# **Design Document for React URL Shortener Web App**

## **Project Overview**

This project is a fully functional, responsive **React-based URL Shortener Web Application**. The app allows users to:

* Input a valid URL and optional expiry time.
* Generate a shortened version of the URL.
* Store and display a list of shortened URLs on the frontend.

## **Key Design & Architectural Decisions**

### **1. Technology Stack**

|  |  |  |
| --- | --- | --- |
| **Component** | **Tool Used** | **Reasoning** |
| Frontend | React (Vite + JSX) | Lightweight, fast build system with modular UI development using JSX. |
| Styling | CSS | Simple styling used directly with component-specific files like App.css. |
| State Management | useState | React hook for managing form and URL list state locally. |

## **Component Structure**

### **src/**

* App.jsx: Main application logic, state management, form handling, and rendering.
* App.css: Contains styling for the layout and user interface.
* assets/: Can be used for logos or any future assets.

## **Routing Strategy**

As this is a single-page application, routing is not implemented. However, this can be extended in the future using React Router for:

* Redirecting shortened URLs (if backend or Firebase is added).
* Dedicated page for analytics or history.

## **URL Handling Logic**

### **➕ Validation**

A regular expression is used to validate if a given string is a properly formatted URL:

const isValidUrl = (string) => {  
 const res = string.match(/(https?:\/\/[^\s]+)/g);  
 return res !== null;  
};

### **🔗 Shortening Logic**

For the frontend version, we simulate shortening by creating random strings. Backend integration can be added later for real-world shortening services.

## **State Management**

const [url, setUrl] = useState('');  
const [expiry, setExpiry] = useState('');  
const [shortenedUrls, setShortenedUrls] = useState([]);

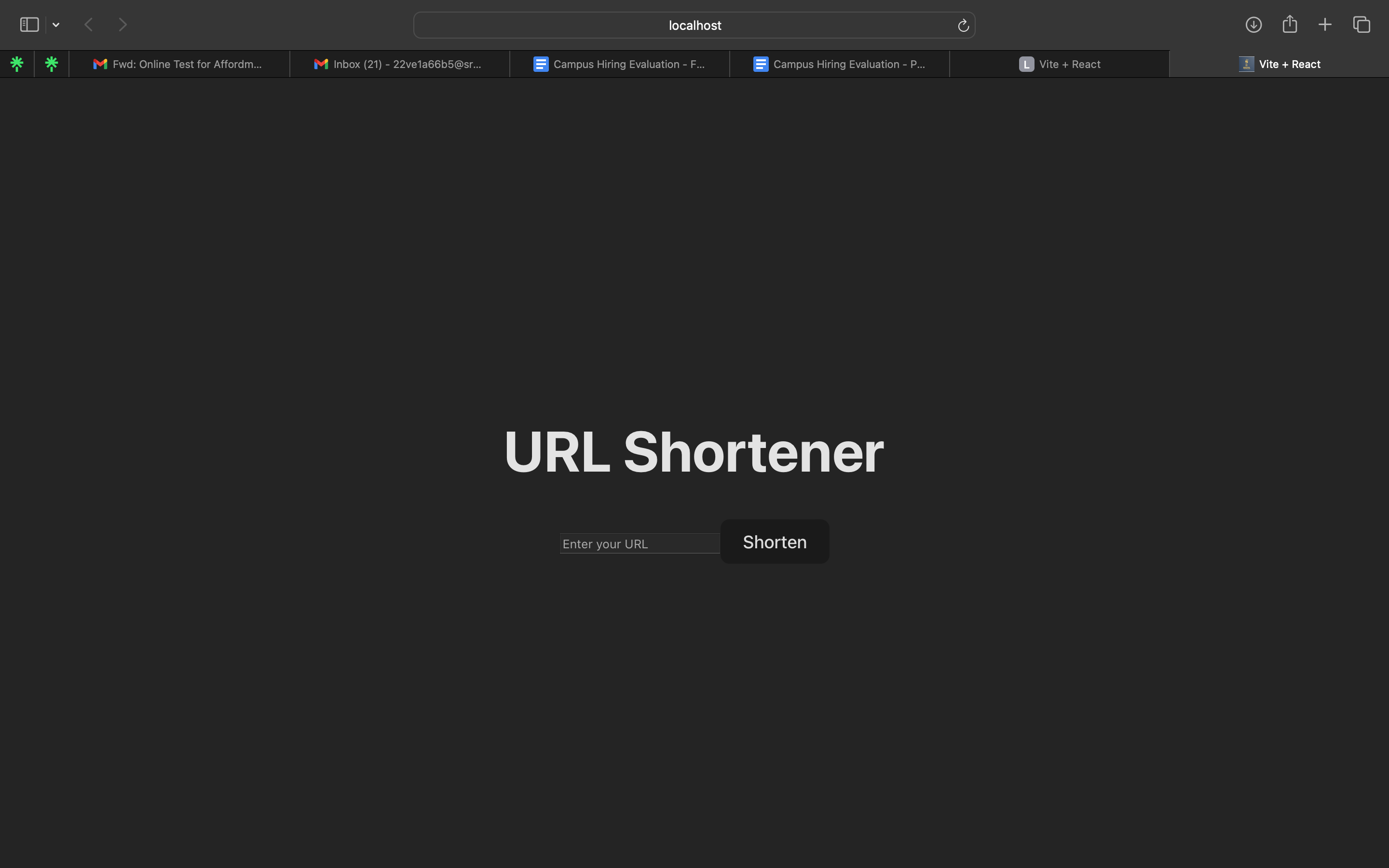
Each shortened URL is stored in shortenedUrls with optional expiry time and shown to the user dynamically.

## **Error Handling**

* Alerts are triggered when the input is empty or invalid.
* Graceful fallbacks and clear messaging improve user experience.

## **Expiry Feature**

While expiry time is stored in state, backend support would be required to enforce real expiration. For now, it’s displayed for frontend UI demo purposes.

 **Assumptions Made**

* No backend is required during evaluation.
* Data persistence is not required; data is lost on page refresh.
* Users will enter valid HTTPS URLs.
* App runs in modern browsers.

## **Testing & Evaluation**

* Manual testing performed on local development server (npm run dev).
* Validations checked for:
  + Empty URL
  + Invalid URL format
* Git initialized and ready for push with standard commands.

## **Conclusion**

This React application was developed with:

* Clean modular code
* Clear state separation
* Easy expandability for backend or routing
* Focus on user-friendly, intuitive interface

