

# Step-by-Step Coding Tutorial: Building URL Shortener from Scratch

## Table of Contents

1. [Prerequisites](#)
  2. [Phase 1: Project Setup](#)
  3. [Phase 2: Basic Structure](#)
  4. [Phase 3: Static Pages](#)
  5. [Phase 4: Routing Setup](#)
  6. [Phase 5: Authentication](#)
  7. [Phase 6: API Integration](#)
  8. [Phase 7: URL Shortening](#)
  9. [Phase 8: Dashboard](#)
  10. [Phase 9: Analytics](#)
  11. [Phase 10: Polish](#)
- 

## Prerequisites

### What You Need to Know

- Basic HTML/CSS
- Basic JavaScript (variables, functions, arrays, objects)
- Command line basics (cd, ls, npm commands)

### What You'll Learn

- React fundamentals
- Component-based architecture
- State management
- API integration
- Modern web development workflow

### Tools You Need

- Node.js (version 18+)
  - Code editor (VS Code recommended)
  - Terminal/Command prompt
  - Web browser
- 

## Phase 1: Project Setup

### Step 1.1: Create React Project

```
# Create new React project with Vite
npm create vite@latest my-url-shortener --template react

# Navigate to project folder
cd my-url-shortener
```

```
# Install dependencies
npm install
```

### What just happened?

- Created a new React project with modern tooling
- Vite is a fast build tool (alternative to Create React App)
- Template includes basic React setup

## Step 1.2: Install Required Dependencies

```
# Install all packages we'll need
npm install react-router-dom axios react-query tailwindcss autoprefixer postcss
npm install react-hook-form react-hot-toast react-icons
npm install @mui/material @emotion/react @emotion/styled
npm install chart.js react-chartjs-2 dayjs framer-motion
npm install react-copy-to-clipboard react-loader-spinner
```

### What each package does:

- `react-router-dom` : Navigation between pages
- `axios` : Making API calls to backend
- `react-query` : Managing server data
- `tailwindcss` : Styling framework
- `react-hook-form` : Form handling
- `react-hot-toast` : Notifications
- `chart.js` : Creating graphs
- `dayjs` : Date formatting

## Step 1.3: Setup Tailwind CSS

```
# Initialize Tailwind
npx tailwindcss init -p
```

### Update `tailwind.config.js` :

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    './index.html',
    './src/**/*..{js,ts,jsx,tsx}',
  ],
  theme: {
    extend: {
      colors: {
        btnColor: "#3b82f6",
        linkColor: "#1d4ed8",
      },
      backgroundImage: {
        'custom-gradient': 'linear-gradient(135deg, #667eea 0%, #764ba2 100%)',
      }
    }
  }
}
```

```
    },  
  },  
  plugins: [],  
}
```

**Update** `src/index.css` :

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

## Step 1.4: Test Your Setup

```
# Start development server  
npm run dev
```

Visit `http://localhost:5173` - you should see the default React page.

**Checkpoint:** ☒ You have a working React app with Tailwind CSS

---

## Phase 2: Basic Structure

### Step 2.1: Create Folder Structure

Create these folders in `src/` :

```
src/  
├─ components/  
├─ pages/  
├─ hooks/  
├─ api/  
├─ context/  
├─ utils/  
└─ assets/
```

#### Why this structure?

- `components/` : Reusable UI pieces
- `pages/` : Full page components
- `hooks/` : Custom React hooks
- `api/` : Backend communication
- `context/` : Global state
- `utils/` : Helper functions

### Step 2.2: Create Basic Components

**Create** `src/components/Layout.jsx` :

```
function Layout({ children }) {  
  return (  
    <div className="min-h-screen bg-gray-50">
```

```

    <nav className="bg-blue-600 text-white p-4">
      <h1 className="text-2xl font-bold">URL Shortener</h1>
    </nav>
    <main className="container mx-auto px-4 py-8">
      {children}
    </main>
  </div>
);
}

export default Layout;

```

#### What this does:

- Creates a basic layout with navigation
- `children` prop allows content to be inserted
- Uses Tailwind classes for styling

#### Create `src/pages/HomePage.jsx` :

