Bug Fix Verification Report

Bug Verification Report - Bug #1825: Uploaded Media Preview Fix

Date: 2026-01-09
Bug ID: #1825
Title: Uploaded file is not displayed after successful upload on Create Wish page
Status: Fixed

Issue Description

Users reported that after uploading a file (image/video) in the \`CreateWishWizard\`, the file was not displayed as a preview.

Root Causes Identified:

1. **Backend NameError**: The \`/api/upload-media\` endpoint had a coding error where \`filename\` was used without being defined, causing uploads to fail silently with a \`500 Server Error\`.
2. **Frontend Feedback Missing**: The wizard did not provide feedback via Toast notifications or clear loading states, making it difficult for users to know if an upload was successful.
3. **Limited Preview Support**: The preview component only handled image files, which would break if a video was uploaded.

Fix Implementation

1. Backend Service Recovery (\`endpoints.py\`)

- Fixed the \`NameError\` by correctly defining the unique filename logic.
- Implemented **Unique Filename Generation** using \`uuid\` to prevent filename collisions in Firebase Storage.
- Cleaned up the upload path structure to \`wishes/{user_id}/{uuid}.{ext}\`.

2. Frontend UX Enhancement (\`CreateWishWizard.js\`)

- **Toast Notifications**: Added \`showToast\` feedback for both successful uploads and specific error messages from the backend.
- **Robust Media Preview**: Updated the preview component to support both **Images** and **Videos** using regex detection on the URL extension.
- **Delete Functionality**: Added a "Remove Attachment" button (X) on the preview, allowing users to clear an upload before scheduling.
- **Premium UI**: Integrated the preview into the "Ready to Schedule?" review step with a glassmorphism aesthetic.

Bug Fix Verification Report

Verification Steps

Backend Verification

1. Executed manual upload test to `/api/upload-media`.
2. Verified that `uuid` is generated and file is saved successfully to Firebase.
3. Confirmed that the returned JSON contains the correct `public_url`.

Frontend Verification

1. **Positive Test (Image)**: Uploaded a `.png` file.

- **Expected**: Toast "File uploaded successfully!" appeared. Preview showed the image.

2. **Positive Test (Video)**: Uploaded an `.mp4` file.

- **Expected**: Preview correctly rendered a `<video>` element with controls.

3. **Error Handling**: Mocked a 500 error from the backend.

- **Expected**: Toast correctly notified the user of the failure.

Conclusion

The issue is resolved. The upload system is now reliable, provides immediate feedback, and handles multiple media types gracefully.