Document Preview Images Fix

Problem

Documents in the inbox were showing white/blue placeholder images (FileText icons) instead of actual document preview thumbnails.

Root Cause

The frontend components were using hardcoded `FileText` icons instead of displaying the actual thumbnail images that were being generated and stored in the backend.

Solution Overview

The fix involved three main areas:

1. Frontend Display Logic (client/src/pages/inbox.tsx)

Changed: Replaced hardcoded FileText icon with actual image rendering logic
- Now checks for `document.thumbnailPath` first
- Falls back to `document.filePath` for image files if no thumbnail exists
- Only shows FileText icon as final fallback
- Added proper error handling for failed image loads

2. Type Definitions (shared/types/inbox.ts)

Added: `thumbnailPath?: string` field to InboxItem type
- This ensures type safety across frontend and backend

3. Database Schema (shared/schema.ts)

Added: `thumbnailPath: varchar("thumbnail_path")` field to inboxItems table
- Stores the S3 URL or data URL of the thumbnail image

4. Backend API Updates

a. GET /api/inbox endpoint (server/routes.ts)

Changed: Added `thumbnailPath: inboxItems.thumbnailPath` to the select query
- Now returns thumbnail path in the API response

b. POST /api/inbox/register endpoint (server/routes/ai-inbox.ts)

Changed:
- Added `thumbnailPath` to RegisterBody schema
- Added `thumbnailPath: body.thumbnailPath` to the insert values
- Now saves thumbnail path when creating inbox items

5. Inbox Panel Component (client/src/components/family/inbox-panel.tsx)

Changed: Updated thumbnail rendering logic
- Prioritizes `item.thumbnailPath` over `item.fileUrl`

- Only shows FileText icon when no thumbnail or image is available
- Improved conditional rendering logic

Technical Details

Thumbnail Generation

The backend already had thumbnail generation logic in place (server/routes/trustworthy.ts):
- Uses Sharp library to create 300x300 thumbnails for images
- Uploads thumbnails to S3 or creates data URLs for local development
- Thumbnails are stored in the `thumbnailPath` field

Image Display Priority

1. **thumbnailPath** - Optimized thumbnail (300x300)
2. **filePath** - Original image (for image files only)
3. **FileText icon** - Fallback for non-image documents or missing images

Error Handling

- Images that fail to load automatically fall back to the FileText icon
- Prevents broken image displays
- Maintains consistent UI even with network issues

Files Modified

1. `client/src/pages/inbox.tsx` - Main inbox page display logic
2. `client/src/components/family/inbox-panel.tsx` - Inbox panel component
3. `shared/types/inbox.ts` - TypeScript type definitions
4. `shared/schema.ts` - Database schema
5. `server/routes.ts` - GET /api/inbox endpoint
6. `server/routes/ai-inbox.ts` - POST /api/inbox/register endpoint

Testing Recommendations

1. Upload a new image document - verify thumbnail appears
2. Upload a PDF document - verify FileText icon appears (PDFs don’t have thumbnails yet)
3. Check existing documents - verify they show appropriate previews
4. Test with slow network - verify fallback behavior works
5. Test with broken image URLs - verify FileText icon fallback

Future Enhancements

- Add PDF thumbnail generation using pdf-lib or similar
- Add thumbnail generation for other document types
- Implement lazy loading for thumbnails
- Add thumbnail caching strategy
- Consider adding thumbnail regeneration endpoint

Database Migration Note

If the database doesn’t have the `thumbnailPath` column yet, you’ll need to run a migration:

```
ALTER TABLE inbox_items ADD COLUMN thumbnail_path VARCHAR;
```

For existing documents without thumbnails, they will gracefully fall back to showing the FileText icon until they are re-uploaded or thumbnails are regenerated.