```

import Layout from '../components/Layout';

function HomePage() {
  return (
    <Layout>
      <div className="text-center">
        <h1 className="text-4xl font-bold text-gray-800 mb-4">
          Shorten Your URLs
        </h1>
        <p className="text-gray-600 mb-8">
          Create short, memorable links that are easy to share
        </p>
        <div className="bg-white p-6 rounded-lg shadow-md max-w-md mx-auto">
          <input
            type="url"
            placeholder="Enter your long URL here"
            className="w-full p-3 border border-gray-300 rounded-md mb-4"
          />
          <button className="w-full bg-blue-600 text-white p-3 rounded-md hover:bg-blue-700">
            Shorten URL
          </button>
        </div>
      </div>
    </Layout>
  );
}

export default HomePage;


```

### Step 2.3: Update App.jsx

```
import HomePage from '../pages/HomePage';

function App() {
  return <HomePage />;
}

export default App;
```

**Checkpoint:**  You have a basic homepage with a URL input form

---

## Phase 3: Static Pages

### Step 3.1: Create All Page Components

Create `src/pages/LoginPage.jsx` :

```
import Layout from '../components/Layout';

function LoginPage() {
  return (
    <Layout>
      <div className="max-w-md mx-auto">
        <h2 className="text-3xl font-bold text-center mb-6">Login</h2>
        <form className="bg-white p-6 rounded-lg shadow-md">
          <div className="mb-4">
            <label className="block text-gray-700 mb-2">Username</label>
            <input
              type="text"
              className="w-full p-3 border border-gray-300 rounded-md"
              placeholder="Enter username"
            />
          </div>
          <div className="mb-6">
            <label className="block text-gray-700 mb-2">Password</label>
            <input
              type="password"
              className="w-full p-3 border border-gray-300 rounded-md"
              placeholder="Enter password"
            />
          </div>
          <button className="w-full bg-blue-600 text-white p-3 rounded-md hover:bg-blue-700">
            Login
          </button>
        </form>
      </div>
    </Layout>
  );
}
```

```
export default LoginPage;
```

Create `src/pages/RegisterPage.jsx` :

```
import Layout from '../components/Layout';

function RegisterPage() {
  return (
    <Layout>
      <div className="max-w-md mx-auto">
        <h2 className="text-3xl font-bold text-center mb-6">Register</h2>
        <form className="bg-white p-6 rounded-lg shadow-md">
          <div className="mb-4">
            <label className="block text-gray-700 mb-2">Username</label>
            <input
              type="text"
              className="w-full p-3 border border-gray-300 rounded-md"
              placeholder="Choose username"
            />
          </div>
          <div className="mb-4">
            <label className="block text-gray-700 mb-2">Email</label>
            <input
              type="email"
              className="w-full p-3 border border-gray-300 rounded-md"
              placeholder="Enter email"
            />
          </div>
          <div className="mb-6">
            <label className="block text-gray-700 mb-2">Password</label>
            <input
              type="password"
              className="w-full p-3 border border-gray-300 rounded-md"
              placeholder="Create password"
            />
          </div>
          <button className="w-full bg-blue-600 text-white p-3 rounded-md hover:bg-blue-700">
            Register
          </button>
        </form>
      </div>
    </Layout>
  );
}

export default RegisterPage;
```

Create `src/pages/DashboardPage.jsx` :

```
import Layout from '../components/Layout';

function DashboardPage() {
  return (
    <Layout>
      <div>
        <h2 className="text-3xl font-bold mb-6">My Dashboard</h2>
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div className="bg-white p-6 rounded-lg shadow-md">
            <h3 className="text-xl font-semibold mb-4">Create New Short URL</h3>
            <input
              type="url"
              placeholder="Enter URL to shorten"
              className="w-full p-3 border border-gray-300 rounded-md mb-4"
            />
            <button className="w-full bg-blue-600 text-white p-3 rounded-md
hover:bg-blue-700">
              Create Short URL
            </button>
          </div>
          <div className="bg-white p-6 rounded-lg shadow-md">
            <h3 className="text-xl font-semibold mb-4">Your URLs</h3>
            <p className="text-gray-600">No URLs created yet</p>
          </div>
        </div>
      </div>
    </Layout>
  );
}

export default DashboardPage;
```

**Checkpoint:** ☒ You have all basic page layouts

---

## Phase 4: Routing Setup

### Step 4.1: Install and Setup React Router

**Update** `src/App.jsx` :

```
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import HomePage from './pages/HomePage';
import LoginPage from './pages/LoginPage';
import RegisterPage from './pages/RegisterPage';
import DashboardPage from './pages/DashboardPage';

function App() {
  return (
    <Router>
      <Routes>
```

```

    <Route path="/" element={<HomePage />} />
    <Route path="/login" element={<LoginPage />} />
    <Route path="/register" element={<RegisterPage />} />
    <Route path="/dashboard" element={<DashboardPage />} />
  </Routes>
</Router>
);
}

export default App;

```

## Step 4.2: Add Navigation Links

Update `src/components/Layout.jsx` :

