**Lost and Found with PWA**

**Build Your First Web App Step-by-Step!**

**✨ Preface**

This guide was created for aspiring developers who want to learn how to build a real-world Progressive Web App (PWA) using HTML, CSS, JavaScript, and LocalStorage—without needing any backend server. Together, we will walk through every step of building a "Lost and Found" platform that works both online and offline, is installable like an app, and includes features like user login, admin panel, item reporting, and claim handling.

**✉️ Table of Contents**

1. Introduction to PWAs and Project Goals
2. Setting Up Your Project Folder
3. Creating the Basic HTML Pages
4. Styling with CSS for a Clean UI
5. Adding Interactivity with JavaScript
6. Storing Data in LocalStorage (Users, Items, Claims)
7. Building the Signup and Login System
8. User Dashboard: Reporting and Searching Items
9. Admin Dashboard: Managing Claims
10. Making the App Installable (Manifest + Icons)
11. Adding Offline Support (Service Worker)
12. Protecting Pages Based on User Role
13. Toast Notifications and UI Polishing
14. Deploying the App to GitHub Pages
15. Installing the App on Phone/Desktop
16. Final Thoughts and What You Learned

**✨ Chapter 1: What is a PWA?**

**Progressive Web Apps** combine the best of web and mobile apps. They work offline, can be installed on your device, and run fullscreen like native apps. We chose PWA to make our Lost and Found system accessible from both desktops and smartphones, without needing an app store.

**Key Benefits:**

* Works offline
* Installable
* Fast loading
* Uses browser storage (no server required)

**📚 Chapter 2: Setting Up Your Project**

Create a folder named lost-found-pwa. Inside, add these files:

/index.html

/signup.html

/login.html

/admin.html

/style.css

/index.js

/signup.js

/login.js

/admin.js

/manifest.json

/service-worker.js

/icon-192.png

/icon-512.png

Use a code editor like VS Code.

**🖋️ Chapter 3: Designing the HTML Pages**

Each page serves a role:

* **index.html** → User dashboard (report, search, claim)
* **signup.html** → Create account (user or admin)
* **login.html** → Sign in
* **admin.html** → Admin panel to approve claims

Every page uses <div class="container"> for centered layout and consistent design.

**💄 Chapter 4: Styling with CSS**

Create a style.css that includes:

* Basic layout
* Form styling
* Button hover effects
* Toast notification styles
* Responsive layout

**✨ Chapter 5: Adding JavaScript Logic**

Each JS file manages a specific page:

* **index.js** → handles report, search, and claim logic
* **signup.js** → handles user registration and validation
* **login.js** → authenticates user and redirects to the right page
* **admin.js** → loads pending claims and lets admin approve/reject them

Use localStorage to save:

* users (email, password, role)
* loggedInUser (session tracking)
* lostItems (item name, color, location, description)
* claims (item ID + user email)

**🔒 Chapter 6: Handling Users & LocalStorage**

Signups are stored like this:

[

{ email: "user1@example.com", password: "12345", role: "user" },

{ email: "admin1@example.com", password: "adminpass", role: "admin" }

]

Claims are stored in a similar JSON structure with timestamps.

**🔧 Chapter 7: Signup & Login Flow**

* Signup form validates passwords and saves to users
* Login form checks credentials and stores the loggedInUser
* Redirect to:
  + index.html for users
  + admin.html for admins

**📍 Chapter 8: User Dashboard Features**

Users can:

* Report an item (with name, description, location, color, optional photo)
* Search items by name
* View search results
* Claim items with a button (claim goes to claims array)

**📅 Chapter 9: Admin Panel**

Admins see pending claims like:

Item: Laptop

Claimed by: user1@example.com

[Approve] [Reject]

They can approve/reject. Approved items are marked as claimed: true in lostItems.

**🌍 Chapter 10: Make It a Real App (Manifest)**

Create manifest.json:

{

"name": "Lost and Found",

"short\_name": "LostFound",

"start\_url": ".",

"display": "standalone",

"background\_color": "#ffffff",

"theme\_color": "#0078D7",

"icons": [

{ "src": "icon-192.png", "sizes": "192x192", "type": "image/png" },

{ "src": "icon-512.png", "sizes": "512x512", "type": "image/png" }

]

}

Link it in <head> of all HTML files.

**🔄 Chapter 11: Add Offline Support (Service Worker)**

Create service-worker.js:

self.addEventListener('install', event => {

event.waitUntil(

caches.open('lost-found-cache-v1').then(cache => {

return cache.addAll([

'/', '/index.html', '/style.css', '/index.js',

'/manifest.json', '/icon-192.png', '/icon-512.png'

]);

})

);

});

self.addEventListener('fetch', event => {

event.respondWith(

caches.match(event.request).then(response => response || fetch(event.request))

);

});

Register it in JS:

navigator.serviceWorker.register('service-worker.js');

**🔐 Chapter 12: Page Protection**

In index.js, add:

const currentUser = JSON.parse(localStorage.getItem('loggedInUser'));

if (!currentUser) window.location.href = 'login.html';

Same for admin panel with extra role === 'admin' check.

**✨ Chapter 13: Toasts & Polishing**

Add a <div id="toast"></div> in HTML. Show messages like:

function showToast(msg) {

const toast = document.getElementById('toast');

toast.textContent = msg;

toast.classList.add('show');

setTimeout(() => toast.classList.remove('show'), 3000);

}

**🚀 Chapter 14: Deploy to GitHub Pages**

1. Create a new public repo
2. Push your project with:

git init

git add .

git commit -m "Initial commit"

git branch -M main

git remote add origin <your-github-url>

git push -u origin main

1. Go to Settings > Pages > Branch = main, Folder = /root
2. Get your live link like:

https://yourusername.github.io/lost-found-pwa/

**📱 Chapter 15: Installing the App**

On mobile or desktop:

* Visit the link
* You’ll see the Install button (PWA prompt)
* Install it like a real app
* It now works offline and shows on home screen/app drawer

**🎉 Chapter 16: What You Learned**

* How to use HTML, CSS, JS to build real projects
* Use localStorage as a mini database
* Add login, role-based access, admin panel
* Make a real PWA with offline support
* Deploy your project live to the world

You didn’t just build an app. You built a launchpad to real-world development!

**🏆 Congratulations!**

Welcome to the world of real developers. You have created something meaningful, modern, and powerful. Now take this knowledge and build more amazing things!

**🎉 Made with ❤️ by ChatGPT for an awesome developer.**