ChatGPT

Confirmed Issues and Causes

- **PDF.js Worker Misconfiguration:** The app's `setupPdfWorker.ts` hard-codes a worker URL `/pdf.worker.js` and CDN fallbacks for pdfjs-dist v4.8.69 [1]. In practice, Vite never bundles the worker file, so React-PDF falls back to a "fake worker" and then fails to load it – producing the console error `Failed to fetch dynamically imported module: pdf.worker.js?import` [2] [3]. Moreover, the default bundled worker was at version 3.11.174, causing a version mismatch with pdfjs-dist v4.8.69.

- **File Persistence/Ephemeral Storage:** Uploaded PDFs are written to `process.cwd()/data/uploads` (via multer), which on Replit (and similar hosts) is not persisted across restarts. Thus a newly uploaded PDF can render once, but after a restart the physical file is gone even though the DB record remains [4]. A MissingPDFException (`404 Not Found`) occurs on reload: the backend no longer finds the file, so the React viewer reports "Missing PDF...".

- **Incorrect File Paths in Download Route:** The download handler builds lookup paths inconsistently. The DB `filePath` is stored as `"/uploads/<filename>"` [5] (leading slash). The code then does `path.join(process.cwd(), 'public', document.filePath)` – but with a leading `/` this yields `"/uploads/..."` (root) instead of the real `"<project>/public/uploads/..."`. In effect, the second fallback path is wrong. Although the first (`data/uploads/<file>`) and third (`public/uploads/<file>`) entries should catch the file, the slash causes confusion and redundant checks [6] [5].

- **Missing Existence Checks:** The `/api/documents/:id/download` route streams the file as soon as it finds it, but only returns 404 after exhausting all fallbacks [7]. This delay means React-PDF only sees the 404 after attempting to fetch, triggering MissingPDFException. It would be better to check existence up front and immediately return 404 with a clear message if the file is gone.

- **React-PDF `<Document>` Props Not Memoized:** The `<Document>` component in `PdfViewer.tsx` is given an inline `options={{…}}` object on each render [8]. This triggers a warning (`Options prop passed to <Document/> changed… consider memoizing`) [9] and can cause unnecessary reloads.

- **Accessibility Labels:** Some icon buttons (zoom, close, etc.) lack `aria-label`s. While the sidebar uses ARIA attributes (e.g. `role="button"`, `aria-label` on list items) [10] [11], the PDF viewer toolbar buttons should likewise have accessible names (e.g. `<Button aria-label="Zoom in">…</Button>`).

Refactor Plan

1. **Fix the PDF Worker Setup.** In `client/src/lib/setupPdfWorker.ts`, replace the static path with Vite's `?url` import. For example:

   ```
   import { pdfjs } from 'react-pdf';
   import workerURL from 'pdfjs-dist/build/pdf.worker.min.js?url';
   pdfjs.GlobalWorkerOptions.workerSrc = workerURL;
   ```

   Remove the `WORKER_SOURCES[0] = '/pdf.worker.js'` fallback. This forces Vite to bundle the correct worker file. Ensure `resetPdfWorker` also uses the same mechanism if fallbacks are needed. After this, the DevTools Network tab should show `pdf.worker.min.js` loading with 200, and the "Setting up fake worker" warning will disappear [3] [12].

2. **Use a Persistent Upload Directory.** Change the upload destination from `path.join(process.cwd(), 'data', 'uploads')` to a directory guaranteed persistent (e.g. `path.join(process.cwd(), 'replit-storage', 'uploads')` on Replit, or an environment-provided path). Update the multer config accordingly, and ensure the folder is created if missing (as suggested in [48†L114-L117]). This ensures files survive restarts. For example:

   ```
   const UPLOAD_DIR = path.join(process.cwd(), 'replit-storage', 'uploads');
   // ... in multer.diskStorage.destination:
   if (!fs.existsSync(UPLOAD_DIR)) fs.mkdirSync(UPLOAD_DIR, { recursive:
   true });
   cb(null, UPLOAD_DIR);
   ```

3. **Correct the Stored File Path.** Remove the leading slash when saving `filePath`. Change:

   ```
   const filePath = `/uploads/${path.basename(req.file.path)}`;
   ```

   to something like:

   ```
   const base = path.basename(req.file.path);
   const filePath = `uploads/${base}`;  // no leading slash
   ```

   (Alternatively use `path.join` to build the relative path.) This way, `path.join(process.cwd(), 'public', filePath)` will resolve properly.

4. **Simplify Download Lookup and Add Existence Check.** In the `/download` route, streamline the lookup logic. For example:

```
const relPath = document.filePath; // e.g. 'uploads/foo.pdf'
const tryPaths = [
  path.join(PERSIST_PATH, relPath),
  path.join(PUBLIC_PATH, relPath)
];
let foundPath = null;
for (const p of tryPaths) {
  if (fs.existsSync(p)) { foundPath = p; break; }
}
if (!foundPath) {
  console.error(`File not found:`, tryPaths);
  return res.status(404).json({ error: 'File not found', message: 'Please
re-upload the document.' });
}
res.setHeader('Content-Type', document.fileType);
res.setHeader('Cache-Control', 'no-store, must-revalidate');
if (document.fileType === 'application/pdf') {
  res.setHeader('Content-Disposition', `inline; filename="$
{encodeURIComponent(document.fileName)}"`);
} else {
  res.setHeader('Content-Disposition', `attachment; filename="$
{encodeURIComponent(document.fileName)}"`);
}
const stream = fs.createReadStream(foundPath);
stream.pipe(res).on('error', err => {
  console.error('Stream error', err);
  res.status(500).json({ error: 'Stream error', message: 'Could not send
file.' });
});
```

This immediately 404s if no file exists. It removes the redundant search by similar names (that code was a last-ditch “maybe the filename was mangled” attempt) because correct pathing should avoid that scenario.

