

PEMROGRAMAN WEB LANJUT

“FILE UPLOAD”



Kelas : TI-2H

Disusun Oleh :

Fanesabhirawaning Sulistyono

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 65141

A. PERSIAPAN AWAL

1. Membuat beberapa file yang diperlukan, yang pertama controller :

php artisan make:controller FileUploadController

```
▼ TERMINAL
PS C:\laragon\www\PWL_POS> php artisan make:controller FileUploadController

INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\FileUploadController.php] created successfully.
```

2. Buat 2 buah method yaitu method fileUpload() dan prosesFileUpload. Method fileUpload() berisi kode program yang akan memanggil file view file-upload. blade.php. File blade inilah yang berisi kode HTML dan CSS untuk membuat form (akan kita buat sesaat lagi). Sedangkan method prosesFileUpload() untuk memproses hasil submit form. kita akan banyak membuat kode program, diantaranya validasi form dan serta proses pemindahan file yang sudah di upload. Sebagai argument terdapat **variabel \$request** yang akan berisi **Request object**.

```
FileUploadController.php U X
app > Http > Controllers > FileUploadController.php > FileUploadController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class FileUploadController extends Controller
8  {
9      public function fileUpload()
10     {
11         return view('file-upload');
12     }
13     public function prosesFileUpload(Request $request)
14     {
15         return "Pemrosesan file upload di sini";
16     }
17 }
```

3. Membuat route pada web.php

```
32 Route::get('/', function () {
33     return view('welcome');
34 });
35
36 Route::get('/file-upload', [FileUploadController::class, 'fileUpload']);
37 Route::post('/file-upload', [FileUploadController::class, 'prosesFileUpload']);
```

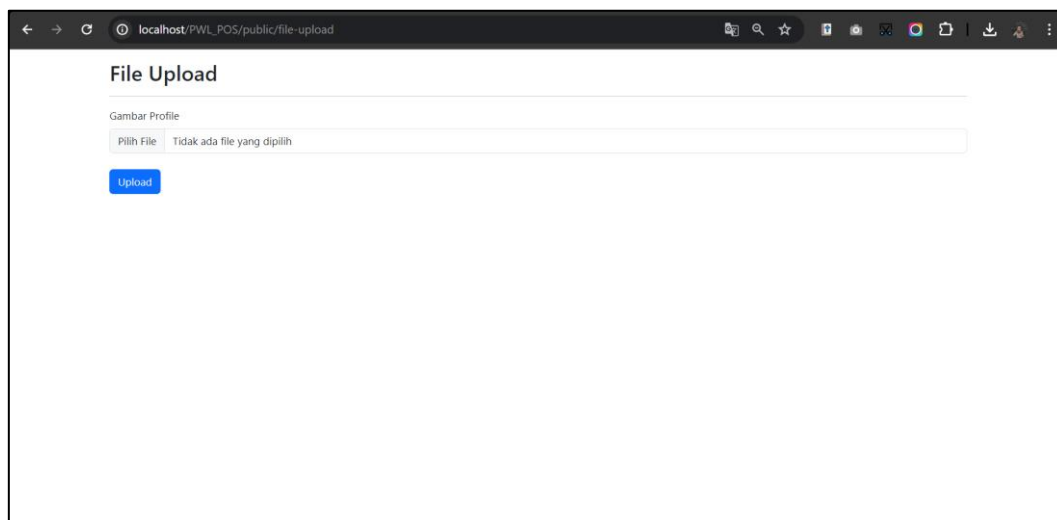
Route pertama dipakai untuk menampilkan form dengan cara mengakses method fileUpload() yang ada di FileUploadController. Alamat URLnya adalah '/file-upload'. Sedangkan route kedua berfungsi untuk pemrosesan form dengan mengakses method

prosesfileUpload() di FileUploadController. Route ini menggunakan method post karena menjadi tujuan saat form di submit.

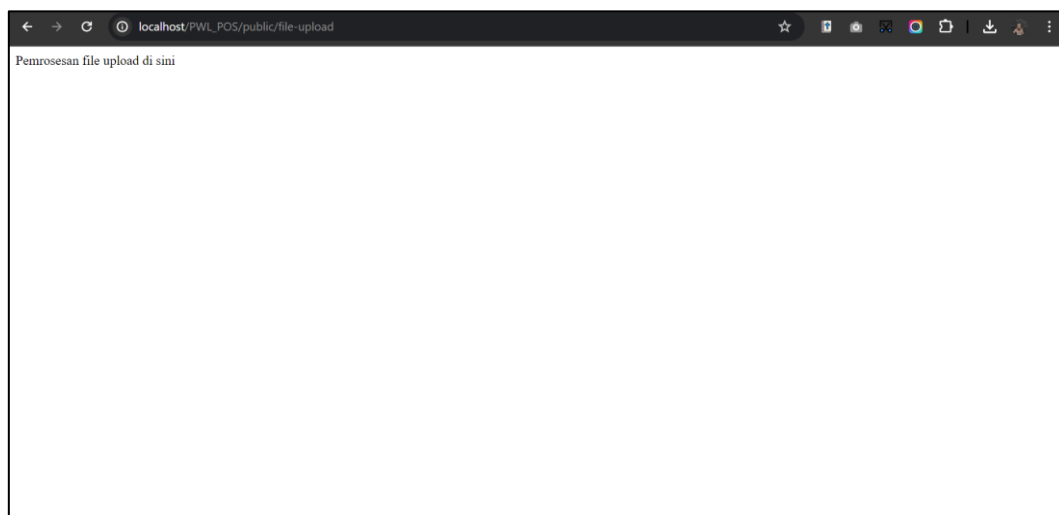
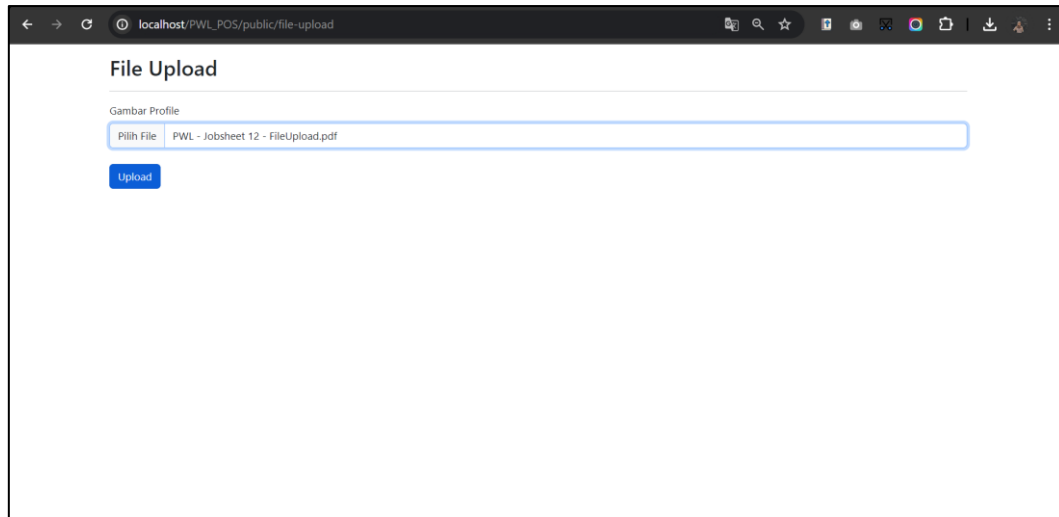
4. Membuat File View file-upload.blade.php dengan kode sebagai berikut :

```
resources > views > file-upload.blade.php > ...
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   {{-- <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
9       integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJ1SAwiGgFaw/dAiS63Xm" crossorigin="anonymous"> --}}
10  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
11      integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH" crossorigin="anonymous">
12  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
13      integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdS1K1eN7N6jIeHz" crossorigin="anonymous">
14  </script>
15  <title>File Upload</title>
16 </head>
17
18 <body>
19   <div class="container mt-3">
20     <h2>File Upload</h2>
21     <hr>
22     <form action="{{ url('/file-upload') }}" method="POST" enctype="multipart/form-data">
23       @csrf
24       <div class="mb-3">
25         <label for="berkas" class="form-label">Gambar Profile</label>
26         <input type="file" class="form-control" id="berkas" name="berkas">
27         @error('berkas')
28           <div class="text-danger">{{ $message }}</div>
29         @enderror
30       </div>
31       <button type="submit" class="btn btn-primary mt-2">Upload</button>
32     </form>
33   </div>
34 </body>
35
36 </html>
```

Gunakan CDN Bootstrap : <https://getbootstrap.com/> agar tampilan lebih menarik
Sehingga tampilan menjadi seperti berikut :



Silahkan upload sembarang file, lalu klik tombol "Upload":