```

import { Link } from 'react-router-dom';

function Layout({ children }) {
  return (
    <div className="min-h-screen bg-gray-50">
      <nav className="bg-blue-600 text-white p-4">
        <div className="container mx-auto flex justify-between items-center">
          <Link to="/" className="text-2xl font-bold">URL Shortener</Link>
          <div className="space-x-4">
            <Link to="/" className="hover:text-blue-200">Home</Link>
            <Link to="/dashboard" className="hover:text-blue-200">Dashboard</Link>
            <Link to="/login" className="hover:text-blue-200">Login</Link>
            <Link to="/register" className="hover:text-blue-200">Register</Link>
          </div>
        </div>
      </nav>
      <main className="container mx-auto px-4 py-8">
        {children}
      </main>
    </div>
  );
}

export default Layout;

```

**Test your routing:**

- Click navigation links
- URLs should change
- Pages should switch without page reload

**Checkpoint:** ☒ Navigation between pages works

## Phase 5: Authentication

### Step 5.1: Create Authentication Context



Create `src/context/AuthContext.jsx` :

```
import { createContext, useContext, useState } from 'react';

const AuthContext = createContext();

export function AuthProvider({ children }) {
  const [user, setUser] = useState(null);
  const [token, setToken] = useState(localStorage.getItem('token'));

  const login = (userData, authToken) => {
    setUser(userData);
    setToken(authToken);
    localStorage.setItem('token', authToken);
  };

  const logout = () => {
    setUser(null);
    setToken(null);
    localStorage.removeItem('token');
  };

  const value = {
    user,
    token,
    login,
    logout,
    isAuthenticated: !!token
  };

  return (
    <AuthContext.Provider value={value}>
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error('useAuth must be used within AuthProvider');
  }
  return context;
}
```

## Step 5.2: Wrap App with AuthProvider

Update `src/main.jsx` :

```
import React from 'react';
import ReactDOM from 'react-dom/client';
```

```
import App from './App.jsx';
import './index.css';
import { AuthProvider } from './context/AuthContext.jsx';

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

### Step 5.3: Update Navigation Based on Auth

Update `src/components/Layout.jsx` :

```
import { Link } from 'react-router-dom';
import { useAuth } from '../context/AuthContext';

function Layout({ children }) {
  const { isAuthenticated, logout, user } = useAuth();

  return (
    <div className="min-h-screen bg-gray-50">
      <nav className="bg-blue-600 text-white p-4">
        <div className="container mx-auto flex justify-between items-center">
          <Link to="/" className="text-2xl font-bold">URL Shortener</Link>
          <div className="space-x-4">
            <Link to="/" className="hover:text-blue-200">Home</Link>
            {isAuthenticated ? (
              <>
                <Link to="/dashboard" className="hover:text-blue-200">Dashboard</Link>
                <span className="text-blue-200">Hello, {user?.username}!</span>
                <button
                  onClick={logout}
                  className="bg-red-500 px-3 py-1 rounded hover:bg-red-600"
                >
                  Logout
                </button>
              </>
            ) : (
              <>
                <Link to="/login" className="hover:text-blue-200">Login</Link>
                <Link to="/register" className="bg-green-500 px-3 py-1 rounded
                  hover:bg-green-600">
                  Register
                </Link>
              </>
            )}
          </div>
        </div>
      </nav>
      <div>{children}</div>
    </div>
  );
}
```

```

        </div>
      </nav>
      <main className="container mx-auto px-4 py-8">
        {children}
      </main>
    </div>
  );
}

export default Layout;

```

**Checkpoint:** ☒ Navigation changes based on login status

---

## Phase 6: API Integration

### Step 6.1: Setup API Configuration

Create `src/api/api.js` :

```

import axios from 'axios';

const api = axios.create({
  baseURL: 'http://localhost:8080', // Your backend URL
  headers: {
    'Content-Type': 'application/json',
  },
});

// Add token to requests automatically
api.interceptors.request.use((config) => {
  const token = localStorage.getItem('token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

export default api;

```

### Step 6.2: Create API Functions

Create `src/api/auth.js` :

```

import api from './api';

export const authAPI = {
  register: async (userData) => {
    const response = await api.post('/api/auth/register', userData);
    return response.data;
  },

```

```

login: async (credentials) => {
  const response = await api.post('/api/auth/login', credentials);
  return response.data;
},
};

```

Create `src/api/urls.js` :

```

import api from './api';

export const urlsAPI = {
  shorten: async (originalUrl) => {
    const response = await api.post('/api/urls/shorten', { originalUrl });
    return response.data;
  },

  getMyUrls: async () => {
    const response = await api.get('/api/urls/myurls');
    return response.data;
  },

  getAnalytics: async (shortCode) => {
    const response = await api.get(`/api/urls/analytics/${shortCode}`);
    return response.data;
  },
};