5. **Improve** `PdfViewer` **Error Handling.** The existing `onDocumentLoadError` does a HEAD-check and switches to fallback if needed [13]. Ensure it properly handles 404: if `fetch(current.downloadUrl, {method: 'HEAD'})` returns 404, immediately show a user-friendly toast like “Document file not found, please re-upload” (as it already does). Also consider using `onError` on the `<iframe>` fallback to catch failures there. (The code already does this for basic viewer [14].)

6. **Memoize PDF Options.** In `PdfViewer.tsx`, wrap the `options` prop in a `useMemo` so it isn’t re-created on every render. For example:

```
const pdfOptions = useMemo(() => ({
  cMapUrl: 'https://cdn.jsdelivr.net/npm/pdfjs-dist@4.8.69/cmaps/',
  cMapPacked: true,
  standardFontDataUrl: 'https://cdn.jsdelivr.net/npm/pdfjs-dist@4.8.69/
standard_fonts/',
}), []);
// ...
<Document options={pdfOptions} file={current.downloadUrl} …>
```

This removes the React warning about the `options` prop changing [9] and prevents unnecessary reloads of the document.

7. **Enhance Accessibility.** Add `aria-label` to the PDF viewer buttons (zoom in/out, rotate, download, close, etc.). For instance:

```
<Button variant="outline" size="sm" onClick={zoomIn} aria-label="Zoom in"
…>
  +
</Button>
<Button variant="outline" size="sm" onClick={zoomOut} aria-label="Zoom
out" …>
  -
</Button>
// etc.
```

Ensure the `<iframe>` has a `title` (already set to the file name) and consider adding `role="document"` or similar to the container div for screen readers. The sidebar already uses `aria-label` on each document row [10], which is good. Also ensure icons (e.g. Download icon) have either accompanying text (they do) or `aria-hidden`.

8. **Modularize and Clean Up.** After these fixes, review code for clarity. For example, extract file-serving logic into a helper function, and split the long upload handler into middleware + handler (the code already does some chaining). Add server-side tests (e.g. supertest) to verify the download route returns 200 + correct content-type and 404 when missing. Optionally, implement a cleanup script to remove DB records whose files are gone (as noted in [48†L119-L128]).

By following this plan, uploaded PDFs will be reliably stored and served. The front-end will use the correct PDF.js worker (so the "fake worker" error goes away) [3] [12], and the viewer will handle missing files gracefully with user feedback [4] [13]. Options will be memoized to eliminate warnings, and UI buttons will be accessible. The documents module will become more robust, modular, and maintainable.

References: Code excerpts are from the project's repository (e.g. `documents.ts` and `PdfViewer.tsx`), and diagnostic posts explaining these issues [2] [4] [3] [1].

---

[1] setupPdfWorker.ts
https://github.com/0xGonz/DealFlowLifecyclev1/blob/96d3b89b3a9a3e1f679a17131a07885fa1ca173c/client/src/lib/setupPdfWorker.ts

[2] [9] Pasted-Query-data-received-from-api-deals-86-Object-id-86-name-Syntrillo-description-Biotech-st-1747186774617.txt
https://github.com/0xGonz/DealFlowLifecyclev1/blob/96d3b89b3a9a3e1f679a17131a07885fa1ca173c/attached_assets/Pasted-Query-data-received-from-api-deals-86-Object-id-86-name-Syntrillo-description-Biotech-st-1747186774617.txt

[3] Pasted-Your-API-calls-are-still-succeeding-the-only-thing-blowing-up-is-the-pdf-js-worker-bootstrap-B-1747015160009.txt
https://github.com/0xGonz/DealFlowLifecyclev1/blob/96d3b89b3a9a3e1f679a17131a07885fa1ca173c/attached_assets/Pasted-Your-API-calls-are-still-succeeding-the-only-thing-blowing-up-is-the-pdf-js-worker-bootstrap-B-1747015160009.txt

[4] Pasted-Thanks-Brett-this-looks-like-a-persistent-documents-upload-and-storage-bug-causing-PDFs-to-disapp-1747185801113.txt
https://github.com/0xGonz/DealFlowLifecyclev1/blob/96d3b89b3a9a3e1f679a17131a07885fa1ca173c/attached_assets/Pasted-Thanks-Brett-this-looks-like-a-persistent-documents-upload-and-storage-bug-causing-PDFs-to-disapp-1747185801113.txt

[5] [6] [7] documents.ts
https://github.com/0xGonz/DealFlowLifecyclev1/blob/96d3b89b3a9a3e1f679a17131a07885fa1ca173c/server/routes/documents.ts

[8] [13] [14] PdfViewer.tsx
https://github.com/0xGonz/DealFlowLifecyclev1/blob/96d3b89b3a9a3e1f679a17131a07885fa1ca173c/client/src/components/documents/PdfViewer.tsx

[10] [11] Sidebar.tsx
https://github.com/0xGonz/DealFlowLifecyclev1/blob/96d3b89b3a9a3e1f679a17131a07885fa1ca173c/client/src/components/documents/Sidebar.tsx

[12] Pasted-What-the-console-is-telling-you-decoded-Backend-is-fine-GET-api-documents-deal-83-200-OK-and-yo-1747013622373.txt
https://github.com/0xGonz/DealFlowLifecyclev1/blob/96d3b89b3a9a3e1f679a17131a07885fa1ca173c/attached_assets/Pasted-What-the-console-is-telling-you-decoded-Backend-is-fine-GET-api-documents-deal-83-200-OK-and-yo-1747013622373.txt