JEB Incubator — Frontend Documentation (React + Vite)

Scope: This document covers the frontend only (React, Vite, SCSS). The backend is delivered by the server team.
Audience: Developers who will run, modify, or deploy the JEB Incubator frontend.

TABLE OF CONTENTS

1. OVERVIEW

JEB Incubator connects startups with investors, mentors, and resources.
The frontend is a React single-page application powered by Vite for fast local development and optimized builds. Styling is handled with SCSS. Typography follows the brand rule: Montserrat for headings, Open Sans for body text.

Key UI delivered:
• News: modern cards with image thumbnail, title, meta line, clamped title, “Read more” expand/collapse.
• Events: filterable grid (search, date range, location/type/audience), robust image fallback, expand/collapse with smooth scroll.

2. TECH STACK & REQUIREMENTS

• React 18+
• Vite (v5+ recommended)
• SCSS (Sass)
• Yarn as package manager (project includes yarn.lock)
• Node.js 18 LTS or newer (recommended)
• Google Fonts: “Montserrat” (headings), “Open Sans” (body)

Note: You can substitute npm or pnpm if you prefer, but scripts below assume Yarn.

3. QUICK START

From the frontend folder (e.g., Survivor/), run:

• Install dependencies
yarn

• Start dev server with hot-reload
yarn dev

• Production build
yarn build

• Preview the production build locally
yarn preview

Vite does not use “start” by default; use “dev” for development and “preview” to serve the built app.

4. PROJECT STRUCTURE

Typical layout (paths may vary slightly in your repository):

```
.
├─ index.html
├─ vite.config.js
├─ package.json
├─ yarn.lock
└─ src/
   ├─ main.jsx
   ├─ App.jsx
   ├─ assets/
   │  ├─ logo.png
   │  └─ Photo_default.svg
   ├─ apis/
   │  └─ BackendApi/
   │  └─ Event.api.js
   ├─ composant/
   │  ├─ Event/
   │  │  ├─ event.jsx
   │  │  └─ event.scss
   │  ├─ News/
   │  │  ├─ news.jsx
   │  │  └─ news.scss
   │  ├─ Home/
   │  │  └─ Home.scss
   │  ├─ Login/
   │  │  └─ Login.scss
   │  └─ template/
   │  └─ Footer/
   │  └─ Footer.scss
   └─ styles/ (optional global styles)
```

Recommended package.json scripts (example):

```
"scripts": {
"dev": "vite",
"build": "vite build",
```

```
"preview": "vite preview --port 5173",
"lint": "eslint "src/**/*.{js,jsx,ts,tsx}"",
"format": "prettier --write ."
}
```

5. CONFIGURATION (.env)

Create a file named .env at the frontend root:

VITE_API_BASE_URL=https://api.example.com

Notes:
- The code reads import.meta.env.VITE_API_BASE_URL and may fall back to http://localhost:4000 where specified.
- After changing .env, restart the dev server.

6. BRANDING: FONTS & COLORS

Fonts (index.html): include a single Google Fonts link for both families.

Global usage (example, in a global SCSS):

```
:root {
--font-heading: "Montserrat", system-ui, -apple-system, "Segoe UI", Roboto, Arial, sans-serif;
--font-body: "Open Sans", system-ui, -apple-system, "Segoe UI", Roboto, Arial, sans-serif;
}
html, body { font-family: var(--font-body); }
h1, h2, h3, h4, h5, h6 { font-family: var(--font-heading); }
```

Theme colors (used in News/Events SCSS; adjust as needed):

```
$brand-from: #F18585;
$brand-to: #C174F2;
$card: #10192e;
$card2: #0e1729;
$muted: rgba(255,255,255,.76);
```

7. UI MODULES

7.1) Events

Features:
• Filters: text search, date range (from/to), location, type, audience.
• Expand/collapse cards; smooth scrolling brings the opened card into view only if needed.
• Image fallback: if event image fails to load, a default SVG is used and cached for further renders.

