:shadow-2xl shadow-sm">
          <h3 className="text-lg font-semibold mb-4">Revenue by Currency</h3>
          <div className="overflow-x-auto">
            <table className="min-w-full">
              <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
                <tr>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Currency</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Sales</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Revenue (Original)</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Revenue (USD)</th>
                  <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Tokens (USD)</th>
                </tr>
              </thead>
              <tbody className="divide-y divide-slate-200 dark:divide-slate-700">
                {Object.entries(currencyBreakdown).map(([currency, data]) => (
```

```
                  <tr key={currency}>
                    <td className="px-4 py-2 font-medium">{currency}</td>
                    <td className="px-4 py-2">{data.totalSales}</td>
                    <td className="px-4 py-2">{formatCurrency(data.totalRevenue,
currency)}</td>
                    <td className="px-4 py-2">{formatCurrency(data.revenueUSD)}</
td>
                    <td className="px-4 py-2">{formatCurrency(data.tokensUSD)}</
td>
                  </tr>
                ))}
              </tbody>
            </table>
          </div>
        </div>

        {/* Daily Breakdown Chart */}
        {dailyBreakdown && dailyBreakdown.length > 0 && (
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-6 rounded-lg shadow-
md dark:shadow-2xl shadow-sm">
            <h3 className="text-lg font-semibold mb-4">Daily Sales Trend</h3>
            <div className="overflow-x-auto">
              <table className="min-w-full">
                <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
                  <tr>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Date</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Sales</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Revenue</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Tokens</th>
                  </tr>
                </thead>
                <tbody className="divide-y divide-slate-200 dark:divide-
slate-700">
                  {dailyBreakdown.slice(-10).map(day => (
                    <tr key={day.date}>
                      <td className="px-4 py-2 font-medium">{day.date}</td>
                      <td className="px-4 py-2">{day.sales}</td>
                      <td className="px-4 py-2">${day.revenue?.toFixed(2) ||
'0.00'}</td>
                      <td className="px-4 py-2">${day.tokens?.toFixed(2) ||
'0.00'}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </div>
        )}
      </div>
    );
  };

  const renderTopCourses = () => {
```

```
    if (!topCoursesData || !topCoursesData.topCourses) {
      return <div className="text-center py-4">No top courses data available</
div>;
    }

    return (
      <div className="overflow-x-auto">
        <table className="min-w-full bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700">
          <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
            <tr>
              <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                Rank
              </th>
              <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                Course
              </th>
              <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                Total Sales
              </th>
              <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                Total Revenue
              </th>
              <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                Average Price
              </th>
              <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                Completion Rate
              </th>
            </tr>
          </thead>
          <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-slate-200 dark:divide-slate-700">
            {topCoursesData.topCourses.map((course, index) => (
              <tr key={course.course} className="hover:bg-slate-50 dark:hover:bg-
slate-800 dark:bg-slate-800 transition-all duration-200 ease-out">
                <td className="px-6 py-4 whitespace-nowrap text-sm font-medium
text-slate-900 dark:text-slate-100">
                  #{index + 1}
                </td>
                <td className="px-6 py-4 whitespace-nowrap text-sm font-medium
text-blue-600">
                  {course.course}
                </td>
                <td className="px-6 py-4 whitespace-nowrap text-sm text-slate-900
dark:text-slate-100">
                  {course.totalSales}
                </td>
                <td className="px-6 py-4 whitespace-nowrap text-sm text-slate-900
dark:text-slate-100">
                  ${course.totalRevenue}
                </td>
```

```
                <td className="px-6 py-4 whitespace-nowrap text-sm text-slate-900
dark:text-slate-100">
                  ${course.averagePrice}
                </td>
                <td className="px-6 py-4 whitespace-nowrap text-sm text-slate-900
dark:text-slate-100">
                  <div className="flex items-center">
                    <div className="w-16 bg-gray-200 dark:bg-slate-600 rounded-
full h-2 mr-2">
                      <div
                        className="bg-green-600 h-2 rounded-full"
                        style={{ width: `${course.completionRate}%` }}
                      ></div>
                    </div>
                    <span>{course.completionRate}%</span>
                  </div>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    );
  };

  const renderStatusAnalysis = () => {
    if (!statusAnalysisData) {
      return <div className="text-center py-4">No status analysis data available</
div>;
    }

    const { statusSummary, detailedSales, selectedStatus: currentStatus } =
statusAnalysisData;

    return (
      <div className="space-y-6">
        {/* Status Summary Cards */}
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-
cols-4 gap-4">
          {statusSummary.map((statusData, index) => (
            <div
              key={statusData.status}
              className={`p-4 rounded-lg border cursor-pointer transition-all
duration-200 ${
                selectedStatus === statusData.status
                  ? 'border-blue-500 bg-blue-50'
                  : 'border-gray-200 dark:border-slate-700 bg-white dark:bg-
slate-900 hover:border-gray-300 dark:border-slate-600'
              }`}
              onClick={() => setSelectedStatus(selectedStatus ===
statusData.status ? null : statusData.status)}
            >
              <div className="flex items-center justify-between">
                <div>
                  <p className="text-sm font-medium text-gray-600 dark:text-
gray-500">{statusData.status}</p>
                  <p className="text-2xl font-bold text-slate-900 dark:text-
slate-100">{statusData.totalSales}</p>
                  <p className="text-xs text-slate-500 dark:text-gray-400">sales</
p>
```

```
                </div>
                <div className="text-right">
                  <p className="text-sm font-medium text-
green-600">{formatCurrency(statusData.totalRevenueUSD)}</p>
                  <p className="text-xs text-slate-500 dark:text-
gray-400">revenue</p>
                </div>
              </div>
              <div className="mt-2 pt-2 border-t border-gray-100">
                <div className="flex justify-between text-xs text-slate-500
dark:text-gray-400">
                  <span>Avg: {formatCurrency(statusData.averageOrderValueUSD)}</
span>
                  <span>Pending: {formatCurrency(statusData.pendingAmountUSD)}</
span>
                </div>
              </div>
            </div>
          ))}
        </div>

        {/* Detailed Sales Table (shown when a status is selected) */}
        {currentStatus && detailedSales && detailedSales.length > 0 && (
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-6 rounded-lg shadow-
md dark:shadow-2xl shadow-sm">
            <h3 className="text-lg font-semibold mb-4">
              Detailed Sales for "{currentStatus}" Status ({detailedSales.length}
sales)
            </h3>
            <div className="overflow-x-auto">
              <table className="min-w-full">
                <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
                  <tr>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Date</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Customer</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Course</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Country</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Sales Person</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Total Cost</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Token</th>
                    <th className="px-4 py-2 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase">Pending</th>
                  </tr>
                </thead>
                <tbody className="divide-y divide-slate-200 dark:divide-
slate-700">
                  {detailedSales.map(sale => (
                    <tr key={sale._id} className="hover:bg-slate-50 dark:hover:bg-
slate-800 dark:bg-slate-800 transition-all duration-200 ease-out">
                      <td className="px-4 py-2 text-sm">{new
```

```
Date(sale.date).toLocaleDateString()}</td>
                      <td className="px-4 py-2 text-sm font-
medium">{sale.customerName}</td>
                      <td className="px-4 py-2 text-sm">{sale.course}</td>
                      <td className="px-4 py-2 text-sm">{sale.country}</td>
                      <td className="px-4 py-2 text-sm">{sale.salesPerson}</td>
                      <td className="px-4 py-2 text-
sm">${sale.totalCost?.toFixed(2) || '0.00'}</td>
                      <td className="px-4 py-2 text-
sm">${sale.tokenAmount?.toFixed(2) || '0.00'}</td>
                      <td className="px-4 py-2 text-
sm">${sale.pendingAmount?.toFixed(2) || '0.00'}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </div>
        )}

        {/* Instructions */}
        <div className="bg-blue-50 p-4 rounded-lg">
          <p className="text-sm text-blue-700">
            <strong>Ø=Ü¡ Tip:</strong> Click on any status card above to view
detailed sales data for that status.
            The cards show total sales count, revenue, and pending amounts for
each status.
          </p>
        </div>
      </div>
    );
  };

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-6">
          <h2 className="text-3xl font-bold text-gray-800 dark:text-
gray-200">Sales Reports & Analytics</h2>
          <button
            onClick={loadAllReports}
            disabled={loading}
            className="bg-blue-600 hover:bg-blue-700 dark:bg-blue-500
dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md transition-all
duration-200 text-white py-2 px-4 rounded-md transition duration-300 flex items-
center"
          >
            {loading ? (
              <div className="animate-spin rounded-full h-4 w-4 border-b-2 border-
white mr-2"></div>
            ) : (
              <FaDownload className="mr-2" />
            )}
            Refresh Reports
          </button>
        </div>

        {/* Course Analysis Section */}
        <div className="bg-white dark:bg-slate-900 border border-slate-200
```

```
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 mb-8 shadow-sm">
          <div className="flex justify-between items-center mb-6">
            <h3 className="text-xl font-semibold text-gray-800 dark:text-gray-200
flex items-center">
              <FaGraduationCap className="mr-2 text-blue-600" />
              Course Sales Analysis
            </h3>
            <select
              value={selectedCourseFilter}
              onChange={(e) => setSelectedCourseFilter(e.target.value)}
              className="border border-slate-300 dark:border-slate-600 rounded-md
px-3 py-2 focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2"
            >
              {courseFilterOptions.map(option => (
                <option key={option.value} value={option.value}>
                  {option.label}
                </option>
              ))}
            </select>
          </div>
          {renderCourseAnalysisTable()}
        </div>

        {/* Revenue Analysis Section */}
        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 mb-8 shadow-sm">
          <div className="flex justify-between items-center mb-6">
            <h3 className="text-xl font-semibold text-gray-800 dark:text-gray-200
flex items-center">
              <FaDollarSign className="mr-2 text-green-600" />
              Revenue Analysis
            </h3>
            <select
              value={selectedRevenueFilter}
              onChange={(e) => setSelectedRevenueFilter(e.target.value)}
              className="border border-slate-300 dark:border-slate-600 rounded-md
px-3 py-2 focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2"
            >
              {revenueFilterOptions.map(option => (
                <option key={option.value} value={option.value}>
                  {option.label}
                </option>
              ))}
            </select>
          </div>
          {renderRevenueAnalysis()}
        </div>

        {/* Top Courses Section */}
        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 shadow-sm">
          <h3 className="text-xl font-semibold text-gray-800 dark:text-gray-200
mb-6 flex items-center">
            <FaChartBar className="mr-2 text-purple-600" />
```

```
            Top Performing Courses (All Time)
          </h3>
          {renderTopCourses()}
        </div>

        {/* Status Analysis Section */}
        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 shadow-sm">
          <div className="flex justify-between items-center mb-6">
            <h3 className="text-xl font-semibold text-gray-800 dark:text-gray-200
flex items-center">
              <FaChartPie className="mr-2 text-pink-600" />
              Status Analysis
            </h3>
            <div className="flex space-x-4">
              <select
                value={selectedStatusFilter}
                onChange={(e) => setSelectedStatusFilter(e.target.value)}
                className="border border-slate-300 dark:border-slate-600 rounded-
md px-3 py-2 focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2"
              >
                <option value="1month">1 Month</option>
                <option value="3month">3 Months</option>
                <option value="6month">6 Months</option>
                <option value="1year">1 Year</option>
                <option value="all">All Time</option>
              </select>
              {selectedStatus && (
                <button
                  onClick={() => setSelectedStatus(null)}
                  className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out0 hover:bg-gray-600 text-white px-3 py-2 rounded-md text-sm"
                >
                  Clear Selection
                </button>
              )}
            </div>
          </div>
          {renderStatusAnalysis()}
        </div>
      </div>
    </Layout>
  );
};

export default AdminReportsPage;
```

src/pages/AdminUsersPage.jsx

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import Layout from "../components/Layout/Layout";
import { authAPI } from "../services/api";
import employeeAPI from "../services/employeeAPI";
import { useAuth } from "../context/AuthContext";
import EditEmployeeDialog from "../components/Employee/EditEmployeeDialog";
```

```
const AdminUsersPage = () => {
  const { user } = useAuth();
  const [activeTab, setActiveTab] = useState("users");
  const [users, setUsers] = useState([]);
  const [employees, setEmployees] = useState([]);
  const [departments, setDepartments] = useState([]);
  const [roles, setRoles] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");
  const [filterRole, setFilterRole] = useState("");
  const [showModal, setShowModal] = useState(false);
  const [showEditModal, setShowEditModal] = useState(false);
  const [showEmployeeEditModal, setShowEmployeeEditModal] = useState(false);
  const [confirmDelete, setConfirmDelete] = useState(null);
  const [editUser, setEditUser] = useState(null);
  const [editEmployeeId, setEditEmployeeId] = useState(null);
  const [editDocuments, setEditDocuments] = useState({
    photograph: null,
    tenthMarksheet: null,
    twelfthMarksheet: null,
    bachelorDegree: null,
    postgraduateDegree: null,
    aadharCard: null,
    panCard: null,
    pcc: null,
    resume: null,
    offerLetter: null
  });
  const [newUser, setNewUser] = useState({
    fullName: "",
    email: "",
    password: "",
    role: "Sales Person",
    // Employee-specific fields
    phoneNumber: "",
    whatsappNumber: "",
    linkedInUrl: "",
    currentAddress: "",
    permanentAddress: "",
    dateOfBirth: "",
    joiningDate: "",
    salary: "",
    department: "",
    employeeRole: "",
    status: "ACTIVE",
    collegeName: "",
    internshipDuration: ""
  });

  const [documents, setDocuments] = useState({
    photograph: null,
    tenthMarksheet: null,
    twelfthMarksheet: null,
    bachelorDegree: null,
    postgraduateDegree: null,
    aadharCard: null,
    panCard: null,
```

```
  pcc: null,
  resume: null,
  offerLetter: null
});

const [uploadProgress, setUploadProgress] = useState({});

useEffect(() => {
  fetchUsers();
  fetchEmployees();
  fetchDepartmentsAndRoles();
}, []);

const fetchUsers = async () => {
  try {
    setLoading(true);
    const response = await authAPI.getUsers();
    if (response.data.success) {
      setUsers(response.data.data);
    } else {
      setError("Failed to load users");
    }
  } catch (err) {
    console.error("Error fetching users:", err);
    setError("Failed to load users. Please try again.");
  } finally {
    setLoading(false);
  }
};

const fetchEmployees = async () => {
  try {
    const response = await employeeAPI.getAll();
    if (response.data.success) {
      setEmployees(response.data.data);
    }
  } catch (err) {
    console.error("Error fetching employees:", err);
  }
};

const fetchDepartmentsAndRoles = async () => {
  try {
    const [departmentsRes, rolesRes] = await Promise.all([
      employeeAPI.getDepartments(),
      employeeAPI.getRoles()
    ]);

    if (departmentsRes.data.success) {
      setDepartments(departmentsRes.data.data);
    }
    if (rolesRes.data.success) {
      setRoles(rolesRes.data.data);
    }
  } catch (err) {
    console.error("Error fetching departments and roles:", err);
  }
};
```

```
const handleInputChange = (e) => {
  setNewUser({
    ...newUser,
    [e.target.name]: e.target.value
  });
};

const handleEditInputChange = (e) => {
  const { name, value } = e.target;
  setEditUser(prev => ({
    ...prev,
    [name]: value
  }));
};

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

  try {
    setLoading(true);
    setError(null);

    // Create FormData for file uploads
    const formData = new FormData();

    // Add main user data directly (not nested in employee object)
    formData.append('fullName', newUser.fullName);
    formData.append('email', newUser.email);
    formData.append('password', newUser.password);
    formData.append('role', newUser.role);

    // Add employee-specific fields
    formData.append('phoneNumber', newUser.phoneNumber || '');
    formData.append('whatsappNumber', newUser.whatsappNumber || '');
    formData.append('linkedInUrl', newUser.linkedInUrl || '');
    formData.append('currentAddress', newUser.currentAddress || '');
    formData.append('permanentAddress', newUser.permanentAddress || '');
    formData.append('dateOfBirth', newUser.dateOfBirth || '');
    formData.append('joiningDate', newUser.joiningDate || '');
    formData.append('salary', newUser.salary || '');
    formData.append('department', newUser.department || '');
    formData.append('employeeRole', newUser.employeeRole || '');
    formData.append('status', newUser.status || 'ACTIVE');
    formData.append('collegeName', newUser.collegeName || '');
    formData.append('internshipDuration', newUser.internshipDuration || '');

    // Add documents
    Object.keys(documents).forEach(key => {
      if (documents[key]) {
        formData.append(key, documents[key]);
      }
    });

    // Create user with documents
    const response = await authAPI.createUserWithDocuments(formData);

    if (response.data.success) {
      fetchUsers();
      fetchEmployees();
```

```
        setShowModal(false);
        resetForm();
        alert(`User created successfully!\nEmployee ID:
${response.data.employeeId || 'Generated'}\nPlease save this information.`);
      } else {
        setError(response.data.message || "Failed to create user");
      }
    } catch (err) {
      console.error("Error creating user:", err);
      setError(err.response?.data?.message || "Failed to create user. Please try
again.");
    } finally {
      setLoading(false);
    }
  };

  const resetForm = () => {
    setNewUser({
      fullName: "",
      email: "",
      password: "",
      role: "Sales Person",
      // Employee-specific fields
      phoneNumber: "",
      whatsappNumber: "",
      linkedInUrl: "",
      currentAddress: "",
      permanentAddress: "",
      dateOfBirth: "",
      joiningDate: "",
      salary: "",
      department: "",
      employeeRole: "",
      status: "ACTIVE",
      collegeName: "",
      internshipDuration: ""
    });
    setDocuments({
      photograph: null,
      tenthMarksheet: null,
      twelfthMarksheet: null,
      bachelorDegree: null,
      postgraduateDegree: null,
      aadharCard: null,
      panCard: null,
      pcc: null,
      resume: null,
      offerLetter: null
    });
    setEditDocuments({
      photograph: null,
      tenthMarksheet: null,
      twelfthMarksheet: null,
      bachelorDegree: null,
      postgraduateDegree: null,
      aadharCard: null,
      panCard: null,
      pcc: null,
      resume: null,
```

```
      offerLetter: null
    });
    setUploadProgress({});
  };

  const generateReferenceId = () => {
    return 'REF' + Date.now().toString(36).toUpperCase();
  };

  const handleDocumentChange = (documentType, file) => {
    setDocuments(prev => ({
      ...prev,
      [documentType]: file
    }));
  };

  const handleEditDocumentChange = (documentType, file) => {
    setEditDocuments(prev => ({
      ...prev,
      [documentType]: file
    }));
  };

  const handleUpdateUser = async (e) => {
    e.preventDefault();
    try {
      // Validate password if provided
      if (editUser.password && editUser.password.length > 0 &&
editUser.password.length < 6) {
        setError("Password must be at least 6 characters");
        return;
      }

      setLoading(true);

      // Create FormData for handling file uploads and all employee fields
      const formData = new FormData();

      // Basic user fields
      formData.append('fullName', editUser.fullName);
      formData.append('email', editUser.email);
      formData.append('role', editUser.role);
      if (editUser.password) {
        formData.append('password', editUser.password);
      }

      // Employee-specific fields
      formData.append('phoneNumber', editUser.phoneNumber || '');
      formData.append('whatsappNumber', editUser.whatsappNumber || '');
      formData.append('linkedInUrl', editUser.linkedInUrl || '');
      formData.append('currentAddress', editUser.currentAddress || '');
      formData.append('permanentAddress', editUser.permanentAddress || '');
      formData.append('dateOfBirth', editUser.dateOfBirth || '');
      formData.append('joiningDate', editUser.joiningDate || '');
      formData.append('salary', editUser.salary || '');
      formData.append('department', editUser.department || '');
      formData.append('employeeRole', editUser.employeeRole || '');
      formData.append('status', editUser.status || 'ACTIVE');
      formData.append('collegeName', editUser.collegeName || '');
```

```
      formData.append('internshipDuration', editUser.internshipDuration || '');

      // Append documents to FormData
      Object.keys(editDocuments).forEach(key => {
        if (editDocuments[key]) {
          formData.append(key, editDocuments[key]);
        }
      });

      // Use the updateUserWithDocuments endpoint
      const response = await authAPI.updateUserWithDocuments(editUser._id,
formData);

      if (response.data && response.data.success) {
        setShowEditModal(false);
        fetchUsers();
        setError(null);
        // Reset edit documents
        setEditDocuments({
          photograph: null,
          tenthMarksheet: null,
          twelfthMarksheet: null,
          bachelorDegree: null,
          postgraduateDegree: null,
          aadharCard: null,
          panCard: null,
          pcc: null,
          resume: null,
          offerLetter: null
        });
      } else {
        setError((response.data && response.data.message) || "Failed to update
user");
      }
    } catch (err) {
      console.error("Error updating user:", err);
      setError(
        err.response?.data?.message ||
        "Failed to update user. Please check your connection and try again."
      );
    } finally {
      setLoading(false);
    }
  };

  const handleDeleteUser = async (userId) => {
    try {
      setLoading(true);
      const response = await authAPI.deleteUser(userId);

      if (response.data && response.data.success) {
        // Refresh user list
        fetchUsers();
        setConfirmDelete(null);
        setError(null);
      } else {
        setError((response.data && response.data.message) || "Failed to delete
user");
      }
```

```
    } catch (err) {
      console.error("Error deleting user:", err);
      setError(
        err.response?.data?.message ||
        "Failed to delete user. Please check your connection and try again."
      );
      setConfirmDelete(null);
    } finally {
      setLoading(false);
    }
  };

  const openEditModal = async (userItem) => {
    try {
      // Fetch employee data for this user
      let employeeData = {};
      if (['Sales Person', 'Lead Person', 'Manager',
'Employee'].includes(userItem.role)) {
        const response = await employeeAPI.getEmployees();
        if (response.data && response.data.success) {
          const employee = response.data.data.find(emp => emp.userId ===
userItem._id);
          if (employee) {
            employeeData = {
              phoneNumber: employee.phoneNumber || '',
              whatsappNumber: employee.whatsappNumber || '',
              linkedInUrl: employee.linkedInUrl || '',
              currentAddress: employee.currentAddress || '',
              permanentAddress: employee.permanentAddress || '',
              dateOfBirth: employee.dateOfBirth ? new
Date(employee.dateOfBirth).toISOString().split('T')[0] : '',
              joiningDate: employee.joiningDate ? new
Date(employee.joiningDate).toISOString().split('T')[0] : '',
              salary: employee.salary || '',
              department: employee.department?._id || '',
              employeeRole: employee.role?._id || '',
              status: employee.status || 'ACTIVE',
              collegeName: employee.collegeName || '',
              internshipDuration: employee.internshipDuration || ''
            };
          }
        }
      }

      // Create a comprehensive edit user object
      setEditUser({
        _id: userItem._id,
        fullName: userItem.fullName,
        email: userItem.email,
        role: userItem.role,
        password: "", // Empty password field for optional update
        // Employee-specific fields
        phoneNumber: employeeData.phoneNumber || '',
        whatsappNumber: employeeData.whatsappNumber || '',
        linkedInUrl: employeeData.linkedInUrl || '',
        currentAddress: employeeData.currentAddress || '',
        permanentAddress: employeeData.permanentAddress || '',
        dateOfBirth: employeeData.dateOfBirth || '',
        joiningDate: employeeData.joiningDate || '',
```

```
      salary: employeeData.salary || '',
      department: employeeData.department || '',
      employeeRole: employeeData.employeeRole || '',
      status: employeeData.status || 'ACTIVE',
      collegeName: employeeData.collegeName || '',
      internshipDuration: employeeData.internshipDuration || ''
    });
    setShowEditModal(true);
  } catch (error) {
    console.error('Error loading employee data:', error);
    // Still show modal with basic user data if employee data fails
    setEditUser({
      _id: userItem._id,
      fullName: userItem.fullName,
      email: userItem.email,
      role: userItem.role,
      password: "", // Empty password field for optional update
      // Employee-specific fields with defaults
      phoneNumber: '',
      whatsappNumber: '',
      linkedInUrl: '',
      currentAddress: '',
      permanentAddress: '',
      dateOfBirth: '',
      joiningDate: '',
      salary: '',
      department: '',
      employeeRole: '',
      status: 'ACTIVE',
      collegeName: '',
      internshipDuration: ''
    });
    setShowEditModal(true);
  }
};

const handleEditEmployee = (employeeId) => {
  setEditEmployeeId(employeeId);
  setShowEmployeeEditModal(true);
};

const handleEmployeeUpdated = () => {
  fetchEmployees();
  setShowEmployeeEditModal(false);
  setEditEmployeeId(null);
};

const createUserAccountForEmployee = async (employee) => {
  try {
    // Check if user already exists
    const existingUser = users.find(u => u.email === employee.email);
    if (existingUser) {
      setError(`User account already exists for ${employee.email}`);
      return;
    }

    const userData = {
      fullName: employee.fullName,
      email: employee.email,
```

```
        password: generateTempPassword(),
        role: employee.role?.name || "Employee"
      };

      const response = await authAPI.createUser(userData);
      if (response.data.success) {
        // Update employee with user ID
        await employeeAPI.update(employee._id, {
          employee: JSON.stringify({ userId: response.data.data._id })
        });

        fetchUsers();
        fetchEmployees();
        alert(`User account created successfully for
${employee.fullName}\nTemporary Password: ${userData.password}\nPlease share this
with the employee securely.`);
      }
    } catch (err) {
      console.error("Error creating user account:", err);
      setError("Failed to create user account. Please try again.");
    }
  };

  const generateTempPassword = () => {
    return Math.random().toString(36).slice(-8) +
Math.random().toString(36).slice(-8).toUpperCase();
  };

  // Format date for display
  const formatDate = (dateString) => {
    const date = new Date(dateString);
    return date.toLocaleDateString();
  };

  // Filter users based on search term and role filter
  const filteredUsers = users.filter(user => {
    const matchesSearch =
user.fullName.toLowerCase().includes(searchTerm.toLowerCase()) ||

user.email.toLowerCase().includes(searchTerm.toLowerCase());
    const matchesRole = filterRole === "" || user.role === filterRole;
    return matchesSearch && matchesRole;
  });

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-3xl font-bold">Manage Users</h1>
          <div className="flex gap-3">
            <Link
              to="/employees"
              className="px-4 py-2 bg-indigo-600 text-white rounded-md hover:bg-
indigo-700 transition flex items-center gap-2"
            >
              <svg xmlns="http://www.w3.org/2000/svg" className="h-4 w-4"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M21 13.255A23.931 23.931 0 0112 15c-3.183
```

```
0-6.22-.62-9-1.745M16 6V4a2 2 0 00-2-2h-4a2 2 0 00-2-2v2m8 0V6a2 2 0 012 2v6a2 2
0 01-2 2H8a2 2 0 01-2-2V8a2 2 0 012-2h8zM8 14v.01M12 14v.01M16 14v.01" />
              </svg>
              Employee Management
            </Link>
            {activeTab === "users" && (
              <button
                onClick={() => setShowModal(true)}
                className="px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-
blue-700 transition"
              >
                Add New User
              </button>
            )}
          </div>
        </div>

        {/* Tabs */}
        <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md mb-6">
          <div className="border-b border-gray-200 dark:border-gray-700">
            <nav className="flex space-x-8 px-6">
              <button
                onClick={() => setActiveTab("users")}
                className={`${
                  activeTab === "users"
                    ? "border-blue-500 text-blue-600 dark:text-blue-400"
                    : "border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300"
                } whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm flex
items-center`}
              >
                <svg xmlns="http://www.w3.org/2000/svg" className="h-4 w-4 mr-2"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 4.354a4 4 0 110 5.292M15 21H3v-1a6 6 0 0112 0v1zm0
0h6v-1a6 6 0 00-9-2.239" />
                </svg>
                User Accounts
              </button>
              <button
                onClick={() => setActiveTab("employees")}
                className={`${
                  activeTab === "employees"
                    ? "border-blue-500 text-blue-600 dark:text-blue-400"
                    : "border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300"
                } whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm flex
items-center`}
              >
                <svg xmlns="http://www.w3.org/2000/svg" className="h-4 w-4 mr-2"
fill="none" viewBox="0 0 24 24" stroke="currentColor">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M17 20h5v-2a3 3 0 00-5.356-1.857M17 20H7m10
0v-2c0-.656-.126-1.283-.356-1.857M7 20H2v-2a3 3 0 015.356-1.857M7
20v-2c0-.656.126-1.283.356-1.857m0 0a5.002 5.002 0 019.288 0M15 7a3 3 0 11-6 0 3
3 0 016 0zm6 3a2 2 0 11-4 0 2 2 0 014 0zM7 10a2 2 0 11-4 0 2 2 0 014 0z" />
                </svg>
                Employees ({employees.length})
              </button>
```

```
            </nav>
          </div>
        </div>

        {error && (
          <div className="bg-red-50 text-red-600 p-4 rounded-lg mb-6">
            {error}
          </div>
        )}

        {/* User Accounts Tab */}
        {activeTab === "users" && (
          <>
            {/* User Statistics */}
            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6
mb-6">
          <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
            <div className="flex items-center">
              <div className="p-3 rounded-full bg-blue-100 dark:bg-blue-900">
                <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6 text-
blue-600 dark:text-blue-400" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 4.354a4 4 0 110 5.292M15 21H3v-1a6 6 0 0112 0v1zm0
0h6v-1a6 6 0 00-9-2.239" />
                </svg>
              </div>
              <div className="ml-4">
                <p className="text-sm font-medium text-gray-600 dark:text-
gray-400">Total Users</p>
                <p className="text-2xl font-semibold text-gray-900 dark:text-
white">{users.length}</p>
              </div>
            </div>
          </div>

          <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
            <div className="flex items-center">
              <div className="p-3 rounded-full bg-green-100 dark:bg-green-900">
                <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6 text-
green-600 dark:text-green-400" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 12l2 2 4-4m6 2a9 9 0 11-18 0 9 9 0 0118 0z" />
                </svg>
              </div>
              <div className="ml-4">
                <p className="text-sm font-medium text-gray-600 dark:text-
gray-400">Sales Team</p>
                <p className="text-2xl font-semibold text-gray-900 dark:text-
white">
                  {users.filter(u => u.role === "Sales Person").length}
                </p>
              </div>
            </div>
          </div>

          <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
            <div className="flex items-center">
```

```
              <div className="p-3 rounded-full bg-purple-100 dark:bg-purple-900">
                <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6 text-
purple-600 dark:text-purple-400" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M19 21V5a2 2 0 00-2-2H7a2 2 0 00-2 2v16m14 0h2m-2 0h-5m-9
0H3m2 0h5M9 7h1m-1 4h1m4-4h1m-1 4h1m-5 10v-5a1 1 0 011-1h2a1 1 0 011 1v5m-4 0h4" /
>
                </svg>
              </div>
              <div className="ml-4">
                <p className="text-sm font-medium text-gray-600 dark:text-
gray-400">Managers</p>
                <p className="text-2xl font-semibold text-gray-900 dark:text-
white">
                  {users.filter(u => u.role === "Manager").length}
                </p>
              </div>
            </div>
          </div>

          <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md p-6">
            <div className="flex items-center">
              <div className="p-3 rounded-full bg-indigo-100 dark:bg-indigo-900">
                <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6 text-
indigo-600 dark:text-indigo-400" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M21 13.255A23.931 23.931 0 0112 15c-3.183
0-6.22-.62-9-1.745M16 6V4a2 2 0 00-2-2h-4a2 2 0 00-2-2v2m8 0V6a2 2 0 012 2v6a2 2
0 01-2 2H8a2 2 0 01-2-2V8a2 2 0 012-2h8zM8 14v.01M12 14v.01M16 14v.01" />
                </svg>
              </div>
              <div className="ml-4">
                <p className="text-sm font-medium text-gray-600 dark:text-
gray-400">HR & Employees</p>
                <p className="text-2xl font-semibold text-gray-900 dark:text-
white">
                  {users.filter(u => u.role === "HR" || u.role ===
"Employee").length}
                </p>
              </div>
            </div>
          </div>
        </div>

        {/* Filters */}
        <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out rounded-lg shadow-md dark:shadow-black/25 p-6 mb-6">
          <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
            <div>
              <label htmlFor="search" className="block text-sm font-medium text-
gray-700 dark:text-gray-400 mb-1">
                Search Users
              </label>
              <input
                type="text"
                id="search"
                placeholder="Search by name or email"
```

```
                value={searchTerm}
                onChange={(e) => setSearchTerm(e.target.value)}
                className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100 focus:outline-
none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
              />
            </div>
            <div>
              <label htmlFor="roleFilter" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                Filter by Role
              </label>
              <select
                id="roleFilter"
                value={filterRole}
                onChange={(e) => setFilterRole(e.target.value)}
                className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100 focus:outline-
none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-blue-500"
              >
                <option value="">All Roles</option>
                <option value="Sales Person">Sales Person</option>
                <option value="Lead Person">Lead Person</option>
                <option value="Manager">Manager</option>
                <option value="Admin">Admin</option>
                <option value="HR">HR</option>
                <option value="Employee">Employee</option>
              </select>
            </div>
            <div className="flex items-end">
              <button
                onClick={() => {
                  setSearchTerm("");
                  setFilterRole("");
                }}
                className="px-4 py-2 border border-gray-300 dark:border-slate-600
text-gray-700 dark:text-gray-400 rounded-md hover:bg-gray-50 dark:bg-slate-800
transition"
              >
                Clear Filters
              </button>
            </div>
          </div>
        </div>

        {/* Users Table */}
        {loading ? (
          <div className="flex justify-center items-center h-64">
            <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-
b-2 border-blue-500"></div>
          </div>
        ) : (
          <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out rounded-lg shadow-md dark:shadow-black/25 overflow-hidden">
            {filteredUsers.length === 0 ? (
              <div className="p-6 text-center text-gray-500 dark:text-gray-500">
                No users found. Try adjusting your filters.
              </div>
            ) : (
```

```
              <div className="overflow-x-auto">
                <table className="min-w-full">
                  <thead className="bg-gray-50 dark:bg-slate-800">
                    <tr>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">
                        Name
                      </th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">
                        Email
                      </th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">
                        Role
                      </th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">
                        Created
                      </th>
                      <th className="px-6 py-3 text-right text-xs font-medium
text-gray-500 dark:text-gray-500 uppercase tracking-wider">
                        Actions
                      </th>
                    </tr>
                  </thead>
                  <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-gray-200 dark:divide-slate-700">
                    {filteredUsers.map((userItem) => (
                      <tr key={userItem._id} className="hover:bg-gray-50 dark:bg-
slate-800">
                        <td className="px-6 py-4 whitespace-nowrap">
                          <div className="flex items-center">
                            <div className="h-10 w-10 flex-shrink-0 bg-gray-200
dark:bg-slate-600 rounded-full flex items-center justify-center text-gray-500
dark:text-gray-500">
                              {userItem.fullName.charAt(0).toUpperCase()}
                            </div>
                            <div className="ml-4">
                              <div className="text-sm font-medium text-gray-900
dark:text-white">{userItem.fullName}</div>
                              <div className="text-sm text-gray-500 dark:text-
gray-500">ID: {userItem._id}</div>
                            </div>
                          </div>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap">
                          <div className="text-sm text-gray-900 dark:text-
white">{userItem.email}</div>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap">
                          <span className={`px-2 py-1 inline-flex text-xs
leading-5 font-semibold rounded-full ${
                            userItem.role === "Admin" ? "bg-purple-100 text-
purple-800 dark:bg-purple-900 dark:text-purple-200" :
                            userItem.role === "Manager" ? "bg-blue-100 text-
blue-800 dark:bg-blue-900 dark:text-blue-200" :
                            userItem.role === "Lead Person" ? "bg-green-100 text-
green-800 dark:bg-green-900 dark:text-green-200" :
```

```
                            userItem.role === "HR" ? "bg-indigo-100 text-
indigo-800 dark:bg-indigo-900 dark:text-indigo-200" :
                            userItem.role === "Employee" ? "bg-gray-100 text-
gray-800 dark:bg-gray-700 dark:text-gray-200" :
                            "bg-yellow-100 text-yellow-800 dark:bg-yellow-900
dark:text-yellow-200"
                          }`}>
                            {userItem.role}
                          </span>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-500">
                          {formatDate(userItem.createdAt)}
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-right
text-sm font-medium">
                          {userItem._id !== user.id && (
                            <div className="flex justify-end space-x-2">
                              <button
                                onClick={() => openEditModal(userItem)}
                                className="text-blue-600 hover:text-blue-900 px-2
py-1"
                              >
                                Edit
                              </button>
                              <button
                                onClick={() => setConfirmDelete(userItem)}
                                className="text-red-600 hover:text-red-900 px-2
py-1"
                              >
                                Delete
                              </button>
                            </div>
                          )}
                          {userItem._id === user.id && (
                            <span className="text-gray-500 dark:text-
gray-500">Current User</span>
                          )}
                        </td>
                      </tr>
                    ))}
                  </tbody>
                </table>
              </div>
            )}
          </div>
        )}
          </>
        )}

        {/* Employees Tab */}
        {activeTab === "employees" && (
          <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out rounded-lg shadow-md dark:shadow-black/25 overflow-hidden">
            <div className="px-6 py-4 border-b border-gray-200 dark:border-
gray-700">
              <h3 className="text-lg font-semibold text-gray-900 dark:text-white">
                Registered Employees
              </h3>
```

```
              <p className="text-sm text-gray-600 dark:text-gray-400 mt-1">
                Manage employee details and create user accounts for CRM access
              </p>
            </div>

            {loading ? (
              <div className="flex justify-center items-center h-64">
                <div className="animate-spin rounded-full h-12 w-12 border-t-2
border-b-2 border-blue-500"></div>
              </div>
            ) : employees.length === 0 ? (
              <div className="p-6 text-center text-gray-500 dark:text-gray-400">
                <svg xmlns="http://www.w3.org/2000/svg" className="h-12 w-12 mx-
auto mb-4 text-gray-400" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M17 20h5v-2a3 3 0 00-5.356-1.857M17 20H7m10
0v-2c0-.656-.126-1.283-.356-1.857M7 20H2v-2a3 3 0 015.356-1.857M7
20v-2c0-.656.126-1.283.356-1.857m0 0a5.002 5.002 0 019.288 0M15 7a3 3 0 11-6 0 3
3 0 016 0zm6 3a2 2 0 11-4 0 2 2 0 014 0zM7 10a2 2 0 11-4 0 2 2 0 014 0z" />
                </svg>
                <h3 className="text-lg font-medium mb-2">No employees found</h3>
                <p className="text-sm">No employees are registered in the system
yet.</p>
              </div>
            ) : (
              <div className="overflow-x-auto">
                <table className="min-w-full">
                  <thead className="bg-gray-50 dark:bg-slate-800">
                    <tr>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                        Employee
                      </th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                        Contact
                      </th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                        Department
                      </th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                        Role
                      </th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                        Status
                      </th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-400 uppercase tracking-wider">
                        User Account
                      </th>
                      <th className="px-6 py-3 text-right text-xs font-medium
text-gray-500 dark:text-gray-400 uppercase tracking-wider">
                        Actions
                      </th>
                    </tr>
                  </thead>
```

```
                  <tbody className="bg-white dark:bg-slate-900 divide-y divide-
gray-200 dark:divide-gray-700">
                    {employees.map((employee) => {
                      const hasUserAccount = users.some(u => u.email ===
employee.email);
                      return (
                        <tr key={employee._id} className="hover:bg-gray-50
dark:hover:bg-slate-800">
                          <td className="px-6 py-4 whitespace-nowrap">
                            <div className="flex items-center">
                              <div className="h-10 w-10 flex-shrink-0 bg-gray-200
dark:bg-slate-600 rounded-full flex items-center justify-center text-gray-500
dark:text-gray-400">
                                {employee.fullName?.charAt(0)?.toUpperCase() ||
'E'}
                              </div>
                              <div className="ml-4">
                                <div className="text-sm font-medium text-gray-900
dark:text-white">
                                  {employee.fullName || 'N/A'}
                                </div>
                                <div className="text-sm text-gray-500 dark:text-
gray-400">
                                  ID: {employee.employeeId || employee._id}
                                </div>
                              </div>
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap">
                            <div className="text-sm text-gray-900 dark:text-
white">
                              {employee.email || 'N/A'}
                            </div>
                            <div className="text-sm text-gray-500 dark:text-
gray-400">
                              {employee.phoneNumber || 'N/A'}
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap">
                            <div className="text-sm text-gray-900 dark:text-
white">
                              {employee.department?.name || 'N/A'}
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap">
                            <div className="text-sm text-gray-900 dark:text-
white">
                              {employee.role?.name || 'N/A'}
                            </div>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap">
                            <span className={`px-2 py-1 inline-flex text-xs
leading-5 font-semibold rounded-full ${
                              employee.status === 'Active' ? 'bg-green-100 text-
green-800 dark:bg-green-900 dark:text-green-200' :
                              employee.status === 'Inactive' ? 'bg-red-100 text-
red-800 dark:bg-red-900 dark:text-red-200' :
                              'bg-gray-100 text-gray-800 dark:bg-gray-700
dark:text-gray-200'
```

```
                            }`}>
                              {employee.status || 'N/A'}
                            </span>
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap">
                            {hasUserAccount ? (
                              <span className="px-2 py-1 inline-flex text-xs
leading-5 font-semibold rounded-full bg-green-100 text-green-800 dark:bg-
green-900 dark:text-green-200">
                                <svg xmlns="http://www.w3.org/2000/svg"
className="h-3 w-3 mr-1" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                                  <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M5 13l4 4L19 7" />
                                </svg>
                                Active
                              </span>
                            ) : (
                              <span className="px-2 py-1 inline-flex text-xs
leading-5 font-semibold rounded-full bg-red-100 text-red-800 dark:bg-red-900
dark:text-red-200">
                                <svg xmlns="http://www.w3.org/2000/svg"
className="h-3 w-3 mr-1" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                                  <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M6 18L18 6M6 6l12 12" />
                                </svg>
                                None
                              </span>
                            )}
                          </td>
                          <td className="px-6 py-4 whitespace-nowrap text-right
text-sm font-medium">
                            <div className="flex justify-end space-x-2">
                              <button
                                onClick={() => handleEditEmployee(employee._id)}
                                className="text-blue-600 hover:text-blue-900
dark:text-blue-400 dark:hover:text-blue-300 px-2 py-1 rounded"
                                title="Edit Employee Details"
                              >
                                <svg xmlns="http://www.w3.org/2000/svg"
className="h-4 w-4" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                                  <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M11 5H6a2 2 0 00-2 2v11a2 2 0 002
2h11a2 2 0 002-2v-5m-1.414-9.414a2 2 0 112.828 2.828L11.828
15H9v-2.828l8.586-8.586z" />
                                </svg>
                              </button>
                              {!hasUserAccount && employee.email && (
                                <button
                                  onClick={() =>
createUserAccountForEmployee(employee)}
                                  className="text-green-600 hover:text-green-900
dark:text-green-400 dark:hover:text-green-300 px-2 py-1 rounded"
                                  title="Create User Account"
                                >
                                  <svg xmlns="http://www.w3.org/2000/svg"
className="h-4 w-4" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                                    <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M12 6v6m0 0v6m0-6h6m-6 0H6" />
                                  </svg>
```

```
                                </button>
                              )}
                              {hasUserAccount && (
                                <span className="text-gray-400 dark:text-gray-500
px-2 py-1" title="User account already exists">
                                  <svg xmlns="http://www.w3.org/2000/svg"
className="h-4 w-4" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                                    <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M5 13l4 4L19 7" />
                                  </svg>
                                </span>
                              )}
                            </div>
                          </td>
                        </tr>
                      );
                    })}
                  </tbody>
                </table>
              </div>
            )}
          </div>
        )}
      </div>

      {/* Add User Modal */}
      {showModal && (
        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
          <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out rounded-lg shadow-lg dark:shadow-black/25 max-w-4xl w-full mx-4 max-h-
[90vh] overflow-y-auto">
            <div className="flex justify-between items-center bg-blue-600 text-
white px-6 py-4 rounded-t-lg">
              <h3 className="text-lg font-medium">Create User Account & Employee
Profile</h3>
              <button
                onClick={() => setShowModal(false)}
                className="text-white hover:text-gray-200 focus:outline-none"
              >
                &times;
              </button>
            </div>
            <form onSubmit={handleCreateUser} className="px-6 py-4">
              {/* Basic User Information */}
              <div className="mb-6">
                <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Basic Information</h4>
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                  <div>
                    <label htmlFor="fullName" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Full Name *
                    </label>
                    <input
                      type="text"
                      id="fullName"
                      name="fullName"
                      value={newUser.fullName}
```

```
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      required
                    />
                  </div>
                  <div>
                    <label htmlFor="email" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                      Email *
                    </label>
                    <input
                      type="email"
                      id="email"
                      name="email"
                      value={newUser.email}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      required
                    />
                  </div>
                  <div>
                    <label htmlFor="password" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Password *
                    </label>
                    <input
                      type="password"
                      id="password"
                      name="password"
                      value={newUser.password}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      required
                      minLength={6}
                    />
                    <p className="text-xs text-gray-500 dark:text-gray-500
mt-1">Password must be at least 6 characters</p>
                  </div>
                  <div>
                    <label htmlFor="role" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                      System Role *
                    </label>
                    <select
                      id="role"
                      name="role"
                      value={newUser.role}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
```

```
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      required
                    >
                      <option value="Sales Person">Sales Person</option>
                      <option value="Lead Person">Lead Person</option>
                      <option value="Manager">Manager</option>
                      <option value="Admin">Admin</option>
                      <option value="HR">HR</option>
                      <option value="Employee">Employee</option>
                    </select>
                  </div>
                </div>
              </div>

              {/* Contact Information */}
              <div className="mb-6">
                <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Contact Information</h4>
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                  <div>
                    <label htmlFor="phoneNumber" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Phone Number
                    </label>
                    <input
                      type="tel"
                      id="phoneNumber"
                      name="phoneNumber"
                      value={newUser.phoneNumber}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                  <div>
                    <label htmlFor="whatsappNumber" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      WhatsApp Number
                    </label>
                    <input
                      type="tel"
                      id="whatsappNumber"
                      name="whatsappNumber"
                      value={newUser.whatsappNumber}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                  <div className="md:col-span-2">
                    <label htmlFor="linkedInUrl" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      LinkedIn URL
                    </label>
```

```
                    <input
                      type="url"
                      id="linkedInUrl"
                      name="linkedInUrl"
                      value={newUser.linkedInUrl}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      placeholder="https://linkedin.com/in/..."
                    />
                  </div>
                </div>
              </div>

              {/* Address Information */}
              <div className="mb-6">
                <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Address Information</h4>
                <div className="grid grid-cols-1 gap-4">
                  <div>
                    <label htmlFor="currentAddress" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Current Address
                    </label>
                    <textarea
                      id="currentAddress"
                      name="currentAddress"
                      value={newUser.currentAddress}
                      onChange={handleInputChange}
                      rows="3"
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                  <div>
                    <label htmlFor="permanentAddress" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                      Permanent Address
                    </label>
                    <textarea
                      id="permanentAddress"
                      name="permanentAddress"
                      value={newUser.permanentAddress}
                      onChange={handleInputChange}
                      rows="3"
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                </div>
              </div>

              {/* Professional Information */}
```

```
              <div className="mb-6">
                <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Professional Information</h4>
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                  <div>
                    <label htmlFor="dateOfBirth" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Date of Birth
                    </label>
                    <input
                      type="date"
                      id="dateOfBirth"
                      name="dateOfBirth"
                      value={newUser.dateOfBirth}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                  <div>
                    <label htmlFor="joiningDate" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Joining Date
                    </label>
                    <input
                      type="date"
                      id="joiningDate"
                      name="joiningDate"
                      value={newUser.joiningDate}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                  <div>
                    <label htmlFor="salary" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                      Salary ( ¹)
                    </label>
                    <input
                      type="number"
                      id="salary"
                      name="salary"
                      value={newUser.salary}
                      onChange={handleInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      placeholder="Enter salary amount"
                    />
                  </div>
                  <div>
                    <label htmlFor="department" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
```

```
                    Department
                  </label>
                  <select
                    id="department"
                    name="department"
                    value={newUser.department}
                    onChange={handleInputChange}
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                  >
                    <option value="">Select Department</option>
                    {departments.map((dept) => (
                      <option key={dept._id} value={dept._id}>
                        {dept.name}
                      </option>
                    ))}
                  </select>
                </div>
                <div>
                  <label htmlFor="employeeRole" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                    Employee Role
                  </label>
                  <select
                    id="employeeRole"
                    name="employeeRole"
                    value={newUser.employeeRole}
                    onChange={handleInputChange}
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                  >
                    <option value="">Select Role</option>
                    {roles.map((role) => (
                      <option key={role._id} value={role._id}>
                        {role.name}
                      </option>
                    ))}
                  </select>
                </div>
                <div>
                  <label htmlFor="status" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                    Status
                  </label>
                  <select
                    id="status"
                    name="status"
                    value={newUser.status}
                    onChange={handleInputChange}
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                  >
                    <option value="ACTIVE">Active</option>
```

```
                    <option value="INACTIVE">Inactive</option>
                    <option value="TERMINATED">Terminated</option>
                  </select>
                </div>
              </div>
            </div>

            {/* Educational Information */}
            <div className="mb-6">
              <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Educational Information</h4>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label htmlFor="collegeName" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                    College Name
                  </label>
                  <input
                    type="text"
                    id="collegeName"
                    name="collegeName"
                    value={newUser.collegeName}
                    onChange={handleInputChange}
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                  />
                </div>
                <div>
                  <label htmlFor="internshipDuration" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                    Internship Duration (months)
                  </label>
                  <input
                    type="number"
                    id="internshipDuration"
                    name="internshipDuration"
                    value={newUser.internshipDuration}
                    onChange={handleInputChange}
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    placeholder="Enter duration in months"
                  />
                </div>
              </div>
            </div>

            {/* Document Upload Section */}
            <div className="mt-6 border-t border-gray-200 dark:border-gray-700
pt-6">
              <h4 className="text-sm font-medium text-gray-900 dark:text-white
mb-4">Document Uploads</h4>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                {/* Photograph */}
                <div>
                  <label className="block text-sm font-medium text-gray-700
```

```
dark:text-gray-400 mb-1">
                      Photograph <span className="text-red-500">*</span>
                    </label>
                    <input
                      type="file"
                      accept="image/*"
                      onChange={(e) => handleDocumentChange('photograph',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                      required
                    />
                  </div>

                  {/* 10th Marksheet */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      10th Marksheet <span className="text-red-500">*</span>
                    </label>
                    <input
                      type="file"
                      accept=".pdf,.jpg,.jpeg,.png"
                      onChange={(e) => handleDocumentChange('tenthMarksheet',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                      required
                    />
                  </div>

                  {/* 12th Marksheet */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      12th Marksheet <span className="text-red-500">*</span>
                    </label>
                    <input
                      type="file"
                      accept=".pdf,.jpg,.jpeg,.png"
                      onChange={(e) => handleDocumentChange('twelfthMarksheet',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                      required
                    />
                  </div>

                  {/* Bachelor Degree */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      Bachelor Degree <span className="text-red-500">*</span>
```

```
                    </label>
                    <input
                      type="file"
                      accept=".pdf,.jpg,.jpeg,.png"
                      onChange={(e) => handleDocumentChange('bachelorDegree',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                      required
                    />
                  </div>

                  {/* Postgraduate Degree */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      Postgraduate Degree
                    </label>
                    <input
                      type="file"
                      accept=".pdf,.jpg,.jpeg,.png"
                      onChange={(e) => handleDocumentChange('postgraduateDegree',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                    />
                  </div>

                  {/* Aadhar Card */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      Aadhar Card <span className="text-red-500">*</span>
                    </label>
                    <input
                      type="file"
                      accept=".pdf,.jpg,.jpeg,.png"
                      onChange={(e) => handleDocumentChange('aadharCard',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                      required
                    />
                  </div>

                  {/* PAN Card */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      PAN Card <span className="text-red-500">*</span>
                    </label>
                    <input
                      type="file"
```

```
                      accept=".pdf,.jpg,.jpeg,.png"
                      onChange={(e) => handleDocumentChange('panCard',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                      required
                    />
                  </div>

                  {/* PCC */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      Police Clearance Certificate
                    </label>
                    <input
                      type="file"
                      accept=".pdf,.jpg,.jpeg,.png"
                      onChange={(e) => handleDocumentChange('pcc',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                    />
                  </div>

                  {/* Resume */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      Resume <span className="text-red-500">*</span>
                    </label>
                    <input
                      type="file"
                      accept=".pdf,.doc,.docx"
                      onChange={(e) => handleDocumentChange('resume',
e.target.files[0])}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                      required
                    />
                  </div>

                  {/* Offer Letter */}
                  <div>
                    <label className="block text-sm font-medium text-gray-700
dark:text-gray-400 mb-1">
                      Offer Letter <span className="text-red-500">*</span>
                    </label>
                    <input
                      type="file"
                      accept=".pdf,.doc,.docx"
                      onChange={(e) => handleDocumentChange('offerLetter',
e.target.files[0])}
```

```
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
file:mr-4 file:py-2 file:px-4 file:rounded-l file:border-0 file:bg-blue-50
file:text-blue-700 hover:file:bg-blue-100"
                    required
                  />
                </div>
              </div>
              <p className="text-xs text-gray-500 dark:text-gray-500 mt-2">
                <span className="text-red-500">*</span> Required documents.
                Accepted formats: PDF, JPG, JPEG, PNG (for images), DOC, DOCX
(for documents)
              </p>
            </div>

            <div className="mt-6 flex justify-end space-x-2">
              <button
                type="button"
                onClick={() => setShowModal(false)}
                className="px-4 py-2 border border-gray-300 dark:border-
slate-600 text-gray-700 dark:text-gray-400 rounded-md hover:bg-gray-50 dark:bg-
slate-800 transition"
              >
                Cancel
              </button>
              <button
                type="submit"
                className="px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-
blue-700 transition"
                disabled={loading}
              >
                {loading ? 'Creating...' : 'Create User & Employee'}
              </button>
            </div>
          </form>
        </div>
      </div>
    )}

    {/* Edit User Modal */}
    {showEditModal && editUser && (
      <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
        <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out rounded-lg shadow-lg dark:shadow-black/25 max-w-4xl w-full mx-4 max-h-
[90vh] overflow-y-auto">
          <div className="flex justify-between items-center bg-blue-600 text-
white px-6 py-4 rounded-t-lg">
            <h3 className="text-lg font-medium">Edit User Account & Employee
Profile</h3>
            <button
              onClick={() => setShowEditModal(false)}
              className="text-white hover:text-gray-200 focus:outline-none"
            >
              &times;
            </button>
          </div>
          <form onSubmit={handleUpdateUser} className="px-6 py-4">
            {/* Basic User Information */}
```

```
              <div className="mb-6">
                <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Basic Information</h4>
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                  <div>
                    <label htmlFor="editFullName" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Full Name *
                    </label>
                    <input
                      type="text"
                      id="editFullName"
                      name="fullName"
                      value={editUser.fullName}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      required
                    />
                  </div>
                  <div>
                    <label htmlFor="editEmail" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Email *
                    </label>
                    <input
                      type="email"
                      id="editEmail"
                      name="email"
                      value={editUser.email}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      required
                    />
                  </div>
                  <div>
                    <label htmlFor="editPassword" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Password
                    </label>
                    <input
                      type="password"
                      id="editPassword"
                      name="password"
                      value={editUser.password}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      placeholder="Leave blank to keep current password"
                      minLength={editUser.password ? 6 : 0}
                    />
                    <p className="text-xs text-gray-500 dark:text-gray-500
```

```
mt-1">Leave blank to keep current password. New password must be at least 6
characters.</p>
                  </div>
                  <div>
                    <label htmlFor="editRole" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      System Role *
                    </label>
                    <select
                      id="editRole"
                      name="role"
                      value={editUser.role}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      required
                    >
                      <option value="Sales Person">Sales Person</option>
                      <option value="Lead Person">Lead Person</option>
                      <option value="Manager">Manager</option>
                      <option value="Admin">Admin</option>
                      <option value="HR">HR</option>
                      <option value="Employee">Employee</option>
                    </select>
                  </div>
                </div>
              </div>

              {/* Contact Information */}
              <div className="mb-6">
                <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Contact Information</h4>
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                  <div>
                    <label htmlFor="editPhoneNumber" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                      Phone Number
                    </label>
                    <input
                      type="tel"
                      id="editPhoneNumber"
                      name="phoneNumber"
                      value={editUser.phoneNumber}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                  <div>
                    <label htmlFor="editWhatsappNumber" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                      WhatsApp Number
                    </label>
                    <input
                      type="tel"
```

```
                      id="editWhatsappNumber"
                      name="whatsappNumber"
                      value={editUser.whatsappNumber}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                  <div className="md:col-span-2">
                    <label htmlFor="editLinkedInUrl" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                      LinkedIn URL
                    </label>
                    <input
                      type="url"
                      id="editLinkedInUrl"
                      name="linkedInUrl"
                      value={editUser.linkedInUrl}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      placeholder="https://linkedin.com/in/..."
                    />
                  </div>
                </div>
              </div>

              {/* Address Information */}
              <div className="mb-6">
                <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Address Information</h4>
                <div className="grid grid-cols-1 gap-4">
                  <div>
                    <label htmlFor="editCurrentAddress" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                      Current Address
                    </label>
                    <textarea
                      id="editCurrentAddress"
                      name="currentAddress"
                      value={editUser.currentAddress}
                      onChange={handleEditInputChange}
                      rows="3"
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    />
                  </div>
                  <div>
                    <label htmlFor="editPermanentAddress" className="block text-
sm font-medium text-gray-700 dark:text-gray-400 mb-1">
                      Permanent Address
                    </label>
                    <textarea
```

```
                    id="editPermanentAddress"
                    name="permanentAddress"
                    value={editUser.permanentAddress}
                    onChange={handleEditInputChange}
                    rows="3"
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                  />
                </div>
              </div>
            </div>

            {/* Professional Information */}
            <div className="mb-6">
              <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Professional Information</h4>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label htmlFor="editDateOfBirth" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                    Date of Birth
                  </label>
                  <input
                    type="date"
                    id="editDateOfBirth"
                    name="dateOfBirth"
                    value={editUser.dateOfBirth}
                    onChange={handleEditInputChange}
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                  />
                </div>
                <div>
                  <label htmlFor="editJoiningDate" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                    Joining Date
                  </label>
                  <input
                    type="date"
                    id="editJoiningDate"
                    name="joiningDate"
                    value={editUser.joiningDate}
                    onChange={handleEditInputChange}
                    className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                  />
                </div>
                <div>
                  <label htmlFor="editSalary" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                    Salary ( ¹)
                  </label>
                  <input
```

```
                      type="number"
                      id="editSalary"
                      name="salary"
                      value={editUser.salary}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      placeholder="Enter salary amount"
                    />
                  </div>
                  <div>
                    <label htmlFor="editDepartment" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Department
                    </label>
                    <select
                      id="editDepartment"
                      name="department"
                      value={editUser.department}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    >
                      <option value="">Select Department</option>
                      {departments.map((dept) => (
                        <option key={dept._id} value={dept._id}>
                          {dept.name}
                        </option>
                      ))}
                    </select>
                  </div>
                  <div>
                    <label htmlFor="editEmployeeRole" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                      Employee Role
                    </label>
                    <select
                      id="editEmployeeRole"
                      name="employeeRole"
                      value={editUser.employeeRole}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    >
                      <option value="">Select Role</option>
                      {roles.map((role) => (
                        <option key={role._id} value={role._id}>
                          {role.name}
                        </option>
                      ))}
                    </select>
                  </div>
                  <div>
```

```
                    <label htmlFor="editStatus" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Status
                    </label>
                    <select
                      id="editStatus"
                      name="status"
                      value={editUser.status}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                    >
                      <option value="ACTIVE">Active</option>
                      <option value="INACTIVE">Inactive</option>
                      <option value="TERMINATED">Terminated</option>
                    </select>
                  </div>
                </div>
              </div>

              {/* Educational Information */}
              <div className="mb-6">
                <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Educational Information</h4>
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                  <div>
                    <label htmlFor="editCollegeName" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">
                      College Name
                    </label>
                    <input
                      type="text"
                      id="editCollegeName"
                      name="collegeName"
                      value={editUser.collegeName}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
                      placeholder="Enter college name"
                    />
                  </div>
                  <div>
                    <label htmlFor="editInternshipDuration" className="block text-
sm font-medium text-gray-700 dark:text-gray-400 mb-1">
                      Internship Duration (months)
                    </label>
                    <input
                      type="number"
                      id="editInternshipDuration"
                      name="internshipDuration"
                      value={editUser.internshipDuration}
                      onChange={handleEditInputChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100
focus:outline-none focus:ring-blue-500 dark:focus:border-blue-400 focus:border-
blue-500"
```

```
                    placeholder="Enter duration in months"
                  />
                </div>
              </div>
            </div>

            {/* Document Upload Section */}
            <div className="mb-6">
              <h4 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Update Documents (Optional)</h4>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                {Object.keys(editDocuments).map((docType) => (
                  <div key={docType} className="flex items-center space-x-2">
                    <input
                      type="file"
                      id={`edit-${docType}`}
                      onChange={(e) => handleEditDocumentChange(docType,
e.target.files[0])}
                      accept=".pdf,.jpg,.jpeg,.png"
                      className="hidden"
                    />
                    <label
                      htmlFor={`edit-${docType}`}
                      className="flex-1 px-3 py-2 bg-gray-100 dark:bg-slate-700
border border-gray-300 dark:border-gray-600 rounded text-sm cursor-pointer
hover:bg-gray-200 dark:hover:bg-slate-600 transition"
                    >
                      <span className="text-gray-600 dark:text-gray-400">
                        {docType.charAt(0).toUpperCase() +
docType.slice(1).replace(/([A-Z])/g, ' $1')}:
                      </span>
                      <span className="ml-2 text-gray-800 dark:text-gray-200">
                        {editDocuments[docType] ? editDocuments[docType].name :
'Choose file'}
                      </span>
                    </label>
                  </div>
                ))}
              </div>
            </div>

            <div className="mt-6 flex justify-end space-x-2">
              <button
                type="button"
                onClick={() => setShowEditModal(false)}
                className="px-4 py-2 border border-gray-300 dark:border-
slate-600 text-gray-700 dark:text-gray-400 rounded-md hover:bg-gray-50 dark:bg-
slate-800 transition"
              >
                Cancel
              </button>
              <button
                type="submit"
                className="px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-
blue-700 transition"
                disabled={loading}
              >
                {loading ? 'Updating...' : 'Update User & Employee'}
              </button>
```

```
              </div>
            </form>
          </div>
        </div>
      )}

      {/* Delete Confirmation Modal */}
      {confirmDelete && (
        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
          <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out rounded-lg shadow-lg dark:shadow-black/25 max-w-md w-full p-6">
            <h3 className="text-lg font-medium mb-4">Confirm Delete</h3>
            <p className="mb-4">Are you sure you want to delete user <span
className="font-semibold">{confirmDelete.fullName}</span>? This action cannot be
undone.</p>
            <div className="flex justify-end space-x-2">
              <button
                onClick={() => setConfirmDelete(null)}
                className="px-4 py-2 border border-gray-300 dark:border-slate-600
text-gray-700 dark:text-gray-400 rounded-md hover:bg-gray-50 dark:bg-slate-800
transition"
              >
                Cancel
              </button>
              <button
                onClick={() => handleDeleteUser(confirmDelete._id)}
                className="px-4 py-2 bg-red-600 text-white rounded-md hover:bg-
red-700 transition"
              >
                Delete
              </button>
            </div>
          </div>
        </div>
      )}

      {/* Employee Edit Modal */}
      {showEmployeeEditModal && editEmployeeId && (
        <EditEmployeeDialog
          employeeId={editEmployeeId}
          onClose={() => setShowEmployeeEditModal(false)}
          onUpdate={handleEmployeeUpdated}
          departments={departments}
          roles={roles}
        />
      )}
    </Layout>
  );
};

export default AdminUsersPage;
```

src/pages/Countries.jsx

```
import React, { useState } from "react";
import { Card, CardContent } from "../components/ui/card";
import { CheckCircle, Ban } from "lucide-react";
```

```
const countries = [
  { name: "United States", code: "US", status: "allowed" },
  { name: "United Kingdom", code: "UK", status: "allowed" },
  { name: "France", code: "FR", status: "allowed" },
  { name: "Italy", code: "IT", status: "allowed" },
  { name: "Spain", code: "ES", status: "allowed" },
  { name: "Portugal", code: "PT", status: "allowed" },
  { name: "Greece", code: "GR", status: "allowed" },
  { name: "Brazil", code: "BR", status: "allowed" },
  { name: "Argentina", code: "AR", status: "allowed" },
  { name: "Chile", code: "CL", status: "allowed" },
  { name: "Mexico", code: "MX", status: "allowed" },
  { name: "Colombia", code: "CO", status: "allowed" },
  { name: "Peru", code: "PE", status: "allowed" },
  { name: "Venezuela", code: "VE", status: "allowed" },
  { name: "Ecuador", code: "EC", status: "allowed" },
  { name: "Bolivia", code: "BO", status: "allowed" },
  { name: "Paraguay", code: "PY", status: "allowed" },
  { name: "Uruguay", code: "UY", status: "allowed" },
  { name: "Saudi Arabia", code: "SA", status: "allowed" },
  { name: "United Arab Emirates", code: "AE", status: "allowed" },
  { name: "Qatar", code: "QA", status: "allowed" },
  { name: "Kuwait", code: "KW", status: "allowed" },
  { name: "Bahrain", code: "BH", status: "allowed" },
  { name: "Oman", code: "OM", status: "allowed" },
  { name: "Jordan", code: "JO", status: "allowed" },
  { name: "Canada", code: "CA", status: "allowed" },
  { name: "Ukraine", code: "UA", status: "allowed" },
  { name: "Switzerland", code: "CH", status: "allowed" },
  { name: "Singapore", code: "SG", status: "allowed" },
  { name: "Romania", code: "RO", status: "allowed" },
  { name: "Netherlands", code: "NL", status: "allowed" },
  { name: "New Zealand", code: "NZ", status: "allowed" },
  { name: "Pakistan", code: "PK", status: "blocked" },
  { name: "Sri Lanka", code: "LK", status: "blocked" },
  { name: "Nepal", code: "NP", status: "blocked" },
  // { name: "Africa", code: "AF", status: "blocked" },
  { name: "China", code: "CN", status: "blocked" },
  { name: "Hong Kong", code: "HK", status: "blocked" },
  { name: "Russia", code: "RU", status: "blocked" },
  { name: "Philippines", code: "PH", status: "blocked" },
  { name: "Japan", code: "JP", status: "blocked" },
  { name: "Vietnam", code: "VN", status: "blocked" },
  { name: "Bangladesh", code: "BD", status: "blocked" },
  { name: "Afghanistan", code: "AF", status: "blocked" },
  { name: "Tajikistan", code: "TJ", status: "blocked" },
  { name: "Kazakhstan", code: "KZ", status: "blocked" },
  { name: "Kyrgyzstan", code: "KG", status: "blocked" },
  { name: "Turkmenistan", code: "TM", status: "blocked" },
  { name: "Papua New Guinea", code: "PG", status: "blocked" },
  { name: "Myanmar", code: "MM", status: "blocked" },
  { name: "Indonesia", code: "ID", status: "blocked" },
  { name: "Iran", code: "IR", status: "blocked" },
  { name: "Iraq", code: "IQ", status: "blocked" },
  { name: "Israel", code: "IL", status: "blocked" },
  { name: "Lebanon", code: "LB", status: "blocked" },
  { name: "Palestine", code: "PS", status: "blocked" },
  { name: "Syria", code: "SY", status: "blocked" },
```

```
  { name: "Turkey", code: "TR", status: "blocked" },
  { name: "Yemen", code: "YE", status: "blocked" },
  { name: "Zimbabwe", code: "ZW", status: "blocked" },

  // Add more as needed
];

const LeadPermissionPage = () => {
  const [filter, setFilter] = useState("all");

  const filteredCountries = countries.filter((c) =>
    filter === "all" ? true : c.status === filter
  );

  return (
    <div className="p-6 space-y-6">
      <h1 className="text-3xl font-bold">Lead Geo-Permission Matrix Ø<ß </h1>

      <div className="flex gap-4">
        <button
          className={`px-4 py-2 rounded-xl ${filter === "all" ? "bg-blue-600 text-
white" : "bg-gray-200 dark:bg-slate-600"}`}
          onClick={() => setFilter("all")}
        >
          All
        </button>
        <button
          className={`px-4 py-2 rounded-xl ${filter === "allowed" ? "bg-green-600
text-white" : "bg-gray-200 dark:bg-slate-600"}`}
          onClick={() => setFilter("allowed")}
        >
          Allowed '
        </button>
        <button
          className={`px-4 py-2 rounded-xl ${filter === "blocked" ? "bg-red-600
text-white" : "bg-gray-200 dark:bg-slate-600"}`}
          onClick={() => setFilter("blocked")}
        >
          Blocked 'L
        </button>
      </div>

      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
        {filteredCountries.map((country) => (
          <Card key={country.code} className="rounded-2xl shadow-md dark:shadow-
black/25">
            <CardContent className="p-4 flex items-center justify-between">
              <div>
                <h2 className="text-xl font-semibold">{country.name}</h2>
                <p className="text-sm text-gray-500 dark:text-gray-500">Code:
{country.code}</p>
              </div>
              {country.status === "allowed" ? (
                <CheckCircle className="text-green-500 w-6 h-6" />
              ) : (
                <Ban className="text-red-500 w-6 h-6" />
              )}
            </CardContent>
          </Card>
```

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

export default LeadPermissionPage;
```

src/pages/CustomerDashboard.jsx

```
import React, { useState, useEffect } from 'react';
import Layout from '../components/Layout/Layout';
import { useAuth } from '../context/AuthContext';
import { useChat } from '../context/ChatContext';
import { FaComments, FaUser, FaEnvelope, FaPhone, FaCalendar } from 'react-icons/
fa';

const CustomerDashboard = () => {
  const { user } = useAuth();
  const { setIsChatOpen, onlineUsers, unreadCounts } = useChat();
  const [supportTeam, setSupportTeam] = useState([]);

  useEffect(() => {
    // Filter support team members (non-customers)
    const team = onlineUsers.filter(u => u.role !== 'Customer');
    setSupportTeam(team);
  }, [onlineUsers]);

  const totalUnreadMessages = Object.values(unreadCounts).reduce((sum, count) =>
sum + count, 0);

  const handleStartChat = () => {
    setIsChatOpen(true);
  };

  return (
    <Layout>
      <div className="min-h-screen bg-gray-50 dark:bg-slate-800 py-8">
        <div className="container mx-auto px-4">
          {/* Welcome Header */}
          <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out rounded-lg shadow-md dark:shadow-black/25 p-6 mb-6">
            <div className="flex items-center justify-between">
              <div>
                <h1 className="text-3xl font-bold text-gray-800 dark:text-
gray-200">
                  Welcome, {user?.fullName}!
                </h1>
                <p className="text-gray-600 dark:text-gray-500 mt-2">
                  Your customer portal - Chat with our team and get support
                </p>
              </div>
              <div className="text-right">
                <div className="text-sm text-gray-500 dark:text-
gray-500">Customer ID</div>
                <div className="font-mono text-sm">{user?._id?.slice(-8)}</div>
              </div>
            </div>
```

