■ Lost and Found Frontend Project Export

React Frontend Application with Python Backend

■ Project File Structure:

- ■ Lost_and_Found_Frontend_Export.pdf
- ■ README.md
  - ■ app.py
- ■ hash.py
- ■ package.json
- ■ postcss.config.js
  - ■ index.html
  - ■ manifest.json
  - ■ robots.txt
- ■ script.py
  - ■ App.css
  - ■ App.js
  - ■ App.test.js
    - ■ AdminTable.js
    - ■ ClaimsTable.js
    - ■ FeedbackForm.js
    - ■ FlashMessage.js
    - ■ Header.js
    - ■ ItemTable.js
    - ■ MessageList.js
  - ■ index.css
  - ■ index.js
    - ■ AdminPage.js
    - ■ HomePage.js
    - ■ ItemRegisterPage.js
    - ■ ItemsPage.js
    - ■ LoginPage.js
    - ■ RegisterPage.js
  - ■ reportWebVitals.js
  - ■ setupTests.js
- ■ tailwind.config.js

■ File: Lost_and_Found_Frontend_Export.pdf

```
================================================================================
[Binary file - .pdf format]

--------------------------------------------------------------------------------
```

■ File: README.md

```
================================================================================
# Getting Started with Create React App

This project was bootstrapped with [Create React App](https://github.com/facebook/create-react-app).

## Available Scripts

In the project directory, you can run:

### `npm start`

Runs the app in the development mode.\
Open [http://localhost:3000](http://localhost:3000) to view it in your browser.

The page will reload when you make changes.\
You may also see any lint errors in the console.

### `npm test`

Launches the test runner in the interactive watch mode.\
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more i

### `npm run build`

Builds the app for production to the `build` folder.\
It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.\
Your app is ready to be deployed!

See the section about [deployment](https://facebook.github.io/create-react-app/docs/deployment) for more informa

### `npm run eject`

**Note: this is a one-way operation. Once you `eject`, you can't go back!**

If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command

Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc)

You don't have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and yo

## Learn More

You can learn more in the [Create React App documentation](https://facebook.github.io/create-react-app/docs/gett

To learn React, check out the [React documentation](https://reactjs.org/).

### Code Splitting

This section has moved here: [https://facebook.github.io/create-react-app/docs/code-splitting](https://facebook.

### Analyzing the Bundle Size

This section has moved here: [https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size](https:

### Making a Progressive Web App

This section has moved here: [https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app](htt

### Advanced Configuration

This section has moved here: [https://facebook.github.io/create-react-app/docs/advanced-configuration](https://f

### Deployment
```

```
This section has moved here: [https://facebook.github.io/create-react-app/docs/deployment](https://facebook.github

### `npm run build` fails to minify

This section has moved here: [https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fai

--------------------------------------------------------------------------------
```

■ File: backend\app.py

```
================================================================================

--------------------------------------------------------------------------------
```

■ File: hash.py

```
================================================================================
  1: import bcrypt
  2:
  3: # Normal password
  4: password = input("Admin@123").encode('utf-8')
  5:
  6: # Generate a salt
  7: salt = bcrypt.gensalt()
  8:
  9: # Hash the password
 10: hashed_password = bcrypt.hashpw(password, salt)
 11:
 12: print("Bcrypt Hash:", hashed_password.decode('utf-8'))

--------------------------------------------------------------------------------
```

■■ File: package.json

```
================================================================================
  1: {
  2:   "name": "lost-and-found-frontend",
  3:   "version": "0.1.0",
  4:   "private": true,
  5:   "dependencies": {
  6:     "@testing-library/dom": "^10.4.0",
  7:     "@testing-library/jest-dom": "^6.6.3",
  8:     "@testing-library/react": "^16.3.0",
  9:     "@testing-library/user-event": "^13.5.0",
 10:     "react": "^19.1.0",
 11:     "react-dom": "^19.1.0",
 12:     "react-router-dom": "^7.6.3",
 13:     "react-scripts": "5.0.1",
 14:     "web-vitals": "^2.1.4"
 15:   },
 16:   "scripts": {
 17:     "start": "react-scripts start",
 18:     "build": "react-scripts build",
 19:     "test": "react-scripts test",
 20:     "eject": "react-scripts eject"
 21:   },
 22:   "eslintConfig": {
 23:     "extends": [
 24:       "react-app",
 25:       "react-app/jest"
 26:     ]
 27:   },
 28:   "browserslist": {
 29:     "production": [
 30:       ">0.2%",
 31:       "not dead",
 32:       "not op_mini all"
 33:     ],
 34:     "development": [
 35:       "last 1 chrome version",
 36:       "last 1 firefox version",
 37:       "last 1 safari version"
 38:     ]
 39:   },
 40:   "devDependencies": {
 41:     "autoprefixer": "^10.4.21",
```

```
 42:     "postcss": "^8.5.6",
 43:     "tailwindcss": "^3.4.17"
 44:   }
 45: }
```

■ File: postcss.config.js

```
  1: module.exports = {
  2:   plugins: {
  3:     tailwindcss: {},
  4:     autoprefixer: {},
  5:   },
  6: }
```

■ File: public\index.html

```
  1: <!DOCTYPE html>
  2: <html lang="en">
  3:   <head>
  4:     <meta charset="utf-8" />
  5:     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  6:     <meta name="viewport" content="width=device-width, initial-scale=1" />
  7:     <meta name="theme-color" content="#000000" />
  8:     <meta
  9:       name="description"
 10:       content="Web site created using create-react-app"
 11:     />
 12:     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
 13:     <!--
 14:       manifest.json provides metadata used when your web app is installed on a
 15:       user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
 16:     -->
 17:     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
 18:     <!--
 19:       Notice the use of %PUBLIC_URL% in the tags above.
 20:       It will be replaced with the URL of the `public` folder during the build.
 21:       Only files inside the `public` folder can be referenced from the HTML.
 22:
 23:       Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
 24:       work correctly both with client-side routing and a non-root public URL.
 25:       Learn how to configure a non-root public URL by running `npm run build`.
 26:     -->
 27:     <title>React App</title>
 28:   </head>
 29:   <body>
 30:     <noscript>You need to enable JavaScript to run this app.</noscript>
 31:     <div id="root"></div>
 32:     <!--
 33:       This HTML file is a template.
 34:       If you open it directly in the browser, you will see an empty page.
 35:
 36:       You can add webfonts, meta tags, or analytics to this file.
 37:       The build step will place the bundled scripts into the <body> tag.
 38:
 39:       To begin the development, run `npm start` or `yarn start`.
 40:       To create a production bundle, use `npm run build` or `yarn build`.
 41:     -->
 42:   </body>
 43: </html>
```

