Personal workspace Clarily Free Development S

Watching for users

What will your users see?

Install the SDK, run your dev server, and sign yourself up. The moment we detect that first account, installation is complete.

Next.js React Astro Expo iOS

Overview Users Organizations Subscriptions Beta Configure

JS JavaScript

Note

Clerk also provides a React Router SDK. See the **quickstart** in the Clerk Docs.

Copy this quickstart guide as a prompt for LLMs to implement Clerk in your React application.

Copy prompt

1 Set up a React application using Vite

Run the following commands to create a new React app using Vite:

npm yarn pnpm

terminal

```
npm create vite@latest clerk-react -- --template react-ts
cd clerk-react
npm install
```

2 **Install `@clerk/clerk-react`**

The Clerk React SDK gives you access to prebuilt components, hooks, and helpers to make user authentication easier.

Run the following command to install the SDK:

npm yarn pnpm

terminal

```
npm install @clerk/clerk-react
```

3 **Set your Clerk API keys**

Add your Clerk Publishable Key to your `.env` or create the file if it doesn't exist. Retrieve this key anytime from the **API keys** page.

.env

```
VITE_CLERK_PUBLISHABLE_KEY=pk_test_cmFOaW9uYWwtc3BpZGVyLTkOLmNsZXJrLmFjY291br
```

4 **Import the Clerk Publishable Key**

In your `src/main.tsx` file, import your Clerk Publishable Key. You can add an `if` statement to check that it is imported and that it exists. This will prevent running the app without the Publishable Key, and will also prevent TypeScript errors.

src/main.tsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'
import './index.css'

// Import your Publishable Key
const PUBLISHABLE_KEY = import.meta.env.VITE_CLERK_PUBLISHABLE_KEY

if (!PUBLISHABLE_KEY) {
  throw new Error('Missing Publishable Key')
}

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

5 **Add `ClerkProvider` to your app**

The `ClerkProvider` component provides session and user context to Clerk's hooks and components. It's recommended to wrap your entire app at the entry point with `ClerkProvider` to make authentication globally accessible. See the **reference docs** for other configuration options.

Pass your Publishable Key as a prop to the component.

src/main.tsx

```
import { StrictMode } from 'react'
  import { createRoot } from 'react-dom/client'
  import './index.css'
  import App from './App.tsx'
  import { ClerkProvider } from '@clerk/clerk-react'

  // Import your Publishable Key
```

```
const PUBLISHABLE_KEY = import.meta.env.VITE_CLERK_PUBLISHABLE_KEY

if (!PUBLISHABLE_KEY) {
  throw new Error('Add your Clerk Publishable Key to the .env file')
}

createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <ClerkProvider publishableKey={PUBLISHABLE_KEY}>
      <App />
    </ClerkProvider>
  </StrictMode>,
)
```

6 **Create a header with Clerk components**

You can control which content signed in and signed out users can see with Clerk's prebuilt components. To get started, create a header using the following components:

- `<SignedIn />`: Children of this component can only be seen while signed in.
- `<SignedOut />`: Children of this component can only be seen while signed out.
- `<UserButton />`: Shows the signed-in user's avatar. Selecting it opens a dropdown menu with account management options.
- `<SignInButton />`: An unstyled component that links to the sign-in page. In this example, since no props or **environment variables** are set for the sign-in URL, this component links to the **Account Portal sign-in page**.

src/App.tsx

```
import { SignedIn, SignedOut, SignInButton, UserButton } from '@clerk/clerk-r

export default function App() {
  return (
    <header>
      <SignedOut>
        <SignInButton />
      </SignedOut>
      <SignedIn>
        <UserButton />
      </SignedIn>
    </header>
```

```
  );
}
```

Create your first user

Run your project. Then, visit your app's homepage at `http://localhost:5173` and sign up to create your first user.

npm yarn pnpm

terminal

```
npm run dev
```

Next steps

Add routing with React Router

React Router supports three different routing strategies, or **modes**. The declarative mode is the easiest way to get started. It enables basic routing features like matching URLs to components, navigating around the app, and providing active states with APIs like `<Link />`, `useNavigate()`, and `useLocation()`.

Continue to the guide in Clerk Docs