```

### Step 6.3: Setup React Query

Update `src/main.jsx` :

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import { QueryClient, QueryClientProvider } from 'react-query';
import App from './App.jsx';
import './index.css';
import { AuthProvider } from './context/AuthContext.jsx';

const queryClient = new QueryClient();

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <QueryClientProvider client={queryClient}>
      <AuthProvider>
        <App />
      </AuthProvider>
    </QueryClientProvider>
  </React.StrictMode>,
);

```

**Checkpoint:** ☒ API infrastructure is ready

---

## Phase 7: Functional Authentication

### Step 7.1: Create Reusable Input Component

Create `src/components/Input.jsx` :

```
function Input({ label, type = "text", error, ...props }) {
  return (
    <div className="mb-4">
      <label className="block text-gray-700 mb-2">{label}</label>
      <input
        type={type}
        className={`w-full p-3 border rounded-md ${
          error ? 'border-red-500' : 'border-gray-300'
        }`}
        {...props}
      />
      {error && <p className="text-red-500 text-sm mt-1">{error}</p>}
    </div>
  );
}

export default Input;
```

### Step 7.2: Make Login Form Functional

Update `src/pages/LoginPage.jsx` :

```
import { useState } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import Layout from '../components/Layout';
import Input from '../components/Input';
import { useAuth } from '../context/AuthContext';
import { authAPI } from '../api/auth';

function LoginPage() {
  const [formData, setFormData] = useState({ username: '', password: '' });
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState('');
  const { login } = useAuth();
  const navigate = useNavigate();

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

  const handleSubmit = async (e) => {
```

```

e.preventDefault();
setLoading(true);
setError('');

try {
  const response = await authAPI.login(formData);
  login(response.user, response.token);
  navigate('/dashboard');
} catch (err) {
  setError('Login failed. Please check your credentials.');
```

```

} finally {
  setLoading(false);
}
};

return (
  <Layout>
    <div className="max-w-md mx-auto">
      <h2 className="text-3xl font-bold text-center mb-6">Login</h2>
      <form onSubmit={handleSubmit} className="bg-white p-6 rounded-lg shadow-md">
        {error && (
          <div className="bg-red-100 border border-red-400 text-red-700 px-4 py-3
rounded mb-4">
            {error}
          </div>
        )}

        <Input
          label="Username"
          name="username"
          value={formData.username}
          onChange={handleChange}
          placeholder="Enter username"
          required
        />

        <Input
          label="Password"
          type="password"
          name="password"
          value={formData.password}
          onChange={handleChange}
          placeholder="Enter password"
          required
        />

        <button
          type="submit"
          disabled={loading}
          className="w-full bg-blue-600 text-white p-3 rounded-md hover:bg-blue-
700 disabled:bg-blue-300"
        >

```

```

        {loading ? 'Logging in...' : 'Login'}
      </button>

      <p className="text-center mt-4 text-gray-600">
        Don't have an account? <Link to="/register" className="text-blue-600
        hover:underline">Register</Link>
      </p>
    </form>
  </div>
</Layout>
);
}

export default LoginPage;

```

### Step 7.3: Make Register Form Functional

**Update** `src/pages/RegisterPage.jsx` (similar pattern to login):

```

import { useState } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import Layout from '../components/Layout';
import Input from '../components/Input';
import { authAPI } from '../api/auth';

function RegisterPage() {
  const [formData, setFormData] = useState({
    username: '',
    email: '',
    password: ''
  });

  const [loading, setLoading] = useState(false);
  const [error, setError] = useState('');
  const navigate = useNavigate();

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

  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    setError('');

    try {
      await authAPI.register(formData);
      navigate('/login');
      // You could add a success message here
    }
  };
}

```

```

    } catch (err) {
      setError('Registration failed. Please try again.');
```

```

    } finally {
      setLoading(false);
    }
  };

  return (
    <Layout>
      <div className="max-w-md mx-auto">
        <h2 className="text-3xl font-bold text-center mb-6">Register</h2>
        <form onSubmit={handleSubmit} className="bg-white p-6 rounded-lg shadow-md">
          {error && (
            <div className="bg-red-100 border border-red-400 text-red-700 px-4 py-3
rounded mb-4">
              {error}
            </div>
          )}

          <Input
            label="Username"
            name="username"
            value={formData.username}
            onChange={handleChange}
            placeholder="Choose username"
            required
          />

          <Input
            label="Email"
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            placeholder="Enter email"
            required
          />

          <Input
            label="Password"
            type="password"
            name="password"
            value={formData.password}
            onChange={handleChange}
            placeholder="Create password"
            required
          />

          <button
            type="submit"
            disabled={loading}
            className="w-full bg-blue-600 text-white p-3 rounded-md hover:bg-blue-

```