■■ File: public\manifest.json

```
  1: {
  2:   "short_name": "React App",
  3:   "name": "Create React App Sample",
  4:   "icons": [
  5:     {
```

```
  6:       "src": "favicon.ico",
  7:       "sizes": "64x64 32x32 24x24 16x16",
  8:       "type": "image/x-icon"
  9:     },
 10:     {
 11:       "src": "logo192.png",
 12:       "type": "image/png",
 13:       "sizes": "192x192"
 14:     },
 15:     {
 16:       "src": "logo512.png",
 17:       "type": "image/png",
 18:       "sizes": "512x512"
 19:     }
 20:   ],
 21:   "start_url": ".",
 22:   "display": "standalone",
 23:   "theme_color": "#000000",
 24:   "background_color": "#ffffff"
 25: }
```

■ File: public\robots.txt

```
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow:
```

■ File: script.py

```
  1: import os
  2: from reportlab.lib.pagesizes import A4
  3: from reportlab.lib.units import mm
  4: from reportlab.pdfgen import canvas
  5:
  6: def get_code_files(directory, excluded_files=None, excluded_dirs=None):
  7:     """Fetch all project files except specified exclusions."""
  8:     if excluded_files is None:
  9:         excluded_files = {
 10:             '__pycache__',
 11:             '.pyc',
 12:             '.pyo',
 13:             '.DS_Store',
 14:             'Thumbs.db',
 15:             'Desktop.ini',
 16:             '.gitignore',
 17:             '.env',
 18:             'package-lock.json',  # Large auto-generated file
 19:             '.git'
 20:         }
 21:
 22:     if excluded_dirs is None:
 23:         excluded_dirs = {
 24:             '__pycache__',
 25:             '.git',
 26:             'venv',
 27:             'env',
 28:             '.venv',
 29:             'instance',
 30:             '.pytest_cache',
 31:             'logs',
 32:             'migrations',
 33:             'node_modules',  # Node.js dependencies
 34:             'build',         # React build output
 35:             'dist',          # Distribution files
 36:             '.next',         # Next.js build
 37:             'coverage'       # Test coverage
 38:         }
 39:
 40:     code_files = {}
```

```
41:
42:     for root, dirs, files in os.walk(directory):
43:         # Skip excluded directories
44:         dirs[:] = [d for d in dirs if d not in excluded_dirs]
45:
46:         # Skip if current directory is an excluded directory
47:         if any(excluded_dir in root.split(os.sep) for excluded_dir in excluded_dirs):
48:             continue
49:
50:         for file in files:
51:             # Skip excluded files
52:             if file in excluded_files or file.endswith(('.pyc', '.pyo', '.log')):
53:                 continue
54:
55:             file_path = os.path.join(root, file)
56:
57:             # Get file extension
58:             _, ext = os.path.splitext(file)
59:
60:             try:
61:                 # Try to read as text file first
62:                 if ext.lower() in {'.py', '.html', '.css', '.js', '.jsx', '.ts', '.tsx',
63:                                    '.sql', '.txt', '.md', '.json', '.xml', '.yaml', '.yml',
64:                                    '.config', '.ini', '.jinja2', '.j2', '.template',
65:                                    '.requirements', '.gitignore', '.env.example'}:
66:                     with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
67:                         code_files[file_path] = f.readlines()
68:                 elif file in {'requirements.txt', 'Dockerfile', 'Makefile', 'LICENSE',
69:                               'README', 'package.json', 'tailwind.config.js',
70:                               'postcss.config.js', 'hash.py'}:
71:                     # Handle files without extensions that are typically text
72:                     with open(file_path, "r", encoding="utf-8", errors="ignore") as f:
73:                         code_files[file_path] = f.readlines()
74:                 else:
75:                     # For binary files, just note them as binary
76:                     code_files[file_path] = [f"[Binary file - {ext} format]"]
77:
78:             except Exception as e:
79:                 print(f"? Error reading {file_path}: {e}")
80:                 code_files[file_path] = [f"[Error reading file: {str(e)}]"]
81:
82:     return code_files
83:
84:
85: def create_pdf(code_data, output_pdf="Lost_and_Found_Frontend_Export.pdf"):
86:     c = canvas.Canvas(output_pdf, pagesize=A4)
87:     width, height = A4
88:     margin = 20 * mm
89:     line_height = 10
90:     y = height - margin
91:
92:     # Title
93:     c.setFont("Helvetica-Bold", 16)
94:     c.drawString(margin, y, "? Lost and Found Frontend Project Export")
95:     y -= 2 * line_height
96:
97:     # Subtitle
98:     c.setFont("Helvetica", 12)
99:     c.drawString(margin, y, "React Frontend Application with Python Backend")
100:     y -= 2 * line_height
101:
102:     c.setFont("Helvetica-Bold", 12)
103:     c.drawString(margin, y, "? Project File Structure:")
104:     y -= 2 * line_height
105:
106:     file_paths = sorted(list(code_data.keys()))
107:
108:     # 1. File list with project structure
109:     c.setFont("Courier", 8)
110:     for path in file_paths:
111:         if y < margin:
112:             c.showPage()
113:             c.setFont("Courier", 8)
```

```
            y = height - margin

        display_path = os.path.relpath(path)
        # Add proper indentation for project structure
        indent = "  " * (display_path.count(os.sep))
        file_name = os.path.basename(display_path)
        c.drawString(margin, y, f"{indent}? {file_name}")
        y -= line_height

    # Add page break before code content
    c.showPage()
    y = height - margin

    # 2. File contents
    for file_path in file_paths:
        lines = code_data[file_path]
        print(f"? Adding: {file_path}")

        if y < margin + 3 * line_height:
            c.showPage()
            y = height - margin

        # File header
        rel_path = os.path.relpath(file_path)
        c.setFont("Helvetica-Bold", 12)

