View Button Debugging Fix

Problem Statement

The View button in the onboarding section doesn't work - nothing happens when clicked. Files are correctly uploaded to S3 and references are stored in the database, but clicking the View button produces no visible action or errors.

Debugging Improvements Implemented

1. Enhanced Frontend Logging (`my-onboarding/page.tsx`)

Added comprehensive console logging to the `handleViewFile` function to trace the entire execution flow:

Logging Points:
- ✅ Button click detection
- ✅ File path reception and validation
- ✅ Loading state changes
- ✅ Toast notification display
- ✅ downloadFile function call
- ✅ Signed URL generation result
- ✅ window.open execution and result
- ✅ Error capture with full stack traces

Fallback Mechanism:
- Added fallback download link creation if window.open is blocked by popup blockers
- Creates and clicks an `<a>` element as an alternative approach
- Provides better error messages to diagnose issues

Code Location: Lines 168-225

```
const handleViewFile = async (filePath: string) => {
  console.log("=== HANDLE VIEW FILE START ===");
  console.log("1. Button clicked!");
  console.log("2. File path received:", filePath);
  // ... 14 detailed logging points

  // Fallback mechanism
  if (!newWindow) {
    const link = document.createElement('a');
    link.href = url;
    link.target = '_blank';
    link.rel = 'noopener noreferrer';
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
  }
};
```

2. Enhanced S3 Function Logging (`lib/s3.ts`)

Added detailed logging to the `downloadFile` function to trace S3 operations:

Logging Points:
- ✅ Input key validation
- ✅ Bucket name and folder prefix verification
- ✅ Key transformation via buildKey function
- ✅ Content type detection
- ✅ Inline display determination
- ✅ File name extraction
- ✅ S3 command creation
- ✅ Signed URL generation
- ✅ Error capture in S3 operations

Code Location: Lines 116-166

```
export async function downloadFile(key: string, expiresIn: number = 3600):
Promise<string> {
  console.log("=== DOWNLOAD FILE (S3) START ===");
  console.log("1. Input key:", key);
  console.log("2. Bucket name:", bucketName);
  console.log("3. Folder prefix:", folderPrefix);
  // ... 11 detailed logging points

  try {
    const signedUrl = await getSignedUrl(s3Client, command, { expiresIn });
    console.log("10. Signed URL generated successfully");
    return signedUrl;
  } catch (error) {
    console.error("=== ERROR IN DOWNLOAD FILE (S3) ===");
    throw error;
  }
}
```

What The Debugging Will Reveal

When you click the View button, the browser console will now show:

Success Case

```
=== HANDLE VIEW FILE START ===
1. Button clicked!
2. File path received: uploads/onboarding/user123/question456/1234567890_document.pdf
3. File path type: string
4. File path length: 63
5. Setting loading state...
6. Showing toast notification...
7. Calling downloadFile function...
=== DOWNLOAD FILE (S3) START ===
1. Input key: uploads/onboarding/user123/question456/1234567890_document.pdf
2. Bucket name: your-bucket-name
3. Folder prefix: payroll-saas/
4. Final key after buildKey: payroll-saas/uploads/onboarding/user123/question456/
1234567890_document.pdf
5. Detected content type: application/pdf
6. Display inline: true
7. File name: 1234567890_document.pdf
8. Command created: {...}
9. Generating signed URL...
10. Signed URL generated successfully
11. URL preview: https://your-bucket.s3.amazonaws.com/payroll-saas/uploads/...
=== DOWNLOAD FILE (S3) END ===
8. Signed URL generated successfully!
9. Opening URL in new window...
10. Window.open result: [object Window]
11. File opened successfully in new tab
14. Clearing loading state...
=== HANDLE VIEW FILE END ===
```

Failure Cases Will Show

Case 1: Button Click Not Registered

- If you don't see "=== HANDLE VIEW FILE START ===" at all, the onClick handler isn't attached

Case 2: File Path Issues

- If step 2 shows `undefined` or wrong path, database format doesn't match expectations

Case 3: S3 Configuration Issues

- If logging stops at step 7-9, S3 credentials or bucket config is wrong

Case 4: URL Generation Failures

- Error logs will show exact AWS SDK errors with stack traces

Case 5: Popup Blocker

- Step 10 will show `null` and fallback mechanism will activate

Testing Instructions

Prerequisites

1. Ensure `.env` file is properly configured with:
   ```
   env
   DATABASE_URL="postgresql://..."
   NEXTAUTH_SECRET="..."
   NEXTAUTH_URL="http://localhost:3000"
   ```

```
AWS_BUCKET_NAME="..."
AWS_FOLDER_PREFIX="..."
AWS_ACCESS_KEY_ID="..."
AWS_SECRET_ACCESS_KEY="..."
AWS_REGION="..."
```