```

700 disabled:bg-blue-300"
      >
        {loading ? 'Creating account...' : 'Register'}
      </button>

      <p className="text-center mt-4 text-gray-600">
        Already have an account? <Link to="/login" className="text-blue-600
hover:underline">Login</Link>
      </p>
    </form>
  </div>
</Layout>
);
}

export default RegisterPage;

```

**Checkpoint:** ☒ Users can register and login (if backend is running)

---

## Phase 8: URL Shortening

### Step 8.1: Make Homepage URL Form Functional

**Update** `src/pages/HomePage.jsx` :

```

import { useState } from 'react';
import Layout from '../components/Layout';
import { urlsAPI } from '../api/urls';

function HomePage() {
  const [url, setUrl] = useState('');
  const [shortUrl, setShortUrl] = useState('');
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (!url) return;

    setLoading(true);
    setError('');

    try {
      const response = await urlsAPI.shorten(url);
      setShortUrl(`${window.location.origin}/s/${response.shortCode}`);
    } catch (err) {
      setError('Failed to shorten URL. Please try again.');
```

```

const copyToClipboard = () => {
  navigator.clipboard.writeText(shortUrl);
  alert('Copied to clipboard!');
};

return (
  <Layout>
    <div className="text-center">
      <h1 className="text-4xl font-bold text-gray-800 mb-4">
        Shorten Your URLs
      </h1>
      <p className="text-gray-600 mb-8">
        Create short, memorable links that are easy to share
      </p>

      <div className="bg-white p-6 rounded-lg shadow-md max-w-md mx-auto">
        <form onSubmit={handleSubmit}>
          <input
            type="url"
            value={url}
            onChange={(e) => setUrl(e.target.value)}
            placeholder="Enter your long URL here"
            className="w-full p-3 border border-gray-300 rounded-md mb-4"
            required
          />
          <button
            type="submit"
            disabled={loading}
            className="w-full bg-blue-600 text-white p-3 rounded-md hover:bg-blue-
700 disabled:bg-blue-300"
          >
            {loading ? 'Shortening...' : 'Shorten URL'}
          </button>
        </form>

        {error && (
          <div className="bg-red-100 border border-red-400 text-red-700 px-4 py-3
rounded mt-4">
            {error}
          </div>
        )}

        {shortUrl && (
          <div className="mt-6 p-4 bg-green-50 border border-green-200 rounded">
            <p className="text-green-800 font-semibold mb-2">Your short URL:</p>
            <div className="flex items-center gap-2">
              <input
                type="text"
                value={shortUrl}
                readOnly
                className="flex-1 p-2 border border-gray-300 rounded bg-white"
              />
            </div>
          </div>
        )}
      </div>
    </div>
  </Layout>
);

```

```

        />
        <button
          onClick={copyToClipboard}
          className="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-
600"
        >
          Copy
        </button>
      </div>
    </div>
  )}
</div>
</div>
</Layout>
);
}

export default HomePage;

```

**Checkpoint:** ☒ Basic URL shortening works

## Phase 9: Dashboard with URL Management

### Step 9.1: Create URL List Component

Create `src/components/UrlItem.jsx` :

```

import { useState } from 'react';

function UrlItem({ url }) {
  const [copied, setCopied] = useState(false);

  const copyToClipboard = () => {
    navigator.clipboard.writeText(url.shortUrl);
    setCopied(true);
    setTimeout(() => setCopied(false), 2000);
  };

  return (
    <div className="bg-white p-4 rounded-lg shadow-md border">
      <div className="flex justify-between items-start mb-2">
        <div className="flex-1">
          <p className="font-semibold text-blue-600 mb-1">{url.shortUrl}</p>
          <p className="text-gray-600 text-sm break-all">{url.originalUrl}</p>
        </div>
        <button
          onClick={copyToClipboard}
          className={`ml-4 px-3 py-1 rounded text-sm ${
            copied
              ? 'bg-green-500 text-white'
              : 'bg-blue-500 text-white hover:bg-blue-600'
            }`
        >

```

```

        `}`
      >
      {copied ? 'Copied!' : 'Copy'}
    </button>
  </div>

  <div className="flex justify-between items-center text-sm text-gray-500">
    <span>Clicks: {url.clickCount || 0}</span>
    <span>Created: {new Date(url.createdAt).toLocaleDateString()}</span>
  </div>
</div>
);
}

export default UrlItem;

```

## Step 9.2: Update Dashboard with Real Data

Update `src/pages/DashboardPage.jsx` :