        # Add file type emoji based on extension
        file_emoji = get_file_emoji(rel_path)
        c.drawString(margin, y, f"{file_emoji} File: {rel_path}")
        y -= line_height

        # Add separator line
        c.setFont("Courier", 8)
        c.drawString(margin, y, "=" * 80)
        y -= line_height

        # File content
        for line_num, line in enumerate(lines, 1):
            if y < margin:
                c.showPage()
                c.setFont("Courier", 8)
                y = height - margin

            # Clean and truncate line
            line = line.strip("\n").encode("latin-1", "replace").decode("latin-1")

            # Add line numbers for code files
            if rel_path.endswith(('.py', '.html', '.css', '.js', '.jsx', '.ts', '.tsx', '.sql', '.json')):
                display_line = f"{line_num:3d}: {line[:280]}"
            else:
                display_line = line[:300]

            c.drawString(margin, y, display_line)
            y -= line_height

        # Add spacing between files
        y -= line_height
        if y > margin:
            c.setFont("Courier", 8)
            c.drawString(margin, y, "-" * 80)
            y -= 2 * line_height

    c.save()
    print(f"? PDF successfully created: {output_pdf}")
    print(f"? Total files processed: {len(code_data)}")


def get_file_emoji(file_path):
    """Return appropriate emoji based on file type."""
    ext = os.path.splitext(file_path)[1].lower()
    filename = os.path.basename(file_path).lower()

    if ext == '.py':
```

```
        return '?'
    elif ext in ['.html', '.htm']:
        return '?'
    elif ext in ['.css', '.scss', '.sass']:
        return '?'
    elif ext in ['.js', '.jsx']:
        return '?'
    elif ext in ['.ts', '.tsx']:
        return '?'
    elif ext == '.sql':
        return '??'
    elif ext in ['.json', '.yaml', '.yml']:
        return '??'
    elif ext in ['.txt', '.md']:
        return '?'
    elif filename == 'package.json':
        return '?'
    elif filename == 'requirements.txt':
        return '?'
    elif 'dockerfile' in filename:
        return '?'
    elif filename in ['tailwind.config.js', 'postcss.config.js']:
        return '?'
    elif filename == 'hash.py':
        return '?'
    elif ext in ['.ico', '.png', '.jpg', '.jpeg', '.gif', '.svg']:
        return '??'
    else:
        return '?'


def main():
    # Get the directory where this script is located
    script_dir = os.path.dirname(os.path.abspath(__file__))

    # Look for lost-and-found-frontend directory or use current directory
    if os.path.exists(os.path.join(script_dir, 'lost-and-found-frontend')):
        root_dir = os.path.join(script_dir, 'lost-and-found-frontend')
    elif os.path.exists(os.path.join(script_dir, 'src')) and os.path.exists(os.path.join(script_dir, 'public
        # If we're already in the React project directory
        root_dir = script_dir
    else:
        root_dir = script_dir

    # Files to exclude
    excluded_files = {
        '.DS_Store',
        'Thumbs.db',
        'Desktop.ini',
        '.gitignore',
        '.env',
        '*.pyc',
        '*.pyo',
        '*.log',
        'package-lock.json',  # Large auto-generated file
        'favicon.ico',        # Binary files
        'logo192.png',
        'logo512.png',
        'logo.svg'
    }

    # Directories to exclude
    excluded_dirs = {
        '__pycache__',
        '.git',
        'venv',
        'env',
        '.venv',
        'instance',
        '.pytest_cache',
        'logs',
        'migrations',
        'node_modules',       # Node.js dependencies
```

```
260:         'build',              # React build output
261:         'dist',               # Distribution files
262:         '.next',              # Next.js build
263:         'coverage'            # Test coverage
264:     }
265: 
266:     print("? Scanning Lost and Found frontend project files...")
267:     print(f"? Root directory: {root_dir}")
268: 
269:     code_files = get_code_files(root_dir, excluded_files, excluded_dirs)
270: 
271:     if not code_files:
272:         print("? No files found to process!")
273:         print("Make sure you're running this script in the correct directory.")
274:         return
275: 
276:     print(f"? Found {len(code_files)} files to include in PDF")
277: 
278:     # Display found files
279:     print("\n? Files to be included:")
280:     for file_path in sorted(code_files.keys()):
281:         rel_path = os.path.relpath(file_path)
282:         print(f"  - {rel_path}")
283: 
284:     create_pdf(code_files)
285: 
286:     print("\n? PDF export completed!")
287:     print("? You can now share this PDF with others or submit it as documentation.")
288: 
289: 
290: if __name__ == "__main__":
291:     main()
```

■ File: src\App.css

```
================================================================================
```

■ File: src\App.js

```
================================================================================
  1: import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
  2: import LoginPage from "./pages/LoginPage";
  3: import RegisterPage from "./pages/RegisterPage";
  4: import HomePage from "./pages/HomePage";
  5: import AdminPage from "./pages/AdminPage";
  6: import ItemRegisterPage from "./pages/ItemRegisterPage";
  7: import ItemsPage from "./pages/ItemsPage";
  8: import Header from "./components/Header";
  9: 
 10: function App() {
 11:   return (
 12:     <Router>
 13:       <Header />
 14:       <Routes>
 15:         <Route path="/" element={<LoginPage />} />
 16:         <Route path="/register" element={<RegisterPage />} />
 17:         <Route path="/home" element={<HomePage />} />
 18:         <Route path="/admin" element={<AdminPage />} />
 19:         <Route path="/register-item" element={<ItemRegisterPage />} />
 20:         <Route path="/lost-found-items" element={<ItemsPage />} />
 21:       </Routes>
 22:     </Router>
 23:   );
 24: }
 25: 
 26: export default App;
```

