How to upload an document to database from a web page

Uploading a document to a database from a web page involves several steps, typically involving both front-end and back-end development. Here's a general outline of how to accomplish this using common technologies such as HTML, JavaScript, and a back-end language like Python or Node.js, along with a database like MongoDB or MySQL.

❖ Front-End (HTML and JavaScript)

1. Create the HTML Form

```
<!DOCTYPE html>

<html>

<head>

  <title>Upload Document</title>

</head>

<body>

  <form id="uploadForm" enctype="multipart/form-data">

    <input type="file" id="fileInput" name="document" required />

    <button type="submit">Upload</button>

  </form>

  <script src="upload.js"></script>

</body>

</html>
```

```
);
```

2. JavaScript to Handle Form Submission

```
// upload.js
document.getElementById('uploadForm').addEventListener('submit', function(event) {
  event.preventDefault();

  let formData = new FormData();
  let fileInput = document.getElementById('fileInput');

  formData.append('document', fileInput.files[0]);

  fetch('/upload', {
    method: 'POST',
    body: formData
  })
  .then(response => response.json())
  .then(data => {
    if (data.success) {
      alert('File uploaded successfully');
    } else {
      alert('File upload failed');
    }
  })
  .catch(error => {
    console.error('Error:', error);
  });
});
```

- **Back-End (Node.js Example)**

1. Set Up Express Server

```
// server.js

const express = require('express');

const multer = require('multer');

const path = require('path');

const fs = require('fs');

const { MongoClient } = require('mongodb');


const app = express();

const upload = multer({ dest: 'uploads/' });


const uri = 'mongodb://localhost:27017';

const client = new MongoClient(uri);


app.post('/upload', upload.single('document'), async (req, res) => {

  if (!req.file) {

    return res.status(400).json({ success: false, message: 'No file uploaded' });

  }


  try {

    await client.connect();

    const database = client.db('mydatabase');

    const collection = database.collection('documents');


    const document = {

      filename: req.file.originalname,
            path: req.file.path,
            mimetype: req.file.mimetype,
            size: req.file.size,
            uploadDate: new Date()
        };

        await collection.insertOne(document);

        res.json({ success: true, message: 'File uploaded successfully' });
    } catch (error) {
        console.error(error);
        res.status(500).json({ success: false, message: 'Internal Server Error' });
    } finally {
        await client.close();
    }
});

app.listen(3000, () => {
    console.log('Server started on http://localhost:3000');
});
```

Database (MongoDB Example)

Ensure you have a running MongoDB instance. You can use a cloud service like MongoDB Atlas or run a local instance.

Shell:

```
mongod --dbpath /path/to/your/data/directory
```

Steps Explanation

1. **Front-End**:
    - The HTML form allows users to select a file.
    - JavaScript handles the form submission by preventing the default action, creating a `FormData` object, and sending it to the server via a POST request.
2. **Back-End**:
    - The Express server handles the `/upload` endpoint.
    - Multer middleware processes the file upload.
    - The uploaded file is temporarily stored on the server.
    - The server connects to MongoDB, inserts a document describing the file, and then returns a success or failure response.
3. **Database**:
    - MongoDB stores metadata about the uploaded file. You can also store the file itself in GridFS if needed, but in this example, the file is stored in the filesystem, and its metadata is stored in MongoDB.

This is a basic example, and in a production environment, you would need to add additional error handling, security measures, and possibly more complex file handling (e.g., virus scanning, file type validation).