

# Onboarding File Viewing Fix - Documentation

## Changes Made

### 1. Improved S3 Path Structure ✓

#### Previous Structure:

```
uploads/onboarding/{timestamp}-{filename}
```

#### New Structure:

```
uploads/onboarding/{userId}/{questionId}/{timestamp}_{filename}
```

#### Benefits:

- **Better Organization:** Files are organized by user and question
- **Easy Identification:** Can quickly find which question a file belongs to
- **Conflict Prevention:** Timestamp + organized structure prevents filename conflicts
- **Better Security:** Easier to implement user-specific access control
- **Easier Cleanup:** Can delete all files for a user or question easily

#### Implementation:

- **File:** `app/api/upload/route.ts` (lines 32-72)
- The upload API now accepts `userId` and `questionId` in `FormData`
- Falls back to old structure for non-onboarding uploads (backward compatible)

### 2. Fixed File Viewing for All File Types ✓

#### Problem:

The `downloadFile` function was hardcoded to use `application/pdf` content type, which broke viewing for:

- Images (JPEG, PNG, GIF)
- Word Documents (.doc, .docx)
- Other file types

#### Solution:

Created smart content-type detection based on file extension:

#### New Functions Added (`lib/s3.ts`):

1. `getContentTypeFromKey(key: string)` : Detects content type from file extension
2. `shouldDisplayInline(contentType: string)` : Determines if file should open inline or download

#### Supported File Types:

- **Images:** JPG, PNG, GIF, WebP, SVG → Opens inline in browser
- **PDFs:** → Opens inline in browser
- **Word/Excel/PowerPoint:** → Downloads automatically

- **Text/CSV:** → Opens inline in browser
- **Archives:** → Downloads automatically

#### **Implementation Files:**

- `lib/s3.ts` : Updated `downloadFile()` and `getSignedUrlForKey()` functions
  - Signed URLs expire after 1 hour (configurable)
- 

## **3. Enhanced View Button Functionality**

#### **Improvements:**

- 1. Loading State:** Button shows “Loading...” while generating signed URL
- 2. Better Error Handling:** Detailed error messages with console logging
- 3. User Feedback:** Toast notifications for:
  - Generating secure link
  - File opened successfully
  - Pop-up blocker warnings
  - Error messages
- 4. Button Disabled During Load:** Prevents multiple clicks

#### **Updated Files:**

- `app/(dashboard)/(modules)/onboarding/my-onboarding/page.tsx`
- `app/(dashboard)/(modules)/onboarding/page.tsx`

#### **Changes:**

```
// Added loading state tracking
const [loadingFile, setLoadingFile] = useState<string | null>(null);

// Enhanced error handling
const handleViewFile = async (filePath: string) => {
  try {
    setLoadingFile(filePath);
    toast.info("Generating secure link...");

    const url = await downloadFile(filePath);
    const newWindow = window.open(url, "_blank");

    if (!newWindow) {
      toast.warning("Please allow pop-ups to view the file");
    } else {
      toast.success("File opened in new tab");
    }
  } catch (err: any) {
    console.error("Error viewing file:", err);
    toast.error("Failed to open file: " + (err.message || "Unknown error"));
  } finally {
    setLoadingFile(null);
  }
};
```

## 4. Frontend Upload Updates ✓

Updated the file upload flow to pass `userId` and `questionId`:

**File:** `app/(dashboard)/(modules)/onboarding/my-onboarding/page.tsx`

```
const handleSubmitFile = async () => {
  // ... validation ...

  const formData = new FormData();
  formData.append("file", file);
  formData.append("type", "onboarding");
  formData.append("userId", data.id);           // NEW
  formData.append("questionId", currentQuestion.id); // NEW

  // ... upload logic ...
};
```

## Testing Guide

### Prerequisites:

1. AWS S3 bucket configured with credentials
2. User with onboarding template assigned
3. Multiple file types for testing (PDF, image, Word doc)

### Test Scenarios:

#### Test 1: Upload with New Path Structure

1. Go to “My Onboarding” page
2. Select a file upload question
3. Upload a file (e.g., PDF)
4. Check S3 bucket - verify path is: `uploads/onboarding/{userId}/{questionId}/{timestamp}_{filename}`

**Expected Result:** ✓ File stored in organized path structure

#### Test 2: View PDF File

1. Upload a PDF file to an onboarding question
2. Click “View File” button
3. Verify:
  - Button shows “Loading...” during load
  - Toast notification appears
  - PDF opens in new tab
  - PDF displays inline (not downloaded)

**Expected Result:** ✓ PDF opens inline in new browser tab

### Test 3: View Image File

1. Upload an image (JPG/PNG) to an onboarding question
2. Click “View File” button
3. Verify:
  - Image opens in new tab
  - Image displays inline
  - Correct image is shown

**Expected Result:**  Image displays inline in browser

---

### Test 4: View Word Document

1. Upload a Word document (.docx) to an onboarding question
2. Click “View File” button
3. Verify:
  - File downloads automatically
  - Filename is preserved
  - File can be opened in Word

**Expected Result:**  Word doc downloads automatically

---

### Test 5: Error Handling

1. Modify a file path in database to an invalid S3 key
2. Try to view the file
3. Verify:
  - Error toast appears with message
  - Console shows detailed error
  - Button returns to normal state

**Expected Result:**  Error handled gracefully

---

### Test 6: Admin View (All Onboardings)

1. Login as admin
2. Go to “All Onboardings” page
3. Click “View Details & Review” for a user with file uploads
4. Click “View File” button
5. Verify file opens correctly

**Expected Result:**  Admin can view user files

---

### Test 7: Multiple File Types

Test with various file types:

-  PDF (should open inline)
-  JPG/PNG (should open inline)

