Electron GUI Installer Scaffold for iTechSmart

This scaffold creates a cross-platform **Electron** installer application that wraps and runs your existing `hardened_docker_install.sh` script (or other platform-specific installers). It provides a simple multi-step GUI wizard to configure `.env` values and run the installation with elevation where required.

Files included in this scaffold (copy each into project root under a new folder `electron-installer/`)

---

1) `package.json`

```
{
  "name": "itechsmart-installer",
  "version": "0.1.0",
  "description": "Cross-platform GUI installer for iTechSmart (wraps
hardened_docker_install.sh)",
  "main": "src/main.js",
  "scripts": {
    "start": "electron .",
    "pack": "electron-builder --dir",
    "dist": "electron-builder",
    "lint": "echo \"no lint configured\""
  },
  "build": {
    "appId": "com.itechsmart.installer",
    "productName": "iTechSmart Installer",
    "files": ["src/**/*", "../hardened_docker_install.sh"],
    "directories": { "buildResources": "build" },
    "linux": {
      "target": ["AppImage", "deb"],
      "category": "Utility"
    },
    "win": { "target": ["nsis"] },
    "mac": { "target": ["dmg", "pkg"] }
  },
  "dependencies": {
    "electron-store": "^8.0.0",
    "sudo-prompt": "^9.2.1"
  },
  "devDependencies": {
    "electron": "^26.0.0",
```

```
    "electron-builder": "^24.0.0"
  }
}
```

---

2) `src/main.js` (Electron main process)

```
const { app, BrowserWindow, ipcMain, dialog } = require('electron');
const path = require('path');
const isDev = process.env.ELECTRON_DEV === '1' || process.env.NODE_ENV ===
'development';

function createWindow() {
  const win = new BrowserWindow({
    width: 780,
    height: 640,
    webPreferences: {
      preload: path.join(__dirname, 'preload.js'),
      contextIsolation: true,
      sandbox: false
    },
    resizable: false
  });

  win.loadFile(path.join(__dirname, 'index.html'));
  if (isDev) win.webContents.openDevTools({ mode: 'detach' });
}

app.whenReady().then(() => {
  createWindow();

  app.on('activate', function () {
    if (BrowserWindow.getAllWindows().length === 0) createWindow();
  });
});

app.on('window-all-closed', function () {
  if (process.platform !== 'darwin') app.quit();
});

// IPC handlers
ipcMain.handle('select-script', async () => {
  const { canceled, filePaths } = await dialog.showOpenDialog({
    properties: ['openFile'],
    filters: [{ name: 'Shell Scripts', extensions: ['sh', 'bat', 'ps1'] }]
  });
```

```
  if (canceled) return null;
  return filePaths[0];
});

ipcMain.handle('show-message', (ev, opts) => {
  dialog.showMessageBox(opts);
});
```

---

3) `src/preload.js` (safe IPC bridge)

```
const { contextBridge, ipcRenderer } = require('electron');

contextBridge.exposeInMainWorld('electronAPI', {
  selectScript: () => ipcRenderer.invoke('select-script'),
  showMessage: (opts) => ipcRenderer.invoke('show-message', opts),
  runInstaller: (payload) => ipcRenderer.invoke('run-installer', payload)
});
```

> Note: `run-installer` will be implemented in the renderer using `sudo-prompt` (so we call it from renderer Node context). We keep main process minimal for security.

---

4) `src/index.html` (renderer)

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="Content-Security-Policy" content="default-src 'self'
'unsafe-inline' data:;">
    <title>iTechSmart Installer</title>
    <style>
      body { font-family: Arial, Helvetica, sans-serif; margin: 18px; }
      .card { border-radius: 12px; box-shadow: 0 6px 18px rgba(0,0,0,0.08);
padding: 18px; }
      .row { margin-bottom: 12px; }
      label { display:block; font-weight:600; margin-bottom:6px; }
      input[type=text], textarea { width:100%; padding:8px; border-radius:6px;
border:1px solid #ccc }
      button { padding:10px 16px; border-radius:8px; border:none;
cursor:pointer }
      .primary { background:#2563eb; color:#fff }
      .secondary { background:#eeeeee }
```

```
      #log { height:200px; overflow:auto; background:#0b1220; color:#e6e6e6;
padding:8px; border-radius:6px; }
    </style>
  </head>
  <body>
    <div class="card">
      <h2>iTechSmart — Installer</h2>
      <div class="row">
        <button id="btn-select" class="secondary">Select installer script</
button>
        <span id="script-path">(will use bundled hardened_docker_install.sh)</
span>
      </div>

      <div class="row">
        <label>Environment variables (key=value per line)</label>
        <textarea id="env-text"
placeholder="DB_PASS=secret\nADMIN_EMAIL=admin@example.com"></textarea>
      </div>

      <div class="row">
        <button id="btn-run" class="primary">Run installer (elevated)</button>
        <button id="btn-stop" style="display:none">Cancel</button>
      </div>

      <h3>Installer Log</h3>
      <pre id="log">Log will appear here...</pre>
    </div>