■ File: src\App.test.js

```
  1: import { render, screen } from '@testing-library/react';
  2: import App from './App';
  3:
  4: test('renders learn react link', () => {
  5:   render(<App />);
  6:   const linkElement = screen.getByText(/learn react/i);
  7:   expect(linkElement).toBeInTheDocument();
  8: });
```

■ File: src\components\AdminTable.js

```
  1: export default function AdminTable({ title, type, data, headers, rowRenderer }) {
  2:   return (
  3:     <section>
  4:       <h2 className="text-xl font-semibold mb-2 text-gray-700">{title}</h2>
  5:
  6:       {data.length === 0 ? (
  7:         <p className="text-gray-500 text-sm">No {type}s found.</p>
  8:       ) : (
  9:         <div className="overflow-x-auto border rounded">
 10:           <table className="min-w-full text-sm text-left">
 11:             <thead className="bg-gray-100 text-gray-600 uppercase">
 12:               <tr>
 13:                 {headers.map((h, idx) => (
 14:                   <th key={idx} className="px-4 py-2">
 15:                     {h}
 16:                   </th>
 17:                 ))}
 18:               </tr>
 19:             </thead>
 20:             <tbody>
 21:               {data.map((item, i) => (
 22:                 <tr
 23:                   key={i}
 24:                   className="border-t hover:bg-gray-50 transition-colors"
 25:                 >
 26:                   {rowRenderer(item).map((cell, j) => (
 27:                     <td key={j} className="px-4 py-2">
 28:                       {cell}
 29:                     </td>
 30:                   ))}
 31:                 </tr>
 32:               ))}
 33:             </tbody>
 34:           </table>
 35:         </div>
 36:       )}
 37:     </section>
 38:   );
 39: }
```

■ File: src\components\ClaimsTable.js

```
  1: export default function ClaimsTable({ claims }) {
  2:   if (!claims.length) return <p className="text-gray-500">No claims yet.</p>;
  3:
  4:   return (
  5:     <div className="overflow-x-auto border rounded">
  6:       <table className="min-w-full text-sm">
  7:         <thead className="bg-gray-100 text-gray-700 uppercase">
  8:           <tr>
  9:             <th>Item</th>
 10:             <th>Location</th>
 11:             <th>Claimed At</th>
 12:           </tr>
 13:         </thead>
 14:         <tbody>
```

```
15:           {claims.map((claim) => (
16:             <tr key={claim.id} className="border-t hover:bg-gray-50">
17:               <td>{claim.item_name}</td>
18:               <td>{claim.location}</td>
19:               <td>{claim.claimed_at}</td>
20:             </tr>
21:           ))}
22:         </tbody>
23:       </table>
24:     </div>
25:   );
26: }
```

■ File: src\components\FeedbackForm.js

```
  1: import { useState } from "react";
  2:
  3: export default function FeedbackForm() {
  4:   const [text, setText] = useState("");
  5:   const [status, setStatus] = useState("");
  6:
  7:   const handleSubmit = async (e) => {
  8:     e.preventDefault();
  9:     const res = await fetch("/submit_feedback", {
 10:       method: "POST",
 11:       headers: { "Content-Type": "application/json" },
 12:       credentials: "include",
 13:       body: JSON.stringify({ feedback: text }),
 14:     });
 15:
 16:     if (res.ok) {
 17:       setStatus("Thank you for your feedback!");
 18:       setText("");
 19:     } else {
 20:       setStatus("Error submitting feedback.");
 21:     }
 22:   };
 23:
 24:   return (
 25:     <form onSubmit={handleSubmit} className="space-y-2">
 26:       <textarea
 27:         className="w-full border rounded p-2"
 28:         rows="3"
 29:         required
 30:         value={text}
 31:         onChange={(e) => setText(e.target.value)}
 32:         placeholder="Write your feedback..."
 33:       />
 34:       <button className="bg-green-600 text-white px-4 py-2 rounded">
 35:         Submit
 36:       </button>
 37:       {status && <p className="text-sm text-gray-600 mt-2">{status}</p>}
 38:     </form>
 39:   );
 40: }
```

■ File: src\components\FlashMessage.js

■ File: src\components\Header.js

```
  1: import { Link, useNavigate } from "react-router-dom";
  2: import { useEffect, useState } from "react";
  3:
  4: export default function Header() {
  5:   const [isAdmin, setIsAdmin] = useState(false);
  6:   const [loggedIn, setLoggedIn] = useState(false);
```

```
  const navigate = useNavigate();

  useEffect(() => {
    fetch("/check_user", { credentials: "include" })
      .then((res) => res.json())
      .then((data) => {
        if (data && data.email) {
          setLoggedIn(true);
          setIsAdmin(data.role === "admin");
        }
      });
  }, []);

  const handleLogout = async () => {
    await fetch("/logout", { credentials: "include" });
    navigate("/");
  };

  return (
    <header className="bg-blue-600 text-white shadow">
      <div className="max-w-7xl mx-auto px-4 py-3 flex justify-between items-center">
        <Link to={isAdmin ? "/admin" : "/home"}>
          <h1 className="text-lg font-bold">? Lost & Found</h1>
        </Link>

        {loggedIn ? (
          <nav className="flex items-center gap-4 text-sm">
            {!isAdmin && (
              <>
                <Link
                  to="/register-item"
                  className="hover:underline hover:text-gray-100"
                >
                  Register Item
                </Link>
                <Link
                  to="/lost-found-items"
                  className="hover:underline hover:text-gray-100"
                >
                  Browse Items
                </Link>
              </>
            )}
            {isAdmin && (
              <Link to="/admin" className="hover:underline hover:text-gray-100">
                Admin Dashboard
              </Link>
            )}
            <button
              onClick={handleLogout}
              className="bg-white text-blue-600 px-3 py-1 rounded hover:bg-gray-100 transition"
            >
              Logout
            </button>
          </nav>
        ) : (
          <nav className="flex gap-3 text-sm">
            <Link to="/" className="hover:underline">
              Login
            </Link>
            <Link to="/register" className="hover:underline">
              Register
            </Link>
          </nav>
        )}
      </div>
    </header>
  );
}
```