B. INFORMASI FILE UPLOAD

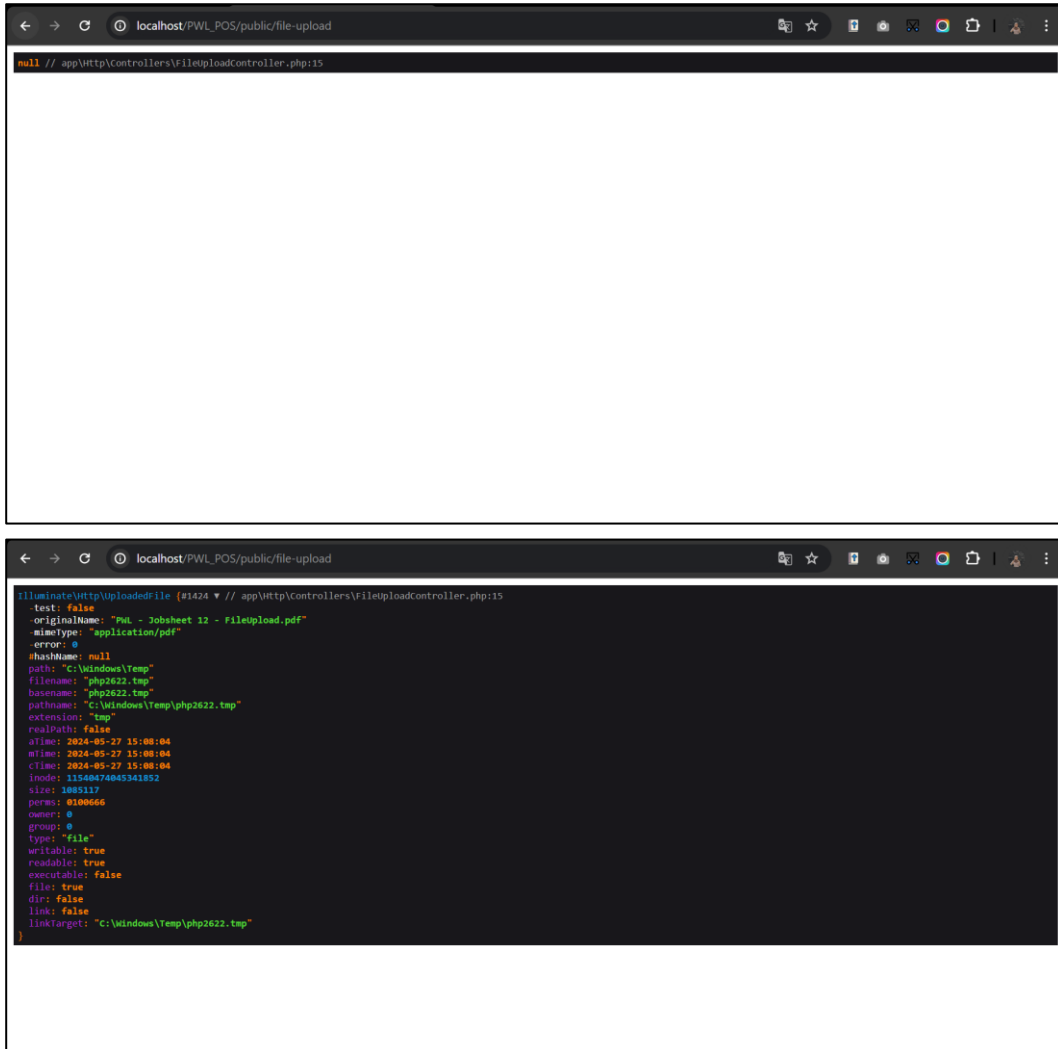
1. Informasi seputar file yang sudah di upload bisa kita akses melalui **Request object**, yakni variabel `$request`. Silahkan isi kode berikut ke dalam **method** `prosesFileUpload()` di **FileUploadController**

```
app > Http > Controllers > FileUploadController.php > FileUploadController > prosesFileUpload
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class FileUploadController extends Controller
8  {
9      public function fileUpload()
10     {
11         return view('file-upload');
12     }
13     public function prosesFileUpload(Request $request)
14     {
15         dump($request->berkas);
16         dump($request->file('file'));
17         // return "Pemrosesan file upload di sini";
18     }
19 }
```

Isi dari method ini hanya 1 baris, langsung **men-dump** `$request->berkas`. Nama **"berkas"** ini berasal dari nilai atribut name pada tag `<input type="file" name="berkas">`.

```
resources > views > file-upload.blade.php > ...
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      {{-- <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
9           integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous"> --}}
10     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
11           integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhJ6hW+ALEwIH" crossorigin="anonymous">
12     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
13           integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous">
14     </script>
15     <title>File Upload</title>
16 </head>
17
18 <body>
19     <div class="container mt-3">
20         <h2>File Upload</h2>
21         <hr>
22         <form action="{{ url('/file-upload') }}" method="POST" enctype="multipart/form-data">
23             @csrf
24             <div class="mb-3">
25                 <label for="berkas" class="form-label">Gambar Profile</label>
26                 <input type="file" class="form-control" id="berkas" name="berkas">
27                 @error('berkas')
28                 <div class="text-danger">{{ $message }}</div>
29                 @enderror
30             </div>
31             <button type="submit" class="btn btn-primary mt-2">Upload</button>
32         </form>
33     </div>
34 </body>
35
36 </html>
```

2. lakukan 2 percobaan, **pertama langsung submit form tanpa mengisi file apapun.** Dan **kedua, upload file sembarang dan submit.** Screen Shot bagaimana hasilnya?

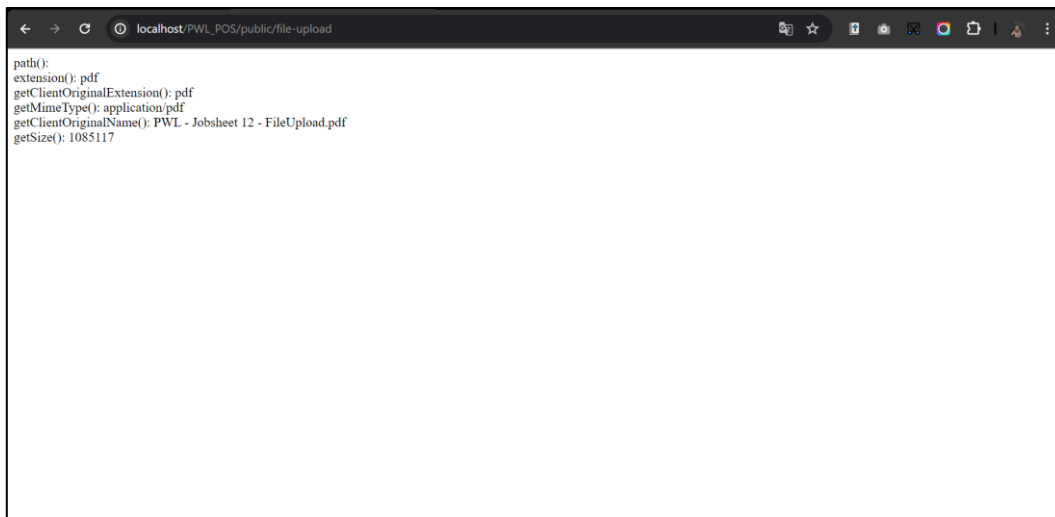


- Jika tidak ada file yang di upload, **perintah \$request->berkas** akan mengembalikan **nilai NULL**. Namun jika ada file yang di upload, akan menampilkan beragam informasi mengenai file tersebut. Diantaranya nama file, mimetype, tanggal upload serta ukuran file yang di upload. Untuk memeriksa apakah terdapat sebuah file yang di upload, bisa menggunakan perintah **\$request->hasFile('berkas')**. Hasilnya **true** jika ada file yang di upload dan **false** jika tidak ada file yang di upload.

3. Berikut cara mengambil beberapa info dari file yang di upload:
(app/Http/Controllers/FileUploadController.php)

```
FileUploadController.php M X web.php kategori.blade.php file-upload.blade.php
app > Http > Controllers > FileUploadController.php > FileUploadController > prosesFileUpload
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class FileUploadController extends Controller
8 {
9     public function fileUpload()
10     {
11         return view('file-upload');
12     }
13     public function prosesFileUpload(Request $request)
14     {
15         // dump($request->berkas);
16         // dump($request->file('file'));
17         // return "Pemrosesan file upload di sini";
18
19         if ($request->hasFile('berkas')) {
20             echo "path(): " . $request->berkas->path();
21             echo "<br>";
22             echo "extension(): " . $request->berkas->extension();
23             echo "<br>";
24             echo "getClientOriginalExtension(): " . $request->berkas->getClientOriginalExtension();
25             echo "<br>";
26             echo "getMimeType(): " . $request->berkas->getMimeType();
27             echo "<br>";
28             echo "getClientOriginalName(): " . $request->berkas->getClientOriginalName();
29             echo "<br>";
30             echo "getSize(): " . $request->berkas->getSize();
31         } else {
32             echo "Tidak ada berkas yang diupload";
33         }
34     }
35 }
```

4. Lakukan percobaan upload file sehingga akan muncul keterangan sbb:



```
path(): pdf
extension(): pdf
getClientOriginalExtension(): pdf
getMimeType(): application/pdf
getClientOriginalName(): PWL - Jobsheet 12 - FileUpload.pdf
getSize(): 1085117
```

Penjelasan :

Di **baris 18** pada **method prosesFileUpload** terdapat sebuah kondisi if yang akan dijalankan jika `$request->hasFile('berkas')` bernilai true. Di dalam blok ini, untuk mengakses beberapa method. File sample yang diupload adalah sebuah file pdf . Berikut

penjelasan dari method **prosesFileUpload** yang ada di baris 20 - 32:

- **\$request->berkas->path()**, menampilkan alamat path dari file yang sudah di upload. Perhatikan bahwa isinya adalah alamat temporary dari pengaturan PHP.
- **\$request->berkas->extension()**, menampilkan extension file. Dalam contoh ini file yang diupload file pdf, sehingga extensionnya adalah pdf.
- **\$request->berkas->getClientOriginalExtension()**, menampilkan extension yang diambil dari nama file. Karena file yang di upload adalah pdf, maka hasil method ini adalah pdf.
- **\$request->berkas->getMimeType()**, menampilkan mimetype dari file yang di upload. Dalam contoh ini hasilnya pdf.
- **\$request->berkas->getClientOriginalName()**, menampilkan nama asli dari file yang diupload.
- **\$request->berkas->getSize()**, menampilkan ukuran file yang di upload (dalam satuan byte).