```
          </div>

          <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
            {/* Chat Section */}
            <div className="lg:col-span-2">
              <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow-md dark:shadow-black/25 p-6">
                <div className="flex items-center justify-between mb-4">
                  <h2 className="text-xl font-semibold text-gray-800 dark:text-
gray-200 flex items-center">
                    <FaComments className="mr-2 text-blue-600" />
                    Chat with Support Team
                  </h2>
                  {totalUnreadMessages > 0 && (
                    <span className="bg-red-500 text-white px-2 py-1 rounded-full
text-sm">
                      {totalUnreadMessages} new messages
                    </span>
                  )}
                </div>

                <div className="text-center py-8">
                  <div className="mb-4">
                    <FaComments className="mx-auto text-6xl text-blue-200" />
                  </div>
                  <h3 className="text-lg font-medium text-gray-700 dark:text-
gray-400 mb-2">
                    Need Help? Start a Conversation
                  </h3>
                  <p className="text-gray-500 dark:text-gray-500 mb-4">
                    Our support team is here to help you. Click below to start
chatting.
                  </p>
                  <button
                    onClick={handleStartChat}
                    className="bg-blue-600 hover:bg-blue-700 text-white px-6 py-3
rounded-lg font-medium transition-colors"
                  >
                    Open Chat
                  </button>
                </div>

                {/* Support Team Status */}
                {supportTeam.length > 0 && (
                  <div className="mt-6 border-t pt-4">
                    <h4 className="text-sm font-medium text-gray-700 dark:text-
gray-400 mb-3">
                      Support Team Online ({supportTeam.length})
                    </h4>
                    <div className="flex flex-wrap gap-2">
                      {supportTeam.slice(0, 5).map(member => (
                        <div key={member._id} className="flex items-center bg-
green-50 px-3 py-1 rounded-full">
                          <div className="w-2 h-2 bg-green-500 rounded-full
mr-2"></div>
                          <span className="text-sm text-
green-700">{member.fullName}</span>
                          <span className="text-xs text-green-600
ml-1">({member.role})</span>
```

```
                      </div>
                    ))}
                    {supportTeam.length > 5 && (
                      <div className="flex items-center bg-gray-50 dark:bg-
slate-800 px-3 py-1 rounded-full">
                        <span className="text-sm text-gray-600 dark:text-
gray-500">+{supportTeam.length - 5} more</span>
                      </div>
                    )}
                  </div>
                </div>
              )}
            </div>
          </div>

          {/* Customer Info Sidebar */}
          <div className="space-y-6">
            {/* Account Information */}
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow-md dark:shadow-black/25 p-6">
              <h3 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4 flex items-center">
                <FaUser className="mr-2 text-blue-600" />
                Account Information
              </h3>
              <div className="space-y-3">
                <div className="flex items-center">
                  <FaUser className="text-gray-400 dark:text-gray-400 mr-3" />
                  <div>
                    <div className="text-sm text-gray-500 dark:text-
gray-500">Full Name</div>
                    <div className="font-medium">{user?.fullName}</div>
                  </div>
                </div>
                <div className="flex items-center">
                  <FaEnvelope className="text-gray-400 dark:text-gray-400
mr-3" />
                  <div>
                    <div className="text-sm text-gray-500 dark:text-
gray-500">Email</div>
                    <div className="font-medium">{user?.email}</div>
                  </div>
                </div>
                <div className="flex items-center">
                  <FaCalendar className="text-gray-400 dark:text-gray-400
mr-3" />
                  <div>
                    <div className="text-sm text-gray-500 dark:text-
gray-500">Member Since</div>
                    <div className="font-medium">
                      {user?.createdAt ? new
Date(user.createdAt).toLocaleDateString() : 'N/A'}
                    </div>
                  </div>
                </div>
              </div>
            </div>

            {/* Quick Actions */}
```

```
              <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-lg shadow-md dark:shadow-black/25 p-6">
                <h3 className="text-lg font-semibold text-gray-800 dark:text-
gray-200 mb-4">Quick Actions</h3>
                <div className="space-y-3">
                  <button
                    onClick={handleStartChat}
                    className="w-full bg-blue-50 hover:bg-blue-100 text-blue-700
px-4 py-3 rounded-lg text-left transition-colors"
                  >
                    <FaComments className="inline mr-2" />
                    Start New Chat
                  </button>
                  <button
                    onClick={() => window.location.href = '/profile'}
                    className="w-full bg-gray-50 dark:bg-slate-700 hover:bg-
gray-100 text-gray-700 dark:text-gray-400 px-4 py-3 rounded-lg text-left
transition-colors"
                  >
                    <FaUser className="inline mr-2" />
                    Edit Profile
                  </button>
                </div>
              </div>

              {/* Support Information */}
              <div className="bg-blue-50 rounded-lg p-6">
                <h3 className="text-lg font-semibold text-blue-800 mb-2">Need
Help?</h3>
                <p className="text-blue-700 text-sm mb-3">
                  Our support team is available to help you with any questions or
issues.
                </p>
                <div className="text-sm text-blue-600">
                  <div>Ø=Üç Email: support@traincapetech.in</div>
                  <div>Ø=ÝR Hours: 9 AM - 6 PM (Mon-Fri)</div>
                  <div>Ø=Ü¬ Live Chat: Available now</div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </Layout>
  );
};

export default CustomerDashboard;
```

src/pages/DocumentationPage.jsx

```
import React, { useState } from 'react';
import { useAuth } from '../context/AuthContext';
import api from '../services/api';
import { toast } from 'react-hot-toast';
import { FaDownload, FaSpinner } from 'react-icons/fa';

const DocumentationPage = () => {
```

```
  const { user } = useAuth();
  const [loading, setLoading] = useState(false);

  const handleDownload = async () => {
    try {
      setLoading(true);
      const response = await api.get('/api/documentation/project', {
        responseType: 'blob'
      });

      // Create blob link to download
      const url = window.URL.createObjectURL(new Blob([response.data]));
      const link = document.createElement('a');
      link.href = url;
      link.setAttribute('download', 'CRM_Project_Documentation.pdf');

      // Append to html link element page
      document.body.appendChild(link);

      // Start download
      link.click();

      // Clean up and remove the link
      link.parentNode.removeChild(link);
      toast.success('Documentation downloaded successfully!');
    } catch (error) {
      console.error('Download error:', error);
      toast.error('Failed to download documentation');
    } finally {
      setLoading(false);
    }
  };

  if (!['Admin', 'HR', 'Manager'].includes(user?.role)) {
    return (
      <div className="flex items-center justify-center min-h-screen bg-gray-100">
        <div className="text-center text-red-600">
          Not authorized to access documentation.
        </div>
      </div>
    );
  }

  return (
    <div className="min-h-screen bg-gray-100 py-8 px-4">
      <div className="max-w-4xl mx-auto">
        <div className="bg-white rounded-lg shadow-md p-6">
          <h1 className="text-2xl font-bold text-gray-800 mb-6">
            Project Documentation
          </h1>

          <div className="mb-8">
            <h2 className="text-xl font-semibold text-gray-700 mb-4">
              CRM System Documentation
            </h2>
            <p className="text-gray-600 mb-4">
              This documentation provides a comprehensive overview of the CRM
system including:
            </p>
```

```
            <ul className="list-disc list-inside text-gray-600 space-y-2 mb-6">
              <li>Authentication System</li>
              <li>Employee Management</li>
              <li>Payroll System</li>
              <li>Chat System</li>
              <li>Technical Stack</li>
              <li>API Endpoints</li>
              <li>Future Enhancements</li>
            </ul>
          </div>

          <button
            onClick={handleDownload}
            disabled={loading}
            className={`flex items-center justify-center w-full md:w-auto px-6
py-3 bg-blue-600 text-white rounded-lg hover:bg-blue-700 transition-colors ${
              loading ? 'opacity-75 cursor-not-allowed' : ''
            }`}
          >
            {loading ? (
              <FaSpinner className="animate-spin mr-2" />
            ) : (
              <FaDownload className="mr-2" />
            )}
            {loading ? 'Downloading...' : 'Download Documentation'}
          </button>
        </div>
      </div>
    </div>
  );
};

export default DocumentationPage;
```

src/pages/DocumentManagementPage.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../context/AuthContext';
import Layout from '../components/Layout/Layout';
import { authAPI } from '../services/api';
import { toast } from 'react-toastify';
import {
  FaDownload,
  FaEye,
  FaUpload,
  FaFileAlt,
  FaImage,
  FaUser,
  FaSearch,
  FaFilter,
  FaCheck,
  FaTimes,
  FaClock
} from 'react-icons/fa';

const DocumentManagementPage = () => {
  const { user } = useAuth();
  const [employees, setEmployees] = useState([]);
```

```
  const [filteredEmployees, setFilteredEmployees] = useState([]);
  const [selectedEmployee, setSelectedEmployee] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [searchTerm, setSearchTerm] = useState('');
  const [departmentFilter, setDepartmentFilter] = useState('');
  const [documentFilter, setDocumentFilter] = useState('');
  const [uploadingDocument, setUploadingDocument] = useState(null);
  const [showUploadModal, setShowUploadModal] = useState(false);
  const [uploadFile, setUploadFile] = useState(null);
  const [uploadDocumentType, setUploadDocumentType] = useState('');

  // Document types with their display names and requirements
  const documentTypes = [
    { key: 'photograph', label: 'Photograph', required: true, icon: FaImage,
accept: 'image/*' },
    { key: 'tenthMarksheet', label: '10th Marksheet', required: true, icon:
FaFileAlt, accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'twelfthMarksheet', label: '12th Marksheet', required: true, icon:
FaFileAlt, accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'bachelorDegree', label: 'Bachelor Degree', required: true, icon:
FaFileAlt, accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'postgraduateDegree', label: 'Postgraduate Degree', required: false,
icon: FaFileAlt, accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'aadharCard', label: 'Aadhar Card', required: true, icon: FaFileAlt,
accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'panCard', label: 'PAN Card', required: true, icon: FaFileAlt, accept:
'.pdf,.jpg,.jpeg,.png' },
    { key: 'pcc', label: 'Police Clearance Certificate', required: false, icon:
FaFileAlt, accept: '.pdf,.jpg,.jpeg,.png' },
    { key: 'resume', label: 'Resume', required: true, icon: FaFileAlt, accept:
'.pdf,.doc,.docx' },
    { key: 'offerLetter', label: 'Offer Letter', required: true, icon: FaFileAlt,
accept: '.pdf,.doc,.docx' },
  ];

  // Fetch employees data
  useEffect(() => {
    fetchEmployees();
  }, []);

  // Apply filters
  useEffect(() => {
    applyFilters();
  }, [employees, searchTerm, departmentFilter, documentFilter]);

  const fetchEmployees = async () => {
    try {
      setLoading(true);

      if (user?.role === 'Admin' || user?.role === 'Manager') {
        // Fetch all employees for admin/manager
        const response = await authAPI.getAllUsers();
        if (response.data && response.data.success) {
          setEmployees(response.data.data);
        }
      } else {
        // For regular employees, just show their own data
        const currentUserData = await authAPI.getMe();
```

```
        if (currentUserData.data && currentUserData.data.success) {
          setEmployees([currentUserData.data.data]);
          setSelectedEmployee(currentUserData.data.data);
        }
      }
    } catch (err) {
      console.error('Error fetching employees:', err);
      setError('Failed to load employee data');
      toast.error('Failed to load employee data');
    } finally {
      setLoading(false);
    }
  };

  const applyFilters = () => {
    let filtered = [...employees];

    // Search filter
    if (searchTerm) {
      const term = searchTerm.toLowerCase();
      filtered = filtered.filter(emp =>
        (emp.fullName && emp.fullName.toLowerCase().includes(term)) ||
        (emp.email && emp.email.toLowerCase().includes(term)) ||
        (emp.employeeId && emp.employeeId.toLowerCase().includes(term)) ||
        (emp.department && emp.department.toLowerCase().includes(term))
      );
    }

    // Department filter
    if (departmentFilter) {
      filtered = filtered.filter(emp => emp.department === departmentFilter);
    }

    // Document completeness filter
    if (documentFilter) {
      filtered = filtered.filter(emp => {
        const completionStatus = getDocumentCompletionStatus(emp);
        return completionStatus.status === documentFilter;
      });
    }

    setFilteredEmployees(filtered);
  };

  const getDocumentCompletionStatus = (employee) => {
    const documents = employee.documents || {};
    const requiredDocs = documentTypes.filter(doc => doc.required);
    const uploadedRequired = requiredDocs.filter(doc => documents[doc.key] &&
documents[doc.key].filename).length;
    const totalRequired = requiredDocs.length;
    const totalUploaded = documentTypes.filter(doc => documents[doc.key] &&
documents[doc.key].filename).length;

    const percentage = Math.round((uploadedRequired / totalRequired) * 100);

    let status = 'incomplete';
    if (uploadedRequired === totalRequired) {
      status = 'complete';
    } else if (uploadedRequired > 0) {
```

```
    status = 'partial';
  }

  return {
    status,
    percentage,
    uploadedRequired,
    totalRequired,
    totalUploaded,
    totalDocuments: documentTypes.length
  };
};

const getDocumentStatusIcon = (employee, docType) => {
  const documents = employee.documents || {};
  const hasDocument = documents[docType.key] && documents[docType.key].filename;

  if (hasDocument) {
    return <FaCheck className="text-green-500" />;
  } else if (docType.required) {
    return <FaTimes className="text-red-500" />;
  } else {
    return <FaClock className="text-yellow-500" />;
  }
};

const downloadDocument = async (employee, documentType) => {
  try {
    const documents = employee.documents || {};
    const docInfo = documents[documentType.key];

    if (!docInfo || !docInfo.filename) {
      toast.error('Document not found');
      return;
    }

    // Use the API endpoint for downloading documents
    const baseUrl = import.meta.env.VITE_API_URL || 'http://localhost:8080';
    const fileUrl = `${baseUrl}/api/auth/documents/${docInfo.filename}`;

    // Get the token from localStorage
    const token = localStorage.getItem('token');

    // Fetch the file with authentication
    const response = await fetch(fileUrl, {
      headers: {
        'Authorization': `Bearer ${token}`
      }
    });

    if (!response.ok) {
      throw new Error('Failed to download document');
    }

    // Get the file blob
    const blob = await response.blob();

    // Create download link
    const link = document.createElement('a');
```

```
      link.href = window.URL.createObjectURL(blob);
      link.download = `${employee.fullName || employee.email}
_${documentType.label}_${docInfo.originalName || docInfo.filename}`;
      document.body.appendChild(link);
      link.click();
      document.body.removeChild(link);

      // Clean up the blob URL
      window.URL.revokeObjectURL(link.href);

      toast.success(`Downloaded ${documentType.label}`);
    } catch (err) {
      console.error('Error downloading document:', err);
      toast.error('Failed to download document');
    }
  };

  const viewDocument = (employee, documentType) => {
    try {
      const documents = employee.documents || {};
      const docInfo = documents[documentType.key];

      if (!docInfo || !docInfo.filename) {
        toast.error('Document not found');
        return;
      }

      // Use the API endpoint for viewing documents
      const baseUrl = import.meta.env.VITE_API_URL || 'http://localhost:8080';
      const fileUrl = `${baseUrl}/api/auth/documents/${docInfo.filename}`;
      const token = localStorage.getItem('token');

      // Create a new window/tab with the authenticated URL
      const newWindow = window.open();

      // Fetch the file with authentication
      fetch(fileUrl, {
        headers: {
          'Authorization': `Bearer ${token}`
        }
      })
      .then(response => {
        if (!response.ok) {
          throw new Error('Failed to load document');
        }
        return response.blob();
      })
      .then(blob => {
        const blobUrl = window.URL.createObjectURL(blob);
        newWindow.location.href = blobUrl;
      })
      .catch(err => {
        console.error('Error viewing document:', err);
        toast.error('Failed to view document');
        newWindow.close();
      });
    } catch (err) {
      console.error('Error viewing document:', err);
      toast.error('Failed to view document');
```

```
    }
  };

  const handleUploadDocument = async () => {
    if (!uploadFile || !uploadDocumentType || !selectedEmployee) {
      toast.error('Please select a file and document type');
      return;
    }

    try {
      setUploadingDocument(uploadDocumentType);

      const formData = new FormData();
      formData.append(uploadDocumentType, uploadFile);

      // Add other required fields for the update
      formData.append('fullName', selectedEmployee.fullName);
      formData.append('email', selectedEmployee.email);
      formData.append('role', selectedEmployee.role);

      const response = await
authAPI.updateUserWithDocuments(selectedEmployee._id, formData);

      if (response.data && response.data.success) {
        toast.success('Document uploaded successfully');
        setShowUploadModal(false);
        setUploadFile(null);
        setUploadDocumentType('');
        fetchEmployees(); // Refresh data
      }
    } catch (err) {
      console.error('Error uploading document:', err);
      toast.error('Failed to upload document');
    } finally {
      setUploadingDocument(null);
    }
  };

  const openUploadModal = (employee, docType) => {
    setSelectedEmployee(employee);
    setUploadDocumentType(docType);
    setShowUploadModal(true);
  };

  const getDepartments = () => {
    const departments = [...new Set(employees.map(emp =>
emp.department).filter(Boolean))];
    return departments;
  };

  if (loading) {
    return (
      <Layout>
        <div className="flex items-center justify-center min-h-screen">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-
blue-500"></div>
          <span className="ml-3 text-lg">Loading documents...</span>
        </div>
      </Layout>
```

```
    );
  }

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-3xl font-bold text-gray-900 dark:text-white">
            Document Management
          </h1>
          {user?.role !== 'Admin' && user?.role !== 'Manager' && (
            <button
              onClick={() => openUploadModal(selectedEmployee, '')}
              className="bg-blue-600 hover:bg-blue-700 text-white px-4 py-2
rounded-md flex items-center"
            >
              <FaUpload className="mr-2" />
              Upload Document
            </button>
          )}
        </div>

        {error && (
          <div className="mb-4 p-4 bg-red-50 border border-red-200 rounded-md">
            <p className="text-red-700">{error}</p>
          </div>
        )}

        {/* Filters - Only show for Admin/Manager */}
        {(user?.role === 'Admin' || user?.role === 'Manager') && (
          <div className="bg-white dark:bg-slate-900 p-4 rounded-lg shadow-md
mb-6">
            <h3 className="text-lg font-semibold mb-4 flex items-center">
              <FaFilter className="mr-2" />
              Filters
            </h3>
            <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
              {/* Search */}
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Search Employees
                </label>
                <div className="relative">
                  <FaSearch className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-400" />
                  <input
                    type="text"
                    value={searchTerm}
                    onChange={(e) => setSearchTerm(e.target.value)}
                    placeholder="Name, email, employee ID..."
                    className="w-full pl-10 pr-4 py-2 border border-gray-300
dark:border-gray-600 rounded-md bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                  />
                </div>
              </div>

              {/* Department Filter */}
```

```
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Department
                </label>
                <select
                  value={departmentFilter}
                  onChange={(e) => setDepartmentFilter(e.target.value)}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
                >
                  <option value="">All Departments</option>
                  {getDepartments().map(dept => (
                    <option key={dept} value={dept}>{dept}</option>
                  ))}
                </select>
              </div>

              {/* Document Status Filter */}
              <div>
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-1">
                  Document Status
                </label>
                <select
                  value={documentFilter}
                  onChange={(e) => setDocumentFilter(e.target.value)}
                  className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
                >
                  <option value="">All Statuses</option>
                  <option value="complete">Complete</option>
                  <option value="partial">Partial</option>
                  <option value="incomplete">Incomplete</option>
                </select>
              </div>
            </div>
          </div>
        )}

        {/* Employee List - Only show for Admin/Manager */}
        {(user?.role === 'Admin' || user?.role === 'Manager') && (
          <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
            <div className="lg:col-span-1">
              <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md">
                <div className="p-4 border-b border-gray-200 dark:border-
gray-700">
                  <h3 className="text-lg font-semibold flex items-center">
                    <FaUser className="mr-2" />
                    Employees ({filteredEmployees.length})
                  </h3>
                </div>
                <div className="max-h-96 overflow-y-auto">
                  {filteredEmployees.map(employee => {
                    const completionStatus =
getDocumentCompletionStatus(employee);
                    return (
                      <div
                        key={employee._id}
```

```
                        onClick={() => setSelectedEmployee(employee)}
                        className={`p-4 border-b border-gray-200 dark:border-
gray-700 cursor-pointer hover:bg-gray-50 dark:hover:bg-slate-800 ${
                          selectedEmployee?._id === employee._id ? 'bg-blue-50
dark:bg-blue-900/20' : ''
                        }`}
                      >
                        <div className="flex items-center justify-between">
                          <div>
                            <h4 className="font-medium text-gray-900 dark:text-
white">
                              {employee.fullName || employee.email}
                            </h4>
                            <p className="text-sm text-gray-500 dark:text-
gray-400">
                              {employee.department} • {employee.role}
                            </p>
                            <p className="text-xs text-gray-400 dark:text-
gray-500">
                              ID: {employee.employeeId || 'N/A'}
                            </p>
                          </div>
                          <div className="text-right">
                            <div className={`text-sm font-medium ${
                              completionStatus.status === 'complete' ? 'text-
green-600' :
                              completionStatus.status === 'partial' ? 'text-
yellow-600' : 'text-red-600'
                            }`}>
                              {completionStatus.percentage}%
                            </div>
                            <div className="text-xs text-gray-500 dark:text-
gray-400">
                              {completionStatus.uploadedRequired}/
{completionStatus.totalRequired} required
                            </div>
                          </div>
                        </div>
                      </div>
                    );
                  })}
                </div>
              </div>
            </div>

            <div className="lg:col-span-2">
              {selectedEmployee ? (
                <DocumentDetails
                  employee={selectedEmployee}
                  documentTypes={documentTypes}
                  getDocumentStatusIcon={getDocumentStatusIcon}
                  downloadDocument={downloadDocument}
                  viewDocument={viewDocument}
                  openUploadModal={openUploadModal}
                  userRole={user?.role}
                />
              ) : (
                <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md
p-8 text-center">
```

```
                <FaFileAlt className="mx-auto text-gray-400 text-6xl mb-4" />
                <h3 className="text-xl font-medium text-gray-900 dark:text-
white mb-2">
                  Select an Employee
                </h3>
                <p className="text-gray-500 dark:text-gray-400">
                  Choose an employee from the list to view their documents
                </p>
              </div>
            )}
          </div>
        </div>
      )}

      {/* Employee's own documents - Show for non-admin users */}
      {user?.role !== 'Admin' && user?.role !== 'Manager' && selectedEmployee
&& (
        <DocumentDetails
          employee={selectedEmployee}
          documentTypes={documentTypes}
          getDocumentStatusIcon={getDocumentStatusIcon}
          downloadDocument={downloadDocument}
          viewDocument={viewDocument}
          openUploadModal={openUploadModal}
          userRole={user?.role}
          isOwnDocuments={true}
        />
      )}

      {/* Upload Modal */}
      {showUploadModal && (
        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
          <div className="bg-white dark:bg-slate-900 rounded-lg shadow-lg max-w-
md w-full mx-4">
            <div className="flex justify-between items-center bg-blue-600 text-
white px-6 py-4 rounded-t-lg">
              <h3 className="text-lg font-medium">Upload Document</h3>
              <button
                onClick={() => {
                  setShowUploadModal(false);
                  setUploadFile(null);
                  setUploadDocumentType('');
                }}
                className="text-white hover:text-gray-200"
              >
                &times;
              </button>
            </div>
            <div className="p-6">
              <div className="mb-4">
                <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                  Document Type
                </label>
                <select
                  value={uploadDocumentType}
                  onChange={(e) => setUploadDocumentType(e.target.value)}
                  className="w-full px-3 py-2 border border-gray-300
```

```
dark:border-gray-600 rounded-md bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                  >
                    <option value="">Select document type</option>
                    {documentTypes.map(docType => (
                      <option key={docType.key} value={docType.key}>
                        {docType.label} {docType.required ? '(Required)' :
'(Optional)'}
                      </option>
                    ))}
                  </select>
                </div>

                <div className="mb-4">
                  <label className="block text-sm font-medium text-gray-700
dark:text-gray-300 mb-2">
                    Select File
                  </label>
                  <input
                    type="file"
                    onChange={(e) => setUploadFile(e.target.files[0])}
                    accept={uploadDocumentType ? documentTypes.find(dt => dt.key
=== uploadDocumentType)?.accept : '*'}
                    className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                  />
                  {uploadDocumentType && (
                    <p className="text-xs text-gray-500 mt-1">
                      Accepted formats: {documentTypes.find(dt => dt.key ===
uploadDocumentType)?.accept}
                    </p>
                  )}
                </div>

                <div className="flex justify-end space-x-3">
                  <button
                    onClick={() => {
                      setShowUploadModal(false);
                      setUploadFile(null);
                      setUploadDocumentType('');
                    }}
                    className="px-4 py-2 border border-gray-300 dark:border-
gray-600 text-gray-700 dark:text-gray-300 rounded-md hover:bg-gray-50
dark:hover:bg-gray-700"
                  >
                    Cancel
                  </button>
                  <button
                    onClick={handleUploadDocument}
                    disabled={!uploadFile || !uploadDocumentType ||
uploadingDocument}
                    className="px-4 py-2 bg-blue-600 text-white rounded-md
hover:bg-blue-700 disabled:opacity-50 disabled:cursor-not-allowed"
                  >
                    {uploadingDocument ? 'Uploading...' : 'Upload'}
                  </button>
                </div>
              </div>
```

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

// Document Details Component
const DocumentDetails = ({
  employee,
  documentTypes,
  getDocumentStatusIcon,
  downloadDocument,
  viewDocument,
  openUploadModal,
  userRole,
  isOwnDocuments = false
}) => {
  const documents = employee.documents || {};

  return (
    <div className="bg-white dark:bg-slate-900 rounded-lg shadow-md">
      <div className="p-6 border-b border-gray-200 dark:border-gray-700">
        <h3 className="text-xl font-semibold text-gray-900 dark:text-white">
          {isOwnDocuments ? 'My Documents' : `${employee.fullName ||
employee.email}'s Documents`}
        </h3>
        <p className="text-gray-500 dark:text-gray-400 mt-1">
          {employee.department} • {employee.role} • ID: {employee.employeeId ||
'N/A'}
        </p>
      </div>

      <div className="p-6">
        <div className="grid grid-cols-1 gap-4">
          {documentTypes.map(docType => {
            const docInfo = documents[docType.key];
            const hasDocument = docInfo && docInfo.filename;
            const IconComponent = docType.icon;

            return (
              <div
                key={docType.key}
                className="flex items-center justify-between p-4 border border-
gray-200 dark:border-gray-700 rounded-lg hover:bg-gray-50 dark:hover:bg-slate-800"
              >
                <div className="flex items-center">
                  <IconComponent className="text-gray-400 mr-3" />
                  <div>
                    <h4 className="font-medium text-gray-900 dark:text-white">
                      {docType.label}
                      {docType.required && <span className="text-red-500 ml-1">*</
span>}
                    </h4>
                    <p className="text-sm text-gray-500 dark:text-gray-400">
                      {hasDocument ? `Uploaded: ${docInfo.originalName ||
docInfo.filename}` : 'Not uploaded'}
                    </p>
```

```
                  </div>
                </div>

                <div className="flex items-center space-x-2">
                  {getDocumentStatusIcon(employee, docType)}

                  {hasDocument && (
                    <>
                      <button
                        onClick={() => viewDocument(employee, docType)}
                        className="p-2 text-blue-600 hover:bg-blue-50
dark:hover:bg-blue-900/20 rounded"
                        title="View Document"
                      >
                        <FaEye />
                      </button>
                      <button
                        onClick={() => downloadDocument(employee, docType)}
                        className="p-2 text-green-600 hover:bg-green-50
dark:hover:bg-green-900/20 rounded"
                        title="Download Document"
                      >
                        <FaDownload />
                      </button>
                    </>
                  )}

                  {/* Allow upload for own documents or admin/manager */}
                  {(isOwnDocuments || userRole === 'Admin' || userRole ===
'Manager') && (
                    <button
                      onClick={() => openUploadModal(employee, docType.key)}
                      className="p-2 text-blue-600 hover:bg-blue-50 dark:hover:bg-
blue-900/20 rounded"
                      title={hasDocument ? 'Replace Document' : 'Upload Document'}
                    >
                      <FaUpload />
                    </button>
                  )}
                </div>
              </div>
            );
          })}
        </div>
      </div>
    </div>
  );
};

export default DocumentManagementPage;
```

src/pages/EmployeeManagementPage.jsx

```
import React, { useState } from 'react';
import { FaUsers, FaCalendarAlt, FaClock, FaAddressBook, FaCog, FaMoneyBillWave }
from 'react-icons/fa';
import Layout from '../components/Layout/Layout';
import EmployeeList from '../components/Employee/EmployeeList';
```

```
import EmployeeDirectory from '../components/Employee/EmployeeDirectory';
import BulkOperations from '../components/Employee/BulkOperations';
import LeaveManagement from '../components/Employee/LeaveManagement';
import LeaveApproval from '../components/Admin/LeaveApproval';
import AttendanceManagement from '../components/Employee/AttendanceManagement';
import PayrollComponent from '../components/Employee/PayrollComponent';
import { useAuth } from '../context/AuthContext';

const EmployeeManagementPage = () => {
  const [activeTab, setActiveTab] = useState('employees');
  const { user } = useAuth();

  const tabs = [
    { id: 'employees', label: 'Employees', icon: FaUsers },
    { id: 'directory', label: 'Directory', icon: FaAddressBook },
    ...(user?.role === 'Admin' || user?.role === 'Manager' || user?.role ===
'HR' ?
      [{ id: 'bulk-ops', label: 'Bulk Operations', icon: FaCog }] : []),
    { id: 'leave', label: user?.role === 'Admin' || user?.role === 'Manager' ?
'Leave Approvals' : 'Leave Management', icon: FaCalendarAlt },
    { id: 'attendance', label: 'Attendance', icon: FaClock },
    ...(user?.role === 'Admin' || user?.role === 'Manager' || user?.role ===
'HR' ?
      [{ id: 'payroll', label: 'Payroll Management', icon: FaMoneyBillWave }] :
[])
  ];

  const renderContent = () => {
    switch (activeTab) {
      case 'employees':
        return <EmployeeList />;
      case 'directory':
        return <EmployeeDirectory />;
      case 'bulk-ops':
        return <BulkOperations />;
      case 'leave':
        return user?.role === 'Admin' || user?.role === 'Manager' ?
          <LeaveApproval /> :
          <LeaveManagement userRole={user?.role} />;
      case 'attendance':
        return <AttendanceManagement userRole={user?.role} />;
      case 'payroll':
        return <PayrollComponent />;
      default:
        return <EmployeeList />;
    }
  };

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="mb-8">
          <h1 className="text-3xl font-bold text-gray-900 dark:text-white">
            Employee Management
          </h1>
          <p className="mt-2 text-gray-600 dark:text-gray-400">
            Manage your organization's employees, leave requests, and attendance
          </p>
        </div>
```

```
        {/* Tab Navigation */}
        <div className="mb-6">
          <div className="border-b border-gray-200 dark:border-gray-700">
            <nav className="-mb-px flex space-x-8">
              {tabs.map((tab) => {
                const Icon = tab.icon;
                return (
                  <button
                    key={tab.id}
                    onClick={() => setActiveTab(tab.id)}
                    className={`${
                      activeTab === tab.id
                        ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                        : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
                    } whitespace-nowrap py-2 px-1 border-b-2 font-medium text-sm
flex items-center`}
                  >
                    <Icon className="mr-2 h-4 w-4" />
                    {tab.label}
                  </button>
                );
              })}
            </nav>
          </div>
        </div>

        {/* Tab Content */}
        {renderContent()}
      </div>
    </Layout>
  );
};

export default EmployeeManagementPage;
```

src/pages/HomePage.jsx

```
// src/pages/HomePage.jsxÐ
import React from "react";Ð
import { Link } from "react-router-dom";Ð
import Layout from "../components/Layout/Layout";Ð
import { useAuth } from "../context/AuthContext";Ð
import { componentClasses, darkModeClasses } from '../utils/darkModeClasses';Ð
import GuestChat from "../components/Chat/GuestChat";Ð
Ð
import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';Ð
const HomePage = () => {Ð
  const { user } = useAuth();Ð
Ð
  return (Ð
    <Layout>Ð
      {/* Hero Section - Improved responsiveness */}Ð
      <div className="bg-gradient-to-br from-blue-700 to-indigo-800 text-white
py-12 md:py-20 flex items-center">Ð
        <div className="container mx-auto px-4 sm:px-6 lg:px-8">Ð
```

```
          <div className="max-w-3xl mx-auto">Ð
            <h1 className="text-3xl sm:text-4xl md:text-5xl font-bold mb-4 text-
center sm:text-left">TrainCape CRM</h1>Ð
            <p className="text-lg sm:text-xl mb-6 text-center sm:text-
left">Manage leads and track sales in one centralized platform.</p>Ð
            {!user ? (Ð
              <div className="flex flex-wrap gap-4 justify-center sm:justify-
start">Ð
                <Link to="/login" className="w-full sm:w-auto px-6 py-3 bg-
blue-800 text-white font-semibold rounded-lg shadow-md dark:shadow-black/25
hover:bg-blue-900 transition text-center">Ð
                  Sign InÐ
                </Link>Ð
                <Link to="/customer-signup" className="w-full sm:w-auto px-6 py-3
bg-blue-800 text-white font-semibold rounded-lg shadow-md dark:shadow-black/25
hover:bg-blue-900 transition text-center">Ð
                  Customer Sign UpÐ
                </Link>Ð
              </div>Ð
            ) : (Ð
              <div className="flex flex-wrap gap-4 justify-center sm:justify-
start">Ð
                {user.role === "Sales Person" && (Ð
                  <Link to="/sales" className="w-full sm:w-auto px-6 py-3 bg-
transparent border-2 border-white text-white font-semibold rounded-lg hover:bg-
white dark:bg-slate-900 transition-all duration-200 ease-out text-center">Ð
                    My LeadsÐ
                  </Link>Ð
                )}Ð
                {(user.role === "Lead Person" || user.role === "Manager" ||
user.role === "Admin") && (Ð
                  <Link to="/leads" className="w-full sm:w-auto px-6 py-3 bg-
transparent border-2 border-white text-white font-semibold rounded-lg hover:bg-
white dark:bg-slate-900 transition-all duration-200 ease-out text-center">Ð
                    Manage LeadsÐ
                  </Link>Ð
                )}Ð
                <Link to="/profile" className="w-full sm:w-auto px-6 py-3 bg-
transparent border-2 border-white text-white font-semibold rounded-lg hover:bg-
white dark:bg-slate-900 transition-all duration-200 ease-out text-center">Ð
                  My ProfileÐ
                </Link>Ð
              </div>Ð
            )}Ð
          </div>Ð
        </div>Ð
      </div>Ð
      Ð
      {/* Quick Access Cards - Improved for mobile */}Ð
      <div className="bg-gray-50 dark:bg-slate-800 transition-all duration-200
ease-out py-6">Ð
        <div className="container mx-auto px-4 sm:px-6 lg:px-8 -mt-6 sm:-mt-8 md:-
mt-10 relative z-10">Ð
          <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-4">Ð
            <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 rounded-lg p-4 sm:p-5 shadow-sm hover:shadow-md transition-
all duration-200 ease-out">Ð
              <div className="flex flex-col sm:flex-row items-center sm:items-
start">Ð
```

```
                <div className="bg-blue-100 dark:bg-blue-900/30 text-blue-700
dark:text-blue-300 p-3 rounded-lg mb-3 sm:mb-0 sm:mr-4">Ð
                  <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6"
fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M17 20h5v-2a3 3 0 00-5.356-1.857M17 20H7m10
0v-2c0-.656-.126-1.283-.356-1.857M7 20H2v-2a3 3 0 015.356-1.857M7
20v-2c0-.656.126-1.283.356-1.857m0 0a5.002 5.002 0 019.288 0M15 7a3 3 0 11-6 0 3
3 0 016 0zm6 3a2 2 0 11-4 0 2 2 0 014 0zM7 10a2 2 0 11-4 0 2 2 0 014 0z" />Ð
                  </svg>Ð
                </div>Ð
                <div className="text-center sm:text-left">Ð
                  <h3 className="text-lg font-bold mb-1 text-gray-900 dark:text-
white">Contact Management</h3>Ð
                  <p className="text-gray-600 dark:text-gray-300 text-
sm">Organize client data with powerful filters and segments.</p>Ð
                  <Link to="/management-contacts" className="text-blue-700
dark:text-blue-400 font-medium text-sm mt-2 inline-block hover:underline">Access
Contacts !’</Link>Ð
                </div>Ð
              </div>Ð
            </div>Ð
            Ð
            <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 rounded-lg p-4 sm:p-5 shadow-sm hover:shadow-md transition-
all duration-200 ease-out">Ð
              <div className="flex flex-col sm:flex-row items-center sm:items-
start">Ð
                <div className="bg-green-100 dark:bg-green-900/30 text-green-700
dark:text-green-300 p-3 rounded-lg mb-3 sm:mb-0 sm:mr-4">Ð
                  <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6"
fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0
002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2
2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z" />Ð
                  </svg>Ð
                </div>Ð
                <div className="text-center sm:text-left">Ð
                  <h3 className="text-lg font-bold mb-1 text-gray-900 dark:text-
white">Sales Pipeline</h3>Ð
                  <p className="text-gray-600 dark:text-gray-300 text-sm">Track
deals from lead to close with customizable processes.</p>Ð
                  <Link to="/sales" className="text-green-700 dark:text-green-400
font-medium text-sm mt-2 inline-block hover:underline">View Pipeline !’</Link>Ð
                </div>Ð
              </div>Ð
            </div>Ð
            Ð
            <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 rounded-lg p-4 sm:p-5 shadow-sm hover:shadow-md transition-
all duration-200 ease-out">Ð
              <div className="flex flex-col sm:flex-row items-center sm:items-
start">Ð
                <div className="bg-purple-100 dark:bg-purple-900/30 text-
purple-700 dark:text-purple-300 p-3 rounded-lg mb-3 sm:mb-0 sm:mr-4">Ð
                  <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6"
fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
```

```
strokeWidth={2} d="M9 5H7a2 2 0 00-2 2v12a2 2 0 002 2h10a2 2 0 002-2V7a2 2 0
00-2-2h-2M9 5a2 2 0 002 2h2a2 2 0 002-2M9 5a2 2 0 012-2h2a2 2 0 012 2" />Ð
                  </svg>Ð
                </div>Ð
                <div className="text-center sm:text-left">Ð
                  <h3 className="text-lg font-bold mb-1 text-gray-900 dark:text-
white">Task Management</h3>Ð
                  <p className="text-gray-600 dark:text-gray-300 text-
sm">Schedule exams and get automated reminders.</p>Ð
                  <Link to="/tasks" className="text-purple-700 dark:text-
purple-400 font-medium text-sm mt-2 inline-block hover:underline">Manage Tasks !’</
Link>Ð
                </div>Ð
              </div>Ð
            </div>Ð
            Ð
            {user && (user.role === "Manager" || user.role === "Admin") && (Ð
              <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 rounded-lg p-4 sm:p-5 shadow-sm hover:shadow-md transition-
all duration-200 ease-out">Ð
                <div className="flex flex-col sm:flex-row items-center sm:items-
start">Ð
                  <div className="bg-indigo-100 dark:bg-indigo-900/30 text-
indigo-700 dark:text-indigo-300 p-3 rounded-lg mb-3 sm:mb-0 sm:mr-4">Ð
                    <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6"
fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M12 6V4m0 2a2 2 0 100 4m0-4a2 2 0 110 4m-6 8a2 2 0 100-4m0 4a2
2 0 110-4m0 4v2m0-6V4m6 6v10m6-2a2 2 0 100-4m0 4a2 2 0 110-4m0 4v2m0-6V4" />Ð
                    </svg>Ð
                  </div>Ð
                  <div className="text-center sm:text-left">Ð
                    <h3 className="text-lg font-bold mb-1 text-gray-900 dark:text-
white">Manager Dashboard</h3>Ð
                    <p className="text-gray-600 dark:text-gray-300 text-
sm">Manage users, roles and system access.</p>Ð
                    <Link to="/manager" className="text-indigo-700 dark:text-
indigo-400 font-medium text-sm mt-2 inline-block hover:underline">Access
Dashboard !’</Link>Ð
                  </div>Ð
                </div>Ð
              </div>Ð
            )}Ð
          </div>Ð
        </div>Ð
      </div>Ð
      Ð
      {/* Quick Stats - Made mobile friendly */}Ð
      <div className="bg-white dark:bg-slate-900 transition-all duration-200 ease-
out py-8 sm:py-10">Ð
        <div className="container mx-auto px-4 sm:px-6 lg:px-8">Ð
          <div className="grid grid-cols-2 md:grid-cols-4 gap-3 sm:gap-4">Ð
            {[Ð
              { label: "Leads", value: "Today", icon: "Ø=ÜÊ" },Ð
              { label: "Tasks", value: "Pending", icon: "' " },Ð
              { label: "Sales", value: "This Month", icon: "Ø=Ü°" },Ð
              { label: "Team", value: "Members", icon: "Ø=Üe" }Ð
            ].map((stat, index) => (Ð
              <Link key={index} to={index === 0 ? "sales" : index === 1 ? "/
```

```
tasks" : index === 2 ? "sales-tracking" : "/profile"} Đ
                  className="bg-gray-50 dark:bg-slate-800 rounded-lg p-3 sm:p-4
text-center hover:bg-gray-100 dark:hover:bg-slate-700 transition-all duration-200
ease-out">Đ
                  <div className="text-xl sm:text-2xl mb-1">{stat.icon}</div>Đ
                  <div className="text-base sm:text-lg font-bold text-gray-900
dark:text-white">{stat.label}</div>Đ
                  <div className="text-xs sm:text-sm text-gray-600 dark:text-
gray-300">{stat.value}</div>Đ
                </Link>Đ
              ))}Đ
            </div>Đ
          </div>Đ
        </div>Đ
        Đ
        {/* Role-based Access - Improved layout for mobile */}Đ
        <div className="bg-gray-50 dark:bg-slate-800 transition-all duration-200
ease-out py-8 sm:py-10">Đ
          <div className="container mx-auto px-4 sm:px-6 lg:px-8">Đ
            <h2 className="text-xl sm:text-2xl font-bold mb-4 sm:mb-6 text-center
text-gray-900 dark:text-white">Your Role: {user?.role || "Not Logged In"}</h2>Đ
            <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-4">Đ
              <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 sm:p-5 rounded-lg
shadow shadow-sm dark:shadow-black/25">Đ
                <h3 className="font-bold text-blue-700 dark:text-blue-400 mb-2 text-
center sm:text-left">Lead Person</h3>Đ
                <ul className="space-y-2">Đ
                  <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Đ
                    <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Đ
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Đ
                    </svg>Đ
                    <span>Create and manage leads</span>Đ
                  </li>Đ
                  <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Đ
                    <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Đ
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Đ
                    </svg>Đ
                    <span>Assign leads to team members</span>Đ
                  </li>Đ
                  <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Đ
                    <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Đ
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Đ
                    </svg>Đ
                    <span>Track lead progress</span>Đ
                  </li>Đ
                </ul>Đ
              </div>Đ
              Đ
              <div className="bg-white dark:bg-slate-900 border border-slate-200
```

```
dark:border-slate-700 transition-all duration-200 ease-out p-4 sm:p-5 rounded-lg
shadow shadow-sm dark:shadow-black/25">Ð
              <h3 className="font-bold text-green-700 dark:text-green-400 mb-2
text-center sm:text-left">Sales Person</h3>Ð
              <ul className="space-y-2">Ð
                <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Ð
                  <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Ð
                  </svg>Ð
                  <span>Access assigned leads</span>Ð
                </li>Ð
                <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Ð
                  <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Ð
                  </svg>Ð
                  <span>Update lead status</span>Ð
                </li>Ð
                <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Ð
                  <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Ð
                  </svg>Ð
                  <span>Schedule and track exams</span>Ð
                </li>Ð
              </ul>Ð
            </div>Ð
            Ð
            <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 sm:p-5 rounded-lg
shadow shadow-sm dark:shadow-black/25">Ð
              <h3 className="font-bold text-purple-700 dark:text-purple-400 mb-2
text-center sm:text-left">Manager/Admin</h3>Ð
              <ul className="space-y-2">Ð
                <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Ð
                  <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Ð
                  </svg>Ð
                  <span>Full system access</span>Ð
                </li>Ð
                <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Ð
                  <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Ð
                  </svg>Ð
                  <span>View all leads and sales</span>Ð
                </li>Ð
```

```
                <li className="flex items-start text-sm text-gray-700 dark:text-
gray-300">Ð
                  <svg className="h-4 w-4 text-green-500 dark:text-green-400 mr-2
mt-0.5 flex-shrink-0" fill="none" viewBox="0 0 24 24" stroke="currentColor">Ð
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M5 13l4 4L19 7" />Ð
                  </svg>Ð
                  <span>Manage team and generate reports</span>Ð
                </li>Ð
                {user && (user.role === "Manager" || user.role === "Admin") && (Ð
                  <li className="mt-4">Ð
                    <Link to="/manager" className="inline-block w-full bg-
purple-600 hover:bg-purple-700 text-white font-semibold py-2 px-4 rounded text-
center transition">Ð
                      Access Manager DashboardÐ
                    </Link>Ð
                  </li>Ð
                )}Ð
              </ul>Ð
            </div>Ð
          </div>Ð
        </div>Ð
      </div>Ð
      Ð
      {/* CTA Section - Improved for mobile */}Ð
      <div className="bg-gradient-to-r from-blue-600 to-indigo-700 text-white
py-8 sm:py-10">Ð
        <div className="container mx-auto px-4 sm:px-6 lg:px-8 text-center">Ð
          {!user ? (Ð
            <div>Ð
              <h2 className="text-xl sm:text-2xl font-bold mb-4 text-white">Sign
in to access your dashboard</h2>Ð
              <div className="flex flex-wrap gap-4 justify-center">Ð
                <Link to="/login" className="px-6 py-3 bg-white dark:bg-slate-900
border border-slate-200 dark:border-slate-700 text-blue-700 dark:text-blue-400
font-semibold rounded-lg shadow-sm hover:bg-gray-50 dark:hover:bg-slate-800
transition-all duration-200 ease-out inline-block">Ð
                  LoginÐ
                </Link>Ð
                <Link to="/customer-signup" className="px-6 py-3 bg-blue-800 text-
white font-semibold rounded-lg shadow hover:bg-blue-900 transition inline-block">Ð
                  Customer Sign UpÐ
                </Link>Ð
              </div>Ð
            </div>Ð
          ) : (Ð
            <div>Ð
              <h2 className="text-xl sm:text-2xl font-bold mb-3 text-
white">Welcome, {user.fullName}</h2>Ð
              <p className="mb-4 text-sm sm:text-base text-white">Access your
dashboard to manage your tasks.</p>Ð
              <Link Ð
                to={Ð
                  user.role === "Customer" ? "/customer" :Ð
                  user.role === "Sales Person" ? "/sales" : "/leads"Ð
                } Ð
                className="px-6 py-3 bg-white dark:bg-slate-900 border border-
slate-200 dark:border-slate-700 text-blue-700 dark:text-blue-400 font-semibold
rounded-lg shadow-sm hover:bg-gray-50 dark:hover:bg-slate-800 transition-all
```

```
duration-200 ease-out inline-block w-full sm:w-auto"Đ
              >Đ
                Go to DashboardĐ
              </Link>Đ
            </div>Đ
          )}Đ
        </div>Đ
      </div>Đ
      Đ
      {/* Guest Chat for non-registered users */}Đ
      {!user && <GuestChat />}Đ
    </Layout>Đ
  );Đ
};Đ
Đ
export default HomePage;Đ
```

src/pages/LeadSalesSheet.jsx

```
import React, { useState, useEffect } from 'react';
import { salesAPI, authAPI } from '../services/api';
import { useAuth } from '../context/AuthContext';
import Layout from '../components/Layout/Layout';
import { FaDownload, FaEdit, FaSave, FaTimesCircle, FaFilter, FaCalendar,
FaSync } from 'react-icons/fa';
import * as XLSX from 'xlsx';
import { toast } from 'react-toastify';
import axios from 'axios';
import LoggingService from '../services/loggingService'; // Add LoggingService
import

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const LeadSalesSheet = () => {
  const [sales, setSales] = useState([]);
  const [filteredSales, setFilteredSales] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [salesPersons, setSalesPersons] = useState([]);
  const [editingSaleId, setEditingSaleId] = useState(null);
  const [editData, setEditData] = useState({});
  const [refreshing, setRefreshing] = useState(false);
  const { user } = useAuth();

  // Date filtering state
  const [filterMonth, setFilterMonth] = useState(new Date().getMonth() + 1); //
Current month (1-12)
  const [filterYear, setFilterYear] = useState(new Date().getFullYear()); //
Current year
  const [showCurrentMonth, setShowCurrentMonth] = useState(false); // Changed to
false - don't filter by default
  const [showAllSales, setShowAllSales] = useState(true); // Add showAllSales
state - default to true

  // Generate month options
  const months = [
    { value: 1, label: "January" },
    { value: 2, label: "February" },
```

```
  { value: 3, label: "March" },
  { value: 4, label: "April" },
  { value: 5, label: "May" },
  { value: 6, label: "June" },
  { value: 7, label: "July" },
  { value: 8, label: "August" },
  { value: 9, label: "September" },
  { value: 10, label: "October" },
  { value: 11, label: "November" },
  { value: 12, label: "December" }
];

// Generate year options (5 years back from current year)
const currentYear = new Date().getFullYear();
const years = Array.from({ length: 6 }, (_, i) => currentYear - i);

useEffect(() => {
  loadSalesData();
  loadUsers();

  // Set up automatic refresh every 2 minutes
  const refreshInterval = setInterval(() => {
    console.log("Auto-refreshing sales data...");
    loadSalesData(true);
  }, 120000); // 2 minutes

  return () => {
    clearInterval(refreshInterval);
  };
}, []);

// Apply date filters when sales data changes
useEffect(() => {
  if (sales.length > 0) {
    applyDateFilters();
  }
}, [sales, filterMonth, filterYear, showCurrentMonth, showAllSales]);

// Function to filter sales by selected date
const applyDateFilters = () => {
  // If showAllSales is true, show all sales without filtering
  if (showAllSales) {
    setFilteredSales(sales);
    return;
  }

  if (showCurrentMonth) {
    // Show current month data
    const currentMonth = new Date().getMonth() + 1; // 1-12
    const currentYear = new Date().getFullYear();

    const filtered = sales.filter(sale => {
      // Make sure we have a valid date to work with
      if (!sale.date && !sale.createdAt) {
        console.log('Sale has no date:', sale);
        return false;
      }

      const saleDate = new Date(sale.date || sale.createdAt);
```

```
        const saleMonth = saleDate.getMonth() + 1; // Convert to 1-12 format
        const saleYear = saleDate.getFullYear();

        return (
          saleMonth === currentMonth &&
          saleYear === currentYear
        );
      });

      console.log(`Filtered to current month: ${currentMonth}/${currentYear}.
Found ${filtered.length} sales.`);
      setFilteredSales(filtered);
    } else {
      // Show selected month/year data
      const filtered = sales.filter(sale => {
        // Make sure we have a valid date to work with
        if (!sale.date && !sale.createdAt) {
          console.log('Sale has no date:', sale);
          return false;
        }

        const saleDate = new Date(sale.date || sale.createdAt);
        const saleMonth = saleDate.getMonth() + 1; // Convert to 1-12 format
        const saleYear = saleDate.getFullYear();

        console.log(`Sale date: ${saleDate.toISOString()}, Month: ${saleMonth},
Year: ${saleYear}, Filter: ${filterMonth}/${filterYear}`);

        return (
          saleMonth === filterMonth &&
          saleYear === filterYear
        );
      });

      console.log(`Filtered to ${filterMonth}/${filterYear}. Found
${filtered.length} sales.`);
      setFilteredSales(filtered);
    }
  };

  const loadSalesData = async (isAutoRefresh = false) => {
    try {
      if (!isAutoRefresh) {
        setLoading(true);
      } else {
        setRefreshing(true);
      }
      setError(null);

      console.log('Fetching lead sales for user:', user?.fullName, user?.role,
user?._id);

      try {
        const res = await salesAPI.getAllForced();

        if (res.data && res.data.success) {
          // Filter the sales to only show ones where this user is the lead person
          const leadPersonSales = res.data.data.filter(sale =>
            sale.leadPerson === user._id ||
```

```
            (sale.leadPerson && sale.leadPerson._id === user._id)
          );

          console.log(`Lead sales data loaded: ${leadPersonSales.length} sales`);
          setSales(leadPersonSales);

          // Initialize filteredSales with all sales
          if (!isAutoRefresh) {
            setFilteredSales(leadPersonSales);
          } else {
            // When auto-refreshing, maintain filters but update underlying data
            applyDateFilters();
          }

          if (leadPersonSales.length === 0) {
            console.log('No sales found. This could be because:');
            console.log('1. No sales have been assigned to this lead person');
            console.log('2. Sales were created without selecting this lead
person');
          } else {
            // Log a sample record for debugging
            console.log('Sample sales data:', leadPersonSales[0]);
          }

          if (isAutoRefresh) {
            toast.info("Sales data refreshed automatically");
          }
        } else {
          console.error('Failed to load lead sales data:', res.data);
          setError('Failed to load sales data: ' + (res.data?.message || 'Unknown
error'));
        }
      } catch (apiError) {
        console.error('API service call failed:', apiError);
        setError('Failed to load sales data. Please try again.');
      }
    } catch (err) {
      console.error('Error loading sales data:', err);
      if (err.response) {
        console.error('Error details:', err.response.data);
        console.error('Error status:', err.response.status);
      }
      setError('Failed to load sales data. Please try again.');
    } finally {
      setLoading(false);
      setRefreshing(false);
    }
  };

  // Handle month change
  const handleMonthChange = (e) => {
    setFilterMonth(parseInt(e.target.value));
    setShowCurrentMonth(false);
    setShowAllSales(false); // Disable show all when selecting specific month
  };

  // Handle year change
  const handleYearChange = (e) => {
    setFilterYear(parseInt(e.target.value));
```

```
    setShowCurrentMonth(false);
    setShowAllSales(false); // Disable show all when selecting specific year
  };

  // Handle reset to current month
  const handleResetToCurrentMonth = () => {
    const today = new Date();
    setFilterMonth(today.getMonth() + 1);
    setFilterYear(today.getFullYear());
    setShowCurrentMonth(true);
    setShowAllSales(false); // Disable show all when resetting to current month
  };

  const loadUsers = async () => {
    try {
      // Get sales persons
      const salesRes = await authAPI.getUsers('Sales Person');
      setSalesPersons(salesRes.data.data || []);
    } catch (err) {
      console.error('Error loading users:', err);
    }
  };

  const handleEdit = (sale) => {
    setEditingSaleId(sale._id);
    // Ensure we have default values for all fields to prevent undefined errors
    setEditData({
      DATE: sale.date || new Date().toISOString(),
      NAME: sale.customerName || '',
      COUNTRY: sale.country || '',
      COURSE: sale.course || '',
      CODE: sale.countryCode || '+1',
      NUMBER: sale.contactNumber || '',
      'E-MAIL': sale.email || '',
      'PSUDO ID': sale.pseudoId || '',
      'SALE PERSON': (sale.salesPerson && (sale.salesPerson._id ||
sale.salesPerson)) || '',
      'LEAD PERSON': (sale.leadPerson && (sale.leadPerson._id ||
sale.leadPerson)) || user.id || '',
      SOURSE: sale.source || '',
      'CLIENT REMARK': sale.clientRemark || '',
      FEEDBACK: sale.feedback || '',
      TOTAL_COST: sale.totalCost || 0,
      TOTAL_COST_CURRENCY: sale.totalCostCurrency || 'USD',
      TOKEN_AMOUNT: sale.tokenAmount || 0,
      TOKEN_AMOUNT_CURRENCY: sale.tokenAmountCurrency || 'USD'
    });
    console.log('Edit data populated:', sale);
  };

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

  const handleSave = async () => {
```

```
    try {
      console.log('Saving with data:', editData);

      const saleData = {
        date: editData.DATE,
        customerName: editData.NAME,
        country: editData.COUNTRY,
        course: editData.COURSE,
        countryCode: editData.CODE,
        contactNumber: editData.NUMBER,
        email: editData['E-MAIL'],
        pseudoId: editData['PSUDO ID'],
        salesPerson: editData['SALE PERSON'],
        leadPerson: editData['LEAD PERSON'],
        source: editData.SOURSE,
        clientRemark: editData['CLIENT REMARK'],
        feedback: editData.FEEDBACK,
        totalCost: parseFloat(editData.TOTAL_COST) || 0,
        totalCostCurrency: editData.TOTAL_COST_CURRENCY,
        tokenAmount: parseFloat(editData.TOKEN_AMOUNT) || 0,
        tokenAmountCurrency: editData.TOKEN_AMOUNT_CURRENCY
      };

      const response = await salesAPI.update(editingSaleId, saleData);
      if (response.data && response.data.success) {
        // Log the sale update
        try {
          await LoggingService.logSaleUpdate(editingSaleId, saleData);
        } catch (logError) {
          console.error('Error logging sale update:', logError);
        }

        console.log('Sale updated successfully:', response.data);
        console.log('Updated sale data from server:', response.data.data);
        toast.success("Sale updated successfully");

        // Update the local sales state with the updated sale
        setSales(prevSales => prevSales.map(sale => {
          if (sale._id === editingSaleId) {
            console.log('Updating sale in local state:', response.data.data);
            return response.data.data;
          }
          return sale;
        }));

        // Temporarily disable date filtering to show all sales so user can see
their changes
        const wasShowingAllSales = showAllSales;
        if (!wasShowingAllSales) {
          setShowAllSales(true);
          toast.info("Showing all sales to display your changes. Use filters to
narrow down if needed.");
        }

      } else {
        console.error('Failed to update sale:', response.data);
        setError("Failed to update sale: " + (response.data?.message || "Unknown
error"));
      }
```

```
      setEditingSaleId(null);

      // Only reload data if the update failed, otherwise we've already updated
the local state
      if (!response.data || !response.data.success) {
        loadSalesData();
      }
    } catch (err) {
      console.error('Error updating sale:', err);
      setError('Failed to update sale. Please try again.');
      setEditingSaleId(null);
      // Reload data on error to ensure consistency
      loadSalesData();
    }
  };

  const handleCancel = () => {
    setEditingSaleId(null);
  };

  const exportToExcel = () => {
    const fileName = `Lead-Sales-Sheet-${new Date().toISOString().split('T')
[0]}.xlsx`;

    // Format data for export - use the currently filtered sales
    const exportData = filteredSales.map(sale => ({
      'DATE': new Date(sale.date).toLocaleDateString(),
      'NAME': sale.customerName,
      'COUNTRY': sale.country,
      'COURSE': sale.course,
      'CODE': sale.countryCode || '',
      'NUMBER': sale.contactNumber,
      'E-MAIL': sale.email || '',
      'PSUDO ID': sale.pseudoId || '',
      'SALE PERSON': sale.salesPerson?.fullName || '',
      'LEAD PERSON': sale.leadPerson?.fullName || '',
      'SOURSE': sale.source || '',
      'CLIENT REMARK': sale.clientRemark || '',
      'FEEDBACK': sale.feedback || '',
      'TOTAL COST': `${sale.totalCost?.toFixed(2) || '0.00'}
${sale.totalCostCurrency || 'USD'}`,
      'TOKEN AMOUNT': `${sale.tokenAmount?.toFixed(2) || '0.00'}
${sale.tokenAmountCurrency || 'USD'}`
    }));

    // Create worksheet
    const worksheet = XLSX.utils.json_to_sheet(exportData);

    // Create workbook
    const workbook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(workbook, worksheet, 'Lead Sales');

    // Generate Excel file
    XLSX.writeFile(workbook, fileName);
  };

  return (
    <Layout>
```

```
      <div className="container mx-auto px-4 py-8">
        <div className="flex justify-between items-center mb-6">
          <div>
            <h1 className="text-2xl font-bold text-gray-800 dark:text-
gray-200">Lead Sales Sheet</h1>
            <p className="text-sm text-gray-600 dark:text-gray-500 mt-1">
              This page shows sales where you are the Lead Person.
              Sales created by Sales Persons who select you as the Lead Person
will appear here.
            </p>
          </div>

          <div className="flex space-x-2">
            <button
              onClick={() => loadSalesData(false)}
              disabled={refreshing}
              className={`px-4 py-2 bg-blue-500 text-white rounded hover:bg-
blue-600 flex items-center ${refreshing ? 'opacity-70 cursor-not-allowed' : ''}`}
            >
              {refreshing ? (
                <>
                  <FaSync className="mr-2 animate-spin" /> Refreshing...
                </>
              ) : (
                <>
                  <FaSync className="mr-2" /> Refresh
                </>
              )}
            </button>
            <button
              onClick={exportToExcel}
              className="px-4 py-2 bg-green-500 text-white rounded hover:bg-
green-600 flex items-center"
            >
              <FaDownload className="mr-2" /> Export
            </button>
          </div>
        </div>

        {/* Date Filter Controls */}
        <div className="mb-6 p-4 bg-white dark:bg-slate-900 border border-
slate-200 dark:border-slate-700 transition-all duration-200 ease-out rounded-lg
shadow-md dark:shadow-2xl shadow-sm">
          <h3 className="text-lg font-semibold mb-3 flex items-center">
            <FaFilter className="mr-2 text-blue-500" /> Filter Sales by Date
          </h3>
          <div className="flex flex-wrap items-center gap-4">
            <div>
              <label htmlFor="month" className="block text-sm font-medium text-
gray-600 dark:text-gray-500 mb-1">Month</label>
              <select
                id="month"
                value={filterMonth}
                onChange={handleMonthChange}
                className="border border-slate-300 dark:border-slate-600 rounded-
md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-
blue-400 focus:ring-offset-2 focus:border-blue-500"
                disabled={showCurrentMonth}
              >
```

```
                {months.map(month => (
                  <option key={month.value} value={month.value}>
                    {month.label}
                  </option>
                ))}
              </select>
            </div>

            <div>
              <label htmlFor="year" className="block text-sm font-medium text-
gray-600 dark:text-gray-500 mb-1">Year</label>
              <select
                id="year"
                value={filterYear}
                onChange={handleYearChange}
                className="border border-slate-300 dark:border-slate-600 rounded-
md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-
blue-400 focus:ring-offset-2 focus:border-blue-500"
                disabled={showCurrentMonth}
              >
                {years.map(year => (
                  <option key={year} value={year}>
                    {year}
                  </option>
                ))}
              </select>
            </div>

            <div className="flex items-center ml-4">
              <input
                id="currentMonth"
                type="checkbox"
                checked={showCurrentMonth}
                onChange={() => {
                  setShowCurrentMonth(!showCurrentMonth);
                  if (!showCurrentMonth) {
                    setShowAllSales(false); // Disable show all when showing
current month
                  }
                }}
                className="h-4 w-4 text-blue-600 focus:ring-2 focus:ring-blue-500
dark:focus:ring-offset-slate-900 focus:ring-offset-2 border-slate-300 dark:border-
slate-600 rounded"
              />
              <label htmlFor="currentMonth" className="ml-2 block text-sm text-
slate-700 dark:text-slate-300">
                Show Current Month Only
              </label>
            </div>

            <div className="flex items-center ml-4">
              <input
                id="showAllSales"
                type="checkbox"
                checked={showAllSales}
                onChange={() => {
                  setShowAllSales(!showAllSales);
                  if (!showAllSales) {
                    // When enabling show all, disable other filters
```

```
                  setShowCurrentMonth(false);
                }
              }}
              className="h-4 w-4 text-blue-600 focus:ring-2 focus:ring-blue-500
dark:focus:ring-offset-slate-900 focus:ring-offset-2 border-slate-300 dark:border-
slate-600 rounded"
            />
            <label htmlFor="showAllSales" className="ml-2 block text-sm text-
slate-700 dark:text-slate-300 font-semibold text-blue-600">
              Show All Sales (No Date Filter)
            </label>
          </div>

          <button
            onClick={handleResetToCurrentMonth}
            className="bg-gray-200 dark:bg-slate-600 hover:bg-gray-300 text-
gray-800 dark:text-gray-200 py-2 px-4 rounded-md ml-auto transition duration-300
flex items-center"
          >
            <FaCalendar className="mr-2" /> Reset to Current Month
          </button>
        </div>

        <div className="mt-3 text-sm text-slate-500 dark:text-gray-400">
          {showAllSales ? (
            <p>Showing all sales regardless of date: {sales.length} total
records</p>
          ) : showCurrentMonth ? (
            <p>Showing sales for current month: {months[new
Date().getMonth()].label} {new Date().getFullYear()}</p>
          ) : (
            <p>Showing sales for: {months[filterMonth - 1].label} {filterYear}</
p>
          )}
          <p>Total: {filteredSales.length} records</p>
        </div>
      </div>

      {error && (
        <div className="bg-red-100 border-l-4 border-red-500 text-red-700 p-4
mb-4">
          <p>{error}</p>
        </div>
      )}

      {loading ? (
        <div className="flex justify-center items-center h-64">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-
blue-500"></div>
        </div>
      ) : (
        <div className="overflow-x-auto">
          <table className="min-w-full bg-white dark:bg-slate-900 transition-
all duration-200 ease-out border border-slate-200 dark:border-slate-700">
            <thead>
              <tr className="bg-gray-100 dark:bg-slate-700">
                <th className="border px-4 py-2">DATE</th>
                <th className="border px-4 py-2">NAME</th>
                <th className="border px-4 py-2">COURSE</th>
```

```
                  <th className="border px-4 py-2">NUMBER</th>
                  <th className="border px-4 py-2">EMAIL</th>
                  <th className="border px-4 py-2">COUNTRY</th>
                  <th className="border px-4 py-2">SALES PERSON</th>
                  <th className="border px-4 py-2">TOTAL COST</th>
                  <th className="border px-4 py-2">TOKEN AMOUNT</th>
                  <th className="border px-4 py-2">LEAD PERSON</th>
                  <th className="border px-4 py-2">ACTIONS</th>
                </tr>
              </thead>
              <tbody>
                {filteredSales.length === 0 ? (
                  <tr>
                    <td colSpan="11" className="border px-4 py-2 text-center">
                      <div className="py-6">
                        <p className="text-gray-600 dark:text-gray-500 mb-2">No
sales data found for you as a Lead Person.</p>
                        <p className="text-sm text-slate-500 dark:text-gray-400">
                          When Sales Persons create sales and select you as the
Lead Person, they will appear here.
                          <br />
                          Ask Sales Persons to make sure they select your name in
the Lead Person dropdown when creating sales.
                        </p>
                      </div>
                    </td>
                  </tr>
                ) : (
                  filteredSales.map(sale => (
                    <tr key={sale._id} className="hover:bg-slate-50 dark:hover:bg-
slate-800 dark:bg-slate-800 transition-all duration-200 ease-out">
                      {editingSaleId === sale._id ? (
                        // Edit mode
                        <>
                          <td className="border px-4 py-2">
                            <input
                              type="date"
                              name="DATE"
                              value={editData.DATE ? new
Date(editData.DATE).toISOString().split('T')[0] : ''}
                              onChange={handleChange}
                              className="w-full p-1 border"
                            />
                          </td>
                          <td className="border px-4 py-2">
                            <input
                              type="text"
                              name="NAME"
                              value={editData.NAME}
                              onChange={handleChange}
                              className="w-full p-1 border"
                            />
                          </td>
                          <td className="border px-4 py-2">
                            <input
                              type="text"
                              name="COURSE"
                              value={editData.COURSE}
                              onChange={handleChange}
```

```
            className="w-full p-1 border"
          />
        </td>
        <td className="border px-4 py-2">
          <input
            type="text"
            name="NUMBER"
            value={editData.NUMBER}
            onChange={handleChange}
            className="w-full p-1 border"
          />
        </td>
        <td className="border px-4 py-2">
          <input
            type="text"
            name="E-MAIL"
            value={editData['E-MAIL']}
            onChange={handleChange}
            className="w-full p-1 border"
          />
        </td>
        <td className="border px-4 py-2">
          <input
            type="text"
            name="COUNTRY"
            value={editData.COUNTRY}
            onChange={handleChange}
            className="w-full p-1 border"
          />
        </td>
        <td className="border px-4 py-2">
          <select
            name="SALE PERSON"
            value={editData['SALE PERSON']}
            onChange={handleChange}
            className="w-full p-1 border"
          >
            <option value="">Select</option>
            {salesPersons.map(person => (
              <option key={person._id} value={person._id}>
                {person.fullName}
              </option>
            ))}
          </select>
        </td>
        <td className="border px-4 py-2">
          <div className="flex items-center space-x-1">
            <input
              type="number"
              name="TOTAL_COST"
              value={editData.TOTAL_COST}
              onChange={handleChange}
              className="w-2/3 p-1 border"
            />
            <select
              name="TOTAL_COST_CURRENCY"
              value={editData.TOTAL_COST_CURRENCY}
              onChange={handleChange}
              className="w-1/3 p-1 border text-xs"
```