```

import { useState, useEffect } from 'react';
import { useQuery } from 'react-query';
import Layout from '../components/Layout';
import Input from '../components/Input';
import UrlItem from '../components/UrlItem';
import { urlsAPI } from '../api/urls';
import { useAuth } from '../context/AuthContext';

function DashboardPage() {
  const [newUrl, setNewUrl] = useState('');
  const [creating, setCreating] = useState(false);
  const { isAuthenticated } = useAuth();

  const { data: urls = [], isLoading, refetch } = useQuery(
    'myUrls',
    urlsAPI.getMyUrls,
    {
      enabled: isAuthenticated, // Only fetch if logged in
    }
  );

  const handlecreateUrl = async (e) => {
    e.preventDefault();
    if (!newUrl) return;

    setCreating(true);
    try {
      await urlsAPI.shorten(newUrl);
      setNewUrl('');
      refetch(); // Refresh the list
    } catch (err) {

```



**Checkpoint:**  Dashboard shows user's URLs and allows creating new ones

```
import { Navigate } from 'react-router-dom';
import { useAuth } from '../context/AuthContext';

function ProtectedRoute({ children, requireAuth = true }) {
  const { isAuthenticated } = useAuth();

  if (requireAuth && !isAuthenticated) {
    return <Navigate to="/login" replace />;
  }

  if (!requireAuth && isAuthenticated) {
```

```

    return <Navigate to="/dashboard" replace />;
  }

  return children;
}

export default ProtectedRoute;

```

## Step 10.2: Update Routes with Protection

Update `src/App.jsx` :

```

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import HomePage from './pages/HomePage';
import LoginPage from './pages/LoginPage';
import RegisterPage from './pages/RegisterPage';
import DashboardPage from './pages/DashboardPage';
import ProtectedRoute from './components/ProtectedRoute';

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<HomePage />} />

        {/* Public only routes - redirect to dashboard if logged in */}
        <Route
          path="/login"
          element={
            <ProtectedRoute requireAuth={false}>
              <LoginPage />
            </ProtectedRoute>
          }
        />
        <Route
          path="/register"
          element={
            <ProtectedRoute requireAuth={false}>
              <RegisterPage />
            </ProtectedRoute>
          }
        />

        {/* Private routes - require authentication */}
        <Route
          path="/dashboard"
          element={
            <ProtectedRoute requireAuth={true}>
              <DashboardPage />
            </ProtectedRoute>
          }
        />
      </Routes>
    </Router>
  );
}

```

```
        />
      </Routes>
    </Router>
  );
}

export default App;
```

**Checkpoint:** ☒ Routes are protected based on authentication

---

## Phase 11: Analytics and Charts

### Step 11.1: Create Simple Chart Component

Create `src/components/Chart.jsx` :

```
import { Bar } from 'react-chartjs-2';
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  BarElement,
  Title,
  Tooltip,
  Legend,
} from 'chart.js';

ChartJS.register(
  CategoryScale,
  LinearScale,
  BarElement,
  Title,
  Tooltip,
  Legend
);

function Chart({ data }) {
  const chartData = {
    labels: data.map(item => item.date),
    datasets: [
      {
        label: 'Clicks',
        data: data.map(item => item.count),
        backgroundColor: 'rgba(59, 130, 246, 0.5)',
        borderColor: 'rgb(59, 130, 246)',
        borderWidth: 1,
      },
    ],
  };

  const options = {
```



```

    responsive: true,
    plugins: {
      legend: {
        position: 'top',
      },
      title: {
        display: true,
        text: 'Click Analytics',
      },
    },
  },
};

return <Bar data={chartData} options={options} />;
}

export default Chart;

```

## Step 11.2: Add Analytics to URL Items

**Update** `src/components/UrlItem.jsx` :

```

import { useState } from 'react';
import { useQuery } from 'react-query';
import Chart from './Chart';
import { urlsAPI } from '../api/urls';

function UrlItem({ url }) {
  const [copied, setCopied] = useState(false);
  const [showAnalytics, setShowAnalytics] = useState(false);

  const { data: analytics, isLoading: analyticsLoading } = useQuery(
    ['analytics', url.shortCode],
    () => urlsAPI.getAnalytics(url.shortCode),
    {
      enabled: showAnalytics, // Only fetch when analytics are shown
    }
  );

  const copyToClipboard = () => {
    navigator.clipboard.writeText(url.shortUrl);
    setCopied(true);
    setTimeout(() => setCopied(false), 2000);
  };

  return (
    <div className="bg-white p-4 rounded-lg shadow-md border">
      <div className="flex justify-between items-start mb-2">
        <div className="flex-1">
          <p className="font-semibold text-blue-600 mb-1">{url.shortUrl}</p>
          <p className="text-gray-600 text-sm break-all">{url.originalUrl}</p>
        </div>

```