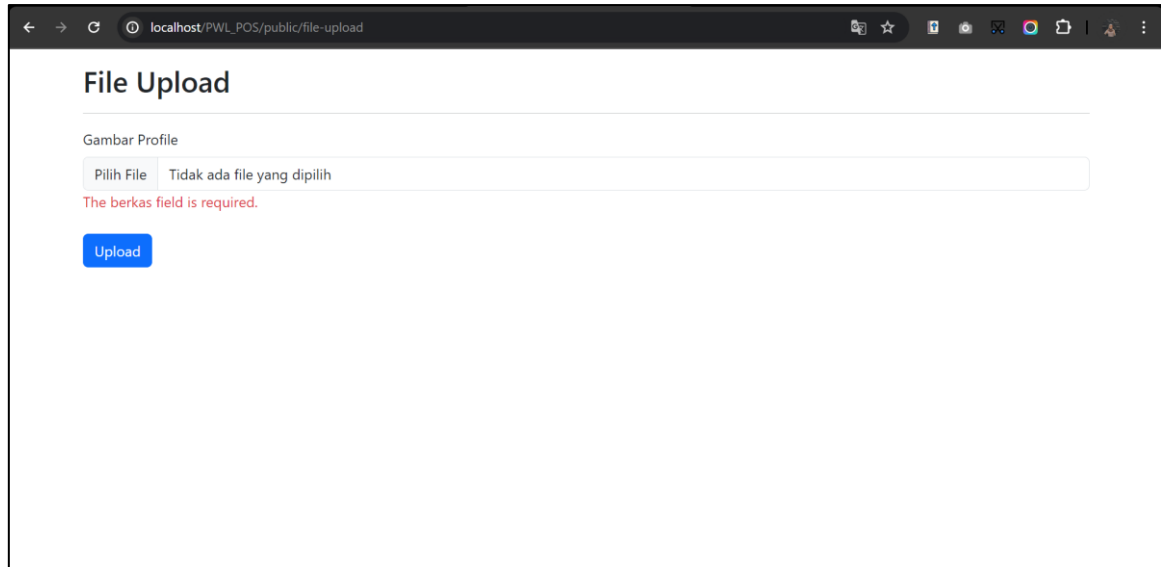
C. VALIDASI FILE UPLOAD

File yang di upload tentunya butuh proses validasi. Malah ini menjadi lebih penting karena tentu kita tidak ingin ada yang mengupload file berbahaya, atau malah mengupload file .php yang berisi kode/script tertentu.

1. Membuat validasi file upload mirip seperti cara membuat validasi inputan form biasa, yakni menggunakan method `$request->validate()`. Berikut contoh penggunaannya: (**app/Http/Controllers/FileUploadController.php**)

```
35     $request->validate([
36         'berkas' => 'required',
37     ]);
38     echo $request->berkas->getClientOriginalName() . "lolos validasi";
```

Terdapat syarat required untuk file berkas, sehingga akan tampil pesan error jika form di submit tanpa meng-upload sebuah file:



Jika syarat validasi tidak terpenuhi, akan langsung di redirect untuk menampilkan pesan error. Di halaman view file-upload.blade.php kita sudah menyiapkan **perintah @error('berkas')** untuk menampilkan pesan error ini.

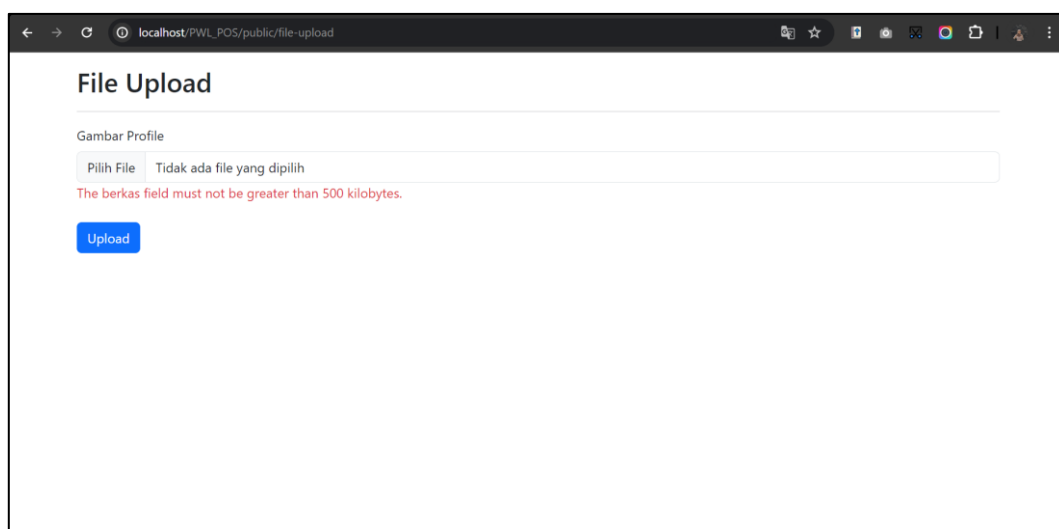
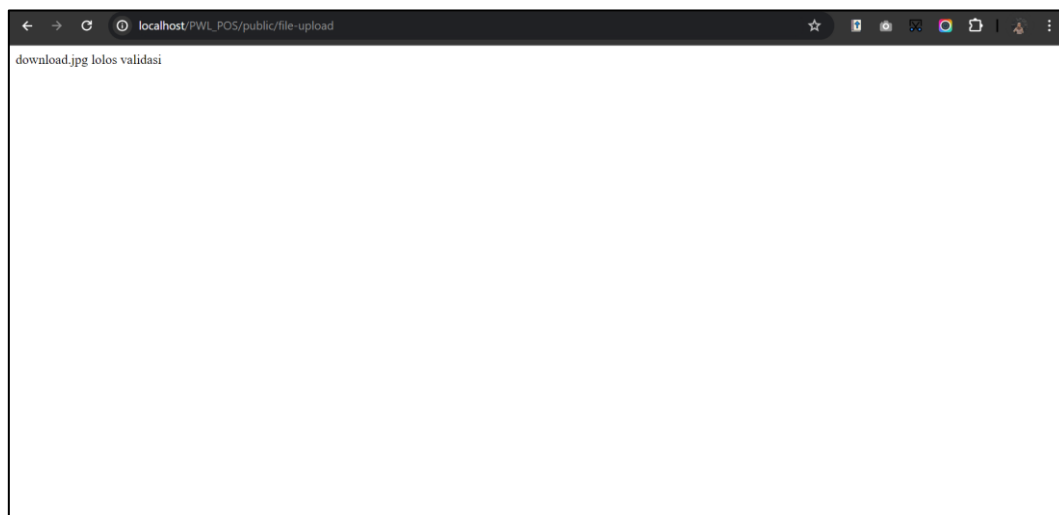
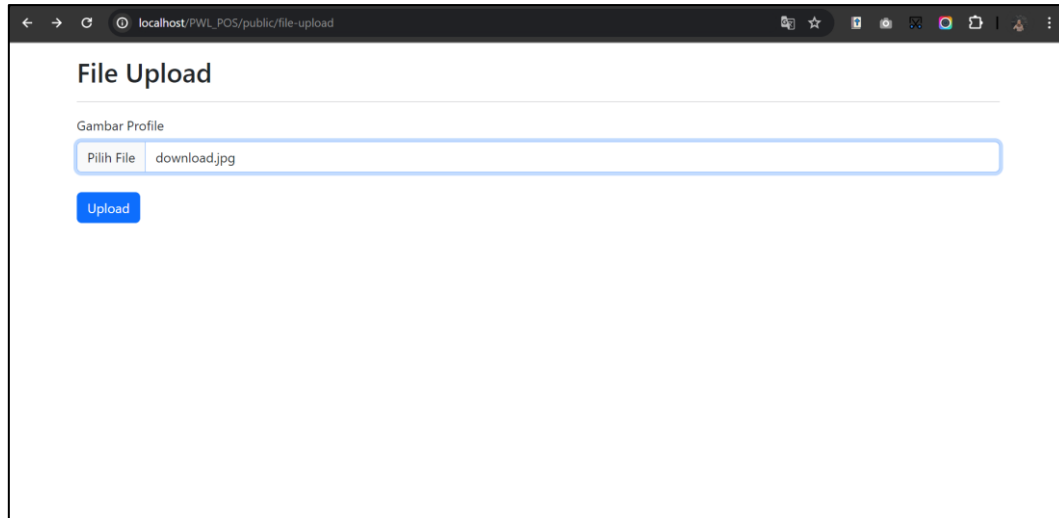
2. Syarat validasi selanjutnya adalah membatasi jenis file dan maksimal ukuran file : (tambahkan code pada line 20)

```
35     $request->validate([
36         'berkas' => 'required|file|image|max:5000',
37     ]);
38
39     echo $request->berkas->getClientOriginalName() . " lolos validasi";
```