■ File: src\components\ItemTable.js

```
export default function ItemTable({ items }) {
  if (!items.length) return <p className="text-gray-500">No items yet.</p>;

  return (
    <div className="overflow-x-auto border rounded">
      <table className="min-w-full text-sm text-left">
        <thead className="bg-gray-100 text-gray-700 uppercase">
          <tr>
            <th className="px-4 py-2">Item</th>
            <th>Status</th>
            <th>Location</th>
            <th>Created At</th>
          </tr>
        </thead>
        <tbody>
          {items.map((item) => (
            <tr
              key={item.id}
              className="border-t hover:bg-gray-50 transition-colors"
            >
              <td className="px-4 py-2 font-medium">{item.name}</td>
              <td>
                <span
                  className={`px-2 py-1 text-xs rounded ${
                    item.status === "found"
                      ? "bg-green-100 text-green-700"
                      : item.status === "claimed"
                      ? "bg-blue-100 text-blue-700"
                      : "bg-red-100 text-red-700"
                  }`}
                >
                  {item.status}
                </span>
              </td>
              <td>{item.location}</td>
              <td>{item.created_at}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}
```

■ File: src\components\MessageList.js

```
import { useState } from "react";

export default function MessageList({ messages }) {
  const [replies, setReplies] = useState({});

  const handleReply = async (msgId, senderId, itemId) => {
    const replyText = replies[msgId];
    if (!replyText) return;

    await fetch("/reply_message", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      credentials: "include",
      body: JSON.stringify({
        receiver_id: senderId,
        item_id: itemId,
        reply: replyText,
      }),
    });

    setReplies((r) => ({ ...r, [msgId]: "" }));
    alert("Reply sent.");
  };
```

```
 24:
 25:   if (!messages.length)
 26:     return <p className="text-gray-500">No messages yet.</p>;
 27:
 28:   return (
 29:     <div className="space-y-4">
 30:       {messages.map((msg) => (
 31:         <div
 32:           key={msg.id}
 33:           className="border p-4 rounded shadow-sm bg-white space-y-2"
 34:         >
 35:           <h4 className="font-bold">From: {msg.sender_name}</h4>
 36:           <p className="text-sm">Item: {msg.item_name}</p>
 37:           <p>{msg.message}</p>
 38:           <textarea
 39:             placeholder="Reply..."
 40:             className="w-full border rounded p-2 text-sm"
 41:             rows="2"
 42:             value={replies[msg.id] || ""}
 43:             onChange={(e) =>
 44:               setReplies((prev) => ({ ...prev, [msg.id]: e.target.value }))
 45:             }
 46:           />
 47:           <button
 48:             className="mt-2 px-3 py-1 text-sm bg-blue-600 text-white rounded"
 49:             onClick={() => handleReply(msg.id, msg.sender_id, msg.item_id)}
 50:           >
 51:             Send Reply
 52:           </button>
 53:         </div>
 54:       ))}
 55:     </div>
 56:   );
 57: }
```

■ File: src\index.css

```
  1: @tailwind base;
  2: @tailwind components;
  3: @tailwind utilities;
```

■ File: src\index.js

```
  1: import React from 'react';
  2: import ReactDOM from 'react-dom/client';
  3: import './index.css';
  4: import App from './App';
  5: import reportWebVitals from './reportWebVitals';
  6:
  7: const root = ReactDOM.createRoot(document.getElementById('root'));
  8: root.render(
  9:   <React.StrictMode>
 10:     <App />
 11:   </React.StrictMode>
 12: );
 13:
 14: // If you want to start measuring performance in your app, pass a function
 15: // to log results (for example: reportWebVitals(console.log))
 16: // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
 17: reportWebVitals();
```

■ File: src\pages\AdminPage.js

```
  1: import { useEffect, useState } from "react";
  2: import AdminTable from "../components/AdminTable";
  3:
  4: export default function AdminPage() {
```

```
  const [items, setItems] = useState([]);
  const [claims, setClaims] = useState([]);
  const [users, setUsers] = useState([]);
  const [feedback, setFeedback] = useState([]);
  const [message, setMessage] = useState("");

  const fetchData = async () => {
    try {
      const [itemsRes, claimsRes, usersRes, feedbackRes] = await Promise.all([
        fetch("/admin/items", { credentials: "include" }),
        fetch("/admin/claims", { credentials: "include" }),
        fetch("/admin/users", { credentials: "include" }),
        fetch("/admin/feedback", { credentials: "include" }),
      ]);

      setItems(await itemsRes.json());
      setClaims(await claimsRes.json());
      setUsers(await usersRes.json());
      setFeedback(await feedbackRes.json());
    } catch (err) {
      console.error("Admin fetch error", err);
      setMessage("Failed to load admin data.");
    }
  };

  useEffect(() => {
    fetchData();
  }, []);

  const handleDelete = async (type, id) => {
    const url = `/delete_${type}/${id}`;
    const res = await fetch(url, {
      method: "POST", // or DELETE depending on your backend
      credentials: "include",
    });

    if (res.ok) {
      setMessage(`? ${type} deleted successfully.`);
      fetchData();
    } else {
      setMessage(`? Failed to delete ${type}.`);
    }
  };

  return (
    <div className="max-w-7xl mx-auto px-6 py-8 space-y-8">
      <h1 className="text-2xl font-bold text-center text-blue-600">
        Admin Dashboard
      </h1>

      {message && <div className="text-center text-green-700">{message}</div>}

      <AdminTable
        title="Registered Items"
        type="item"
        data={items}
        headers={["Name", "Location", "Status", "Actions"]}
        rowRenderer={(item) => [
          item.name,
          item.location,
          item.status,
          <button
            onClick={() => handleDelete("item", item.id)}
            className="text-red-600 hover:underline text-sm"
          >
            Delete
          </button>,
        ]}
      />

      <AdminTable
        title="Claims"
        type="claim"
```

```
        data={claims}
        headers={["Item", "User", "Location", "Actions"]}
        rowRenderer={(claim) => [
          claim.item_name,
          claim.claimed_by,
          claim.location,
          <button
            onClick={() => handleDelete("claim", claim.id)}
            className="text-red-600 hover:underline text-sm"
          >
            Delete
          </button>,
        ]}
      />

      <AdminTable
        title="Users"
        type="user"
        data={users}
        headers={["Name", "Email", "Role", "Actions"]}
        rowRenderer={(user) => [
          user.name,
          user.email,
          user.role || "user",
          <button
            onClick={() => handleDelete("user", user.id)}
            className="text-red-600 hover:underline text-sm"
          >
            Delete
          </button>,
        ]}
      />

      <AdminTable
        title="Feedback"
        type="feedback"
        data={feedback}
        headers={["User", "Message", "Actions"]}
        rowRenderer={(fb) => [
          fb.user_name || "Anonymous",
          fb.message,
          <button
            onClick={() => handleDelete("feedback", fb.id)}
            className="text-red-600 hover:underline text-sm"
          >
            Delete
          </button>,
        ]}
      />
    </div>
  );
}
```

