TECH STACK

⚙️ Full Tech Stack for QR-Based URL Safety Checker (PWA)

🧩 1. Frontend (User Interface)

Layer	Technology / Library	Purpose
Markup	**HTML5**	Structure of the UI
Styling	**CSS3** (+ optional Tailwind)	Styling the app, light/dark mode support
Interactivity	**JavaScript (Vanilla)**	Core logic, QR scanning, UI updates
QR Scanner	html5-qrcode	JS library to access camera and scan QR codes [npm install html5-qrcode]
Theme Toggle	Custom JS / Local Storage	Toggle and save light/dark theme state
PWA Support	manifest.json, service worker	Installable, splash screen, offline shell
Animations	CSS transitions / JS / Lottie	Visual feedback for safety results
Scan History	localStorage	Store past scanned URLs locally

🔗 2. Backend (API & Processing Layer)

Layer	Technology / Library	Purpose
Framework	**Flask (Python)**	Handle API routes for checking URL safety
HTTP Requests	requests (Python lib)	To call Google Safe Browsing API
Threat Parsing	JSON parsing (json module)	Parse response from GSB API
Reporting	Custom logic / file storage	Save suspicious reports (optional)
CORS	flask-cors	Enable frontend-backend communication

🔐 3. External Services

Service	Use	Notes
Google Safe Browsing API	URL safety verification	Requires API key, free tier available
LottieFiles (optional)	Animated icons for feedback	Can embed JSON animations in frontend

---

📦 4. PWA Integration

Component	Tool/Config	Purpose
Web Manifest	manifest.json	Defines app name, icons, colors, etc.
Service Worker	sw.js (basic version)	Enables installability, shell caching
App Icons	PNG / SVG	For home screen & splash screen
Prompt Logic	Custom JS	Show "Add to Home Screen" logic

💾 5. Deployment (Suggested Stack)

Part	Tool / Service	Notes
Frontend	Netlify / GitHub Pages	Easy deployment, PWA support
Backend	Render / PythonAnywhere	Host Flask app, handles API calls

---

🛠️ 6. Development Tools

Tool	Purpose
VS Code	Development IDE
Postman	API testing
Browser DevTools	Testing PWA behavior & responsiveness
Git + GitHub	Version control
Icons8 / Canva	Icons & splash screen design