```
                            >
                              <option value="USD">USD</option>
                              <option value="EUR">EUR</option>
                              <option value="GBP">GBP</option>
                              <option value="CAD">CAD</option>
                              <option value="AUD">AUD</option>
                              <option value="INR">INR</option>
                              <option value="JPY">JPY</option>
                              <option value="CNY">CNY</option>
                            </select>
                          </div>
                        </td>
                        <td className="border px-4 py-2">
                          <div className="flex items-center space-x-1">
                            <input
                              type="number"
                              name="TOKEN_AMOUNT"
                              value={editData.TOKEN_AMOUNT}
                              onChange={handleChange}
                              className="w-2/3 p-1 border"
                            />
                            <select
                              name="TOKEN_AMOUNT_CURRENCY"
                              value={editData.TOKEN_AMOUNT_CURRENCY}
                              onChange={handleChange}
                              className="w-1/3 p-1 border text-xs"
                            >
                              <option value="USD">USD</option>
                              <option value="EUR">EUR</option>
                              <option value="GBP">GBP</option>
                              <option value="CAD">CAD</option>
                              <option value="AUD">AUD</option>
                              <option value="INR">INR</option>
                              <option value="JPY">JPY</option>
                              <option value="CNY">CNY</option>
                            </select>
                          </div>
                        </td>
                        <td className="border px-4 py-2">
                          {/* Lead Person cannot be changed here */}
                          {sale.leadPerson?.fullName}
                        </td>
                        <td className="border px-4 py-2">
                          <div className="flex justify-center space-x-1">
                            <button
                              onClick={handleSave}
                              className="p-1 bg-green-500 text-white rounded
text-sm hover:bg-green-600"
                            >
                              <FaSave />
                            </button>
                            <button
                              onClick={handleCancel}
                              className="p-1 bg-red-500 text-white rounded text-
sm hover:bg-red-600"
                            >
                              <FaTimesCircle />
                            </button>
                          </div>
```

```
                          </td>
                        </>
                      ) : (
                        // View mode
                        <>
                          <td className="border px-4 py-2">{new
Date(sale.date).toLocaleDateString()}</td>
                          <td className="border px-4 py-2">{sale.customerName}</
td>
                          <td className="border px-4 py-2">{sale.course}</td>
                          <td className="border px-4 py-2">
                            {sale.countryCode && sale.contactNumber ?
                              `${sale.countryCode} ${sale.contactNumber}` :
                              sale.contactNumber || 'N/A'}
                          </td>
                          <td className="border px-4 py-2">{sale.email || 'N/A'}</
td>
                          <td className="border px-4 py-2">{sale.country || 'N/A'}
</td>
                          <td className="border px-4 py-2">
                            {sale.salesPerson?.fullName ||
                             (typeof sale.salesPerson === 'string' ?
sale.salesPerson : 'N/A')}
                          </td>
                          <td className="border px-4 py-2">
                            {(sale.totalCost !== undefined ?
sale.totalCost.toFixed(2) : '0.00')} {sale.totalCostCurrency || 'USD'}
                          </td>
                          <td className="border px-4 py-2">
                            {(sale.tokenAmount !== undefined ?
sale.tokenAmount.toFixed(2) : '0.00')} {sale.tokenAmountCurrency || 'USD'}
                          </td>
                          <td className="border px-4 py-2">
                            {sale.leadPerson?.fullName ||
                             (typeof sale.leadPerson === 'string' ?
sale.leadPerson : 'N/A')}
                          </td>
                          <td className="border px-4 py-2 flex justify-center">
                            <button
                              onClick={() => handleEdit(sale)}
                              className="px-2 py-1 bg-blue-500 text-white rounded
text-sm hover:bg-blue-600 hover:bg-blue-700 dark:bg-blue-500 dark:hover:bg-
blue-600 shadow-sm dark:shadow-xl hover:shadow-md transition-all duration-200"
                            >
                              <FaEdit />
                            </button>
                          </td>
                        </>
                      )}
                    </tr>
                  ))
                )}
              </tbody>
            </table>
          </div>
        )}
      </div>
    </Layout>
  );
```

```
};

export default LeadSalesSheet;
```

src/pages/LeadSalesUpdatePage.jsx

```
import React, { useState, useEffect } from 'react';
import { salesAPI, authAPI, leadPersonSalesAPI } from '../services/api';
import { useAuth } from '../context/AuthContext';
import Layout from '../components/Layout/Layout';
import { FaPlus, FaTrash, FaEdit, FaSave, FaTimes } from 'react-icons/fa';
import toast from 'react-hot-toast';
import LoggingService from '../services/loggingService'; // Add LoggingService
import

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const LeadSalesUpdatePage = () => {
  const [salesList, setSalesList] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [salesPersons, setSalesPersons] = useState([]);
  const [showForm, setShowForm] = useState(false);
  const [editingSaleId, setEditingSaleId] = useState(null);
  const [formData, setFormData] = useState({
    DATE: new Date().toISOString().split('T')[0],
    NAME: '',
    COUNTRY: '',
    COURSE: '',
    CODE: '',
    NUMBER: '',
    'E-MAIL': '',
    'PSUDO ID': '',
    'SALE PERSON': '',
    'LEAD PERSON': '',
    SOURSE: '',
    'CLIENT REMARK': '',
    FEEDBACK: '',
    'TOTAL COST': 0,
    'TOTAL COST CURRENCY': 'USD',
    'TOKEN AMOUNT': 0,
    'TOKEN AMOUNT CURRENCY': 'USD'
  });

  // Get user from auth context
  const { user, loadUser } = useAuth();

  // Log user info right away
  console.log('User object in LeadSalesUpdatePage initial load:', user);
  console.log('User ID formats:', user ? {
    id: user.id,
    _id: user._id,
    userId: user.userId
  } : 'No user data');

  // Try to refresh user data if it's missing
  useEffect(() => {
    if (!user) {
```

```
      console.log('No user data found, attempting to reload user data');
      loadUser();
    } else {
      console.log('User data available:', user);
      console.log('User ID variations:', {
        id: user.id,
        _id: user._id,
        userId: user.userId
      });
    }
  }, []);

  useEffect(() => {
    // Log user information for debugging
    console.log('User information on load:', user);

    // Only try to load data if we have user information
    if (user) {
      loadSalesData();
      loadUsers();
    }
  }, [user]);  // Re-run when user changes

  // Another useEffect to set the initial lead person when user data becomes
available
  useEffect(() => {
    if (user) {
      // Try all possible ID formats
      const userId = user.id || user._id || user.userId;
      console.log('Setting lead person ID to:', userId);

      if (userId) {
        setFormData(prev => ({
          ...prev,
          'LEAD PERSON': userId
        }));
      }
    }
  }, [user]);

  const loadSalesData = async () => {
    try {
      setLoading(true);
      setError(null);

      // Get all lead-related sales (both lead person sales and assigned sales)
      const res = await salesAPI.getLeadSheet();
      setSalesList(res.data.data);
    } catch (err) {
      console.error('Error loading sales data:', err);
      setError('Failed to load sales data. Please try again.');
      toast.error('Failed to load sales data');
    } finally {
      setLoading(false);
    }
  };

  const loadUsers = async () => {
    try {
```

```
    // Get sales persons
    const salesRes = await authAPI.getUsers('Sales Person');
    setSalesPersons(salesRes.data.data || []);
  } catch (err) {
    console.error('Error loading users:', err);
    toast.error('Failed to load sales persons');
  }
};

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

const handleSubmit = async (e) => {
  e.preventDefault();
  setLoading(true);
  setError(null);

  try {
    // Check if user info is available and find the correct ID format
    if (!user) {
      console.error('No user object available');
      throw new Error('User information not available. Please log in again.');
    }

    console.log('Full user object:', user);

    // Try all possible ID formats
    const userId = user.id || user._id || user.userId;

    if (!userId) {
      console.error('No user ID found in any format:', user);
      throw new Error('User ID not available. Please log in again.');
    }

    console.log('Using user ID:', userId);

    const saleData = {
      date: formData.DATE,
      customerName: formData.NAME,
      country: formData.COUNTRY,
      course: formData.COURSE,
      countryCode: formData.CODE,
      contactNumber: formData.NUMBER,
      email: formData['E-MAIL'],
      pseudoId: formData['PSUDO ID'],
      salesPerson: formData['SALE PERSON'],
      totalCost: parseFloat(formData['TOTAL COST']),
      totalCostCurrency: formData['TOTAL COST CURRENCY'],
      tokenAmount: parseFloat(formData['TOKEN AMOUNT']),
      tokenAmountCurrency: formData['TOKEN AMOUNT CURRENCY'],
      leadPerson: userId,
      source: formData.SOURSE,
      clientRemark: formData['CLIENT REMARK'],
      feedback: formData.FEEDBACK,
```

```
  isLeadPersonSale: true // Mark as lead person sale
};

console.log('Sending sale data:', saleData);

let res;

if (editingSaleId) {
  // Update existing sale
  res = await leadPersonSalesAPI.update(editingSaleId, saleData);

  // Log the sale update
  try {
    await LoggingService.logSaleUpdate(editingSaleId, saleData);
  } catch (logError) {
    console.error('Error logging sale update:', logError);
  }

  toast.success('Sale updated successfully');
} else {
  // Create new sale
  res = await leadPersonSalesAPI.create(saleData);

  // Log the sale creation
  try {
    await LoggingService.logSaleCreate(res.data.data);
  } catch (logError) {
    console.error('Error logging sale creation:', logError);
  }

  toast.success('Sale created successfully');
}

if (res.data.success) {
  // Reset form
  setFormData({
    DATE: new Date().toISOString().split('T')[0],
    NAME: '',
    COUNTRY: '',
    COURSE: '',
    CODE: '',
    NUMBER: '',
    'E-MAIL': '',
    'PSUDO ID': '',
    'SALE PERSON': '',
    'LEAD PERSON': userId,
    SOURSE: '',
    'CLIENT REMARK': '',
    FEEDBACK: '',
    'TOTAL COST': 0,
    'TOTAL COST CURRENCY': 'USD',
    'TOKEN AMOUNT': 0,
    'TOKEN AMOUNT CURRENCY': 'USD'
  });

  setShowForm(false);
  setEditingSaleId(null);
  loadSalesData();
}
```

```
    } catch (err) {
      console.error('Error with sale:', err);
      setError(err.response?.data?.message || 'Failed to process sale. Please try
again.');
      toast.error(err.response?.data?.message || 'Failed to process sale');
    } finally {
      setLoading(false);
    }
  };

  // Handle delete sale
  const handleDeleteSale = async (saleId) => {
    if (!window.confirm('Are you sure you want to delete this sale?')) {
      return;
    }

    try {
      setLoading(true);
      await leadPersonSalesAPI.delete(saleId);
      toast.success('Sale deleted successfully');
      loadSalesData();
    } catch (err) {
      console.error('Error deleting sale:', err);
      toast.error('Failed to delete sale');
    } finally {
      setLoading(false);
    }
  };

  // Handle edit sale
  const handleEditSale = (sale) => {
    setEditingSaleId(sale._id);

    // Map the sale data to form fields
    setFormData({
      DATE: new Date(sale.date).toISOString().split('T')[0],
      NAME: sale.customerName || '',
      COUNTRY: sale.country || '',
      COURSE: sale.course || '',
      CODE: sale.countryCode || '',
      NUMBER: sale.contactNumber || '',
      'E-MAIL': sale.email || '',
      'PSUDO ID': sale.pseudoId || '',
      'SALE PERSON': sale.salesPerson?._id || sale.salesPerson || '',
      'LEAD PERSON': sale.leadPerson?._id || sale.leadPerson || '',
      SOURSE: sale.source || '',
      'CLIENT REMARK': sale.clientRemark || '',
      FEEDBACK: sale.feedback || '',
      'TOTAL COST': sale.totalCost || 0,
      'TOTAL COST CURRENCY': sale.totalCostCurrency || 'USD',
      'TOKEN AMOUNT': sale.tokenAmount || 0,
      'TOKEN AMOUNT CURRENCY': sale.tokenAmountCurrency || 'USD'
    });

    setShowForm(true);

    // Scroll to form
    window.scrollTo({
      top: 0,
```

```
      behavior: 'smooth'
    });
  };

  // Reset form and cancel editing
  const handleCancelEdit = () => {
    setEditingSaleId(null);
    setFormData({
      DATE: new Date().toISOString().split('T')[0],
      NAME: '',
      COUNTRY: '',
      COURSE: '',
      CODE: '',
      NUMBER: '',
      'E-MAIL': '',
      'PSUDO ID': '',
      'SALE PERSON': '',
      'LEAD PERSON': user?.id || user?._id || user?.userId || '',
      SOURSE: '',
      'CLIENT REMARK': '',
      FEEDBACK: '',
      'TOTAL COST': 0,
      'TOTAL COST CURRENCY': 'USD',
      'TOKEN AMOUNT': 0,
      'TOKEN AMOUNT CURRENCY': 'USD'
    });
    setShowForm(false);
  };

  return (
    <Layout>
      <div className="container mx-auto p-4">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-2xl font-bold">Lead Sales Management</h1>
          <button
            onClick={() => {
              setEditingSaleId(null);
              setShowForm(!showForm);
            }}
            className="flex items-center px-4 py-2 bg-blue-600 hover:bg-blue-700
dark:bg-blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white rounded-md"
          >
            {showForm ? 'Cancel' : <>
              <FaPlus className="mr-2" /> Add New Sale
            </>}
          </button>
        </div>

        {error && (
          <div className="bg-red-100 border border-red-400 text-red-700 px-4 py-3
rounded relative mb-4">
            {error}
          </div>
        )}

        {showForm && (
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out shadow-md
```

```
dark:shadow-2xl rounded-lg p-6 mb-6 shadow-sm">
            <h2 className="text-xl font-semibold mb-4">
              {editingSaleId ? 'Edit Sale' : 'Add New Sale'}
            </h2>
            <form onSubmit={handleSubmit}>
              <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3
gap-4">
                {/* Date */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Date</label>
                  <input
                    type="date"
                    name="DATE"
                    value={formData.DATE}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                    required
                  />
                </div>

                {/* Customer Name */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Customer Name</label>
                  <input
                    type="text"
                    name="NAME"
                    value={formData.NAME}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                    required
                  />
                </div>

                {/* Country */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Country</label>
                  <input
                    type="text"
                    name="COUNTRY"
                    value={formData.COUNTRY}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                    required
                  />
                </div>

                {/* Course */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Course</label>
                  <input
                    type="text"
                    name="COURSE"
```

```
                    value={formData.COURSE}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                    required
                  />
                </div>

                {/* Country Code */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Country Code</label>
                  <input
                    type="text"
                    name="CODE"
                    value={formData.CODE}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  />
                </div>

                {/* Phone Number */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Phone Number</label>
                  <input
                    type="text"
                    name="NUMBER"
                    value={formData.NUMBER}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                    required
                  />
                </div>

                {/* Email */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Email</label>
                  <input
                    type="email"
                    name="E-MAIL"
                    value={formData['E-MAIL']}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  />
                </div>

                {/* Pseudo ID */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Pseudo ID</label>
                  <input
                    type="text"
                    name="PSUDO ID"
                    value={formData['PSUDO ID']}
```

```
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  />
                </div>

                {/* Sales Person */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Sales Person</label>
                  <select
                    name="SALE PERSON"
                    value={formData['SALE PERSON']}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  >
                    <option value="">Select Sales Person</option>
                    {salesPersons.map(person => (
                      <option key={person._id} value={person._id}>
                        {person.fullName}
                      </option>
                    ))}
                  </select>
                </div>

                {/* Source */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Source</label>
                  <input
                    type="text"
                    name="SOURSE"
                    value={formData.SOURSE}
                    onChange={handleChange}
                    className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  />
                </div>

                {/* Total Cost */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Total Cost</label>
                  <div className="flex">
                    <input
                      type="number"
                      name="TOTAL COST"
                      value={formData['TOTAL COST']}
                      onChange={handleChange}
                      className="w-2/3 p-2 border border-slate-300 dark:border-
slate-600 rounded-l-md"
                      required
                    />
                    <select
                      name="TOTAL COST CURRENCY"
                      value={formData['TOTAL COST CURRENCY']}
                      onChange={handleChange}
                      className="w-1/3 p-2 border border-slate-300 dark:border-
```

```
slate-600 rounded-r-md"
                    >
                      <option value="USD">USD</option>
                      <option value="EUR">EUR</option>
                      <option value="GBP">GBP</option>
                      <option value="INR">INR</option>
                    </select>
                  </div>
                </div>

                {/* Token Amount */}
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Token Amount</label>
                  <div className="flex">
                    <input
                      type="number"
                      name="TOKEN AMOUNT"
                      value={formData['TOKEN AMOUNT']}
                      onChange={handleChange}
                      className="w-2/3 p-2 border border-slate-300 dark:border-
slate-600 rounded-l-md"
                      required
                    />
                    <select
                      name="TOKEN AMOUNT CURRENCY"
                      value={formData['TOKEN AMOUNT CURRENCY']}
                      onChange={handleChange}
                      className="w-1/3 p-2 border border-slate-300 dark:border-
slate-600 rounded-r-md"
                    >
                      <option value="USD">USD</option>
                      <option value="EUR">EUR</option>
                      <option value="GBP">GBP</option>
                      <option value="INR">INR</option>
                    </select>
                  </div>
                </div>
              </div>

              {/* Client Remark */}
              <div className="mt-4">
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Client Remark</label>
                <textarea
                  name="CLIENT REMARK"
                  value={formData['CLIENT REMARK']}
                  onChange={handleChange}
                  rows="2"
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                ></textarea>
              </div>

              {/* Feedback */}
              <div className="mt-4">
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Feedback</label>
                <textarea
```

```
                  name="FEEDBACK"
                  value={formData.FEEDBACK}
                  onChange={handleChange}
                  rows="2"
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                ></textarea>
              </div>

              <div className="mt-6 flex justify-end space-x-4">
                <button
                  type="button"
                  onClick={handleCancelEdit}
                  className="px-4 py-2 bg-gray-300 text-gray-800 dark:text-
gray-200 rounded-md hover:bg-gray-400"
                >
                  Cancel
                </button>
                <button
                  type="submit"
                  disabled={loading}
                  className="px-4 py-2 bg-blue-600 hover:bg-blue-700 dark:bg-
blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white rounded-md disabled:bg-blue-300"
                >
                  {loading ? 'Processing...' : (editingSaleId ? 'Update Sale' :
'Add Sale')}
                </button>
              </div>
            </form>
          </div>
        )}

        {/* Sales List */}
        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out shadow-md
dark:shadow-2xl rounded-lg overflow-hidden shadow-sm">
          <h2 className="text-xl font-semibold p-4 bg-gray-50 dark:bg-slate-800
transition-all duration-200 ease-out">Sales List</h2>

          {loading && !showForm ? (
            <div className="flex justify-center items-center p-8">
              <div className="animate-spin rounded-full h-12 w-12 border-t-2
border-b-2 border-blue-500"></div>
            </div>
          ) : salesList.length === 0 ? (
            <div className="p-6 text-center text-slate-500 dark:text-gray-400">
              No sales found. Add your first sale above.
            </div>
          ) : (
            <div className="overflow-x-auto">
              <table className="min-w-full divide-y divide-slate-200 dark:divide-
slate-700">
                <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
                  <tr>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Date</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
```

```
slate-500 dark:text-gray-400 uppercase tracking-wider">Customer</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Course</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Sales Person</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Total</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Token</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Type</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Actions</th>
                  </tr>
                </thead>
                <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-slate-200 dark:divide-slate-700">
                  {salesList.map(sale => (
                    <tr key={sale._id} className="hover:bg-slate-50 dark:hover:bg-
slate-800 dark:bg-slate-800 transition-all duration-200 ease-out">
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">
                        {new Date(sale.date).toLocaleDateString()}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">
                        <div>{sale.customerName}</div>
                        <div className="text-xs text-slate-500 dark:text-
gray-400">{sale.country}</div>
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">{sale.course}</td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">
                        {sale.salesPerson?.fullName || 'N/A'}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">
                        {sale.totalCost} {sale.totalCostCurrency}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">
                        {sale.tokenAmount} {sale.tokenAmountCurrency}
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm">
                        <span className={`px-2 py-1 rounded-full text-xs
${sale.saleType === 'Lead Person Sale' ? 'bg-blue-100 text-blue-800' : 'bg-
green-100 text-green-800'}`}>
                          {sale.saleType || (sale.isLeadPersonSale ? 'Lead Person
Sale' : 'Sales Person Sale')}
                        </span>
                      </td>
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                        <div className="flex space-x-2">
                          <button
                            onClick={() => handleEditSale(sale)}
                            className="text-indigo-600 hover:text-indigo-900"
                          >
```

```
                            <FaEdit className="h-5 w-5" />
                          </button>
                          {/* Only allow deleting lead person sales */}
                          {(!sale.saleType || sale.saleType === 'Lead Person
Sale' || sale.isLeadPersonSale) && (
                            <button
                              onClick={() => handleDeleteSale(sale._id)}
                              className="text-red-600 hover:text-red-900"
                            >
                              <FaTrash className="h-5 w-5" />
                            </button>
                          )}
                        </div>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          )}
        </div>
      </div>
    </Layout>
  );
};

export default LeadSalesUpdatePage;
```

src/pages/LeadsPage.jsx

```
// src/pages/LeadsPage.jsxÐ
import React, { useState, useEffect, useCallback } from "react";Ð
import { Link } from "react-router-dom";Ð
import { leadsAPI } from "../services/api";Ð
import { useAuth } from "../context/AuthContext";Ð
import LeadForm from "../components/Leads/LeadForm";Ð
import Layout from "../components/Layout/Layout";Ð
import AOS from 'aos';Ð
import 'aos/dist/aos.css';Ð
import { FaEdit, FaTrash, FaFilter, FaPlus } from 'react-icons/fa';Ð
import { toast } from 'react-hot-toast';Ð
import LoggingService from '../services/loggingService'; // Add LoggingService
importÐ
Ð
import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';Ð
Ð
const LeadsPage = () => {Ð
  const { user } = useAuth();Ð
  const [leads, setLeads] = useState([]);Ð
  const [filteredLeads, setFilteredLeads] = useState([]);Ð
  const [loading, setLoading] = useState(true);Ð
  const [error, setError] = useState(null);Ð
  const [showAddForm, setShowAddForm] = useState(false);Ð
  const [selectedLead, setSelectedLead] = useState(null);Ð
  Ð
  // Date filtering state - Default to current month instead of March 2025Ð
  const currentDate = new Date();Ð
```

```
  const [filterMonth, setFilterMonth] = useState(currentDate.getMonth() + 1); //
Current monthÐ
  const [filterYear, setFilterYear] = useState(currentDate.getFullYear()); //
Current yearÐ
  const [showCurrentMonth, setShowCurrentMonth] = useState(true); // Show current
month by defaultÐ
  Ð
  // Generate month optionsÐ
  const months = [Ð
    { value: 1, label: "January" },Ð
    { value: 2, label: "February" },Ð
    { value: 3, label: "March" },Ð
    { value: 4, label: "April" },Ð
    { value: 5, label: "May" },Ð
    { value: 6, label: "June" },Ð
    { value: 7, label: "July" },Ð
    { value: 8, label: "August" },Ð
    { value: 9, label: "September" },Ð
    { value: 10, label: "October" },Ð
    { value: 11, label: "November" },Ð
    { value: 12, label: "December" }Ð
  ];Ð
  Ð
  // Generate year options (include current year + 1 and 5 years back)Ð
  const currentYear = new Date().getFullYear();Ð
  const years = Array.from({ length: 7 }, (_, i) => currentYear + 1 - i); //
Include next year and 5 years backÐ
  Ð
  // Function to fetch leads from the APIÐ
  const fetchLeads = useCallback(async () => {Ð
    try {Ð
      setLoading(true);Ð
      console.log('Current user:', user);Ð
      Ð
      // EMERGENCY FIX: Remove date filtering to show all leadsÐ
      console.log('Fetching ALL leads without date filters');Ð
      const response = await leadsAPI.getAll({});Ð
      console.log('API Response:', response.data);Ð
      Ð
      // Log the structure of the first lead to see available fieldsÐ
      if (response.data && response.data.data && response.data.data.length > 0) {Ð
        const firstLead = response.data.data[0];Ð
        console.log('First lead object structure:', firstLead);Ð
        console.log('All field names:', Object.keys(firstLead));Ð
        console.log('Name value:', firstLead.name);Ð
        console.log('NAME value:', firstLead.NAME);Ð
      }Ð
      Ð
      // Set leads directly from API (no frontend filtering needed)Ð
      setLeads(response.data.data || []);Ð
      setFilteredLeads(response.data.data || []);Ð
      console.log('Leads set in state:', response.data.data);Ð
      setError(null);Ð
    } catch (err) {Ð
      console.error("Error fetching leads:", err);Ð
      setError("Failed to load leads. Please try again.");Ð
      // Set empty arrays to prevent undefined errorsÐ
      setLeads([]);Ð
      setFilteredLeads([]);Ð
```

```
    } finally {Đ
      setLoading(false);Đ
    }Đ
  }, [user]);Đ
  Đ
  // Fetch leads on component mount onlyĐ
  useEffect(() => {Đ
    console.log('Component mounted, fetching all leads');Đ
    fetchLeads();Đ
  }, [fetchLeads]);Đ
  Đ
  // Filter leads based on selected month and yearĐ
  useEffect(() => {Đ
    if (leads.length === 0) return;Đ
    Đ
    console.log('Filtering leads for:', { filterMonth, filterYear,
showCurrentMonth });Đ
    Đ
    let filtered = leads;Đ
    Đ
    if (showCurrentMonth) {Đ
      // Filter for current monthĐ
      const currentDate = new Date();Đ
      const currentMonth = currentDate.getMonth() + 1;Đ
      const currentYear = currentDate.getFullYear();Đ
      Đ
      filtered = leads.filter(lead => {Đ
        const leadDate = new Date(lead.createdAt);Đ
        const leadMonth = leadDate.getMonth() + 1;Đ
        const leadYear = leadDate.getFullYear();Đ
        return leadMonth === currentMonth && leadYear === currentYear;Đ
      });Đ
    } else {Đ
      // Filter for selected month/yearĐ
      filtered = leads.filter(lead => {Đ
        const leadDate = new Date(lead.createdAt);Đ
        const leadMonth = leadDate.getMonth() + 1;Đ
        const leadYear = leadDate.getFullYear();Đ
        return leadMonth === filterMonth && leadYear === filterYear;Đ
      });Đ
    }Đ
    Đ
    console.log(`Filtered ${filtered.length} leads from ${leads.length} total
leads`);Đ
    setFilteredLeads(filtered);Đ
  }, [leads, filterMonth, filterYear, showCurrentMonth]);Đ
  Đ
  // Note: Date filtering is now handled on the backend via API query parametersĐ
  // This ensures sales persons only receive leads for the selected time periodĐ
  // instead of loading all leads and filtering on frontendĐ
  Đ
  // Debug function to log current stateĐ
  const debugCurrentState = useCallback(() => {Đ
    console.log('============= DEBUG CURRENT STATE =============');Đ
    console.log('Current leads count:', leads.length);Đ
    console.log('Current filteredLeads count:', filteredLeads.length);Đ
    console.log('Filter settings:', { filterMonth, filterYear,
showCurrentMonth });Đ
    console.log('Selected lead:', selectedLead);Đ
```

```
    console.log('Current date:', new Date());Ð
    console.log('Filter date range:', {Ð
      start: showCurrentMonth Ð
        ? new Date(new Date().getFullYear(), new Date().getMonth(), 1)Ð
        : new Date(filterYear, filterMonth - 1, 1),Ð
      end: showCurrentMonthÐ
        ? new Date(new Date().getFullYear(), new Date().getMonth() + 1, 0)Ð
        : new Date(filterYear, filterMonth, 0)Ð
    });Ð
    console.log('First 3 leads in leads array:', leads.slice(0, 3).map(l => ({ Ð
      id: l._id, Ð
      name: l.name, Ð
      createdAt: l.createdAt,Ð
      createdAtFormatted: new Date(l.createdAt).toLocaleDateString(),Ð
      month: new Date(l.createdAt).getMonth() + 1,Ð
      year: new Date(l.createdAt).getFullYear()Ð
    })));Ð
    console.log('First 3 leads in filteredLeads array:', filteredLeads.slice(0,
3).map(l => ({ Ð
      id: l._id, Ð
      name: l.name, Ð
      createdAt: l.createdAt,Ð
      createdAtFormatted: new Date(l.createdAt).toLocaleDateString(),Ð
      month: new Date(l.createdAt).getMonth() + 1,Ð
      year: new Date(l.createdAt).getFullYear()Ð
    })));Ð
    console.log('============= END DEBUG STATE =============');Ð
  }, [leads, filteredLeads, filterMonth, filterYear, showCurrentMonth,
selectedLead]);Ð
  Ð
  // Force refetch function that bypasses all cachingÐ
  const forceRefetch = useCallback(async () => {Ð
    console.log('=== FORCE REFETCH TRIGGERED ===');Ð
    setLoading(true);Ð
    try {Ð
      // Clear current state firstÐ
      setLeads([]);Ð
      setFilteredLeads([]);Ð
      Ð
      // Wait a moment to ensure state is clearedÐ
      await new Promise(resolve => setTimeout(resolve, 100));Ð
      Ð
      // Fetch fresh dataÐ
      await fetchLeads();Ð
      Ð
      toast.success('Data refreshed successfully!');Ð
    } catch (error) {Ð
      console.error('Force refetch failed:', error);Ð
      toast.error('Failed to refresh data');Ð
    } finally {Ð
      setLoading(false);Ð
    }Ð
  }, [fetchLeads]);Ð
  Ð
  // Function to handle successful lead creation/updateÐ
  const handleLeadSuccess = useCallback(async (lead) => {Ð
    console.log('============= HANDLE LEAD SUCCESS =============');Ð
    console.log('Lead success callback called with:', lead);Ð
    Ð
```

```
  if (selectedLead) {Đ
    console.log('Lead updated successfully');Đ
    Đ
    // Log lead updateĐ
    try {Đ
      await LoggingService.logLeadUpdate(lead._id, lead);Đ
    } catch (logError) {Đ
      console.error('Error logging lead update:', logError);Đ
    }Đ
    Đ
    setSelectedLead(null);Đ
    Đ
    // Show success message immediatelyĐ
    toast.success('Lead updated successfully!');Đ
    Đ
    // Simple solution: Just refetch the dataĐ
    fetchLeads();Đ
    Đ
  } else {Đ
    // Add new leadĐ
    console.log('Adding new lead to list');Đ
    Đ
    // Log lead creationĐ
    try {Đ
      await LoggingService.logLeadCreate(lead);Đ
    } catch (logError) {Đ
      console.error('Error logging lead creation:', logError);Đ
    }Đ
    Đ
    setShowAddForm(false);Đ
    fetchLeads();Đ
    toast.success('Lead created successfully!');Đ
  }Đ
  Đ
  console.log('============= END HANDLE LEAD SUCCESS =============');Đ
}, [selectedLead, fetchLeads]);Đ
Đ
// Handle month changeĐ
const handleMonthChange = (e) => {Đ
  const newMonth = parseInt(e.target.value);Đ
  console.log("Changing month to:", newMonth);Đ
  setFilterMonth(newMonth);Đ
  setShowCurrentMonth(false);Đ
};Đ
Đ
// Handle year changeĐ
const handleYearChange = (e) => {Đ
  const newYear = parseInt(e.target.value);Đ
  console.log("Changing year to:", newYear);Đ
  setFilterYear(newYear);Đ
  setShowCurrentMonth(false);Đ
};Đ
Đ
// Handle reset to current monthĐ
const handleResetToCurrentMonth = () => {Đ
  setFilterMonth(new Date().getMonth() + 1);Đ
  setFilterYear(new Date().getFullYear());Đ
  setShowCurrentMonth(true);Đ
};Đ
```

```
  Đ
  // Format date for displayĐ
  const formatDate = (dateString) => {Đ
    const date = new Date(dateString);Đ
    return date.toLocaleDateString();Đ
  };Đ
Đ
  // Add AOS initialization in useEffectĐ
  useEffect(() => {Đ
    AOS.init({Đ
      duration: 600,Đ
      easing: 'ease-in-out',Đ
      once: trueĐ
    });Đ
  }, []);Đ
Đ
  // Handle edit button clickĐ
  const handleEditClick = (lead) => {Đ
    console.log("Edit clicked for lead:", lead);Đ
    setSelectedLead(lead);Đ
  };Đ
Đ
  // Function to analyze date ranges in leads dataĐ
  const analyzeDateRanges = (leadsData) => {Đ
    if (!leadsData || leadsData.length === 0) return null;Đ
    Đ
    const dateRanges = {};Đ
    leadsData.forEach(lead => {Đ
      const date = new Date(lead.createdAt);Đ
      const monthYear = `${date.getMonth() + 1}/${date.getFullYear()}`;Đ
      const monthName = months[date.getMonth()].label;Đ
      const year = date.getFullYear();Đ
      Đ
      if (!dateRanges[monthYear]) {Đ
        dateRanges[monthYear] = {Đ
          month: date.getMonth() + 1,Đ
          year: year,Đ
          monthName: monthName,Đ
          count: 0,Đ
          leads: []Đ
        };Đ
      }Đ
      dateRanges[monthYear].count++;Đ
      dateRanges[monthYear].leads.push(lead);Đ
    });Đ
    Đ
    return dateRanges;Đ
  };Đ
Đ
  // Function to get date range suggestionsĐ
  const getDateRangeSuggestions = () => {Đ
    const ranges = analyzeDateRanges(leads);Đ
    if (!ranges) return null;Đ
    Đ
    const sortedRanges = Object.values(ranges).sort((a, b) => {Đ
      if (a.year !== b.year) return b.year - a.year;Đ
      return b.month - a.month;Đ
    });Đ
    Đ
```

```
    return sortedRanges.slice(0, 3); // Top 3 date rangesÐ
  };Ð
Ð
  // Add logging for lead assignment if there's a function that handles itÐ
  const handleLeadAssign = async (leadId, newAssignee) => {Ð
    try {Ð
      const lead = leads.find(l => l._id === leadId);Ð
      const previousAssignee = lead?.assignedTo;Ð
      Ð
      const response = await leadsAPI.update(leadId, { assignedTo: newAssignee });Ð
      Ð
      if (response.data.success) {Ð
        // Log the lead assignmentÐ
        try {Ð
          await LoggingService.logLeadAssign(leadId, newAssignee,
previousAssignee);Ð
        } catch (logError) {Ð
          console.error('Error logging lead assignment:', logError);Ð
        }Ð
        Ð
        toast.success('Lead assigned successfully');Ð
        fetchLeads();Ð
      }Ð
    } catch (error) {Ð
      console.error('Error assigning lead:', error);Ð
      toast.error('Failed to assign lead');Ð
    }Ð
  };Ð
Ð
  return (Ð
    <Layout>Ð
      <div className="bg-gray-50 dark:bg-slate-800 transition-all duration-200
ease-out min-h-screen pb-12">Ð
        {/* Header with gradient background */}Ð
        <div className="bg-gradient-to-r from-blue-600 to-indigo-700 p-6 mb-8">Ð
          <div className="container mx-auto">Ð
            <div className="flex flex-col md:flex-row justify-between items-
center">Ð
              <div>Ð
                <h1 className="text-3xl font-bold text-white mb-2">Leads
Management</h1>Ð
                <p className="text-blue-100 text-sm md:text-base">Ð
                  {filteredLeads.length} leads • {showCurrentMonth ? 'Current
Month' : `${months.find(m => m.value === parseInt(filterMonth))?.label}
${filterYear}`}Ð
                </p>Ð
              </div>Ð
              Ð
              <buttonÐ
                onClick={() => setShowAddForm(true)}Ð
                className="mt-4 md:mt-0 bg-white dark:bg-slate-900 border border-
slate-200 dark:border-slate-700 transition-all duration-200 ease-out duration-300
flex items-center font-medium shadow-sm"Ð
              >Ð
                <FaPlus className="h-4 w-4 mr-2" />Ð
                Add New LeadÐ
              </button>Ð
            </div>Ð
          </div>Ð
```

```
        </div>Đ
        Đ
        <div className="container mx-auto px-4">Đ
          {/* Error message */}Đ
          {error && (Đ
            <div className="mb-6 p-4 bg-red-50 text-red-700 border border-red-200
rounded-lg shadow-sm dark:shadow-black/25">Đ
              <div className="flex items-center">Đ
                <svg xmlns="http://www.w3.org/2000/svg" className="h-5 w-5 mr-2"
viewBox="0 0 20 20" fill="currentColor">Đ
                  <path fillRule="evenodd" d="M18 10a8 8 0 11-16 0 8 8 0 0116
0zm-7 4a1 1 0 11-2 0 1 1 0 012 0zm-1-9a1 1 0 00-1 1v4a1 1 0 102 0V6a1 1 0
00-1-1z" clipRule="evenodd" />Đ
                </svg>Đ
                {error}Đ
              </div>Đ
            </div>Đ
          )}Đ
          Đ
          {/* Add/Edit Lead Form */}Đ
          {(showAddForm || selectedLead) && (Đ
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-xl shadow-md dark:shadow-black/25 overflow-hidden
mb-8">Đ
              <div className="px-6 py-4 bg-blue-50 border-b border-blue-100">Đ
                <h2 className="text-xl font-bold text-gray-800 dark:text-
gray-200">Đ
                  {selectedLead ? "Edit Lead" : "Add New Lead"}Đ
                </h2>Đ
              </div>Đ
              <div className="p-6">Đ
                <LeadForm Đ
                  lead={selectedLead}Đ
                  onSuccess={handleLeadSuccess}Đ
                />Đ
                Đ
                <buttonĐ
                  onClick={() => {Đ
                    setShowAddForm(false);Đ
                    setSelectedLead(null);Đ
                  }}Đ
                  className="mt-6 text-blue-600 hover:text-blue-800 font-medium
flex items-center"Đ
                >Đ
                  <svg xmlns="http://www.w3.org/2000/svg" className="h-5 w-5
mr-1" viewBox="0 0 20 20" fill="currentColor">Đ
                    <path fillRule="evenodd" d="M9.707 16.707a1 1 0 01-1.414
0l-6-6a1 1 0 010-1.414l6-6a1 1 0 011.414 1.414L5.414 9H17a1 1 0 110 2H5.414l4.293
4.293a1 1 0 010 1.414z" clipRule="evenodd" />Đ
                  </svg>Đ
                  Back to Leads ListĐ
                </button>Đ
              </div>Đ
            </div>Đ
          )}Đ
          Đ
          {/* Leads List */}Đ
          {!showAddForm && !selectedLead && (Đ
            <>Đ
```

```
              {/* Date Filter Controls */}Ð
              <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-xl shadow-md dark:shadow-black/25 overflow-hidden
mb-8">Ð
                <div className="px-6 py-4 bg-gray-50 dark:bg-slate-800 transition-
all duration-200 ease-out border-b border-gray-100 flex justify-between items-
center">Ð
                  <h3 className="text-lg font-bold text-gray-800 dark:text-
gray-200">Filter Leads by Month</h3>Ð
                  <FaFilter className="text-slate-500 dark:text-gray-400" />Ð
                </div>Ð
                <div className="p-6">Ð
                  <div className="grid grid-cols-1 md:grid-cols-3 gap-6">Ð
                    <div>Ð
                      <label htmlFor="month" className="block text-sm font-medium
text-slate-700 dark:text-slate-300 mb-1">Month</label>Ð
                      <selectÐ
                        id="month"Ð
                        value={filterMonth}Ð
                        onChange={handleMonthChange}Ð
                        className="w-full border border-slate-300 dark:border-
slate-600 rounded-lg p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"Ð
                      >Ð
                        {months.map(month => (Ð
                          <option key={month.value} value={month.value}>Ð
                            {month.label}Ð
                          </option>Ð
                        ))}Ð
                      </select>Ð
                    </div>Ð
                    Ð
                    <div>Ð
                      <label htmlFor="year" className="block text-sm font-medium
text-slate-700 dark:text-slate-300 mb-1">Year</label>Ð
                      <selectÐ
                        id="year"Ð
                        value={filterYear}Ð
                        onChange={handleYearChange}Ð
                        className="w-full border border-slate-300 dark:border-
slate-600 rounded-lg p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"Ð
                      >Ð
                        {years.map(year => (Ð
                          <option key={year} value={year}>Ð
                            {year}Ð
                          </option>Ð
                        ))}Ð
                      </select>Ð
                    </div>Ð
                    Ð
                    <div className="flex items-end">Ð
                      <div className="grid grid-cols-1 gap-2 w-full">Ð
                        <buttonÐ
                          onClick={handleResetToCurrentMonth}Ð
                          className="w-full py-2 px-4 bg-blue-600 hover:bg-
blue-700 dark:bg-blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl
hover:shadow-md transition-all duration-200 text-white rounded-lg focus:outline-
none focus:ring-2 focus:ring-blue-500 dark:focus:ring-offset-slate-900 focus:ring-
offset-2"Ð
```

```
                        >Ð
                          Show Current MonthÐ
                        </button>Ð
                        <buttonÐ
                          onClick={() => {Ð
                            setFilterMonth(3);Ð
                            setFilterYear(2025);Ð
                            setShowCurrentMonth(false);Ð
                          }}Ð
                          className="w-full py-2 px-4 bg-green-600 text-white
rounded-lg hover:bg-green-700 focus:outline-none focus:ring-2 focus:ring-
green-500 focus:ring-offset-2 text-sm"Ð
                        >Ð
                          Ø=ÜÁ March 2025 (Imported Leads)Ð
                        </button>Ð
                      </div>Ð
                    </div>Ð
                  </div>Ð
                  Ð
                  <div className="mt-4 p-3 bg-blue-50 rounded-lg text-sm text-
blue-800 flex items-center">Ð
                    <svg xmlns="http://www.w3.org/2000/svg" className="h-5 w-5
text-blue-500 mr-2" viewBox="0 0 20 20" fill="currentColor">Ð
                      <path fillRule="evenodd" d="M18 10a8 8 0 11-16 0 8 8 0 0116
0zm-7-4a1 1 0 11-2 0 1 1 0 012 0zM9 9a1 1 0 000 2v3a1 1 0 001 1h1a1 1 0
100-2v-3a1 1 0 00-1-1H9z" clipRule="evenodd" />Ð
                    </svg>Ð
                    <span>Showing {filteredLeads.length} leads for
{showCurrentMonth ? 'current month' : `${months.find(m => m.value ===
parseInt(filterMonth))?.label} ${filterYear}`}</span>Ð
                  </div>Ð
                </div>Ð
              </div>Ð
              Ð
              {/* Leads Cards */}Ð
              <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out rounded-xl shadow-md dark:shadow-black/25 overflow-hidden">Ð
                <div className="px-6 py-4 bg-gray-50 dark:bg-slate-800 transition-
all duration-200 ease-out border-b border-gray-100">Ð
                  <h3 className="text-lg font-bold text-gray-800 dark:text-
gray-200">Leads Overview ({filteredLeads.length})</h3>Ð
                </div>Ð
                Ð
                {loading ? (Ð
                  <div className="p-12 flex justify-center">Ð
                    <div className="animate-spin rounded-full h-12 w-12 border-
b-2 border-blue-600"></div>Ð
                    <p className="ml-4 text-gray-600 dark:text-gray-500">Loading
leads...</p>Ð
                  </div>Ð
                ) : filteredLeads.length === 0 ? (Ð
                  <div className="p-12 text-center">Ð
                    <svg xmlns="http://www.w3.org/2000/svg" className="h-16 w-16
mx-auto text-gray-300 dark:text-gray-500 mb-4" fill="none" viewBox="0 0 24 24"
stroke="currentColor">Ð
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={1} d="M9 12h6m-6 4h6m2 5H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0
01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z" />Ð
                    </svg>Ð
```

```
                    <h3 className="text-xl font-medium text-slate-700 dark:text-
slate-300 mb-2">No leads found</h3>Ð
                    <p className="text-slate-500 dark:text-gray-400 mb-6">No
leads were found for the selected time period</p>Ð
                    <buttonÐ
                      onClick={() => setShowAddForm(true)}Ð
                      className="inline-flex items-center px-4 py-2 border border-
transparent text-sm font-medium rounded-md shadow-sm dark:shadow-xl text-white bg-
blue-600 hover:bg-blue-700 dark:bg-blue-500 dark:hover:bg-blue-600 hover:shadow-
md transition-all duration-200 focus:outline-none focus:ring-2 focus:ring-
offset-2 focus:ring-blue-500 dark:focus:ring-offset-slate-900"Ð
                    >Ð
                      <svg xmlns="http://www.w3.org/2000/svg" className="h-5 w-5
mr-2" viewBox="0 0 20 20" fill="currentColor">Ð
                        <path fillRule="evenodd" d="M10 5a1 1 0 011 1v3h3a1 1 0
110 2h-3v3a1 1 0 11-2 0v-3H6a1 1 0 110-2h3V6a1 1 0 011-1z" clipRule="evenodd" />Ð
                      </svg>Ð
                      Add Your First LeadÐ
                    </button>Ð
                  </div>Ð
                ) : (Ð
                  <div className="p-6">Ð
                    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-
cols-3 gap-6">Ð
                      {filteredLeads.map((lead) => (Ð
                        <div key={lead._id} className="bg-white dark:bg-slate-900
border border-slate-200 dark:border-slate-700 transition-all duration-200 ease-
out-shadow shadow-sm">Ð
                          <div className="flex justify-between items-start mb-4">Ð
                            <div className="flex-1">Ð
                              <h3 className="text-lg font-semibold text-slate-900
dark:text-slate-100 mb-1">{lead.name}</h3>Ð
                                <span className={`px-2 py-1 inline-flex text-xs
leading-5 font-semibold rounded-full ${Ð
                                  lead.feedback === 'Converted' Ð
                                    ? 'bg-green-100 text-green-800' Ð
                                    : lead.feedback === 'Not Interested' Ð
                                      ? 'bg-red-100 text-red-800'Ð
                                      : 'bg-yellow-100 text-yellow-800'Ð
                                }`}>Ð
                                  {lead.feedback || 'Pending'}Ð
                                </span>Ð
                              </div>Ð
                              <buttonÐ
                                onClick={() => setSelectedLead(lead)}Ð
                                className="ml-2 inline-flex items-center justify-
center w-8 h-8 text-indigo-600 hover:text-white hover:bg-indigo-600 rounded-full
transition-colors duration-200"Ð
                                title="Edit Lead"Ð
                              >Ð
                                <FaEdit className="h-4 w-4" />Ð
                              </button>Ð
                            </div>Ð
                            Ð
                            <div className="space-y-2 text-sm text-gray-600
dark:text-gray-500">Ð
                              <div className="flex items-center">Ð
                                <span className="font-medium text-slate-700
dark:text-slate-300 w-16">Course:</span>Ð
```

```
                              <span>{lead.course}</span>Ð
                            </div>Ð
                            <div className="flex items-center">Ð
                              <span className="font-medium text-slate-700
dark:text-slate-300 w-16">Phone:</span>Ð
                              <span>{lead.phone || lead.countryCode}</span>Ð
                            </div>Ð
                            {lead.email && (Ð
                              <div className="flex items-center">Ð
                                <span className="font-medium text-slate-700
dark:text-slate-300 w-16">Email:</span>Ð
                                <span className="truncate">{lead.email}</span>Ð
                              </div>Ð
                            )}Ð
                            <div className="flex items-center">Ð
                              <span className="font-medium text-slate-700
dark:text-slate-300 w-16">Country:</span>Ð
                              <span>{lead.country || lead.countryCode}</span>Ð
                            </div>Ð
                            <div className="flex items-center">Ð
                              <span className="font-medium text-slate-700
dark:text-slate-300 w-16">Date:</span>Ð
                              <span>{formatDate(lead.createdAt)}</span>Ð
                            </div>Ð
                            {lead.assignedTo && (Ð
                              <div className="flex items-center">Ð
                                <span className="font-medium text-slate-700
dark:text-slate-300 w-16">Assigned:</span>Ð
                                <span className="text-blue-600">Ð
                                  {typeof lead.assignedTo === 'object' Ð
                                    ? lead.assignedTo.fullName ||
lead.assignedTo.name || 'Unknown User'Ð
                                    : lead.assignedToÐ
                                  }Ð
                                </span>Ð
                              </div>Ð
                            )}Ð
                          </div>Ð
                        </div>Ð
                      ))}Ð
                    </div>Ð
                  </div>Ð
                )}Ð
              </div>Ð
            </>Ð
          )}Ð
        </div>Ð
      </div>Ð
    </Layout>Ð
  );Ð
};Ð
Ð
export default LeadsPage;Ð
```

src/pages/ManagementContactsPage.jsx

```
import React from "react";
import Layout from "../components/Layout/Layout";
```

```
import { FaEnvelope, FaPhoneAlt, FaLinkedin, FaUser, FaUserTie, FaCode,
FaHeadset } from "react-icons/fa";

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const ManagementContactsPage = () => {
  // Management contacts data
  const contacts = [
    {
      id: 1,
      name: "Parichay Singh",
      position: "CEO & Founder",
      email: "sales@traincapetech.info",
      phone: "+91 6280281505",
      linkedin: "https://www.linkedin.com/in/parichay-singh-rana?lipi=urn%3Ali%3Ap
age%3Ad_flagship3_profile_view_base_contact_details%3B%2BLVd4W1EQj2KhxGQFwlQuQ%3D%
3D",
      image: "https://i.pravatar.cc/300?img=11",
      icon: <FaUserTie className="text-3xl text-blue-700" />
    },
    {
      id: 2,
      name: "Shivam Singh",
      position: "Operations Manager",
      email: "shivam@traincapetech.in",
      phone: "+91 9354364160",
      // linkedin: "https://linkedin.com/in/janesmith",
      image: "https://i.pravatar.cc/300?img=13",
      icon: <FaUser className="text-3xl text-green-600" />
    },
    {
      id: 3,
      name: "Saurav Kumar",
      position: "Technical Lead",
      email: "saurav@traincapetech.in",
      phone: "+62 852-8223-3601",
      // linkedin: "https://linkedin.com/in/michaeljohnson",
      image: "https://i.pravatar.cc/300?img=12",
      icon: <FaCode className="text-3xl text-purple-600" />
    },
    // {
    //   id: 4,
    //   name: "Lisa Brown",
    //   position: "Customer Support Manager",
    //   email: "support@traincapetech.com",
    //   phone: "+1 (123) 456-7893",
    //   linkedin: "https://linkedin.com/in/lisabrown",
    //   image: "https://i.pravatar.cc/300?img=10",
    //   icon: <FaHeadset className="text-3xl text-red-600" />
    // }
  ];

  return (
    <Layout>
      <div className="bg-gradient-to-r from-blue-600 to-indigo-700 py-10">
        <div className="container mx-auto px-4 text-center text-white">
          <h1 className="text-3xl md:text-4xl font-bold mb-4">Management
Contacts</h1>
          <p className="text-lg max-w-2xl mx-auto">
```

```
            Meet our leadership team at TrainCape Technology. Reach out to us
directly for any questions or inquiries.
          </p>
        </div>
      </div>

      <div className="container mx-auto px-4 py-12">
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-8">
          {contacts.map((contact) => (
            <div
              key={contact.id}
              className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out-transform hover:-
translate-y-1 hover:shadow-xl shadow-sm"
            >
              <div className="bg-gray-100 dark:bg-slate-700 p-6 flex justify-
center">
                {contact.image ? (
                  <img
                    src={contact.image}
                    alt={contact.name}
                    className="h-32 w-32 rounded-full object-cover border-4
border-white shadow-md dark:shadow-black/25"
                  />
                ) : (
                  <div className="h-32 w-32 rounded-full bg-blue-100 flex items-
center justify-center">
                    {contact.icon || <FaUser className="text-4xl text-blue-600" /
>}
                  </div>
                )}
              </div>

              <div className="p-6">
                <h3 className="text-xl font-bold text-gray-800 dark:text-
gray-200">{contact.name}</h3>
                <p className="text-blue-600 font-medium mb-4">{contact.position}</
p>

                <div className="space-y-3">
                  <div className="flex items-center">
                    <FaEnvelope className="text-blue-600 mr-3" />
                    <a
                      href={`mailto:${contact.email}`}
                      className="text-slate-700 dark:text-slate-300 hover:text-
blue-600 transition"
                    >
                      {contact.email}
                    </a>
                  </div>
                  <div className="flex items-center">
                    <FaPhoneAlt className="text-blue-600 mr-3" />
                    <a
                      href={`tel:${contact.phone}`}
                      className="text-slate-700 dark:text-slate-300 hover:text-
blue-600 transition"
                    >
                      {contact.phone}
                    </a>
```

```
                </div>
                <div className="flex items-center">
                  <FaLinkedin className="text-blue-600 mr-3" />
                  <a
                    href={contact.linkedin}
                    target="_blank"
                    rel="noopener noreferrer"
                    className="text-slate-700 dark:text-slate-300 hover:text-
blue-600 transition"
                  >
                    LinkedIn Profile
                  </a>
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>

      <div className="mt-16 bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-lg
dark:shadow-lg p-8 shadow-sm">
        <h2 className="text-2xl font-bold text-gray-800 dark:text-gray-200
mb-6">General Contact Information</h2>

        <div className="grid grid-cols-1 md:grid-cols-2 gap-8">
          <div>
            <h3 className="text-lg font-bold text-slate-700 dark:text-slate-300
mb-3">Main Office</h3>
            <p className="text-gray-600 dark:text-gray-500 mb-1">Khandolia
Plaza, 118C, Dabri - Palam Rd, Vaishali, Vaishali Colony, Dashrath Puri</p>
            <p className="text-gray-600 dark:text-gray-500 mb-1">New Delhi,
Delhi, 110045</p>
            <p className="text-gray-600 dark:text-gray-500">India</p>
          </div>

          <div>
            <h3 className="text-lg font-bold text-slate-700 dark:text-slate-300
mb-3">Contact Details</h3>
            <p className="text-gray-600 dark:text-gray-500 mb-1">
              <span className="font-medium">General Inquiries:</span>
sales@traincapetech.in
            </p>
            <p className="text-gray-600 dark:text-gray-500 mb-1">
              <span className="font-medium">Customer Support:</span>
sales@traincapetech.in
            </p>
            <p className="text-gray-600 dark:text-gray-500 mb-1">
              <span className="font-medium">Phone:</span> +91 6280281505
            </p>
            <p className="text-gray-600 dark:text-gray-500">
              <span className="font-medium">Working Hours:</span> Monday-
Saturday, 11AM-7PM IST
            </p>
          </div>
        </div>

        <div className="mt-8">
          <h3 className="text-lg font-bold text-slate-700 dark:text-slate-300
```

```
mb-3">Send Us a Message</h3>
            <div className="bg-blue-50 border-l-4 border-blue-500 p-4">
              <p className="text-blue-700">
                For any questions or inquiries, please feel free to reach out to
our team through the contact information provided above.
                We strive to respond to all messages within 24 business hours.
              </p>
            </div>
          </div>
        </div>
      </div>
    </Layout>
  );
};

export default ManagementContactsPage;
```

src/pages/ManagerDashboard.jsx

```
import React, { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import Layout from '../components/Layout/Layout';
import { authAPI, salesAPI } from '../services/api';
import { FaEdit, FaTrash, FaPlus, FaTimes, FaLock, FaChartLine, FaUsers,
FaFileAlt, FaMoneyBillWave } from 'react-icons/fa';
import toast from 'react-hot-toast';
import { useAuth } from '../context/AuthContext';
import axios from 'axios';
import { getDirectSalesCount } from '../utils/helpers';

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const ManagerDashboard = () => {
  const { user: currentUser } = useAuth();
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [showModal, setShowModal] = useState(false);
  const [editUser, setEditUser] = useState(null);
  const [formData, setFormData] = useState({
    fullName: '',
    email: '',
    password: '',
    role: 'Sales Person'
  });

  // Add dashboard stats
  const [stats, setStats] = useState({
    totalUsers: 0,
    totalSales: 0,
    totalLeads: 0,
    userCounts: {
      salesPerson: 0,
      leadPerson: 0,
      manager: 0,
      admin: 0
    }
  });
```

```
  // Check if current user is admin
  const isAdmin = currentUser?.role === 'Admin';

  useEffect(() => {
    fetchUsers();
    fetchDashboardStats();
  }, []);

  // New function to fetch dashboard stats
  const fetchDashboardStats = async () => {
    try {
      // Try to get direct sales count first
      let salesCount = 0;
      let leadCount = 0;

      // NEW APPROACH: Use direct sales count utility
      try {
        salesCount = await getDirectSalesCount();
        console.log("Manager Dashboard - Got direct sales count:", salesCount);
      } catch (directCountError) {
        console.error("Manager Dashboard - Error getting direct sales count:",
directCountError);
      }

      // Also fetch leads count
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const apiUrl = isDevelopment ? 'http://localhost:8080' : 'https://crm-
backend-o36v.onrender.com/api';
      const leadsResponse = await axios.get(`${apiUrl}${isDevelopment ? '/api' :
''}/leads`, {
        headers: {
          'Authorization': `Bearer ${localStorage.getItem('token')}`,
          'Content-Type': 'application/json'
        }
      });

      if (leadsResponse.data && leadsResponse.data.data &&
Array.isArray(leadsResponse.data.data)) {
        leadCount = leadsResponse.data.data.length;
      }

      console.log("Final counts - Sales:", salesCount, "Leads:", leadCount);

      // Update stats with the fetched counts
      setStats(prev => ({
        ...prev,
        totalSales: salesCount,
        totalLeads: leadCount,
        totalUsers: users.length,
        userCounts: {
          salesPerson: users.filter(u => u.role === "Sales Person").length,
          leadPerson: users.filter(u => u.role === "Lead Person").length,
          manager: users.filter(u => u.role === "Manager").length,
          admin: users.filter(u => u.role === "Admin").length
        }
      }));
    } catch (err) {
```

```
      console.error("Error fetching dashboard stats:", err);
    }
  };

  const fetchUsers = async () => {
    try {
      setLoading(true);
      const response = await authAPI.getUsers();
      if (response.data.success) {
        setUsers(response.data.data);

        // Update user counts in stats
        setStats(prev => ({
          ...prev,
          totalUsers: response.data.data.length,
          userCounts: {
            salesPerson: response.data.data.filter(u => u.role === "Sales
Person").length,
            leadPerson: response.data.data.filter(u => u.role === "Lead
Person").length,
            manager: response.data.data.filter(u => u.role === "Manager").length,
            admin: response.data.data.filter(u => u.role === "Admin").length
          }
        }));
      } else {
        setError('Failed to load users');
      }
    } catch (err) {
      console.error('Error fetching users:', err);
      setError('Failed to load users. Please try again.');
    } finally {
      setLoading(false);
    }
  };

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

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

    // Prevent non-admin users from creating or editing Admin accounts
    if (!isAdmin && (formData.role === 'Admin' || (editUser && editUser.role ===
'Admin'))) {
      toast.error('Only Admins can manage Admin accounts');
      return;
    }

    setLoading(true);

    try {
      let response;

      if (editUser) {
        // Update existing user
```

```
      const userData = { ...formData };
      // Only include password if it's provided
      if (!userData.password) delete userData.password;

      response = await authAPI.updateUser(editUser._id, userData);
      toast.success('User updated successfully');
    } else {
      // Create new user
      response = await authAPI.createUser(formData);
      toast.success('User created successfully');
    }

    setShowModal(false);
    resetForm();
    fetchUsers();
  } catch (err) {
    console.error('Error with user:', err);
    toast.error(err.response?.data?.message || 'Failed to process user');
  } finally {
    setLoading(false);
  }
};

const handleDeleteUser = async (userId) => {
  // Prevent non-admin users from deleting Admin accounts
  const userToDelete = users.find(u => u._id === userId);
  if (!isAdmin && userToDelete.role === 'Admin') {
    toast.error('Only Admins can delete Admin accounts');
    return;
  }

  if (!window.confirm('Are you sure you want to delete this user?')) {
    return;
  }

  try {
    setLoading(true);
    await authAPI.deleteUser(userId);
    toast.success('User deleted successfully');
    fetchUsers();
  } catch (err) {
    console.error('Error deleting user:', err);
    toast.error('Failed to delete user');
  } finally {
    setLoading(false);
  }
};

const handleEditUser = (user) => {
  // Prevent non-admin users from editing Admin accounts
  if (!isAdmin && user.role === 'Admin') {
    toast.error('Only Admins can edit Admin accounts');
    return;
  }

  setEditUser(user);
  setFormData({
    fullName: user.fullName,
    email: user.email,
```

```
      password: '', // Empty password field
      role: user.role
    });
    setShowModal(true);
  };

  const resetForm = () => {
    setFormData({
      fullName: '',
      email: '',
      password: '',
      role: 'Sales Person'
    });
    setEditUser(null);
  };

  return (
    <Layout>
      <div className="container mx-auto p-4">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-2xl font-bold">Manager Dashboard</h1>
          <button
            onClick={() => {
              resetForm();
              setShowModal(true);
            }}
            className="flex items-center px-4 py-2 bg-blue-600 hover:bg-blue-700
dark:bg-blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white rounded-md"
          >
            <FaPlus className="mr-2" /> Add New User
          </button>
        </div>

        {/* Dashboard Stats Section */}
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6
mb-6">
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-4 shadow-sm">
            <div className="flex justify-between items-start">
              <div>
                <h3 className="text-slate-500 dark:text-gray-400 text-sm font-
medium">Total Users</h3>
                <p className="text-2xl font-bold mt-1">{stats.totalUsers}</p>
              </div>
              <div className="bg-blue-100 p-3 rounded-full">
                <FaUsers className="h-5 w-5 text-blue-600" />
              </div>
            </div>
          </div>

          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-4 shadow-sm">
            <div className="flex justify-between items-start">
              <div>
                <h3 className="text-slate-500 dark:text-gray-400 text-sm font-
medium">Total Sales</h3>
```

```
              <p className="text-2xl font-bold mt-1">{stats.totalSales}</p>
            </div>
            <div className="bg-green-100 p-3 rounded-full">
              <FaMoneyBillWave className="h-5 w-5 text-green-600" />
            </div>
          </div>
          <div className="mt-2">
            <Link to="/sales-tracking" className="text-sm text-blue-600
hover:underline">View all sales</Link>
          </div>
        </div>

        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-4 shadow-sm">
          <div className="flex justify-between items-start">
            <div>
              <h3 className="text-slate-500 dark:text-gray-400 text-sm font-
medium">Total Leads</h3>
              <p className="text-2xl font-bold mt-1">{stats.totalLeads}</p>
            </div>
            <div className="bg-yellow-100 p-3 rounded-full">
              <FaFileAlt className="h-5 w-5 text-yellow-600" />
            </div>
          </div>
        </div>

        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-4 shadow-sm">
          <div className="flex justify-between items-start">
            <div>
              <h3 className="text-slate-500 dark:text-gray-400 text-sm font-
medium">Team Breakdown</h3>
              <p className="text-sm mt-1">Sales: {stats.userCounts.salesPerson}
</p>
              <p className="text-sm">Leads: {stats.userCounts.leadPerson}</p>
            </div>
            <div className="bg-purple-100 p-3 rounded-full">
              <FaChartLine className="h-5 w-5 text-purple-600" />
            </div>
          </div>
        </div>
      </div>

      {/* User Management Section */}
      <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out shadow-md
dark:shadow-2xl rounded-lg p-6 mb-6 shadow-sm">
        <h2 className="text-xl font-semibold mb-4">User Management</h2>

        {!isAdmin && (
          <div className="bg-yellow-50 border border-yellow-400 text-yellow-700
px-4 py-3 rounded relative mb-4">
            <strong>Note:</strong> As a Manager, you cannot modify Admin
accounts.
          </div>
        )}
```

```
          {loading && !showModal ? (
            <div className="flex justify-center items-center h-32">
              <div className="animate-spin rounded-full h-12 w-12 border-t-2
border-b-2 border-blue-500"></div>
            </div>
          ) : error ? (
            <div className="bg-red-100 border border-red-400 text-red-700 px-4
py-3 rounded relative mb-4">
              {error}
            </div>
          ) : (
            <div className="overflow-x-auto">
              <table className="min-w-full divide-y divide-slate-200 dark:divide-
slate-700">
                <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
                  <tr>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Name</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Email</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Role</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Actions</th>
                  </tr>
                </thead>
                <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-slate-200 dark:divide-slate-700">
                  {users.length === 0 ? (
                    <tr>
                      <td colSpan="4" className="px-6 py-4 text-center text-
slate-500 dark:text-gray-400">No users found</td>
                    </tr>
                  ) : (
                    users.map(user => (
                      <tr key={user._id} className={`hover:bg-slate-50
dark:hover:bg-slate-800 ${!isAdmin && user.role === 'Admin' ? 'bg-gray-50 dark:bg-
slate-800' : ''}`}>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">{user.fullName}</td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">{user.email}</td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-900 dark:text-slate-100">
                          <span className={`px-2 py-1 inline-flex text-xs
leading-5 font-semibold rounded-full
                            ${user.role === 'Admin' ? 'bg-purple-100 text-
purple-800' :
                              user.role === 'Manager' ? 'bg-blue-100 text-
blue-800' :
                              user.role === 'Lead Person' ? 'bg-green-100 text-
green-800' :
                              'bg-gray-100 dark:bg-slate-700 text-gray-800
dark:text-gray-200'}`}>
                            {user.role}
                          </span>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
```

```
slate-500 dark:text-gray-400">
                          <div className="flex space-x-3">
                            {(!isAdmin && user.role === 'Admin') ? (
                              <span className="text-gray-400 dark:text-gray-400"
title="Admin users can only be managed by other Admins">
                                <FaLock className="h-5 w-5" />
                              </span>
                            ) : (
                              <>
                                <button
                                  onClick={() => handleEditUser(user)}
                                  className="text-indigo-600 hover:text-
indigo-900"
                                  title="Edit user"
                                >
                                  <FaEdit className="h-5 w-5" />
                                </button>
                                <button
                                  onClick={() => handleDeleteUser(user._id)}
                                  className="text-red-600 hover:text-red-900"
                                  title="Delete user"
                                >
                                  <FaTrash className="h-5 w-5" />
                                </button>
                              </>
                            )}
                          </div>
                        </td>
                      </tr>
                    ))
                  )}
                </tbody>
              </table>
            </div>
          )}
        </div>
      </div>

      {/* User Modal */}
      {showModal && (
        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50">
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg p-6 max-w-
md w-full shadow-sm dark:shadow-black/25">
            <div className="flex justify-between items-center mb-4">
              <h2 className="text-xl font-semibold">
                {editUser ? 'Edit User' : 'Add New User'}
              </h2>
              <button
                onClick={() => setShowModal(false)}
                className="text-slate-500 dark:text-slate-300 hover:text-
slate-700"
              >
                <FaTimes className="h-5 w-5" />
              </button>
            </div>

            <form onSubmit={handleSubmit}>
```

```
              <div className="mb-4">
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                  Full Name
                </label>
                <input
                  type="text"
                  name="fullName"
                  value={formData.fullName}
                  onChange={handleInputChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required
                />
              </div>

              <div className="mb-4">
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                  Email
                </label>
                <input
                  type="email"
                  name="email"
                  value={formData.email}
                  onChange={handleInputChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required
                />
              </div>

              <div className="mb-4">
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                  {editUser ? 'Password (leave empty to keep current)' :
'Password'}
                </label>
                <input
                  type="password"
                  name="password"
                  value={formData.password}
                  onChange={handleInputChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required={!editUser}
                />
              </div>

              <div className="mb-6">
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                  Role
                </label>
                <select
                  name="role"
                  value={formData.role}
                  onChange={handleInputChange}
                  className="w-full p-2 border border-slate-300 dark:border-
```

```
slate-600 rounded-md"
                  disabled={!isAdmin && formData.role === 'Admin'}
                >
                  <option value="Sales Person">Sales Person</option>
                  <option value="Lead Person">Lead Person</option>
                  <option value="Manager">Manager</option>
                  {isAdmin && <option value="Admin">Admin</option>}
                </select>
                {!isAdmin && (
                  <p className="mt-1 text-sm text-slate-500 dark:text-
gray-400">Only Admins can create or modify Admin accounts</p>
                )}
              </div>

              <div className="flex justify-end space-x-3">
                <button
                  type="button"
                  onClick={() => setShowModal(false)}
                  className="px-4 py-2 bg-gray-300 text-gray-800 dark:text-
gray-200 rounded-md hover:bg-gray-400"
                >
                  Cancel
                </button>
                <button
                  type="submit"
                  disabled={loading || (!isAdmin && formData.role === 'Admin')}
                  className="px-4 py-2 bg-blue-600 hover:bg-blue-700 dark:bg-
blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white rounded-md disabled:bg-blue-300"
                >
                  {loading ? 'Processing...' : (editUser ? 'Update User' : 'Add
User')}
                </button>
              </div>
            </form>
          </div>
        </div>
      )}
    </Layout>
  );
};

export default ManagerDashboard;
```

src/pages/ProfilePage.jsx

```
import React, { useState, useRef, useCallback, useEffect } from "react";
import { useAuth } from "../context/AuthContext";
import { Navigate } from "react-router-dom";
import Layout from "../components/Layout/Layout";
import { authAPI } from "../services/api";
import employeeAPI from "../services/employeeAPI";
import { FaCamera, FaCheck, FaTimes, FaCrop, FaUser, FaIdCard, FaPhone,
FaEnvelope, FaMapMarkerAlt, FaBriefcase, FaCalendarAlt, FaEdit, FaClock,
FaFileAlt, FaRupeeSign, FaDollarSign, FaBuilding, FaUserTag, FaCalendarCheck,
FaChartBar, FaSignOutAlt, FaSignInAlt } from 'react-icons/fa';
import ReactCrop, { centerCrop, makeAspectCrop } from 'react-image-crop';
import 'react-image-crop/dist/ReactCrop.css';
```

```
import axios from "axios";
import LeaveManagement from "../components/Employee/LeaveManagement";
import AttendanceManagement from "../components/Employee/AttendanceManagement";
import EmployeeSelfService from "../components/Employee/EmployeeSelfService";
import AttendanceWidget from '../components/AttendanceWidget';
import SalarySlipWidget from '../components/SalarySlipWidget';

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
import { toast } from "react-toastify";

const ProfilePage = () => {
  const { user, loading, setUser } = useAuth();
  const [uploading, setUploading] = useState(false);
  const [uploadError, setUploadError] = useState(null);
  const [previewUrl, setPreviewUrl] = useState(null);
  const [image, setImage] = useState(null);
  const [selectedFile, setSelectedFile] = useState(null);
  const [crop, setCrop] = useState();
  const [completedCrop, setCompletedCrop] = useState(null);
  const [isCropping, setIsCropping] = useState(false);
  const [employeeData, setEmployeeData] = useState(null);
  const [loadingEmployee, setLoadingEmployee] = useState(false);
  const [activeTab, setActiveTab] = useState('profile');
  const fileInputRef = useRef(null);
  const imgRef = useRef(null);