    <script src="renderer.js"></script>
  </body>
</html>
```

---

5) `src/renderer.js` **(renderer process; runs installer with** `sudo-prompt` **)**

```
const fs = require('fs');
const path = require('path');
const { spawn } = require('child_process');
const sudo = require('sudo-prompt');
const os = require('os');

const defaultScript = path.resolve(__dirname, '..', '..',
'hardened_docker_install.sh');
let chosenScript = defaultScript;
```

```
const btnSelect = document.getElementById('btn-select');
const scriptPathElem = document.getElementById('script-path');
const envText = document.getElementById('env-text');
const btnRun = document.getElementById('btn-run');
const logEl = document.getElementById('log');

scriptPathElem.innerText = chosenScript;

btnSelect.addEventListener('click', async () => {
  const p = await window.electronAPI.selectScript();
  if (p) {
    chosenScript = p;
    scriptPathElem.innerText = chosenScript;
  }
});

function appendLog(line) {
  logEl.textContent += '\n' + line;
  logEl.scrollTop = logEl.scrollHeight;
}

btnRun.addEventListener('click', async () => {
  // write env to temp file
  const envContent = envText.value || '';
  const tmpDir = os.tmpdir();
  const envPath = path.join(tmpDir, `itechsmart_env_${Date.now()}.env`);
  fs.writeFileSync(envPath, envContent, { mode: 0o600 });
  appendLog(`Wrote env file to ${envPath}`);

  // Build command to run: ensure script is executable
  try {
    fs.chmodSync(chosenScript, 0o755);
  } catch (e) {
    appendLog('Failed to chmod script: ' + String(e));
  }

  // Copy temp env next to script's directory or pass as env file path if script
supports it
  // Here we will copy env into script's directory as .env
  const scriptDir = path.dirname(chosenScript);
  const destEnv = path.join(scriptDir, '.env');
  try {
    fs.copyFileSync(envPath, destEnv);
    appendLog(`Copied env to ${destEnv}`);
  } catch (e) {
    appendLog('Failed to copy env: ' + String(e));
  }
```

```
  const cmd = `bash "${chosenScript}"`;
  appendLog('Running installer with elevation...');

  const opts = { name: 'iTechSmart Installer' };
  sudo.exec(cmd, opts, (error, stdout, stderr) => {
    if (error) {
      appendLog('Installer error: ' + error);
      window.electronAPI.showMessage({
        type: 'error',
        title: 'Installation failed',
        message: String(error)
      });
      return;
    }
    if (stdout) appendLog('STDOUT:\n' + stdout);
    if (stderr) appendLog('STDERR:\n' + stderr);
    appendLog('Installer finished.');
    window.electronAPI.showMessage({
      type: 'info',
      title: 'Installer finished',
      message: 'Installation completed. Check logs in installer application.'
    });
  });
});
```

Notes: `sudo-prompt` will show native elevation dialogs on macOS and Windows and run the command as admin/root. On Linux it uses `pkexec` / `gksudo` / `kdesudo` depending on availability.

---

6) `build/` (assets folder)

Place optional icons and background images used by electron-builder here (e.g. `icon.png`, `icon.icns`, `icon.ico`). For quick testing, you can skip icons.

---

7) `README_FOR_INSTALLER.md` (how to build and run)

```
# iTechSmart Electron Installer

## Prerequisites
- Node.js (18+ recommended)
- npm or yarn
- Your project root should contain `hardened_docker_install.sh` (bundled) or
point the GUI to a script
```

````
## Setup
1. Create a folder next to your project root named `electron-installer` and copy
scaffold files there (src/, package.json, build/)
2. From inside `electron-installer` run:
   ```bash
   npm install
   npm run start
   ```
   This will open the installer UI.

## Build distributables
- To create platform-specific installers (requires native toolchains):
  ```bash
  npm run dist
  ```
  - On macOS to build `.dmg`/`.pkg` you need Xcode tooling.
  - On Windows you need to run the build on Windows to produce `.exe`
(recommended), or use cross-compilation CI images.
  - For Linux AppImage and .deb, build on Linux.

## Security notes
- The UI writes `.env` into the same folder as the script before running it. You
can change this behaviour to pass env via stdin or other secure mechanisms.
- `sudo-prompt` elevates and runs the script; review the script before bundling.
````

What I *did not* include (optional improvements you may want):

- A secure secret-entry component that encrypts secrets before writing to disk.
- Integration with OS keychain (macOS Keychain / Windows Credential Manager).
- Rollback/undo features.
- Progress streaming inside the UI (advanced: use a native node child_process with PTY and stream stdout, but that requires bundling Node integration and tighter security considerations).

Next steps I can do immediately for you (pick any, I will generate files):

1. Create the full `electron-installer/` folder as a downloadable zip in the workspace (ready-to-run).
2. Modify the renderer to stream live logs incrementally (safer use of Node child_process and pty).
3. Add a settings page that maps `.env.example` keys into the UI form fields automatically.
4. Create CI config (GitHub Actions) to build installers for Linux/macOS/Windows.

Reply with numbers (like `1` or `1,3`) or say `start` and I will generate the chosen items.