✅ **Summary of Key Packages to Install**

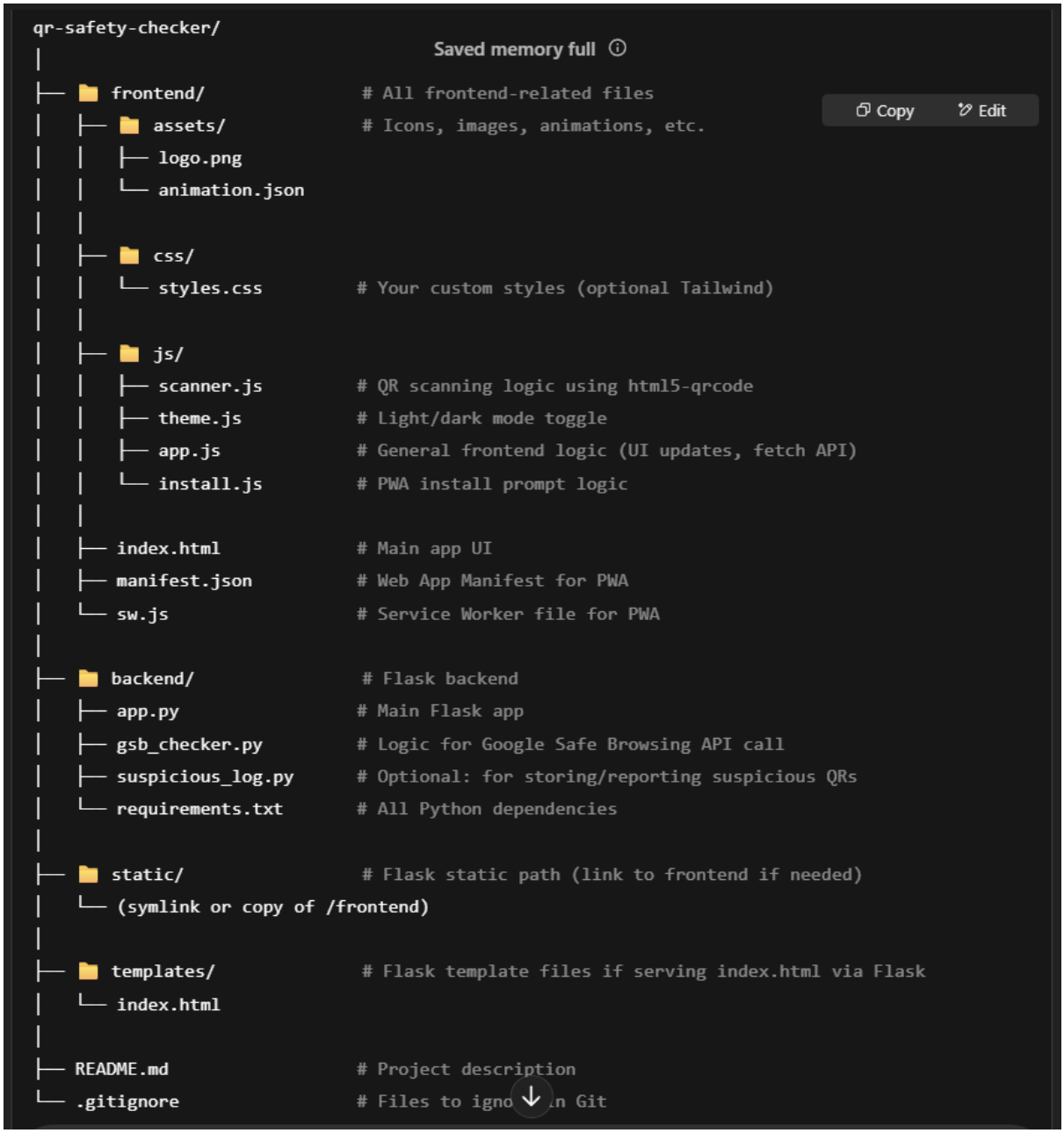
Frontend (if managing with npm)

npm install html5-qrcode

Backend (Python)

pip install flask flask-cors requests

📁 Project Folder Structure




```
qr-safety-checker/
│
├── frontend/                # All frontend-related files
│   ├── assets/              # Icons, images, animations, etc.
│   │   ├── logo.png
│   │   └── animation.json
│   │
│   ├── css/
│   │   └── styles.css       # Your custom styles (optional Tailwind)
│   │
│   ├── js/
│   │   ├── scanner.js       # QR scanning logic using html5-qrcode
│   │   ├── theme.js         # Light/dark mode toggle
│   │   ├── app.js           # General frontend logic (UI updates, fetch API)
│   │   └── install.js       # PWA install prompt logic
│   │
│   ├── index.html           # Main app UI
│   ├── manifest.json        # Web App Manifest for PWA
│   └── sw.js                # Service Worker file for PWA
│
├── backend/                 # Flask backend
│   ├── app.py               # Main Flask app
│   ├── gsb_checker.py       # Logic for Google Safe Browsing API call
│   ├── suspicious_log.py    # Optional: for storing/reporting suspicious QRs
│   └── requirements.txt     # All Python dependencies
│
├── static/                  # Flask static path (link to frontend if needed)
│   └── (symlink or copy of /frontend)
│
├── templates/               # Flask template files if serving index.html via Flask
│   └── index.html
│
├── README.md                # Project description
└── .gitignore               # Files to igno n Git
```

🔍 How the Structure Works

- frontend/ contains your **PWA UI**: HTML, JS, CSS, icons, service worker, etc.
- backend/ contains your **Flask API server** which:
  - Receives scanned URLs
  - Checks them via Google Safe Browsing API

    - Returns results
- static/ and templates/ are Flask defaults:
    - If you choose to serve your frontend directly via Flask, move your index.html into templates/ and assets into static/
- You can use README.md to document:
    - Project goals
    - Setup instructions
    - API key usage

END