  // Format date for better readability
  const formatDate = (dateString) => {
    const date = new Date(dateString);
    return date.toLocaleDateString() + " " + date.toLocaleTimeString();
  };

  const centerAspectCrop = (mediaWidth, mediaHeight, aspect) => {
    return centerCrop(
      makeAspectCrop(
        {
          unit: '%',
          width: 90,
        },
        aspect,
        mediaWidth,
        mediaHeight,
      ),
      mediaWidth,
      mediaHeight,
    );
  };

  const onImageLoad = useCallback((e) => {
    const { width, height } = e.currentTarget;
    setCrop(centerAspectCrop(width, height, 1));
  }, []);

  const handleFileChange = (e) => {
    const file = e.target.files[0];
    if (!file) return;

    if (!file.type.match('image.*')) {
```

```
      setUploadError('Please select an image file');
      return;
    }

    if (file.size > 5 * 1024 * 1024) {
      setUploadError('Image size should be less than 5MB');
      return;
    }

    setUploadError(null);
    setSelectedFile(file);
    setImage(URL.createObjectURL(file));
    setIsCropping(true);
  };

  const getCroppedImg = (image, crop) => {
    if (!crop || !image) return null;

    const canvas = document.createElement('canvas');
    const scaleX = image.naturalWidth / image.width;
    const scaleY = image.naturalHeight / image.height;
    canvas.width = crop.width;
    canvas.height = crop.height;
    const ctx = canvas.getContext('2d');

    ctx.drawImage(
      image,
      crop.x * scaleX,
      crop.y * scaleY,
      crop.width * scaleX,
      crop.height * scaleY,
      0,
      0,
      crop.width,
      crop.height
    );

    return new Promise((resolve) => {
      canvas.toBlob((blob) => {
        if (blob) {
          const file = new File([blob], 'profile-picture.jpg', { type: 'image/
jpeg' });
          resolve({
            file: file,
            url: URL.createObjectURL(blob)
          });
        } else {
          resolve(null);
        }
      }, 'image/jpeg', 0.9);
    });
  };

  const handleCropComplete = (crop) => {
    setCompletedCrop(crop);
  };

  const applyCrop = async () => {
    if (!completedCrop || !imgRef.current) return;
```

```
    const croppedData = await getCroppedImg(imgRef.current, completedCrop);
    if (croppedData) {
      setSelectedFile(croppedData.file);
      setPreviewUrl(croppedData.url);
      setIsCropping(false);
    }
  };

  const uploadProfilePicture = async () => {
    if (!previewUrl) return;

    try {
      setUploading(true);
      setUploadError(null);

      const formData = new FormData();
      formData.append('profilePicture', selectedFile);

      const response = await authAPI.updateProfilePicture(formData);

      if (response.data && response.data.success) {
        // Update user context with the new profile picture URL
        const updatedUser = {
          ...user,
          profilePicture: response.data.data.profilePicture
        };
        setUser(updatedUser);

        // Clear preview and reset state
        setPreviewUrl(null);
        setImage(null);
        setSelectedFile(null);

        // Show success message
        toast.success('Profile picture updated successfully');
      } else {
        setUploadError('Failed to update profile picture');
        toast.error('Failed to update profile picture');
      }
    } catch (error) {
      console.error('Error uploading profile picture:', error);
      const errorMessage = error.response?.data?.message || 'Failed to upload
profile picture. Please try again.';
      setUploadError(errorMessage);
      toast.error(errorMessage);
    } finally {
      setUploading(false);
    }
  };

  const cancelUpload = () => {
    // Clean up URL objects to prevent memory leaks
    if (previewUrl) {
      URL.revokeObjectURL(previewUrl);
    }
    if (image) {
      URL.revokeObjectURL(image);
    }
```

```
    setPreviewUrl(null);
    setImage(null);
    setSelectedFile(null);
    setIsCropping(false);
    setUploadError(null);
    setCrop(undefined);
    setCompletedCrop(null);
    if (fileInputRef.current) {
      fileInputRef.current.value = '';
    }
  };

  // Cleanup URLs on component unmount
  useEffect(() => {
    return () => {
      if (previewUrl) {
        URL.revokeObjectURL(previewUrl);
      }
      if (image) {
        URL.revokeObjectURL(image);
      }
    };
  }, [previewUrl, image]);

  const fetchEmployeeData = async () => {
    if (user) {
      try {
        setLoadingEmployee(true);
        const response = await employeeAPI.getAll();
        const employees = response.data?.data || response.data || [];

        // Ensure employees is an array before using .find()
        if (Array.isArray(employees)) {
          // Find employee record linked to this user
          let linkedEmployee = employees.find(emp =>
            emp.userId === user._id ||
            emp._id === user.employeeId ||
            (emp.personalInfo?.email && emp.personalInfo.email === user.email) ||
            (emp.email && emp.email === user.email)
          );

          // If no employee record found, create a basic one from user data
          if (!linkedEmployee && (user.role === 'Sales Person' || user.role ===
'Lead Person' || user.role === 'Manager' || user.role === 'Employee')) {
            try {
              const newEmployeeData = {
                fullName: user.fullName,
                email: user.email,
                userId: user._id,
                employeeId: `EMP${Date.now()}`,
                role: { name: user.role },
                department: { name: user.role === 'Sales Person' ? 'Sales' :
user.role === 'Lead Person' ? 'Leads' : 'General' },
                status: 'Active',
                joiningDate: new Date().toISOString(),
                personalInfo: {
                  firstName: user.fullName.split(' ')[0] || '',
                  lastName: user.fullName.split(' ').slice(1).join(' ') || '',
```

```
                email: user.email
              },
              professionalInfo: {
                role: user.role,
                department: user.role === 'Sales Person' ? 'Sales' : user.role
=== 'Lead Person' ? 'Leads' : 'General',
                joiningDate: new Date().toISOString().split('T')[0]
              }
            };

            // Create employee record
            const formData = new FormData();
            formData.append('employee', JSON.stringify(newEmployeeData));

            const createResponse = await employeeAPI.create(formData);
            if (createResponse.data.success) {
              linkedEmployee = createResponse.data.data;
              console.log('Employee profile created automatically for user:',
user.fullName);
            }
          } catch (createError) {
            console.error('Error creating employee profile:', createError);
          }
        }

        if (linkedEmployee) {
          setEmployeeData(linkedEmployee);
        }
      }
    } catch (error) {
      console.error('Error fetching employee data:', error);
    } finally {
      setLoadingEmployee(false);
    }
  }
};

useEffect(() => {
  if (user && !loading) {
    fetchEmployeeData();
  }
}, [user, loading]);

if (loading) {
  return (
    <div className="flex justify-center items-center min-h-screen">
      <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2
border-blue-500"></div>
    </div>
  );
}

if (!user) {
  return <Navigate to="/login" />;
}

const tabs = [
  { id: 'profile', label: 'Profile', icon: FaUser },
  { id: 'self-service', label: 'Self Service', icon: FaEdit },
```

```
    { id: 'attendance', label: 'Attendance', icon: FaClock },
    { id: 'salary', label: 'Salary', icon: FaRupeeSign },
    { id: 'leave', label: 'Leave Management', icon: FaCalendarAlt }
  ];

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="max-w-2xl mx-auto bg-white dark:bg-slate-900 border
border-slate-200 dark:border-slate-700 transition-all duration-200 ease-out
rounded-lg shadow-md dark:shadow-2xl overflow-hidden shadow-sm">
          <div className="bg-blue-600 hover:bg-blue-700 dark:bg-blue-500
dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md transition-all
duration-200 text-white p-4">
            <h2 className="text-2xl font-bold">User Profile</h2>
          </div>
          <div className="p-6">
            <div className="mb-8 text-center">
              {/* Cropping UI */}
              {isCropping && image && (
                <div className="mb-6">
                  <h4 className="text-lg font-medium mb-2">Adjust Your Profile
Picture</h4>
                  <div className="max-w-md mx-auto">
                    <ReactCrop
                      crop={crop}
                      onChange={(_, percentCrop) => setCrop(percentCrop)}
                      onComplete={handleCropComplete}
                      aspect={1}
                      circularCrop
                    >
                      <img
                        ref={imgRef}
                        src={image}
                        alt="Crop Preview"
                        onLoad={onImageLoad}
                        className="max-w-full max-h-96"
                      />
                    </ReactCrop>
                    <div className="flex justify-center mt-4 space-x-3">
                      <button
                        type="button"
                        onClick={applyCrop}
                        className="px-4 py-2 bg-green-500 text-white rounded
hover:bg-green-600 flex items-center"
                      >
                        <FaCrop className="mr-2" /> Apply Crop
                      </button>
                      <button
                        type="button"
                        onClick={cancelUpload}
                        className="px-4 py-2 bg-gray-500 text-white rounded
hover:bg-gray-600"
                      >
                        Cancel
                      </button>
                    </div>
                  </div>
                </div>
```

```
              )}

              {/* Profile Picture Preview */}
              {!isCropping && (
                <div className="relative h-24 w-24 rounded-full mx-auto mb-4">
                  {previewUrl ? (
                    <img
                      src={previewUrl}
                      alt="Profile Preview"
                      className="h-24 w-24 rounded-full object-cover border-2
border-blue-300"
                    />
                  ) : employeeData?.documents?.photo ? (
                    <img
                      src={`/api/employees/file/${employeeData.documents.photo}`}
                      alt={user.fullName}
                      className="h-24 w-24 rounded-full object-cover border-2
border-blue-200"
                    />
                  ) : user.profilePicture ? (
                    <img
                      src={user.profilePicture.startsWith('/uploads') ?
                        `${import.meta.env.DEV ? 'http://localhost:8080' :
''}${user.profilePicture}` :
                        user.profilePicture}
                      alt={user.fullName}
                      className="h-24 w-24 rounded-full object-cover"
                      onError={(e) => {
                        e.target.style.display = 'none';
                        e.target.nextSibling.style.display = 'flex';
                      }}
                    />
                  ) : (
                    <div className="h-24 w-24 rounded-full bg-blue-100 flex items-
center justify-center">
                      <span className="text-3xl font-bold text-blue-600">
                        {employeeData?.personalInfo?.firstName ?

employeeData.personalInfo.firstName.charAt(0).toUpperCase() :
                          user.fullName ? user.fullName.charAt(0).toUpperCase() :
"U"}
                      </span>
                    </div>
                  )}

                  {/* Fallback div for failed image loads */}
                  {user.profilePicture && (
                    <div className="h-24 w-24 rounded-full bg-blue-100 flex items-
center justify-center" style={{display: 'none'}}>
                      <span className="text-3xl font-bold text-blue-600">
                        {user.fullName ? user.fullName.charAt(0).toUpperCase() :
"U"}
                      </span>
                    </div>
                  )}

                  <label
                    htmlFor="profile-upload"
                    className="absolute bottom-0 right-0 bg-blue-600 hover:bg-
```

```
blue-700 text-white rounded-full p-2 cursor-pointer"
                  >
                    <FaCamera size={12} />
                  </label>
                  <input
                    type="file"
                    id="profile-upload"
                    className="hidden"
                    accept="image/*"
                    onChange={handleFileChange}
                    ref={fileInputRef}
                  />
                </div>
              )}

              {/* Preview Controls */}
              {previewUrl && !isCropping && (
                <div className="flex justify-center space-x-3 mb-4">
                  <button
                    onClick={uploadProfilePicture}
                    disabled={uploading}
                    className={`p-2 rounded-full ${uploading ? 'bg-gray-300' :
'bg-green-500 hover:bg-green-600'} text-white`}
                  >
                    <FaCheck size={12} />
                  </button>
                  <button
                    onClick={cancelUpload}
                    disabled={uploading}
                    className="p-2 rounded-full bg-red-500 hover:bg-red-600 text-
white"
                  >
                    <FaTimes size={12} />
                  </button>
                </div>
              )}

              {uploadError && (
                <div className="text-red-500 text-sm mb-2">{uploadError}</div>
              )}

              <h3 className="text-xl font-bold mt-2">
                {employeeData?.personalInfo?.firstName &&
employeeData?.personalInfo?.lastName ?
                  `${employeeData.personalInfo.firstName}
${employeeData.personalInfo.lastName}` :
                  user.fullName}
              </h3>
              <p className="text-gray-600 dark:text-gray-500">
                {employeeData?.personalInfo?.email || user.email}
              </p>
              <div className="flex flex-wrap gap-2 mt-2 justify-center">
                <span className="inline-block px-3 py-1 bg-blue-100 text-blue-800
rounded-full text-sm font-semibold">
                  {user.role}
                </span>
                {employeeData?.employeeId && (
                  <span className="inline-block px-3 py-1 bg-green-100 text-
green-800 rounded-full text-sm font-semibold">
```

```
                    ID: {employeeData.employeeId}
                  </span>
                )}
                {employeeData?.department?.name && (
                  <span className="inline-block px-3 py-1 bg-purple-100 text-
purple-800 rounded-full text-sm font-semibold">
                    {employeeData.department.name}
                  </span>
                )}
              </div>
            </div>

            {/* Tab Navigation */}
            <div className="mb-6">
              <div className="border-b border-gray-200 dark:border-gray-700">
                <nav className="-mb-px flex space-x-8">
                  {tabs.map((tab) => (
                    <button
                      key={tab.id}
                      onClick={() => setActiveTab(tab.id)}
                      className={`${
                        activeTab === tab.id
                          ? 'border-blue-500 text-blue-600 dark:text-blue-400'
                          : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300 dark:text-gray-400 dark:hover:text-gray-300'
                      } whitespace-nowrap py-2 px-1 border-b-2 font-medium text-
sm flex items-center`}
                    >
                      <tab.icon className="mr-2 h-4 w-4" />
                      {tab.label}
                    </button>
                  ))}
                </nav>
              </div>
            </div>

            {/* Tab Content */}
            {activeTab === 'profile' && (
              <div className="space-y-4">
                <div className="border-t border-slate-200 dark:border-slate-700
pt-4">
                  <h4 className="text-lg font-bold mb-2">Account Details</h4>
                  <div className="grid grid-cols-2 gap-4">
                    <div>
                      <p className="text-sm text-gray-600 dark:text-
gray-500">User ID</p>
                      <p className="font-medium">{user._id}</p>
                    </div>
                    <div>
                      <p className="text-sm text-gray-600 dark:text-
gray-500">Created</p>
                      <p className="font-medium">{formatDate(user.createdAt)}</p>
                    </div>
                  </div>
                </div>

                {user.role === "Sales Person" && (
                  <div className="border-t border-slate-200 dark:border-slate-700
pt-4">
```

```
                    <h4 className="text-lg font-bold mb-2">Sales Information</h4>
                    <p className="text-gray-600 dark:text-gray-500">You have
access to view and manage leads assigned to you.</p>
                    <div className="mt-2 p-2 bg-blue-50 dark:bg-blue-900/20
rounded-lg">
                      <p className="text-sm text-blue-800 dark:text-blue-200">
                        Ø=Ü¡ <strong>Tip:</strong> Use the <strong>"My Leave"</
strong> and <strong>"My Attendance"</strong> tabs above to manage your leave
applications and attendance.
                      </p>
                    </div>
                  </div>
                )}

                {user.role === "Lead Person" && (
                  <div className="border-t border-slate-200 dark:border-slate-700
pt-4">
                    <h4 className="text-lg font-bold mb-2">Lead Management</h4>
                    <p className="text-gray-600 dark:text-gray-500">You have
access to create leads and view leads assigned to you.</p>
                    <div className="mt-2 p-2 bg-blue-50 dark:bg-blue-900/20
rounded-lg">
                      <p className="text-sm text-blue-800 dark:text-blue-200">
                        Ø=Ü¡ <strong>Tip:</strong> Use the <strong>"My Leave"</
strong> and <strong>"My Attendance"</strong> tabs above to manage your leave
applications and attendance.
                      </p>
                    </div>
                  </div>
                )}

                {(user.role === "Admin" || user.role === "Manager") && (
                  <div className="border-t border-slate-200 dark:border-slate-700
pt-4">
                    <h4 className="text-lg font-bold mb-2">Administrative Access</
h4>
                    <p className="text-gray-600 dark:text-gray-500">You have
access to all leads and administrative functions.</p>
                    <div className="mt-2 p-2 bg-green-50 dark:bg-green-900/20
rounded-lg">
                      <p className="text-sm text-green-800 dark:text-green-200">
                        Ø=Ý' <strong>Admin/Manager:</strong> Approve leave
applications in <strong>Employee Management !’ Leave Approvals</strong>.
                        Use <strong>"My Leave"</strong> tab above for your own
leave applications.
                      </p>
                    </div>
                  </div>
                )}

                {user.role === "HR" && (
                  <div className="border-t border-slate-200 dark:border-slate-700
pt-4">
                    <h4 className="text-lg font-bold mb-2">HR Access</h4>
                    <p className="text-gray-600 dark:text-gray-500">You have
access to employee management and HR functions.</p>
                    <div className="mt-2 p-2 bg-purple-50 dark:bg-purple-900/20
rounded-lg">
                      <p className="text-sm text-purple-800 dark:text-purple-200">
```

```
                        Ø=Üe <strong>HR:</strong> Manage employee records and
documents. Use <strong>"My Leave"</strong> tab above for your own leave
applications.
                      </p>
                    </div>
                  </div>
                )}

                {user.role === "Employee" && (
                  <div className="border-t border-slate-200 dark:border-slate-700
pt-4">
                    <h4 className="text-lg font-bold mb-2">Employee Access</h4>
                    <p className="text-gray-600 dark:text-gray-500">You have
access to general employee functions and self-service features.</p>
                    <div className="mt-2 p-2 bg-blue-50 dark:bg-blue-900/20
rounded-lg">
                      <p className="text-sm text-blue-800 dark:text-blue-200">
                        Ø=Ü¡ <strong>Tip:</strong> Use the <strong>"My Leave"</
strong> and <strong>"My Attendance"</strong> tabs above to manage your leave
applications and attendance.
                      </p>
                    </div>
                  </div>
                )}

                {/* Employee Information Section */}
                <div className="bg-gray-50 dark:bg-slate-800 rounded-lg p-4">
                  <h4 className="text-lg font-bold mb-4 text-gray-900 dark:text-
white flex items-center">
                    <FaBriefcase className="mr-2 text-blue-600" />
                    Employee Information
                  </h4>

                  {loadingEmployee ? (
                    <div className="flex items-center justify-center py-8">
                      <div className="animate-spin rounded-full h-8 w-8 border-
t-2 border-b-2 border-blue-500 mr-3"></div>
                      <span className="text-gray-600 dark:text-gray-400">Setting
up your employee profile...</span>
                    </div>
                  ) : employeeData ? (
                    <div className="space-y-4">
                      <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                        <div>
                          <span className="font-medium text-gray-700 dark:text-
gray-300">Employee ID:</span>
                          <p className="text-gray-600 dark:text-
gray-400">{employeeData.employeeId || 'Auto-generated'}</p>
                        </div>
                        <div>
                          <span className="font-medium text-gray-700 dark:text-
gray-300">Department:</span>
                          <p className="text-gray-600 dark:text-
gray-400">{employeeData.department?.name ||
employeeData.professionalInfo?.department || 'Not assigned'}</p>
                        </div>
                        <div>
                          <span className="font-medium text-gray-700 dark:text-
gray-300">Position:</span>
```

```
                          <p className="text-gray-600 dark:text-
gray-400">{employeeData.role?.name || employeeData.professionalInfo?.role ||
user.role}</p>
                        </div>
                        <div>
                          <span className="font-medium text-gray-700 dark:text-
gray-300">Salary:</span>
                          <p className="text-gray-600 dark:text-gray-400">
                            {employeeData.salary ? ` ¹
${employeeData.salary.toLocaleString()}` : 'Not disclosed'}
                          </p>
                        </div>
                        <div>
                          <span className="font-medium text-gray-700 dark:text-
gray-300">Status:</span>
                          <p className="text-gray-600 dark:text-
gray-400">{employeeData.status || 'Active'}</p>
                        </div>
                        <div>
                          <span className="font-medium text-gray-700 dark:text-
gray-300">Joining Date:</span>
                          <p className="text-gray-600 dark:text-gray-400">
                            {employeeData.joiningDate ? new
Date(employeeData.joiningDate).toLocaleDateString() :
                             employeeData.professionalInfo?.joiningDate ? new
Date(employeeData.professionalInfo.joiningDate).toLocaleDateString() :
                             'Recently joined'}
                          </p>
                        </div>
                        <div>
                          <span className="font-medium text-gray-700 dark:text-
gray-300">Phone:</span>
                          <p className="text-gray-600 dark:text-gray-400">
                            {employeeData.phoneNumber ||
employeeData.personalInfo?.phoneNumber || 'Not provided'}
                          </p>
                        </div>
                      </div>

                      {/* Document Status */}
                      {employeeData.documents && (
                        <div className="mt-4 p-3 bg-blue-50 dark:bg-blue-900/20
rounded-lg">
                          <h5 className="font-medium text-blue-900 dark:text-
blue-100 mb-2">Document Status</h5>
                          <div className="grid grid-cols-2 md:grid-cols-3 gap-2
text-sm">
                            {Object.entries(employeeData.documents).map(([key,
value]) => (
                              <div key={key} className="flex items-center">
                                <span className={`w-2 h-2 rounded-full mr-2
${value ? 'bg-green-500' : 'bg-red-500'}`}></span>
                                <span className="text-gray-700 dark:text-
gray-300">{key.replace(/([A-Z])/g, ' $1').replace(/^./, str => str.toUpperCase())}
</span>
                              </div>
                            ))}
                          </div>
                        </div>
```

```
                    )}

                    <div className="mt-4 p-3 bg-green-50 dark:bg-green-900/20
rounded-lg">
                      <p className="text-sm text-green-800 dark:text-green-200">
                        '  Your employee profile is active. Additional details
and documents can be updated by managers or administrators.
                      </p>
                    </div>
                  </div>
                ) : (
                  <div className="text-center py-8">
                    <div className="bg-yellow-50 dark:bg-yellow-900/20 border
border-yellow-200 dark:border-yellow-800 rounded-lg p-4">
                      <div className="flex items-center justify-center mb-2">
                        <FaBriefcase className="text-yellow-600 dark:text-
yellow-400 text-2xl mr-2" />
                        <h5 className="text-lg font-medium text-yellow-800
dark:text-yellow-200">Setting Up Profile</h5>
                      </div>
                      <p className="text-yellow-700 dark:text-yellow-300 mb-3">
                        Your employee profile is being set up automatically
based on your user account.
                      </p>
                      <button
                        onClick={fetchEmployeeData}
                        className="px-4 py-2 bg-yellow-600 text-white rounded-
lg hover:bg-yellow-700 transition-colors"
                      >
                        Refresh Profile
                      </button>
                    </div>
                  </div>
                )}
              </div>
            </div>
          )}

          {/* Attendance Tab */}
          {activeTab === 'attendance' && (
            <div className="space-y-6">
              <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
                <AttendanceWidget />
                <div className="bg-white dark:bg-slate-800 rounded-lg shadow-md
p-6">
                  <h3 className="text-lg font-semibold text-gray-900 dark:text-
white mb-4">
                    Quick Stats
                  </h3>
                  <div className="space-y-3">
                    <div className="flex items-center justify-between">
                      <span className="text-gray-600 dark:text-gray-400">This
Month</span>
                      <span className="font-semibold text-gray-900 dark:text-
white">
                        {/* This will be filled by AttendanceManagement
component */}
                      </span>
                    </div>
```

```
                  </div>
                </div>
              </div>
              {employeeData && (
                <AttendanceManagement employeeId={employeeData._id}
userRole={user?.role} />
              )}
            </div>
          )}

          {/* Salary Tab */}
          {activeTab === 'salary' && (
            <div className="space-y-6">
              <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
                <SalarySlipWidget />
                <div className="bg-white dark:bg-slate-800 rounded-lg shadow-md
p-6">
                  <h3 className="text-lg font-semibold text-gray-900 dark:text-
white mb-4">
                    Salary Information
                  </h3>
                  {employeeData && (
                    <div className="space-y-3">
                      <div className="flex items-center justify-between">
                        <span className="text-gray-600 dark:text-
gray-400">Monthly Salary</span>
                        <span className="font-semibold text-gray-900 dark:text-
white">
                          {employeeData.salary ? `₹
${employeeData.salary.toLocaleString()}` : 'Not disclosed'}
                        </span>
                      </div>
                      <div className="flex items-center justify-between">
                        <span className="text-gray-600 dark:text-
gray-400">Department</span>
                        <span className="font-semibold text-gray-900 dark:text-
white">
                          {employeeData.department?.name || 'Not assigned'}
                        </span>
                      </div>
                      <div className="flex items-center justify-between">
                        <span className="text-gray-600 dark:text-
gray-400">Role</span>
                        <span className="font-semibold text-gray-900 dark:text-
white">
                          {employeeData.role?.name || user.role}
                        </span>
                      </div>
                      <div className="flex items-center justify-between">
                        <span className="text-gray-600 dark:text-
gray-400">Employee Status</span>
                        <span className={`px-2 py-1 rounded-full text-xs font-
medium ${
                          employeeData.status === 'ACTIVE'
                            ? 'bg-green-100 text-green-800 dark:bg-green-900
dark:text-green-200'
                            : 'bg-gray-100 text-gray-800 dark:bg-gray-700
dark:text-gray-300'
                        }`}>
```

```
                          {employeeData.status || 'Active'}
                        </span>
                      </div>
                    </div>
                  )}
                </div>
              </div>
              <div className="bg-white dark:bg-slate-800 rounded-lg shadow-md
p-6">
                <h3 className="text-lg font-semibold text-gray-900 dark:text-
white mb-4">
                  Payroll Information
                </h3>
                <div className="text-gray-600 dark:text-gray-400 text-sm space-
y-2">
                  <p>• Your salary is calculated based on your attendance and
working days</p>
                  <p>• Monthly salary is divided by 30 working days to
calculate daily rate</p>
                  <p>• Incentives and bonuses are added to your base salary</p>
                  <p>• Deductions include PF, ESI, and professional tax as per
government regulations</p>
                  <p>• Download your salary slip once it's approved by HR/
Admin</p>
                </div>
              </div>
            </div>
          )}

          {/* Self Service Tab */}
          {activeTab === 'self-service' && employeeData && (
            <EmployeeSelfService employeeData={employeeData}
onDataUpdated={fetchEmployeeData} />
          )}

          {/* Leave Management Tab */}
          {activeTab === 'leave' && employeeData && (
            <LeaveManagement employeeId={employeeData._id}
userRole={user?.role} />
          )}
        </div>
      </div>
    </div>
  </Layout>
  );
};

export default ProfilePage;
```

src/pages/ProspectsPage.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../context/AuthContext';
import { prospectsAPI, authAPI } from '../services/api';
import { toast } from 'react-hot-toast';
import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
import {
```

```
  FiPlus,
  FiSearch,
  FiEdit,
  FiTrash2,
  FiArrowRight,
  FiPhone,
  FiMail,
  FiUser,
  FiEye,
  FiX
} from 'react-icons/fi';

// Status badge component
const StatusBadge = ({ status }) => {
  const statusColors = {
    'New': 'bg-blue-100 text-blue-800',
    'Contacted': 'bg-yellow-100 text-yellow-800',
    'Interested': 'bg-green-100 text-green-800',
    'Not Interested': 'bg-red-100 text-red-800',
    'Follow Up': 'bg-purple-100 text-purple-800',
    'Qualified': 'bg-indigo-100 text-indigo-800',
    'Converted to Lead': 'bg-emerald-100 text-emerald-800',
    'Lost': 'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-gray-200'
  };

  return (
    <span className={`px-2 py-1 rounded-full text-xs font-medium
${statusColors[status] || 'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-
gray-200'}`}>
      {status}
    </span>
  );
};

// Priority badge component
const PriorityBadge = ({ priority }) => {
  const priorityColors = {
    'High': 'bg-red-100 text-red-800',
    'Medium': 'bg-yellow-100 text-yellow-800',
    'Low': 'bg-green-100 text-green-800'
  };

  return (
    <span className={`px-2 py-1 rounded-full text-xs font-medium
${priorityColors[priority] || 'bg-gray-100 dark:bg-slate-700 text-gray-800
dark:text-gray-200'}`}>
      {priority}
    </span>
  );
};

const ProspectsPage = () => {
  const { user, token } = useAuth();
  const [prospects, setProspects] = useState([]);
  const [loading, setLoading] = useState(true);
  const [showCreateModal, setShowCreateModal] = useState(false);
  const [showEditModal, setShowEditModal] = useState(false);
  const [showViewModal, setShowViewModal] = useState(false);
  const [selectedProspect, setSelectedProspect] = useState(null);
```

```
const [stats, setStats] = useState({});

// Filters and search
const [filters, setFilters] = useState({
  search: '',
  status: '',
  source: '',
  priority: '',
  page: 1,
  limit: 100
});

const [pagination, setPagination] = useState({});

// Fetch prospects
const fetchProspects = async () => {
  try {
    setLoading(true);
    const response = await prospectsAPI.getAll(filters);

    if (response.data.success) {
      setProspects(response.data.data);
      setPagination(response.data.pagination);
    }
  } catch (error) {
    console.error('Error fetching prospects:', error);
    toast.error('Failed to fetch prospects');
  } finally {
    setLoading(false);
  }
};

// Fetch stats
const fetchStats = async () => {
  try {
    const response = await prospectsAPI.getStats();

    if (response.data.success) {
      setStats(response.data.data);
    }
  } catch (error) {
    console.error('Error fetching stats:', error);
  }
};

useEffect(() => {
  fetchProspects();
  fetchStats();
}, [filters]);

// Handle search
const handleSearch = (e) => {
  setFilters(prev => ({ ...prev, search: e.target.value, page: 1 }));
};

// Handle filter change
const handleFilterChange = (key, value) => {
  setFilters(prev => ({ ...prev, [key]: value, page: 1 }));
};
```

```
  // Handle delete
  const handleDelete = async (id) => {
    if (!window.confirm('Are you sure you want to delete this prospect?')) return;

    try {
      const response = await prospectsAPI.delete(id);

      if (response.data.success) {
        toast.success('Prospect deleted successfully');
        fetchProspects();
        fetchStats();
      }
    } catch (error) {
      console.error('Error deleting prospect:', error);
      toast.error(error.response?.data?.message || 'Failed to delete prospect');
    }
  };

  // Handle convert to lead
  const handleConvertToLead = async (id) => {
    if (!window.confirm('Convert this prospect to a lead?')) return;

    try {
      const response = await prospectsAPI.convertToLead(id);

      if (response.data.success) {
        toast.success('Prospect converted to lead successfully');
        fetchProspects();
        fetchStats();
      }
    } catch (error) {
      console.error('Error converting prospect:', error);
      toast.error(error.response?.data?.message || 'Failed to convert prospect');
    }
  };

  return (
    <div className="p-6 max-w-7xl mx-auto">
      {/* Header */}
      <div className="mb-6">
        <div className="flex justify-between items-center mb-4">
          <div>
            <h1 className="text-2xl font-bold text-slate-900 dark:text-
slate-100">Prospects</h1>
            <p className="text-gray-600 dark:text-gray-500">Manage your potential
customers</p>
          </div>
          <button
            onClick={() => setShowCreateModal(true)}
            className="bg-blue-600 hover:bg-blue-700 dark:bg-blue-500
dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md transition-all
duration-200 text-white px-4 py-2 rounded-lg flex items-center gap-2"
          >
            <FiPlus /> Add Prospect
          </button>
        </div>

        {/* Stats Cards */}
```

```
        <div className="grid grid-cols-2 md:grid-cols-4 lg:grid-cols-7 gap-4
mb-6">
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 rounded-lg shadow
shadow-sm dark:shadow-black/25">
            <div className="text-2xl font-bold text-
blue-600">{stats.overview?.total || 0}</div>
            <div className="text-sm text-gray-600 dark:text-gray-500">Total</div>
          </div>
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 rounded-lg shadow
shadow-sm dark:shadow-black/25">
            <div className="text-2xl font-bold text-
blue-500">{stats.overview?.new || 0}</div>
            <div className="text-sm text-gray-600 dark:text-gray-500">New</div>
          </div>
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 rounded-lg shadow
shadow-sm dark:shadow-black/25">
            <div className="text-2xl font-bold text-
yellow-500">{stats.overview?.contacted || 0}</div>
            <div className="text-sm text-gray-600 dark:text-gray-500">Contacted</
div>
          </div>
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 rounded-lg shadow
shadow-sm dark:shadow-black/25">
            <div className="text-2xl font-bold text-
green-500">{stats.overview?.interested || 0}</div>
            <div className="text-sm text-gray-600 dark:text-gray-500">Interested</
div>
          </div>
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 rounded-lg shadow
shadow-sm dark:shadow-black/25">
            <div className="text-2xl font-bold text-
indigo-500">{stats.overview?.qualified || 0}</div>
            <div className="text-sm text-gray-600 dark:text-gray-500">Qualified</
div>
          </div>
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 rounded-lg shadow
shadow-sm dark:shadow-black/25">
            <div className="text-2xl font-bold text-
emerald-500">{stats.overview?.converted || 0}</div>
            <div className="text-sm text-gray-600 dark:text-gray-500">Converted</
div>
          </div>
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-4 rounded-lg shadow
shadow-sm dark:shadow-black/25">
            <div className="text-2xl font-bold text-
red-500">{stats.overview?.lost || 0}</div>
            <div className="text-sm text-gray-600 dark:text-gray-500">Lost</div>
          </div>
        </div>

        {/* Filters */}
        <div className="bg-white dark:bg-slate-900 border border-slate-200
```

```
dark:border-slate-700 transition-all duration-200 ease-out p-4 rounded-lg shadow
mb-6 shadow-sm dark:shadow-black/25">
          <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
            {/* Search */}
            <div className="relative">
              <FiSearch className="absolute left-3 top-3 text-gray-400 dark:text-
gray-400" />
              <input
                type="text"
                placeholder="Search prospects..."
                value={filters.search}
                onChange={handleSearch}
                className="pl-10 pr-4 py-2 border rounded-lg w-full focus:ring-2
focus:ring-blue-500 dark:focus:ring-offset-slate-900 focus:ring-offset-2
focus:border-transparent"
              />
            </div>

            {/* Status Filter */}
            <select
              value={filters.status}
              onChange={(e) => handleFilterChange('status', e.target.value)}
              className="px-4 py-2 border rounded-lg focus:ring-2 focus:ring-
blue-500 dark:focus:ring-offset-slate-900 focus:ring-offset-2 focus:border-
transparent"
            >
              <option value="">All Status</option>
              <option value="New">New</option>
              <option value="Contacted">Contacted</option>
              <option value="Interested">Interested</option>
              <option value="Not Interested">Not Interested</option>
              <option value="Follow Up">Follow Up</option>
              <option value="Qualified">Qualified</option>
              <option value="Converted to Lead">Converted to Lead</option>
              <option value="Lost">Lost</option>
            </select>

            {/* Source Filter */}
            <select
              value={filters.source}
              onChange={(e) => handleFilterChange('source', e.target.value)}
              className="px-4 py-2 border rounded-lg focus:ring-2 focus:ring-
blue-500 dark:focus:ring-offset-slate-900 focus:ring-offset-2 focus:border-
transparent"
            >
              <option value="">All Sources</option>
              <option value="LinkedIn">LinkedIn</option>
              <option value="Website">Website</option>
              <option value="Referral">Referral</option>
              <option value="Cold Call">Cold Call</option>
              <option value="Email Campaign">Email Campaign</option>
              <option value="Social Media">Social Media</option>
              <option value="Event">Event</option>
              <option value="Other">Other</option>
            </select>

            {/* Priority Filter */}
            <select
              value={filters.priority}
```

```
              onChange={(e) => handleFilterChange('priority', e.target.value)}
              className="px-4 py-2 border rounded-lg focus:ring-2 focus:ring-
blue-500 dark:focus:ring-offset-slate-900 focus:ring-offset-2 focus:border-
transparent"
            >
              <option value="">All Priorities</option>
              <option value="High">High</option>
              <option value="Medium">Medium</option>
              <option value="Low">Low</option>
            </select>
          </div>
        </div>
      </div>

      {/* Prospects Table */}
      <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow
overflow-hidden shadow-sm dark:shadow-black/25">
        {loading ? (
          <div className="p-8 text-center">
            <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-
blue-600 mx-auto"></div>
            <p className="mt-2 text-gray-600 dark:text-gray-500">Loading
prospects...</p>
          </div>
        ) : prospects.length === 0 ? (
          <div className="p-8 text-center">
            <FiUser className="mx-auto h-12 w-12 text-gray-400 dark:text-
gray-400" />
            <h3 className="mt-2 text-sm font-medium text-slate-900 dark:text-
slate-100">No prospects</h3>
            <p className="mt-1 text-sm text-slate-500 dark:text-gray-400">Get
started by creating a new prospect.</p>
            <div className="mt-6">
              <button
                onClick={() => setShowCreateModal(true)}
                className="bg-blue-600 hover:bg-blue-700 dark:bg-blue-500
dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md transition-all
duration-200 text-white px-4 py-2 rounded-lg flex items-center gap-2 mx-auto"
              >
                <FiPlus /> Add Prospect
              </button>
            </div>
          </div>
        ) : (
          <>
            {/* Desktop Table */}
            <div className="hidden lg:block overflow-x-auto">
              <table className="min-w-full divide-y divide-slate-200 dark:divide-
slate-700">
                <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
                  <tr>
                    <th scope="col" className="px-4 py-3 text-left text-xs font-
medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                      #
                    </th>
                    <th className="px-4 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
```

```
                    Contact Info
                  </th>
                  <th className="px-4 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                    Company
                  </th>
                  <th className="px-4 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                    Source
                  </th>
                  <th className="px-4 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                    Status
                  </th>
                  <th className="px-4 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                    Priority
                  </th>
                  <th className="px-4 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                    Assigned To
                  </th>
                  <th className="px-4 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                    Created
                  </th>
                  <th className="px-4 py-3 text-center text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">
                    Actions
                  </th>
                </tr>
              </thead>
              <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-slate-200 dark:divide-slate-700">
                {prospects.map((prospect, index) => (
                  <tr key={prospect._id} className={index % 2 === 0 ? 'bg-white
dark:bg-slate-800' : 'bg-slate-50 dark:bg-slate-800'}>
                    <td className="px-4 py-3 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                      {index + 1}
                    </td>
                    <td className="px-4 py-3">
                      <div className="max-w-xs">
                        <div className="text-sm font-medium text-slate-900
dark:text-slate-100 truncate">
                          {prospect.name || 'No Name'}
                        </div>
                        <div className="text-xs text-slate-500 dark:text-
gray-400">
                          {prospect.email && (
                            <div className="flex items-center gap-1 truncate">
                              <FiMail className="w-3 h-3 flex-shrink-0" />
                              <span className="truncate">{prospect.email}</span>
                            </div>
                          )}
                          {prospect.phone && (
                            <div className="flex items-center gap-1 truncate">
                              <FiPhone className="w-3 h-3 flex-shrink-0" />
                              <span className="truncate">{prospect.phone}</span>
```

```
                            </div>
                          )}
                        </div>
                      </div>
                    </td>
                    <td className="px-4 py-3">
                      <div className="max-w-xs">
                        <div className="text-sm text-slate-900 dark:text-
slate-100 truncate">{prospect.company || '-'}</div>
                        <div className="text-xs text-slate-500 dark:text-
gray-400 truncate">{prospect.designation || '-'}</div>
                      </div>
                    </td>
                    <td className="px-4 py-3">
                      <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.source}</div>
                    </td>
                    <td className="px-4 py-3">
                      <StatusBadge status={prospect.status} />
                    </td>
                    <td className="px-4 py-3">
                      <PriorityBadge priority={prospect.priority} />
                    </td>
                    <td className="px-4 py-3">
                      <div className="text-sm text-slate-900 dark:text-
slate-100 max-w-xs truncate">
                        {prospect.assignedTo?.fullName || 'Unassigned'}
                      </div>
                    </td>
                    <td className="px-4 py-3 text-sm text-slate-500 dark:text-
gray-400">
                      {new Date(prospect.createdAt).toLocaleDateString()}
                    </td>
                    <td className="px-4 py-3">
                      <div className="flex items-center justify-center gap-2">
                        <button
                          onClick={() => {
                            setSelectedProspect(prospect);
                            setShowViewModal(true);
                          }}
                          className="text-blue-600 hover:text-blue-900 p-1
rounded hover:bg-blue-50"
                          title="View Details"
                        >
                          <FiEye className="w-4 h-4" />
                        </button>

                        <button
                          onClick={() => {
                            setSelectedProspect(prospect);
                            setShowEditModal(true);
                          }}
                          className="text-green-600 hover:text-green-900 p-1
rounded hover:bg-green-50"
                          title="Edit"
                        >
                          <FiEdit className="w-4 h-4" />
                        </button>
```

```
                        {prospect.status !== 'Converted to Lead' && (
                          <button
                            onClick={() => handleConvertToLead(prospect._id)}
                            className="text-purple-600 hover:text-purple-900
p-1 rounded hover:bg-purple-50"
                            title="Convert to Lead"
                          >
                            <FiArrowRight className="w-4 h-4" />
                          </button>
                        )}

                        {['Admin', 'Manager'].includes(user?.role) && (
                          <button
                            onClick={() => handleDelete(prospect._id)}
                            className="text-red-600 hover:text-red-900 p-1
rounded hover:bg-red-50"
                            title="Delete"
                          >
                            <FiTrash2 className="w-4 h-4" />
                          </button>
                        )}
                      </div>
                    </td>
                  </tr>
                ))}
              </tbody>
            </table>
          </div>

          {/* Mobile Card View */}
          <div className="lg:hidden">
            {prospects.map((prospect) => (
              <div key={prospect._id} className="border-b border-slate-200
dark:border-slate-700 p-4">
                <div className="flex justify-between items-start mb-3">
                  <div className="flex-1">
                    <h3 className="text-sm font-medium text-slate-900 dark:text-
slate-100">
                      {prospect.name || 'No Name'}
                    </h3>
                    <p className="text-xs text-slate-500 dark:text-
gray-400">{prospect.company || 'No Company'}</p>
                  </div>
                  <div className="flex gap-2 ml-4">
                    <button
                      onClick={() => {
                        setSelectedProspect(prospect);
                        setShowViewModal(true);
                      }}
                      className="text-blue-600 hover:text-blue-900 p-2 rounded
hover:bg-blue-50"
                      title="View Details"
                    >
                      <FiEye className="w-4 h-4" />
                    </button>

                    <button
                      onClick={() => {
                        setSelectedProspect(prospect);
```

```
                        setShowEditModal(true);
                      }}
                      className="text-green-600 hover:text-green-900 p-2
rounded hover:bg-green-50"
                      title="Edit"
                    >
                      <FiEdit className="w-4 h-4" />
                    </button>
                  </div>
                </div>

                <div className="grid grid-cols-2 gap-2 text-xs">
                  <div>
                    <span className="text-slate-500 dark:text-
gray-400">Status:</span>
                    <div className="mt-1">
                      <StatusBadge status={prospect.status} />
                    </div>
                  </div>
                  <div>
                    <span className="text-slate-500 dark:text-
gray-400">Priority:</span>
                    <div className="mt-1">
                      <PriorityBadge priority={prospect.priority} />
                    </div>
                  </div>
                  <div>
                    <span className="text-slate-500 dark:text-
gray-400">Source:</span>
                    <div className="mt-1 text-slate-900 dark:text-
slate-100">{prospect.source}</div>
                  </div>
                  <div>
                    <span className="text-slate-500 dark:text-
gray-400">Assigned:</span>
                    <div className="mt-1 text-slate-900 dark:text-slate-100
truncate">
                      {prospect.assignedTo?.fullName || 'Unassigned'}
                    </div>
                  </div>
                </div>

                {(prospect.email || prospect.phone) && (
                  <div className="mt-3 pt-3 border-t border-gray-100">
                    {prospect.email && (
                      <div className="flex items-center gap-2 text-xs text-
gray-600 dark:text-gray-500 mb-1">
                        <FiMail className="w-3 h-3" />
                        <span className="truncate">{prospect.email}</span>
                      </div>
                    )}
                    {prospect.phone && (
                      <div className="flex items-center gap-2 text-xs text-
gray-600 dark:text-gray-500">
                        <FiPhone className="w-3 h-3" />
                        <span>{prospect.phone}</span>
                      </div>
                    )}
                  </div>
```

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

          {/* Pagination */}
          {pagination.pages > 1 && (
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out px-4 py-3 flex items-center justify-between border-t border-
slate-200 dark:border-slate-700 sm:px-6">
              <div className="flex-1 flex justify-between sm:hidden">
                <button
                  onClick={() => handleFilterChange('page', Math.max(1,
filters.page - 1))}
                  disabled={filters.page === 1}
                  className="relative inline-flex items-center px-4 py-2 border
border-slate-300 dark:border-slate-600 text-sm font-medium rounded-md text-
slate-700 dark:text-slate-300 bg-white dark:bg-slate-800 transition-all
duration-200 ease-out hover:bg-slate-50 dark:hover:bg-slate-800
disabled:opacity-50"
                >
                  Previous
                </button>
                <button
                  onClick={() => handleFilterChange('page',
Math.min(pagination.pages, filters.page + 1))}
                  disabled={filters.page === pagination.pages}
                  className="ml-3 relative inline-flex items-center px-4 py-2
border border-slate-300 dark:border-slate-600 text-sm font-medium rounded-md text-
slate-700 dark:text-slate-300 bg-white dark:bg-slate-800 transition-all
duration-200 ease-out hover:bg-slate-50 dark:hover:bg-slate-800
disabled:opacity-50"
                >
                  Next
                </button>
              </div>
              <div className="hidden sm:flex-1 sm:flex sm:items-center
sm:justify-between">
                <div>
                  <p className="text-sm text-slate-700 dark:text-slate-300">
                    Showing <span className="font-medium">{((filters.page - 1)
* filters.limit) + 1}</span> to{' '}
                    <span className="font-medium">
                      {Math.min(filters.page * filters.limit, pagination.total)}
                    </span>{' '}
                    of <span className="font-medium">{pagination.total}</span>
results
                  </p>
                </div>
                <div>
                  <nav className="relative z-0 inline-flex rounded-md shadow-sm
dark:shadow-black/25 -space-x-px">
                    {/* Previous Page Button */}
                    <button
                      onClick={() => handleFilterChange('page', Math.max(1,
filters.page - 1))}
                      disabled={filters.page === 1}
                      className="relative inline-flex items-center px-2 py-2
rounded-l-md border border-gray-300 dark:border-slate-600 bg-white dark:bg-
```

```
slate-900 text-sm font-medium text-gray-500 dark:text-gray-400 hover:bg-slate-50
dark:hover:bg-slate-800 disabled:opacity-50"
                      >
                        <span className="sr-only">Previous</span>
                        <svg className="h-5 w-5" xmlns="http://www.w3.org/2000/
svg" viewBox="0 0 20 20" fill="currentColor" aria-hidden="true">
                          <path fillRule="evenodd" d="M12.707 5.293a1 1 0 010
1.414L9.414 10l3.293 3.293a1 1 0 01-1.414 1.414l-4-4a1 1 0 010-1.414l4-4a1 1 0
011.414 0z" clipRule="evenodd" />
                        </svg>
                      </button>

                      {/* Page Numbers */}
                      {Array.from({ length: pagination.pages }, (_, i) => i +
1).map((page) => (
                        <button
                          key={page}
                          onClick={() => handleFilterChange('page', page)}
                          className={`relative inline-flex items-center px-4 py-2
border text-sm font-medium ${
                            page === filters.page
                              ? 'z-10 bg-blue-50 dark:bg-blue-900/20 border-
blue-500 text-blue-600 dark:text-blue-400'
                              : 'bg-white dark:bg-slate-900 border-gray-300
dark:border-slate-600 text-gray-500 dark:text-gray-400 hover:bg-slate-50
dark:hover:bg-slate-800'
                          }`}
                        >
                          {page}
                        </button>
                      ))}

                      {/* Next Page Button */}
                      <button
                        onClick={() => handleFilterChange('page',
Math.min(pagination.pages, filters.page + 1))}
                        disabled={filters.page === pagination.pages}
                        className="relative inline-flex items-center px-2 py-2
rounded-r-md border border-gray-300 dark:border-slate-600 bg-white dark:bg-
slate-900 text-sm font-medium text-gray-500 dark:text-gray-400 hover:bg-slate-50
dark:hover:bg-slate-800 disabled:opacity-50"
                      >
                        <span className="sr-only">Next</span>
                        <svg className="h-5 w-5" xmlns="http://www.w3.org/2000/
svg" viewBox="0 0 20 20" fill="currentColor" aria-hidden="true">
                          <path fillRule="evenodd" d="M7.293 14.707a1 1 0
010-1.414L10.586 10 7.293 6.707a1 1 0 011.414-1.414l4 4a1 1 0 010 1.414l-4 4a1 1
0 01-1.414 0z" clipRule="evenodd" />
                        </svg>
                      </button>
                    </nav>
                  </div>
                </div>
              </div>
            )}
          </>
        )}
      </div>
```

```
      {/* Create/Edit Modal */}
      {(showCreateModal || showEditModal) && (
        <ProspectModal
          isOpen={showCreateModal || showEditModal}
          onClose={() => {
            setShowCreateModal(false);
            setShowEditModal(false);
            setSelectedProspect(null);
          }}
          prospect={selectedProspect}
          onSuccess={() => {
            fetchProspects();
            fetchStats();
            setShowCreateModal(false);
            setShowEditModal(false);
            setSelectedProspect(null);
          }}
        />
      )}

      {/* View Details Modal */}
      {showViewModal && selectedProspect && (
        <ViewProspectModal
          isOpen={showViewModal}
          onClose={() => {
            setShowViewModal(false);
            setSelectedProspect(null);
          }}
          prospect={selectedProspect}
          onEdit={() => {
            setShowViewModal(false);
            setShowEditModal(true);
          }}
          onConvert={() => {
            setShowViewModal(false);
            handleConvertToLead(selectedProspect._id);
          }}
          onDelete={() => {
            setShowViewModal(false);
            handleDelete(selectedProspect._id);
          }}
          userRole={user?.role}
        />
      )}
    </div>
  );
};

// Prospect Modal Component
const ProspectModal = ({ isOpen, onClose, prospect, onSuccess }) => {
  const { user, token } = useAuth();
  const [loading, setLoading] = useState(false);
  const [users, setUsers] = useState([]);
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    phone: '',
    company: '',
    designation: '',
```

```
    source: 'Other',
    sourceDetails: '',
    industry: '',
    companySize: 'Unknown',
    budget: '',
    budgetCurrency: 'USD',
    serviceInterest: '',
    requirements: '',
    timeline: 'Not specified',
    status: 'New',
    priority: 'Medium',
    assignedTo: '',
    lastContactDate: '',
    nextFollowUpDate: '',
    contactMethod: '',
    notes: '',
    tags: '',
    linkedinProfile: '',
    websiteUrl: ''
  });

  useEffect(() => {
    if (prospect) {
      setFormData({
        name: prospect.name || '',
        email: prospect.email || '',
        phone: prospect.phone || '',
        company: prospect.company || '',
        designation: prospect.designation || '',
        source: prospect.source || 'Other',
        sourceDetails: prospect.sourceDetails || '',
        industry: prospect.industry || '',
        companySize: prospect.companySize || 'Unknown',
        budget: prospect.budget || '',
        budgetCurrency: prospect.budgetCurrency || 'USD',
        serviceInterest: prospect.serviceInterest || '',
        requirements: prospect.requirements || '',
        timeline: prospect.timeline || 'Not specified',
        status: prospect.status || 'New',
        priority: prospect.priority || 'Medium',
        assignedTo: prospect.assignedTo?._id || '',
        lastContactDate: prospect.lastContactDate ?
prospect.lastContactDate.split('T')[0] : '',
        nextFollowUpDate: prospect.nextFollowUpDate ?
prospect.nextFollowUpDate.split('T')[0] : '',
        contactMethod: prospect.contactMethod || '',
        notes: prospect.notes || '',
        tags: prospect.tags?.join(', ') || '',
        linkedinProfile: prospect.linkedinProfile || '',
        websiteUrl: prospect.websiteUrl || ''
      });
    }
  }, [prospect]);

  // Fetch users for assignment
  useEffect(() => {
    const fetchUsers = async () => {
      try {
        const response = await authAPI.getUsers();
```

```
        if (response.data.success) {
          setUsers(response.data.data.filter(u =>
            ['Sales Person', 'Manager', 'Admin'].includes(u.role)
          ));
        }
      } catch (error) {
        console.error('Error fetching users:', error);
      }
    };
    fetchUsers();
  }, [token]);

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

    try {
      const submitData = {
        ...formData,
        budget: formData.budget ? parseFloat(formData.budget) : undefined,
        tags: formData.tags ? formData.tags.split(',').map(tag =>
tag.trim()).filter(Boolean) : [],
        lastContactDate: formData.lastContactDate || undefined,
        nextFollowUpDate: formData.nextFollowUpDate || undefined
      };

      // Remove empty strings
      Object.keys(submitData).forEach(key => {
        if (submitData[key] === '') {
          delete submitData[key];
        }
      });

      const response = prospect
        ? await prospectsAPI.update(prospect._id, submitData)
        : await prospectsAPI.create(submitData);

      if (response.data.success) {
        toast.success(prospect ? 'Prospect updated successfully' : 'Prospect
created successfully');
        onSuccess();
      }
    } catch (error) {
      console.error('Error saving prospect:', error);
      toast.error(error.response?.data?.message || 'Failed to save prospect');
    } finally {
      setLoading(false);
    }
  };

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

  if (!isOpen) return null;
```

```
  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center p-4 z-50">
      <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg max-w-4xl w-
full max-h-[90vh] overflow-y-auto shadow-sm dark:shadow-black/25">
        <div className="p-6">
          <div className="flex justify-between items-center mb-6">
            <h2 className="text-xl font-bold text-slate-900 dark:text-slate-100">
              {prospect ? 'Edit Prospect' : 'Add New Prospect'}
            </h2>
            <button
              onClick={onClose}
              className="text-gray-400 dark:text-gray-300 hover:text-gray-600"
            >
              '
            </button>
          </div>

          <form onSubmit={handleSubmit} className="space-y-6">
            {/* Basic Information */}
            <div>
              <h3 className="text-lg font-medium text-slate-900 dark:text-
slate-100 mb-4">Basic Information</h3>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Name
                  </label>
                  <input
                    type="text"
                    name="name"
                    value={formData.name}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="Contact person name"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Email
                  </label>
                  <input
                    type="email"
                    name="email"
                    value={formData.email}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="email@example.com"
                  />
                </div>
```

```
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Phone
                  </label>
                  <input
                    type="tel"
                    name="phone"
                    value={formData.phone}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="+1 (555) 123-4567"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Company
                  </label>
                  <input
                    type="text"
                    name="company"
                    value={formData.company}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="Company name"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Designation
                  </label>
                  <input
                    type="text"
                    name="designation"
                    value={formData.designation}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="Job title"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Industry
                  </label>
                  <input
                    type="text"
                    name="industry"
```

```
                    value={formData.industry}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="Technology, Healthcare, etc."
                  />
                </div>
              </div>
            </div>

            {/* Source & Business Info */}
            <div>
              <h3 className="text-lg font-medium text-slate-900 dark:text-
slate-100 mb-4">Source & Business Information</h3>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Source
                  </label>
                  <select
                    name="source"
                    value={formData.source}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  >
                    <option value="LinkedIn">LinkedIn</option>
                    <option value="Website">Website</option>
                    <option value="Referral">Referral</option>
                    <option value="Cold Call">Cold Call</option>
                    <option value="Email Campaign">Email Campaign</option>
                    <option value="Social Media">Social Media</option>
                    <option value="Event">Event</option>
                    <option value="Other">Other</option>
                  </select>
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Source Details
                  </label>
                  <input
                    type="text"
                    name="sourceDetails"
                    value={formData.sourceDetails}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="Additional source information"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
```

```
dark:text-slate-300 mb-1">
                    Company Size
                  </label>
                  <select
                    name="companySize"
                    value={formData.companySize}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  >
                    <option value="1-10">1-10</option>
                    <option value="11-50">11-50</option>
                    <option value="51-200">51-200</option>
                    <option value="201-500">201-500</option>
                    <option value="501-1000">501-1000</option>
                    <option value="1000+">1000+</option>
                    <option value="Unknown">Unknown</option>
                  </select>
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Budget
                  </label>
                  <div className="flex">
                    <select
                      name="budgetCurrency"
                      value={formData.budgetCurrency}
                      onChange={handleChange}
                      className="px-3 py-2 border border-slate-300 dark:border-
slate-600 rounded-l-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-offset-
slate-900 focus:ring-offset-2 focus:border-transparent"
                    >
                      <option value="USD">USD</option>
                      <option value="EUR">EUR</option>
                      <option value="GBP">GBP</option>
                      <option value="INR">INR</option>
                    </select>
                    <input
                      type="number"
                      name="budget"
                      value={formData.budget}
                      onChange={handleChange}
                      className="flex-1 px-3 py-2 border border-l-0 border-
slate-300 dark:border-slate-600 rounded-r-md focus:ring-2 focus:ring-blue-500
dark:focus:ring-offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                      placeholder="0"
                      min="0"
                    />
                  </div>
                </div>
              </div>
            </div>

            {/* Interest & Requirements */}
            <div>
              <h3 className="text-lg font-medium text-slate-900 dark:text-
```

```
slate-100 mb-4">Interest & Requirements</h3>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Service Interest
                  </label>
                  <input
                    type="text"
                    name="serviceInterest"
                    value={formData.serviceInterest}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="What services are they interested in?"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Timeline
                  </label>
                  <select
                    name="timeline"
                    value={formData.timeline}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  >
                    <option value="Immediate">Immediate</option>
                    <option value="Within 1 month">Within 1 month</option>
                    <option value="1-3 months">1-3 months</option>
                    <option value="3-6 months">3-6 months</option>
                    <option value="6+ months">6+ months</option>
                    <option value="Not specified">Not specified</option>
                  </select>
                </div>

                <div className="md:col-span-2">
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Requirements
                  </label>
                  <textarea
                    name="requirements"
                    value={formData.requirements}
                    onChange={handleChange}
                    rows={3}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="Detailed requirements and needs"
                  />
                </div>
              </div>
            </div>
```

```
            {/* Status & Assignment */}
            <div>
              <h3 className="text-lg font-medium text-slate-900 dark:text-
slate-100 mb-4">Status & Assignment</h3>
              <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Status
                  </label>
                  <select
                    name="status"
                    value={formData.status}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  >
                    <option value="New">New</option>
                    <option value="Contacted">Contacted</option>
                    <option value="Interested">Interested</option>
                    <option value="Not Interested">Not Interested</option>
                    <option value="Follow Up">Follow Up</option>
                    <option value="Qualified">Qualified</option>
                    <option value="Lost">Lost</option>
                  </select>
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Priority
                  </label>
                  <select
                    name="priority"
                    value={formData.priority}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  >
                    <option value="High">High</option>
                    <option value="Medium">Medium</option>
                    <option value="Low">Low</option>
                  </select>
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Assigned To
                  </label>
                  <select
                    name="assignedTo"
                    value={formData.assignedTo}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
```

```
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  >
                    <option value="">Unassigned</option>
                    {users.map(user => (
                      <option key={user._id} value={user._id}>
                        {user.fullName} ({user.role})
                      </option>
                    ))}
                  </select>
                </div>
              </div>
            </div>

            {/* Follow-up Information */}
            <div>
              <h3 className="text-lg font-medium text-slate-900 dark:text-
slate-100 mb-4">Follow-up Information</h3>
              <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Last Contact Date
                  </label>
                  <input
                    type="date"
                    name="lastContactDate"
                    value={formData.lastContactDate}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Next Follow-up Date
                  </label>
                  <input
                    type="date"
                    name="nextFollowUpDate"
                    value={formData.nextFollowUpDate}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Contact Method
                  </label>
                  <select
                    name="contactMethod"
                    value={formData.contactMethod}
                    onChange={handleChange}
```

```
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  >
                    <option value="">Select method</option>
                    <option value="Email">Email</option>
                    <option value="Phone">Phone</option>
                    <option value="LinkedIn">LinkedIn</option>
                    <option value="WhatsApp">WhatsApp</option>
                    <option value="Meeting">Meeting</option>
                    <option value="Other">Other</option>
                  </select>
                </div>
              </div>
            </div>

            {/* Additional Information */}
            <div>
              <h3 className="text-lg font-medium text-slate-900 dark:text-
slate-100 mb-4">Additional Information</h3>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    LinkedIn Profile
                  </label>
                  <input
                    type="url"
                    name="linkedinProfile"
                    value={formData.linkedinProfile}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="https://linkedin.com/in/username"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Website URL
                  </label>
                  <input
                    type="url"
                    name="websiteUrl"
                    value={formData.websiteUrl}
                    onChange={handleChange}
                    className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                    placeholder="https://company.com"
                  />
                </div>

                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Tags
```

```
                </label>
                <input
                  type="text"
                  name="tags"
                  value={formData.tags}
                  onChange={handleChange}
                  className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  placeholder="tag1, tag2, tag3"
                />
                <p className="text-xs text-slate-500 dark:text-gray-400
mt-1">Separate tags with commas</p>
              </div>

              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                  Notes
                </label>
                <textarea
                  name="notes"
                  value={formData.notes}
                  onChange={handleChange}
                  rows={3}
                  className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:ring-2 focus:ring-blue-500 dark:focus:ring-
offset-slate-900 focus:ring-offset-2 focus:border-transparent"
                  placeholder="Additional notes and observations"
                />
              </div>
            </div>
          </div>