Berikut ini penjelasannya :

Berikut penjelasan dari syarat validasi ini:

- required: inputan form tidak boleh kosong.
- file: memastikan bahwa file sudah berhasil di upload.
- image: file yang di upload harus file image (gambar), salah satu dari jpeg, png, bmp, gif, svg, atau webp.
- max:5000, artinya file yang di upload berukuran maksimal 5000 byte atau sekitar 5 MB.



D. MEMINDAH FILE UPLOAD

Semua file yang di upload akan **tersimpan sementara ke folder temporary** yang dalam PHP bawaan XAMPP berada di C:\xampp\tmp\. Agar bisa diakses, file ini harus di pindah ke folder aplikasi kita. Laravel menyediakan beragam cara untuk memindahkan file ini, diantaranya method `store()`, `storeAs()`, dan `move()`. Kita akan bahas secara bertahap.

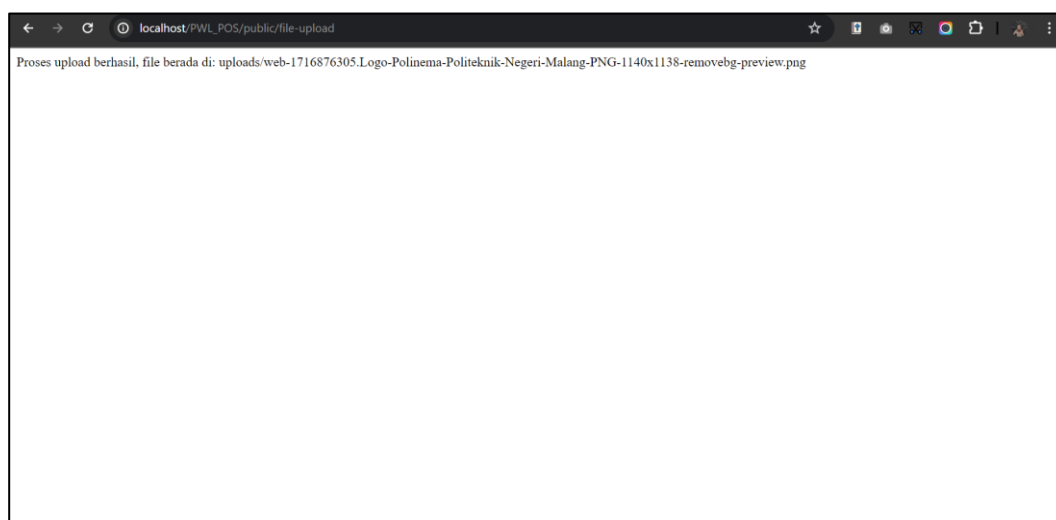
1. Cara pertama adalah menggunakan method `store()` yang bisa diakses dari Request object.

Berikut contoh penggunaannya:

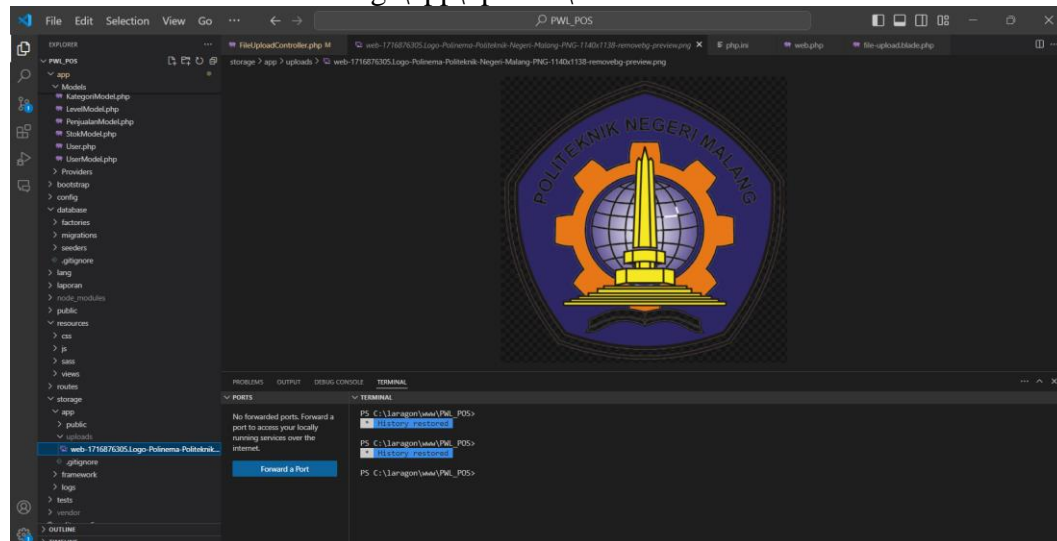
```
35     $request->validate([
36         'berkas' => 'required|file|image|max:5000',]);
37     $path = $request->berkas->store('uploads');
38     echo "Proses upload berhasil, file berada di: $path";
39     // echo $request->berkas->getClientOriginalName() . " lolos validasi";
```

Di baris 17-20 terdapat kode untuk proses validasi yang sudah di bahas sebelumnya. Proses pemindahan file dilakukan di baris **19 dengan perintah `$request->berkas->store('uploads')`**. Method `store()` akan mengambil file upload dari folder temporary dan memindahkannya ke folder `storage\app\` di aplikasi Laravel. **Method `store()`** ini butuh sebuah argument berupa nama folder, sehingga perintah `$request->berkas->store('uploads')` akan memindahkan file upload ke **folder `storage\app\uploads`**. Nilai kembalian dari method ini berupa alamat path dari file akhir, yang dalam contoh ini disimpan ke dalam **variabel `$path`**.

Mari kita coba, silahkan upload sebuah file gambar dengan ukuran kurang dari 1MB:



Setelah itu buka folder storage\app\uploads\ untuk melihat file ini:



file yang di upload sudah bisa diakses. Namun kenapa nama file acak seperti itu?

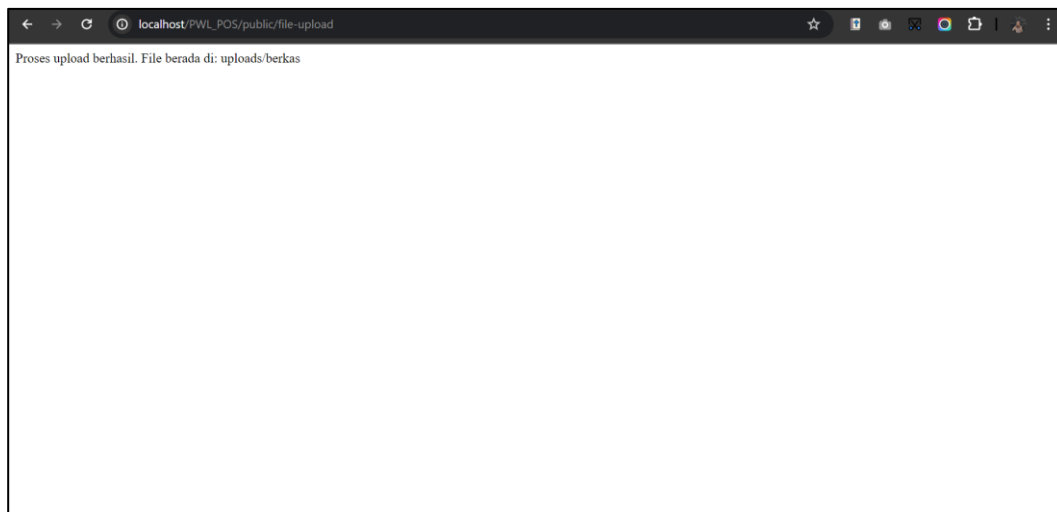
Method store() memang akan **men-generate nama acak untuk setiap file yang di upload**. Kesannya memang agak aneh, tapi sebenarnya sangat bermanfaat:

- Menghindari kemungkinan file ditimpa. Jika nama file yang sudah di upload sama dengan file asal, bisa saja di kemudian hari ada yang mengupload file dengan nama yang sama. Jika ini terjadi, maka file yang baru akan menimpa file lama.
- Menghindari error karena nama file mengandung spasi. Di dalam penulisan alamat path, karakter spasi ini sering menjadi masalah.