```

<div className="flex gap-2 ml-4">
  <button
    onClick={copyToClipboard}
    className={`px-3 py-1 rounded text-sm ${
      copied
        ? 'bg-green-500 text-white'
        : 'bg-blue-500 text-white hover:bg-blue-600'
      }`}
  >
    {copied ? 'Copied!' : 'Copy'}
  </button>
  <button
    onClick={() => setShowAnalytics(!showAnalytics)}
    className="px-3 py-1 rounded text-sm bg-purple-500 text-white hover:bg-
purple-600"
  >
    {showAnalytics ? 'Hide' : 'Analytics'}
  </button>
</div>
</div>

<div className="flex justify-between items-center text-sm text-gray-500">
  <span>Clicks: {url.clickCount || 0}</span>
  <span>Created: {new Date(url.createdAt).toLocaleDateString()}</span>
</div>

{showAnalytics && (
  <div className="mt-4 pt-4 border-t border-gray-200">
    {analyticsLoading ? (
      <p className="text-center text-gray-500">Loading analytics...</p>
    ) : analytics && analytics.length > 0 ? (
      <Chart data={analytics} />
    ) : (
      <p className="text-center text-gray-500">No analytics data available</p>
    )}
  </div>
)}
</div>
);
}

export default UrlItem;

```

**Checkpoint:** ☒ Users can view analytics for their URLs

---

## Phase 12: URL Redirection

### Step 12.1: Create Redirect Handler

Create `src/pages/RedirectPage.jsx` :

```
import { useEffect } from 'react';
import { useParams } from 'react-router-dom';

function RedirectPage() {
  const { shortCode } = useParams();

  useEffect(() => {
    // Redirect to backend which will handle the actual redirect
    // and count the click
    window.location.href = `http://localhost:8080/${shortCode}`;
  }, [shortCode]);

  return (
    <div className="min-h-screen flex items-center justify-center">
      <div className="text-center">
        <div className="animate-spin rounded-full h-32 w-32 border-b-2 border-blue-600 mx-auto mb-4"></div>
        <p className="text-gray-600">Redirecting...</p>
      </div>
    </div>
  );
}

export default RedirectPage;
```

## Step 12.2: Add Redirect Route

Update `src/App.jsx` :

```
import RedirectPage from './pages/RedirectPage';

// Add this route:
<Route path="/s/:shortCode" element={<RedirectPage />} />
```

**Checkpoint:** ☒ Short URLs redirect properly

---

## Phase 13: Polish and User Experience

### Step 13.1: Add Loading States

Create `src/components/LoadingSpinner.jsx` :

```
function LoadingSpinner({ message = "Loading..." }) {
  return (
    <div className="text-center py-8">
      <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-blue-600 mx-auto mb-4"></div>
      <p className="text-gray-600">{message}</p>
    </div>
  );
}
```

```
}

export default LoadingSpinner;
```

### Step 13.2: Add Error Handling

Create `src/components/ErrorMessage.jsx` :

```
function ErrorMessage({ message, onRetry }) {
  return (
    <div className="bg-red-50 border border-red-200 rounded-lg p-4 text-center">
      <p className="text-red-800 mb-3">{message}</p>
      {onRetry && (
        <button
          onClick={onRetry}
          className="bg-red-600 text-white px-4 py-2 rounded hover:bg-red-700"
        >
          Try Again
        </button>
      )}
    </div>
  );
}

export default ErrorMessage;
```

### Step 13.3: Add Notifications

Install and setup react-hot-toast:

```
// In your components, replace alert() with:
import toast from 'react-hot-toast';

// Success
toast.success('URL copied to clipboard!');

// Error
toast.error('Failed to create short URL');

// Loading
toast.loading('Creating short URL...');
```

Add Toaster to App.jsx:

```
import { Toaster } from 'react-hot-toast';

function App() {
  return (
    <Router>
      <Routes>
```

```
        {/* your routes */}  
      </Routes>  
      <Toaster position="bottom-center" />  
    </Router>  
  );  
}
```

---

## Development Best Practices

### 1. Start Simple, Add Complexity

```
// Step 1: Hardcoded data  
const urls = [{ id: 1, shortUrl: 'abc123', originalUrl: 'https://example.com' }];  
  
// Step 2: State management  
const [urls, setUrls] = useState([]);  
  
// Step 3: API integration  
const { data: urls } = useQuery('urls', fetchUrls);
```

### 2. Component Composition

```
// Instead of one giant component:  
function Dashboard() {  
  return (  
    <Layout>  
      <DashboardHeader />  
      <CreateUrlForm />  
      <UrlList />  
      <Analytics />  
    </Layout>  
  );  
}
```

### 3. Custom Hooks for Logic