          {/* Form Actions */}
          <div className="flex justify-end gap-4 pt-6 border-t">
            <button
              type="button"
              onClick={onClose}
              className="px-4 py-2 text-slate-700 dark:text-slate-300 bg-
gray-100 dark:bg-slate-600 rounded-lg hover:bg-gray-200"
            >
              Cancel
            </button>
            <button
              type="submit"
              disabled={loading}
              className="px-4 py-2 bg-blue-600 hover:bg-blue-700 dark:bg-
blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white rounded-lg disabled:opacity-50 flex items-
center gap-2"
            >
              {loading && <div className="animate-spin rounded-full h-4 w-4
border-b-2 border-white"></div>}
              {prospect ? 'Update Prospect' : 'Create Prospect'}
            </button>
          </div>
        </form>
      </div>
```

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

// View Prospect Modal Component
const ViewProspectModal = ({ isOpen, onClose, prospect, onEdit, onConvert,
onDelete, userRole }) => {
  // Local badge components for this modal
  const StatusBadge = ({ status }) => {
    const statusColors = {
      'New': 'bg-blue-100 text-blue-800',
      'Contacted': 'bg-yellow-100 text-yellow-800',
      'Interested': 'bg-green-100 text-green-800',
      'Not Interested': 'bg-red-100 text-red-800',
      'Follow Up': 'bg-purple-100 text-purple-800',
      'Qualified': 'bg-indigo-100 text-indigo-800',
      'Converted to Lead': 'bg-emerald-100 text-emerald-800',
      'Lost': 'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-gray-200'
    };

    return (
      <span className={`px-2 py-1 rounded-full text-xs font-medium
${statusColors[status] || 'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-
gray-200'}`}>
        {status}
      </span>
    );
  };

  const PriorityBadge = ({ priority }) => {
    const priorityColors = {
      'High': 'bg-red-100 text-red-800',
      'Medium': 'bg-yellow-100 text-yellow-800',
      'Low': 'bg-green-100 text-green-800'
    };

    return (
      <span className={`px-2 py-1 rounded-full text-xs font-medium
${priorityColors[priority] || 'bg-gray-100 dark:bg-slate-700 text-gray-800
dark:text-gray-200'}`}>
        {priority}
      </span>
    );
  };

  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center p-4 z-50">
      <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg max-w-4xl w-
full max-h-[90vh] overflow-y-auto shadow-sm dark:shadow-black/25">
        <div className="p-6">
          <div className="flex justify-between items-center mb-6">
            <h2 className="text-xl font-bold text-slate-900 dark:text-slate-100">
              View Prospect Details
            </h2>
```

```
            <button
              onClick={onClose}
              className="text-gray-400 dark:text-gray-300 hover:text-gray-600"
            >
              '
            </button>
          </div>

          <div className="space-y-6">
            {/* Prospect Details */}
            <div>
              <h3 className="text-lg font-medium text-slate-900 dark:text-
slate-100 mb-4">Prospect Details</h3>
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                  <div>
                    <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                      Name
                    </label>
                    <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.name || 'No Name'}</div>
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                      Email
                    </label>
                    <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.email || 'No Email'}</div>
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                      Phone
                    </label>
                    <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.phone || 'No Phone'}</div>
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                      Company
                    </label>
                    <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.company || 'No Company'}</div>
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                      Designation
                    </label>
                    <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.designation || 'No Designation'}</div>
                  </div>
                  <div>
                    <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                      Industry
                    </label>
```

```
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.industry || 'No Industry'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Source
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.source || 'No Source'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Status
                  </label>
                  <div className="text-sm text-slate-900 dark:text-slate-100">
                    <StatusBadge status={prospect.status} />
                  </div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Priority
                  </label>
                  <div className="text-sm text-slate-900 dark:text-slate-100">
                    <PriorityBadge priority={prospect.priority} />
                  </div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Assigned To
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.assignedTo?.fullName || 'Unassigned'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Created
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{new Date(prospect.createdAt).toLocaleDateString()}</div>
                </div>
              </div>
            </div>

            {/* Additional Details */}
            <div>
              <h3 className="text-lg font-medium text-slate-900 dark:text-
slate-100 mb-4">Additional Details</h3>
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Source Details
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
```

```
slate-100">{prospect.sourceDetails || 'No source details'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Company Size
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.companySize || 'Unknown'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Budget
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.budget ? `${prospect.budget} ${prospect.budgetCurrency}` :
'No budget'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Service Interest
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.serviceInterest || 'No service interest'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Timeline
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.timeline || 'Not specified'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Requirements
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.requirements || 'No requirements'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Last Contact Date
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.lastContactDate ? new
Date(prospect.lastContactDate).toLocaleDateString() : 'No last contact date'}</
div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Next Follow-up Date
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
```

```
slate-100">{prospect.nextFollowUpDate ? new
Date(prospect.nextFollowUpDate).toLocaleDateString() : 'No next follow-up date'}</
div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Contact Method
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.contactMethod || 'No contact method'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    LinkedIn Profile
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.linkedinProfile || 'No LinkedIn profile'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Website URL
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.websiteUrl || 'No website URL'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Tags
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.tags?.join(', ') || 'No tags'}</div>
                </div>
                <div>
                  <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">
                    Notes
                  </label>
                  <div className="text-sm text-slate-900 dark:text-
slate-100">{prospect.notes || 'No notes'}</div>
                </div>
              </div>
            </div>

            {/* Actions */}
            <div className="flex justify-end gap-4 pt-6 border-t">
              <button
                onClick={onClose}
                className="px-4 py-2 text-slate-700 dark:text-slate-300 bg-
gray-100 dark:bg-slate-600 rounded-lg hover:bg-gray-200"
              >
                Close
              </button>

              <button
                onClick={onEdit}
```

```
                  className="px-4 py-2 bg-blue-600 hover:bg-blue-700 dark:bg-
blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white rounded-lg"
                >
                  Edit
                </button>

                {prospect.status !== 'Converted to Lead' && (
                  <button
                    onClick={onConvert}
                    className="px-4 py-2 bg-purple-600 text-white rounded-lg
hover:bg-purple-700"
                  >
                    Convert to Lead
                  </button>
                )}

                {['Admin', 'Manager'].includes(userRole) && (
                  <button
                    onClick={onDelete}
                    className="px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-
red-700"
                  >
                    Delete
                  </button>
                )}
              </div>
            </div>
          </div>
        </div>
      </div>
    );
};

export default ProspectsPage;
```

src/pages/RepeatCustomersPage.jsx

```
import React, { useState, useEffect } from "react";
import Layout from "../components/Layout/Layout";
import { leadsAPI } from "../services/api";
import { useAuth } from "../context/AuthContext";

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const RepeatCustomersPage = () => {
  const { user } = useAuth();
  const [repeatCustomers, setRepeatCustomers] = useState([]);
  const [stats, setStats] = useState({ totalRepeatCustomers: 0, totalLeads: 0,
averageCoursesPerCustomer: 0 });
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [expandedCustomer, setExpandedCustomer] = useState(null);

  useEffect(() => {
    fetchRepeatCustomers();
  }, []);
```

```
  const fetchRepeatCustomers = async () => {
    try {
      setLoading(true);
      const response = await leadsAPI.getRepeatCustomers();

      if (response.data.success) {
        setRepeatCustomers(response.data.data);
        setStats(response.data.stats);
        setError(null);
      } else {
        setError("Failed to load repeat customer data");
      }
    } catch (err) {
      console.error("Error fetching repeat customers:", err);
      setError("Failed to load repeat customer data. Please try again.");
    } finally {
      setLoading(false);
    }
  };

  // Format date for display
  const formatDate = (dateString) => {
    const date = new Date(dateString);
    return date.toLocaleDateString();
  };

  // Toggle expanded customer view
  const toggleCustomerExpansion = (index) => {
    if (expandedCustomer === index) {
      setExpandedCustomer(null);
    } else {
      setExpandedCustomer(index);
    }
  };

  return (
    <Layout>
      <div className="container mx-auto px-4 py-8">
        <h1 className="text-2xl font-bold mb-6">Repeat Customers</h1>

        {/* Stats Cards */}
        <div className="grid grid-cols-1 md:grid-cols-3 gap-4 mb-8">
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-6 rounded-lg shadow-
md dark:shadow-2xl shadow-sm">
            <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Total Repeat Customers</p>
            <p className="text-3xl font-bold text-
blue-600">{stats.totalRepeatCustomers}</p>
          </div>
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-6 rounded-lg shadow-
md dark:shadow-2xl shadow-sm">
            <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Total Courses Purchased</p>
            <p className="text-3xl font-bold text-green-600">{stats.totalLeads}</
p>
          </div>
          <div className="bg-white dark:bg-slate-900 border border-slate-200
```

```
dark:border-slate-700 transition-all duration-200 ease-out p-6 rounded-lg shadow-
md dark:shadow-2xl shadow-sm">
            <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Avg. Courses per Customer</p>
            <p className="text-3xl font-bold text-
purple-600">{stats.averageCoursesPerCustomer}</p>
          </div>
        </div>

        {/* Error Message */}
        {error && (
          <div className="mb-4 p-4 bg-red-50 text-red-700 border border-red-200
rounded-md">
            {error}
          </div>
        )}

        {/* Loading Indicator */}
        {loading ? (
          <div className="flex justify-center items-center h-64">
            <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-
b-2 border-blue-500"></div>
          </div>
        ) : (
          <>
            {/* Customer List */}
            {repeatCustomers.length === 0 ? (
              <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-6 rounded-lg shadow-
md dark:shadow-2xl text-center shadow-sm">
                <p className="text-slate-500 dark:text-gray-400">No repeat
customers found.</p>
              </div>
            ) : (
              <div className="space-y-4">
                {repeatCustomers.map((customer, index) => (
                  <div key={index} className="bg-white dark:bg-slate-900 border
border-slate-200 dark:border-slate-700 transition-all duration-200 ease-out
rounded-lg shadow-md dark:shadow-2xl overflow-hidden shadow-sm">
                    {/* Customer Header - Always Visible */}
                    <div
                      className="p-4 bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out border-b border-slate-200 dark:border-slate-700 flex
justify-between items-center cursor-pointer hover:bg-slate-100 dark:hover:bg-
slate-700"
                      onClick={() => toggleCustomerExpansion(index)}
                    >
                      <div>
                        <h3 className="text-lg font-
medium">{customer.customerInfo.name}</h3>
                        <p className="text-sm text-slate-500 dark:text-gray-400">
                          {customer.customerInfo.email && (
                            <span className="mr-3">{customer.customerInfo.email}</
span>
                          )}
                          {customer.customerInfo.phone && (
                            <span>{customer.customerInfo.countryCode}
{customer.customerInfo.phone}</span>
                          )}
```

```
                        </p>
                      </div>
                      <div className="flex items-center">
                        <span className="inline-flex items-center px-3 py-1
rounded-full text-sm font-medium bg-purple-100 text-purple-800 mr-3">
                          {customer.leads.length} courses
                        </span>
                        <svg
                          className={`h-5 w-5 text-gray-500 dark:text-gray-400
dark:text-gray-400 transform ${expandedCustomer === index ? 'rotate-180' : ''}`}
                          fill="none"
                          viewBox="0 0 24 24"
                          stroke="currentColor"
                        >
                          <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M19 9l-7 7-7-7" />
                        </svg>
                      </div>
                    </div>

                    {/* Expandable Details Section */}
                    {expandedCustomer === index && (
                      <div className="p-4">
                        <h4 className="text-md font-medium mb-2">Course History</
h4>
                        <div className="overflow-x-auto">
                          <table className="min-w-full divide-y divide-slate-200
dark:divide-slate-700">
                            <thead className="bg-gray-50 dark:bg-slate-800
transition-all duration-200 ease-out">
                              <tr>
                                <th scope="col" className="px-6 py-3 text-left
text-xs font-medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                                  Course
                                </th>
                                <th scope="col" className="px-6 py-3 text-left
text-xs font-medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                                  Date
                                </th>
                                <th scope="col" className="px-6 py-3 text-left
text-xs font-medium text-slate-500 dark:text-gray-400 uppercase tracking-wider">
                                  Sales Person
                                </th>
                              </tr>
                            </thead>
                            <tbody className="bg-white dark:bg-slate-900
transition-all duration-200 ease-out divide-y divide-slate-200 dark:divide-
slate-700">
                              {customer.leads.map((lead, leadIndex) => (
                                <tr key={leadIndex} className="hover:bg-slate-50
dark:hover:bg-slate-800 dark:bg-slate-800 transition-all duration-200 ease-out">
                                  <td className="px-6 py-4 whitespace-nowrap text-
sm font-medium text-slate-900 dark:text-slate-100">
                                    {lead.course}
                                  </td>
                                  <td className="px-6 py-4 whitespace-nowrap text-
sm text-slate-500 dark:text-gray-400">
                                    {formatDate(lead.createdAt)}
                                  </td>
```

```
                                    <td className="px-6 py-4 whitespace-nowrap text-
sm text-slate-500 dark:text-gray-400">
                                      {lead.salesPerson}
                                    </td>
                                  </tr>
                                ))}
                              </tbody>
                            </table>
                          </div>
                        </div>
                      )}
                    </div>
                  ))}
                </div>
              )}
            </>
          )}
        </div>
      </Layout>
    );
};

export default RepeatCustomersPage;
```

src/pages/SalesCreatePage.jsx

```
import React, { useState, useEffect } from 'react';
import { salesAPI, authAPI, currencyAPI } from '../services/api';
import { useAuth } from '../context/AuthContext';
import Layout from '../components/Layout/Layout';
import { FaPlus, FaSave, FaTimes } from 'react-icons/fa';
import toast from 'react-hot-toast';
import CurrencySelector from '../components/CurrencySelector';

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const SalesCreatePage = () => {
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);
  const [leadPersons, setLeadPersons] = useState([]);
  const [exchangeRates, setExchangeRates] = useState({});
  const [formData, setFormData] = useState({
    date: new Date().toISOString().split('T')[0],
    customerName: '',
    country: '',
    course: '',
    countryCode: '',
    contactNumber: '',
    email: '',
    pseudoId: '',
    leadPerson: '',
    source: '',
    clientRemark: '',
    feedback: '',
    totalCost: 0,
    totalCostCurrency: 'USD',
    tokenAmount: 0,
    tokenAmountCurrency: 'USD'
```

```
});

// Get user from auth context
const { user } = useAuth();

// Load lead persons and currency rates when component mounts
useEffect(() => {
  loadLeadPersons();
  loadCurrencyRates();
}, []);

const loadLeadPersons = async () => {
  try {
    const res = await authAPI.getUsers('Lead Person');
    if (res.data.success) {
      setLeadPersons(res.data.data || []);
    } else {
      toast.error('Failed to load lead persons');
    }
  } catch (err) {
    console.error('Error loading lead persons:', err);
    toast.error('Failed to load lead persons');
  }
};

const loadCurrencyRates = async () => {
  try {
    const res = await currencyAPI.getRates();
    if (res.data.success) {
      setExchangeRates(res.data.rates || {});
    }
  } catch (err) {
    console.error('Error loading currency rates:', err);
  }
};

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

const handleCurrencyChange = (field, currency) => {
  setFormData(prev => ({
    ...prev,
    [`${field}Currency`]: currency
  }));
};

const handleSubmit = async (e) => {
  e.preventDefault();
  setLoading(true);
  setError(null);

  try {
    // Validate the form
    if (!formData.leadPerson) {
```

```
        throw new Error('Please select a Lead Person');
      }

      // Check if user info is available
      if (!user) {
        throw new Error('User information not available. Please log in again.');
      }

      // Prepare sale data for submission
      const saleData = {
        date: formData.date,
        customerName: formData.customerName,
        country: formData.country,
        course: formData.course,
        countryCode: formData.countryCode,
        contactNumber: formData.contactNumber,
        email: formData.email,
        pseudoId: formData.pseudoId,
        totalCost: parseFloat(formData.totalCost),
        totalCostCurrency: formData.totalCostCurrency,
        tokenAmount: parseFloat(formData.tokenAmount),
        tokenAmountCurrency: formData.tokenAmountCurrency,
        source: formData.source,
        clientRemark: formData.clientRemark,
        feedback: formData.feedback
      };

      // Use our new API method to create a sale with a specific lead person
      const res = await salesAPI.createSaleWithLeadPerson(saleData,
formData.leadPerson);

      if (res.data.success) {
        toast.success('Sale created successfully! This sale will appear on the
lead person\'s dashboard.');

        // Reset form
        setFormData({
          date: new Date().toISOString().split('T')[0],
          customerName: '',
          country: '',
          course: '',
          countryCode: '',
          contactNumber: '',
          email: '',
          pseudoId: '',
          leadPerson: '',
          source: '',
          clientRemark: '',
          feedback: '',
          totalCost: 0,
          totalCostCurrency: 'USD',
          tokenAmount: 0,
          tokenAmountCurrency: 'USD'
        });
      }
    } catch (err) {
      console.error('Error creating sale:', err);
      setError(err.response?.data?.message || err.message || 'Failed to create
sale');
```

```
      toast.error(err.response?.data?.message || err.message || 'Failed to create
sale');
    } finally {
      setLoading(false);
    }
  };

  return (
    <Layout>
      <div className="container mx-auto p-4">
        <div className="flex justify-between items-center mb-6">
          <h1 className="text-2xl font-bold">Create Sale for Lead Person</h1>
        </div>

        {error && (
          <div className="bg-red-100 border border-red-400 text-red-700 px-4 py-3
rounded relative mb-4">
            {error}
          </div>
        )}

        <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out shadow-md
dark:shadow-2xl rounded-lg p-6 mb-6 shadow-sm">
          <form onSubmit={handleSubmit}>
            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
              {/* Date */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Date</label>
                <input
                  type="date"
                  name="date"
                  value={formData.date}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required
                />
              </div>

              {/* Customer Name */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Customer Name</label>
                <input
                  type="text"
                  name="customerName"
                  value={formData.customerName}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required
                />
              </div>

              {/* Country */}
              <div>
                <label className="block text-sm font-medium text-slate-700
```

```
dark:text-slate-300 mb-1">Country</label>
                <input
                  type="text"
                  name="country"
                  value={formData.country}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required
                />
              </div>

              {/* Course */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Course</label>
                <input
                  type="text"
                  name="course"
                  value={formData.course}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required
                />
              </div>

              {/* Country Code */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Country Code</label>
                <input
                  type="text"
                  name="countryCode"
                  value={formData.countryCode}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  placeholder="e.g. +1, +91"
                />
              </div>

              {/* Phone Number */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Phone Number</label>
                <input
                  type="text"
                  name="contactNumber"
                  value={formData.contactNumber}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required
                />
              </div>

              {/* Email */}
              <div>
```

```
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Email</label>
                <input
                  type="email"
                  name="email"
                  value={formData.email}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                />
              </div>

              {/* Pseudo ID */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Pseudo ID</label>
                <input
                  type="text"
                  name="pseudoId"
                  value={formData.pseudoId}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                />
              </div>

              {/* Lead Person - This is the key field */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Lead Person</label>
                <select
                  name="leadPerson"
                  value={formData.leadPerson}
                  onChange={handleChange}
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                  required
                >
                  <option value="">Select Lead Person</option>
                  {leadPersons.map(person => (
                    <option key={person._id} value={person._id}>
                      {person.fullName}
                    </option>
                  ))}
                </select>
                <p className="mt-1 text-xs text-slate-500 dark:text-gray-400">
                  The lead person will see this sale on their dashboard
                </p>
              </div>

              {/* Source */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Source</label>
                <input
                  type="text"
                  name="source"
                  value={formData.source}
                  onChange={handleChange}
```

```
                  className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
                />
              </div>

              {/* Total Cost */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Total Cost</label>
                <div className="flex">
                  <input
                    type="number"
                    name="totalCost"
                    value={formData.totalCost}
                    onChange={handleChange}
                    className="w-2/3 p-2 border border-slate-300 dark:border-
slate-600 rounded-l-md"
                    required
                  />
                  <select
                    name="totalCostCurrency"
                    value={formData.totalCostCurrency}
                    onChange={e => handleCurrencyChange('totalCost',
e.target.value)}
                    className="w-1/3 p-2 border border-slate-300 dark:border-
slate-600 rounded-r-md"
                  >
                    <option value="USD">USD</option>
                    <option value="EUR">EUR</option>
                    <option value="GBP">GBP</option>
                    <option value="INR">INR</option>
                  </select>
                </div>
              </div>

              {/* Token Amount */}
              <div>
                <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Token Amount</label>
                <div className="flex">
                  <input
                    type="number"
                    name="tokenAmount"
                    value={formData.tokenAmount}
                    onChange={handleChange}
                    className="w-2/3 p-2 border border-slate-300 dark:border-
slate-600 rounded-l-md"
                    required
                  />
                  <select
                    name="tokenAmountCurrency"
                    value={formData.tokenAmountCurrency}
                    onChange={e => handleCurrencyChange('tokenAmount',
e.target.value)}
                    className="w-1/3 p-2 border border-slate-300 dark:border-
slate-600 rounded-r-md"
                  >
                    <option value="USD">USD</option>
                    <option value="EUR">EUR</option>
```

```
                  <option value="GBP">GBP</option>
                  <option value="INR">INR</option>
                </select>
              </div>
            </div>
          </div>

          {/* Client Remark */}
          <div className="mt-4">
            <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Client Remark</label>
            <textarea
              name="clientRemark"
              value={formData.clientRemark}
              onChange={handleChange}
              rows="2"
              className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
            ></textarea>
          </div>

          {/* Feedback */}
          <div className="mt-4">
            <label className="block text-sm font-medium text-slate-700
dark:text-slate-300 mb-1">Feedback</label>
            <textarea
              name="feedback"
              value={formData.feedback}
              onChange={handleChange}
              rows="2"
              className="w-full p-2 border border-slate-300 dark:border-
slate-600 rounded-md"
            ></textarea>
          </div>

          <div className="mt-6 flex justify-end">
            <button
              type="submit"
              disabled={loading}
              className="px-4 py-2 bg-blue-600 hover:bg-blue-700 dark:bg-
blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white rounded-md disabled:bg-blue-300 flex items-
center"
            >
              {loading ? (
                <>
                  <div className="animate-spin mr-2 h-4 w-4 border-2 border-
white rounded-full border-t-transparent"></div>
                  Processing...
                </>
              ) : (
                <>
                  <FaSave className="mr-2" /> Create Sale for Lead Person
                </>
              )}
            </button>
          </div>
        </form>
      </div>
```

```
        <div className="bg-blue-50 border border-blue-200 text-blue-700 p-4
rounded-md mb-6">
          <h2 className="text-lg font-semibold mb-2">How This Works</h2>
          <p className="mb-2">
            When you create a sale using this form and select a Lead Person, the
sale will:
          </p>
          <ul className="list-disc pl-5 space-y-1">
            <li>Be credited to you as the Sales Person</li>
            <li>Appear on the Lead Person's dashboard</li>
            <li>Allow the Lead Person to track your sales progress</li>
            <li>Create transparency and collaboration between Sales and Lead
teams</li>
          </ul>
        </div>
      </div>
    </Layout>
  );
};

export default SalesCreatePage;
```

src/pages/SalesPage.jsx

```
// src/pages/SalesPage.jsxÐ
import React, { useState, useEffect } from "react";Ð
import { leadsAPI } from "../services/api";Ð
import { useAuth } from "../context/AuthContext";Ð
import Layout from "../components/Layout/Layout";Ð
import { FaWhatsapp, FaEnvelope, FaPhone, FaFilter, FaCalendarAlt } from "react-
icons/fa";Ð
Ð
import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const SalesPage = () => {Ð
  const { user } = useAuth();Ð
  const [leads, setLeads] = useState([]);Ð
  const [filteredLeads, setFilteredLeads] = useState([]);Ð
  const [loading, setLoading] = useState(true);Ð
  const [error, setError] = useState(null);Ð
  const [feedback, setFeedback] = useState({});Ð
  const [updating, setUpdating] = useState({});Ð
  const [selectedLead, setSelectedLead] = useState(null);Ð
  const [refreshTrigger, setRefreshTrigger] = useState(0); // To trigger refreshesÐ
  Ð
  // Date filtering stateÐ
  const [filterMonth, setFilterMonth] = useState(new Date().getMonth() + 1); //
Current month (1-12)Ð
  const [filterYear, setFilterYear] = useState(new Date().getFullYear()); //
Current yearÐ
  const [showCurrentMonth, setShowCurrentMonth] = useState(true); // Flag to show
current month by defaultÐ
  Ð
  // Status update stateÐ
  const [statusUpdating, setStatusUpdating] = useState({});Ð
  Ð
  // Lead status optionsÐ
```

```
  const statusOptions = [Đ
    'New', Đ
    'Contacted', Đ
    'Qualified',  Đ
    'Lost', Đ
    'Converted', Đ
    'Introduction', Đ
    'Acknowledgement', Đ
    'Question', Đ
    'Future Promise', Đ
    'Payment', Đ
    'Analysis'Đ
  ];Đ
  Đ
  // Generate month optionsĐ
  const months = [Đ
    { value: 1, label: "January" },Đ
    { value: 2, label: "February" },Đ
    { value: 3, label: "March" },Đ
    { value: 4, label: "April" },Đ
    { value: 5, label: "May" },Đ
    { value: 6, label: "June" },Đ
    { value: 7, label: "July" },Đ
    { value: 8, label: "August" },Đ
    { value: 9, label: "September" },Đ
    { value: 10, label: "October" },Đ
    { value: 11, label: "November" },Đ
    { value: 12, label: "December" }Đ
  ];Đ
  Đ
  // Generate year options (5 years back from current year)Đ
  const currentYear = new Date().getFullYear();Đ
  const years = Array.from({ length: 6 }, (_, i) => currentYear - i);Đ
Đ
  // Fetch leads assigned to the sales personĐ
  useEffect(() => {Đ
    fetchLeads();Đ
  }, [user, refreshTrigger]);Đ
  Đ
  // Apply date filters when leads, month, or year changesĐ
  useEffect(() => {Đ
    if (leads.length > 0) {Đ
      filterLeadsByDate();Đ
    }Đ
  }, [leads, filterMonth, filterYear, showCurrentMonth]);Đ
  Đ
  // Function to filter leads by selected dateĐ
  const filterLeadsByDate = () => {Đ
    if (showCurrentMonth) {Đ
      // Show current month dataĐ
      const currentMonth = new Date().getMonth() + 1; // 1-12Đ
      const currentYear = new Date().getFullYear();Đ
      Đ
      const filtered = leads.filter(lead => {Đ
        const leadDate = new Date(lead.createdAt);Đ
        return (Đ
          leadDate.getMonth() + 1 === currentMonth && Đ
          leadDate.getFullYear() === currentYearĐ
        );Đ
```

```
      });Đ
      Đ
      setFilteredLeads(filtered);Đ
    } else {Đ
      // Show selected month/year dataĐ
      const filtered = leads.filter(lead => {Đ
        const leadDate = new Date(lead.createdAt);Đ
        return (Đ
          leadDate.getMonth() + 1 === filterMonth && Đ
          leadDate.getFullYear() === filterYearĐ
        );Đ
      });Đ
      Đ
      setFilteredLeads(filtered);Đ
    }Đ
  };Đ
Đ
  const fetchLeads = async () => {Đ
    try {Đ
      setLoading(true);Đ
      console.log("Fetching leads for sales person:", user?.fullName);Đ
      Đ
      const response = await leadsAPI.getAssigned();Đ
      console.log("Leads response:", response.data);Đ
      Đ
      if (response.data.success) {Đ
        setLeads(response.data.data);Đ
        Đ
        // Initialize feedback state with current valuesĐ
        const feedbackState = {};Đ
        response.data.data.forEach(lead => {Đ
          feedbackState[lead._id] = lead.feedback || '';Đ
        });Đ
        setFeedback(feedbackState);Đ
      } else {Đ
        setError("Failed to load leads");Đ
      }Đ
    } catch (err) {Đ
      console.error("Error fetching leads:", err);Đ
      setError("Failed to load leads. Please try again.");Đ
    } finally {Đ
      setLoading(false);Đ
    }Đ
  };Đ
Đ
  // Handle feedback changeĐ
  const handleFeedbackChange = (leadId, value) => {Đ
    setFeedback(prev => ({Đ
      ...prev,Đ
      [leadId]: valueĐ
    }));Đ
  };Đ
Đ
  // Handle feedback updateĐ
  const updateFeedback = async (leadId) => {Đ
    if (!leadId || !feedback[leadId]) return;Đ
    Đ
    try {Đ
      setUpdating(prev => ({ ...prev, [leadId]: true }));Đ
```

```
      Ð
      const response = await leadsAPI.updateFeedback(leadId, feedback[leadId]);Ð
      Ð
      if (response.data.success) {Ð
        // Update the leads state with the updated leadÐ
        setLeads(leads.map(lead => Ð
          lead._id === leadId ? response.data.data : leadÐ
        ));Ð
        alert("Feedback updated successfully!");Ð
      } else {Ð
        setError("Failed to update feedback");Ð
      }Ð
    } catch (err) {Ð
      console.error("Error updating feedback:", err);Ð
      setError("Failed to update feedback. Please try again.");Ð
    } finally {Ð
      setUpdating(prev => ({ ...prev, [leadId]: false }));Ð
    }Ð
  };Ð
Ð
  // Handle status updateÐ
  const updateLeadStatus = async (leadId, newStatus) => {Ð
    try {Ð
      setStatusUpdating(prev => ({ ...prev, [leadId]: true }));Ð
      setError(null); // Clear any previous errorsÐ
      Ð
      const leadToUpdate = leads.find(lead => lead._id === leadId);Ð
      if (!leadToUpdate) return;Ð
      Ð
      // Only send the status field to updateÐ
      const updatedData = { status: newStatus };Ð
      Ð
      console.log("Updating lead status:", leadId, newStatus);Ð
      const response = await leadsAPI.update(leadId, updatedData);Ð
      Ð
      if (response.data.success) {Ð
        // Update the leads state with the updated leadÐ
        setLeads(leads.map(lead => Ð
          lead._id === leadId ? response.data.data : leadÐ
        ));Ð
        Ð
        // Also update filtered leads to see the change immediatelyÐ
        setFilteredLeads(filteredLeads.map(lead => Ð
          lead._id === leadId ? response.data.data : leadÐ
        ));Ð
        Ð
        console.log("Lead status updated successfully");Ð
        // Show success messageÐ
        setFeedback({Ð
          ...feedback,Ð
          statusSuccess: `Status updated to "${newStatus}" successfully!`,Ð
          leadIdÐ
        });Ð
        Ð
        // Clear success message after 3 secondsÐ
        setTimeout(() => {Ð
          setFeedback(prev => ({Ð
            ...prev,Ð
            statusSuccess: null,Ð
```

```
            leadId: nullÐ
          }));Ð
        }, 3000);Ð
        Ð
        // Trigger a refresh of leadsÐ
        setRefreshTrigger(prev => prev + 1);Ð
      } else {Ð
        console.error("Failed to update lead status:", response.data);Ð
        setError("Failed to update lead status. Please try again.");Ð
      }Ð
    } catch (err) {Ð
      console.error("Error updating lead status:", err);Ð
      if (err.response && err.response.status === 403) {Ð
        setError("You don't have permission to update this lead status. Contact
your manager.");Ð
      } else {Ð
        setError("Failed to update lead status. Please try again.");Ð
      }Ð
    } finally {Ð
      setStatusUpdating(prev => ({ ...prev, [leadId]: false }));Ð
    }Ð
  };Ð
Ð
  // Handle month changeÐ
  const handleMonthChange = (e) => {Ð
    setFilterMonth(parseInt(e.target.value));Ð
    setShowCurrentMonth(false);Ð
  };Ð
  Ð
  // Handle year changeÐ
  const handleYearChange = (e) => {Ð
    setFilterYear(parseInt(e.target.value));Ð
    setShowCurrentMonth(false);Ð
  };Ð
  Ð
  // Handle reset to current monthÐ
  const handleResetToCurrentMonth = () => {Ð
    setFilterMonth(new Date().getMonth() + 1);Ð
    setFilterYear(new Date().getFullYear());Ð
    setShowCurrentMonth(true);Ð
  };Ð
Ð
  // Format date for displayÐ
  const formatDate = (dateString) => {Ð
    const date = new Date(dateString);Ð
    return date.toLocaleDateString();Ð
  };Ð
Ð
  // View lead detailsÐ
  const viewLeadDetails = (lead) => {Ð
    setSelectedLead(lead);Ð
  };Ð
Ð
  // Close lead details modalÐ
  const closeLeadDetails = () => {Ð
    setSelectedLead(null);Ð
  };Ð
Ð
  // Open WhatsApp with the phone numberÐ
```

```
  const openWhatsApp = (phone, countryCode) => {Đ
    if (!phone) return;Đ
    Đ
    // Format the phone number for WhatsAppĐ
    const formattedPhone = `${countryCode || '+91'}${phone.replace(/\D/g, '')}`;Đ
    const whatsappUrl = `https://wa.me/${formattedPhone.replace(/\+/g, '')}`;Đ
    Đ
    // Open WhatsApp in a new tabĐ
    window.open(whatsappUrl, '_blank');Đ
  };Đ
Đ
  // Open email client with the email addressĐ
  const openEmail = (email) => {Đ
    if (!email) return;Đ
    window.location.href = `mailto:${email}`;Đ
  };Đ
Đ
  return (Đ
    <Layout>Đ
      <div className="container mx-auto p-6">Đ
        <h2 className="text-3xl font-bold mb-6">My Assigned Leads</h2>Đ
        Đ
        {/* Error message with dismissal option */}Đ
        {error && (Đ
          <div className="mb-4 p-3 bg-red-50 text-red-700 border border-red-200
rounded-md flex justify-between items-center">Đ
            <div className="flex-1">{error}</div>Đ
            <button Đ
              onClick={() => setError(null)} Đ
              className="text-red-700 hover:text-red-900"Đ
            >Đ
              &times;Đ
            </button>Đ
          </div>Đ
        )}Đ
        Đ
        {/* Loading indicator */}Đ
        {loading ? (Đ
          <div className="text-center py-10">Đ
            <div className="inline-block animate-spin rounded-full h-8 w-8 border-
t-2 border-b-2 border-blue-500"></div>Đ
            <p className="mt-2 text-gray-600 dark:text-gray-500">Loading your
assigned leads...</p>Đ
          </div>Đ
        ) : leads.length === 0 ? (Đ
          <div className="text-center py-10 bg-gray-50 dark:bg-slate-800
transition-all duration-200 ease-out rounded-lg">Đ
            <p className="text-gray-600 dark:text-gray-500">No leads assigned to
you yet.</p>Đ
          </div>Đ
        ) : (Đ
          <>Đ
            {/* Date Filter Controls - Enhanced */}Đ
            <div className="mb-6 p-4 bg-white dark:bg-slate-900 border border-
slate-200 dark:border-slate-700 transition-all duration-200 ease-out rounded-lg
shadow-md dark:shadow-2xl shadow-sm">Đ
              <h3 className="text-lg font-semibold mb-3 flex items-center">Đ
                <FaFilter className="mr-2 text-blue-500" /> Đ
                Filter Leads by DateĐ
```

```
              </h3>Đ
              <div className="flex flex-wrap items-center gap-4">Đ
                <div className="flex items-center">Đ
                  <FaCalendarAlt className="text-blue-500 mr-2" />Đ
                  <div>Đ
                    <label htmlFor="month" className="block text-sm font-medium
text-gray-600 dark:text-gray-500 mb-1">Month</label>Đ
                    <selectĐ
                      id="month"Đ
                      value={filterMonth}Đ
                      onChange={handleMonthChange}Đ
                      className="border border-slate-300 dark:border-slate-600
rounded-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"Đ
                      disabled={showCurrentMonth}Đ
                    >Đ
                      {months.map(month => (Đ
                        <option key={month.value} value={month.value}>Đ
                          {month.label}Đ
                        </option>Đ
                      ))}Đ
                    </select>Đ
                  </div>Đ
                </div>Đ
                Đ
                <div>Đ
                  <label htmlFor="year" className="block text-sm font-medium text-
gray-600 dark:text-gray-500 mb-1">Year</label>Đ
                  <selectĐ
                    id="year"Đ
                    value={filterYear}Đ
                    onChange={handleYearChange}Đ
                    className="border border-slate-300 dark:border-slate-600
rounded-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"Đ
                    disabled={showCurrentMonth}Đ
                  >Đ
                    {years.map(year => (Đ
                      <option key={year} value={year}>Đ
                        {year}Đ
                      </option>Đ
                    ))}Đ
                  </select>Đ
                </div>Đ
                Đ
                <div className="flex items-center ml-4">Đ
                  <inputĐ
                    id="currentMonth"Đ
                    type="checkbox"Đ
                    checked={showCurrentMonth}Đ
                    onChange={() => setShowCurrentMonth(!showCurrentMonth)}Đ
                    className="h-4 w-4 text-blue-600 focus:ring-2 focus:ring-
blue-500 dark:focus:ring-offset-slate-900 focus:ring-offset-2 border-slate-300
dark:border-slate-600 rounded"Đ
                  />Đ
                  <label htmlFor="currentMonth" className="ml-2 text-sm text-
gray-600 dark:text-gray-500">Đ
                    Show current month onlyĐ
                  </label>Đ
```

```
                </div>Đ
                Đ
                <buttonĐ
                  onClick={handleResetToCurrentMonth}Đ
                  className="px-4 py-2 bg-blue-100 text-blue-700 rounded-md
hover:bg-blue-200 ml-auto flex items-center"Đ
                >Đ
                  <FaCalendarAlt className="mr-2" />Đ
                  Reset to Current MonthĐ
                </button>Đ
              </div>Đ
              Đ
              {/* Display filter result summary */}Đ
              <div className="mt-3 text-sm text-gray-600 dark:text-gray-500">Đ
                Showing {filteredLeads.length} lead{filteredLeads.length !== 1 ?
's' : ''} Đ
                {showCurrentMonth Đ
                  ? ' for the current month' Đ
                  : ` for ${months.find(m => m.value === filterMonth)?.label}
${filterYear}`Đ
                }Đ
              </div>Đ
            </div>Đ
            Đ
            {/* Leads Table */}Đ
            <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl overflow-hidden shadow-sm">Đ
              <div className="overflow-x-auto">Đ
                <table className="min-w-full divide-y divide-slate-200
dark:divide-slate-700">Đ
                  <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">Đ
                    <tr>Đ
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Name</th>Đ
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Contact</th>Đ
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Course</th>Đ
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Status</th>Đ
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Date</th>Đ
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Feedback</th>Đ
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Actions</th>Đ
                    </tr>Đ
                  </thead>Đ
                  <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-slate-200 dark:divide-slate-700">Đ
                    {filteredLeads.map(lead => (Đ
                      <tr key={lead._id} className="hover:bg-slate-50
dark:hover:bg-slate-800 dark:bg-slate-800 transition-all duration-200 ease-out">Đ
                        <td className="px-6 py-4 whitespace-nowrap">Đ
                          <div className="text-sm font-medium text-slate-900
dark:text-slate-100">{lead.name}</div>Đ
                        </td>Đ
```

```
                        <td className="px-6 py-4">Ð
                          <div className="flex flex-col space-y-1">Ð
                            {lead.phone && (Ð
                              <div className="flex items-center">Ð
                                <button Ð
                                  onClick={() => openWhatsApp(lead.phone,
lead.countryCode)}Ð
                                  className="text-sm text-slate-900 dark:text-
slate-100 flex items-center hover:text-green-600"Ð
                                  title="Open in WhatsApp"Ð
                                >Ð
                                  <FaWhatsapp className="mr-1 text-green-500" /> Ð
                                  {lead.countryCode || '+91'} {lead.phone}Ð
                                </button>Ð
                              </div>Ð
                            )}Ð
                            {lead.email && (Ð
                              <div className="flex items-center">Ð
                                <button Ð
                                  onClick={() => openEmail(lead.email)}Ð
                                  className="text-sm text-slate-500 dark:text-
gray-400 flex items-center hover:text-blue-600"Ð
                                  title="Send email"Ð
                                >Ð
                                  <FaEnvelope className="mr-1 text-blue-500" /> Ð
                                  {lead.email}Ð
                                </button>Ð
                              </div>Ð
                            )}Ð
                          </div>Ð
                        </td>Ð
                        <td className="px-6 py-4 whitespace-nowrap">Ð
                          <div className="text-sm text-slate-900 dark:text-
slate-100">{lead.course}</div>Ð
                        </td>Ð
                        <td className="px-6 py-4 whitespace-nowrap">Ð
                          <div className="relative">Ð
                            <selectÐ
                              value={lead.status}Ð
                              onChange={(e) => updateLeadStatus(lead._id,
e.target.value)}Ð
                              disabled={statusUpdating[lead._id]}Ð
                              className={`text-sm px-2 py-1 rounded ${Ð
                                lead.status === 'New' ? 'bg-blue-100 text-
blue-800' :Ð
                                lead.status === 'Contacted' ? 'bg-teal-100 text-
teal-800' :Ð
                                lead.status === 'Qualified' ? 'bg-indigo-100 text-
indigo-800' :Ð
                                lead.status === 'Lost' ? 'bg-red-100 text-
red-800' :Ð
                                lead.status === 'Converted' ? 'bg-green-100 text-
green-800' :Ð
                                lead.status === 'Introduction' ? 'bg-purple-100
text-purple-800' :Ð
                                lead.status === 'Acknowledgement' ? 'bg-
yellow-100 text-yellow-800' :Ð
                                lead.status === 'Question' ? 'bg-indigo-100 text-
indigo-800' :Ð
```

```
                                lead.status === 'Future Promise' ? 'bg-cyan-100
text-cyan-800' :Đ
                                lead.status === 'Payment' ? 'bg-green-100 text-
green-800' :Đ
                                lead.status === 'Analysis' ? 'bg-red-100 text-
red-800' :Đ
                                'bg-gray-100 dark:bg-slate-700 text-gray-800
dark:text-gray-200'Đ
                              } border border-transparent focus:outline-none
focus:ring-2 focus:ring-blue-500 dark:focus:ring-blue-400 focus:border-blue-500
dark:focus:border-blue-400 appearance-none cursor-pointer w-full pr-8`}Đ
                            >Đ
                              {statusOptions.map(status => (Đ
                                <option key={status} value={status}>{status}</
option>Đ
                              ))}Đ
                            </select>Đ
                            <div className="pointer-events-none absolute inset-
y-0 right-0 flex items-center px-2 text-slate-700 dark:text-slate-300">Đ
                              <svg className="fill-current h-4 w-4" xmlns="http://
www.w3.org/2000/svg" viewBox="0 0 20 20">Đ
                                <path d="M9.293 12.95l.707.707L15.657
8l-1.414-1.414L10 10.828 5.757 6.586 4.343 8z"/>Đ
                              </svg>Đ
                            </div>Đ
                            {statusUpdating[lead._id] && (Đ
                              <div className="absolute top-1/2 right-8 transform -
translate-y-1/2">Đ
                                <div className="animate-spin h-4 w-4 border-2
border-blue-500 rounded-full border-t-transparent"></div>Đ
                              </div>Đ
                            )}Đ
                            {feedback.statusSuccess && feedback.leadId ===
lead._id && (Đ
                              <div className="absolute top-full left-0 mt-1 bg-
green-100 text-green-800 text-xs px-2 py-1 rounded whitespace-normal w-48 z-10">Đ
                                {feedback.statusSuccess}Đ
                              </div>Đ
                            )}Đ
                          </div>Đ
                        </td>Đ
                        <td className="px-6 py-4 whitespace-nowrap">Đ
                          <div className="text-sm text-slate-500 dark:text-
gray-400">{formatDate(lead.createdAt)}</div>Đ
                        </td>Đ
                        <td className="px-6 py-4">Đ
                          <div className="relative">Đ
                            <textareaĐ
                              value={feedback[lead._id] || ''}Đ
                              onChange={(e) => handleFeedbackChange(lead._id,
e.target.value)}Đ
                              placeholder="Add feedback..."Đ
                              className="text-sm w-full h-20 p-2 border border-
slate-300 dark:border-slate-600 rounded-md focus:outline-none focus:ring-2
focus:ring-blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-
blue-500"Đ
                            />Đ
                            <buttonĐ
                              onClick={() => updateFeedback(lead._id)}Đ
```

```
                              disabled={updating[lead._id] || !feedback[lead._id]}Đ
                              className={`mt-1 px-3 py-1 text-xs rounded ${Đ
                                updating[lead._id] || !feedback[lead._id]Đ
                                  ? 'bg-gray-300 text-gray-500 dark:text-gray-400
dark:text-gray-400 cursor-not-allowed'Đ
                                  : 'bg-blue-600 text-white hover:bg-blue-700'Đ
                              }`}Đ
                            >Đ
                              {updating[lead._id] ? 'Saving...' : 'Save Feedback'}Đ
                            </button>Đ
                          </div>Đ
                        </td>Đ
                        <td className="px-6 py-4 whitespace-nowrap text-right
text-sm font-medium">Đ
                          <buttonĐ
                            onClick={() => viewLeadDetails(lead)}Đ
                            className="text-blue-600 hover:text-blue-900"Đ
                          >Đ
                            View DetailsĐ
                          </button>Đ
                        </td>Đ
                      </tr>Đ
                    ))}Đ
                  </tbody>Đ
                </table>Đ
              </div>Đ
            </div>Đ
          </>Đ
        )}Đ
        Đ
        {/* Lead Details Modal */}Đ
        {selectedLead && (Đ
          <div className="fixed inset-0 z-50 flex items-center justify-center bg-
black bg-opacity-50">Đ
            <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg p-6 w-full
max-w-2xl max-h-[90vh] overflow-y-auto shadow-sm dark:shadow-black/25">Đ
              <div className="flex justify-between items-center mb-4">Đ
                <h3 className="text-xl font-semibold text-slate-900 dark:text-
slate-100">Lead Details</h3>Đ
                <buttonĐ
                  onClick={closeLeadDetails}Đ
                  className="text-gray-400 dark:text-slate-400 hover:text-
slate-500"Đ
                >Đ
                  &times;Đ
                </button>Đ
              </div>Đ
              Đ
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4 mb-4">Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Name</p>Đ
                  <p className="text-base">{selectedLead.name}</p>Đ
                </div>Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Email</p>Đ
                  <p className="text-base">Đ
```

```
                    <button Đ
                      onClick={() => openEmail(selectedLead.email)}Đ
                      className="text-blue-600 hover:text-blue-800 flex items-
center"Đ
                    >Đ
                      <FaEnvelope className="mr-1" /> {selectedLead.email}Đ
                    </button>Đ
                  </p>Đ
                </div>Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Phone</p>Đ
                  <p className="text-base">Đ
                    <button Đ
                      onClick={() => openWhatsApp(selectedLead.phone,
selectedLead.countryCode)}Đ
                      className="text-green-600 hover:text-green-800 flex items-
center"Đ
                    >Đ
                      <FaWhatsapp className="mr-1" /> {selectedLead.countryCode
|| '+91'} {selectedLead.phone}Đ
                    </button>Đ
                  </p>Đ
                </div>Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Course</p>Đ
                  <p className="text-base">{selectedLead.course}</p>Đ
                </div>Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Status</p>Đ
                  <div className="relative">Đ
                    <selectĐ
                      value={selectedLead.status}Đ
                      onChange={(e) => {Đ
                        updateLeadStatus(selectedLead._id, e.target.value);Đ
                        setSelectedLead({...selectedLead, status:
e.target.value});Đ
                      }}Đ
                      className={`px-2 py-1 rounded ${Đ
                        selectedLead.status === 'New' ? 'bg-blue-100 text-
blue-800' :Đ
                        selectedLead.status === 'Contacted' ? 'bg-teal-100 text-
teal-800' :Đ
                        selectedLead.status === 'Qualified' ? 'bg-indigo-100 text-
indigo-800' :Đ
                        selectedLead.status === 'Lost' ? 'bg-red-100 text-
red-800' :Đ
                        selectedLead.status === 'Converted' ? 'bg-green-100 text-
green-800' :Đ
                        selectedLead.status === 'Introduction' ? 'bg-purple-100
text-purple-800' :Đ
                        selectedLead.status === 'Acknowledgement' ? 'bg-
yellow-100 text-yellow-800' :Đ
                        selectedLead.status === 'Question' ? 'bg-indigo-100 text-
indigo-800' :Đ
                        selectedLead.status === 'Future Promise' ? 'bg-cyan-100
text-cyan-800' :Đ
```

```
                        selectedLead.status === 'Payment' ? 'bg-green-100 text-
green-800' :Đ
                        selectedLead.status === 'Analysis' ? 'bg-red-100 text-
red-800' :Đ
                        'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-
gray-200'Đ
                      } border border-transparent focus:outline-none focus:ring-2
focus:ring-blue-500 dark:focus:ring-blue-400 focus:border-blue-500
dark:focus:border-blue-400 appearance-none cursor-pointer pr-8 relative`}Đ
                    >Đ
                      {statusOptions.map(status => (Đ
                        <option key={status} value={status}>{status}</option>Đ
                      ))}Đ
                    </select>Đ
                    <div className="pointer-events-none absolute right-0 top-1/2
transform -translate-y-1/2 flex items-center px-2 text-slate-700 dark:text-
slate-300">Đ
                      <svg className="fill-current h-4 w-4" xmlns="http://
www.w3.org/2000/svg" viewBox="0 0 20 20">Đ
                        <path d="M9.293 12.95l.707.707L15.657 8l-1.414-1.414L10
10.828 5.757 6.586 4.343 8z"/>Đ
                      </svg>Đ
                    </div>Đ
                    {statusUpdating[selectedLead._id] && (Đ
                      <div className="absolute -right-6 top-1/2 transform -
translate-y-1/2">Đ
                        <div className="animate-spin h-4 w-4 border-2 border-
blue-500 rounded-full border-t-transparent"></div>Đ
                      </div>Đ
                    )}Đ
                    {feedback.statusSuccess && feedback.leadId ===
selectedLead._id && (Đ
                      <div className="absolute top-full left-0 mt-1 bg-green-100
text-green-800 text-xs px-2 py-1 rounded whitespace-normal w-48 z-10">Đ
                        {feedback.statusSuccess}Đ
                      </div>Đ
                    )}Đ
                  </div>Đ
                </div>Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Created At</p>Đ
                  <p className="text-base">{formatDate(selectedLead.createdAt)}</
p>Đ
                </div>Đ
                Đ
                {/* Additional client data */}Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Country</p>Đ
                  <p className="text-base">{selectedLead.country || 'N/A'}</p>Đ
                </div>Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Company</p>Đ
                  <p className="text-base">{selectedLead.company || 'N/A'}</p>Đ
                </div>Đ
                <div>Đ
                  <p className="text-sm font-medium text-slate-500 dark:text-
```

```
gray-400">Client</p>Ð
                  <p className="text-base">{selectedLead.client || 'N/A'}</p>Ð
                </div>Ð
                <div>Ð
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Pseudo ID</p>Ð
                  <p className="text-base">{selectedLead.pseudoId || 'N/A'}</p>Ð
                </div>Ð
                <div>Ð
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">Source</p>Ð
                  <p className="text-base">{selectedLead.source || 'N/A'}</p>Ð
                </div>Ð
                <div>Ð
                  <p className="text-sm font-medium text-slate-500 dark:text-
gray-400">LinkedIn</p>Ð
                  <p className="text-base">Ð
                    {selectedLead.sourceLink ? (Ð
                      <a Ð
                        href={selectedLead.sourceLink.startsWith('http') ?
selectedLead.sourceLink : `https://${selectedLead.sourceLink}`} Ð
                        target="_blank" Ð
                        rel="noopener noreferrer"Ð
                        className="text-blue-600 hover:text-blue-800 underline"Ð
                      >Ð
                        {selectedLead.sourceLink}Ð
                      </a>Ð
                    ) : 'N/A'}Ð
                  </p>Ð
                </div>Ð
              </div>Ð
              Ð
              <div className="mb-4">Ð
                <p className="text-sm font-medium text-slate-500 dark:text-
gray-400 mb-1">Remarks</p>Ð
                <p className="text base p-2 bg-gray-50 dark:bg-slate-800
transition-all duration-200 ease-out rounded min-h-[40px]">Ð
                  {selectedLead.remarks || 'No remarks available'}Ð
                </p>Ð
              </div>Ð
              Ð
              <div className="mb-4">Ð
                <p className="text-sm font-medium text-slate-500 dark:text-
gray-400 mb-1">Feedback</p>Ð
                <textareaÐ
                  value={feedback[selectedLead._id] || ''}Ð
                  onChange={(e) => handleFeedbackChange(selectedLead._id,
e.target.value)}Ð
                  placeholder="Add feedback..."Ð
                  className="w-full h-24 p-2 border border-slate-300 dark:border-
slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"Ð
                />Ð
                <buttonÐ
                  onClick={() => updateFeedback(selectedLead._id)}Ð
                  disabled={updating[selectedLead._id] || !
feedback[selectedLead._id]}Ð
                  className={`mt-2 px-4 py-2 rounded ${Ð
                    updating[selectedLead._id] || !feedback[selectedLead._id]Ð
```

```
                    ? 'bg-gray-300 text-gray-500 dark:text-gray-400 dark:text-
gray-400 cursor-not-allowed'Đ
                    : 'bg-blue-600 text-white hover:bg-blue-700'Đ
                }`}Đ
              >Đ
                {updating[selectedLead._id] ? 'Saving...' : 'Save Feedback'}Đ
              </button>Đ
            </div>Đ
            Đ
            <div className="flex justify-end">Đ
              <buttonĐ
                onClick={closeLeadDetails}Đ
                className="px-4 py-2 border border-slate-300 dark:border-
slate-600 rounded-md text-slate-700 dark:text-slate-300 hover:bg-slate-100
dark:hover:bg-slate-700"Đ
              >Đ
                CloseĐ
              </button>Đ
            </div>Đ
          </div>Đ
        </div>Đ
      )}Đ
    </div>Đ
  </Layout>Đ
);Đ
};Đ
Đ
export default SalesPage;Đ
```