Bagaimana jika kita tetap ingin membuat nama file sendiri? Tidak masalah, Laravel menyediakan method storeAs() untuk keperluan ini. Berikut contoh penggunaannya: (line 20)

```
35 $request->validate([
36     'berkas' => 'required|file|image|max:5000',]);
37
38 // $path = $request->berkas->store('uploads');
39 $path = $request->berkas->storeAs('uploads','berkas' );
40 echo "Proses upload berhasil. File berada di: $path";
41
42 //echo $request->berkas->getClientOriginalName() . " lolos validasi";
43
44
```

Method storeAs() butuh 2 argument. Argument pertama berupa nama folder, dan argument kedua berupa nama file yang ingin di buat. Jika kita meng-upload file dengan perintah di atas, nama file akhir adalah "berkas", dan tanpa extension!

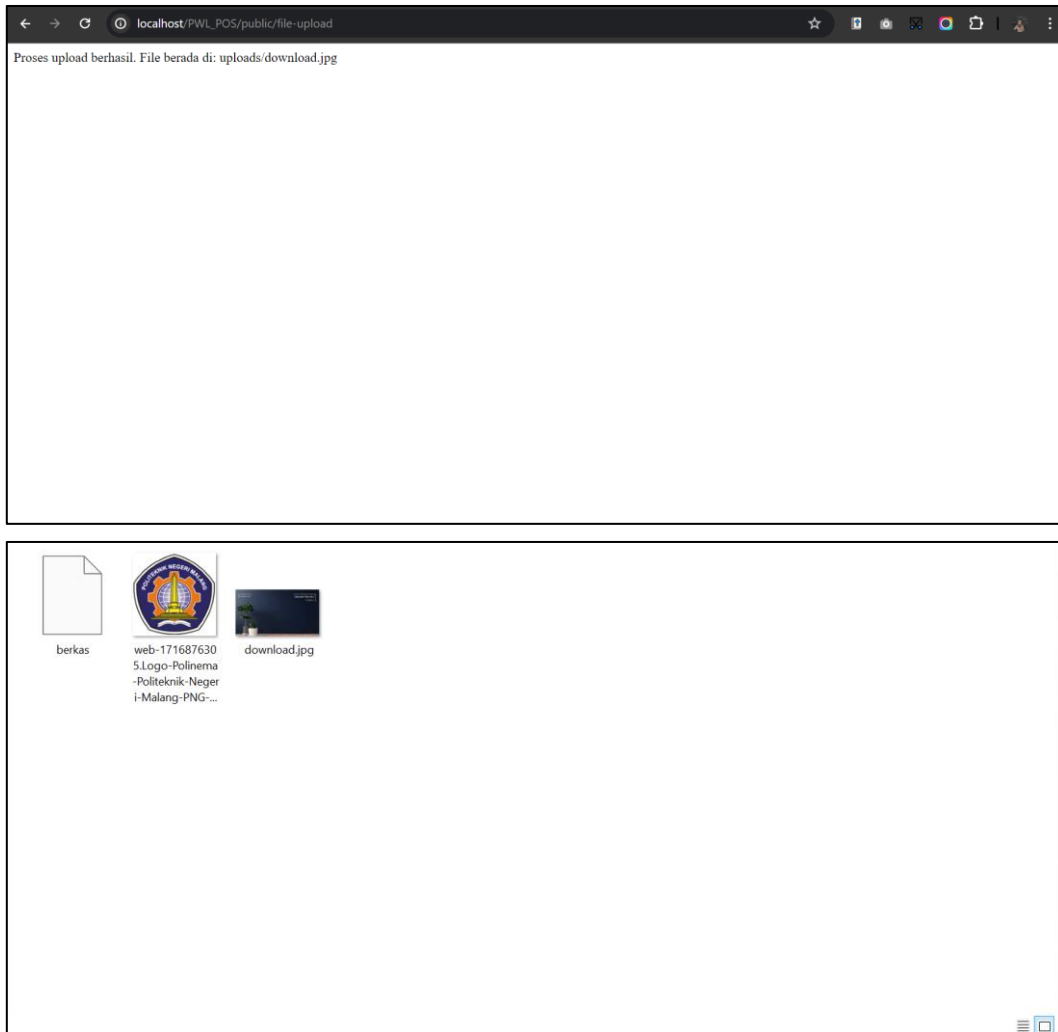


Sehingga kita harus menambah sendiri extension file ini. Selain itu, nama file tidak bisa di tulis langsung seperti ini karena akan terus tertimpa oleh file lain karena sama-sama bernama "berkas".

2. mengambil nama file dari fungsi **getClientOriginalName()** seperti contoh berikut: (app/Http/Controllers/FileUploadController.php)

```
35     $request->validate([
36         'berkas' => 'required|file|image|max:5000',]);
37
38     $namaFile = $request->berkas->getClientOriginalName();
39     // $path = $request->berkas->store('uploads');
40     $path = $request->berkas->storeAs('uploads', $namaFile );
41     echo "Proses upload berhasil. File berada di: $path";
42
43     //echo $request->berkas->getClientOriginalName() . " lolos validasi";
44
45 }
```

Di baris 19 mengambil nama file asal dengan perintah `$request->berkas->getClientOriginalName()`, yang hasilnya ditampung oleh variabel `$namaFile`. Variabel `$namaFile` ini kemudian diinput sebagai argument kedua dari method `storeAs()`. Dengan cara ini maka file yang di upload akan memiliki nama yang sama dengan file asal:



Namun terdapat kelemahan dengan teknik seperti ini. Jika ada yang meng-upload file dengan nama yang sama, maka **file pertama akan tertimpa**. Selain itu bisa timbul masalah jika nama file yang di upload mengandung karakter spasi.

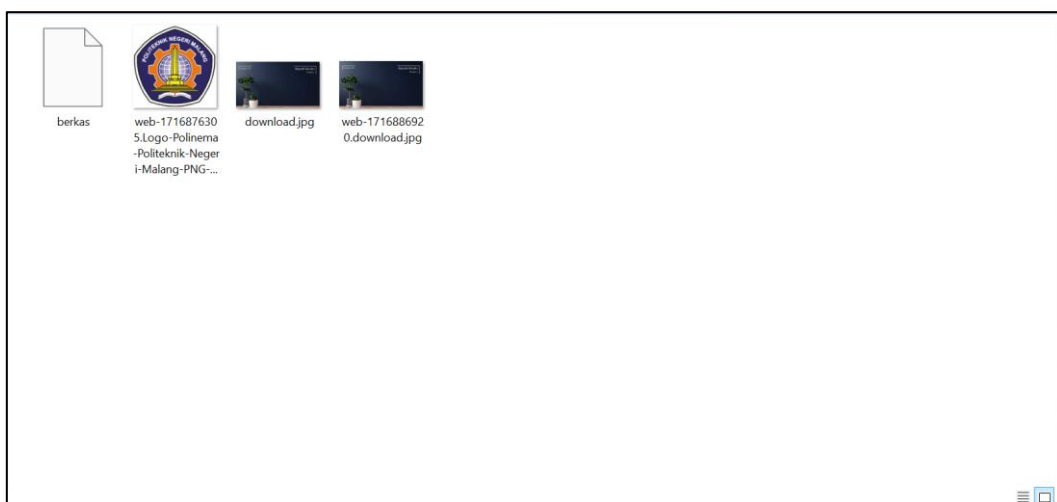
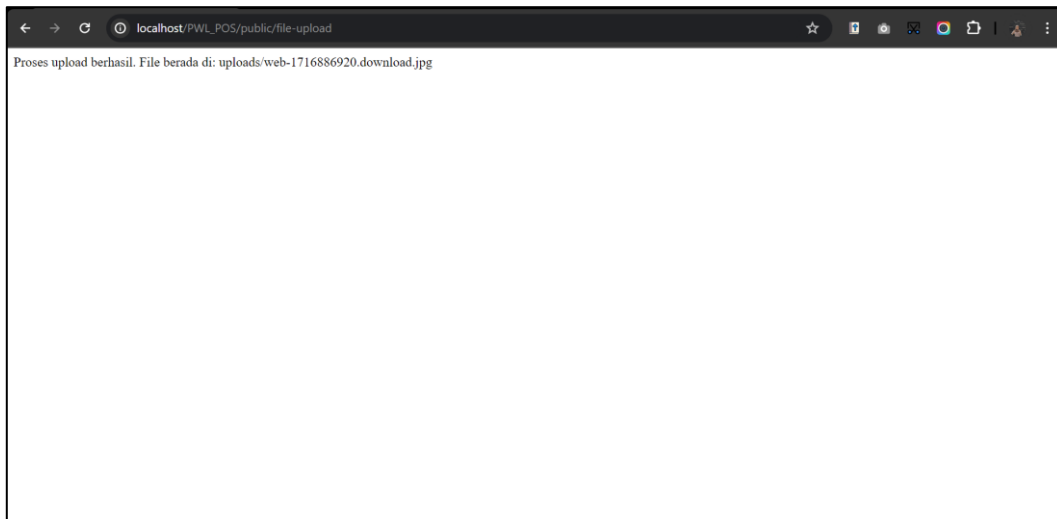
3. Cara lain adalah dengan meng-generate nama acak sendiri. Sebagai contoh, membuat nama file upload berdasarkan **nama user**, ditambah dengan **digit acak dari fungsi time()**. Berikut kode programnya: (**app/Http/Controllers/FileUploadController.php**)

```

35     $request->validate([
36         'berkas' => 'required|file|image|max:5000',]);
37     $extfile = $request->berkas->getClientOriginalName();
38     $namaFile = 'web-' . time() . "." . $extfile;
39     // $path = $request->berkas->store('uploads');
40     $path = $request->berkas->storeAs('uploads', $namaFile );
41     echo "Proses upload berhasil. File berada di: $path";
42
43     //echo $request->berkas->getClientOriginalName() . " lolos validasi";
44
45 }

