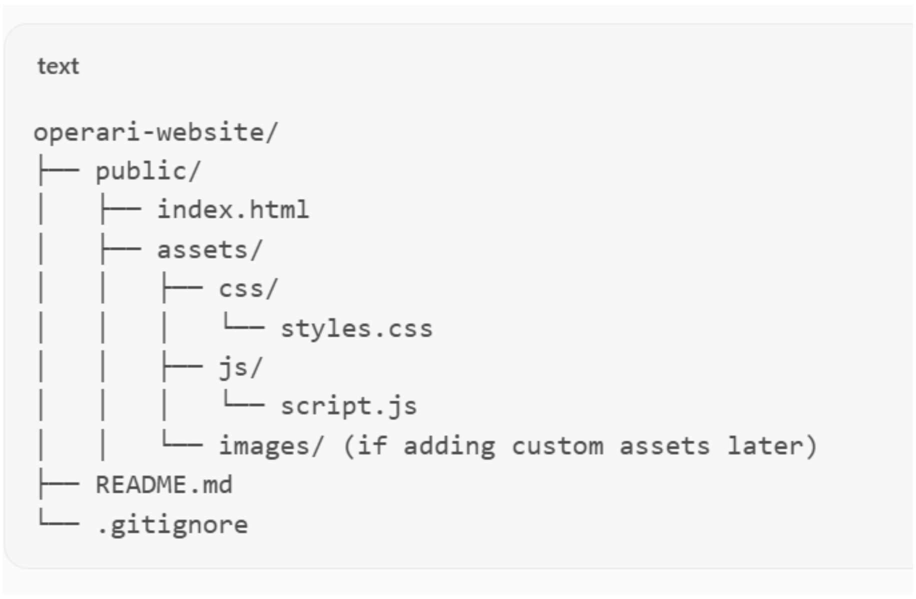
## 1. Project File Structure

This structure separates concerns for better maintainability and scalability.



text

```
operari-website/
├── public/
│   ├── index.html
│   ├── assets/
│   │   ├── css/
│   │   │   └── styles.css
│   │   ├── js/
│   │   │   └── script.js
│   │   └── images/ (if adding custom assets later)
├── README.md
└── .gitignore
```

## 2. public/index.html

This file contains the semantic HTML structure and links to the external assets.

</> HTML

## 3. public/assets/css/styles.css

This file contains all the styling, variables, and responsive queries.

## 4. public/assets/js/script.js

This file modularizes the JavaScript logic into a clean IIFE pattern, handling routing, UI interactions, and form validation.

## 5. Technical Documentation (README.md)

This documentation covers the coding standards, architecture, and deployment instructions.

# Operari Website – Technical Documentation

## 1. Overview

This is the production release of the Operari corporate website. It is a Single Page Application (SPA) designed for high performance, accessibility, and ease of maintenance.

---

## 2. Tech Stack

- **HTML5:** Semantic HTML5 structure
- **CSS3:** Modern CSS3 with Flexbox and Grid. No external CSS frameworks
- **JavaScript (ES6):** Vanilla JavaScript (ES6). No frameworks (React, Vue, etc.) to ensure maximum performance and minimal dependencies

**Libraries:**

- Font Awesome 6.4.0 (Icons)
- Google Fonts (Inter)

---

## 3. Coding Standards & Architecture

### 3.1 Modular Architecture

The codebase is separated into three main concerns to improve scalability and debugging:

- **index.html:** Structure and content
- **css/styles.css:** Styling, animations, and responsiveness
- **js/script.js:** Application logic, state management, and event handling

---

### 3.2 Naming Conventions

- **Files:** kebab-case (e.g., `styles.css`, `script.js`)
- **Classes:** kebab-case with hyphens (e.g., `card-3d-wrapper`)
- **Variables:** camelCase (e.g., `currentPage`)
- **Constants:** UPPER_SNAKE_CASE (e.g., `ROOT_URL`)

---

### 3.3 State Management

The application state is encapsulated within the `App` object in `script.js`. This prevents global namespace pollution and allows for easier debugging (e.g., `App.state.currentPage`).

---

### 3.4 Performance Optimisations

- **CSS Hardware Acceleration:** Animations use `transform` and `opacity` properties instead of `top`, `left`, `width`, or `height`
- **Passive Event Listeners:** Used where appropriate
- **Lazy Loading:** Images are loaded via standard browser lazy-loading or placeholders

---

## 4. Backend & APIs

- **Current Implementation:** Static site
- **Backend:** None (static HTML)
- **APIs:** None
- **Data Persistence:** Client-side session state only (routing history)

**Future Integration:**
If dynamic functionality (e.g., backend contact forms) is required, the modular `script.js` structure allows easy integration with Fetch API calls to endpoints without rewriting the entire logic block.

---

## 5. Deployment Instructions

### 5.1 Getting a Shareable URL (GitHub / GitLab / Netlify)

To make the website publicly accessible, follow one of the options below.

*Option A: Netlify Drop (Easiest for Static Sites)*

1. Compress the `public` folder into a ZIP file (e.g., `operari-site.zip`).
2. Go to `app.netlify.com/drop`.
3. Drag and drop the ZIP file.
4. Netlify will provide a random URL (e.g., `https://operari-site.netlify.app`), which can be used as-is or updated via DNS.

**Result:** A shareable URL is created instantly.

---

#### *Option B: GitHub Pages*

1. Create a new repository on GitHub (e.g., `github.com/yourusername/operari-website`).
2. Upload the file structure provided in this release.
3. Go to **Settings → Pages**.
4. Select the **main** branch as the source and save.

**Result:** The site will be live at `https://yourusername.github.io/operari-website/`

---

#### *Option C: Deploy to www.operari.co.uk*

1. Build the site locally (if a build step exists; otherwise, the HTML is already built).
2. Upload the contents of the `public` folder to your hosting provider's `public_html` or `www` directory via FTP or SFTP (e.g., cPanel, Plesk, AWS S3).
3. **DNS Configuration:** Ensure your domain A record points to your host's IP address.

---

### 5.2 Browser Compatibility

- Chrome / Edge (Recommended)
- Firefox
- Safari
- Mobile browsers (iOS Safari, Chrome Mobile)

---

### 5.3 Maintenance

To update content:

- Edit `index.html` for structural changes
- Edit `css/styles.css` for visual updates
- Edit `js/script.js` for logic changes
- Upload the modified files to your hosting provider

---

### 5.4 Local Development (Pre-launch)

1. Download the files.
2. Use a local server (e.g., VS Code Live Server extension) to open `index.html`.
3. Test routing and responsiveness locally before deployment.