src/pages/SalesTrackingPage.jsx

```
import React, { useState, useEffect } from "react";
import { salesAPI, leadsAPI, authAPI } from "../services/api";
import { useAuth } from "../context/AuthContext";
import Layout from "../components/Layout/Layout";
import { FaWhatsapp, FaEnvelope, FaPhone, FaTrash, FaEdit, FaCheck, FaTimes }
from "react-icons/fa";
import axios from "axios";
import { toast } from "react-toastify";
import PhoneInput from 'react-phone-input-2';
import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
import 'react-phone-input-2/lib/style.css';
import api from "../services/api"; // Import the api instance to access baseURL
import LoggingService from "../services/loggingService"; // Add LoggingService
import

const SalesTrackingPage = () => {
  // Custom styles for PhoneInput
  const phoneInputStyle = {
    container: {
      width: '100%',
    },
    inputStyle: {
      width: '100%',
      height: '42px',
      padding: '8px 8px 8px 50px',
      borderRadius: '0.375rem',
```

```
      fontSize: '0.875rem',
      borderColor: '#D1D5DB',
    },
    buttonStyle: {
      borderTopLeftRadius: '0.375rem',
      borderBottomLeftRadius: '0.375rem',
      borderColor: '#D1D5DB',
    },
    dropdownStyle: {
      width: '300px',
    }
  };

  const { user } = useAuth();
  const [sales, setSales] = useState([]);
  const [filteredSales, setFilteredSales] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [editingSale, setEditingSale] = useState(null);
  const [editValues, setEditValues] = useState({});
  const [showAddModal, setShowAddModal] = useState(false);
  const [newSale, setNewSale] = useState({
    leadId: "",
    amount: 0,
    token: 0,
    pending: 0,
    product: "",
    status: "Pending",
    isReference: false,
    customerName: "",
    email: "",
    contactNumber: "",
    country: "",
    leadPerson: "",
    countryCode: "+1",
    loginId: "",
    password: "",
    leadBy: "",
    saleDate: new Date(),
    currency: "USD" // Default currency
  });
  const [availableLeads, setAvailableLeads] = useState([]);
  const [leadOptions, setLeadOptions] = useState([]);
  const [loadingLeads, setLoadingLeads] = useState(false);
  const [leadPersonOptions, setLeadPersonOptions] = useState([]);
  const [loadingLeadPersons, setLoadingLeadPersons] = useState(false);

  // Date filtering state
  const [filterMonth, setFilterMonth] = useState(new Date().getMonth() + 1); //
Current month (1-12)
  const [filterYear, setFilterYear] = useState(new Date().getFullYear()); //
Current year
  const [showCurrentMonth, setShowCurrentMonth] = useState(false); // Changed to
false - don't filter by default
  const [showAllSales, setShowAllSales] = useState(true); // Changed to true -
show all sales by default

  // Advanced filter state
  const [filters, setFilters] = useState({
```

```
  search: "",
  status: "",
  country: "",
  course: "",
  salesPerson: "",
  leadPerson: "",
  dateFrom: "",
  dateTo: "",
  amountFrom: "",
  amountTo: "",
  currency: ""
});

// Options for filters
const [filterOptions, setFilterOptions] = useState({
  countries: [],
  courses: [],
  salesPersons: [],
  leadPersons: [],
  currencies: []
});

// Currency options
const currencyOptions = [
  { value: "USD", label: "USD ($)", symbol: "$" },
  { value: "EUR", label: "EUR (€)", symbol: "€" },
  { value: "GBP", label: "GBP (£)", symbol: "£" },
  { value: "INR", label: "INR ( ¹)", symbol: " ¹" },
  { value: "CAD", label: "CAD ($)", symbol: "$" },
  { value: "AUD", label: "AUD ($)", symbol: "$" },
  { value: "JPY", label: "JPY (¥)", symbol: "¥" },
  { value: "CNY", label: "CNY (¥)", symbol: "¥" }
];

// Generate month options
const months = [
  { value: 1, label: "January" },
  { value: 2, label: "February" },
  { value: 3, label: "March" },
  { value: 4, label: "April" },
  { value: 5, label: "May" },
  { value: 6, label: "June" },
  { value: 7, label: "July" },
  { value: 8, label: "August" },
  { value: 9, label: "September" },
  { value: 10, label: "October" },
  { value: 11, label: "November" },
  { value: 12, label: "December" }
];

// Generate year options (5 years back from current year)
const currentYear = new Date().getFullYear();
const years = Array.from({ length: 6 }, (_, i) => currentYear - i);

// Sale status options - must match the enum values in the server's Sale model
const statusOptions = [
  "Pending",
  "Completed",
  // Only show Cancelled in dropdown for admin users
```

```
    ...(user?.role === 'Admin' ? ["Cancelled"] : [])
  ];

  // Add a function to get available status options based on user role and
current status
  const getAvailableStatusOptions = (currentStatus) => {
    // If status is already Cancelled, show it in options regardless of role
    if (currentStatus === 'Cancelled') {
      return ["Pending", "Completed", "Cancelled"];
    }
    return statusOptions;
  };

  // Add new state for delete confirmation
  const [deletingSale, setDeletingSale] = useState(null);
  const [confirmDelete, setConfirmDelete] = useState(false);

  // State for collapsible advanced filters
  const [showAdvancedFilters, setShowAdvancedFilters] = useState(false);

  // Fetch sales data
  useEffect(() => {
    fetchSales();
    fetchUserOptions();
  }, [user]);

  // Apply filters when sales or filters change
  useEffect(() => {
    console.log(`Sales data changed: ${sales.length} sales available`);
    console.log('Current filter states:', { showAllSales, showCurrentMonth,
filterMonth, filterYear });

    if (sales.length > 0) {
      applyAllFilters();
      extractFilterOptions();
    } else {
      // If no sales, clear filtered sales
      setFilteredSales([]);
      console.log('No sales data available, clearing filtered sales');
    }
  }, [sales, filters, filterMonth, filterYear, showCurrentMonth, showAllSales]);

  // Auto-expand advanced filters if any are active
  useEffect(() => {
    const hasActiveFilters = Object.values(filters).some(filter => filter !== "");
    if (hasActiveFilters && !showAdvancedFilters) {
      setShowAdvancedFilters(true);
    }
  }, [filters, showAdvancedFilters]);

  // Fetch lead persons when in reference sale mode
  useEffect(() => {
    if (newSale.isReference && leadPersonOptions.length === 0) {
      fetchLeadPersons();
    }
  }, [newSale.isReference]);

  // Function to apply all filters
  const applyAllFilters = () => {
```

```
    // Filter out null/undefined sales first
    let filtered = sales.filter(sale => sale && sale._id);

    // Apply date filters first
    if (showAllSales) {
      // Show all sales regardless of date when showAllSales is true
      console.log('Showing all sales - no date filtering applied');
    } else if (showCurrentMonth) {
      // Show current month data
      const currentMonth = new Date().getMonth() + 1; // 1-12
      const currentYear = new Date().getFullYear();

      filtered = filtered.filter(sale => {
        if (!sale || !sale.date && !sale.createdAt) return false;
        const saleDate = new Date(sale.date || sale.createdAt);
        const saleMonth = saleDate.getMonth() + 1;
        const saleYear = saleDate.getFullYear();
        return saleMonth === currentMonth && saleYear === currentYear;
      });
      console.log(`Filtered to current month (${currentMonth}/${currentYear}):
${filtered.length} sales`);
    } else {
      // Show selected month/year data
      filtered = filtered.filter(sale => {
        if (!sale || !sale.date && !sale.createdAt) return false;
        const saleDate = new Date(sale.date || sale.createdAt);
        const saleMonth = saleDate.getMonth() + 1;
        const saleYear = saleDate.getFullYear();
        return saleMonth === filterMonth && saleYear === filterYear;
      });
      console.log(`Filtered to selected month (${filterMonth}/${filterYear}):
${filtered.length} sales`);
    }

    // Apply advanced filters

    // Text search (customer name, email, product, login ID)
    if (filters.search) {
      const searchTerm = filters.search.toLowerCase();
      filtered = filtered.filter(sale =>
        (sale.customerName &&
sale.customerName.toLowerCase().includes(searchTerm)) ||
        (sale.email && sale.email.toLowerCase().includes(searchTerm)) ||
        (sale.product && sale.product.toLowerCase().includes(searchTerm)) ||
        (sale.course && sale.course.toLowerCase().includes(searchTerm)) ||
        (sale.loginId && sale.loginId.toLowerCase().includes(searchTerm)) ||
        (sale.leadBy && sale.leadBy.toLowerCase().includes(searchTerm))
      );
    }

    // Status filter
    if (filters.status) {
      filtered = filtered.filter(sale => sale.status === filters.status);
    }

    // Country filter
    if (filters.country) {
      filtered = filtered.filter(sale => sale.country === filters.country);
    }
```

```
// Course/Product filter
if (filters.course) {
  filtered = filtered.filter(sale =>
    (sale.course && sale.course === filters.course) ||
    (sale.product && sale.product === filters.course)
  );
}

// Sales Person filter
if (filters.salesPerson) {
  filtered = filtered.filter(sale => {
    if (!sale || !sale.salesPerson) return false;
    const salesPersonId = typeof sale.salesPerson === 'object' ?
      sale.salesPerson._id : sale.salesPerson;
    return salesPersonId === filters.salesPerson;
  });
}

// Lead Person filter
if (filters.leadPerson) {
  filtered = filtered.filter(sale => {
    if (!sale || !sale.leadPerson) return false;
    const leadPersonId = typeof sale.leadPerson === 'object' ?
      sale.leadPerson._id : sale.leadPerson;
    return leadPersonId === filters.leadPerson;
  });
}

// Date range filter
if (filters.dateFrom) {
  const fromDate = new Date(filters.dateFrom);
  filtered = filtered.filter(sale => {
    if (!sale || !sale.date && !sale.createdAt) return false;
    const saleDate = new Date(sale.date || sale.createdAt);
    return saleDate >= fromDate;
  });
}

if (filters.dateTo) {
  const toDate = new Date(filters.dateTo);
  toDate.setHours(23, 59, 59, 999); // End of the day
  filtered = filtered.filter(sale => {
    if (!sale || !sale.date && !sale.createdAt) return false;
    const saleDate = new Date(sale.date || sale.createdAt);
    return saleDate <= toDate;
  });
}

// Amount range filter
if (filters.amountFrom) {
  const minAmount = parseFloat(filters.amountFrom);
  filtered = filtered.filter(sale => {
    if (!sale) return false;
    const amount = parseFloat(sale.amount || sale.totalCost || 0);
    return amount >= minAmount;
  });
}
```

```
    if (filters.amountTo) {
      const maxAmount = parseFloat(filters.amountTo);
      filtered = filtered.filter(sale => {
        if (!sale) return false;
        const amount = parseFloat(sale.amount || sale.totalCost || 0);
        return amount <= maxAmount;
      });
    }

    // Currency filter
    if (filters.currency) {
      filtered = filtered.filter(sale =>
        (sale && sale.currency === filters.currency) ||
        (sale && sale.totalCostCurrency === filters.currency)
      );
    }

    console.log(`Final filtered sales count: ${filtered.length} out of
${sales.length} total sales`);
    setFilteredSales(filtered);
  };

  // Extract filter options from sales data
  const extractFilterOptions = () => {
    const countries = [...new Set(sales.map(sale =>
sale.country).filter(Boolean))];
    const courses = [...new Set(sales.map(sale => sale.course ||
sale.product).filter(Boolean))];
    const currencies = [...new Set(sales.map(sale => sale.currency ||
sale.totalCostCurrency).filter(Boolean))];

    setFilterOptions(prev => ({
      ...prev,
      countries,
      courses,
      currencies
    }));
  };

  // Fetch user options for filters
  const fetchUserOptions = async () => {
    try {
      const salesPersonsResponse = await authAPI.getUsers("Sales Person");
      const leadPersonsResponse = await authAPI.getUsers("Lead Person");

      setFilterOptions(prev => ({
        ...prev,
        salesPersons: salesPersonsResponse.data.data || [],
        leadPersons: leadPersonsResponse.data.data || []
      }));
    } catch (err) {
      console.error("Error fetching user options:", err);
    }
  };

  // Handle filter changes
  const handleFilterChange = (e) => {
    const { name, value } = e.target;
    setFilters(prev => ({ ...prev, [name]: value }));
```

```
    // Auto-expand filters when a filter is applied
    if (value && !showAdvancedFilters) {
      setShowAdvancedFilters(true);
    }
  };

  // Reset all filters
  const resetFilters = () => {
    setFilters({
      search: "",
      status: "",
      country: "",
      course: "",
      salesPerson: "",
      leadPerson: "",
      dateFrom: "",
      dateTo: "",
      amountFrom: "",
      amountTo: "",
      currency: ""
    });
    setShowCurrentMonth(false);
    setShowAllSales(true);
  };

  // Fetch sales data
  const fetchSales = async () => {
    try {
      setLoading(true);
      setError(null);

      console.log('Fetching sales data...');

      const response = await salesAPI.getAllForced();

      if (response.data && response.data.success &&
Array.isArray(response.data.data)) {
        console.log(`Successfully loaded ${response.data.data.length} sales`);

        // Process and set the sales data
        const processedSales = response.data.data
          .filter(sale => sale && sale._id) // Filter out invalid entries
          .map(sale => ({
            ...sale,
            amount: parseFloat(sale.totalCost || sale.amount || 0),
            token: parseFloat(sale.tokenAmount || sale.token || 0),
            pending: sale.status === 'Completed' ? 0 :
              parseFloat(sale.totalCost || sale.amount || 0) -
parseFloat(sale.tokenAmount || sale.token || 0),
            product: sale.course || sale.product || 'Unknown',
            status: sale.status || 'Pending',
            currency: sale.currency || 'USD',
            date: sale.date || sale.createdAt,
            remarks: sale.remarks || '' // Ensure remarks are included and
defaulted to empty string
          }));

        setSales(processedSales);
```

```
        setFilteredSales(processedSales); // Initialize filtered sales

        if (processedSales.length === 0) {
          toast.info('No sales records found.');
        }
      } else {
        throw new Error('Invalid data format received');
      }
    } catch (err) {
      console.error('Error fetching sales:', err);
      setError('Failed to fetch sales data');
      toast.error('Failed to load sales data');
    } finally {
      setLoading(false);
    }
  };

  // Fetch available leads for selection in add sale form
  const fetchAvailableLeads = async () => {
    try {
      setLoadingLeads(true);
      let leadsData = [];

      // If sales person, fetch assigned leads
      if (user.role === 'Sales Person') {
        const response = await leadsAPI.getAssigned();
        if (response.data.success) {
          leadsData = response.data.data.filter(lead => lead.status !==
'Converted');
        }
      } else {
        // For admin and manager, fetch all leads
        const response = await leadsAPI.getAll();
        if (response.data.success) {
          leadsData = response.data.data.filter(lead => lead.status !==
'Converted');
        }
      }

      setAvailableLeads(leadsData);

      // Create options for select dropdown
      const options = leadsData.map(lead => ({
        value: lead._id,
        label: `${lead.name} - ${lead.course} (${lead.status})`,
        data: lead
      }));

      setLeadOptions(options);
    } catch (err) {
      console.error("Error fetching leads:", err);
      setError("Failed to load leads. Please try again.");
    } finally {
      setLoadingLeads(false);
    }
  };

  // Fetch available lead persons (for lead selection in sales)
  const fetchLeadPersons = async () => {
```

```
  try {
    setLoadingLeadPersons(true);
    const response = await authAPI.getUsers('Lead Person');
    if (response.data.success) {
      const options = response.data.data.map(user => ({
        value: user._id,
        label: user.fullName || user.email,
      }));
      setLeadPersonOptions(options);
    }
  } catch (err) {
    console.error("Error fetching lead persons:", err);
  } finally {
    setLoadingLeadPersons(false);
  }
};

// Handle opening add sale modal
const handleAddSaleClick = () => {
  fetchAvailableLeads();
  // Always fetch lead persons when opening the add sale modal
  fetchLeadPersons();
  setShowAddModal(true);
  // Reset new sale form
  setNewSale({
    leadId: "",
    amount: 0,
    token: 0,
    pending: 0,
    product: "",
    status: "Pending",
    isReference: false,
    customerName: "",
    email: "",
    contactNumber: "",
    country: "",
    leadPerson: "",
    countryCode: "+1",
    loginId: "",
    password: "",
    leadBy: "",
    saleDate: new Date(),
    currency: "USD" // Default currency
  });
};

// Handle lead selection in add form
const handleLeadSelect = (e) => {
  const selectedLeadId = e.target.value;
  const selectedLead = availableLeads.find(lead => lead._id === selectedLeadId);

  if (selectedLead) {
    setNewSale(prev => ({
      ...prev,
      leadId: selectedLeadId,
      // Get course/product from lead
      product: selectedLead.course || selectedLead.COURSE || '',
      // Store other lead data we'll need for the API
      _selectedLead: selectedLead
```

```
      }));
    }
  };

  // Handle new sale form input changes
  const handleNewSaleChange = (field, value) => {
    setNewSale(prev => {
      const updated = { ...prev, [field]: value };

      // If amount or token changes, recalculate pending
      if (field === 'amount' || field === 'token') {
        const amount = field === 'amount' ? parseFloat(value) :
parseFloat(prev.amount);
        const token = field === 'token' ? parseFloat(value) :
parseFloat(prev.token);
        updated.pending = amount - token;
      }

      // If status is Completed, set pending to 0
      if (field === 'status' && value === 'Completed') {
        updated.pending = 0;
      }

      return updated;
    });
  };

  // Toggle between reference and lead-based sale
  const handleReferenceToggle = (isReference) => {
    // If switching to reference sale, fetch lead persons for selection
    setNewSale({
      ...newSale,
      isReference,
      leadId: isReference ? "" : newSale.leadId,
      customerName: isReference ? newSale.customerName : "",
      contactNumber: isReference ? newSale.contactNumber : "",
      email: isReference ? newSale.email : "",
      country: isReference ? newSale.country : "",
      countryCode: isReference ? newSale.countryCode : "+1",
      // Don't reset leadPerson - we want to preserve this selection
    });

    if (isReference && leadPersonOptions.length === 0) {
      fetchLeadPersons();
    }
  };

  // Submit new sale - simplified without currency conversion complexity
  const handleSubmitNewSale = async (e) => {
    e.preventDefault();

    try {
      // Different validation based on whether it's a reference sale
      if (!newSale.isReference) {
        // Non-reference sale validation
        if (!newSale.leadId) {
          setError("Please select a lead");
          return;
        }
```

```
        if (!newSale.product) {
          setError("Please enter a product/course name");
          return;
        }

        // Require a lead person to be selected for regular sales too
        if (!newSale.leadPerson) {
          setError("Please select a lead person who will see this sale");
          return;
        }
      } else {
        // Reference sale validation
        if (!newSale.customerName) {
          setError("Please enter customer name");
          return;
        }

        if (!newSale.contactNumber) {
          setError("Please enter contact number");
          return;
        }

        if (!newSale.product) {
          setError("Please enter a product/course name");
          return;
        }

        if (!newSale.country) {
          setError("Please enter country");
          return;
        }
      }

      // Get fresh token
      const token = localStorage.getItem('token');

      if (!token) {
        setError("Authentication required. Please log in again.");
        return;
      }

      let saleData;

      if (!newSale.isReference) {
        // Process normal sale from lead
        const selectedLead = newSale._selectedLead || availableLeads.find(lead =>
lead._id === newSale.leadId);

        if (!selectedLead) {
          setError("Lead information not found. Please select a lead again.");
          return;
        }

        // Extract lead details for the sale using the required schema fields
        saleData = {
          // Customer details from lead
          customerName: selectedLead.name || selectedLead.NAME ||
selectedLead.customerName || 'Unknown',
```

```
          country: selectedLead.country || selectedLead.COUNTRY || 'Unknown',
          course: newSale.product, // Use the product field as course
          countryCode: selectedLead.countryCode || selectedLead.CODE || '+1',
          contactNumber: selectedLead.phone || selectedLead.contactNumber ||
selectedLead.NUMBER || selectedLead.MOBILE || '0000000000',
          email: selectedLead.email || selectedLead['E-MAIL'] ||
selectedLead.EMAIL || '',

          // ID references
          salesPerson: user._id, // Current user is the sales person
          // Use the explicitly selected lead person
          leadPerson: newSale.leadPerson,

          // Optional fields - new
          loginId: newSale.loginId || '',
          password: newSale.password || '',
          leadBy: newSale.leadBy || '',

          // Source info
          source: selectedLead.source || selectedLead.SOURSE || '',
          clientRemark: selectedLead.client || selectedLead['CLIENT REMARK'] ||
'',

          // Financial info - with currency
          totalCost: parseFloat(newSale.amount) || 0,
          tokenAmount: parseFloat(newSale.token) || 0,
          currency: newSale.currency || 'USD',

          // Status info
          pending: newSale.status === 'Completed' ? false :
parseFloat(newSale.pending) > 0, // Set to false if status is completed
          status: newSale.status || 'Pending',

          // Creation metadata
          createdBy: user._id,
          date: newSale.saleDate || new Date(), // Use selected date or current
date

          // Flag to ensure this shows in lead person's dashboard
          isLeadPersonSale: true
        };
      } else {
        // Process reference sale with manually entered data
        saleData = {
          // Customer details from form
          customerName: newSale.customerName,
          country: newSale.country,
          course: newSale.product,
          countryCode: newSale.countryCode || '+1',
          contactNumber: newSale.contactNumber,
          email: newSale.email || '',

          // ID references
          salesPerson: user._id, // Current user is the sales person
          leadPerson: newSale.leadPerson, // Use selected lead person

          // Optional fields - new
          loginId: newSale.loginId || '',
          password: newSale.password || '',
```

```
          leadBy: newSale.leadBy || '',

          // Source info
          source: 'Reference', // Mark as reference
          isReference: true,

          // Financial info - with currency
          totalCost: parseFloat(newSale.amount) || 0,
          tokenAmount: parseFloat(newSale.token) || 0,
          currency: newSale.currency || 'USD',

          // Status info
          pending: newSale.status === 'Completed' ? false :
parseFloat(newSale.pending) > 0, // Set to false if status is completed
          status: newSale.status || 'Pending',

          // Creation metadata
          createdBy: user._id,
          date: newSale.saleDate || new Date(), // Use selected date or current
date

          // Flag to ensure this shows in lead person's dashboard
          isLeadPersonSale: true
        };
      }

      // Use the API service - explicitly set isLeadPersonSale flag to true for
both types
      const response = await salesAPI.create({ ...saleData, isLeadPersonSale:
true });

      if (response.data && response.data.success) {
        // Add new sale to the list
        setSales(prev => [response.data.data, ...prev]);

        // Log the sale creation
        try {
          await LoggingService.logSaleCreate(response.data.data);
        } catch (logError) {
          console.error('Error logging sale creation:', logError);
        }

        // Close modal and reset form
        setShowAddModal(false);
        setNewSale({
          leadId: "",
          amount: 0,
          token: 0,
          pending: 0,
          product: "",
          status: "Pending",
          isReference: false,
          customerName: "",
          email: "",
          contactNumber: "",
          country: "",
          leadPerson: "",
          countryCode: "+1",
          loginId: "",
```

```
          password: "",
          leadBy: "",
          saleDate: new Date(),
          currency: "USD" // Default currency
        });

        // Show success message
        toast.success("Sale added successfully!");

        // Refresh data to ensure we have the latest
        refreshData();
      } else {
        setError(response.data?.message || "Failed to add sale");
      }
    } catch (err) {
      // Detailed error handling
      if (err.response) {
        if (err.response.status === 400) {
          // Better handling for validation errors
          const errorMsg = err.response.data?.message || "Invalid data";
          if (errorMsg.includes('enum value for path `status`')) {
            toast.error(`Invalid status value. Allowed values are:
${statusOptions.join(', ')}`);
          } else {
            toast.error(`Bad request: ${errorMsg}`);
          }
        } else if (err.response.status === 401) {
          setError("Authentication failed. Please log in again.");
        } else {
          setError(`Failed to add sale: ${err.response.data?.message ||
err.message}`);
        }
      } else {
        setError("Network error while adding sale");
      }
    }
  };

  // Improved helper function to safely extract IDs from possibly nested or
string IDs
  const extractId = (obj, field) => {
    if (!obj) return null;

    // If field is direct property and is a string already, return it
    if (typeof obj[field] === 'string') return obj[field];

    // If field is an object with _id, return that
    if (obj[field] && obj[field]._id) {
      // MongoDB ObjectId is stored as an object that can be converted to string
      if (typeof obj[field]._id === 'object' && obj[field]._id.toString) {
        return obj[field]._id.toString();
      }
      return obj[field]._id;
    }

    // If field itself is an object with an ID property, return that
    if (obj[field] && typeof obj[field].id === 'string') return obj[field].id;

    // If the field is an ObjectId that can be converted to string
```

```
    if (obj[field] && typeof obj[field] === 'object' && obj[field].toString) {
      return obj[field].toString();
    }

    // Return null if nothing found
    return null;
  };

  // Initialize edit values
  const handleEdit = (sale) => {
    if (!sale || !sale._id) {
      console.error('Invalid sale object:', sale);
      toast.error('Cannot edit this sale - invalid data');
      return;
    }

    setEditingSale(sale._id);
    setEditValues({
      amount: parseFloat(Number(sale.amount || sale.totalCost || 0).toFixed(2)),
      token: parseFloat(Number(sale.token || sale.tokenAmount || 0).toFixed(2)),
      pending: parseFloat(Number(sale.pending || ((sale.amount || sale.totalCost
|| 0) - (sale.token || sale.tokenAmount || 0))).toFixed(2)),
      status: sale.status || 'Pending',
      product: sale.product || sale.course || '',
      saleDate: sale.date || new Date(),
      loginId: sale.loginId || '',
      password: sale.password || '',
      leadBy: sale.leadBy || '',
      currency: sale.currency || 'USD',
      remarks: sale.remarks || '' // Initialize remarks from existing sale
    });
  };

  // Handle saving edits
  const handleSave = async (saleId) => {
    try {
      // Find the original sale for data we don't want to change
      const originalSale = sales.find(sale => sale && sale._id === saleId);
      if (!originalSale) {
        setError("Sale not found");
        return;
      }

      // Create update data matching the schema
      const updateData = {
        // Keep original customer info
        customerName: originalSale.customerName,
        country: originalSale.country,
        course: editValues.product || originalSale.course, // Update course/
product
        countryCode: originalSale.countryCode,
        contactNumber: originalSale.contactNumber,
        email: originalSale.email,

        // Updated fields
        salesPerson: editValues.salesPerson || originalSale.salesPerson?._id ||
originalSale.salesPerson,
        leadPerson: originalSale.leadPerson?._id || originalSale.leadPerson,
        source: originalSale.source,
```

```
      clientRemark: originalSale.clientRemark,
      totalCost: parseFloat(editValues.amount) || originalSale.totalCost,
      tokenAmount: parseFloat(editValues.token) || originalSale.tokenAmount,
      currency: editValues.currency || originalSale.currency,
      status: editValues.status || originalSale.status,
      remarks: editValues.remarks || originalSale.remarks || '',
      isReference: originalSale.isReference || false,
      date: editValues.saleDate || originalSale.date,
      loginId: editValues.loginId || originalSale.loginId || '',
      password: editValues.password || originalSale.password || '',
      leadBy: editValues.leadBy || originalSale.leadBy || '',

      // Update metadata
      updatedBy: user._id,
      updatedAt: new Date()
    };

    // Use the API service
    const response = await salesAPI.update(saleId, updateData);

    if (response.data && response.data.success) {
      // Log the sale update
      try {
        await LoggingService.logSaleUpdate(saleId, updateData);
      } catch (logError) {
        console.error('Error logging sale update:', logError);
      }

      // Immediately update the local state with the new data
      const updatedSale = response.data.data;

      setSales(prevSales =>
        prevSales.map(sale =>
          sale && sale._id === saleId ? updatedSale : sale
        )
      );

      setFilteredSales(prevFiltered =>
        prevFiltered.map(sale =>
          sale && sale._id === saleId ? updatedSale : sale
        )
      );

      // Clear edit state
      setEditingSale(null);
      setEditValues({});

      toast.success('Sale updated successfully');
    } else {
      toast.error(response.data?.message || 'Failed to update sale');
    }
  } catch (err) {
    console.error('Error updating sale:', err);
    toast.error(err.response?.data?.message || 'Error updating sale');
  }
};

// Fixed handleInputChange to properly handle numeric values
const handleInputChange = (field, value) => {
```

```
    setEditValues(prev => {
      const updated = { ...prev, [field]: value };

      // If we're updating amount or token, recalculate the pending amount with
proper rounding
      if (field === 'amount' || field === 'token') {
        const amount = parseFloat(Number(field === 'amount' ? value :
prev.amount).toFixed(2)) || 0;
        const tokenAmount = parseFloat(Number(field === 'token' ? value :
prev.token).toFixed(2)) || 0;
        updated.pending = parseFloat((amount - tokenAmount).toFixed(2));
      }

      // If status is set to Completed, set pending to 0
      if (field === 'status' && value === 'Completed') {
        updated.pending = 0;
      }

      return updated;
    });
  };

  // Fixed function to determine if user can edit a sale
  const canEditSale = (sale) => {
    if (!sale || !user || !user._id || !user.role) return false;

    // Get sales person ID handling both object and string formats
    const salesPersonId = extractId(sale, 'salesPerson');
    const userId = user._id;

    // Sales person can edit their own sales
    if (user.role === 'Sales Person') {
      const isOwnSale = salesPersonId && userId && salesPersonId.toString() ===
userId.toString();
      return isOwnSale;
    }

    // Lead person can edit sales they are the lead for
    if (user.role === 'Lead Person') {
      const leadPersonId = extractId(sale, 'leadPerson');
      return leadPersonId && userId && leadPersonId.toString() ===
userId.toString();
    }

    // Admins and managers can edit any sale
    return ['Admin', 'Manager'].includes(user.role);
  };

  // Fixed function to determine if user can delete a sale
  const canDeleteSale = (sale) => {
    if (!sale || !user || !user._id || !user.role) return false;

    // Get sales person ID handling both object and string formats
    const salesPersonId = extractId(sale, 'salesPerson');
    const userId = user._id;

    // Sales person can delete their own sales
    if (user.role === 'Sales Person') {
      const isOwnSale = salesPersonId && userId && salesPersonId.toString() ===
userId.toString();
```

```
      return isOwnSale;
    }

    // Admins and managers can delete any sale
    return ['Admin', 'Manager'].includes(user.role);
  };

  // Handle delete sale - improved to fix 403 errors
  const handleDeleteSale = async (saleId) => {
    try {
      setDeletingSale(saleId);

      // Find the original sale to include necessary data
      const saleToDelete = sales.find(sale => sale._id === saleId);
      if (!saleToDelete) {
        toast.error("Sale not found");
        return;
      }

      // Use the API service instead of direct Axios call
      const response = await salesAPI.delete(saleId);

      if (response.data && response.data.success) {
        // Remove the deleted sale from state
        setSales(prevSales => prevSales.filter(sale => sale._id !== saleId));
        toast.success("Sale deleted successfully!");

        // Refresh sales data to ensure we have the latest
        fetchSales();
      } else {
        toast.error(response.data?.message || "Failed to delete sale");
      }
    } catch (err) {
      // Detailed error handling
      if (err.response) {
        if (err.response.status === 403) {
          toast.error("You don't have permission to delete this sale. Only the
creator or an admin can delete it.");
        } else if (err.response.status === 401) {
          toast.error("Your session has expired. Please log in again.");
        } else {
          toast.error(err.response.data?.message || "Server error while deleting
sale");
        }
      } else {
        toast.error("Network error while deleting sale");
      }
    } finally {
      setDeletingSale(null);
      setConfirmDelete(false);
    }
  };

  // Format date for display
  const formatDate = (dateString) => {
    const date = new Date(dateString);
    return date.toLocaleDateString();
  };
```

```
// Handle month change
const handleMonthChange = (e) => {
  setFilterMonth(parseInt(e.target.value));
  setShowCurrentMonth(false);
  setShowAllSales(false); // Disable show all when selecting specific month
};

// Handle year change
const handleYearChange = (e) => {
  setFilterYear(parseInt(e.target.value));
  setShowCurrentMonth(false);
  setShowAllSales(false); // Disable show all when selecting specific year
};

// Handle reset to current month
const handleResetToCurrentMonth = () => {
  const today = new Date();
  setFilterMonth(today.getMonth() + 1);
  setFilterYear(today.getFullYear());
  setShowCurrentMonth(true);
  setShowAllSales(false); // Disable show all when resetting to current month
};

// Handle show all sales toggle for Managers and Admins
const handleShowAllSales = () => {
  setShowAllSales(!showAllSales);
  if (!showAllSales) {
    // When enabling show all, disable other filters
    setShowCurrentMonth(false);
  }
};

const handleRemarksChange = async (saleId, newRemarks) => {
  try {
    const targetSale = sales.find(sale => sale._id === saleId);
    if (!targetSale) return;

    if (!canEditSale(targetSale)) {
      toast.error("You don't have permission to update this sale");
      return;
    }

    setEditingSale(saleId);

    // Only send necessary fields for update
    const updateData = {
      remarks: newRemarks,
      updatedBy: user.id
    };

    // Make API call to update remarks using salesAPI
    const response = await salesAPI.update(saleId, updateData);

    if (response.data.success) {
      // Update local state with new remarks
      setSales(prevSales =>
        prevSales.map(sale =>
          sale._id === saleId
            ? { ...sale, remarks: newRemarks }
```

```
            : sale
        )
      );
      toast.success('Remarks updated successfully');
    } else {
      toast.error('Failed to update remarks');
    }
  } catch (error) {
    console.error('Error updating remarks:', error);
    toast.error('Error updating remarks');
  } finally {
    setEditingSale(null);
  }
};

const handleStatusChange = async (saleId, newStatus) => {
  try {
    // Only proceed if user can edit this sale
    const targetSale = sales.find(sale => sale._id === saleId);
    if (!targetSale) return;

    if (!canEditSale(targetSale)) {
      toast.error("You don't have permission to update this sale");
      return;
    }

    // Check if user has permission to set status to Cancelled
    if (newStatus === 'Cancelled' && user?.role !== 'Admin') {
      toast.error("Only administrators can cancel sales");
      return;
    }

    setEditingSale(saleId);

    // Only send necessary fields for update
    const updateData = {
      status: newStatus,
      remarks: targetSale.remarks || '', // Preserve existing remarks
      updatedBy: user.id
    };

    // Make API call to update status using salesAPI
    const response = await salesAPI.update(saleId, updateData);

    if (response.data.success) {
      // Update local state
      setSales(prevSales =>
        prevSales.map(sale =>
          sale._id === saleId
            ? { ...sale, status: newStatus }
            : sale
        )
      );
      toast.success('Status updated successfully');
    } else {
      toast.error('Failed to update status');
    }
  } catch (error) {
    console.error('Error updating status:', error);
```

```
      toast.error('Error updating status');
    } finally {
      setEditingSale(null);
    }
  };

  // Open WhatsApp with the phone number
  const openWhatsApp = (phone, countryCode) => {
    if (!phone) return;

    try {
      // Remove non-digit characters
      const cleanPhone = phone.toString().replace(/\D/g, '');

      // Format the phone number for WhatsApp without the + symbol
      // Make sure the country code doesn't have a + in the beginning
      const dialCode = countryCode ? countryCode.toString().replace(/^\+/, '') :
'1';

      // Build the WhatsApp URL - format is: https://wa.me/[countrycode][number]
      // WhatsApp API doesn't use the + symbol
      const whatsappUrl = `https://wa.me/${dialCode}${cleanPhone}`;

      // Open WhatsApp in a new tab
      window.open(whatsappUrl, '_blank', 'noopener,noreferrer');
    } catch (error) {
      toast.error('Could not open WhatsApp. Please try again.');
    }
  };

  // Open email client with the email address
  const openEmail = (email) => {
    if (!email) return;
    window.location.href = `mailto:${email}`;
  };

  // Add function to refresh data after operations
  const refreshData = async () => {
    try {
      // Clear the state first to ensure update
      setSales([]);
      setFilteredSales([]);

      // Set loading state
      setLoading(true);

      // Use the API service with cache busting parameter
      const timestamp = new Date().getTime();
      const response = await salesAPI.getAll();

                    if (response.data && response.data.success) {
        // Initialize sales with additional fields we want to track
        // Add extra safety checks for data structure
        if (!Array.isArray(response.data.data)) {
          console.error('RefreshData: Sales data is not an array:',
response.data.data);
          toast.error("Invalid data format received from server");
          return;
        }
```

```
        // Filter out null/undefined values and items without _id before
processing
        const validSales = response.data.data.filter(sale => {
          // More robust filtering
          return sale &&
                 typeof sale === 'object' &&
                 sale._id !== null &&
                 sale._id !== undefined &&
                 sale._id !== '';
        });

        console.log(`RefreshData: Original array length:
${response.data.data.length}, Valid sales: ${validSales.length}`);

        if (response.data.data.length !== validSales.length) {
          console.log(`RefreshData: Filtered out ${response.data.data.length -
validSales.length} null/invalid sales from ${response.data.data.length} total`);
        }

        const processedSales = validSales.map(sale => {
          // Final safety check in case somehow a null value got through
          if (!sale || !sale._id) {
            console.error('Null sale in refreshData processing:', sale);
            return null;
          }

          // Create a properly formatted sale object with consistent fields
          const formattedSale = {
            ...sale,
            _id: sale._id,
            // Convert ObjectId to string if needed
            salesPerson: typeof sale.salesPerson === 'object' ?
                         (sale.salesPerson._id || sale.salesPerson) :
                         sale.salesPerson,
            // Ensure all financial fields exist
            amount: parseFloat(sale.totalCost || 0),
            token: parseFloat(sale.tokenAmount || 0),
            pending: sale.status === 'Completed' ? 0 : parseFloat(sale.totalCost
|| 0) - parseFloat(sale.tokenAmount || 0),
            // Ensure we have product info
            product: sale.course || sale.product || 'Unknown',
            // Ensure we have a status
            status: sale.status || 'Pending',
            // Ensure login credentials are preserved
            loginId: sale.loginId || '',
            password: sale.password || '',
            leadBy: sale.leadBy || '',
            // Ensure currency is preserved
            currency: sale.currency || 'USD',
            // Ensure date is properly captured
            date: sale.date || sale.createdAt,
            // Ensure remarks are preserved
            remarks: sale.remarks || ''
          };

          return formattedSale;
        });
```

```
        // Filter out any null values that might have been returned by the map
function
        const finalSales = processedSales.filter(sale => sale !== null);

        // Update the sales state
        setSales(finalSales);
      } else {
        toast.error("Failed to refresh sales data. Please try again.");
      }
    } catch (err) {
      toast.error("Failed to refresh data. Please reload the page.");
    } finally {
      setLoading(false);

      // For new sales, refresh leads too
      if (showAddModal) {
        fetchAvailableLeads();
      }
    }
  };



  // Safely get a value from a possibly undefined property
  const safeGet = (obj, path, defaultValue = 'N/A') => {
    if (!obj) return defaultValue;
    const keys = path.split('.');
    let result = obj;

    for (const key of keys) {
      if (result === null || result === undefined || typeof result !== 'object') {
        return defaultValue;
      }
      result = result[key];
    }

    return result === null || result === undefined ? defaultValue : result;
  };

  // Add a formatter for customer name that handles different field formats
  const formatCustomerName = (sale) => {
    if (!sale) return 'N/A';

    // Try different possible locations for customer name
    return sale.customerName ||
           safeGet(sale, 'leadId.name') ||
           safeGet(sale, 'leadId.NAME') ||
           'Unknown Customer';
  };

  // Add a formatter for sales person name that handles different field formats
  const formatSalesPersonName = (sale) => {
    if (!sale) return 'N/A';

    // If salesPerson is an object with fullName or name
    if (typeof sale.salesPerson === 'object' && sale.salesPerson) {
      return sale.salesPerson.fullName || sale.salesPerson.name ||
sale.salesPerson.email || 'Unknown Sales Person';
    }
```

```
    // If salesPerson is just an ID, try to find the name in filterOptions
    if (typeof sale.salesPerson === 'string' && filterOptions.salesPersons) {
      const salesPersonData = filterOptions.salesPersons.find(sp => sp._id ===
sale.salesPerson);
      return salesPersonData ? (salesPersonData.fullName || salesPersonData.name
|| salesPersonData.email) : 'Unknown Sales Person';
    }

    // If we have a direct salesPersonName field
    if (sale.salesPersonName) {
      return sale.salesPersonName;
    }

    return 'N/A';
  };

  // Render a small tooltip with permission info, shown on hover
  const PermissionTooltip = ({ role }) => {
    let message = '';

    if (role === 'Sales Person') {
      message = 'As a Sales Person, you can only update the status of your own
sales.';
    } else if (role === 'Lead Person') {
      message = 'As a Lead Person, you can only edit your own leads.';
    }

    return message ? (
      <div className="relative inline-block ml-1 text-gray-400 dark:text-
gray-400">
        <span className="cursor-help text-sm">
          <i className="fas fa-info-circle"></i>
          <span className="tooltip absolute invisible group-hover:visible bg-
gray-800 text-white p-2 rounded text-xs w-48 -mt-16 -ml-24 z-10">
            {message}
          </span>
        </span>
      </div>
    ) : null;
  };

  // Replace the 'You don't have permission' message with a more helpful one
  const getPermissionMessage = (sale, userRole) => {
    if (!sale) return "Invalid sale data";

    if (userRole === 'Sales Person') {
      const salesPersonId = extractId(sale, 'salesPerson');
      const userId = user?._id;

      if (salesPersonId && userId && salesPersonId.toString() ===
userId.toString()) {
        return "As a Sales Person, you can edit and delete your own sales";
      } else {
        return "You can only edit and delete sales you created as a Sales Person";
      }
    }

    // Default message
```

```
    return "You don't have permission to update this sale. Only the creator or an
admin can update it.";
  };

  // Render a sale row
  const renderSaleRow = (sale, index) => (
    <tr key={sale._id} className={index % 2 === 0 ? 'bg-white dark:bg-gray-800' :
'bg-gray-50 dark:bg-gray-700'}>
      {/* Date Column */}
      <td className="px-6 py-4 whitespace-nowrap">
        {editingSale === sale._id ? (
          <div className="flex flex-col space-y-2">
            <input
              type="date"
              value={editValues.saleDate ? new
Date(editValues.saleDate).toISOString().split('T')[0] : new Date(sale.date ||
sale.createdAt).toISOString().split('T')[0]}
              onChange={(e) => handleInputChange('saleDate', new
Date(e.target.value))}
              className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
            />
            <div className="flex space-x-2">
              <button
                onClick={() => handleSave(sale._id)}
                className="text-green-600 hover:text-green-900 flex items-center
text-xs px-2 py-1 bg-green-50 rounded"
              >
                <FaCheck className="mr-1" /> Save
              </button>
              <button
                onClick={() => {
                  setEditingSale(null);
                  setEditValues({});
                }}
                className="text-red-600 hover:text-red-900 flex items-center text-
xs px-2 py-1 bg-red-50 rounded"
              >
                <FaTimes className="mr-1" /> Cancel
              </button>
            </div>
          </div>
        ) : (
          <div className="flex flex-col space-y-2">
            <div className="text-sm text-gray-900 dark:text-white">
              {formatDate(sale.date || sale.createdAt || new Date())}
            </div>
            {canEditSale(sale) && (
              <button
                onClick={() => handleEdit(sale)}
                className="text-blue-600 hover:text-blue-900 flex items-center
text-xs px-2 py-1 bg-blue-50 rounded w-16"
              >
                <FaEdit className="mr-1" /> Edit
              </button>
            )}
          </div>
        )}
      </td>
```

```
      {/* Customer Column */}
      <td className="px-6 py-4 whitespace-nowrap">
        <div className="text-sm font-medium text-gray-900 dark:text-white">
          {formatCustomerName(sale)}
        </div>
        <div className="text-xs text-gray-500 dark:text-gray-500">
          {sale.product || safeGet(sale, 'leadId.course') || 'No product'}
        </div>
        {editingSale === sale._id && (
          <div className="mt-2">
            <input
              type="text"
              placeholder="Lead By (Optional)"
              value={editValues.leadBy || ''}
              onChange={(e) => handleInputChange('leadBy', e.target.value)}
              className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
            />
            <div className="text-xs text-gray-500 dark:text-gray-500 mt-1 flex
items-center">
              <svg xmlns="http://www.w3.org/2000/svg" className="h-3 w-3 text-
gray-400 dark:text-gray-400 mr-1" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M13 16h-1v-4h-1m1-4h.01M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />
              </svg>
              <span>Name of person who led this sale</span>
            </div>
          </div>
        )}
        {!editingSale === sale._id && sale.leadBy && (
          <div className="text-xs text-gray-600 dark:text-gray-500 mt-1">
            Lead By: {sale.leadBy}
          </div>
        )}
      </td>
      {/* Contact Column */}
      <td className="px-6 py-4">
        <div className="flex flex-col space-y-1">
          {(sale.contactNumber || safeGet(sale, 'leadId.phone')) && (
            <div className="flex items-center">
              <button
                onClick={() => openWhatsApp(
                  sale.contactNumber || safeGet(sale, 'leadId.phone'),
                  sale.countryCode || safeGet(sale, 'leadId.countryCode', '+91')
                )}
                className="text-sm text-gray-900 dark:text-white flex items-
center hover:text-green-600"
                title="Open in WhatsApp"
              >
                <FaWhatsapp className="mr-1 text-green-500" />
                {sale.countryCode || safeGet(sale, 'leadId.countryCode', '+91')}
{sale.contactNumber || safeGet(sale, 'leadId.phone')}
              </button>
            </div>
          )}
          {(sale.email || safeGet(sale, 'leadId.email')) && (
            <div className="flex items-center">
              <button
```

```
                onClick={() => openEmail(sale.email || safeGet(sale,
'leadId.email'))}
                className="text-sm text-gray-500 dark:text-gray-500 flex items-
center hover:text-blue-600"
                title="Send email"
              >
                <FaEnvelope className="mr-1 text-blue-500" />
                {sale.email || safeGet(sale, 'leadId.email')}
              </button>
            </div>
          )}
          {editingSale === sale._id && (
            <div className="mt-2 flex flex-col space-y-2">
              <input
                type="text"
                placeholder="Login ID (Optional)"
                value={editValues.loginId || ''}
                onChange={(e) => handleInputChange('loginId', e.target.value)}
                className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
              />
              <input
                type="text"
                placeholder="Password (Optional)"
                value={editValues.password || ''}
                onChange={(e) => handleInputChange('password', e.target.value)}
                className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
              />
            </div>
          )}
          {!editingSale === sale._id && (sale.loginId || sale.password) && (
            <div className="mt-2 text-xs">
              {sale.loginId && <div>Login ID: {sale.loginId}</div>}
              {sale.password && <div>Password: {sale.password}</div>}
            </div>
          )}
        </div>
      </td>
      {/* Product Column */}
      <td className="px-6 py-4 whitespace-nowrap">
        {editingSale === sale._id ? (
          <div>
            <input
              type="text"
              value={editValues.product || ''}
              onChange={(e) => handleInputChange('product', e.target.value)}
              className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
              placeholder="Product or course name"
            />
            {(sale.leadPerson && typeof sale.leadPerson === 'object' &&
sale.leadPerson.fullName) && (
              <div className="text-xs text-gray-500 dark:text-gray-500 mt-1">
                Lead Person: {sale.leadPerson.fullName}
              </div>
            )}
          </div>
        ) : (
```

```
          <div className="text-sm text-gray-900 dark:text-white">{sale.product ||
safeGet(sale, 'course') || 'N/A'}</div>
        )}
        {!editingSale === sale._id && (sale.leadPerson && typeof sale.leadPerson
=== 'object' && sale.leadPerson.fullName) && (
          <div className="text-xs text-gray-500 dark:text-gray-500 mt-1">
            Lead Person: {sale.leadPerson.fullName}
          </div>
        )}
      </td>
      {/* Sales Person Column */}
      <td className="px-6 py-4 whitespace-nowrap">
        {editingSale === sale._id ? (
          <div className="flex flex-col space-y-2">
            <select
              value={editValues.salesPerson || sale.salesPerson?._id || ''}
              onChange={(e) => handleInputChange('salesPerson', e.target.value)}
              className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
            >
              <option value="">Select Sales Person</option>
              {filterOptions.salesPersons.map(sp => (
                <option key={sp._id} value={sp._id}>
                  {sp.fullName}
                </option>
              ))}
            </select>
          </div>
        ) : (
          <div className="text-sm text-gray-900 dark:text-white">
            {sale.salesPerson?.fullName || 'N/A'}
          </div>
        )}
      </td>
      {/* Amount Column */}
      <td className="px-6 py-4 whitespace-nowrap">
        {editingSale === sale._id ? (
          <div className="flex flex-col space-y-2">
            <div className="flex items-center space-x-2">
              <div className="relative">
                <span className="absolute left-3 top-1/2 transform -translate-
y-1/2 text-gray-500 dark:text-gray-500">
                  {getCurrencySymbol(editValues.currency)}
                </span>
                <input
                  id="amount"
                  type="number"
                  value={editValues.amount !== undefined ?
editValues.amount.toString() : "0"}
                  onChange={(e) => handleInputChange('amount', e.target.value)}
                  className="w-24 px-2 pl-7 border border-gray-300 dark:border-
slate-600 rounded"
                />
              </div>
              <select
                value={editValues.currency || 'USD'}
                onChange={(e) => handleInputChange('currency', e.target.value)}
                className="border border-gray-300 dark:border-slate-600 rounded
p-1 text-xs"
```

```
              >
                {currencyOptions.map(option => (
                  <option key={option.value} value={option.value}>{option.label}</
option>
                ))}
              </select>
            </div>
          </div>
        ) : (
          <div className="text-sm font-medium text-gray-900 dark:text-white">
            {formatCurrency(sale.amount || sale.totalCost || 0, sale.currency ||
'USD')}
          </div>
        )}
      </td>
      {/* Token Column */}
      <td className="px-6 py-4 whitespace-nowrap">
        {editingSale === sale._id ? (
          <div className="relative">
            <span className="absolute left-3 top-1/2 transform -translate-y-1/2
text-gray-500 dark:text-gray-500">
              {getCurrencySymbol(editValues.currency)}
            </span>
            <input
              id="token"
              type="number"
              value={editValues.token !== undefined ?
editValues.token.toString() : "0"}
              onChange={(e) => handleInputChange('token', e.target.value)}
              className="w-24 px-2 pl-7 border border-gray-300 dark:border-
slate-600 rounded"
            />
          </div>
        ) : (
          <div className="text-sm font-medium text-gray-900 dark:text-white">
            {formatCurrency(sale.token || sale.tokenAmount || 0, sale.currency ||
'USD')}
          </div>
        )}
      </td>
      {/* Pending Column */}
      <td className="px-6 py-4 whitespace-nowrap">
        {editingSale === sale._id ? (
          <div className="relative">
            <span className="absolute left-3 top-1/2 transform -translate-y-1/2
text-gray-500 dark:text-gray-500">
              {getCurrencySymbol(editValues.currency)}
            </span>
            <input
              id="pending"
              type="number"
              value={editValues.pending !== undefined ?
editValues.pending.toString() : "0"}
              onChange={(e) => handleInputChange('pending', e.target.value)}
              className="w-24 px-2 pl-7 border border-gray-300 dark:border-
slate-600 rounded"
              disabled={editValues.status === 'Completed'}
            />
          </div>
```

```
        ) : (
          <div className="text-sm font-medium text-gray-900 dark:text-white">
            {formatCurrency(
              sale.status === 'Completed' ? 0 :
              sale.pending !== undefined ? sale.pending :
              (sale.amount || sale.totalCost || 0) - (sale.token ||
sale.tokenAmount || 0),
              sale.currency || 'USD'
            )}
          </div>
        )}
      </td>
      {/* Status Column */}
      <td className="px-6 py-4 whitespace-nowrap">
        {editingSale === sale._id ? (
          <select
            value={editValues.status || sale.status}
            onChange={(e) => handleInputChange('status', e.target.value)}
            className={`text-sm px-2 py-1 rounded cursor-pointer ${
              sale.status === 'Pending' ? 'bg-yellow-100 text-yellow-800' :
              sale.status === 'Completed' ? 'bg-green-100 text-green-800' :
              sale.status === 'Cancelled' ? 'bg-red-100 text-red-800' :
              'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-gray-200'
            } border border-transparent focus:outline-none focus:ring-2
focus:ring-blue-500 dark:focus:ring-blue-400 focus:border-blue-500
dark:focus:border-blue-400`}
            disabled={!canEditSale(sale)}
          >
            {getAvailableStatusOptions(sale.status).map(status => (
              <option key={status} value={status}>{status}</option>
            ))}
          </select>
        ) : (
          <div className="relative">
            <select
              value={sale.status || 'Pending'}
              onChange={(e) => handleStatusChange(sale._id, e.target.value)}
              className={`text-sm px-2 py-1 rounded cursor-pointer appearance-
none w-auto pr-8 ${
                sale.status === 'Pending' ? 'bg-yellow-100 text-yellow-800' :
                sale.status === 'Completed' ? 'bg-green-100 text-green-800' :
                sale.status === 'Cancelled' ? 'bg-red-100 text-red-800' :
                'bg-gray-100 dark:bg-slate-700 text-gray-800 dark:text-gray-200'
              } border border-transparent focus:outline-none focus:ring-2
focus:ring-blue-500 dark:focus:ring-blue-400 focus:border-blue-500
dark:focus:border-blue-400`}
              disabled={!canEditSale(sale)}
            >
              {getAvailableStatusOptions(sale.status).map(status => (
                <option key={status} value={status}>{status}</option>
              ))}
            </select>
            <div className="pointer-events-none absolute inset-y-0 right-0 flex
items-center px-2 text-gray-700 dark:text-gray-400">
              <svg className="fill-current h-4 w-4" xmlns="http://www.w3.org/2000/
svg" viewBox="0 0 20 20">
                <path d="M9.293 12.95l.707.707L15.657 8l-1.414-1.414L10 10.828
5.757 6.586 4.343 8z"/>
              </svg>
```

```
            </div>
            {!canEditSale(sale) && (
              <div className="absolute left-0 -bottom-5 w-full">
                <div className="text-xs text-gray-500 dark:text-gray-500 italic">
                  {user?.role === 'Sales Person' ? "Can only update your own
sales" : "No edit permission"}
                </div>
              </div>
            )}
          </div>
        )}
      </td>
      {/* Remarks Column */}
      <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500 dark:text-
gray-400">
        {editingSale === sale._id ? (
          <div className="space-y-2">
            <textarea
              value={editValues.remarks || ''}
              onChange={(e) => handleInputChange('remarks', e.target.value)}
              className="w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:ring-blue-400 focus:border-blue-500 dark:focus:border-blue-400 bg-
white dark:bg-gray-700 text-gray-900 dark:text-gray-100"
              placeholder="Enter remarks for this update"
              rows="2"
              required
            />
            <div className="text-xs text-red-500">* Required</div>
          </div>
        ) : (
          <div className="text-sm max-w-xs overflow-hidden">
            {sale.remarks || '-'}
          </div>
        )}
      </td>
      {/* Actions Column */}
      <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-500 dark:text-
gray-400">
        <div className="flex items-center space-x-2">
          {canEditSale(sale) && (
            <button
              onClick={() => handleEdit(sale)}
              className="text-blue-600 hover:text-blue-900 dark:text-blue-400
dark:hover:text-blue-300 transition-colors duration-150"
              title="Edit sale"
            >
              <FaEdit className="h-5 w-5" />
            </button>
          )}
          {canDeleteSale(sale) && (
            <button
              onClick={() => handleDelete(sale._id)}
              className="text-red-600 hover:text-red-900 dark:text-red-400
dark:hover:text-red-300 transition-colors duration-150"
              title="Delete sale"
            >
              <FaTrash className="h-5 w-5" />
            </button>
```

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

  // Format currency for display
  const formatCurrency = (value, currencyCode = 'USD') => {
    // Get the currency symbol
    const currency = currencyOptions.find(c => c.value === currencyCode) ||
currencyOptions[0];
    const symbol = currency.symbol;

    // Return formatted currency
    return `${symbol}${parseFloat(value || 0).toFixed(2)}`;
  };

  // Get currency symbol
  const getCurrencySymbol = (currencyCode = 'USD') => {
    const currency = currencyOptions.find(c => c.value === currencyCode) ||
currencyOptions[0];
    return currency.symbol;
  };

  // Add this function near other handler functions
  const handleDelete = async (saleId) => {
    try {
      if (!window.confirm('Are you sure you want to delete this sale?')) {
        return;
      }

      const response = await salesAPI.delete(saleId);

      if (response.data.success) {
        // Remove the sale from both sales and filtered sales
        setSales(prevSales => prevSales.filter(sale => sale._id !== saleId));
        setFilteredSales(prevSales => prevSales.filter(sale => sale._id !==
saleId));
        toast.success('Sale deleted successfully');
      } else {
        toast.error('Failed to delete sale');
      }
    } catch (error) {
      console.error('Error deleting sale:', error);
      toast.error('Error deleting sale');
    }
  };

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-6">
          <h2 className="text-3xl font-bold">Sales Tracking</h2>
          <div className="flex gap-2">
            <button
              onClick={fetchSales}
              className="bg-blue-600 hover:bg-blue-700 text-white py-2 px-4
rounded-md transition duration-300"
              disabled={loading}
```

```
            >
              {loading ? 'Loading...' : 'Refresh'}
            </button>
            <button
              onClick={handleAddSaleClick}
              className="bg-green-600 hover:bg-green-700 text-white py-2 px-4
rounded-md transition duration-300"
            >
              Add New Sale
            </button>
          </div>
        </div>

        {/* Error message */}
        {error && (
          <div className="mb-4 p-3 bg-red-50 text-red-700 border border-red-200
rounded-md">
            {error}
          </div>
        )}

        {/* Debug information */}
        {!loading && (
          <div className="mb-4 p-3 bg-blue-50 text-blue-700 border border-
blue-200 rounded-md text-sm">
            <strong>Debug Info:</strong> Total Sales: {sales.length} |
            Filtered Sales: {filteredSales.length} |
            Show All: {showAllSales ? 'Yes' : 'No'} |
            Show Current Month: {showCurrentMonth ? 'Yes' : 'No'} |
            Filter Month/Year: {filterMonth}/{filterYear}
          </div>
        )}

        {/* Loading indicator */}
        {loading ? (
          <div className="text-center py-10">
            <div className="inline-block animate-spin rounded-full h-8 w-8 border-
t-2 border-b-2 border-blue-500"></div>
            <p className="mt-2 text-gray-600 dark:text-gray-500">Loading sales
data...</p>
          </div>
        ) : (
          <>
            {/* Advanced Filters */}
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700 rounded-lg
shadow-md dark:shadow-2xl mb-6 shadow-sm">
              <div className="flex justify-between items-center p-4 border-b
border-gray-200 dark:border-slate-700">
                <div className="flex items-center space-x-3">
                  <h2 className="text-lg font-medium">Advanced Filters</h2>
                  {Object.values(filters).some(filter => filter !== "") && (
                    <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                      {Object.values(filters).filter(filter => filter !==
"").length} active
                    </span>
                  )}
                </div>
```

```
                <div className="flex items-center space-x-3">
                  <button
                    onClick={resetFilters}
                    className="text-sm text-blue-600 hover:underline"
                  >
                    Reset All
                  </button>
                  <button
                    onClick={() => setShowAdvancedFilters(!showAdvancedFilters)}
                    className="flex items-center text-sm text-gray-600 dark:text-
gray-200 hover:text-gray-800"
                  >
                    {showAdvancedFilters ? 'Hide Filters' : 'Show Filters'}
                    <svg
                      className={`ml-1 h-4 w-4 transform transition-transform
${showAdvancedFilters ? 'rotate-180' : ''}`}
                      fill="none"
                      viewBox="0 0 24 24"
                      stroke="currentColor"
                    >
                      <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M19 9l-7 7-7-7" />
                    </svg>
                  </button>
                </div>
              </div>

              {showAdvancedFilters && (
                <div className="p-6">

              <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3
gap-4">
                {/* Search Field */}
                <div>
                  <label htmlFor="search" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                    Search
                  </label>
                  <input
                    type="text"
                    id="search"
                    name="search"
                    placeholder="Search customer, email, product..."
                    value={filters.search}
                    onChange={handleFilterChange}
                    className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                  />
                </div>

                {/* Status Filter */}
                <div>
                  <label htmlFor="status" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                    Status
                  </label>
                  <select
```

```
                    id="status"
                    name="status"
                    value={filters.status}
                    onChange={handleFilterChange}
                    className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                  >
                    <option value="">All Statuses</option>
                    {statusOptions.map(status => (
                      <option key={status} value={status}>{status}</option>
                    ))}
                  </select>
                </div>

                {/* Sales Person Filter - Only show for Admin/Manager */}
                {(user?.role === 'Admin' || user?.role === 'Manager') && (
                  <div>
                    <label htmlFor="salesPerson" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                      Sales Person
                    </label>
                    <select
                      id="salesPerson"
                      name="salesPerson"
                      value={filters.salesPerson}
                      onChange={handleFilterChange}
                      className="w-full border border-gray-300 dark:border-
gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                    >
                      <option value="">All Sales Persons</option>
                      {filterOptions.salesPersons.map(salesPerson => (
                        <option key={salesPerson._id} value={salesPerson._id}>
                          {salesPerson.fullName}
                        </option>
                      ))}
                    </select>
                  </div>
                )}

                {/* Country Filter */}
                <div>
                  <label htmlFor="country" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                    Country
                  </label>
                  <select
                    id="country"
                    name="country"
                    value={filters.country}
                    onChange={handleFilterChange}
                    className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                  >
```

```
                    <option value="">All Countries</option>
                    {filterOptions.countries.map(country => (
                      <option key={country} value={country}>{country}</option>
                    ))}
                  </select>
                </div>

                {/* Course/Product Filter */}
                <div>
                  <label htmlFor="course" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                    Course/Product
                  </label>
                  <select
                    id="course"
                    name="course"
                    value={filters.course}
                    onChange={handleFilterChange}
                    className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                  >
                    <option value="">All Courses</option>
                    {filterOptions.courses.map(course => (
                      <option key={course} value={course}>{course}</option>
                    ))}
                  </select>
                </div>

                {/* Lead Person Filter */}
                <div>
                  <label htmlFor="leadPerson" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                    Lead Person
                  </label>
                  <select
                    id="leadPerson"
                    name="leadPerson"
                    value={filters.leadPerson}
                    onChange={handleFilterChange}
                    className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                  >
                    <option value="">All Lead Persons</option>
                    {filterOptions.leadPersons.map(leadPerson => (
                      <option key={leadPerson._id} value={leadPerson._id}>
                        {leadPerson.fullName}
                      </option>
                    ))}
                  </select>
                </div>

                {/* Date Range - From */}
                <div>
                  <label htmlFor="dateFrom" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
```

```
                  Date From
                </label>
                <input
                  type="date"
                  id="dateFrom"
                  name="dateFrom"
                  value={filters.dateFrom}
                  onChange={handleFilterChange}
                  className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                />
              </div>

              {/* Date Range - To */}
              <div>
                <label htmlFor="dateTo" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                  Date To
                </label>
                <input
                  type="date"
                  id="dateTo"
                  name="dateTo"
                  value={filters.dateTo}
                  onChange={handleFilterChange}
                  className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                />
              </div>

              {/* Amount Range - From */}
              <div>
                <label htmlFor="amountFrom" className="block text-sm font-
medium text-gray-700 dark:text-gray-400 mb-1">
                  Amount From
                </label>
                <input
                  type="number"
                  id="amountFrom"
                  name="amountFrom"
                  placeholder="Min amount"
                  value={filters.amountFrom}
                  onChange={handleFilterChange}
                  className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                />
              </div>

              {/* Amount Range - To */}
              <div>
                <label htmlFor="amountTo" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                  Amount To
```

```
                  </label>
                  <input
                    type="number"
                    id="amountTo"
                    name="amountTo"
                    placeholder="Max amount"
                    value={filters.amountTo}
                    onChange={handleFilterChange}
                    className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                  />
                </div>

                {/* Currency Filter */}
                <div>
                  <label htmlFor="currency" className="block text-sm font-medium
text-gray-700 dark:text-gray-400 mb-1">
                    Currency
                  </label>
                  <select
                    id="currency"
                    name="currency"
                    value={filters.currency}
                    onChange={handleFilterChange}
                    className="w-full border border-gray-300 dark:border-gray-600
rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-gray-100-md
focus:outline-none focus:ring-2 focus:ring-blue-500 dark:focus:border-blue-400
focus:ring-offset-2 focus:border-blue-500"
                  >
                    <option value="">All Currencies</option>
                    {currencyOptions.map(currency => (
                      <option key={currency.value} value={currency.value}
>{currency.label}</option>
                    ))}
                  </select>
                </div>
              </div>

                              {/* Filter Summary */}
                <div className="mt-4 p-3 bg-gray-50 dark:bg-slate-800 rounded-
md">
                  <div className="text-sm text-gray-600 dark:text-gray-500">
                    <strong>{filteredSales.length}</strong> sales found from a
total of <strong>{sales.length}</strong> sales
                    {Object.values(filters).some(filter => filter !== "") && (
                      <span className="ml-2 text-blue-600">
                        (Filters applied)
                      </span>
                    )}
                  </div>
                </div>
              </div>
            )}
          </div>

          {/* Date Filter Controls */}
          <div className="mb-6 p-4 bg-white dark:bg-slate-900 border border-
```

```
slate-200 dark:border-slate-700 rounded-lg shadow-md dark:shadow-2xl transition-
all duration-200 ease-out shadow-sm">
                            <h3 className="text-lg font-semibold mb-3 text-
slate-900 dark:text-slate-100">Quick Date Filters</h3>
              <div className="flex flex-wrap items-center gap-4">
                <div>
                  <label htmlFor="month" className="block text-sm font-medium
text-gray-600 dark:text-gray-500 mb-1">Month</label>
                  <select
                    id="month"
                    value={filterMonth}
                    onChange={handleMonthChange}
                    className="border border-gray-300 dark:border-slate-600
rounded-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                    disabled={showCurrentMonth}
                  >
                    {months.map(month => (
                      <option key={month.value} value={month.value}>
                        {month.label}
                      </option>
                    ))}
                  </select>
                </div>

                <div>
                  <label htmlFor="year" className="block text-sm font-medium text-
gray-600 dark:text-gray-500 mb-1">Year</label>
                  <select
                    id="year"
                    value={filterYear}
                    onChange={handleYearChange}
                    className="border border-gray-300 dark:border-slate-600
rounded-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                    disabled={showCurrentMonth}
                  >
                    {years.map(year => (
                      <option key={year} value={year}>
                        {year}
                      </option>
                    ))}
                  </select>
                </div>

                <div className="flex items-center ml-4">
                  <input
                    id="currentMonth"
                    type="checkbox"
                    checked={showCurrentMonth}
                    onChange={() => {
                      setShowCurrentMonth(!showCurrentMonth);
                      if (!showCurrentMonth) {
                        setShowAllSales(false); // Disable show all when showing
current month
                      }
                    }}
                    className="h-4 w-4 text-blue-600 focus:ring-2 focus:ring-
blue-500 dark:focus:ring-offset-slate-900 focus:ring-offset-2 border-gray-300
```

```
dark:border-slate-600 rounded"
                  />
                  <label htmlFor="currentMonth" className="ml-2 block text-sm
text-gray-700 dark:text-gray-400">
                    Show Current Month Only
                  </label>
                </div>

                {/* Show All Sales option for Managers and Admins */}
                {(user?.role === 'Manager' || user?.role === 'Admin') && (
                  <div className="flex items-center ml-4">
                    <input
                      id="showAllSales"
                      type="checkbox"
                      checked={showAllSales}
                      onChange={handleShowAllSales}
                      className="h-4 w-4 text-blue-600 focus:ring-2 focus:ring-
blue-500 dark:focus:ring-offset-slate-900 focus:ring-offset-2 border-gray-300
dark:border-slate-600 rounded"
                    />
                    <label htmlFor="showAllSales" className="ml-2 block text-sm
text-gray-700 dark:text-gray-400 font-semibold text-blue-600">
                      Show All Sales (No Date Filter)
                    </label>
                  </div>
                )}

                <button
                  onClick={handleResetToCurrentMonth}
                  className="bg-gray-200 dark:bg-slate-600 hover:bg-gray-300 text-
gray-800 dark:text-gray-200 py-2 px-4 rounded-md ml-auto transition duration-300"
                >
                  Reset to Current Month
                </button>
              </div>

              <div className="mt-3 text-sm text-gray-500 dark:text-gray-500">
                {showAllSales ? (
                  <p>Quick filter: All sales regardless of date</p>
                ) : showCurrentMonth ? (
                  <p>Quick filter: Current month ({months[new
Date().getMonth()].label} {new Date().getFullYear()})</p>
                ) : (
                  <p>Quick filter: {months[filterMonth - 1].label} {filterYear}</
p>
                )}
                <p className="text-xs text-gray-400 dark:text-gray-400 mt-1">
                  Note: Advanced filters above will further refine these results
                </p>
              </div>
            </div>

            {/* Active Filters Summary */}
            {Object.values(filters).some(filter => filter !== "") && (
              <div className="mb-4 p-3 bg-blue-50 border border-blue-200 rounded-
lg">
                <div className="flex flex-wrap items-center gap-2">
                  <span className="text-sm font-medium text-blue-800">Active
Filters:</span>
```

```
                {filters.search && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Search: "{filters.search}"
                  </span>
                )}
                {filters.status && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Status: {filters.status}
                  </span>
                )}
                {filters.salesPerson && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Sales Person: {filterOptions.salesPersons.find(sp => sp._id
=== filters.salesPerson)?.fullName}
                  </span>
                )}
                {filters.country && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Country: {filters.country}
                  </span>
                )}
                {filters.course && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Course: {filters.course}
                  </span>
                )}
                {filters.leadPerson && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Lead Person: {filterOptions.leadPersons.find(lp => lp._id
=== filters.leadPerson)?.fullName}
                  </span>
                )}
                {(filters.dateFrom || filters.dateTo) && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Date: {filters.dateFrom || 'Start'} to {filters.dateTo ||
'End'}
                  </span>
                )}
                {(filters.amountFrom || filters.amountTo) && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Amount: {filters.amountFrom || '0'} - {filters.amountTo ||
'" '}
                  </span>
                )}
                {filters.currency && (
                  <span className="inline-flex items-center px-2 py-1 rounded-
full text-xs font-medium bg-blue-100 text-blue-800">
                    Currency: {filters.currency}
                  </span>
                )}
              </div>
```

```
            </div>
          )}

          <div className="overflow-x-auto bg-white dark:bg-slate-900 border
border-slate-200 dark:border-slate-700 rounded-lg shadow-md dark:shadow-2xl
transition-all duration-200 ease-out shadow-sm">
                            <table className="min-w-full divide-y divide-
slate-200 dark:divide-slate-700">
                <thead className="bg-gray-50 dark:bg-slate-800">
                <tr>
                  <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-500 uppercase tracking-wider">
                    #
                  </th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Date</th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Lead</th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Contact/Login</th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Product</th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Sales Person</th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Amount</th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Token</th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Pending</th>
                  <th className="px-6 py-3 text-left text-xs font-medium text-
gray-500 dark:text-gray-500 uppercase tracking-wider">Status</th>
                  {/* Add Remarks Column Header */}
                  <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-400 uppercase tracking-wider">
                    Remarks
                  </th>
                  {/* Actions Column Header */}
                  <th scope="col" className="px-6 py-3 text-left text-xs font-
medium text-gray-500 dark:text-gray-400 uppercase tracking-wider">
                    Actions
                  </th>
                </tr>
              </thead>
              <tbody className="bg-white dark:bg-gray-800 divide-y divide-
gray-200 dark:divide-gray-700">
                {filteredSales.map((sale, index) => (
                  <tr key={sale._id} className={index % 2 === 0 ? 'bg-white
dark:bg-gray-800' : 'bg-gray-50 dark:bg-gray-700'}>
                    <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-400">
                      {index + 1}
                    </td>
                    {/* Date Column */}
                    <td className="px-6 py-4 whitespace-nowrap">
                      {editingSale === sale._id ? (
                        <div className="flex flex-col space-y-2">
                          <input
                            type="date"
```

```
                              value={editValues.saleDate ? new
Date(editValues.saleDate).toISOString().split('T')[0] : new Date(sale.date ||
sale.createdAt).toISOString().split('T')[0]}
                              onChange={(e) => handleInputChange('saleDate', new
Date(e.target.value))}
                              className="w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                            />
                            <div className="flex space-x-2">
                              <button
                                onClick={() => handleSave(sale._id)}
                                className="text-green-600 hover:text-green-900
flex items-center text-xs px-2 py-1 bg-green-50 rounded"
                              >
                                <FaCheck className="mr-1" /> Save
                              </button>
                              <button
                                onClick={() => {
                                  setEditingSale(null);
                                  setEditValues({});
                                }}
                                className="text-red-600 hover:text-red-900 flex
items-center text-xs px-2 py-1 bg-red-50 rounded"
                              >
                                <FaTimes className="mr-1" /> Cancel
                              </button>
                            </div>
                          </div>
                        ) : (
                          <div className="flex flex-col space-y-2">
                            <div className="text-sm text-gray-900 dark:text-
white">
                              {formatDate(sale.date || sale.createdAt || new
Date())}
                            </div>
                            {canEditSale(sale) && (
                              <button
                                onClick={() => handleEdit(sale)}
                                className="text-blue-600 hover:text-blue-900 flex
items-center text-xs px-2 py-1 bg-blue-50 rounded w-16"
                              >
                                <FaEdit className="mr-1" /> Edit
                              </button>
                            )}
                          </div>
                        )}
                      </td>
                      {/* Customer Column */}
                      <td className="px-6 py-4 whitespace-nowrap">
                        <div className="text-sm font-medium text-gray-900
dark:text-white">
                          {formatCustomerName(sale)}
                        </div>
                        <div className="text-xs text-gray-500 dark:text-gray-500">
                          {sale.product || safeGet(sale, 'leadId.course') || 'No
product'}
                        </div>
                        {editingSale === sale._id && (
```

```
                          <div className="mt-2">
                            <input
                              type="text"
                              placeholder="Lead By (Optional)"
                              value={editValues.leadBy || ''}
                              onChange={(e) => handleInputChange('leadBy',
e.target.value)}
                              className="w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                            />
                            <div className="text-xs text-gray-500 dark:text-
gray-500 mt-1 flex items-center">
                              <svg xmlns="http://www.w3.org/2000/svg"
className="h-3 w-3 text-gray-400 dark:text-gray-400 mr-1" fill="none" viewBox="0
0 24 24" stroke="currentColor">
                                <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M13 16h-1v-4h-1m1-4h.01M21 12a9 9 0
11-18 0 9 9 0 0118 0z" />
                              </svg>
                              <span>Name of person who led this sale</span>
                            </div>
                          </div>
                        )}
                        {!editingSale === sale._id && sale.leadBy && (
                          <div className="text-xs text-gray-600 dark:text-
gray-500 mt-1">
                            Lead By: {sale.leadBy}
                          </div>
                        )}
                      </td>
                      {/* Contact Column */}
                      <td className="px-6 py-4">
                        <div className="flex flex-col space-y-1">
                          {(sale.contactNumber || safeGet(sale, 'leadId.phone'))
&& (
                            <div className="flex items-center">
                              <button
                                onClick={() => openWhatsApp(
                                  sale.contactNumber || safeGet(sale,
'leadId.phone'),
                                  sale.countryCode || safeGet(sale,
'leadId.countryCode', '+91')
                                )}
                                className="text-sm text-gray-900 dark:text-white
flex items-center hover:text-green-600"
                                title="Open in WhatsApp"
                              >
                                <FaWhatsapp className="mr-1 text-green-500" />
                                {sale.countryCode || safeGet(sale,
'leadId.countryCode', '+91')} {sale.contactNumber || safeGet(sale,
'leadId.phone')}
                              </button>
                            </div>
                          )}
                          {(sale.email || safeGet(sale, 'leadId.email')) && (
                            <div className="flex items-center">
                              <button
                                onClick={() => openEmail(sale.email ||
```

```
safeGet(sale, 'leadId.email'))}
                                  className="text-sm text-gray-500 dark:text-
gray-500 flex items-center hover:text-blue-600"
                                  title="Send email"
                                >
                                  <FaEnvelope className="mr-1 text-blue-500" />
                                  {sale.email || safeGet(sale, 'leadId.email')}
                                </button>
                              </div>
                            )}
                            {editingSale === sale._id && (
                              <div className="mt-2 flex flex-col space-y-2">
                                <input
                                  type="text"
                                  placeholder="Login ID (Optional)"
                                  value={editValues.loginId || ''}
                                  onChange={(e) => handleInputChange('loginId',
e.target.value)}
                                  className="w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                                />
                                <input
                                  type="text"
                                  placeholder="Password (Optional)"
                                  value={editValues.password || ''}
                                  onChange={(e) => handleInputChange('password',
e.target.value)}
                                  className="w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                                />
                              </div>
                            )}
                            {!editingSale === sale._id && (sale.loginId ||
sale.password) && (
                              <div className="mt-2 text-xs">
                                {sale.loginId && <div>Login ID: {sale.loginId}</
div>}
                                {sale.password && <div>Password: {sale.password}</
div>}
                              </div>
                            )}
                          </div>
                        </td>
                        {/* Product Column */}
                        <td className="px-6 py-4 whitespace-nowrap">
                          {editingSale === sale._id ? (
                            <div>
                              <input
                                type="text"
                                value={editValues.product || ''}
                                onChange={(e) => handleInputChange('product',
e.target.value)}
                                className="w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                                placeholder="Product or course name"
                              />
```

```
                            {(sale.leadPerson && typeof sale.leadPerson ===
'object' && sale.leadPerson.fullName) && (
                              <div className="text-xs text-gray-500 dark:text-
gray-500 mt-1">
                                Lead Person: {sale.leadPerson.fullName}
                              </div>
                            )}
                          </div>
                        ) : (
                          <div className="text-sm text-gray-900 dark:text-
white">{sale.product || safeGet(sale, 'course') || 'N/A'}</div>
                        )}
                        {!editingSale === sale._id && (sale.leadPerson && typeof
sale.leadPerson === 'object' && sale.leadPerson.fullName) && (
                          <div className="text-xs text-gray-500 dark:text-
gray-500 mt-1">
                            Lead Person: {sale.leadPerson.fullName}
                          </div>
                        )}
                      </td>
                      {/* Sales Person Column */}
                      <td className="px-6 py-4 whitespace-nowrap">
                        {editingSale === sale._id ? (
                          <div className="flex flex-col space-y-2">
                            <select
                              value={editValues.salesPerson ||
sale.salesPerson?._id || ''}
                              onChange={(e) => handleInputChange('salesPerson',
e.target.value)}
                              className="w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                            >
                              <option value="">Select Sales Person</option>
                              {filterOptions.salesPersons.map(sp => (
                                <option key={sp._id} value={sp._id}>
                                  {sp.fullName}
                                </option>
                              ))}
                            </select>
                          </div>
                        ) : (
                          <div className="text-sm text-gray-900 dark:text-white">
                            {sale.salesPerson?.fullName || 'N/A'}
                          </div>
                        )}
                      </td>
                      {/* Amount Column */}
                      <td className="px-6 py-4 whitespace-nowrap">
                        {editingSale === sale._id ? (
                          <div className="flex flex-col space-y-2">
                            <div className="flex items-center space-x-2">
                              <div className="relative">
                                <span className="absolute left-3 top-1/2
transform -translate-y-1/2 text-gray-500 dark:text-gray-500">
                                  {getCurrencySymbol(editValues.currency)}
                                </span>
                                <input
                                  id="amount"
```

```
                                    type="number"
                                    value={editValues.amount !== undefined ?
editValues.amount.toString() : "0"}
                                    onChange={(e) => handleInputChange('amount',
e.target.value)}
                                    className="w-24 px-2 pl-7 border border-
gray-300 dark:border-slate-600 rounded"
                                  />
                                </div>
                                <select
                                  value={editValues.currency || 'USD'}
                                  onChange={(e) => handleInputChange('currency',
e.target.value)}
                                  className="border border-gray-300 dark:border-
slate-600 rounded p-1 text-xs"
                                >
                                  {currencyOptions.map(option => (
                                    <option key={option.value} value={option.value}
>{option.label}</option>
                                  ))}
                                </select>
                              </div>
                            </div>
                          ) : (
                            <div className="text-sm font-medium text-gray-900
dark:text-white">
                              {formatCurrency(sale.amount || sale.totalCost || 0,
sale.currency || 'USD')}
                            </div>
                          )}
                        </td>
                        {/* Token Column */}
                        <td className="px-6 py-4 whitespace-nowrap">
                          {editingSale === sale._id ? (
                            <div className="relative">
                              <span className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-500 dark:text-gray-500">
                                {getCurrencySymbol(editValues.currency)}
                              </span>
                              <input
                                id="token"
                                type="number"
                                value={editValues.token !== undefined ?
editValues.token.toString() : "0"}
                                onChange={(e) => handleInputChange('token',
e.target.value)}
                                className="w-24 px-2 pl-7 border border-gray-300
dark:border-slate-600 rounded"
                              />
                            </div>
                          ) : (
                            <div className="text-sm font-medium text-gray-900
dark:text-white">
                              {formatCurrency(sale.token || sale.tokenAmount || 0,
sale.currency || 'USD')}
                            </div>
                          )}
                        </td>
                        {/* Pending Column */}
```

```
                      <td className="px-6 py-4 whitespace-nowrap">
                        {editingSale === sale._id ? (
                          <div className="relative">
                            <span className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-500 dark:text-gray-500">
                              {getCurrencySymbol(editValues.currency)}
                            </span>
                            <input
                              id="pending"
                              type="number"
                              value={editValues.pending !== undefined ?
editValues.pending.toString() : "0"}
                              onChange={(e) => handleInputChange('pending',
e.target.value)}
                              className="w-24 px-2 pl-7 border border-gray-300
dark:border-slate-600 rounded"
                              disabled={editValues.status === 'Completed'}
                            />
                          </div>
                        ) : (
                          <div className="text-sm font-medium text-gray-900
dark:text-white">
                            {formatCurrency(
                              sale.status === 'Completed' ? 0 :
                              sale.pending !== undefined ? sale.pending :
                              (sale.amount || sale.totalCost || 0) - (sale.token
|| sale.tokenAmount || 0),
                              sale.currency || 'USD'
                            )}
                          </div>
                        )}
                      </td>
                      {/* Status Column */}
                      <td className="px-6 py-4 whitespace-nowrap">
                        {editingSale === sale._id ? (
                          <select
                            value={editValues.status || sale.status}
                            onChange={(e) => handleInputChange('status',
e.target.value)}
                            className={`text-sm px-2 py-1 rounded cursor-pointer
${
                              sale.status === 'Pending' ? 'bg-yellow-100 text-
yellow-800' :
                              sale.status === 'Completed' ? 'bg-green-100 text-
green-800' :
                              sale.status === 'Cancelled' ? 'bg-red-100 text-
red-800' :
                              'bg-gray-100 dark:bg-slate-700 text-gray-800
dark:text-gray-200'
                            } border border-transparent focus:outline-none
focus:ring-2 focus:ring-blue-500 dark:focus:ring-blue-400 focus:border-blue-500
dark:focus:border-blue-400`}
                            disabled={!canEditSale(sale)}
                          >
                            {getAvailableStatusOptions(sale.status).map(status =>
(
                              <option key={status} value={status}>{status}</
option>
                            ))}
```

```
                          </select>
                        ) : (
                          <div className="relative">
                            <select
                              value={sale.status || 'Pending'}
                              onChange={(e) => handleStatusChange(sale._id,
e.target.value)}
                              className={`text-sm px-2 py-1 rounded cursor-
pointer appearance-none w-auto pr-8 ${
                                sale.status === 'Pending' ? 'bg-yellow-100 text-
yellow-800' :
                                sale.status === 'Completed' ? 'bg-green-100 text-
green-800' :
                                sale.status === 'Cancelled' ? 'bg-red-100 text-
red-800' :
                                'bg-gray-100 dark:bg-slate-700 text-gray-800
dark:text-gray-200'
                              } border border-transparent focus:outline-none
focus:ring-2 focus:ring-blue-500 dark:focus:ring-blue-400 focus:border-blue-500
dark:focus:border-blue-400`}
                              disabled={!canEditSale(sale)}
                            >
                              {getAvailableStatusOptions(sale.status).map(status
=> (
                                <option key={status} value={status}>{status}</
option>
                              ))}
                            </select>
                            <div className="pointer-events-none absolute inset-
y-0 right-0 flex items-center px-2 text-gray-700 dark:text-gray-400">
                              <svg className="fill-current h-4 w-4" xmlns="http://
www.w3.org/2000/svg" viewBox="0 0 20 20">
                                <path d="M9.293 12.95l.707.707L15.657
8l-1.414-1.414L10 10.828 5.757 6.586 4.343 8z"/>
                              </svg>
                            </div>
                            {!canEditSale(sale) && (
                              <div className="absolute left-0 -bottom-5 w-full">
                                <div className="text-xs text-gray-500 dark:text-
gray-500 italic">
                                  {user?.role === 'Sales Person' ? "Can only
update your own sales" : "No edit permission"}
                                </div>
                              </div>
                            )}
                          </div>
                        )}
                      </td>
                      {/* Remarks Column */}
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-400">
                        {editingSale === sale._id ? (
                          <div className="space-y-2">
                            <textarea
                              value={editValues.remarks || ''}
                              onChange={(e) => handleInputChange('remarks',
e.target.value)}
                              className="w-full px-3 py-2 border border-gray-300
dark:border-gray-600 rounded-md shadow-sm focus:outline-none focus:ring-2
```

```
focus:ring-blue-500 dark:focus:ring-blue-400 focus:border-blue-500
dark:focus:border-blue-400 bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100"
                              placeholder="Enter remarks for this update"
                              rows="2"
                              required
                            />
                            <div className="text-xs text-red-500">* Required</div>
                          </div>
                        ) : (
                          <div className="text-sm max-w-xs overflow-hidden">
                            {sale.remarks || '-'}
                          </div>
                        )}
                      </td>
                      {/* Actions Column */}
                      <td className="px-6 py-4 whitespace-nowrap text-sm text-
gray-500 dark:text-gray-400">
                        <div className="flex items-center space-x-2">
                          {canEditSale(sale) && (
                            <button
                              onClick={() => handleEdit(sale)}
                              className="text-blue-600 hover:text-blue-900
dark:text-blue-400 dark:hover:text-blue-300 transition-colors duration-150"
                              title="Edit sale"
                            >
                              <FaEdit className="h-5 w-5" />
                            </button>
                          )}
                          {canDeleteSale(sale) && (
                            <button
                              onClick={() => handleDelete(sale._id)}
                              className="text-red-600 hover:text-red-900
dark:text-red-400 dark:hover:text-red-300 transition-colors duration-150"
                              title="Delete sale"
                            >
                              <FaTrash className="h-5 w-5" />
                            </button>
                          )}
                        </div>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </>
        )}

        {/* Add Sale Modal */}
        {showAddModal && (
          <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center
justify-center z-50 p-4">
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out border border-slate-200 dark:border-slate-700 rounded-lg
shadow-lg dark:shadow-lg max-w-3xl w-full max-h-[90vh] overflow-y-auto shadow-sm">
              <div className="bg-green-600 text-white p-4 flex justify-between
items-center">
                <h3 className="text-xl font-bold">Add New Sale</h3>
```

```
                <button
                  onClick={() => setShowAddModal(false)}
                  className="text-white hover:text-gray-200 text-2xl"
                >
                  &times;
                </button>
              </div>

              <form onSubmit={handleSubmitNewSale} className="p-6">
                <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
                  {/* Reference Sale Toggle */}
                  <div className="col-span-2 mb-4">
                    <div className="flex items-center space-x-2">
                      <span className="text-sm font-medium text-gray-700
dark:text-gray-400">Sale Type:</span>
                      <div className="flex border border-gray-300 dark:border-
slate-600 rounded-md overflow-hidden">
                        <button
                          type="button"
                          className={`px-4 py-2 text-sm font-medium ${!
newSale.isReference ? 'bg-blue-600 text-white' : 'bg-gray-100 dark:bg-slate-700
text-gray-700 dark:text-gray-300 dark:text-gray-400'}`}
                          onClick={() => handleReferenceToggle(false)}
                        >
                          From Lead
                        </button>
                        <button
                          type="button"
                          className={`px-4 py-2 text-sm font-medium
${newSale.isReference ? 'bg-blue-600 text-white' : 'bg-gray-100 dark:bg-slate-700
text-gray-700 dark:text-gray-300 dark:text-gray-400'}`}
                          onClick={() => handleReferenceToggle(true)}
                        >
                          From Reference
                        </button>
                      </div>
                    </div>
                  </div>

                  {/* FROM LEAD UI */}
                  {!newSale.isReference && (
                    <>
                      {/* Lead Selection */}
                      <div className="col-span-2">
                        <label htmlFor="leadId" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Lead</label>
                        <select
                          id="leadId"
                          value={newSale.leadId}
                          onChange={handleLeadSelect}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                        >
                          <option value="">Select a lead</option>
                          {availableLeads.map(lead => (
                            <option key={lead._id} value={lead._id}>{lead.name} -
{lead.course}</option>
```

```
                        ))}
                      </select>
                    </div>

                    {/* Lead Person Selection */}
                    <div className="col-span-2">
                      <label htmlFor="leadPerson" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">
                        Lead Person
                        <span className="ml-1 text-xs text-blue-600 font-
normal">(Who should see this sale on their dashboard)</span>
                      </label>
                      <select
                        id="leadPerson"
                        value={newSale.leadPerson}
                        onChange={(e) => handleNewSaleChange('leadPerson',
e.target.value)}
                        className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                      >
                        <option value="">Select a lead person</option>
                        {leadPersonOptions.map(person => (
                          <option key={person.value} value={person.value}
>{person.label}</option>
                        ))}
                      </select>
                      {loadingLeadPersons && (
                        <div className="mt-1 text-sm text-gray-500 dark:text-
gray-500">Loading lead persons...</div>
                      )}
                      <p className="mt-1 text-xs text-gray-500 dark:text-
gray-500">
                        The selected lead person will see this sale on their
dashboard
                      </p>
                    </div>

                    {/* Sale Date */}
                    <div className="col-span-2 md:col-span-1">
                      <label htmlFor="saleDate" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Sale Date</label>
                      <input
                        id="saleDate"
                        type="date"
                        value={newSale.saleDate ? new
Date(newSale.saleDate).toISOString().split('T')[0] : new
Date().toISOString().split('T')[0]}
                        onChange={(e) => handleNewSaleChange('saleDate', new
Date(e.target.value))}
                        className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                      />
                    </div>

                    {/* Lead By (Optional) */}
```

```
                    <div className="col-span-2 md:col-span-1">
                      <label htmlFor="leadBy" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">
                        Lead By (Optional)
                        <span className="ml-1 inline-block relative group">
                          <svg xmlns="http://www.w3.org/2000/svg"
className="h-4 w-4 text-gray-400 dark:text-gray-400 cursor-help" fill="none"
viewBox="0 0 24 24" stroke="currentColor">
                            <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M13 16h-1v-4h-1m1-4h.01M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />
                          </svg>
                          <span className="absolute bottom-full left-1/2
transform -translate-x-1/2 w-48 bg-gray-800 text-white text-xs rounded py-1 px-2
hidden group-hover:block">
                            Name of the person who led this sale (can be
different from the Lead Person assigned in the system)
                          </span>
                        </span>
                      </label>
                      <input
                        id="leadBy"
                        type="text"
                        value={newSale.leadBy}
                        onChange={(e) => handleNewSaleChange('leadBy',
e.target.value)}
                        className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                        placeholder="Who led this sale?"
                      />
                    </div>

                    {/* Product (pulled from lead but can be modified) */}
                    <div className="col-span-2">
                      <label htmlFor="product" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Product</label>
                      <input
                        id="product"
                        type="text"
                        value={newSale.product}
                        onChange={(e) => handleNewSaleChange('product',
e.target.value)}
                        className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                        placeholder="Product or course name"
                      />
                    </div>

                    {/* Login Credentials (Optional) */}
                    <div className="col-span-2 md:col-span-1">
                      <label htmlFor="loginId" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Login ID (Optional)</label>
                      <input
                        id="loginId"
                        type="text"
                        value={newSale.loginId}
```

```
                          onChange={(e) => handleNewSaleChange('loginId',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                          placeholder="Customer login ID"
                        />
                      </div>

                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="password" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Password (Optional)</label>
                        <input
                          id="password"
                          type="text"
                          value={newSale.password}
                          onChange={(e) => handleNewSaleChange('password',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                          placeholder="Customer password"
                        />
                      </div>

                      {/* Currency */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="currency" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Currency</label>
                        <select
                          id="currency"
                          value={newSale.currency}
                          onChange={(e) => handleNewSaleChange('currency',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                        >
                          {currencyOptions.map(option => (
                            <option key={option.value} value={option.value}
>{option.label}</option>
                          ))}
                        </select>
                      </div>

                      {/* Amount */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="amount" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Amount</label>
                        <div className="relative mt-1">
                          <span className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-500 dark:text-
gray-500">{getCurrencySymbol(newSale.currency)}</span>
                          <input
                            id="amount"
                            type="number"
```

```
                            value={newSale.amount}
                            onChange={(e) => handleNewSaleChange('amount',
e.target.value)}
                            className="block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                            placeholder="0.00"
                          />
                        </div>
                      </div>

                      {/* Token */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="token" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Token</label>
                          <div className="relative mt-1">
                            <span className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-500 dark:text-
gray-500">{getCurrencySymbol(newSale.currency)}</span>
                            <input
                              id="token"
                              type="number"
                              value={newSale.token}
                              onChange={(e) => handleNewSaleChange('token',
e.target.value)}
                              className="block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                              placeholder="0.00"
                            />
                        </div>
                      </div>

                      {/* Pending (calculated automatically) */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="pending" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Pending</label>
                          <div className="relative mt-1">
                            <span className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-500 dark:text-
gray-500">{getCurrencySymbol(newSale.currency)}</span>
                            <input
                              id="pending"
                              type="number"
                              value={newSale.pending}
                              onChange={(e) => handleNewSaleChange('pending',
e.target.value)}
                              className="block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                              placeholder="0.00"
                              disabled={newSale.status === 'Completed'}
                            />
                        </div>
                      </div>
```

```
                    {/* Status */}
                    <div className="col-span-2 md:col-span-1">
                      <label htmlFor="status" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Status</label>
                      <select
                        id="status"
                        value={newSale.status}
                        onChange={(e) => handleNewSaleChange('status',
e.target.value)}
                        className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                      >
                        {statusOptions.map(status => (
                          <option key={status} value={status}>{status}</option>
                        ))}
                      </select>
                    </div>
                  </>
                )}

                {/* REFERENCE SALE UI */}
                {newSale.isReference && (
                  <>
                    {/* Customer Name */}
                    <div className="col-span-2 md:col-span-1">
                      <label htmlFor="customerName" className="block text-sm
font-medium text-gray-700 dark:text-gray-400">Customer Name</label>
                      <input
                        id="customerName"
                        type="text"
                        value={newSale.customerName}
                        onChange={(e) => handleNewSaleChange('customerName',
e.target.value)}
                        className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                        placeholder="Enter customer name"
                      />
                    </div>

                    {/* Sale Date */}
                    <div className="col-span-2 md:col-span-1">
                      <label htmlFor="saleDate" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Sale Date</label>
                      <input
                        id="saleDate"
                        type="date"
                        value={newSale.saleDate ? new
Date(newSale.saleDate).toISOString().split('T')[0] : new
Date().toISOString().split('T')[0]}
                        onChange={(e) => handleNewSaleChange('saleDate', new
Date(e.target.value))}
                        className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
```

```
                        />
                      </div>

                      {/* Contact Number */}
                      <div className="col-span-2">
                        <label htmlFor="contactNumber" className="block text-sm
font-medium text-gray-700 dark:text-gray-400 mb-1">Contact Number</label>
                        <PhoneInput
                          country={'us'}
                          value={newSale.contactNumber}
                          onChange={(value, data) => {
                            // Update both the contact number and country code
                            handleNewSaleChange('contactNumber', value);
                            handleNewSaleChange('countryCode', `+${data.dialCode}
`);
                          }}
                          inputProps={{
                            id: 'contactNumber',
                            name: 'contactNumber',
                            required: true,
                          }}
                          containerStyle={phoneInputStyle.container}
                          inputStyle={phoneInputStyle.inputStyle}
                          buttonStyle={phoneInputStyle.buttonStyle}
                          dropdownStyle={phoneInputStyle.dropdownStyle}
                          enableSearch={true}
                          searchPlaceholder="Search country..."
                        />
                      </div>

                      {/* Email */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="email" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Email</label>
                        <input
                          id="email"
                          type="email"
                          value={newSale.email}
                          onChange={(e) => handleNewSaleChange('email',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                          placeholder="customer@example.com"
                        />
                      </div>