2. Database should be migrated and seeded:
```
bash
npx prisma generate
npx prisma db push
npx prisma db seed  # If seed script exists
```

Testing Steps

1. **Start the development server:**
```
bash
npm run dev
```

2. **Log in as a regular user (Payroll role)**

3. **Navigate to "My Onboarding" page**

4. **Open Browser DevTools** (F12) and go to Console tab

5. **Click on any "View File" button** for an uploaded document

6. **Observe the console output:**
   - Look for the detailed logging output
   - Check each numbered step
   - Identify where the process fails (if it does)

7. **Check for:**
   - Toast notifications appearing
   - Button loading state changing
   - New tab opening (or fallback link click)
   - Any error messages in console

What To Look For

✅ **If everything works:**
- All 14 log points appear in sequence
- File opens in new tab
- No errors in console

❌ **If it fails:**
- Note which log point is the last one that appears
- Copy the full error message and stack trace
- Check the file path format in the database
- Verify S3 credentials and permissions

Potential Issues & Solutions

Issue 1: Nothing in Console

Symptom: Clicking View button shows no console logs at all

Cause: onClick handler not properly attached or React component not re-rendering

Solution:
- Check if button is actually clickable (not covered by another element)
- Verify `loadingFile` state isn't blocking clicks
- Check React DevTools for component state

Issue 2: File Path Format Mismatch

Symptom: Logs show unexpected file path format

Database stores: `https://bucket.s3.amazonaws.com/path/file.pdf`
Code expects: `path/file.pdf`

Solution:
- Extract S3 key from full URL in handleViewFile before calling downloadFile
- Update database to store only the S3 key, not full URL

Issue 3: S3 Credentials Error

Symptom: Error at step 9-10 in downloadFile

Cause: Invalid AWS credentials or permissions

Solution:
- Verify AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
- Check IAM user has `s3:GetObject` permission on the bucket
- Verify bucket name is correct

Issue 4: CORS Issues

Symptom: File opens but shows access denied

Cause: S3 bucket CORS not configured for signed URLs

Solution:

```
{
  "AllowedOrigins": ["http://localhost:3000"],
  "AllowedMethods": ["GET"],
  "AllowedHeaders": ["*"],
  "MaxAgeSeconds": 3000
}
```

Issue 5: Popup Blocker

Symptom: Step 10 shows `null`, but fallback creates link

Cause: Browser blocking window.open

Solution:
- Fallback mechanism will handle this automatically
- User might need to allow popups for localhost:3000

Files Modified

1. **app/(dashboard)/(modules)/onboarding/my-onboarding/page.tsx**
   - Enhanced `handleViewFile` function with extensive logging
   - Added fallback mechanism for popup blockers
   - Lines 168-225

2. **lib/s3.ts**
   - Enhanced `downloadFile` function with detailed S3 operation logging
   - Added error handling and logging
   - Lines 116-166

Next Steps

1. **Test with the debugging enabled** - This will reveal the exact point of failure
2. **Check database file path format** - Ensure it matches what the code expects
3. **Verify S3 configuration** - Bucket, credentials, and permissions
4. **Check browser console** - Look for any JavaScript errors not caught by our logging
5. **Test different file types** - PDF, images, documents

Rollback

If you need to remove the verbose logging after debugging:

```
git diff HEAD app/(dashboard)/(modules)/onboarding/my-onboarding/page.tsx
git diff HEAD lib/s3.ts
```

Or keep the fallback mechanism and just remove console.log statements.

Additional Improvements Made

Fallback Download Mechanism

If `window.open()` is blocked by popup blockers, the code now:
1. Creates an invisible `<a>` element
2. Sets href to the signed URL
3. Sets target=”_blank” and rel=”noopener noreferrer”
4. Programmatically clicks it
5. Removes the element from DOM

This provides a more robust user experience.

Better Error Messages

All errors now include:
- Error object
- Error message
- Full stack trace
- Context about where the error occurred

Date: December 23, 2025
Branch: fix/enum-casing-mismatch
Status: Ready for testing with proper environment configuration