```

Di baris19, method `$request->berkas->getClientOriginalExtension()` di pakai untuk mengambil extension file asal. Kemudian di baris 20 menyambung 3 string, yakni web-' yang diibaratkan sebagai nama user, hasil timestamp dari fungsi `time()` bawaan PHP, serta extension file yang berasal dari variabel `$extFile`.



Hasilnya, file yang di upload bernama `web-1574953613.jpg`. Nama user "web" ini nantinya bisa berasal dari database, yang akan berbeda-beda sesuai dengan id login. Nama ini relatif tidak bermasalah, kecuali user tersebut meng-upload beberapa file

dalam detik yang sama. Agar lebih aman lagi (supaya tidak ada nama file yang bentrok), bisa ditambah dengan karakter acak lain seperti hasil fungsi hash md5(). Atau daripada repot, bisa pakai method store() saja supaya Laravel yang langsung menggenerate nama file secara otomatis.

E. MEMBUAT SYMLINK

File yang kita upload sudah bisa diakses, tapi baru secara manual menggunakan Windows Explorer. File tersebut belum bisa dibuka dari halaman web karena secara default (dan memang sudah seharusnya), Laravel hanya mengizinkan web browser mengakses file yang ada di folder public saja. Sebenarnya bisa saja file upload langsung disimpan ke folder public, tapi Laravel memilih solusi yang lebih elegan memakai metode yang disebut sebagai symbolic link, atau sering disingkat sebagai symlink. Symlink mirip seperti shortcut tapi seolah-olah mengcopy isi satu folder ke folder lain (mirroring). Pada saat ini, semua file yang sudah di upload berada di **folder storage\app**. Kita bisa membuat symlink agar file ini juga bisa diakses dari **folder public**.

1. Membuat symlink yang menghubungkan folder storage\app dengan folder public\

php artisan storage:link

```
PS C:\laragon\www\PWL_POS> php artisan storage : link
Laravel Framework 10.47.0

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display help for the given command. When no command is given display help for the list command
  -q, --quiet                Do not output any message
  -V, --version              Display this application version
  --ansi|--no-ansi           Force (or disable --no-ansi) ANSI output
  -n, --no-interaction       Do not ask any interactive question
  --env[=ENV]                The environment the command should run under
  -v|vv|vvv, --verbose       Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands for the "storage" namespace:
  storage:link               Create the symbolic links configured for the application
  storage:unlink              Delete existing symbolic links configured for the application
PS C:\laragon\www\PWL_POS>
```

Dengan perintah ini, akan tampil sebuah symlink bernama storage di folder public, silahkan buka folder ini:

Name	Date modified	Type	Size
adminlte	3/29/2024 9:44 PM	File folder	
storage	5/15/2024 3:31 PM	File folder	
vendor	3/20/2024 1:35 PM	File folder	
.htaccess	2/12/2024 6:23 PM	HTACCESS File	1 KB
favicon.ico	2/12/2024 6:23 PM	ICO File	0 KB
index.php	2/12/2024 6:23 PM	PHP Source File	2 KB
robots.txt	2/12/2024 6:23 PM	Text Source File	1 KB

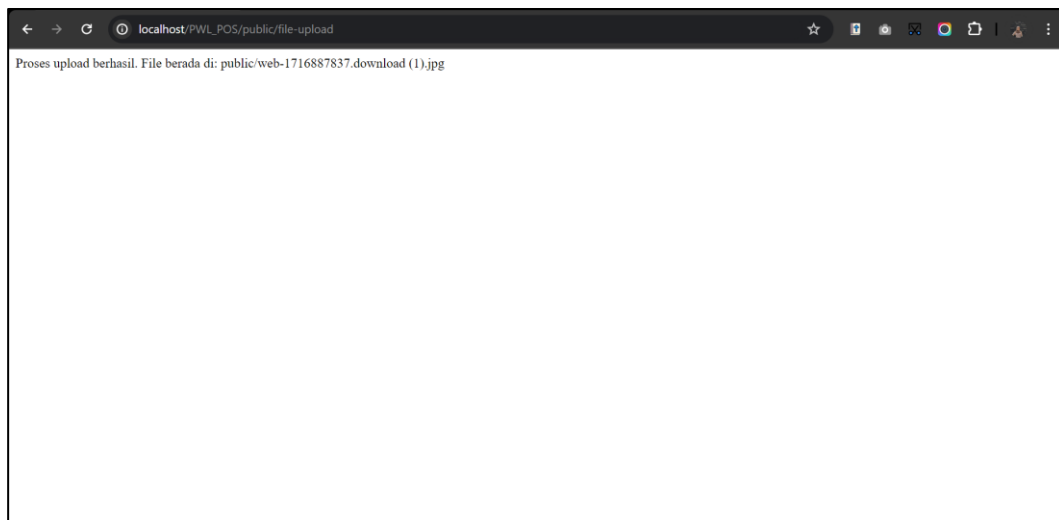
Ketika di klik, terdapat file .gitignore di dalam symlink storage. Ini hanya file bantu yang berfungsi sebagai tanda agar isi folder storage tidak perlu di backup oleh aplikasi Git. Symlink storage pada dasarnya merupakan shortcut atau mirror dari folder **storage\app\public**.

Dengan kata lain, semua file yang akan kita simpan **di storage\app\public**, juga akan ada di **folder public\storage**. Untuk uji coba, silahkan copy sebuah file ke **folder storage\app\public**, maka file tersebut juga ada di **folder public\storage**. Dan ketika file itu di hapus, maka di folder lain akan ikut terhapus. Dan ini berlaku dua arah, jika file di **folder public\storage** di tambah atau di hapus, file yang ada **di storage\app\public** jika akan mengikuti.

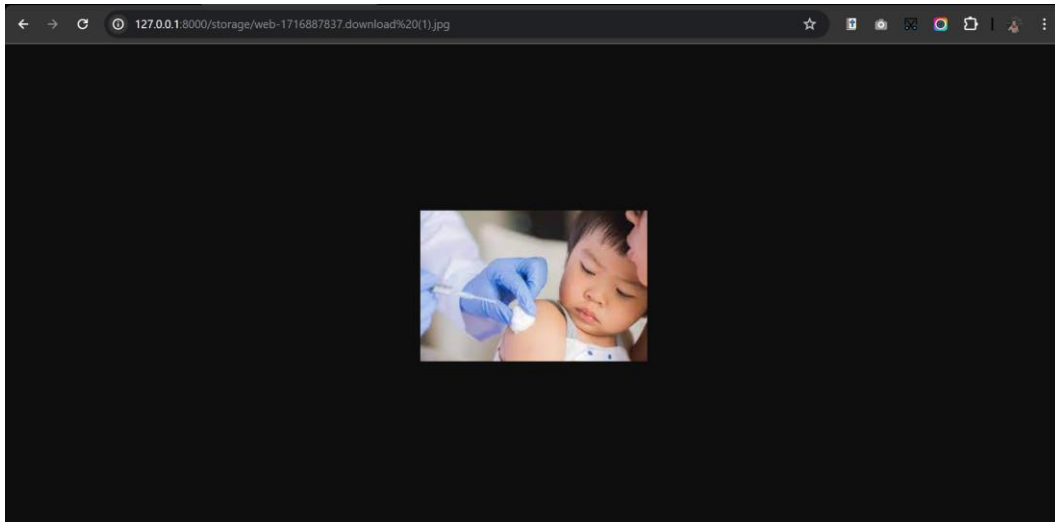
2. Modifikasi dari method prosesFileUpload():

```
35     $request->validate([
36         'berkas' => 'required|file|image|max:5000',]);
37     $extfile = $request->berkas->getClientOriginalName();
38     $namaFile = 'web-' . time() . "." . $extfile;
39     // $path = $request->berkas->store('uploads');
40     $path = $request->berkas->storeAs('public', $namaFile );
41     echo "Proses upload berhasil. File berada di: $path";
42
43     //echo $request->berkas->getClientOriginalName() . " lolos validasi";
44
45 }
```