                      {/* Country */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="country" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Country</label>
                        <input
                          id="country"
                          type="text"
                          value={newSale.country}
                          onChange={(e) => handleNewSaleChange('country',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
```

```
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                          placeholder="Enter country"
                        />
                      </div>

                      {/* Lead By (Optional) */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="leadBy" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">
                          Lead By (Optional)
                          <span className="ml-1 inline-block relative group">
                            <svg xmlns="http://www.w3.org/2000/svg"
className="h-4 w-4 text-gray-400 dark:text-gray-400 cursor-help" fill="none"
viewBox="0 0 24 24" stroke="currentColor">
                              <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M13 16h-1v-4h-1m1-4h.01M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />
                            </svg>
                            <span className="absolute bottom-full left-1/2
transform -translate-x-1/2 w-48 bg-gray-800 text-white text-xs rounded py-1 px-2
hidden group-hover:block">
                              Name of the person who led this sale (can be
different from the Lead Person assigned in the system)
                            </span>
                          </span>
                        </label>
                        <input
                          id="leadBy"
                          type="text"
                          value={newSale.leadBy}
                          onChange={(e) => handleNewSaleChange('leadBy',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                          placeholder="Who led this sale?"
                        />
                      </div>

                      {/* Currency */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="currency" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Currency</label>
                        <select
                          id="currency"
                          value={newSale.currency}
                          onChange={(e) => handleNewSaleChange('currency',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                        >
                          {currencyOptions.map(option => (
                            <option key={option.value} value={option.value}
>{option.label}</option>
                          ))}
```

```
                        </select>
                      </div>

                      {/* Product */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="product" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Product</label>
                        <input
                          id="product"
                          type="text"
                          value={newSale.product}
                          onChange={(e) => handleNewSaleChange('product',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                          placeholder="Product or course name"
                        />
                      </div>

                      {/* Login Credentials (Optional) */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="loginId" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Login ID (Optional)</label>
                        <input
                          id="loginId"
                          type="text"
                          value={newSale.loginId}
                          onChange={(e) => handleNewSaleChange('loginId',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                          placeholder="Customer login ID"
                        />
                      </div>

                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="password" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Password (Optional)</label>
                        <input
                          id="password"
                          type="text"
                          value={newSale.password}
                          onChange={(e) => handleNewSaleChange('password',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                          placeholder="Customer password"
                        />
                      </div>

                      {/* Lead Person Selection */}
                      <div className="col-span-2">
                        <label htmlFor="leadPerson" className="block text-sm font-
```

```
medium text-gray-700 dark:text-gray-400">Lead Person</label>
                        <select
                          id="leadPerson"
                          value={newSale.leadPerson}
                          onChange={(e) => handleNewSaleChange('leadPerson',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                        >
                          <option value="">Select a lead person</option>
                          {leadPersonOptions.map(person => (
                            <option key={person.value} value={person.value}
>{person.label}</option>
                          ))}
                        </select>
                        {loadingLeadPersons && (
                          <div className="mt-1 text-sm text-gray-500 dark:text-
gray-500">Loading lead persons...</div>
                        )}
                      </div>

                      {/* Amount */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="amount" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Amount</label>
                        <div className="relative mt-1">
                          <span className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-500 dark:text-
gray-500">{getCurrencySymbol(newSale.currency)}</span>
                          <input
                            id="amount"
                            type="number"
                            value={newSale.amount}
                            onChange={(e) => handleNewSaleChange('amount',
e.target.value)}
                            className="block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                            placeholder="0.00"
                          />
                        </div>
                      </div>

                      {/* Token */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="token" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Token</label>
                        <div className="relative mt-1">
                          <span className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-500 dark:text-
gray-500">{getCurrencySymbol(newSale.currency)}</span>
                          <input
                            id="token"
                            type="number"
                            value={newSale.token}
                            onChange={(e) => handleNewSaleChange('token',
e.target.value)}
```

```
                            className="block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                            placeholder="0.00"
                          />
                        </div>
                      </div>

                      {/* Pending (calculated automatically) */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="pending" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Pending</label>
                        <div className="relative mt-1">
                          <span className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-500 dark:text-
gray-500">{getCurrencySymbol(newSale.currency)}</span>
                          <input
                            id="pending"
                            type="number"
                            value={newSale.pending}
                            onChange={(e) => handleNewSaleChange('pending',
e.target.value)}
                            className="block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                            placeholder="0.00"
                            disabled={newSale.status === 'Completed'}
                          />
                        </div>
                      </div>

                      {/* Status */}
                      <div className="col-span-2 md:col-span-1">
                        <label htmlFor="status" className="block text-sm font-
medium text-gray-700 dark:text-gray-400">Status</label>
                        <select
                          id="status"
                          value={newSale.status}
                          onChange={(e) => handleNewSaleChange('status',
e.target.value)}
                          className="mt-1 block w-full border border-gray-300
dark:border-gray-600 rounded bg-white dark:bg-gray-700 text-gray-900 dark:text-
gray-100-md p-2 focus:outline-none focus:ring-2 focus:ring-blue-500
dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500"
                        >
                          {statusOptions.map(status => (
                            <option key={status} value={status}>{status}</option>
                          ))}
                        </select>
                      </div>
                    </>
                  )}
                </div>
                <div className="mt-6">
                  <button
                    type="submit"
                    className="px-4 py-2 bg-blue-600 hover:bg-blue-700 dark:bg-
```

```
blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white rounded-md"
                  >
                    Add Sale
                  </button>
                </div>
              </form>
            </div>
          </div>
        )}
      </div>
    </Layout>
  );
};

export default SalesTrackingPage;
```

src/pages/TaskManagementPage.jsx

```
import React, { useState, useEffect } from "react";
import Layout from "../components/Layout/Layout";
import { useAuth } from "../context/AuthContext";
import axios from "axios";
import { format } from "date-fns";
import { toast } from "react-toastify";
import { leadsAPI, tasksAPI } from "../services/api";
import notificationService from "../services/notificationService";

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
// Get API URL for Vite (fixes "process is not defined" error)
const API_URL = import.meta.env.VITE_API_URL || 'http://localhost:8080';

// Directly get token from localStorage instead of only relying on AuthContext
const getToken = () => localStorage.getItem('token');

const TaskManagementPage = () => {
  const { user } = useAuth();
  const [token, setToken] = useState(localStorage.getItem('token'));
  const [tasks, setTasks] = useState([]);
  const [leads, setLeads] = useState([]);
  const [loading, setLoading] = useState(true);
  const [leadsLoading, setLeadsLoading] = useState(false);
  const [modalOpen, setModalOpen] = useState(false);
  const [currentTask, setCurrentTask] = useState(null);
  // New states for manual entry and search
  const [manualCustomerMode, setManualCustomerMode] = useState(false);
  const [leadSearchQuery, setLeadSearchQuery] = useState('');

  // Debug token information
  useEffect(() => {
    // Get fresh token for each API call
    const freshToken = getToken();
    setToken(freshToken);

    if (!freshToken) {
      toast.error("No authentication token found. Please try logging out and back
in.");
```

```
  }
}, []);

// Form state
const [formData, setFormData] = useState({
  title: "",
  description: "",
  taskType: "Exam",
  customer: "",
  examDate: "",
  examTime: "",
  // New fields for manual customer entry
  manualCustomerName: "",
  manualCustomerEmail: "",
  manualCustomerPhone: "",
  manualCustomerCourse: ""
});

const fetchTasks = async () => {
  try {
    setLoading(true);

    // Get fresh token for API call
    const freshToken = getToken();

    // Check token validity first
    if (!freshToken) {
      toast.error("Authentication token missing. Please log in again.");
      setLoading(false);
      return;
    }

    // Use tasksAPI service instead of direct Axios
    const response = await tasksAPI.getAll();

    if (response.data && response.data.data) {
      console.log('Tasks loaded:', response.data.data);

      // Check if customer data is properly populated
      const tasksWithValidCustomers = response.data.data.map(task => {
        // If customer isn't properly populated, add a placeholder
        if (!task.customer || typeof task.customer === 'string') {
          console.log('Task has unpopulated customer:', task);
          // Try to find matching customer from leads list
          const matchingLead = leads.find(lead => lead._id === task.customer);
          if (matchingLead) {
            task.customer = matchingLead;
          }
        }
        return task;
      });

      setTasks(tasksWithValidCustomers);
    } else {
      setTasks([]);
      console.log('No tasks found in response');
    }
  } catch (error) {
    console.error('Error fetching tasks:', error);
```

```
      if (error.response && error.response.status === 401) {
        toast.error("Authentication failed. Please log in again.");
      } else {
        toast.error("Failed to fetch tasks: " + (error.message || "Unknown
error"));
      }

      setTasks([]);
    } finally {
      setLoading(false);
    }
  };

  const fetchLeads = async () => {
    setLeadsLoading(true);

    try {
      // Get fresh token for API call
      const freshToken = getToken();

      // Check token validity first
      if (!freshToken) {
        toast.error("Authentication token missing. Please log in again.");
        setLeadsLoading(false);
        return;
      }

      // Use the API service to get all leads
      const response = await leadsAPI.getAll();

      // Extract customers data from response
      let apiCustomers = [];
      if (response.data?.data && Array.isArray(response.data.data)) {
        apiCustomers = response.data.data;
      } else if (response.data?.leads && Array.isArray(response.data.leads)) {
        apiCustomers = response.data.leads;
      } else if (Array.isArray(response.data)) {
        apiCustomers = response.data;
      }

      if (apiCustomers.length > 0) {
        setLeads(apiCustomers);

        // Show success message with breakdown
        const leadCount = apiCustomers.filter(c => !c.isReferenceSale).length;
        const refCount = apiCustomers.filter(c => c.isReferenceSale).length;

        if (leadCount === 0 && refCount === 0) {
          toast.info("No customer data available");
        }
      }
    } catch (error) {
      // Detailed error logging
      if (error.response) {
        if (error.response.status === 401) {
          toast.error("Authentication failed. Your session may have expired.");
        } else {
          toast.error("Could not fetch customer data: " +
(error.response.data?.message || "Server error"));
```

```
      }
    } else if (error.request) {
      toast.error("No response from server. Check your connection.");
    } else {
      toast.error("Error: " + error.message);
    }
  } finally {
    setLeadsLoading(false);
  }
};

useEffect(() => {
  if (token) {
    fetchTasks();
    fetchLeads();
  }
}, [token]);

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

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

  try {
    // Client-side validation
    if (manualCustomerMode) {
      // Validate required manual customer fields
      if (!formData.manualCustomerName) {
        toast.error("Please enter customer name");
        return;
      }
      if (!formData.manualCustomerPhone) {
        toast.error("Please enter customer phone number");
        return;
      }
      if (!formData.manualCustomerCourse) {
        toast.error("Please enter course/exam name");
        return;
      }
    } else {
      // Validate customer selection
      if (!formData.customer) {
        toast.error("Please select a customer");
        return;
      }
    }

    // Combine date and time
    const combinedDateTime = new Date(`${formData.examDate}T${formData.examTime}
`);

    let taskData = {
      title: formData.title,
```

```
        description: formData.description,
        taskType: formData.taskType,
        examDate: combinedDateTime,
        examDateTime: combinedDateTime,
        assignedTo: user._id,
        salesPerson: user._id,
        course: manualCustomerMode ? formData.manualCustomerCourse :
(formData.customer ? 'Selected Customer Course' : 'General Course')
      };

      // Handle manual customer entry vs. selecting existing customer
      if (manualCustomerMode) {
        // Create a custom customer object for the task
        taskData.manualCustomer = {
          name: formData.manualCustomerName,
          email: formData.manualCustomerEmail,
          contactNumber: formData.manualCustomerPhone,
          course: formData.manualCustomerCourse
        };
      } else {
        // Use selected customer from dropdown
        taskData.customer = formData.customer;
      }

      let response;
      if (currentTask) {
        // Update existing task
        response = await tasksAPI.update(currentTask._id, taskData);
      } else {
        // Create new task
        response = await tasksAPI.create(taskData);
      }

      if (response.data.success) {
        toast.success(currentTask ? "Task updated successfully" : "Task created
successfully");

        // Reset form and close modal
        resetForm();

        // Refresh tasks list
        fetchTasks();
      } else {
        toast.error(response.data.message || "Error saving task");
      }
    } catch (error) {
      // Detailed error handling
      if (error.response) {
        if (error.response.status === 401) {
          toast.error("Authentication failed. Please log out and log back in.");
        } else {
          toast.error(error.response?.data?.message || "Failed to save task");
        }
      } else {
        toast.error("Network error - could not connect to server");
      }
    }
  };
```

```
  const handleEditTask = (task) => {
    console.log('Editing task:', task);

    const examDate = format(new Date(task.examDate), "yyyy-MM-dd");
    const examTime = format(new Date(task.examDate), "HH:mm");

    // Check if this is a manual customer entry
    const isManualCustomer = task.customer?.isManualEntry ||
                            (!task.customer && task.manualCustomer);

    // Handle different customer data formats
    let customerId = '';
    if (!isManualCustomer && task.customer) {
      if (typeof task.customer === 'string') {
        customerId = task.customer;
      } else if (task.customer._id && task.customer._id !== 'manual') {
        customerId = task.customer._id;
      }
    }

    // Set manual mode based on the task type
    setManualCustomerMode(isManualCustomer);

    // Get customer data for manual entry fields
    const customerData = isManualCustomer ?
      (task.manualCustomer || task.customer) : null;

    setFormData({
      title: task.title,
      description: task.description,
      taskType: task.taskType || 'Exam',
      customer: customerId,
      examDate,
      examTime,
      // Populate manual customer fields if this is a manual entry
      manualCustomerName: customerData?.name || '',
      manualCustomerEmail: customerData?.email || '',
      manualCustomerPhone: customerData?.contactNumber || '',
      manualCustomerCourse: customerData?.course || ''
    });

    setCurrentTask(task);
    setModalOpen(true);

    // If customer ID is missing or invalid, show a message
    if (!isManualCustomer && !customerId) {
      toast.warning("Customer data may be incomplete. Please select a customer
again.");
    }
  };

  const handleDeleteTask = async (id) => {
    if (window.confirm("Are you sure you want to delete this task?")) {
      try {
        const response = await tasksAPI.delete(id);

        if (response.data && response.data.success) {
          toast.success("Task deleted successfully");
          // Refresh tasks list
```

```
        fetchTasks();
      } else {
        toast.error(response.data?.message || "Failed to delete task");
      }
    } catch (error) {
      if (error.response && error.response.status === 401) {
        toast.error("Authentication failed. Please log in again.");
      } else {
        toast.error("Failed to delete task");
      }
    }
  }
};

const handleMarkCompleted = async (id, completed) => {
  try {
    const response = await tasksAPI.markCompleted(id, completed);

    if (response.data && response.data.success) {
      toast.success("Task status updated");
      // Refresh tasks list
      fetchTasks();
    } else {
      toast.error(response.data?.message || "Failed to update task status");
    }
  } catch (error) {
    if (error.response && error.response.status === 401) {
      toast.error("Authentication failed. Please log in again.");
    } else {
      toast.error("Failed to update task status");
    }
  }
};

const formatDateTime = (dateString) => {
  const date = new Date(dateString);
  return format(date, "MMM dd, yyyy 'at' h:mm a");
};

// Helper function to display lead/customer name
const getLeadName = (lead) => {
  // Handle different data structures for leads
  if (!lead) return 'No Name';
  console.log('Getting name for lead:', lead);

  // Check if lead is just an ID (string) rather than an object
  if (typeof lead === 'string') {
    return `Customer (ID: ${lead.substring(0, 6)}...)`;
  }

  // If lead is not populated properly
  if (!Object.keys(lead).length) {
    return 'Unknown Customer';
  }

  let name = '';

  // Try all possible name fields
  const possibleNameFields = ['name', 'NAME', 'customerName', 'fullName'];
```

```
    for (const field of possibleNameFields) {
      if (lead[field] && typeof lead[field] === 'string' && lead[field].trim() !
== '') {
        name = lead[field];
        break;
      }
    }

    // If still no name found, look for any field containing 'name'
    if (!name) {
      const nameKeys = Object.keys(lead).filter(key =>
        key.toLowerCase().includes('name') &&
        lead[key] &&
        typeof lead[key] === 'string' &&
        lead[key].trim() !== ''
      );

      if (nameKeys.length > 0) {
        name = lead[nameKeys[0]];
      }
    }

    // If still no name, use course info or ID as fallback
    if (!name || name === 'Unnamed Customer') {
      if (lead.course || lead.COURSE) {
        name = `Student for ${lead.course || lead.COURSE}`;
      } else if (lead._id) {
        name = `Customer ${lead._id.substring(0, 6)}...`;
      } else {
        name = 'Unnamed Customer';
      }
    }

    // Add Reference tag if it's a reference customer
    if (lead.isReferenceSale) {
      return `${name} [Reference]`;
    }

    // Add course info if available
    const course = lead.course || lead.COURSE;
    if (course && !name.includes(course)) {
      return `${name} (${course})`;
    }

    return name;
  };

  // Helper function to get lead contact info (email or phone)
  const getLeadContact = (lead) => {
    if (!lead) return '';
    let contact = '';

    // Try to find email
    if (lead.email) {
      contact = lead.email;
    } else if (lead.EMAIL) {
      contact = lead.EMAIL;
    } else if (lead["E-MAIL"]) {
```

```
      contact = lead["E-MAIL"];
    }
    // If no email, try phone/mobile
    else if (lead.contactNumber) {
      contact = lead.contactNumber;
    } else if (lead.phone) {
      contact = lead.phone;
    } else if (lead.MOBILE) {
      contact = lead.MOBILE;
    } else if (lead.CONTACT) {
      contact = lead.CONTACT;
    } else if (lead.NUMBER) {
      contact = lead.NUMBER;
    }

    return contact ? ` (${contact})` : '';
  };

  // Function to reset the form state
  const resetForm = () => {
    setFormData({
      title: "",
      description: "",
      taskType: "Exam",
      customer: "",
      examDate: "",
      examTime: "",
      manualCustomerName: "",
      manualCustomerEmail: "",
      manualCustomerPhone: "",
      manualCustomerCourse: ""
    });
    setCurrentTask(null);
    setManualCustomerMode(false);
    setLeadSearchQuery('');
  };

  return (
    <Layout>
      <div className="bg-gray-50 dark:bg-slate-800 transition-all duration-200
ease-out min-h-screen py-4 md:py-8">
        <div className="container mx-auto px-4">
          {/* Header */}
          <div className="flex flex-col sm:flex-row justify-between items-center
mb-6 gap-3">
            <h1 className="text-xl sm:text-2xl font-bold text-gray-800 dark:text-
gray-200">Task Management</h1>
            <div className="flex gap-2">
              <button
                onClick={() => {
                  console.log('Ø=Ý  Testing notification sound...');
                  notificationService.playNotificationSound();
                  toast.success('Ø=Ý  Test sound played! (You need to interact with
the page first for sound to work)');
                }}
                className="w-full sm:w-auto px-3 py-2 bg-orange-500 text-white
rounded-md hover:bg-orange-600 text-sm sm:text-base flex items-center gap-2"
                title="Test the exam reminder sound"
              >
```

```
                Ø=Ý  Test Sound
              </button>
              <button
                onClick={() => {
                  resetForm();
                  setModalOpen(true);
                }}
                className="w-full sm:w-auto px-4 py-2 bg-blue-600 hover:bg-
blue-700 dark:bg-blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl
hover:shadow-md transition-all duration-200 text-white rounded-md text-sm sm:text-
base"
              >
                Schedule New Exam
              </button>
            </div>
          </div>

          {/* Add a notification for authentication issues */}
          {!getToken() && (
            <div className="mb-4 p-3 bg-red-50 text-red-700 border border-red-200
rounded-md text-sm">
              <div className="flex items-center">
                <svg xmlns="http://www.w3.org/2000/svg" className="h-5 w-5 mr-2"
viewBox="0 0 20 20" fill="currentColor">
                  <path fillRule="evenodd" d="M18 10a8 8 0 11-16 0 8 8 0 0116
0zm-7 4a1 1 0 11-2 0 1 1 0 012 0zm-1-9a1 1 0 00-1 1v4a1 1 0 102 0V6a1 1 0
00-1-1z" clipRule="evenodd" />
                </svg>
                <div>
                  <p className="font-medium">Authentication Error</p>
                  <p className="text-sm">No authentication token found. Please
log out and log back in to refresh your session.</p>
                  <button
                    onClick={() => window.location.href = '/login'}
                    className="mt-2 px-3 py-1 bg-red-600 text-white text-xs
rounded hover:bg-red-700"
                  >
                    Go to Login
                  </button>
                </div>
              </div>
            </div>
          )}

          {/* Task List */}
          {loading ? (
            <div className="text-center py-8">
              <div className="inline-block animate-spin rounded-full h-8 w-8
border-4 border-blue-500 border-t-transparent"></div>
              <p className="mt-2 text-gray-600 dark:text-gray-500">Loading
tasks...</p>
            </div>
          ) : tasks.length === 0 ? (
            <div className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out p-4 sm:p-8 rounded-md shadow text-center">
              <p className="text-gray-600 dark:text-gray-500">No tasks scheduled
yet.</p>
              <p className="mt-2 text-sm text-slate-500 dark:text-gray-400">Click
"Schedule New Exam" to create your first task.</p>
```

```
            </div>
          ) : (
            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-3
sm:gap-4">
              {tasks.map((task) => (
                <div
                  key={task._id}
                  className={`bg-white dark:bg-slate-900 rounded-lg shadow p-4
border-l-4 ${
                    task.completed
                      ? "border-green-500"
                      : new Date(task.examDate) < new Date()
                      ? "border-red-500"
                      : "border-blue-500"
                  }`}
                >
                  <div className="flex justify-between items-start">
                    <h3 className="font-bold text-base sm:text-lg
mb-2">{task.title}</h3>
                    <div className="flex space-x-2">
                      <button
                        onClick={() => handleEditTask(task)}
                        className="p-1 text-blue-600 hover:text-blue-800"
                      >
                        <svg xmlns="http://www.w3.org/2000/svg" className="h-4
w-4 sm:h-5 sm:w-5" viewBox="0 0 20 20" fill="currentColor">
                          <path d="M13.586 3.586a2 2 0 112.828
2.828l-.793.793-2.828-2.828.793-.793zM11.379 5.793L3
14.172V17h2.828l8.38-8.379-2.83-2.828z" />
                        </svg>
                      </button>
                      <button
                        onClick={() => handleDeleteTask(task._id)}
                        className="p-1 text-red-600 hover:text-red-800"
                      >
                        <svg xmlns="http://www.w3.org/2000/svg" className="h-4
w-4 sm:h-5 sm:w-5" viewBox="0 0 20 20" fill="currentColor">
                          <path fillRule="evenodd" d="M9 2a1 1 0
00-.894.553L7.382 4H4a1 1 0 000 2v10a2 2 0 002 2h8a2 2 0 002-2V6a1 1 0
100-2h-3.382l-.724-1.447A1 1 0 0011 2H9zM7 8a1 1 0 012 0v6a1 1 0 11-2 0V8zm5-1a1
1 0 00-1 1v6a1 1 0 102 0V8a1 1 0 00-1-1z" clipRule="evenodd" />
                        </svg>
                      </button>
                    </div>
                  </div>

                  <p className="text-xs sm:text-sm text-gray-600 dark:text-
gray-500 mb-3">{task.description}</p>

                  <div className="mb-3">
                    <div className="flex items-center mb-1">
                      <svg xmlns="http://www.w3.org/2000/svg" className="h-3 w-3
sm:h-4 sm:w-4 text-slate-500 dark:text-gray-400 mr-2" viewBox="0 0 20 20"
fill="currentColor">
                        <path fillRule="evenodd" d="M10 9a3 3 0 100-6 3 3 0 000
6zm-7 9a7 7 0 1114 0H3z" clipRule="evenodd" />
                      </svg>
                      <p className="text-xs sm:text-sm text-slate-700 dark:text-
slate-300">{getLeadName(task.customer)}</p>
```

```
                    </div>
                    <div className="flex items-center">
                      <svg xmlns="http://www.w3.org/2000/svg" className="h-3 w-3
sm:h-4 sm:w-4 text-slate-500 dark:text-gray-400 mr-2" viewBox="0 0 20 20"
fill="currentColor">
                        <path fillRule="evenodd" d="M6 2a1 1 0 00-1 1v1H4a2 2 0
00-2 2v10a2 2 0 002 2h12a2 2 0 002-2V6a2 2 0 00-2-2h-1V3a1 1 0 10-2 0v1H7V3a1 1 0
00-1-1zm0 5a1 1 0 000 2h8a1 1 0 100-2H6z" clipRule="evenodd" />
                      </svg>
                      <p className="text-xs sm:text-sm text-slate-700 dark:text-
slate-300">{formatDateTime(task.examDate)}</p>
                    </div>
                  </div>

                  <div className="flex flex-col sm:flex-row justify-between items-
center mt-4 pt-3 border-t border-gray-100 gap-2">
                    <span className={`text-xs px-2 py-1 rounded mb-2 sm:mb-0
${task.completed ? "bg-green-100 text-green-800" : "bg-blue-100 text-blue-800"}`}>
                      {task.completed ? "Completed" : "Pending"}
                    </span>
                    <button
                      onClick={() => handleMarkCompleted(task._id, !
task.completed)}
                      className={`text-xs w-full sm:w-auto px-3 py-1 rounded-full
${
                        task.completed
                          ? "bg-gray-200 dark:bg-slate-600 text-gray-700
dark:text-gray-300 dark:text-gray-400 hover:bg-gray-300"
                          : "bg-green-100 text-green-700 hover:bg-green-200"
                      }`}
                    >
                      {task.completed ? "Mark Incomplete" : "Mark Complete"}
                    </button>
                  </div>
                </div>
              ))}
            </div>
          )}

          {/* Modal for creating/editing tasks */}
          {modalOpen && (
            <div className="fixed inset-0 bg-black bg-opacity-50 flex items-
center justify-center z-50 p-4 overflow-y-auto">
              <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-lg
dark:shadow-lg w-full max-w-md my-8 shadow-sm">
                <div className="p-4 sm:p-5 border-b border-slate-200 dark:border-
slate-700">
                  <h3 className="font-bold text-base sm:text-lg">
                    {currentTask ? "Edit Task" : "Schedule New Exam"}
                  </h3>
                </div>
                <form onSubmit={handleSubmit} className="p-4 sm:p-5">
                  <div className="mb-4">
                    <label className="block text-slate-700 dark:text-slate-300
text-xs sm:text-sm font-medium mb-2">
                      Title
                    </label>
                    <input
```

```
                      type="text"
                      name="title"
                      value={formData.title}
                      onChange={handleChange}
                      className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                      placeholder="Exam Title"
                      required
                    />
                  </div>
                  <div className="mb-4">
                    <label className="block text-slate-700 dark:text-slate-300
text-xs sm:text-sm font-medium mb-2">
                      Description
                    </label>
                    <textarea
                      name="description"
                      value={formData.description}
                      onChange={handleChange}
                      className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                      placeholder="Exam details"
                      rows="3"
                    ></textarea>
                  </div>
                  <div className="mb-4">
                    <div className="flex justify-between items-center mb-2">
                      <label className="block text-slate-700 dark:text-slate-300
text-xs sm:text-sm font-medium">
                        Customer
                      </label>
                      <div className="flex items-center">
                        <span className="text-xs text-gray-600 dark:text-gray-500
mr-2">Manual Entry</span>
                        <button
                          type="button"
                          onClick={() => setManualCustomerMode(!
manualCustomerMode)}
                          className={`relative inline-flex h-5 w-10 items-center
rounded-full transition-colors focus:outline-none ${
                            manualCustomerMode ? 'bg-blue-600' : 'bg-gray-200
dark:bg-slate-600'
                          }`}
                        >
                          <span
                            className={`inline-block h-4 w-4 transform rounded-
full bg-white dark:bg-slate-900 transition-transform ${
                              manualCustomerMode ? 'translate-x-5' : 'translate-
x-1'
                            }`}
                          />
                        </button>
                      </div>
                    </div>
```

```
                    {manualCustomerMode ? (
                      // Manual Customer Entry Form
                      <div className="space-y-3">
                        <div>
                          <input
                            type="text"
                            name="manualCustomerName"
                            value={formData.manualCustomerName}
                            onChange={handleChange}
                            className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                            placeholder="Customer Name *"
                            required
                          />
                        </div>
                        <div>
                          <input
                            type="email"
                            name="manualCustomerEmail"
                            value={formData.manualCustomerEmail}
                            onChange={handleChange}
                            className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                            placeholder="Email Address"
                          />
                        </div>
                        <div>
                          <input
                            type="tel"
                            name="manualCustomerPhone"
                            value={formData.manualCustomerPhone}
                            onChange={handleChange}
                            className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                            placeholder="Phone Number *"
                            required
                          />
                        </div>
                        <div>
                          <input
                            type="text"
                            name="manualCustomerCourse"
                            value={formData.manualCustomerCourse}
                            onChange={handleChange}
                            className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                            placeholder="Course/Exam Name *"
                            required
                          />
                        </div>
                      </div>
```

```
                    ) : (
                      // Customer Selection from Leads
                      <div>
                        {/* Search input for leads */}
                        <div className="mb-2">
                          <div className="relative">
                            <input
                              type="text"
                              placeholder="Search leads by name, email, or
course..."
                              value={leadSearchQuery}
                              onChange={(e) => setLeadSearchQuery(e.target.value)}
                              className="w-full px-3 py-2 pl-9 border border-
slate-300 dark:border-slate-600 rounded-md focus:outline-none focus:ring-2
focus:ring-blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-
blue-500 text-sm"
                            />
                            <div className="absolute inset-y-0 left-0 pl-3 flex
items-center pointer-events-none">
                              <svg xmlns="http://www.w3.org/2000/svg"
className="h-4 w-4 text-gray-400 dark:text-gray-400" fill="none" viewBox="0 0 24
24" stroke="currentColor">
                                <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M21 21l-6-6m2-5a7 7 0 11-14 0 7 7 0
0114 0z" />
                              </svg>
                            </div>
                          </div>
                        </div>

                        <select
                          name="customer"
                          value={formData.customer}
                          onChange={handleChange}
                          className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                          required
                        >
                          <option value="">Select Customer</option>

                          {/* Filter leads based on search query */}
                          {(() => {
                            const filteredLeads = leads.filter(lead => {
                              if (!leadSearchQuery) return true;

                              const searchLower = leadSearchQuery.toLowerCase();
                              const name = (lead.name || lead.NAME ||
'').toLowerCase();
                              const email = (lead.email || lead.EMAIL || lead['E-
MAIL'] || '').toLowerCase();
                              const course = (lead.course || lead.COURSE ||
'').toLowerCase();
                              const phone = (lead.contactNumber || lead.phone ||
lead.MOBILE || lead.NUMBER || '').toLowerCase();

                              return name.includes(searchLower) ||
                                     email.includes(searchLower) ||
```

```
                                    course.includes(searchLower) ||
                                    phone.includes(searchLower);
                            });

                            // Count filtered leads for each category
                            const regularLeads = filteredLeads.filter(lead => !
lead.isReferenceSale);
                            const referenceLeads = filteredLeads.filter(lead =>
lead.isReferenceSale);

                            return (
                              <>
                                {/* Group: Actual Leads from CRM */}
                                {regularLeads.length > 0 && (
                                  <optgroup label={`Leads
(${regularLeads.length})`}>
                                    {regularLeads.map(lead => (
                                      <option key={lead._id} value={lead._id}>
                                        {lead.name || lead.NAME || "Unknown"} -
{lead.course || lead.COURSE || "No course"}
                                        {lead.status ? ` (${lead.status})` : ""}
                                      </option>
                                    ))}
                                  </optgroup>
                                )}

                                {/* Group: Reference Customers */}
                                {referenceLeads.length > 0 && (
                                  <optgroup label={`Reference Customers
(${referenceLeads.length})`}>
                                    {referenceLeads.map(lead => (
                                      <option key={lead._id} value={lead._id}>
                                        {lead.name || "Unknown"} - {lead.course
|| "No course"}
                                        {lead.status ? ` (${lead.status})` : ""}
[Reference]
                                      </option>
                                    ))}
                                  </optgroup>
                                )}

                                {filteredLeads.length === 0 && (
                                  <option disabled value="">No leads match your
search</option>
                                )}
                              </>
                            );
                          })()}
                        </select>

                        {leadsLoading ? (
                          <p className="mt-1 text-xs text-slate-500 dark:text-
gray-400">Loading customers...</p>
                        ) : (
                          <p className="mt-1 text-xs text-slate-500 dark:text-
gray-400">
                            {leads.filter(lead => !lead.isReferenceSale).length}
leads +
                            {leads.filter(lead => lead.isReferenceSale).length}
```

```
reference customers available
                          {leadSearchQuery && (
                            <button
                              type="button"
                              onClick={() => setLeadSearchQuery('')}
                              className="ml-2 text-blue-600 hover:text-blue-800"
                            >
                              Clear search
                            </button>
                          )}
                        </p>
                      )}
                    </div>
                  )}
                </div>
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4 mb-4">
                  <div>
                    <label className="block text-slate-700 dark:text-slate-300
text-xs sm:text-sm font-medium mb-2">
                      Exam Date
                    </label>
                    <input
                      type="date"
                      name="examDate"
                      value={formData.examDate}
                      onChange={handleChange}
                      className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                      required
                    />
                  </div>
                  <div>
                    <label className="block text-slate-700 dark:text-slate-300
text-xs sm:text-sm font-medium mb-2">
                      Exam Time
                    </label>
                    <input
                      type="time"
                      name="examTime"
                      value={formData.examTime}
                      onChange={handleChange}
                      className="w-full px-3 py-2 border border-slate-300
dark:border-slate-600 rounded-md focus:outline-none focus:ring-2 focus:ring-
blue-500 dark:focus:border-blue-400 focus:ring-offset-2 focus:border-blue-500
text-sm"
                      required
                    />
                  </div>
                </div>
                <div className="flex flex-col sm:flex-row justify-end sm:space-
x-3 mt-6 space-y-2 sm:space-y-0">
                  <button
                    type="button"
                    onClick={() => setModalOpen(false)}
                    className="w-full sm:w-auto px-4 py-2 bg-gray-200 dark:bg-
slate-600 text-gray-800 dark:text-gray-200 rounded-md hover:bg-gray-300 text-sm
order-2 sm:order-1"
```

```
                    >
                      Cancel
                    </button>
                    <button
                      type="submit"
                      className="w-full sm:w-auto px-4 py-2 bg-blue-600 hover:bg-
blue-700 dark:bg-blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl
hover:shadow-md transition-all duration-200 text-white rounded-md text-sm order-1
sm:order-2"
                    >
                      {currentTask ? "Update Task" : "Create Task"}
                    </button>
                  </div>
                </form>
              </div>
            </div>
          )}
        </div>
      </div>
    </Layout>
  );
};

export default TaskManagementPage;
```

src/pages/TestNotificationsPage.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../context/AuthContext';
import Layout from '../components/Layout/Layout';
import { toast } from 'react-hot-toast';
import notificationService from '../services/notificationService';

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const TestNotificationsPage = () => {
  const { user } = useAuth();
  const [upcomingExams, setUpcomingExams] = useState([]);
  const [loading, setLoading] = useState(false);
  const [connectionStatus, setConnectionStatus] = useState(null);

  // Check connection status
  useEffect(() => {
    const checkStatus = () => {
      const status = notificationService.getStatus();
      setConnectionStatus(status);
    };

    checkStatus();
    const interval = setInterval(checkStatus, 2000);
    return () => clearInterval(interval);
  }, []);

  // Fetch upcoming exams
  const fetchUpcomingExams = async () => {
    try {
      setLoading(true);
      const response = await fetch('/api/test-exam/upcoming-exams');
```

```
    const data = await response.json();

    if (data.success) {
      setUpcomingExams(data.data);
    } else {
      toast.error('Failed to fetch upcoming exams');
    }
  } catch (error) {
    console.error('Error fetching upcoming exams:', error);
    toast.error('Error fetching upcoming exams');
  } finally {
    setLoading(false);
  }
};

// Create test exam
const createTestExam = async (minutesFromNow = 11) => {
  try {
    setLoading(true);
    const response = await fetch('/api/test-exam/create-exam', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        course: `React Development ${Date.now()}`,
        minutesFromNow: minutesFromNow,
        userEmail: user.email
      })
    });

    const data = await response.json();

    if (data.success) {
      toast.success(data.message);
      fetchUpcomingExams();
    } else {
      toast.error(data.message);
    }
  } catch (error) {
    console.error('Error creating test exam:', error);
    toast.error('Error creating test exam');
  } finally {
    setLoading(false);
  }
};

// Trigger notifications manually
const triggerNotifications = async () => {
  try {
    setLoading(true);
    const response = await fetch('/api/test-exam/trigger-notifications', {
      method: 'POST'
    });

    const data = await response.json();

    if (data.success) {
      toast.success('Notification check triggered!');
```

```
    } else {
      toast.error(data.message);
    }
  } catch (error) {
    console.error('Error triggering notifications:', error);
    toast.error('Error triggering notifications');
  } finally {
    setLoading(false);
  }
};

// Clean up test exams
const cleanupTestExams = async () => {
  try {
    setLoading(true);
    const response = await fetch('/api/test-exam/cleanup-test-exams', {
      method: 'DELETE'
    });

    const data = await response.json();

    if (data.success) {
      toast.success(data.message);
      fetchUpcomingExams();
    } else {
      toast.error(data.message);
    }
  } catch (error) {
    console.error('Error cleaning up test exams:', error);
    toast.error('Error cleaning up test exams');
  } finally {
    setLoading(false);
  }
};

// Test audio notification
const testAudio = () => {
  notificationService.playNotificationSound();
  toast.success('Ø=Ý  Audio test played!');
};

// Request notification permission
const requestPermission = async () => {
  const granted = await notificationService.requestNotificationPermission();
  if (granted) {
    toast.success(''  Notification permission granted!');
  } else {
    toast.error(''L Notification permission denied');
  }
};

useEffect(() => {
  fetchUpcomingExams();
}, []);

return (
  <Layout>
    <div className="container mx-auto px-4 py-8">
      <div className="max-w-4xl mx-auto">
```

```
          <h1 className="text-3xl font-bold text-gray-800 dark:text-gray-200
mb-8">Ø>Ýê Exam Notification Testing</h1>

          {/* Connection Status */}
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 mb-6 shadow-sm">
            <h2 className="text-xl font-semibold mb-4">Ø=Üá Connection Status</h2>
            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
              <div className="flex items-center space-x-2">
                <span className={`w-3 h-3 rounded-full
${connectionStatus?.isConnected ? 'bg-green-500' : 'bg-red-500'}`}></span>
                <span className="font-medium">WebSocket:</span>
                <span className={connectionStatus?.isConnected ? 'text-
green-600' : 'text-red-600'}>
                  {connectionStatus?.isConnected ? 'Connected' : 'Disconnected'}
                </span>
              </div>
              <div className="flex items-center space-x-2">
                <span className="font-medium">User ID:</span>
                <span className="text-gray-600 dark:text-
gray-500">{connectionStatus?.userId || 'Not set'}</span>
              </div>
            </div>
          </div>

          {/* Test Controls */}
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 mb-6 shadow-sm">
            <h2 className="text-xl font-semibold mb-4">Ø<ß® Test Controls</h2>
            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
              <button
                onClick={() => createTestExam(11)}
                disabled={loading}
                className="bg-blue-500 hover:bg-blue-600 hover:bg-blue-700
dark:bg-blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-xl hover:shadow-md
transition-all duration-200 text-white px-4 py-2 rounded-lg disabled:opacity-50"
              >
                Ø=ÜÝ Create Test Exam (11 min)
              </button>

              <button
                onClick={() => createTestExam(2)}
                disabled={loading}
                className="bg-orange-500 hover:bg-orange-600 text-white px-4 py-2
rounded-lg disabled:opacity-50"
              >
                &¡ Quick Test (2 min)
              </button>

              <button
                onClick={triggerNotifications}
                disabled={loading}
                className="bg-green-500 hover:bg-green-600 text-white px-4 py-2
rounded-lg disabled:opacity-50"
              >
                Ø=Ý  Trigger Notifications
              </button>
```

```
              <button
                onClick={testAudio}
                disabled={loading}
                className="bg-purple-500 hover:bg-purple-600 text-white px-4 py-2
rounded-lg disabled:opacity-50"
              >
                Ø=Ý  Test Audio
              </button>

              <button
                onClick={requestPermission}
                disabled={loading}
                className="bg-indigo-500 hover:bg-indigo-600 text-white px-4 py-2
rounded-lg disabled:opacity-50"
              >
                Ø=Ý  Request Permission
              </button>

              <button
                onClick={cleanupTestExams}
                disabled={loading}
                className="bg-red-500 hover:bg-red-600 text-white px-4 py-2
rounded-lg disabled:opacity-50"
              >
                Ø=ÝÑþ  Cleanup Test Exams
              </button>
            </div>
          </div>

          {/* Upcoming Exams */}
          <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 shadow-sm">
            <div className="flex justify-between items-center mb-4">
              <h2 className="text-xl font-semibold">Ø=ÜÅ Upcoming Exams</h2>
              <button
                onClick={fetchUpcomingExams}
                disabled={loading}
                className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out0 hover:bg-gray-600 text-white px-3 py-1 rounded text-sm
disabled:opacity-50"
              >
                Ø=Ý  Refresh
              </button>
            </div>

            {loading ? (
              <div className="flex justify-center py-8">
                <div className="animate-spin rounded-full h-8 w-8 border-b-2
border-blue-500"></div>
              </div>
            ) : upcomingExams.length === 0 ? (
              <div className="text-center py-8 text-slate-500 dark:text-gray-400">
                <p>No upcoming exams found</p>
                <p className="text-sm mt-2">Create a test exam to see
notifications in action!</p>
              </div>
            ) : (
```

```
              <div className="overflow-x-auto">
                <table className="min-w-full divide-y divide-slate-200
dark:divide-slate-700">
                  <thead className="bg-gray-50 dark:bg-slate-800 transition-all
duration-200 ease-out">
                    <tr>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Course</th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Exam Time</th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Assigned To</th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Minutes Until</th>
                      <th className="px-6 py-3 text-left text-xs font-medium text-
slate-500 dark:text-gray-400 uppercase tracking-wider">Reminder Sent</th>
                    </tr>
                  </thead>
                  <tbody className="bg-white dark:bg-slate-900 transition-all
duration-200 ease-out divide-y divide-slate-200 dark:divide-slate-700">
                    {upcomingExams.map((exam) => (
                      <tr key={exam.id} className={exam.minutesUntilExam <= 10 ?
'bg-red-50' : ''}>
                        <td className="px-6 py-4 whitespace-nowrap text-sm font-
medium text-slate-900 dark:text-slate-100">
                          {exam.course}
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          {new Date(exam.examDateTime).toLocaleString()}
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          {exam.assignedTo?.name || 'Unassigned'}
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          <span className={`px-2 py-1 rounded-full text-xs ${
                            exam.minutesUntilExam <= 10
                              ? 'bg-red-100 text-red-800'
                              : exam.minutesUntilExam <= 30
                                ? 'bg-yellow-100 text-yellow-800'
                                : 'bg-green-100 text-green-800'
                          }`}>
                            {exam.minutesUntilExam} min
                          </span>
                        </td>
                        <td className="px-6 py-4 whitespace-nowrap text-sm text-
slate-500 dark:text-gray-400">
                          <span className={`px-2 py-1 rounded-full text-xs ${
                            exam.reminderSent
                              ? 'bg-green-100 text-green-800'
                              : 'bg-gray-100 dark:bg-slate-700 text-gray-800
dark:text-gray-200'
                          }`}>
                            {exam.reminderSent ? ''  Sent' : '#ó Pending'}
                          </span>
                        </td>
                      </tr>
```

```
                  ))}
                </tbody>
              </table>
            </div>
          )}
        </div>

        {/* Instructions */}
        <div className="bg-blue-50 rounded-lg p-6 mt-6">
          <h3 className="text-lg font-semibold text-blue-800 mb-3">Ø=ÜË How to
Test</h3>
          <ol className="list-decimal list-inside space-y-2 text-blue-700">
            <li>Click "Request Permission" to enable browser notifications</li>
            <li>Create a test exam using "Create Test Exam (11 min)" - this
will trigger a notification in 1 minute</li>
            <li>For immediate testing, use "Quick Test (2 min)" and then
"Trigger Notifications"</li>
            <li>Watch for:</li>
            <ul className="list-disc list-inside ml-6 mt-1 space-y-1">
              <li>Ø=Ý  Audio beep sounds</li>
              <li>Ø<ß^ Toast notifications in the app</li>
              <li>Ø=Ý¥þ  Browser notifications (if permission granted)</li>
              <li>Ø=Üç Email notifications (check your email)</li>
            </ul>
            <li>Use "Cleanup Test Exams" to remove test data when done</li>
          </ol>
        </div>
      </div>
    </div>
  </Layout>
 );
};

export default TestNotificationsPage;
```

src/pages/TokenDebugPage.jsx

```
import React, { useState, useEffect } from 'react';
import { useAuth } from '../context/AuthContext';
import axios from 'axios';
import Layout from '../components/Layout/Layout';

import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const TokenDebugPage = () => {
  const { user } = useAuth();
  const [token, setToken] = useState('');
  const [testResult, setTestResult] = useState('');
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    // Get token from localStorage
    const savedToken = localStorage.getItem('token');
    setToken(savedToken || 'No token found');
  }, []);

  const testToken = async () => {
    setLoading(true);
```

```
    try {
      const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
      const apiUrl = isDevelopment ? 'http://localhost:8080' : 'https://crm-
backend-o36v.onrender.com/api';
      const response = await axios.get(`${apiUrl}${isDevelopment ? '/api' : ''}/
auth/debug`, {
        headers: {
          Authorization: `Bearer ${token}`
        }
      });
      setTestResult(JSON.stringify(response.data, null, 2));
    } catch (error) {
      setTestResult(JSON.stringify(error.response?.data || error.message, null,
2));
    }
    setLoading(false);
  };

  return (
    <Layout>
      <div className="container mx-auto p-6">
        <h1 className="text-2xl font-bold mb-4">Authentication Debug</h1>

        <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out p-6 rounded shadow-md dark:shadow-black/25 mb-6">
          <h2 className="text-xl font-semibold mb-2">Current User</h2>
          <pre className="bg-gray-100 dark:bg-slate-700 p-4 rounded overflow-
auto">
            {JSON.stringify(user, null, 2) || 'Not logged in'}
          </pre>
        </div>

        <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out p-6 rounded shadow-md dark:shadow-black/25 mb-6">
          <h2 className="text-xl font-semibold mb-2">Stored Token</h2>
          <pre className="bg-gray-100 dark:bg-slate-700 p-4 rounded overflow-auto
mb-4">
            {token}
          </pre>

          <button
            onClick={testToken}
            disabled={loading}
            className="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600
hover:bg-blue-700 dark:bg-blue-500 dark:hover:bg-blue-600 shadow-sm dark:shadow-
xl hover:shadow-md transition-all duration-200"
          >
            {loading ? 'Testing...' : 'Test Token'}
          </button>
        </div>

        {testResult && (
          <div className="bg-white dark:bg-slate-900 transition-all duration-200
ease-out p-6 rounded shadow-md dark:shadow-black/25">
            <h2 className="text-xl font-semibold mb-2">Test Result</h2>
            <pre className="bg-gray-100 dark:bg-slate-700 p-4 rounded overflow-
auto">
              {testResult}
```

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

export default TokenDebugPage;
```

src/pages/TutorialsPage.jsx

```
import React, { useState } from "react";
import { Link } from "react-router-dom";
import Layout from "../components/Layout/Layout";
import { useAuth } from "../context/AuthContext";
import Dashboard from "../assets/Dashboard.png";
import { professionalClasses, transitions, shadows } from '../utils/
professionalDarkMode';
const TutorialsPage = () => {
  const { user } = useAuth();
  const [activeTutorial, setActiveTutorial] = useState("getting-started");

  // Tutorial categories and their respective content
  const tutorials = {
    "getting-started": {
      title: "Getting Started with TrainCape CRM",
      icon: "Ø=Þ€",
      content: [
        {
          heading: "Welcome to TrainCape CRM",
          description: "This tutorial will help you understand the basics of the
CRM system and how to get started.",
          steps: [
            "Log in using your provided credentials",
            "Navigate the dashboard to understand available features",
            "Update your profile information",
            "Explore the main navigation menu for your role-specific features"
          ],
          image: Dashboard,
          tip: "Your role determines which features you can access. The system
has four roles: Admin, Manager, Lead Person, and Sales Person."
        },
        {
          heading: "Understanding Your Dashboard",
          description: "Your dashboard provides quick access to important
information based on your role.",
          steps: [
            "View recent activities and updates",
            "Access quick shortcuts to common tasks",
            "Check pending tasks and notifications",
            "Use the quick stats cards to monitor performance"
          ],
          tip: "Customize your experience by updating your profile settings."
        },
        {
          heading: "Setting Up Your Profile",
          description: "Personalize your account and add a profile picture.",
```

```
          steps: [
            "Navigate to your profile by clicking your name in the top
navigation",
            "Upload a profile picture by clicking the camera icon",
            "Use the crop tool to adjust and position your image",
            "Click Apply Crop to confirm your selection",
            "Save your profile changes"
          ],
          tip: "The new image cropping tool allows you to perfectly position your
profile picture before uploading."
        }
      ]
    },
    "lead-management": {
      title: "Lead Management",
      icon: "Ø=Üe",
      content: [
        {
          heading: "Creating New Leads",
          description: "Learn how to add new potential customers to the system.",
          steps: [
            "Navigate to the Leads section from the main menu",
            "Click the 'Add New Lead' button in the top right corner",
            "Fill in all required fields (marked with *)",
            "For optional fields like email, include information when available",
            "Click 'Save Lead' to create the new entry"
          ],
          tip: "Always include as much information as possible, but email is now
optional to accommodate clients who prefer phone contact."
        },
        {
          heading: "Managing Existing Leads",
          description: "How to update, filter, and organize your lead database.",
          steps: [
            "Use the search bar to quickly find leads by name, email, or phone",
            "Click on any lead to view detailed information",
            "Use the edit button to update lead information",
            "Add feedback notes to track communication history"
          ],
          tip: "Regular updates to lead information ensures everyone has the most
current data."
        },
        {
          heading: "Month-wise Lead Filtering",
          description: "The new filtering system allows you to view leads by
specific months and years for better organization.",
          steps: [
            "Navigate to the Leads page to see the filtering controls",
            "Use the Month dropdown to select any month (January through
December)",
            "Use the Year dropdown to select the desired year",
            "Click 'Show Current Month' to quickly view leads from the current
month",
            "Click 'March 2025 (Imported Leads)' to view previously imported lead
data",
            "The system shows a count of leads for each selected time period",
            "Lead assignments are preserved when filtering - assigned leads stay
with their sales persons"
          ],
```

```
          tip: "When importing new leads each month, they will be automatically
organized by their creation date. This makes it easy to track monthly lead
generation and assign new leads to sales persons without affecting previous
assignments."
        },
        {
          heading: "Assigning Leads to Sales Persons",
          description: "Proper lead assignment ensures timely follow-up.",
          steps: [
            "From the lead details page, click 'Edit'",
            "Use the 'SALE PERSON' dropdown to select the appropriate sales
person",
            "Save the changes to notify the sales person of the new assignment"
          ],
          tip: "Consider workload balance when assigning leads to ensure timely
follow-up."
        },
        {
          heading: "Importing Leads from CSV",
          description: "Bulk import leads from CSV files with automatic Lead
Person and Sales Person assignment.",
          steps: [
            "Navigate to the Admin Import page (Admin/Manager access required)",
            "Select the 'Import Leads' tab",
            "Prepare your CSV file with columns: Name, Email, Phone, Country,
Course, Date",
            "Optional columns: Lead Person, Sales Person (for automatic
assignment)",
            "For dates, use either YYYY-MM-DD (e.g., 2025-04-15) or DD-MM-YYYY
(e.g., 15-04-2025) format",
            "Upload your CSV file and click 'Import Leads'",
            "Review the import summary for successful and failed records"
          ],
          tip: "Include 'Lead Person' and 'Sales Person' columns in your CSV to
automatically assign leads during import. The system will match names and assign
leads to the correct people, regardless of who performs the import."
        }
      ]
    },
    "sales-tracking": {
      title: "Sales Tracking & Management",
      icon: "Ø=Ü°",
      content: [
        {
          heading: "Converting Leads to Sales",
          description: "Learn the process of turning qualified leads into sales
records.",
          steps: [
            "Navigate to your assigned leads list",
            "Update the lead status to 'Qualified' when appropriate",
            "Click 'Convert to Sale' to begin the sales process",
            "Fill in the required sales information including course details and
pricing",
            "Select the appropriate currency for the transaction"
          ],
          tip: "The system now supports multiple currencies for both Total Cost
and Token Amount fields."
        },
        {
```

```
          heading: "Creating Reference Sales",
          description: "Reference sales allow you to track customers who came
through referrals rather than the standard lead process.",
          steps: [
            "From the Sales page, toggle the 'Reference Sale' switch",
            "Select a Lead Person who referred this customer (optional)",
            "Fill in all customer information as this will not be pulled from an
existing lead",
            "Complete all required sales information",
            "Submit the form to create the reference sale"
          ],
          tip: "Reference sales customers will be available in all customer
selection dropdowns throughout the system, including exam scheduling."
        },
        {
          heading: "Lead Person Sales Management",
          description: "Lead persons can now track their own sales separately
from the main sales pipeline.",
          steps: [
            "As a Lead Person, navigate to the Lead Sales Update page",
            "Click 'Add New Sale' to create a new lead person sale",
            "Fill in all required customer and sales information",
            "Select a Sales Person to associate with this sale",
            "Submit the sale to save it to the lead person sales database"
          ],
          tip: "Lead Person sales are stored in a separate database and are only
visible to Lead Persons, Managers, and Admins - not to regular Sales Persons."
        },
        {
          heading: "Managing Your Sales",
          description: "Sales Persons can now fully edit and update their own
sales records.",
          steps: [
            "Navigate to the Sales Tracking page to see all your assigned sales",
            "Click the edit icon on any sale you created",
            "Update any field as needed - you now have full editing privileges
for your own sales",
            "Save your changes to update the record"
          ],
          tip: "While you can edit all aspects of your own sales, you'll still
need an Admin or Manager to delete records if necessary."
        },
        {
          heading: "Using the Sales Sheet",
          description: "The Lead Sales Sheet provides a comprehensive view of all
sales data.",
          steps: [
            "Access the Lead Sales Sheet from the navigation menu",
            "Use filters to narrow down sales by date range, sales person, or
other criteria",
            "Click on column headers to sort the data",
            "Export data to Excel for reporting purposes",
            "Use inline editing for quick updates"
          ],
          tip: "The sales sheet now includes email information and improved
contact details for better customer tracking."
        },
        {
          heading: "Managing Payments & Tracking",
```

```
          description: "Track payments and outstanding balances effectively.",
          steps: [
            "Record token (initial) payments when received",
            "Update payment status as transactions are completed",
            "Monitor pending payments through the Sales Tracking page",
            "Set follow-up tasks for payment collection"
          ],
          tip: "Keep the payment currency consistent with the original sale for
accurate accounting."
        }
      ]
    },
    "task-management": {
      title: "Task & Exam Management",
      icon: "Ø=ÜË",
      content: [
        {
          heading: "Scheduling Exams",
          description: "Use the Task Management system to schedule and track
exams for your customers.",
          steps: [
            "Navigate to the Task Management page",
            "Click 'Schedule New Exam' button",
            "Enter the exam title and description",
            "Select a customer from the dropdown (includes both leads and
reference customers)",
            "Set the exam date and time",
            "Save to create the scheduled exam"
          ],
          tip: "The customer dropdown now includes both regular leads and
reference sales customers, giving you a complete list of all possible candidates."
        },
        {
          heading: "Managing Scheduled Exams",
          description: "Track and update exam status as they progress.",
          steps: [
            "View all scheduled exams on the Task Management page",
            "Use color-coding to quickly identify exam status (pending,
completed, overdue)",
            "Click 'Mark Complete' when an exam has been conducted",
            "Edit or reschedule exams as needed using the edit icon",
            "Delete exams that are no longer needed"
          ],
          tip: "Keep the exam list up to date to ensure nothing falls through the
cracks."
        },
        {
          heading: "Exam Follow-up Process",
          description: "Proper follow-up after exams increases conversion rates.",
          steps: [
            "After marking an exam as completed, update the related lead or sale
with results",
            "Schedule any necessary follow-up tasks",
            "Document exam outcomes in the lead feedback or sales notes",
            "Update the lead status based on exam performance"
          ],
          tip: "Regular communication after exams builds trust and increases the
likelihood of enrollment."
        },
```

```
        {
          heading: "Exam Reminder Notifications & Sound Alerts",
          description: "The system includes automatic exam reminder notifications
with sound alerts to ensure you never miss an exam.",
          steps: [
            "When you schedule an exam, the system automatically sets up reminder
notifications",
            "10 minutes before the exam time, you'll receive multiple
notifications:",
            "• Browser notification (if permissions are granted)",
            "• Toast notification with exam details",
            "• Sound alert with triple beep pattern",
            "• Modal popup with exam information and direct link",
            "Click the 'Ø=Ý  Test Sound' button on the Task Management page to test
the notification sound",
            "Make sure to interact with the page (click anywhere) first to enable
sound in your browser"
          ],
          tip: "Browser sound policies require user interaction before playing
audio. The first time you visit the page, click anywhere to enable sound
notifications. The sound will play three beeps: 800Hz, 1000Hz, then 800Hz again
for maximum attention."
        }
      ]
    },
    "admin-features": {
      title: "Admin & Management Tools",
      icon: "&™Þ ",
      content: [
        {
          heading: "User Management",
          description: "Admin users can manage system access and roles.",
          steps: [
            "Navigate to Admin > Manage Users",
            "Create new user accounts with appropriate role assignments",
            "Edit existing user information or reset passwords",
            "Deactivate accounts when necessary"
          ],
          tip: "Review user activity logs periodically to ensure proper system
usage."
        },
        {
          heading: "Data Import & Export",
          description: "Efficiently move data in and out of the CRM.",
          steps: [
            "Go to Admin > Import Data",
            "Use the template provided for data formatting",
            "Upload your CSV file with lead or sales information",
            "Review and confirm the data mapping",
            "Process the import and check for any errors"
          ],
          tip: "Always make a backup export before performing large imports or
system changes."
        },
        {
          heading: "Permission Management",
          description: "Configure role-based permissions for different user
types.",
          steps: [
```

```
            "Access the Admin Dashboard",
            "Review current permission settings for each role",
            "Adjust permissions as needed for your organization's workflows",
            "Test permission changes with test accounts before rolling out"
          ],
          tip: "The system now allows Sales Persons to fully edit their own sales
while maintaining appropriate restrictions on other users' data."
        },
        {
          heading: "System Configuration",
          description: "Customize the CRM to fit your organization's needs.",
          steps: [
            "Access the Admin Dashboard",
            "Configure notification preferences",
            "Set up automated processes for lead assignment",
            "Customize sales stages and lead statuses"
          ],
          tip: "Document any configuration changes for future reference."
        }
      ]
    },
    "best-practices": {
      title: "CRM Best Practices",
      icon: "Ø<ß ",
      content: [
        {
          heading: "Data Quality Management",
          description: "Maintaining high-quality data is essential for CRM
success.",
          steps: [
            "Regularly update lead and customer information",
            "Remove duplicate records when identified",
            "Use consistent naming conventions",
            "Verify contact information periodically"
          ],
          tip: "Schedule regular data cleaning sessions to maintain database
quality."
        },
        {
          heading: "Managing Reference Customers",
          description: "Reference customers require special attention for ongoing
relationship management.",
          steps: [
            "Properly tag all reference customers when creating sales",
            "Document the source of the reference in notes",
            "Thank referrers for their recommendations",
            "Provide special attention to reference customers to encourage
additional referrals",
            "Track conversion rates from references versus other lead sources"
          ],
          tip: "Reference customers often have higher conversion rates and
retention, so track them separately in your analytics."
        },
        {
          heading: "Effective Follow-up Strategies",
          description: "Consistent follow-up improves conversion rates.",
          steps: [
            "Document all customer interactions in the CRM",
            "Set follow-up tasks with clear deadlines",
```

```
            "Use templates for common follow-up communications",
            "Analyze response patterns to optimize timing"
          ],
          tip: "The 'CLIENT REMARK' and 'FEEDBACK' fields are ideal for tracking
communication history."
        },
        {
          heading: "Reporting & Analytics",
          description: "Leverage data insights for better decision-making.",
          steps: [
            "Use the built-in reporting tools to track performance",
            "Export data for detailed analysis when needed",
            "Track conversion rates from lead to sale",
            "Monitor sales team performance metrics",
            "Compare performance between regular leads and reference customers"
          ],
          tip: "Regular reporting helps identify trends and opportunities for
improvement."
        }
      ]
    },
    "activity-tracking": {
      title: "Employee Activity Tracking",
      icon: "Ø=ÝP",
      content: [
        {
          heading: "Understanding Activity Tracking",
          description: "The CRM now automatically tracks how much time you spend
using the system each day. This helps management understand productivity and
usage patterns.",
          steps: [
            "The system starts tracking automatically when you log in",
            "Timer runs in the background while you use the CRM",
            "Time tracking pauses when you switch to other tabs or applications",
            "Timer resumes automatically when you return to the CRM",
            "Daily totals reset at midnight for accurate daily tracking"
          ],
          tip: "This system is designed to be completely automatic - you don't
need to do anything special to make it work!"
        },
        {
          heading: "What You'll See",
          description: "The activity timer is visible in your navigation bar and
shows real-time information.",
          steps: [
            "Look for the timer widget in the top navigation bar",
            "You'll see your total time for today (e.g., '2h 30m')",
            "Status indicator shows 'Active' when timer is running or 'Paused'
when stopped",
            "Manual pause/resume buttons are available if needed",
            "Time display updates in real-time as you work"
          ],
          tip: "The timer widget is always visible so you can monitor your daily
CRM usage at a glance."
        },
        {
          heading: "How to Use the System",
          description: "Follow these simple steps for accurate time tracking.",
          steps: [
```

```
            "Step 1: Login as usual - Timer starts automatically",
            "Step 2: Work normally in the CRM - Timer runs in background",
            "Step 3: When you leave CRM (switch tabs) - Timer pauses
automatically",
            "Step 4: Return to CRM - Timer resumes automatically",
            "Step 5: Logout when done - Timer stops and saves your daily total"
          ],
          tip: "The system is designed to be hands-off. Just work normally and
let it track your CRM usage automatically."
        },
        {
          heading: "What Gets Tracked",
          description: "Understanding what counts as active time versus paused
time.",
          steps: [
            "ACTIVE TIME: Working with leads, sales, reports, or any CRM
features",
            "ACTIVE TIME: Reading customer information, making updates, or data
entry",
            "PAUSED TIME: Switching to other websites, applications, or taking
breaks",
            "PAUSED TIME: Phone calls outside the CRM or lunch breaks",
            "PAUSED TIME: Computer going to sleep or being locked"
          ],
          tip: "Only actual CRM usage time is counted. The system automatically
excludes time when you're not actively using the CRM."
        },
        {
          heading: "Manual Controls (Optional)",
          description: "While the system is automatic, you can manually control
the timer if needed.",
          steps: [
            "Click the 'Pause' button in the timer widget to manually stop
tracking",
            "Click 'Resume' to restart the timer when you're ready to continue",
            "Use manual pause for extended breaks or meetings",
            "The system remembers your total time even if you refresh the page",
            "Manual controls are helpful for precise time management"
          ],
          tip: "Manual controls are optional - the automatic system handles most
situations perfectly."
        },
        {
          heading: "Privacy and What Admins See",
          description: "Understanding what information is tracked and what
remains private.",
          steps: [
            "ADMINS CAN SEE: Daily usage time for each employee (e.g., '6h 45m')",
            "ADMINS CAN SEE: Weekly and monthly activity reports and statistics",
            "ADMINS CANNOT SEE: What specific pages you visit within the CRM",
            "ADMINS CANNOT SEE: Your conversations, emails, or personal
information",
            "ADMINS CANNOT SEE: Screen recordings, screenshots, or activity
outside CRM"
          ],
          tip: "The system only tracks time spent in the CRM - your privacy is
fully protected for all other activities."
        },
        {
```

```
          heading: "Troubleshooting Common Issues",
          description: "Solutions for common activity tracking problems.",
          steps: [
            "Timer not starting? Refresh the page (F5) or log out and back in",
            "Timer stuck on 'Paused'? Click the 'Resume' button manually",
            "Time seems incorrect? Check if you switched tabs frequently (this
pauses the timer)",
            "Browser issues? Clear your browser cache and cookies",
            "Still having problems? Contact your admin or manager for assistance"
          ],
          tip: "Most issues are resolved by refreshing the page or using the
manual resume button."
        },
        {
          heading: "Best Practices for Accurate Tracking",
          description: "Tips to ensure your activity time is tracked accurately.",
          steps: [
            "DO: Keep the CRM tab active when working on CRM tasks",
            "DO: Use manual pause for long breaks or meetings",
            "DO: Log out properly when finished for the day",
            "DON'T: Leave CRM open when not actually working",
            "DON'T: Try to manipulate the system - it's designed to be fair and
accurate"
          ],
          tip: "The system is designed to track actual work time fairly. Just use
the CRM normally and let the system do its job."
        },
        {
          heading: "Admin Activity Dashboard",
          description: "For Admins and Managers: How to access and use the
Activity Dashboard.",
          steps: [
            "Navigate to your Admin Dashboard",
            "Look for the 'Employee Activity Monitoring' card",
            "Click 'Ø=ÝP Open Activity Dashboard' to access the full dashboard",
            "View today's activity, date ranges, statistics, and employee
details",
            "Use the dashboard to monitor team productivity and identify training
needs"
          ],
          tip: "The Activity Dashboard provides comprehensive insights into team
CRM usage patterns and productivity metrics."
        }
      ]
    }
  };

  // Render tutorial section based on active selection
  const renderTutorial = () => {
    const tutorial = tutorials[activeTutorial];

    return (
      <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-6 shadow-sm">
        <div className="flex items-center mb-6">
          <span className="text-3xl mr-3">{tutorial.icon}</span>
          <h2 className="text-2xl font-bold text-gray-800 dark:text-
gray-200">{tutorial.title}</h2>
```

```
        </div>

        <div className="space-y-8">
          {tutorial.content.map((section, index) => (
            <div key={index} className="border-b border-slate-200 dark:border-
slate-700 pb-6 last:border-0">
              <h3 className="text-xl font-semibold text-blue-700
mb-3">{section.heading}</h3>
              <p className="text-slate-700 dark:text-slate-300
mb-4">{section.description}</p>

              {section.steps && (
                <div className="mb-4">
                  <h4 className="font-medium text-gray-800 dark:text-gray-200
mb-2">Steps:</h4>
                  <ol className="list-decimal list-inside space-y-1 text-
slate-700 dark:text-slate-300">
                    {section.steps.map((step, stepIndex) => (
                      <li key={stepIndex} className="ml-2">{step}</li>
                    ))}
                  </ol>
                </div>
              )}

              {section.image && (
                <div className="my-4 border border-slate-200 dark:border-
slate-700 rounded-md p-2 bg-gray-50 dark:bg-slate-800 transition-all duration-200
ease-out text-center">
                  <img
                    src={section.image}
                    alt={section.heading}
                    className="max-w-full h-auto rounded-md shadow-sm mx-auto"
                    style={{ maxHeight: '400px' }}
                  />
                  <p className="text-sm text-slate-500 dark:text-gray-400
mt-2">Screenshot reference</p>
                </div>
              )}

              {section.tip && (
                <div className="bg-blue-50 border-l-4 border-blue-500 p-4 mt-4">
                  <div className="flex">
                    <div className="flex-shrink-0">
                      <svg className="h-5 w-5 text-blue-500" viewBox="0 0 20 20"
fill="currentColor">
                        <path fillRule="evenodd" d="M18 10a8 8 0 11-16 0 8 8 0
0116 0zm-7-4a1 1 0 11-2 0 1 1 0 012 0zM9 9a1 1 0 000 2v3a1 1 0 001 1h1a1 1 0
100-2v-3a1 1 0 00-1-1H9z" clipRule="evenodd" />
                      </svg>
                    </div>
                    <p className="ml-3 text-sm text-blue-700">{section.tip}</p>
                  </div>
                </div>
              )}
            </div>
          ))}
        </div>
      </div>
    );
```

```
  };

  return (
    <Layout>
      <div className="bg-gray-50 dark:bg-slate-800 transition-all duration-200
ease-out min-h-screen py-8">
        <div className="container mx-auto px-4">
          {/* Header */}
          <div className="mb-8 text-center">
            <h1 className="text-3xl font-bold text-slate-900 dark:text-slate-100
mb-2">CRM Tutorials</h1>
            <p className="text-gray-600 dark:text-gray-500 max-w-2xl mx-auto">
              Learn how to use TrainCape CRM effectively with these step-by-step
tutorials.
              Select a category below to get started.
            </p>
          </div>

          {/* Tutorial Navigation and Content */}
          <div className="flex flex-col md:flex-row gap-6">
            {/* Tutorial Navigation Sidebar */}
            <div className="md:w-1/4">
              <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out rounded-lg shadow-md
dark:shadow-2xl p-4 shadow-sm">
                <h3 className="font-bold text-lg mb-4 text-gray-800 dark:text-
gray-200">Tutorial Topics</h3>
                <nav>
                  <ul className="space-y-2">
                    {Object.keys(tutorials).map((key) => (
                      <li key={key}>
                        <button
                          onClick={() => setActiveTutorial(key)}
                          className={`w-full text-left px-3 py-2 rounded-md
transition-colors ${
                            activeTutorial === key
                              ? "bg-blue-50 text-blue-700 font-medium"
                              : "text-gray-700 dark:text-gray-300 dark:text-
gray-400 hover:bg-slate-100 dark:hover:bg-slate-700"
                            }`}
                        >
                          <span className="mr-2">{tutorials[key].icon}</span>
                          {tutorials[key].title}
                        </button>
                      </li>
                    ))}
                  </ul>
                </nav>

                {/* Role-specific tutorials notice */}
                <div className="mt-6 border-t border-slate-200 dark:border-
slate-700 pt-4">
                  <h4 className="font-medium text-gray-800 dark:text-gray-200
mb-2">Your Role</h4>
                  <p className="text-sm text-gray-600 dark:text-gray-500">
                    {user ? (
                      <>
                        You are logged in as: <span className="font-
medium">{user.role}</span>
```