Fallback image pattern (event.jsx, concept):
– Import once at top
```
import DefaultImg from "src/assets/Photo_default.svg";
```

– Use in <img>:
```
src={event.image || DefaultImg}
loading="lazy"
referrerPolicy="no-referrer"
onError={(ev) => {
if (ev.currentTarget.src.includes("Photo_default.svg")) return;
ev.currentTarget.onerror = null;
ev.currentTarget.src = DefaultImg;
}}
```

Smooth scroll only if needed (to avoid jumping when already visible):
```
function scrollIntoViewIfNeeded(el) {
const r = el.getBoundingClientRect();
const vh = window.innerHeight || document.documentElement.clientHeight;
if (r.top < 0 || r.bottom > vh) {
el.scrollIntoView({ behavior: "smooth", block: "center", inline: "nearest" });
}
}
```

Grid recommendation:
```
.events-grid {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(320px, 1fr));
```

gap: 16px;
}

7.2) News

Behavior:
• Collapsed state shows thumbnail, title, and meta line; the description appears only after “Read more”.
• Button alignment: each card has a minimum height and the “Read more” button uses margin-top: auto to stick to the bottom, resulting in a neat grid.

Examples:
.news-card { min-height: 420px; }
@media (max-width: 600px) { .news-card { min-height: 340px; } }
.news-card .more { margin-top: auto; }

If you render markdown content (from a CMS or rich text):
• Use react-markdown + remark-gfm for GitHub-flavored markdown.
• Sanitize if content can come from untrusted sources (rehype-sanitize).

8. STYLING & RESPONSIVE LAYOUT

• Use CSS Grid for card lists:
grid-template-columns: repeat(auto-fit, minmax(320px, 1fr));

• Thumbnail aspect ratios:
Use a 16/9 aspect-ratio container, object-fit: cover for images.

• Clamp long titles:
```
.title {
display: -webkit-box;
-webkit-box-orient: vertical;
-webkit-line-clamp: 2; /* or 3 */
overflow: hidden;
}
```

• Breakpoints frequently used: 900px, 600px.
• On very small screens, stack controls (filters) into 1 column and cards into 1 column.

9. ACCESSIBILITY (A11y)

• Expand/collapse buttons should expose state:
aria-expanded, aria-controls on the “Read more”/“Close” button.

• Provide meaningful alt text for images when available.
• Maintain visible focus rings for keyboard navigation.
• Ensure color contrast (text/background) meets WCAG AA where possible.

## 10. PERFORMANCE

• Use loading="lazy" for images.
• Prefer modern formats (avif/webp) for large visuals where applicable.
• Only add heavy libraries when necessary; consider dynamic import() to split code.
• If Vite warns about large chunks after build, create manualChunks in rollupOptions or use dynamic imports.

## 11. CODE QUALITY (LINT/FORMAT)

Recommended dev dependencies:
eslint
prettier
eslint-config-prettier
eslint-plugin-react
eslint-plugin-react-hooks

Example .eslintrc.json:

```
{
"env": { "browser": true, "es2021": true },
"extends": ["eslint:recommended", "plugin:react/recommended", "plugin:react-hooks/recommended"],
"parserOptions": { "ecmaVersion": "latest", "sourceType": "module" },
"settings": { "react": { "version": "detect" } },
"rules": { "react/prop-types": "off" }
}
```

Example .prettierrc:

```
{ "singleQuote": true, "semi": true, "printWidth": 100 }
```

Run:
yarn lint
yarn format

12. TESTING (OPTIONAL)

Suggested stack:
- Vitest for unit tests
- @testing-library/react for component tests
- jsdom + @testing-library/jest-dom for DOM assertions

Install:
yarn add -D vitest @testing-library/react @testing-library/jest-dom jsdom

Basic example:
– Render a card and assert title presence.
– Test expand/collapse toggling via user events.