■ File: src\pages\HomePage.js

```
import { useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import ItemTable from "../components/ItemTable";
import MessageList from "../components/MessageList";
import ClaimsTable from "../components/ClaimsTable";
import FeedbackForm from "../components/FeedbackForm";

export default function HomePage() {
  const navigate = useNavigate();
  const [name, setName] = useState("");
  const [items, setItems] = useState([]);
  const [claims, setClaims] = useState([]);
  const [messages, setMessages] = useState([]);

  useEffect(() => {
    const fetchHomeData = async () => {
```

```
17:       try {
18:         const res = await fetch("/home", { credentials: "include" });
19:         if (!res.ok) return navigate("/");
20:
21:         const data = await res.json();
22:         setName(data.name);
23:         setItems(data.items || []);
24:         setClaims(data.claims || []);
25:         setMessages(data.messages || []);
26:       } catch (err) {
27:         console.error(err);
28:         navigate("/");
29:       }
30:     };
31:     fetchHomeData();
32:   }, [navigate]);
33:
34:   return (
35:     <div className="p-6 max-w-6xl mx-auto space-y-8">
36:       <h1 className="text-2xl font-bold text-center text-blue-600">
37:         Welcome, {name}
38:       </h1>
39:
40:       <section>
41:         <h2 className="text-xl font-semibold text-gray-700 mb-2">
42:           My Registered Items
43:         </h2>
44:         <ItemTable items={items} />
45:       </section>
46:
47:       <section>
48:         <h2 className="text-xl font-semibold text-gray-700 mb-2">My Claims</h2>
49:         <ClaimsTable claims={claims} />
50:       </section>
51:
52:       <section>
53:         <h2 className="text-xl font-semibold text-gray-700 mb-2">Messages</h2>
54:         <MessageList messages={messages} />
55:       </section>
56:
57:       <section>
58:         <h2 className="text-xl font-semibold text-gray-700 mb-2">
59:           Platform Feedback
60:         </h2>
61:         <FeedbackForm />
62:       </section>
63:     </div>
64:   );
65: }
```

■ File: src\pages\ItemRegisterPage.js

```
  1: import { useState } from "react";
  2: import { useNavigate } from "react-router-dom";
  3:
  4: export default function RegisterItemPage() {
  5:   const navigate = useNavigate();
  6:   const [formData, setFormData] = useState({
  7:     name: "",
  8:     description: "",
  9:     location: "",
 10:     status: "lost",
 11:   });
 12:   const [image, setImage] = useState(null);
 13:   const [message, setMessage] = useState("");
 14:
 15:   const handleChange = (e) => {
 16:     setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
 17:   };
 18:
 19:   const handleFileChange = (e) => {
```

```
    setImage(e.target.files[0]);
  };

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

    if (!image) {
      setMessage("Please select an image file.");
      return;
    }

    const data = new FormData();
    data.append("name", formData.name);
    data.append("description", formData.description);
    data.append("location", formData.location);
    data.append("status", formData.status);
    data.append("image", image);

    try {
      const res = await fetch("/register_item", {
        method: "POST",
        body: data,
        credentials: "include",
      });

      if (res.ok) {
        setMessage("? Item registered successfully!");
        setTimeout(() => navigate("/lost-found-items"), 2000);
      } else {
        setMessage("? Failed to register item.");
      }
    } catch (err) {
      console.error(err);
      setMessage("Something went wrong.");
    }
  };

  return (
    <div className="max-w-2xl mx-auto mt-10 px-4">
      <h2 className="text-2xl font-bold text-center text-blue-600 mb-6">
        Register Lost / Found Item
      </h2>

      {message && (
        <div className="mb-4 text-center text-sm text-gray-700">
          {message}
        </div>
      )}

      <form
        onSubmit={handleSubmit}
        className="space-y-4 bg-white p-6 rounded shadow"
      >
        <input
          type="text"
          name="name"
          placeholder="Item Name"
          value={formData.name}
          onChange={handleChange}
          required
          className="w-full px-4 py-2 border rounded focus:outline-none focus:ring-2 focus:ring-blue-500"
        />

        <textarea
          name="description"
          placeholder="Item Description"
          value={formData.description}
          onChange={handleChange}
          required
          rows={4}
          className="w-full px-4 py-2 border rounded focus:outline-none focus:ring-2 focus:ring-blue-500"
        />