- DOCX (should download)
  - GIF (should open inline)
- 

## Test 8: Loading States

1. Upload and view a file
2. While loading, verify:
  - Button shows “Loading...”
  - Button is disabled
  - Cannot click again
  - Returns to normal after load

**Expected Result:**  Loading state prevents multiple clicks

---

## Backward Compatibility

### Handling Existing Files:

The code maintains backward compatibility:

1. **Old Path Format Still Works:** Files uploaded before this fix can still be viewed
2. **Content Type Detection:** Works for any file path format
3. **Fallback Structure:** Non-onboarding uploads use original path structure

### Migration (Optional):

If you want to migrate existing files to the new structure:

```
// Example migration script (not included)
// 1. Query all onboarding responses with files
// 2. For each file:
//   - Download from old path
//   - Upload to new path
//   - Update database record
//   - Delete old file
```

---

## Security Considerations

### Signed URLs:

- URLs expire after **1 hour** (configurable in `downloadFile()`)
- Each view generates a new signed URL
- Private S3 bucket recommended

### Path Structure Benefits:

- Files organized by user → easier to implement access control
- Can add S3 bucket policies to restrict access by path prefix
- Reduces risk of filename collisions

## Recommendations:

1. Use private S3 bucket (not public)
  2. Enable S3 versioning for file history
  3. Set lifecycle policies to archive old files
  4. Monitor S3 access logs
- 

## File Structure Changes

### Modified Files:

app/api/upload/route.ts	- Upload API with new path
structure	
lib/s3.ts	- Content type detection
+ signed URLs	
app/(dashboard)/(modules)/onboarding/my-onboarding/page.tsx	- User view with loading states
app/(dashboard)/(modules)/onboarding/page.tsx	- Admin view with loading states

### New Functions:

```
lib/s3.ts:
- getContentTypeFromKey(key: string): string
- shouldDisplayInline(contentType: string): boolean
```

## Troubleshooting

### Issue: “Failed to open file”

#### Possible Causes:

- S3 credentials invalid or expired
- File doesn't exist in S3
- Incorrect S3 bucket name
- Network connectivity issue

#### Solution:

1. Check AWS credentials in .env
  2. Verify file exists in S3 console
  3. Check browser console for detailed error
  4. Verify S3 bucket permissions
- 

### Issue: File downloads instead of opening inline

#### Possible Causes:

- Content type not detected correctly
- File extension not in supported list

**Solution:**

1. Add file extension to `getContentTypeFromKey()` mapping
  2. Add content type to `shouldDisplayInline()` list
- 

**Issue: Pop-up blocker prevents file from opening****Cause:** Browser blocking `window.open()`**Solution:**

- User receives toast notification to allow pop-ups
  - User must allow pop-ups for the site in browser settings
- 

**Performance Considerations****Signed URL Generation:**

- Signed URLs are generated on-demand (not cached)
- Generation is very fast (~50-100ms)
- URLs expire after 1 hour

**Optimization Opportunities:**

1. **Cache signed URLs** (if same file viewed multiple times within hour)
  2. **Pre-generate URLs** (when loading list of files)
  3. **Use CloudFront** (if serving many files globally)
- 

**Future Enhancements****Potential Improvements:**

1. **File Preview Modal:** Show preview in modal instead of new tab
  2. **Download Button:** Separate “View” and “Download” buttons
  3. **File Metadata:** Show file size, upload date in UI
  4. **Bulk Download:** Download multiple files as ZIP
  5. **File Versioning:** Track file changes over time
  6. **Thumbnail Generation:** Generate thumbnails for images/PDFs
  7. **File Compression:** Compress large files before upload
-

## Support Content Types

---

### Currently Supported:

File Type	Extension	Content Type	Display
PDF	.pdf	application/pdf	Inline
JPEG	.jpg, .jpeg	image/jpeg	Inline
PNG	.png	image/png	Inline
GIF	.gif	image/gif	Inline
WebP	.webp	image/webp	Inline
SVG	.svg	image/svg+xml	Inline
Text	.txt	text/plain	Inline
CSV	.csv	text/csv	Inline
Word	.doc	application/msword	Download
Word	.docx	application/vnd... wordprocessingml.document	Download
Excel	.xls	application/vnd.ms-excel	Download
Excel	.xlsx	application/vnd... spreadsheetml.sheet	Download
PowerPoint	.ppt	application/vnd.ms-powerpoint	Download
PowerPoint	.pptx	application/vnd... presentationml.presentation	Download
ZIP	.zip	application/zip	Download
RAR	.rar	application/x-rar-compressed	Download

### To Add New File Type:

Edit `lib/s3.ts` → `getContentTypeFromKey()` :

```
const contentTypes: Record<string, string> = {
  // ... existing types ...
  'newext': 'application/new-type',
};
```

## Summary

### ✓ Problems Fixed:

1. ✗ Disorganized S3 path structure → ✓ Organized by user/question
2. ✗ View button didn't work → ✓ Works for all file types
3. ✗ Only PDFs could be viewed → ✓ All file types supported
4. ✗ No loading feedback → ✓ Loading states + toast notifications
5. ✗ Poor error handling → ✓ Detailed error messages

### 🎯 Benefits Achieved:

- Better file organization in S3
- Proper content type handling
- Enhanced user experience with loading states
- Improved error handling and feedback
- Maintained backward compatibility
- Security improvements with signed URLs

## Questions or Issues?

If you encounter any issues or have questions about this implementation, please check:

1. Browser console for detailed error messages
2. S3 bucket permissions and configuration
3. AWS credentials in environment variables
4. File paths in database records

**Last Updated:** December 23, 2025

**Version:** 1.0.0

**Author:** DeepAgent AI