```
// Extract complex logic into custom hooks  
function useUrlShortener() {  
  const [loading, setLoading] = useState(false);  
  
  const shortenUrl = async (url) => {  
    setLoading(true);  
    try {  
      const result = await urlsAPI.shorten(url);  
      return result;  
    } finally {  
      setLoading(false);  
    }  
  }  
}
```

```
};

return { shortenUrl, loading };
}
```

## 4. Error Boundaries

```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true };
  }

  render() {
    if (this.state.hasError) {
      return <h1>Something went wrong.</h1>;
    }
    return this.props.children;
  }
}
```

---

# Testing Your Application

## Manual Testing Checklist

### Authentication Flow

- ☐ Can register new account
- ☐ Can login with correct credentials
- ☐ Cannot login with wrong credentials
- ☐ Redirected to dashboard after login
- ☐ Cannot access dashboard without login
- ☐ Can logout successfully

### URL Shortening

- ☐ Can create short URL from homepage
- ☐ Can create short URL from dashboard
- ☐ Short URL is copyable
- ☐ Short URL redirects correctly
- ☐ Invalid URLs are rejected

### Dashboard

- ☐ Shows list of user's URLs

- ☐ Shows correct statistics
- ☐ Can view analytics for each URL
- ☐ Loading states work properly
- ☐ Error states handled gracefully

## Debugging Tips

### Common Issues and Solutions

**Issue:** "My API calls aren't working" **Debug Steps:**

1. Check browser Network tab
2. Verify backend is running
3. Check API endpoint URLs
4. Verify request headers

**Issue:** "My state isn't updating" **Debug Steps:**

1. Add console.log to see state changes
2. Check if you're mutating state directly
3. Verify useEffect dependencies

**Issue:** "My components aren't re-rendering" **Debug Steps:**

1. Check if props are changing
2. Verify state updates are correct
3. Look for missing dependencies in useEffect

---

## Environment Setup

### Step 1: Create Environment File

Create `.env` in project root:

```
VITE_BACKEND_URL=http://localhost:8080
VITE_FRONTEND_URL=http://localhost:5173
```

### Step 2: Update API Configuration

Update `src/api/api.js` :

```
import axios from 'axios';

const api = axios.create({
  baseURL: import.meta.env.VITE_BACKEND_URL,
});
```

### Step 3: Backend Requirements

Your backend needs these endpoints:

```
POST /api/auth/register
POST /api/auth/login
```

```
POST /api/urls/shorten
GET /api/urls/myurls
GET /api/urls/analytics/:shortCode
GET /:shortCode (redirect endpoint)
```

---

## Deployment Considerations

### Frontend Deployment (Netlify/Vercel)

1. Build the project: `npm run build`
2. Deploy the `dist` folder
3. Set environment variables in hosting platform
4. Configure redirects for React Router

### Environment Variables for Production








```
VITE_BACKEND_URL=https://your-backend-api.com
VITE_FRONTEND_URL=https://your-frontend.com
```

---







## What You've Learned

By following this tutorial, you've learned:







### React Concepts

-  Components and JSX
-  Props and state
-  Hooks (useState, useEffect, custom hooks)
-  Context API for global state
-  Event handling
-  Conditional rendering
-  Form handling

### Modern Web Development

-  API integration with Axios
-  Data fetching with React Query
-  Routing with React Router
-  Authentication patterns
-  State management strategies
-  Component composition

### Professional Practices

-  Project structure organization
  -  Error handling
  -  Loading states
  -  User experience considerations
  -  Code reusability
  -  Separation of concerns
-



## Next Steps for Advanced Features

### 1. Custom Short Codes

Allow users to choose their own short codes:

```
<input
  placeholder="Custom code (optional)"
  value={customCode}
  onChange={(e) => setCustomCode(e.target.value)}
/>
```

### 2. URL Expiration

Add expiration dates to URLs:

```
<input
  type="date"
  value={expirationDate}
  onChange={(e) => setExpirationDate(e.target.value)}
/>
```

### 3. QR Code Generation

Generate QR codes for URLs:

```
import QRCode from 'qrcode.react';

<QRCode value={shortUrl} size={128} />
```

### 4. Bulk URL Import

Allow CSV upload for multiple URLs:

```
<input
  type="file"
  accept=".csv"
  onChange={handleFileUpload}
/>
```

### 5. Advanced Analytics

- Geographic data
- Device information
- Referrer tracking
- Time-based analytics

Remember: **Build one feature at a time.** Don't try to implement everything at once. Master the basics first, then gradually add more advanced features.

The key to becoming a good developer is **practice and patience**. Start with this foundation and keep building!