```

```
 93:         <input
 94:           type="text"
 95:           name="location"
 96:           placeholder="Lost/Found Location"
 97:           value={formData.location}
 98:           onChange={handleChange}
 99:           required
100:           className="w-full px-4 py-2 border rounded focus:outline-none focus:ring-2 focus:ring-blue-500"
101:         />
102:
103:         <select
104:           name="status"
105:           value={formData.status}
106:           onChange={handleChange}
107:           className="w-full px-4 py-2 border rounded focus:outline-none focus:ring-2 focus:ring-blue-500"
108:         >
109:           <option value="lost">Lost</option>
110:           <option value="found">Found</option>
111:         </select>
112:
113:         <input
114:           type="file"
115:           name="image"
116:           accept="image/*"
117:           onChange={handleFileChange}
118:           className="w-full"
119:         />
120:
121:         <button
122:           type="submit"
123:           className="w-full bg-blue-600 text-white py-2 rounded hover:bg-blue-700 transition"
124:         >
125:           Register Item
126:         </button>
127:       </form>
128:     </div>
129:   );
130: }
```

■ File: src\pages\ItemsPage.js

```
  1: import { useEffect, useState } from "react";
  2:
  3: export default function ItemsPage() {
  4:   const [items, setItems] = useState([]);
  5:   const [search, setSearch] = useState("");
  6:   const [filter, setFilter] = useState("all");
  7:   const [message, setMessage] = useState("");
  8:
  9:   useEffect(() => {
 10:     fetchItems();
 11:   }, []);
 12:
 13:   const fetchItems = async () => {
 14:     try {
 15:       const res = await fetch("/items", { credentials: "include" });
 16:       const data = await res.json();
 17:       setItems(data || []);
 18:     } catch (err) {
 19:       console.error("Failed to fetch items");
 20:     }
 21:   };
 22:
 23:   const handleClaim = async (itemId) => {
 24:     const res = await fetch("/claim_item", {
 25:       method: "POST",
 26:       headers: { "Content-Type": "application/json" },
 27:       credentials: "include",
 28:       body: JSON.stringify({ item_id: itemId }),
 29:     });
 30:
```

```
    if (res.ok) {
      setMessage("? Item claimed successfully!");
      fetchItems(); // refresh list
    } else {
      setMessage("? Failed to claim item.");
    }
  };

  const filteredItems = items.filter((item) => {
    const matchesSearch =
      item.name.toLowerCase().includes(search.toLowerCase()) ||
      item.location.toLowerCase().includes(search.toLowerCase());

    const matchesFilter =
      filter === "all" ? true : item.status.toLowerCase() === filter;

    return matchesSearch && matchesFilter;
  });

  return (
    <div className="max-w-6xl mx-auto p-6 space-y-6">
      <h1 className="text-2xl font-bold text-blue-600 text-center">
        Lost and Found Items
      </h1>

      {message && (
        <div className="text-sm text-center text-green-700">{message}</div>
      )}

      {/* Search & Filter */}
      <div className="flex flex-col sm:flex-row gap-4 justify-between items-center mt-4">
        <input
          type="text"
          placeholder="Search by name or location..."
          className="w-full sm:w-1/2 px-4 py-2 border rounded"
          value={search}
          onChange={(e) => setSearch(e.target.value)}
        />

        <select
          value={filter}
          onChange={(e) => setFilter(e.target.value)}
          className="px-4 py-2 border rounded"
        >
          <option value="all">All</option>
          <option value="lost">Lost</option>
          <option value="found">Found</option>
          <option value="claimed">Claimed</option>
        </select>
      </div>

      {/* Items Grid */}
      <div className="grid gap-6 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 mt-4">
        {filteredItems.map((item) => (
          <div
            key={item.id}
            className="border rounded-lg overflow-hidden shadow bg-white flex flex-col"
          >
            {item.image_url && (
              <img
                src={item.image_url}
                alt={item.name}
                className="w-full h-48 object-cover"
              />
            )}
            <div className="p-4 space-y-2 flex-1 flex flex-col justify-between">
              <div>
                <h2 className="text-lg font-semibold">{item.name}</h2>
                <p className="text-sm text-gray-500">{item.location}</p>
                <p className="text-sm">{item.description}</p>
              </div>

              <div className="mt-3 space-y-1">
```

```
104:                 <span
105:                   className={`inline-block text-xs px-2 py-1 rounded ${
106:                     item.status === "found"
107:                       ? "bg-green-100 text-green-700"
108:                       : item.status === "lost"
109:                       ? "bg-yellow-100 text-yellow-700"
110:                       : "bg-blue-100 text-blue-700"
111:                   }`}
112:                 >
113:                   {item.status.toUpperCase()}
114:                 </span>
115:
116:                 {item.status === "found" && (
117:                   <button
118:                     className="w-full mt-2 text-sm bg-blue-600 text-white py-1 rounded hover:bg-blue-700 tra
119:                     onClick={() => handleClaim(item.id)}
120:                   >
121:                     Claim
122:                   </button>
123:                 )}
124:               </div>
125:             </div>
126:           </div>
127:         ))}
128:       </div>
129:
130:       {filteredItems.length === 0 && (
131:         <p className="text-center text-gray-500 mt-10">No items found.</p>
132:       )}
133:     </div>
134:   );
135: }
```