Perhatikan bahwa argumen pertama dari method storeAs() adalah 'public'(line 21). Inilah folder tempat file upload akan disimpan. Silahkan upload sebuah file, maka akan tampil hasil berikut:

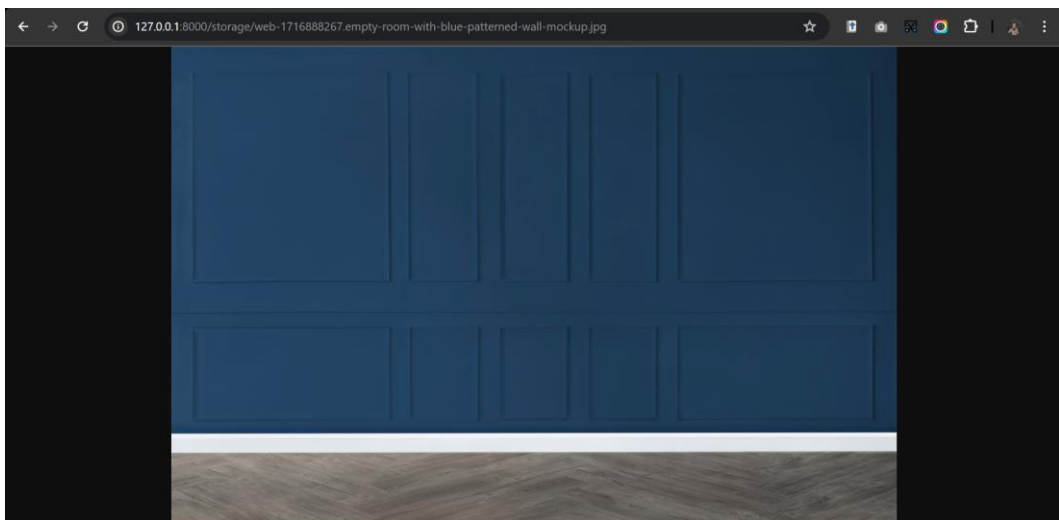
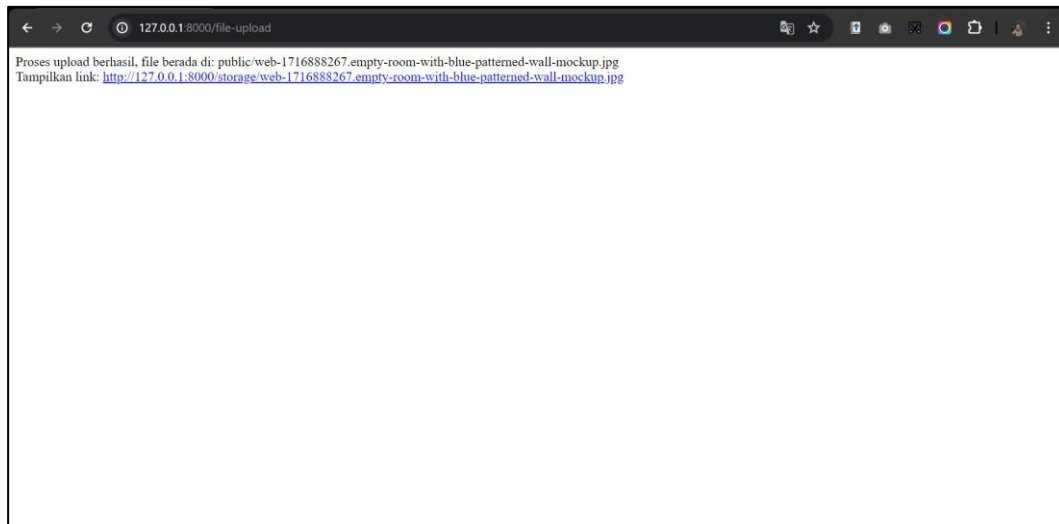
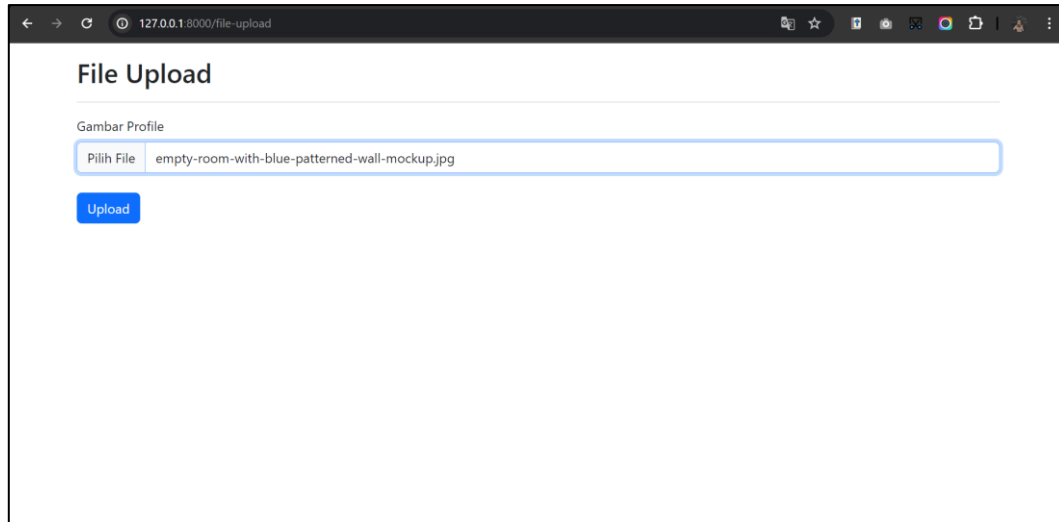


Proses upload berhasil, dan file tersebut ada di `storage\app\public\ web-1714801950.images.png`. Nama file yang akan anda dapati pastinya berbeda karena di sini meng-generate nama file berdasarkan fungsi `time()`. Karena file upload sudah berada di dalam folder symlink, maka bisa diakses dari web browser. Silahkan buka alamat berikut (sesuaikan nama filenya): [http://localhost:8000/storage/ web-1714801950.images.png](http://localhost:8000/storage/web-1714801950.images.png)



3. Menambahkan link agar gambar dapat diakses, tambahkan kode pada method `prosesFileUpload`:

```
35     $request->validate([
36         'berkas' => 'required|file|image|max:5000',]);
37     $extfile = $request->berkas->getClientOriginalName();
38     $namaFile = 'web-' . time() . "." . $extfile;
39     // $path = $request->berkas->store('uploads');
40     $path = $request->berkas->storeAs('public', $namaFile );
41
42     $pathBaru = asset('storage/' . $namaFile);
43     echo "Proses upload berhasil, file berada di: $path";
44     echo "<br>";
45     echo "Tampilkan link: <a href='$pathBaru'$>$pathBaru</a>";
46
47     //echo $request->berkas->getClientOriginalName() . " lolos validasi";
48
49 }
```