```
                      <br />
                      Tutorials are tailored to features available to your role.
                    </>
                  ) : (
                    <>
                      You are not logged in. Some advanced features may require
authentication.
                    </>
                  )}
                </p>
              </div>
            </div>

            {/* Quick Help */}
            <div className="bg-gradient-to-r from-purple-500 to-indigo-600 text-
white p-5 rounded-lg shadow-md dark:shadow-black/25 mt-4">
              <h3 className="font-bold text-lg mb-2">Need Help?</h3>
              <p className="text-sm mb-3">
                Can't find what you're looking for? Contact our support team
for assistance.
              </p>
              <Link to="/support" className="inline-block bg-white dark:bg-
slate-900 transition-all duration-200 ease-out">
                Contact Support
              </Link>
            </div>
          </div>

          {/* Tutorial Content */}
          <div className="md:w-3/4">
            {renderTutorial()}

            {/* Tutorial Navigation */}
            <div className="mt-6 flex justify-between">
              <button
                onClick={() => {
                  const keys = Object.keys(tutorials);
                  const currentIndex = keys.indexOf(activeTutorial);
                  if (currentIndex > 0) {
                    setActiveTutorial(keys[currentIndex - 1]);
                  }
                }}
                disabled={Object.keys(tutorials).indexOf(activeTutorial) === 0}
                className={`px-4 py-2 flex items-center rounded-md ${
                  Object.keys(tutorials).indexOf(activeTutorial) === 0
                    ? "bg-gray-100 dark:bg-slate-700 text-gray-400 dark:text-
gray-500 dark:text-gray-400 cursor-not-allowed"
                    : "bg-blue-50 text-blue-700 hover:bg-blue-100"
                }`}
              >
                <svg className="h-5 w-5 mr-1" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                  <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M15 19l-7-7 7-7" />
                </svg>
                Previous Topic
              </button>

              <button
```

```
                  onClick={() => {
                    const keys = Object.keys(tutorials);
                    const currentIndex = keys.indexOf(activeTutorial);
                    if (currentIndex < keys.length - 1) {
                      setActiveTutorial(keys[currentIndex + 1]);
                    }
                  }}
                  disabled={Object.keys(tutorials).indexOf(activeTutorial) ===
Object.keys(tutorials).length - 1}
                  className={`px-4 py-2 flex items-center rounded-md ${
                    Object.keys(tutorials).indexOf(activeTutorial) ===
Object.keys(tutorials).length - 1
                      ? "bg-gray-100 dark:bg-slate-700 text-gray-400 dark:text-
gray-500 dark:text-gray-400 cursor-not-allowed"
                      : "bg-blue-50 text-blue-700 hover:bg-blue-100"
                  }`}
                >
                  Next Topic
                  <svg className="h-5 w-5 ml-1" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M9 5l7 7-7 7" />
                  </svg>
                </button>
              </div>
            </div>
          </div>

          {/* Additional Resources */}
          <div className="mt-12">
            <h2 className="text-2xl font-bold text-gray-800 dark:text-gray-200
mb-6 text-center">Additional Resources</h2>
            <div className="grid md:grid-cols-3 gap-6">
              <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-5 rounded-lg shadow-
md dark:shadow-2xl border-t-4 border-blue-500 shadow-sm">
                <h3 className="font-bold text-lg mb-2">Video Tutorials</h3>
                <p className="text-gray-600 dark:text-gray-500 mb-4">Watch step-
by-step demonstrations of key CRM features.</p>
                <a href="#" className="text-blue-600 hover:text-blue-800 font-
medium inline-flex items-center">
                  Watch now
                  <svg className="h-4 w-4 ml-1" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M10 6H6a2 2 0 00-2 2v10a2 2 0 002 2h10a2 2 0 002-2v-4M14 4h6m0
0v6m0-6L10 14" />
                  </svg>
                </a>
              </div>

              <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-5 rounded-lg shadow-
md dark:shadow-2xl border-t-4 border-green-500 shadow-sm">
                <h3 className="font-bold text-lg mb-2">FAQ Database</h3>
                <p className="text-gray-600 dark:text-gray-500 mb-4">Find answers
to commonly asked questions about the CRM.</p>
                <a href="#" className="text-green-600 hover:text-green-800 font-
medium inline-flex items-center">
```

```
                  View FAQs
                  <svg className="h-4 w-4 ml-1" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M10 6H6a2 2 0 00-2 2v10a2 2 0 002 2h10a2 2 0 002-2v-4M14 4h6m0
0v6m0-6L10 14" />
                  </svg>
                </a>
              </div>

              <div className="bg-white dark:bg-slate-900 border border-slate-200
dark:border-slate-700 transition-all duration-200 ease-out p-5 rounded-lg shadow-
md dark:shadow-2xl border-t-4 border-purple-500 shadow-sm">
                <h3 className="font-bold text-lg mb-2">Feature Updates</h3>
                <p className="text-gray-600 dark:text-gray-500 mb-4">Stay
informed about the latest CRM features and enhancements.</p>
                <a href="#" className="text-purple-600 hover:text-purple-800 font-
medium inline-flex items-center">
                  Read changelog
                  <svg className="h-4 w-4 ml-1" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                    <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M10 6H6a2 2 0 00-2 2v10a2 2 0 002 2h10a2 2 0 002-2v-4M14 4h6m0
0v6m0-6L10 14" />
                  </svg>
                </a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </Layout>
  );
};

export default TutorialsPage;
```

src/reportWebVitals.js

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```

src/routes/AllRoutes.jsx

```
// src/routes/AllRoutes.jsxÐ
import React from "react";Ð
import { Routes, Route } from "react-router-dom";Ð
```

```
import HomePage from "../pages/HomePage";Đ
import Login from "../components/Auth/Login";Đ
import SignUp from "../components/Auth/SignUp";Đ
import CustomerSignUp from "../components/Auth/CustomerSignUp";Đ
import LeadsPage from "../pages/LeadsPage";Đ
import SalesPage from "../pages/SalesPage";Đ
import SalesTrackingPage from "../pages/SalesTrackingPage";Đ
import SalesCreatePage from "../pages/SalesCreatePage";Đ
import ProfilePage from "../pages/ProfilePage";Đ
import TokenDebugPage from "../pages/TokenDebugPage";Đ
import AdminDashboardPage from "../pages/AdminDashboardPage";Đ
import AdminUsersPage from "../pages/AdminUsersPage";Đ
import AdminLeadsPage from "../pages/AdminLeadsPage";Đ
import AdminImportPage from "../pages/AdminImportPage";Đ
import AdminReportsPage from "../pages/AdminReportsPage";Đ
import AdminActivityPage from "../pages/AdminActivityPage";Đ
import ProtectedRoute from "./ProtectedRoute";Đ
import ForgotPassword from "../components/Auth/ForgotPassword";Đ
import LeadSalesSheet from '../pages/LeadSalesSheet';Đ
import LeadSalesUpdatePage from '../pages/LeadSalesUpdatePage';Đ
import TutorialsPage from '../pages/TutorialsPage';Đ
import TaskManagementPage from '../pages/TaskManagementPage';Đ
import RepeatCustomersPage from '../pages/RepeatCustomersPage';Đ
import CountriesPage from '../pages/Countries';Đ
import ManagementContactsPage from '../pages/ManagementContactsPage';Đ
import ManagerDashboard from '../pages/ManagerDashboard';Đ
import TestNotificationsPage from '../pages/TestNotificationsPage';Đ
import CustomerDashboard from "../pages/CustomerDashboard";Đ
import ProspectsPage from '../pages/ProspectsPage';Đ
import EmployeeManagementPage from '../pages/EmployeeManagementPage';Đ
import DocumentManagementPage from '../pages/DocumentManagementPage';Đ
import AdminActivityLogsPage from '../pages/AdminActivityLogsPage';Đ
Đ
// Removed Router wrapper so it can be used at a higher levelĐ
const AllRoutes = () => {Đ
  return (Đ
    <Routes>Đ
      {/* Public Routes */}Đ
      <Route path="/" element={<HomePage />} />Đ
      <Route path="/login" element={<Login />} />Đ
      {/* <Route path="/signup" element={<SignUp />} /> */}Đ
      <Route path="/customer-signup" element={<CustomerSignUp />} />Đ
      <Route path="/debug" element={<TokenDebugPage />} />Đ
      <Route path="/forgot-password" element={<ForgotPassword />} />Đ
      <Route path="/tutorials" element={<TutorialsPage />} />Đ
      <Route path="/management-contacts" element={<ManagementContactsPage />} />Đ
      <Route path="/countries" element={<CountriesPage />} />Đ
      Đ
      {/* Customer Dashboard */}Đ
      <RouteĐ
        path="/customer"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Customer"]}>Đ
            <CustomerDashboard />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
Đ
      {/* Test Notifications - for testing exam notifications */}Đ
```

```
      <RouteĐ
        path="/test-notifications"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Sales Person", "Lead Person",
"Manager", "Admin"]}>Đ
            <TestNotificationsPage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
Đ
      {/* Protected Routes */}Đ
      <RouteĐ
        path="/leads"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Sales Person", "Lead Person",
"Manager", "Admin"]}>Đ
            <LeadsPage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      {/* Sales Routes */}Đ
      <RouteĐ
        path="/sales"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Sales Person", "Manager", "Admin"]}>Đ
            <SalesPage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      {/* Sales Tracking Route */}Đ
      <RouteĐ
        path="/sales-tracking"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Sales Person", "Manager", "Admin"]}>Đ
            <SalesTrackingPage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      {/* Sales Create Route - for sales persons to create sales for lead persons
*/}Đ
      <RouteĐ
        path="/create-sale"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Sales Person", "Manager", "Admin"]}>Đ
            <SalesCreatePage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      {/* Task Management Route */}Đ
      <RouteĐ
        path="/tasks"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Sales Person", "Lead Person",
"Manager", "Admin"]}>Đ
            <TaskManagementPage />Đ
```

```
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      {/* Profile Route - accessible to all authenticated users including
customers */}Đ
      <RouteĐ
        path="/profile"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Sales Person", "Lead Person",
"Manager", "Admin", "Customer", "HR", "Employee"]}>Đ
            <ProfilePage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      {/* Admin Routes - Allow Lead Person to access dashboard for lead stage
analytics */}Đ
      <RouteĐ
        path="/admin"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Admin", "Lead Person"]}>Đ
            <AdminDashboardPage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      <RouteĐ
        path="/admin/users"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Admin"]}>Đ
            <AdminUsersPage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      <RouteĐ
        path="/admin/leads"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Admin"]}>Đ
            <AdminLeadsPage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
Đ
      <RouteĐ
        path="/admin/import"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Admin"]}>Đ
            <AdminImportPage />Đ
          </ProtectedRoute>Đ
        }Đ
      />Đ
      Đ
      <RouteĐ
        path="/admin/reports"Đ
        element={Đ
          <ProtectedRoute allowedRoles={["Admin"]}>Đ
            <AdminReportsPage />Đ
```

```
    </ProtectedRoute>Đ
  }Đ
/>Đ
Đ
{/* Admin Activity Page - accessible to Admin and Manager */}Đ
<RouteĐ
  path="/admin/activity"Đ
  element={Đ
    <ProtectedRoute allowedRoles={["Admin", "Manager"]}>Đ
      <AdminActivityPage />Đ
    </ProtectedRoute>Đ
  }Đ
/>Đ
Đ
{/* Admin Activity Logs Page - accessible to Admin */}Đ
<RouteĐ
  path="/admin/activity-logs"Đ
  element={Đ
    <ProtectedRoute allowedRoles={["Admin"]}>Đ
      <AdminActivityLogsPage />Đ
    </ProtectedRoute>Đ
  }Đ
/>Đ
Đ
{/* Repeat Customers Page - accessible to Admin and Manager */}Đ
<RouteĐ
  path="/repeat-customers"Đ
  element={Đ
    <ProtectedRoute allowedRoles={["Admin", "Manager"]}>Đ
      <RepeatCustomersPage />Đ
    </ProtectedRoute>Đ
  }Đ
/>Đ
Đ
{/* Lead Sales Sheet - accessible to Lead Person, Manager, and Admin */}Đ
<Route Đ
  path="/lead-sales-sheet" Đ
  element={Đ
    <ProtectedRoute allowedRoles={['Lead Person', 'Manager', 'Admin']}>Đ
      <LeadSalesSheet />Đ
    </ProtectedRoute>Đ
  } Đ
/>Đ
Đ
{/* Lead Sales Update - for lead persons to update sales */}Đ
<Route Đ
  path="/update-sales" Đ
  element={Đ
    <ProtectedRoute allowedRoles={['Lead Person', 'Manager', 'Admin']}>Đ
      <LeadSalesUpdatePage />Đ
    </ProtectedRoute>Đ
  } Đ
/>Đ
Đ
{/* Contact Management Page - accessible to all authenticated users */}Đ
{/* Disabled since we're using ManagementContactsPage insteadĐ
<RouteĐ
  path="/contact-management"Đ
  element={Đ
```

```
          <ProtectedRoute allowedRoles={["Sales Person", "Lead Person",
"Manager", "Admin"]}>Ð
            <ContactManagementPage />Ð
          </ProtectedRoute>Ð
        }Ð
      />Ð
      */}Ð
      Ð
      {/* Prospects - Sales Person, Manager, Admin only */}Ð
      <Route Ð
        path="/prospects" Ð
        element={Ð
          <ProtectedRoute allowedRoles={['Sales Person', 'Manager', 'Admin']}>Ð
            <ProspectsPage />Ð
          </ProtectedRoute>Ð
        } Ð
      />Ð
      Ð
      {/* Employee Management - HR, Manager, Admin only */}Ð
      <Route Ð
        path="/employees" Ð
        element={Ð
          <ProtectedRoute allowedRoles={['HR', 'Manager', 'Admin']}>Ð
            <EmployeeManagementPage />Ð
          </ProtectedRoute>Ð
        } Ð
      />Ð
      Ð
      {/* Document Management - All authenticated users */}Ð
      <Route Ð
        path="/documents" Ð
        element={Ð
          <ProtectedRoute allowedRoles={['Sales Person', 'Lead Person',
'Manager', 'Admin', 'HR', 'Employee']}>Ð
            <DocumentManagementPage />Ð
          </ProtectedRoute>Ð
        } Ð
      />Ð
      Ð
      {/* Catch-all route for 404 errors */}Ð
      <Route path="*" element={<HomePage />} />Ð
    </Routes>Ð
  );Ð
};Ð
Ð
export default AllRoutes;Ð
```

src/routes/ProtectedRoute.jsx

```
// src/routes/ProtectedRoute.jsxÐ
import React from "react";Ð
import { Navigate } from "react-router-dom";Ð
import { useAuth } from "../context/AuthContext";Ð
Ð
const ProtectedRoute = ({ children, allowedRoles }) => {Ð
  const { user, loading } = useAuth();Ð
  Ð
  // Show loading spinner or similar while checking auth stateÐ
```

```
  if (loading) {Đ
    return (Đ
      <div className="min-h-screen flex items-center justify-center">Đ
        <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2
border-blue-500"></div>Đ
      </div>Đ
    );Đ
  }Đ
  Đ
  // If not authenticated, redirect to loginĐ
  if (!user) {Đ
    return <Navigate to="/login" replace />;Đ
  }Đ
  Đ
  // If authenticated but doesn't have required role, redirect to homeĐ
  if (!allowedRoles.includes(user.role)) {Đ
    return <Navigate to="/" replace />;Đ
  }Đ
  Đ
  return children;Đ
};Đ
Đ
export default ProtectedRoute;
```

src/services/api.js

```
import axios from 'axios';

// Determine if we're in development mode
const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';

// API URL configuration - use localhost for development, your Hostinger backend
for production
const API_URL = isDevelopment
  ? 'http://localhost:8080/api'
  : (import.meta.env.VITE_API_URL || 'https://crm-backend-o36v.onrender.com/api');

// Create axios instance
const api = axios.create({
  baseURL: API_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

// Request interceptor to add auth token
api.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem('token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }

    return config;
  },
  (error) => {
    return Promise.reject(error);
```

```
  }
);

// Response interceptor to handle errors
api.interceptors.response.use(
  (response) => {
    return response;
  },
  (error) => {
    if (error.response?.status === 401) {
      // Token might be expired or invalid
      const token = localStorage.getItem('token');
      if (token) {
        // Clear invalid token
        localStorage.removeItem('token');
        // Optionally redirect to login
        window.location.href = '/login';
      }
    }
    return Promise.reject(error);
  }
);

// Auth API endpoints
export const authAPI = {
  login: (credentials) => api.post('/auth/login', credentials),
  register: (userData) => api.post('/auth/register', userData),
  getProfile: () => api.get('/auth/me'),
  updateProfile: (userData) => api.put('/auth/me', userData),
  updateProfilePicture: (formData) => api.put('/auth/profile-picture', formData, {
    headers: {
      'Content-Type': 'multipart/form-data',
    },
  }),
  getUsers: (role) => api.get(`/auth/users${role ? `?role=${role}` : ''}`),
  createUser: (userData) => api.post('/auth/users', userData),
  updateUser: (userId, userData) => api.put(`/auth/users/${userId}`, userData),
  deleteUser: (userId) => api.delete(`/auth/users/${userId}`),
  forgotPassword: (email) => api.post('/auth/forgot-password', { email }),
  resetPassword: (token, password) => api.post('/auth/reset-password', { token,
password })
};

// Leads API
export const leadsAPI = {
  getAll: (filters = {}) => {
    const params = new URLSearchParams();

    // Add date filters to params
    if (filters.month) params.append('month', filters.month);
    if (filters.year) params.append('year', filters.year);
    if (filters.startDate) params.append('startDate', filters.startDate);
    if (filters.endDate) params.append('endDate', filters.endDate);

    const queryString = params.toString();
    const url = queryString ? `/leads?${queryString}` : '/leads';

    return api.get(url);
  },
```

```
  getById: (id) => api.get(`/leads/${id}`),
  create: (leadData) => api.post('/leads', leadData),
  update: (id, leadData) => api.put(`/leads/${id}`, leadData),
  updateFeedback: (id, feedback) => api.put(`/leads/${id}/feedback`,
{ feedback }),
  delete: (id) => api.delete(`/leads/${id}`),
  getAssigned: (filters = {}) => {
    const params = new URLSearchParams();

    // Add date filters to params
    if (filters.month) params.append('month', filters.month);
    if (filters.year) params.append('year', filters.year);
    if (filters.startDate) params.append('startDate', filters.startDate);
    if (filters.endDate) params.append('endDate', filters.endDate);

    const queryString = params.toString();
    const url = queryString ? `/leads/assigned?${queryString}` : '/leads/
assigned';

    return api.get(url);
  },
  getRepeatCustomers: () => api.get('/leads/repeat-customers'),
  import: (leadsData) => api.post('/leads/import', { leads: leadsData }),
  importLeads: (leadsData) => api.post('/leads/import', { leads: leadsData }),
  importCSV: (formData) => api.post('/leads/import-csv', formData, {
    headers: {
      'Content-Type': 'multipart/form-data',
    },
  }),
};

// Sales API
export const salesAPI = {
  getAll: async () => {
    try {
      console.log('Fetching all sales...');
      const response = await api.get('/sales');
      console.log('Sales API response:', response.data);
      return response;
    } catch (error) {
      console.error('Error fetching sales:', error.response?.data ||
error.message);
      throw error;
    }
  },
  getAllForced: async () => {
    try {
      console.log('Fetching all sales (forced)...');
      const response = await api.get('/sales?full=true&nocache=' + new
Date().getTime());
      console.log('Forced sales API response:', response.data);
      return response;
    } catch (error) {
      console.error('Error fetching sales (forced):', error.response?.data ||
error.message);
      throw error;
    }
  },
  getById: (id) => api.get(`/sales/${id}`),
```

```
  create: (saleData) => {
    // Set isLeadPersonSale to true if leadPerson is specified
    if (saleData.leadPerson) {
      saleData.isLeadPersonSale = true;
    }
    return api.post('/sales', saleData);
  },
  update: (id, saleData) => api.put(`/sales/${id}`, saleData),
  delete: (id) => api.delete(`/sales/${id}`),
  getCount: () => api.get('/sales/count'),
  import: (salesData) => api.post('/sales/import', { sales: salesData }),
  importSales: (salesData) => api.post('/sales/import', { sales: salesData }),

  // Reports API
  getCourseAnalysis: (period = 'monthly') => api.get(`/sales/reports/course-
analysis?period=${period}`),
  getRevenueAnalysis: (period = '1month') => api.get(`/sales/reports/revenue-
analysis?period=${period}`),
  getTopCourses: (period = 'all', limit = 10) => api.get(`/sales/reports/top-
courses?period=${period}&limit=${limit}`),
  getStatusAnalysis: (period = '1month', status = null) => {
    const url = status
      ? `/sales/reports/status-analysis?period=${period}&status=${status}`
      : `/sales/reports/status-analysis?period=${period}`;
    return api.get(url);
  },

  // Lead sales specific endpoints
  getLeadSales: (filters = {}) => {
    const params = new URLSearchParams();

    // Add filters to params
    Object.keys(filters).forEach(key => {
      if (filters[key] !== undefined && filters[key] !== null && filters[key] !==
'') {
        params.append(key, filters[key]);
      }
    });

    const queryString = params.toString();
    const url = queryString ? `/lead-sales?${queryString}` : '/lead-sales';

    return api.get(url);
  },

  createLeadSale: (saleData, leadPersonId) => {
    return api.post('/lead-sales', { ...saleData, leadPerson: leadPersonId });
  }
};

// Lead Person Sales API
export const leadPersonSalesAPI = {
  getAll: () => api.get('/lead-person-sales'),
  getById: (id) => api.get(`/lead-person-sales/${id}`),
  create: (saleData) => api.post('/lead-person-sales', saleData),
  update: (id, saleData) => api.put(`/lead-person-sales/${id}`, saleData),
  delete: (id) => api.delete(`/lead-person-sales/${id}`),
};
```

```
// Tasks API
export const tasksAPI = {
  getAll: () => api.get('/tasks'),
  getById: (id) => api.get(`/tasks/${id}`),
  create: (taskData) => api.post('/tasks', taskData),
  update: (id, taskData) => api.put(`/tasks/${id}`, taskData),
  delete: (id) => api.delete(`/tasks/${id}`),
  markCompleted: (id, completed) => api.put(`/tasks/${id}`, { completed }),
};

// Currency API
export const currencyAPI = {
  getRates: () => api.get('/currency/rates'),
  getRate: (from, to) => api.get(`/currency/rate?from=${from}&to=${to}`),
};

// Prospects API
export const prospectsAPI = {
  getAll: (filters = {}) => {
    const params = new URLSearchParams();

    // Add filters to params
    Object.keys(filters).forEach(key => {
      if (filters[key] !== undefined && filters[key] !== null && filters[key] !==
'') {
        params.append(key, filters[key]);
      }
    });

    const queryString = params.toString();
    const url = queryString ? `/prospects?${queryString}` : '/prospects';

    return api.get(url);
  },
  getById: (id) => api.get(`/prospects/${id}`),
  create: (prospectData) => api.post('/prospects', prospectData),
  update: (id, prospectData) => api.put(`/prospects/${id}`, prospectData),
  delete: (id) => api.delete(`/prospects/${id}`),
  getStats: () => api.get('/prospects/stats'),
  convertToLead: (id) => api.post(`/prospects/${id}/convert`),
};

// Activity API
export const activityAPI = {
  startSession: () => api.post('/activity/start-session'),
  endSession: (duration) => api.post('/activity/end-session', { duration }),
  trackActivity: (duration, isActive = true) => api.post('/activity/track',
{ duration, isActive }),
  getMyActivity: (date = null, startDate = null, endDate = null) => {
    const params = new URLSearchParams();
    if (date) params.append('date', date);
    if (startDate) params.append('startDate', startDate);
    if (endDate) params.append('endDate', endDate);

    const queryString = params.toString();
    const url = queryString ? `/activity/my-activity?${queryString}` : '/activity/
my-activity';

    return api.get(url);
```

```
  },
  getAllUsersActivity: (date = null, startDate = null, endDate = null) => {
    const params = new URLSearchParams();
    if (date) params.append('date', date);
    if (startDate) params.append('startDate', startDate);
    if (endDate) params.append('endDate', endDate);

    const queryString = params.toString();
    const url = queryString ? `/activity/all-users?${queryString}` : '/activity/
all-users';

    return api.get(url);
  },
  getStatistics: (days = 7) => api.get(`/activity/statistics?days=${days}`),
};

// Payroll API
export const payrollAPI = {
  getAll: (filters = {}) => {
    const params = new URLSearchParams();

    // Add filters to params
    Object.keys(filters).forEach(key => {
      if (filters[key] !== undefined && filters[key] !== null && filters[key] !==
'') {
        params.append(key, filters[key]);
      }
    });

    const queryString = params.toString();
    const url = queryString ? `/payroll?${queryString}` : '/payroll';

    return api.get(url);
  },
  getById: (id) => api.get(`/payroll/${id}`),
  generate: (payrollData) => api.post('/payroll/generate', payrollData),
  update: (id, payrollData) => api.put(`/payroll/${id}`, payrollData),
  delete: (id) => api.delete(`/payroll/${id}`),
  approve: (id) => api.put(`/payroll/${id}/approve`),
  generateSalarySlip: (id) => api.get(`/payroll/${id}/salary-slip`, {
    responseType: 'blob'
  }),
  downloadSalarySlip: (id) => api.get(`/payroll/${id}/download`, {
    responseType: 'blob'
  }),
};

export default api;
```

src/services/employeeAPI.js

```
import api from './api';

const employeeAPI = {
  // Employee CRUD operations
  getAll: () => api.get('/employees'),
  getById: (id) => api.get(`/employees/${id}`),
  create: (formData) => api.post('/employees', formData, {
```

```
    headers: {
      'Content-Type': 'multipart/form-data',
    },
  }),
  update: (id, formData) => api.put(`/employees/${id}`, formData, {
    headers: {
      'Content-Type': 'multipart/form-data',
    },
  }),
  delete: (id) => api.delete(`/employees/${id}`),

  // Department operations
  getDepartments: () => api.get('/employees/departments'),
  createDepartment: (data) => api.post('/employees/departments', data),

  // Role operations
  getRoles: () => api.get('/employees/roles'),
  createRole: (data) => api.post('/employees/roles', data),

  // Conversion operations
  convertUsers: () => api.post('/employees/convert-users'),
};

export default employeeAPI;
```

src/services/leaveAPI.js

```
import api from './api';

const leaveAPI = {
  // Apply for leave
  applyLeave: (leaveData) => api.post('/leaves', leaveData),

  // Get my leaves
  getMyLeaves: (params = {}) => {
    const query = new URLSearchParams(params);
    return api.get(`/leaves/my-leaves?${query}`);
  },

  // Get all leaves (for managers/admins)
  getAllLeaves: (params = {}) => {
    const query = new URLSearchParams(params);
    return api.get(`/leaves?${query}`);
  },

  // Update leave status (approve/reject)
  updateLeaveStatus: (leaveId, statusData) =>
    api.put(`/leaves/${leaveId}/status`, statusData),

  // Cancel leave
  cancelLeave: (leaveId) => api.put(`/leaves/${leaveId}/cancel`),

  // Get leave statistics
  getLeaveStats: (year) => {
    const params = year ? `?year=${year}` : '';
    return api.get(`/leaves/stats${params}`);
  },
```

```
  // Get leave balance
  getLeaveBalance: (year) => {
    const params = year ? `?year=${year}` : '';
    return api.get(`/leaves/balance${params}`);
  }
};

export default leaveAPI;
```

src/services/loggingService.js

```
import api from './api';

class LoggingService {
  static async createLog(action, details, affectedResource, resourceId = null,
status = 'SUCCESS') {
    try {
      const response = await api.post('/logs', {
        action,
        details,
        affectedResource,
        resourceId,
        status
      });
      return response.data;
    } catch (error) {
      console.error('Error creating log:', error);
      throw error;
    }
  }

  static async logLogin(userId, success) {
    return this.createLog(
      'LOGIN',
      {
        userId,
        success
      },
      'USER',
      userId,
      success ? 'SUCCESS' : 'FAILURE'
    );
  }

  static async logLogout(userId) {
    return this.createLog(
      'LOGOUT',
      {
        userId
      },
      'USER',
      userId
    );
  }

  static async logSaleCreate(saleData) {
    return this.createLog(
      'SALE_CREATE',
```

```
    {
      customerName: saleData.customerName,
      amount: saleData.amount,
      product: saleData.product,
      salesPerson: saleData.salesPerson
    },
    'SALE',
    saleData._id
  );
}

static async logSaleUpdate(saleId, changes) {
  return this.createLog(
    'SALE_UPDATE',
    {
      saleId,
      changes
    },
    'SALE',
    saleId
  );
}

// New methods for lead logging
static async logLeadCreate(leadData) {
  return this.createLog(
    'LEAD_CREATE',
    {
      name: leadData.name || leadData.NAME,
      email: leadData.email || leadData['E-MAIL'],
      course: leadData.course || leadData.COURSE,
      assignedTo: leadData.assignedTo || leadData['SALE PERSON'],
      leadPerson: leadData.leadPerson || leadData['LEAD PERSON']
    },
    'LEAD',
    leadData._id
  );
}

static async logLeadUpdate(leadId, changes) {
  return this.createLog(
    'LEAD_UPDATE',
    {
      leadId,
      changes
    },
    'LEAD',
    leadId
  );
}

static async logLeadDelete(leadId, leadData) {
  return this.createLog(
    'LEAD_DELETE',
    {
      leadId,
      leadData
    },
    'LEAD',
```

```
    leadId
  );
}

static async logLeadAssign(leadId, assignedTo, previousAssignee) {
  return this.createLog(
    'LEAD_ASSIGN',
    {
      leadId,
      assignedTo,
      previousAssignee
    },
    'LEAD',
    leadId
  );
}

static async logEmployeeCreate(employeeData) {
  return this.createLog(
    'EMPLOYEE_CREATE',
    {
      employeeName: employeeData.fullName,
      role: employeeData.role,
      department: employeeData.department
    },
    'EMPLOYEE',
    employeeData._id
  );
}

static async logEmployeeUpdate(employeeId, changes) {
  return this.createLog(
    'EMPLOYEE_UPDATE',
    {
      employeeId,
      changes
    },
    'EMPLOYEE',
    employeeId
  );
}

static async logAttendanceMark(employeeId, type, time) {
  return this.createLog(
    'ATTENDANCE_MARK',
    {
      employeeId,
      type, // 'IN' or 'OUT'
      time
    },
    'ATTENDANCE',
    employeeId
  );
}

static async logLeaveRequest(employeeId, leaveData) {
  return this.createLog(
    'LEAVE_REQUEST',
    {
```

```
        employeeId,
        leaveType: leaveData.type,
        startDate: leaveData.startDate,
        endDate: leaveData.endDate,
        reason: leaveData.reason
      },
      'LEAVE',
      leaveData._id
    );
  }

  static async logLeaveUpdate(leaveId, status, updatedBy) {
    return this.createLog(
      'LEAVE_UPDATE',
      {
        leaveId,
        status,
        updatedBy
      },
      'LEAVE',
      leaveId
    );
  }

  static async logPayrollUpdate(employeeId, payrollData) {
    return this.createLog(
      'PAYROLL_UPDATE',
      {
        employeeId,
        month: payrollData.month,
        year: payrollData.year,
        amount: payrollData.amount
      },
      'PAYROLL',
      employeeId
    );
  }

  static async logSettingsUpdate(settingType, changes) {
    return this.createLog(
      'SETTINGS_UPDATE',
      {
        settingType,
        changes
      },
      'SETTINGS'
    );
  }
}

export default LoggingService;
```

src/services/notificationService.js

```
import { io } from 'socket.io-client';
import { toast } from 'react-hot-toast';
import NotificationSounds from '../assets/sounds/notification-sounds.js';
```

```
class NotificationService {
  constructor() {
    this.socket = null;
    this.isConnected = false;
    this.userId = null;
    this.audioContext = null;
    this.notificationSound = null;
    this.sounds = NotificationSounds;
    this.isAudioEnabled = true;
    this.soundPreferences = {
      messageSound: 'message', // 'message', 'group', 'soft', 'urgent', 'none'
      volume: 0.3,
      enableSounds: true
    };
    this.initializeAudio();
    this.loadPreferences();
  }

  // Load user preferences from localStorage
  loadPreferences() {
    try {
      const saved = localStorage.getItem('notificationPreferences');
      if (saved) {
        this.soundPreferences =
{ ...this.soundPreferences, ...JSON.parse(saved) };
      }
    } catch (error) {
      console.warn('Error loading notification preferences:', error);
    }
  }

  // Save user preferences to localStorage
  savePreferences() {
    try {
      localStorage.setItem('notificationPreferences',
JSON.stringify(this.soundPreferences));
    } catch (error) {
      console.warn('Error saving notification preferences:', error);
    }
  }

  // Update sound preferences
  updatePreferences(newPreferences) {
    this.soundPreferences = { ...this.soundPreferences, ...newPreferences };
    this.savePreferences();
  }

  // Initialize audio context and load notification sound
  initializeAudio() {
    try {
      // Create audio context
      this.audioContext = new (window.AudioContext || window.webkitAudioContext)
();

      // Create notification sound using Web Audio API
      this.createNotificationSound();
    } catch (error) {
      console.warn('Audio context not supported:', error);
    }
```

```
  }

  // Create a beep sound using Web Audio API (legacy support)
  createNotificationSound() {
    if (!this.audioContext) return;

    const createBeep = (frequency = 800, duration = 200) => {
      const oscillator = this.audioContext.createOscillator();
      const gainNode = this.audioContext.createGain();

      oscillator.connect(gainNode);
      gainNode.connect(this.audioContext.destination);

      oscillator.frequency.value = frequency;
      oscillator.type = 'sine';

      gainNode.gain.setValueAtTime(0.3, this.audioContext.currentTime);
      gainNode.gain.exponentialRampToValueAtTime(0.01,
this.audioContext.currentTime + duration / 1000);

      oscillator.start(this.audioContext.currentTime);
      oscillator.stop(this.audioContext.currentTime + duration / 1000);
    };

    this.notificationSound = createBeep;
  }

  // Play notification sound based on type
  async playNotificationSound(type = 'message') {
    if (!this.soundPreferences.enableSounds || !this.isAudioEnabled) {
      return;
    }

    try {
      switch (type) {
        case 'message':
          await this.sounds.playMessageSound();
          break;
        case 'group':
          await this.sounds.playGroupMessageSound();
          break;
        case 'urgent':
          await this.sounds.playUrgentSound();
          break;
        case 'soft':
          await this.sounds.playSoftSound();
          break;
        case 'success':
          await this.sounds.playSuccessSound();
          break;
        case 'error':
          await this.sounds.playErrorSound();
          break;
        default:
          await this.sounds.playMessageSound();
      }
    } catch (error) {
      console.error('Error playing notification sound:', error);
```

```
      // Fallback to legacy sound
      this.playLegacySound();
    }
  }

  // Legacy sound support (fallback)
  playLegacySound() {
    try {
      if (this.audioContext && this.notificationSound) {
        // Resume audio context if suspended
        if (this.audioContext.state === 'suspended') {
          this.audioContext.resume();
        }

        // Play multiple beeps for urgency
        this.notificationSound(800, 300); // First beep
        setTimeout(() => this.notificationSound(1000, 300), 400); // Second beep
        setTimeout(() => this.notificationSound(800, 300), 800); // Third beep
      } else {
        // Fallback: try to play a system beep
        console.log('\u0007'); // Bell character
      }
    } catch (error) {
      console.error('Error playing legacy notification sound:', error);
    }
  }

  // Connect to WebSocket server
  connect(userId) {
    if (this.isConnected && this.userId === userId) {
      return; // Already connected for this user
    }

    this.userId = userId;

    // Determine server URL based on environment
    const isDevelopment = import.meta.env.DEV && import.meta.env.MODE !==
'production';
    const apiUrl = isDevelopment ? 'http://localhost:8080/api' :
(import.meta.env?.VITE_API_URL || 'https://crm-backend-o36v.onrender.com/api');
    const serverUrl = apiUrl.replace('/api', ''); // Remove /api for socket
connection

    console.log('Ø=Ý  Connecting to notification server:', serverUrl);

    this.socket = io(serverUrl, {
      transports: ['websocket', 'polling'],
      timeout: 20000,
      forceNew: true
    });

    this.socket.on('connect', () => {
      console.log('  Connected to notification server');
      this.isConnected = true;

      // Join user's personal room
      this.socket.emit('join-user-room', userId);

      toast.success('Ø=Ý  Notifications enabled');
```

```
    });

    this.socket.on('disconnect', () => {
      console.log(''L Disconnected from notification server');
      this.isConnected = false;
    });

    this.socket.on('connect_error', (error) => {
      console.error('Connection error:', error);
      toast.error('Failed to connect to notification server');
    });

    // Listen for chat message notifications
    this.socket.on('messageNotification', (notification) => {
      this.handleChatMessageNotification(notification);
    });

    // Listen for exam reminders
    this.socket.on('exam-reminder', (notification) => {
      this.handleExamReminder(notification);
    });

    // Listen for other notification types
    this.socket.on('notification', (notification) => {
      this.handleGeneralNotification(notification);
    });

    // Listen for user status updates
    this.socket.on('userStatusUpdate', (statusUpdate) => {
      this.handleUserStatusUpdate(statusUpdate);
    });
  }

  // Handle chat message notifications
  handleChatMessageNotification(notification) {
    console.log('Ø=Ü¬ Chat message notification received:', notification);

    // Determine sound type based on message context
    let soundType = 'message';
    if (notification.isGuest) {
      soundType = 'urgent'; // Guest messages are more urgent
    } else if (notification.isGroup) {
      soundType = 'group'; // Group messages
    }

    // Play notification sound
    this.playNotificationSound(soundType);

    // Show browser notification if permission granted
    this.showBrowserNotification({
      title: `New message from ${notification.senderName}`,
      body: notification.content.substring(0, 100) + (notification.content.length
> 100 ? '...' : ''),
      icon: '/favicon.ico',
      tag: 'chat-message',
      data: {
        senderId: notification.senderId,
        senderName: notification.senderName,
        type: 'chat'
```

```
      }
    });

    // Show toast notification (only if chat is not open)
    if (!this.isChatWindowOpen()) {
      const toastMessage = notification.isGuest
        ? `Ø=Üd ${notification.senderName}: ${notification.content.substring(0,
50)}${notification.content.length > 50 ? '...' : ''}`
        : `Ø=Ü¬ ${notification.senderName}: ${notification.content.substring(0,
50)}${notification.content.length > 50 ? '...' : ''}`;

      toast(toastMessage, {
        icon: notification.isGuest ? 'Ø=Üd' : 'Ø=Ü¬',
        duration: 5000,
        position: 'top-right',
        style: {
          background: notification.isGuest ? '#fef3c7' : '#f3f4f6',
          color: notification.isGuest ? '#92400e' : '#1f2937',
          border: notification.isGuest ? '1px solid #d97706' : '1px solid #d1d5db'
        },
        onClick: () => {
          // Focus the chat window when toast is clicked
          this.focusChatWindow();
        }
      });
    }
  }

  // Handle exam reminder notifications
  handleExamReminder(notification) {
    console.log('Ø=Þ¨ Exam reminder received:', notification);

    // Play urgent sound for exam reminders
    this.playNotificationSound('urgent');

    // Show browser notification if permission granted
    this.showBrowserNotification({
      title: notification.title,
      body: notification.message,
      icon: '/favicon.ico',
      tag: 'exam-reminder',
      requireInteraction: true,
      data: {
        examId: notification.examDetails?.id,
        type: 'exam'
      }
    });

    // Show toast notification with custom content
    const toastContent = `Ø=Þ¨ ${notification.title}\n\n${notification.message}`;

    toast(toastContent, {
      duration: 30000, // Show for 30 seconds
      position: 'top-center',
      style: {
        background: '#fee2e2',
        border: '2px solid #dc2626',
        borderRadius: '8px',
        padding: '16px',
```

```
        maxWidth: '400px',
        fontSize: '14px',
        lineHeight: '1.4'
      },
      icon: 'Ø=Þ¨'
    });

    // Show modal for critical exam notifications
    this.showExamModal(notification);
  }

  // Handle general notifications
  handleGeneralNotification(notification) {
    console.log('Ø=Üâ General notification received:', notification);

    // Play appropriate sound based on notification type
    const soundType = notification.urgent ? 'urgent' : 'soft';
    this.playNotificationSound(soundType);

    // Show browser notification
    this.showBrowserNotification({
      title: notification.title || 'CRM Notification',
      body: notification.message,
      icon: '/favicon.ico',
      tag: 'general-notification',
      data: {
        type: 'general'
      }
    });

    // Show toast notification
    toast(notification.message, {
      icon: notification.icon || 'Ø=Üâ',
      duration: 5000,
      position: 'top-right',
      style: {
        background: notification.urgent ? '#fee2e2' : '#f3f4f6',
        color: notification.urgent ? '#dc2626' : '#1f2937',
        border: notification.urgent ? '1px solid #dc2626' : '1px solid #d1d5db'
      }
    });
  }

  // Handle user status updates
  handleUserStatusUpdate(statusUpdate) {
    console.log('Ø=Üd User status update:', statusUpdate);

    // Play soft sound for status updates (optional)
    if (statusUpdate.status === 'ONLINE' &&
this.soundPreferences.enableStatusSounds) {
      this.playNotificationSound('soft');
    }

    // Show subtle toast for important status changes
    if (statusUpdate.status === 'ONLINE') {
      toast(`${statusUpdate.userName || 'User'} is now online`, {
        icon: 'Ø=ßâ',
        duration: 3000,
        position: 'bottom-right',
```

```
      style: {
        background: '#f0f9ff',
        color: '#0369a1',
        border: '1px solid #0ea5e9'
      }
    });
  }
}

// Show browser notification
showBrowserNotification(options) {
  if ('Notification' in window && Notification.permission === 'granted') {
    const notification = new Notification(options.title, {
      body: options.body,
      icon: options.icon || '/favicon.ico',
      badge: options.badge || '/favicon.ico',
      tag: options.tag || 'crm-notification',
      requireInteraction: options.requireInteraction || false,
      silent: false,
      data: options.data || {}
    });

    notification.onclick = () => {
      window.focus();
      notification.close();

      // Handle click based on notification type
      if (options.data?.type === 'chat') {
        this.focusChatWindow();
      } else if (options.data?.type === 'exam' && options.data?.examLink) {
        window.open(options.data.examLink, '_blank');
      }
    };

    // Auto close after 10 seconds
    setTimeout(() => {
      notification.close();
    }, 10000);
  }
}

// Check if chat window is open
isChatWindowOpen() {
  // This would need to be implemented based on your chat window state
  // For now, return false to always show notifications
  return false;
}

// Focus the chat window
focusChatWindow() {
  // Dispatch custom event to focus chat window
  window.dispatchEvent(new CustomEvent('focusChatWindow'));
}

// Show exam modal
showExamModal(notification) {
  // Dispatch custom event to show exam modal
  window.dispatchEvent(new CustomEvent('showExamModal', {
    detail: notification
```

```
    }));
  }

  // Request notification permission
  async requestNotificationPermission() {
    if ('Notification' in window) {
      const permission = await Notification.requestPermission();
      console.log('Ø=Ý  Notification permission:', permission);

      if (permission === 'granted') {
        toast.success('Ø=Ý  Browser notifications enabled');
        return true;
      } else if (permission === 'denied') {
        toast.error('Ø=Ý  Browser notifications blocked');
        return false;
      }
    }
    return false;
  }

  // Enable/disable audio
  setAudioEnabled(enabled) {
    this.isAudioEnabled = enabled;
    this.updatePreferences({ enableSounds: enabled });
  }

  // Test notification sound
  testNotificationSound(type = 'message') {
    this.playNotificationSound(type);
  }

  // Get current preferences
  getPreferences() {
    return { ...this.soundPreferences };
  }

  // Disconnect from server
  disconnect() {
    if (this.socket) {
      this.socket.disconnect();
      this.socket = null;
    }
    this.isConnected = false;
    this.userId = null;
  }

  // Get connection status
  getStatus() {
    return {
      isConnected: this.isConnected,
      userId: this.userId,
      audioSupported: this.sounds.isAudioSupported(),
      audioState: this.sounds.getAudioState(),
      soundPreferences: this.soundPreferences
    };
  }
}

// Export singleton instance
```

```
export default new NotificationService();
```

src/setupTests.js

```
// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom';
```

src/utils/darkModeClasses.js

```
// Dark mode utility classes for consistent theming across the application

export const darkModeClasses = {
  // Background classes
  bg: {
    primary: "bg-white dark:bg-gray-900",
    secondary: "bg-gray-50 dark:bg-gray-800",
    card: "bg-white dark:bg-gray-800",
    modal: "bg-white dark:bg-gray-800",
    input: "bg-white dark:bg-gray-700",
    button: "bg-blue-600 dark:bg-blue-700",
    buttonSecondary: "bg-gray-200 dark:bg-gray-600",
    table: "bg-white dark:bg-gray-800",
    tableRow: "bg-white dark:bg-gray-800",
    tableHeader: "bg-gray-50 dark:bg-gray-700",
    hover: "hover:bg-gray-50 dark:hover:bg-gray-700"
  },

  // Text classes
  text: {
    primary: "text-gray-900 dark:text-white",
    secondary: "text-gray-700 dark:text-gray-300",
    muted: "text-gray-500 dark:text-gray-400",
    light: "text-gray-400 dark:text-gray-500 dark:text-gray-400",
    heading: "text-gray-900 dark:text-white",
    label: "text-gray-700 dark:text-gray-300",
    placeholder: "text-gray-500 dark:text-gray-400"
  },

  // Border classes
  border: {
    primary: "border-gray-200 dark:border-gray-600",
    secondary: "border-gray-300 dark:border-gray-500",
    input: "border-gray-300 dark:border-gray-600",
    table: "border-gray-200 dark:border-gray-700"
  },

  // Shadow classes
  shadow: {
    card: "shadow-md dark:shadow-lg dark:shadow-2xl dark:shadow-black/25",
    modal: "shadow-lg dark:shadow-2xl",
    button: "shadow-sm dark:shadow-md dark:shadow-xl dark:shadow-black/25"
  },

  // Ring/Focus classes
  ring: {
```

```
    focus: "focus:ring-blue-500 dark:focus:ring-blue-400",
    border: "focus:border-blue-500 dark:focus:border-blue-400"
  }
};

// Helper function to get combined classes
export const getDarkModeClass = (category, type) => {
  return darkModeClasses[category]?.[type] || '';
};

// Common component class combinations
export const componentClasses = {
  card: `${darkModeClasses.bg.card} ${darkModeClasses.shadow.card}
${darkModeClasses.border.primary}`,
  modal: `${darkModeClasses.bg.modal} ${darkModeClasses.shadow.modal}`,
  input: `${darkModeClasses.bg.input} ${darkModeClasses.border.input}
${darkModeClasses.text.primary} ${darkModeClasses.ring.focus}
${darkModeClasses.ring.border}`,
  button: `${darkModeClasses.bg.button} text-white
${darkModeClasses.shadow.button}`,
  buttonSecondary: `${darkModeClasses.bg.buttonSecondary}
${darkModeClasses.text.primary} ${darkModeClasses.shadow.button}`,
  table: `${darkModeClasses.bg.table} ${darkModeClasses.border.table}`,
  tableHeader: `${darkModeClasses.bg.tableHeader}
${darkModeClasses.text.secondary}`,
  tableRow: `${darkModeClasses.bg.tableRow} ${darkModeClasses.border.table}
${darkModeClasses.bg.hover}`,
  heading: `${darkModeClasses.text.heading}`,
  label: `${darkModeClasses.text.label}`,
  muted: `${darkModeClasses.text.muted}`
};

// Transition classes for smooth theme switching
export const transitionClasses = "transition-colors duration-300";

// Complete class builder function
export const buildDarkModeClasses = (baseClasses, darkModeOverrides = {}) => {
  let classes = baseClasses;

  // Add transition by default
  if (!classes.includes('transition')) {
    classes += ` ${transitionClasses}`;
  }

  // Apply dark mode overrides
  Object.entries(darkModeOverrides).forEach(([key, value]) => {
    if (darkModeClasses[key]) {
      classes += ` ${value}`;
    }
  });

  return classes;
};
```

src/utils/fixDarkMode.js

```
// Utility script to fix dark mode classes across the application
// This script provides regex patterns and replacement functions
```

```
export const darkModeReplacements = [
  // Background replacements
  {
    pattern: /className="([^"]*?)bg-white dark:bg-slate-900(?!\s+dark:)([^"]*?)"/
g,
    replacement: 'className="$1bg-white dark:bg-gray-800$2"'
  },
  {
    pattern: /className="([^"]*?)bg-gray-50 dark:bg-slate-800(?!\s+dark:)
([^"]*?)"/g,
    replacement: 'className="$1bg-gray-50 dark:bg-gray-700$2"'
  },

  // Text color replacements
  {
    pattern: /className="([^"]*?)text-gray-900 dark:text-white(?!\s+dark:)
([^"]*?)"/g,
    replacement: 'className="$1text-gray-900 dark:text-white$2"'
  },
  {
    pattern: /className="([^"]*?)text-gray-700 dark:text-gray-300 dark:text-
gray-400(?!\s+dark:)([^"]*?)"/g,
    replacement: 'className="$1text-gray-700 dark:text-gray-500$2"'
  },
  {
    pattern: /className="([^"]*?)text-gray-500 dark:text-gray-400(?!\s+dark:)
([^"]*?)"/g,
    replacement: 'className="$1text-gray-500 dark:text-gray-500$2"'
  },
  {
    pattern: /className="([^"]*?)text-gray-400 dark:text-gray-500(?!\s+dark:)
([^"]*?)"/g,
    replacement: 'className="$1text-gray-400 dark:text-gray-500$2"'
  },

  // Border replacements
  {
    pattern: /className="([^"]*?)border-gray-200 dark:border-slate-700(?!\s+dark:)
([^"]*?)"/g,
    replacement: 'className="$1border-gray-200 dark:border-gray-600$2"'
  },
  {
    pattern: /className="([^"]*?)border-gray-300 dark:border-slate-600(?!\s+dark:)
([^"]*?)"/g,
    replacement: 'className="$1border-gray-300 dark:border-gray-600$2"'
  },

  // Divide replacements
  {
    pattern: /className="([^"]*?)divide-gray-200 dark:divide-slate-700(?!\s+dark:)
([^"]*?)"/g,
    replacement: 'className="$1divide-gray-200 dark:divide-gray-700$2"'
  }
];

// Function to apply dark mode fixes to a string
export const applyDarkModeFixes = (content) => {
  let fixedContent = content;
```

```
  darkModeReplacements.forEach(({ pattern, replacement }) => {
    fixedContent = fixedContent.replace(pattern, replacement);
  });

  return fixedContent;
};

// Common class combinations for quick replacement
export const commonReplacements = {
  // Card components
  'bg-white dark:bg-slate-900 rounded-lg shadow': 'bg-white dark:bg-gray-800
rounded-lg shadow transition-colors duration-300',
  'bg-white dark:bg-slate-900 rounded-lg shadow-md dark:shadow-xl dark:shadow-
black/25': 'bg-white dark:bg-gray-800 rounded-lg shadow-md dark:shadow-xl
dark:shadow-black/25 transition-colors duration-300',
  'bg-white dark:bg-slate-900 p-6 rounded-lg shadow-md dark:shadow-xl dark:shadow-
black/25': 'bg-white dark:bg-gray-800 p-6 rounded-lg shadow-md dark:shadow-xl
dark:shadow-black/25 transition-colors duration-300',

  // Table components
  'bg-white dark:bg-slate-900 divide-y divide-gray-200 dark:divide-slate-700':
'bg-white dark:bg-gray-800 divide-y divide-gray-200 dark:divide-gray-700
transition-colors duration-300',
  'bg-gray-50 dark:bg-slate-800': 'bg-gray-50 dark:bg-gray-700 transition-colors
duration-300',

  // Text components
  'text-gray-900 dark:text-white': 'text-gray-900 dark:text-white',
  'text-gray-700 dark:text-gray-300 dark:text-gray-400': 'text-gray-700 dark:text-
gray-300',
  'text-gray-500 dark:text-gray-400': 'text-gray-500 dark:text-gray-400',

  // Border components
  'border-gray-200 dark:border-slate-700': 'border-gray-200 dark:border-gray-600',
  'border-gray-300 dark:border-slate-600': 'border-gray-300 dark:border-gray-600'
};

// Function to apply common replacements
export const applyCommonReplacements = (content) => {
  let fixedContent = content;

  Object.entries(commonReplacements).forEach(([oldClass, newClass]) => {
    const regex = new RegExp(`className="([^"]*?)${oldClass.replace(/[.*+?^${}()|
[\]\\]/g, '\\$&')}([^"]*?)"`, 'g');
    fixedContent = fixedContent.replace(regex, `className="$1${newClass}$2"`);
  });

  return fixedContent;
};

// List of files that need dark mode fixes
export const filesToFix = [
  'src/pages/ProfilePage.jsx',
  'src/pages/RepeatCustomersPage.jsx',
  'src/pages/ManagerDashboard.jsx',
  'src/pages/LeadSalesUpdatePage.jsx',
  'src/pages/SalesPage.jsx',
  'src/pages/SalesCreatePage.jsx',
```

```
  'src/pages/TutorialsPage.jsx',
  'src/pages/LeadsPage.jsx',
  'src/pages/LeadSalesSheet.jsx',
  'src/pages/ManagementContactsPage.jsx',
  'src/pages/ProspectsPage.jsx',
  'src/pages/TokenDebugPage.jsx',
  'src/pages/TaskManagementPage.jsx',
  'src/pages/TestNotificationsPage.jsx',
  'src/pages/HomePage.jsx',
  'src/pages/AdminLeadsPage.jsx',
  'src/pages/AdminReportsPage.jsx',
  'src/pages/AdminImportPage.jsx'
];

// Priority fixes for most commonly used components
export const priorityFixes = [
  // Modal backgrounds
  {
    find: 'className="bg-white dark:bg-slate-900 transition-all duration-200 ease-
out-colors duration-300'
  },

  // Table headers
  {
    find: 'className="bg-gray-50 dark:bg-slate-800', replace: 'className="bg-
gray-50 dark:bg-gray-700 transition-colors duration-300'
  },

  // Card containers
  {
    find: 'className="bg-white dark:bg-slate-900 transition-all duration-200 ease-
out-colors duration-300'
  },

  // Form inputs
  {
    find: 'className="w-full p-2 border border-gray-300 dark:border-slate-600
rounded-md', replace: 'className="w-full p-2 border border-gray-300 dark:border-
gray-600 rounded-md bg-white dark:bg-gray-700 text-gray-900 dark:text-white
transition-colors duration-300'
  }
];

export default {
  darkModeReplacements,
  applyDarkModeFixes,
  commonReplacements,
  applyCommonReplacements,
  filesToFix,
  priorityFixes
};
```

src/utils/helpers.js

```
/**
 * Currency formatting utility functions
 */
```

```
import axios from 'axios';
import { salesAPI } from '../services/api';

// Default currency settings
let currencySettings = {
  currency: 'USD',  // Default currency code
  locale: 'en-US',  // Default locale for formatting
  rate: 1,          // Exchange rate multiplier (1 for base currency USD)
  exchangeRates: {  // Exchange rates relative to USD
    USD: 1,
    INR: 84.62,     // 1 USD = 84.62 INR (updated rate)
    EUR: 0.92,      // 1 USD = 0.92 EUR
    GBP: 0.79       // 1 USD = 0.79 GBP
  }
};

// Original base currency that amounts are stored in
export const BASE_CURRENCY = 'INR';

/**
 * Convert amount from base currency to target currency
 * @param {number} amount - Amount in base currency
 * @param {string} targetCurrency - Target currency code
 * @returns {number} - Converted amount
 */
export const convertCurrency = (amount, targetCurrency =
currencySettings.currency) => {
  if (!amount || isNaN(amount)) return 0;

  // Get exchange rates
  const rates = currencySettings.exchangeRates || {
    USD: 1,
    INR: 84.62,
    EUR: 0.92,
    GBP: 0.79
  };

  // If base currency is already the target currency, return amount
  if (BASE_CURRENCY === targetCurrency) {
    return amount;
  }

  // Convert from base currency to USD first (if not already USD)
  let amountInUSD;
  if (BASE_CURRENCY === 'USD') {
    amountInUSD = amount;
  } else {
    amountInUSD = amount / rates[BASE_CURRENCY];
  }

  // Then convert from USD to target currency
  return amountInUSD * rates[targetCurrency];
};

/**
 * Format a date string in the specified format
 * @param {string|Date} date - The date to format
 * @param {string} format - Format string (default: 'DD/MM/YYYY')
 * @returns {string} Formatted date string
```

```
 */
export const formatDate = (date, format = 'DD/MM/YYYY') => {
  if (!date) return 'N/A';

  const d = new Date(date);
  if (isNaN(d.getTime())) return 'Invalid Date';

  const day = String(d.getDate()).padStart(2, '0');
  const month = String(d.getMonth() + 1).padStart(2, '0');
  const year = d.getFullYear();

  if (format === 'DD/MM/YYYY') {
    return `${day}/${month}/${year}`;
  } else if (format === 'YYYY-MM-DD') {
    return `${year}-${month}-${day}`;
  } else if (format === 'MMM DD, YYYY') {
    const monthNames = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Oct', 'Nov', 'Dec'];
    return `${monthNames[d.getMonth()]} ${day}, ${year}`;
  }

  return `${day}/${month}/${year}`;
};

/**
 * Format currency value based on current currency settings
 * @param {number} amount - The amount to format
 * @returns {string} Formatted currency string
 */
export const formatCurrency = (amount) => {
  if (amount === null || amount === undefined) return 'N/A';

  const settings = getCurrencySettings();

  try {
    return new Intl.NumberFormat(settings.locale, {
      style: 'currency',
      currency: settings.currency,
      minimumFractionDigits: 2,
      maximumFractionDigits: 2
    }).format(amount);
  } catch (error) {
    console.error('Error formatting currency:', error);
    return `${settings.symbol}${amount.toFixed(2)}`;
  }
};

/**
 * Update global currency settings
 * @param {Object} settings - New currency settings
 * @param {string} settings.currency - Currency code (e.g., 'USD', 'INR')
 * @param {string} settings.locale - Locale string (e.g., 'en-US', 'en-IN')
 * @param {number} settings.rate - Exchange rate
 * @param {Object} settings.exchangeRates - Exchange rates object
 */
export const setCurrencySettings = (settings) => {
  currencySettings = { ...currencySettings, ...settings };
};
```

```
/**
 * Get current currency settings
 * @returns {Object} Current currency settings
 */
export const getCurrencySettings = () => {
  const settings = localStorage.getItem('currencySettings');
  return settings ? JSON.parse(settings) : { currency: 'USD', symbol: '$' };
};

// Add a new utility function to get a direct sales count
export const getDirectSalesCount = async () => {
  try {
    // Strategy 1: Try the dedicated count endpoint
    try {
      const response = await salesAPI.getCount();

      if (response.data && response.data.success && typeof response.data.count
=== 'number') {
        const salesCount = response.data.count;
        return salesCount;
      } else {
        throw new Error('Invalid count response format');
      }
    } catch (countError) {
      // Strategy 2: Fallback to getting all sales and counting them
      try {
        const response = await salesAPI.getAllForced();

        if (response.data && response.data.success) {
          let salesCount = 0;

          if (response.data.count && typeof response.data.count === 'number') {
            salesCount = response.data.count;
          } else if (response.data.data && Array.isArray(response.data.data)) {
            salesCount = response.data.data.length;
          } else if (Array.isArray(response.data)) {
            salesCount = response.data.length;
          }

          return salesCount;
        } else {
          throw new Error('Invalid sales response format');
        }
      } catch (fullFetchError) {
        throw new Error(`All strategies failed. Count error:
${countError.message}, Full fetch error: ${fullFetchError.message}`);
      }
    }
  } catch (error) {
    throw error;
  }
};
```

src/utils/professionalDarkMode.js

```
// Professional Dark Mode Color System
// Following industry standards from GitHub, Discord, Notion, and modern design
systems
```

```
export const professionalColors = {
  // Light mode colors
  light: {
    // Backgrounds
    bg: {
      primary: '#ffffff',
      secondary: '#f8fafc',
      tertiary: '#f1f5f9',
      elevated: '#ffffff',
      overlay: 'rgba(0, 0, 0, 0.1)',
      glass: 'rgba(255, 255, 255, 0.8)',
    },
    // Text colors
    text: {
      primary: '#0f172a',
      secondary: '#475569',
      tertiary: '#64748b',
      muted: '#94a3b8',
      inverse: '#ffffff',
      accent: '#3b82f6',
    },
    // Border colors
    border: {
      primary: '#e2e8f0',
      secondary: '#cbd5e1',
      focus: '#3b82f6',
      error: '#ef4444',
      success: '#10b981',
    },
    // Interactive states
    interactive: {
      hover: '#f1f5f9',
      active: '#e2e8f0',
      disabled: '#f8fafc',
    }
  },

  // Dark mode colors (professional palette)
  dark: {
    // Backgrounds - Using sophisticated grays, not pure black
    bg: {
      primary: '#0f1419',      // Deep blue-gray (like GitHub)
      secondary: '#161b22',    // Slightly lighter
      tertiary: '#21262d',     // Card backgrounds
      elevated: '#30363d',     // Elevated elements
      overlay: 'rgba(0, 0, 0, 0.6)',
      glass: 'rgba(15, 20, 25, 0.8)',
    },
    // Text colors - High contrast, easy on eyes
    text: {
      primary: '#f0f6fc',      // Soft white
      secondary: '#c9d1d9',    // Light gray
      tertiary: '#8b949e',     // Medium gray
      muted: '#6e7681',        // Muted gray
      inverse: '#0f172a',      // Dark for light backgrounds
      accent: '#58a6ff',       // Bright blue accent
    },
    // Border colors
```

```
    border: {
      primary: '#30363d',
      secondary: '#21262d',
      focus: '#58a6ff',
      error: '#f85149',
      success: '#3fb950',
    },
    // Interactive states
    interactive: {
      hover: '#21262d',
      active: '#30363d',
      disabled: '#161b22',
    }
  }
};

// Professional component classes
export const professionalClasses = {
  // Card components
  card: {
    light: 'bg-white dark:bg-slate-900 border border-gray-200 dark:border-
slate-700 shadow-sm dark:shadow-lg dark:shadow-black/25 hover:shadow-md
dark:shadow-xl dark:shadow-black/25 transition-all duration-200',
    dark: 'bg-gray-900 border border-gray-800 shadow-lg dark:shadow-2xl
dark:shadow-black/25 hover:shadow-xl transition-all duration-200',
    unified: 'bg-white dark:bg-gray-900 border border-gray-200 dark:border-
gray-800 shadow-sm dark:shadow-lg dark:shadow-2xl dark:shadow-black/25
hover:shadow-md dark:hover:shadow-xl transition-all duration-200'
  },

  // Text components
  text: {
    heading: 'text-gray-900 dark:text-gray-100 font-semibold',
    body: 'text-gray-700 dark:text-gray-300',
    muted: 'text-gray-500 dark:text-gray-400',
    accent: 'text-blue-600 dark:text-blue-400',
    inverse: 'text-white dark:text-gray-900 dark:text-white'
  },

  // Background components
  background: {
    primary: 'bg-white dark:bg-gray-900',
    secondary: 'bg-gray-50 dark:bg-gray-800',
    tertiary: 'bg-gray-100 dark:bg-gray-700',
    elevated: 'bg-white dark:bg-gray-800',
    overlay: 'bg-black/10 dark:bg-black/60'
  },

  // Interactive components
  button: {
    primary: 'bg-blue-600 hover:bg-blue-700 dark:bg-blue-500 dark:hover:bg-
blue-600 text-white shadow-sm dark:shadow-lg dark:shadow-black/25 hover:shadow-md
dark:shadow-xl dark:shadow-black/25 transition-all duration-200',
    secondary: 'bg-gray-100 dark:bg-slate-700 hover:bg-gray-200 dark:bg-gray-700
dark:hover:bg-gray-600 text-gray-900 dark:text-gray-100 border border-gray-300
dark:border-gray-600 transition-all duration-200',
    ghost: 'hover:bg-gray-100 dark:hover:bg-gray-800 text-gray-700 dark:text-
gray-300 transition-all duration-200'
  },
```

```
  // Form components
  input: {
    base: 'bg-white dark:bg-gray-800 border border-gray-300 dark:border-gray-600
text-gray-900 dark:text-gray-100 placeholder-gray-500 dark:placeholder-gray-400
dark:placeholder-gray-500 focus:ring-2 focus:ring-blue-500 dark:focus:ring-
blue-400 focus:border-blue-500 dark:focus:border-blue-400 transition-all
duration-200',
    error: 'border-red-500 dark:border-red-400 focus:ring-red-500 dark:focus:ring-
red-400',
    success: 'border-green-500 dark:border-green-400 focus:ring-green-500
dark:focus:ring-green-400'
  },

  // Table components
  table: {
    container: 'bg-white dark:bg-gray-900 border border-gray-200 dark:border-
gray-800 rounded-lg overflow-hidden shadow-sm dark:shadow-lg dark:shadow-2xl
dark:shadow-black/25',
    header: 'bg-gray-50 dark:bg-gray-800 border-b border-gray-200 dark:border-
gray-700',
    row: 'border-b border-gray-200 dark:border-gray-700 hover:bg-gray-50
dark:hover:bg-gray-800 transition-colors duration-150',
    cell: 'text-gray-900 dark:text-gray-100'
  },

  // Navigation components
  nav: {
    primary: 'bg-white dark:bg-gray-900 border-b border-gray-200 dark:border-
gray-800 shadow-sm dark:shadow-lg dark:shadow-2xl dark:shadow-black/25',
    sidebar: 'bg-gray-50 dark:bg-gray-900 border-r border-gray-200 dark:border-
gray-800',
    link: 'text-gray-700 dark:text-gray-300 hover:text-gray-900 dark:hover:text-
gray-100 hover:bg-gray-100 dark:hover:bg-gray-800 transition-all duration-200'
  },

  // Status indicators
  status: {
    success: 'bg-green-100 dark:bg-green-900/30 text-green-800 dark:text-
green-300 border border-green-200 dark:border-green-800',
    warning: 'bg-yellow-100 dark:bg-yellow-900/30 text-yellow-800 dark:text-
yellow-300 border border-yellow-200 dark:border-yellow-800',
    error: 'bg-red-100 dark:bg-red-900/30 text-red-800 dark:text-red-300 border
border-red-200 dark:border-red-800',
    info: 'bg-blue-100 dark:bg-blue-900/30 text-blue-800 dark:text-blue-300
border border-blue-200 dark:border-blue-800'
  }
};

// Professional transitions
export const transitions = {
  fast: 'transition-all duration-150 ease-out',
  normal: 'transition-all duration-200 ease-out',
  slow: 'transition-all duration-300 ease-out',
  colors: 'transition-colors duration-200 ease-out',
  transform: 'transition-transform duration-200 ease-out',
  shadow: 'transition-shadow duration-200 ease-out'
};
```

```
// Professional shadows
export const shadows = {
  light: {
    sm: 'shadow-sm dark:shadow-lg dark:shadow-black/25',
    md: 'shadow-md dark:shadow-xl dark:shadow-black/25',
    lg: 'shadow-lg dark:shadow-2xl dark:shadow-black/25',
    xl: 'shadow-xl'
  },
  dark: {
    sm: 'shadow-lg dark:shadow-2xl dark:shadow-black/25 shadow-black/25',
    md: 'shadow-xl shadow-black/25',
    lg: 'shadow-2xl shadow-black/25',
    xl: 'shadow-2xl shadow-black/30'
  },
  unified: {
    sm: 'shadow-sm dark:shadow-lg dark:shadow-black/25',
    md: 'shadow-md dark:shadow-xl dark:shadow-black/25',
    lg: 'shadow-lg dark:shadow-2xl dark:shadow-black/25',
    xl: 'shadow-xl dark:shadow-2xl dark:shadow-black/30'
  }
};

// Helper function to get professional classes
export const getProClass = (component, variant = 'unified') => {
  return professionalClasses[component]?.[variant] || '';
};

// Helper function to combine classes with transitions
export const combineClasses = (...classes) => {
  return classes.filter(Boolean).join(' ');
};

// Professional color palette for specific use cases
export const colorPalette = {
  // Brand colors
  brand: {
    primary: 'bg-blue-600 dark:bg-blue-500',
    secondary: 'bg-indigo-600 dark:bg-indigo-500',
    accent: 'bg-purple-600 dark:bg-purple-500'
  },

  // Semantic colors
  semantic: {
    success: 'bg-green-600 dark:bg-green-500',
    warning: 'bg-yellow-600 dark:bg-yellow-500',
    error: 'bg-red-600 dark:bg-red-500',
    info: 'bg-blue-600 dark:bg-blue-500'
  },

  // Neutral grays (professional palette)
  neutral: {
    50: 'bg-gray-50 dark:bg-gray-900',
    100: 'bg-gray-100 dark:bg-gray-800',
    200: 'bg-gray-200 dark:bg-gray-700',
    300: 'bg-gray-300 dark:bg-gray-600',
    400: 'bg-gray-400 dark:bg-gray-50 dark:bg-slate-8000',
    500: 'bg-gray-50 dark:bg-slate-8000 dark:bg-gray-400',
    600: 'bg-gray-600 dark:bg-gray-300',
    700: 'bg-gray-700 dark:bg-gray-200 dark:bg-slate-600',
```

```
    800: 'bg-gray-800 dark:bg-gray-100 dark:bg-slate-700',
    900: 'bg-gray-900 dark:bg-gray-50 dark:bg-slate-800'
  }
};

export default {
  professionalColors,
  professionalClasses,
  transitions,
  shadows,
  getProClass,
  combineClasses,
  colorPalette
};
```

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */Đ
export default {  Đ
  darkMode: 'class',Đ
    content: [Đ
      "./index.html",Đ
      "./src/**/*.{js,ts,jsx,tsx}",Đ
    ],Đ
    theme: {Đ
      extend: {},Đ
    },Đ
    plugins: [],Đ
  }Đ
```

vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import { resolve } from 'path'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  base: '/', // Ensure correct base path for production
  build: {
    outDir: 'dist',
    assetsDir: 'assets',
    // Ensure proper handling of dynamic imports
    rollupOptions: {
      output: {
        manualChunks: undefined,
      },
    },
  },
  server: {
    port: 5173,
    host: true,
    // This is important for SPA routing
    historyApiFallback: true,
    proxy: {
      '/api': {
```

```
        target: 'http://localhost:8080',
        changeOrigin: true,
        secure: false
      }
    }
  },
  resolve: {
    alias: {
      '@': resolve(__dirname, 'src'),
    },
  },
})
```