Rail Acronym Centre - Technical Documentation

Step by step on how this app works

This is a simple static web application that's just three files plus your data. No servers, no databases, no complicated setup. It's like a digital flashcard system for railway terms.

What's in the Box

You've got four essential files:

1. **index.html** - The webpage itself
2. **style.css** - Makes it look nice
3. **script.js** - The brains of the operation
4. **BritishRailTerms.csv** - Your data (this is what you edit)

How It All Works Together

The Flow:

1. User opens index.html in a browser
2. Browser loads the JavaScript
3. JavaScript reads your CSV file
4. Everything happens right in the browser - no internet needed

The Magic Part (script.js): Here's what happens when you search:

Super simplified version:
1. Read CSV file
2. Split it into lines
3. First line = headers (id, term, fullForm, etc.)
4. Other lines = your railway terms
5. When you type, it filters through all terms
6. Shows matches as you type

The CSV File - Your Database

Your CSV is the heart of this. Think of it as a simple spreadsheet:

```
id,term,fullForm,category,description,commonUse
1,ETCS,European Train Control System,signalling,A standardized system...,Used in modern...
```

Rules for your CSV:

- First line must be those exact headers
- Keep commas between values
- If you have commas in your text, wrap that text in quotes: "This has, a comma"
- Save as plain CSV (not Excel workbook)

The JavaScript - In Plain English

Loading your data:

```
// This grabs your CSV file
fetch('BritishRailTerms.csv')
  .then(response => response.text())
  .then(csvText => {
    // Split it up and make it usable
    const terms = parseCSV(csvText);
    // Now it's ready to search
});
```

Searching: When you type "ETCS", the code:

1. Takes what you typed
2. Looks through all terms
3. Checks: term name, full form, description, category
4. Shows anything that matches

Displaying results: It shows matches on the left in a list. Click one, and it fills the right panel with all the details from your CSV.

Why This Design Works

Simplicity:

- No backend server needed
- No database to manage
- No login system
- Works completely offline

Easy Updates: You just edit a CSV file. That's it. Everyone knows spreadsheets.

Performance: All the data loads once at the start. After that, searching happens instantly because everything is already in memory.

Technical Details (For the Curious)

Browser Compatibility:

- Works in Chrome, Firefox, Safari, Edge (modern versions)
- Needs JavaScript enabled (it is by default)
- Can run from file:// (local files) without issues

Memory Usage: For 1,000 terms, you're looking at 1-2MB of memory.

CSV Parsing: The code handles:

- Regular lines: 1,ETCS,European...,signalling,...
- Lines with commas: 1,ETCS,"European, with comma",signalling,...
- Empty lines (just skips them)
- Different line endings (Windows/Mac/Linux)

If You Want to Modify It

Changing Colours: Open style.css, look for these lines at the top:

```
:root {
  --primary-color: #0057b8;  /* Main blue */
  --secondary-color: #e30613; /* Red accent */
  --accent-color: #009846;   /* Green */
}
```

Change those hex codes to your colours.

Adding a New Category:

1. In index.html, add a new button in the category section:

<button class="category-tag" data-category="your-new-category">Your Category</button>

2. In script.js, the filtering already handles any category name

Want to Add Images? You could add a column to your CSV called "imageUrl" and modify the JavaScript to display images. Would need about 10 lines of code.

Common Issues and Fixes

"My CSV changes aren't showing!"

- Did you save the CSV?
- Did you refresh the browser? (Ctrl+F5 forces a full refresh)
- Check for typos in the CSV format

"Search isn't working!" Open browser Developer Tools (F12):

- Console tab: Look for red errors
- Network tab: See if BritishRailTerms.csv loads

"It looks weird on my phone!"

- Check style.css has the @media queries at the bottom
- Make sure viewport meta tag is in the HTML

Security Considerations

What this application does NOT do:

- Doesn't send data anywhere
- Doesn't store anything
- Doesn't require internet
- Doesn't execute external code

What this application DOES do:

- Reads one CSV file from your local system
- Runs entirely in your browser sandbox
- Forgets everything when you close the tab

Performance Tips

For large datasets (1,000+ terms):

- Keep your descriptions concise
- Consider splitting into multiple CSV files if needed
- The search is case-insensitive and fast

Loading speed: The bottleneck is reading the CSV file. On a local network, 1,000 terms loads in under a second. Over the internet, depends on file size.

Extending the Application

Want to add more fields? Add columns to your CSV:

```
id,term,fullForm,category,description,commonUse,extraField1,extraField2
```

Then update the JavaScript to display those new fields in the detail view.

Want to add export functionality? Could add a button that exports search results as CSV. Would need about 50 lines of JavaScript.

Want to add favourites? Could use browser local Storage to save favourite terms. Simple to implement.

Deployment Scenarios

Single User (Local): Just open index.html. That's it.

Small Team (Network Drive): Put files on shared drive. Everyone opens from there.

Whole Company (Intranet): Put on web server. Works the same.

Read-Only Version: Remove the CSV editing instructions from your user guide. Give users just the HTML, CSS, JS, and a locked-down CSV.

Backup Strategy

Simple: Just copy the BritishRailTerms.csv file regularly.

Better: Add a version number to the filename:

- BritishRailTerms-v1.csv
- BritishRailTerms-v2.csv
- etc.

Best: Use version control (like Git) if you are technical.

The Bottom Line

This is a "keep it simple" application. It does one thing well: lets people search railway terms quickly.

The technical design choices:

1. CSV over database → Easy to edit
2. Static files over server → Easy to deploy
3. Client-side search → Instant results
4. Simple UI → No training needed

It's not fancy, but it works reliably and does exactly what railway teams need.

Built for reliability, not complexity.

Questions? Just ask.