Node.js (Backend)

Now, let's create a simple Node.js backend to handle file uploads. You'll need to use the `express` framework and the `multer` middleware for handling file uploads. Make sure you have Node.js and npm (Node Package Manager) installed.

1. Create a new folder for your project, navigate to it in your terminal, and run `npm init` to create a `package.json` file. Follow the prompts to set up your project.
2. Install the necessary packages:

Code:

```
npm install express multer
```

3. Create a JavaScript file (e.g., `server.js`) for your Node.js backend:

Code:

```
javascript
const express = require('express');
const multer = require('multer');
const path = require('path');

const app = express();
const port = process.env.PORT || 3000;

// Configure multer for file uploads
const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        cb(null, 'uploads/'); // Uploads will be stored in the 'uploads'
directory
    },
    filename: (req, file, cb) => {
        const ext = path.extname(file.originalname);
        cb(null, Date.now() + ext); // Append a timestamp to the filename
    }
});

const upload = multer({ storage });

// Serve static files from the 'public' directory
app.use(express.static('public'));

// Handle file uploads
app.post('/upload', upload.fields([{ name: 'image', maxCount: 1 }, { name:
'video', maxCount: 1 }]), (req, res) => {
    // Handle file uploads here
    // You can access the uploaded files in req.files
    // e.g., req.files.image and req.files.video

    // Add your file processing logic here

    res.send('Files uploaded successfully.');
});

app.listen(port, () => {
    console.log(`Server is running on port ${port}`);
```

```
});
```

4. Create an `uploads` directory in your project folder to store the uploaded files.
5. Create a `public` directory and move your HTML file (the one with the form) into it.
6. Start your Node.js server by running:

```
node server.js
```

Your server should now be running. You can access the file upload form at `http://localhost:3000/your-html-file.html`. When the user selects an image and a video, the client-side validation will ensure that both files are selected. Upon submitting the form, the files will be uploaded to the `uploads` directory on the server.

Please note that this is a basic example, and you should add more error handling and security measures for a production application, such as validating file types and checking file sizes on the server-side and sanitizing file names to prevent security vulnerabilities.