13. BUILD & DEPLOYMENT (GITHUB PAGES)

1. Build the SPA
   yarn build
2. SPA fallback (avoid 404 on page refresh)
   Copy dist/index.html to dist/404.html
3. If deploying to a repository page (not username.github.io), set base in vite.config.js
   export default defineConfig({
   plugins: [react()],
   base: "/<REPO_NAME>/"
   })
4. Publish dist/ to the gh-pages branch (or use a GitHub Action).
   A workflow like .github/workflows/deploy-frontend-pages.yml can automate this.

Alternative: Netlify/Vercel
- Build command: vite build
- Output directory: dist
- Enable SPA redirect to index.html

## 14. TROUBLESHOOTING

Fonts not applied:

- Keep a single Google Fonts link in index.html.
- Use exact family names: "Montserrat" and "Open Sans".
- Check devtools → Network → confirm font files load successfully.

“Missing script: start” after build:

- Vite uses “dev” and “preview” scripts. For built preview run: yarn preview

API requests fail:

- Verify VITE_API_BASE_URL in .env and CORS settings on the backend.

Images fail to load:

- Ensure Photo_default.svg exists and that onError fallback logic is in place.

Open card scrolls incorrectly:

- Use the “scrollIntoViewIfNeeded” approach to center only when the card is out of the viewport.

15. WHERE TO CHANGE WHAT

Global fonts and brand colors:
- index.html → Google Fonts link
- SCSS variables (e.g., in news.scss/event.scss or a shared variables file)

Logo:
- src/assets/logo.png (referenced in index.html or header components)

Default image for events:
- src/assets/Photo_default.svg (used in event.jsx)

Events filters and mapping:
- src/composant/Event/event.jsx
– normalizeEvent mapper
– filters state (q, dates, location, type, audience)

News card behavior and layout:
- src/composant/News/news.jsx and news.scss

Global page container and layout:
- Check page-level SCSS (Home.scss, template/Footer.scss) and shared grid rules.

Vite base path (for GitHub Pages under a subpath):
- vite.config.js → base

Environment:
- .env → VITE_API_BASE_URL

## 16. APPENDIX: USEFUL SNIPPETS

Single Google Fonts link (index.html):

<link rel="preconnect" href="https://fonts.googleapis.com"> <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin> <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@600;700;800;900&family=Open+Sans:wght@400;600;700&display=swap" rel="stylesheet">

CSS grid for responsive cards:

```
.events-grid, .news-grid {
display: grid;
gap: 16px;
grid-template-columns: repeat(auto-fit, minmax(320px, 1fr));
}
```

Clamp long titles:

```
.title {
display: -webkit-box;
-webkit-box-orient: vertical;
-webkit-line-clamp: 2;
overflow: hidden;
}
```

Events smooth scroll (only if needed):

```
function scrollIntoViewIfNeeded(el) {
const r = el.getBoundingClientRect();
const vh = window.innerHeight || document.documentElement.clientHeight;
if (r.top < 0 || r.bottom > vh) {
el.scrollIntoView({ behavior: "smooth", block: "center", inline: "nearest" });
}
}
```

GH Pages SPA fallback:

After build, copy index.html → 404.html inside dist/

## 17. CHANGELOG / ROADMAP

Current (v1):

- News grid with expand/collapse, button anchored at card bottom.

- Events grid with filters, robust default image, smooth scroll logic.
- Brand fonts integrated (Montserrat/Open Sans).
- Vite build configured for static hosting.

Next:

- Add ESLint/Prettier configs to the repo if missing.
- Optional: Introduce unit tests (Vitest + React Testing Library).
- Lazy-load heavy modules; refine manualChunks if bundle grows.
- Enhance A11y coverage (keyboard testing, focus outlines, landmarks).
- Add Netlify/Vercel deploy templates and SPA redirects.