■ File: src\pages\LoginPage.js

```
  1: import { useState } from "react";
  2: import { useNavigate } from "react-router-dom";
  3:
  4: export default function LoginPage() {
  5:   const navigate = useNavigate();
  6:   const [email, setEmail] = useState("");
  7:   const [password, setPassword] = useState("");
  8:   const [error, setError] = useState("");
  9:
 10:   const handleLogin = async (e) => {
 11:     e.preventDefault();
 12:     setError("");
 13:
 14:     const form = new FormData();
 15:     form.append("email", email);
 16:     form.append("password", password);
 17:
 18:     try {
 19:       const res = await fetch("http://localhost:5000/login", {
 20:         method: "POST",
 21:         credentials: "include",
 22:         body: form,
 23:       });
 24:
 25:       if (res.redirected || res.status === 200) {
 26:         // Get user info to determine redirect
 27:         const profile = await fetch("http://localhost:5000/check_user", {
 28:           credentials: "include",
 29:         });
 30:         const data = await profile.json();
 31:         if (data.role === "admin") {
 32:           navigate("/admin");
 33:         } else {
 34:           navigate("/home");
 35:         }
 36:       } else {
```

```
37:         setError("Invalid email or password.");
38:       }
39:     } catch (err) {
40:       setError("Login failed. Please try again.");
41:     }
42:   };
43:
44:   return (
45:     <div className="flex items-center justify-center min-h-screen bg-gray-100 px-4">
46:       <div className="bg-white p-8 rounded-lg shadow-md w-full max-w-md">
47:         <h2 className="text-2xl font-bold text-center text-blue-600 mb-6">
48:           Login
49:         </h2>
50:         {error && (
51:           <div className="text-red-600 text-sm text-center mb-2">{error}</div>
52:         )}
53:         <form onSubmit={handleLogin} className="space-y-4">
54:           <input
55:             type="email"
56:             placeholder="Email"
57:             className="w-full px-4 py-2 border rounded-md"
58:             value={email}
59:             onChange={(e) => setEmail(e.target.value)}
60:             required
61:           />
62:           <input
63:             type="password"
64:             placeholder="Password"
65:             className="w-full px-4 py-2 border rounded-md"
66:             value={password}
67:             onChange={(e) => setPassword(e.target.value)}
68:             required
69:           />
70:           <button
71:             type="submit"
72:             className="w-full bg-blue-600 text-white py-2 rounded hover:bg-blue-700 transition"
73:           >
74:             Login
75:           </button>
76:         </form>
77:       </div>
78:     </div>
79:   );
80: }
```

■ File: src\pages\RegisterPage.js

```
  1: import { useState } from "react";
  2: import { useNavigate } from "react-router-dom";
  3:
  4: export default function RegisterPage() {
  5:   const navigate = useNavigate();
  6:   const [formData, setFormData] = useState({
  7:     name: "",
  8:     email: "",
  9:     password: "",
 10:     confirm_password: "",
 11:   });
 12:   const [error, setError] = useState("");
 13:   const [success, setSuccess] = useState("");
 14:
 15:   const handleChange = (e) => {
 16:     setFormData((prev) => ({
 17:       ...prev,
 18:       [e.target.name]: e.target.value,
 19:     }));
 20:   };
 21:
 22:   const handleSubmit = async (e) => {
 23:     e.preventDefault();
 24:     setError("");
```

```
    setSuccess("");

    if (formData.password !== formData.confirm_password) {
      setError("Passwords do not match");
      return;
    }

    const form = new FormData();
    Object.entries(formData).forEach(([key, value]) => form.append(key, value));

    try {
      const res = await fetch("http://localhost:5000/register", {
        method: "POST",
        credentials: "include",
        body: form,
      });

      if (res.redirected || res.status === 200) {
        setSuccess("Registration successful! Redirecting to login...");
        setTimeout(() => navigate("/"), 2000);
      } else {
        setError("Email already registered.");
      }
    } catch (err) {
      setError("Something went wrong. Please try again.");
    }
  };

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100 px-4">
      <div className="bg-white p-8 rounded-lg shadow-md w-full max-w-md">
        <h2 className="text-2xl font-bold text-center text-blue-600 mb-6">
          Create Account
        </h2>

        {error && (
          <div className="bg-red-100 text-red-700 p-2 rounded text-sm mb-4">
            {error}
          </div>
        )}

        {success && (
          <div className="bg-green-100 text-green-700 p-2 rounded text-sm mb-4">
            {success}
          </div>
        )}

        <form onSubmit={handleSubmit} className="space-y-4">
          <input
            type="text"
            name="name"
            placeholder="Full Name"
            className="w-full px-4 py-2 border rounded"
            onChange={handleChange}
            value={formData.name}
            required
          />
          <input
            type="email"
            name="email"
            placeholder="Email"
            className="w-full px-4 py-2 border rounded"
            onChange={handleChange}
            value={formData.email}
            required
          />
          <input
            type="password"
            name="password"
            placeholder="Password"
            className="w-full px-4 py-2 border rounded"
            onChange={handleChange}
            value={formData.password}
```

```
 98:             required
 99:           />
100:           <input
101:             type="password"
102:             name="confirm_password"
103:             placeholder="Confirm Password"
104:             className="w-full px-4 py-2 border rounded"
105:             onChange={handleChange}
106:             value={formData.confirm_password}
107:             required
108:           />
109:           <button
110:             type="submit"
111:             className="w-full bg-blue-600 text-white py-2 rounded hover:bg-blue-700 transition"
112:           >
113:             Register
114:           </button>
115:         </form>
116:       </div>
117:     </div>
118:   );
119: }
```

■ File: src\reportWebVitals.js

```
  1: const reportWebVitals = onPerfEntry => {
  2:   if (onPerfEntry && onPerfEntry instanceof Function) {
  3:     import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
  4:       getCLS(onPerfEntry);
  5:       getFID(onPerfEntry);
  6:       getFCP(onPerfEntry);
  7:       getLCP(onPerfEntry);
  8:       getTTFB(onPerfEntry);
  9:     });
 10:   }
 11: };
 12:
 13: export default reportWebVitals;
```

■ File: src\setupTests.js

```
  1: // jest-dom adds custom jest matchers for asserting on DOM nodes.
  2: // allows you to do things like:
  3: // expect(element).toHaveTextContent(/react/i)
  4: // learn more: https://github.com/testing-library/jest-dom
  5: import '@testing-library/jest-dom';
```

■ File: tailwind.config.js

```
  1: /** @type {import('tailwindcss').Config} */
  2: module.exports = {
  3:   content: [
  4:     "./src/**/*.{js,jsx,ts,tsx}", // very important for React
  5:   ],
  6:   theme: {
  7:     extend: {},
  8:   },
  9:   plugins: [],
 10: };
```