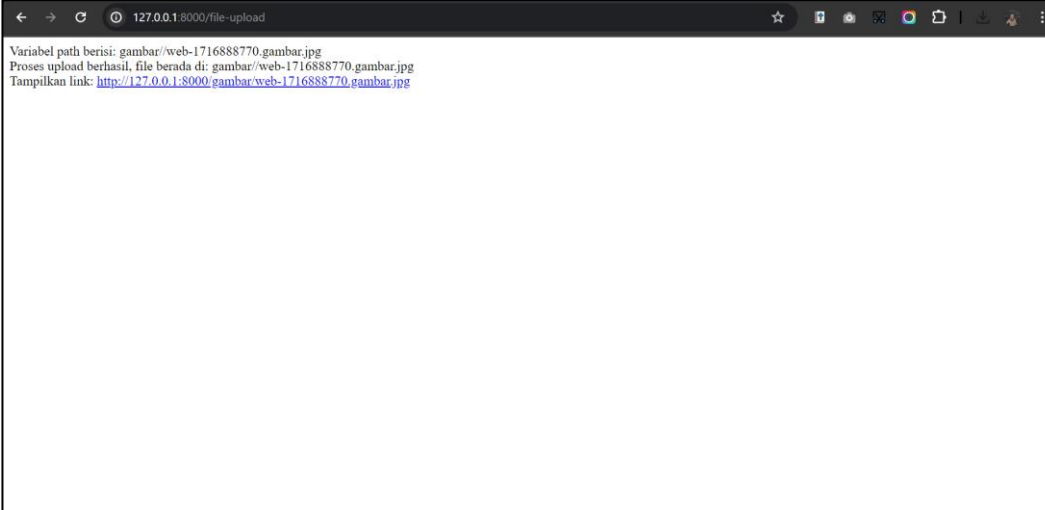


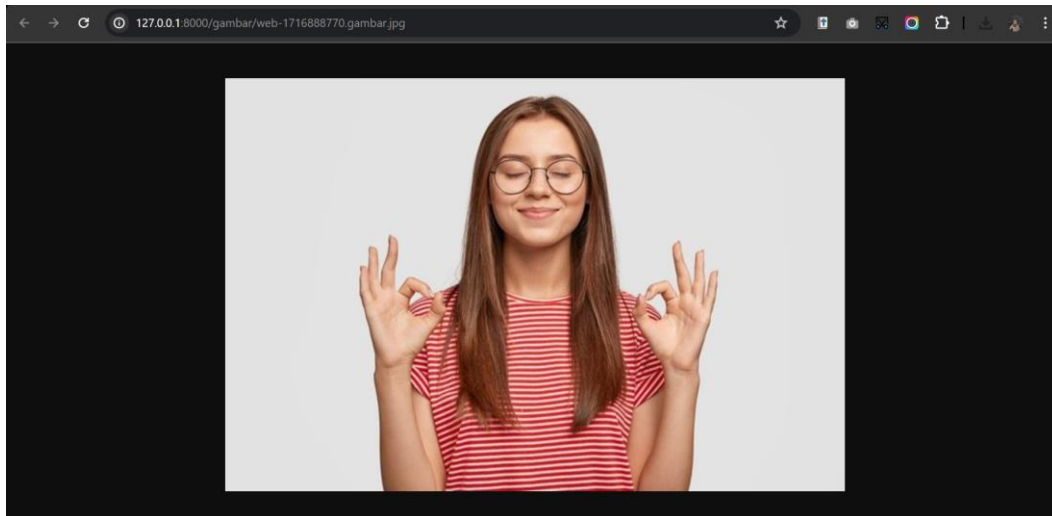
F. METHOD MOVE

Jika karena sesuatu hal kita tidak bisa (atau tidak ingin) menggunakan symlink, Laravel juga menyediakan cara agar file yang di upload langsung tersimpan ke folder public, tidak harus lewat folder storage. Caranya, gunakan method move():

1. Modifikasi controller FileUploadController.php

```
35     $request->validate([
36         'berkas' => 'required|file|image|max:5000',]);
37     $extfile = $request->berkas->getClientOriginalName();
38     $namaFile = 'web-' . time() . "." . $extfile;
39     // $path = $request->berkas->store('uploads');
40     // $path = $request->berkas->storeAs('public', $namaFile );
41
42     $path = $request->berkas->move('gambar', $namaFile);
43     $path = str_replace("\\", "/", $path);
44     echo "Variabel path berisi: $path <br>";
45
46     $pathBaru = asset('gambar/' . $namaFile);
47     echo "Proses upload berhasil, file berada di: $path";
48     echo "<br>";
49     echo "Tampilkan link: <a href='$pathBaru'>$pathBaru</a>";
50
51     //echo $request->berkas->getClientOriginalName() . " lolos validasi";
52
53 }
```





Di baris 22 menggunakan perintah `$request->berkas->move('image',$namaFile)` untuk memindahkan file upload. Sama seperti method `store()` dan `storeAs()`, **method `move()` butuh 2 argument. Argument pertama diisi dengan nama folder penyimpanan, dan argument kedua berupa nama file.**

Hasilnya, di folder public akan muncul folder image yang berisi file <http://127.0.0.1:8000/gambar/web-1714803545.images.png>. Karena sudah berada di dalam folder public, maka file ini bisa langsung di akses dari web browser. Tambahan fungsi `str_replace()` di baris 23 bertujuan untuk mengganti tanda pemisah folder dari forward slash `"\"` menjadi back slash `"/"` untuk variabel `$path`. Misalnya dari `gambar\web1643167529.jpg` menjadi `gambar/web1643167529.jpg`. Tanda pemisah folder ini sebenarnya tergantung jenis OS yang dipakai. Di OS Windows, alamat `http://localhost:8000/gambar\web-1643167529.jpg` tetap bisa diakses. Namun akan lebih rapi jika semua pemisah folder menggunakan karakter back slash `"/"`.

Tugas:

1. buat form file upload dimana kita bisa menentukan sendiri nama file akhir. Nama file ini diambil dari inputan text box yang juga ada di dalam form tersebut. Setelah di submit, langsung tampilkan file tersebut di web browser menggunakan tag .

Jawab:

```
PS C:\laragon\www\PWL_POS> php artisan make:controller FileUploadRenameController

INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\FileUploadRenameController.php] created successfully.

app > Http > Controllers > FileUploadRenameController.php > FileUploadRenameController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class FileUploadRenameController extends Controller
8  {
9      public function fileUpload()
10     {
11         return view('file-upload-rename');
12     }
13     public function prosesFileUpload(Request $request)
14     {
15         $request->validate([
16             'nama_berkas' => 'required|alpha_dash',
17             'berkas' => 'required|file|image|max:500',
18         ]);
19
20         // Penamaan file
21         // $namaFile = $request->nama_berkas . "." . $request->berkas->getClientOriginalExtension();
22         $namaFile = $request->nama_berkas . "." . $request->berkas->extension();
23
24         // Path file
25         // $path = $request->berkas->storeAs('public/gambar', $namaFile);
26         $path = $request->berkas->move('gambar', $namaFile);
27         $path = str_replace("\\", "/", $path);
28
29         // $pathBaru = asset('storage/gambar' . $namaFile);
30         $pathBaru = asset('gambar/' . $namaFile);
31         echo "Gambar berhasil diupload ke <a href='$pathBaru'$>$namaFile</a>";
32         echo "<br><br>";
33         echo "<img src='$pathBaru' style='width: 50vw; height: auto'>";
34     }
35 }
```

```

resources > views > file-upload-rename.blade.php > html > body > div.container.mt-3 > form > div.mb-3 > div.text-danger
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
9          integrity="sha384-QMtkZyPPEjISvSwWaRU90F0eRpk6YctnYmDr5pNlyT2BrjXh0JMHjY6Hw+AEwIH" crossorigin="anonymous">
10     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
11         integrity="sha384-YvpcrYf0tY3lHB60NNKmXc5s9FDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous">
12     </script>
13     <title>File Upload</title>
14 </head>
15
16 <body>
17     <div class="container mt-3">
18         <h2>File Upload</h2>
19         <hr>
20         <form action="{{ url('/file-upload-rename') }}" method="POST" enctype="multipart/form-data">
21             @csrf
22             <div class="mb-3">
23                 <label for="nama_berkas" class="form-label">Nama Gambar</label>
24                 <input type="text" class="form-control" id="nama_berkas" name="nama_berkas"
25                     placeholder="Nama Gambar">
26                 @error('nama_berkas')
27                     <div class="text-danger">{{ $message }}</div>
28                 @enderror
29             </div>
30             <div class="mb-3">
31                 <label for="berkas" class="form-label">Gambar Profile</label>
32                 <input type="file" class="form-control" id="berkas" name="berkas">
33                 @error('berkas')
34                     <div class="text-danger">{{ $message }}</div>
35                 @enderror
36             </div>
37             <button type="submit" class="btn btn-primary mt-2">Upload</button>
38         </form>
39     </div>
40 </body>
41
42 </html>

```

```

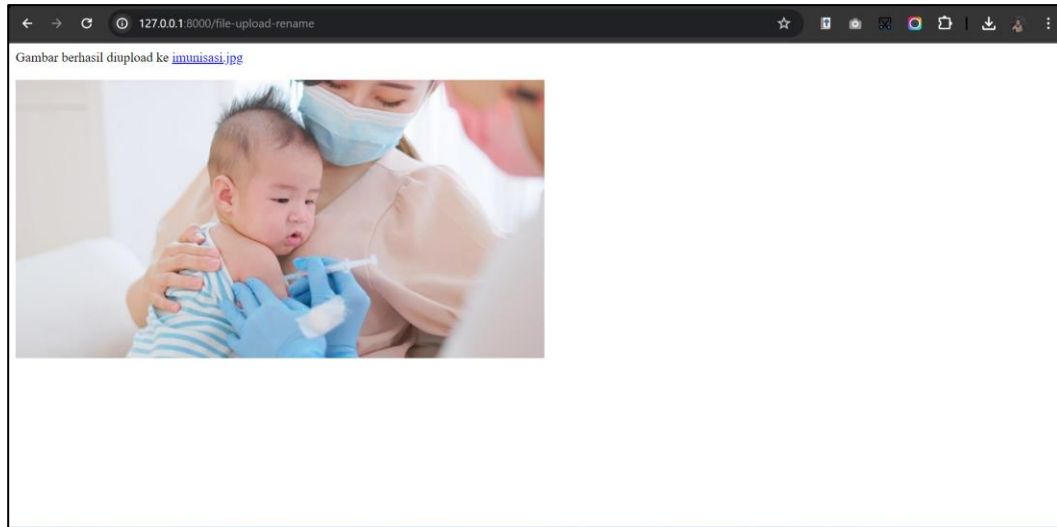
40 // TUGAS JS 12 (NO 1) -- FileUpload
41 Route::get('/file-upload-rename', [FileUploadRenameController::class, 'fileUpload']);
42 Route::post('/file-upload-rename', [FileUploadRenameController::class, 'prosesFileUpload']);

```

The screenshot shows a web browser at the address 127.0.0.1:8000/file-upload-rename. The page title is "File Upload". The form contains two main sections:

- Nama Gambar:** A text input field with the value "imunisasi".
- Gambar Profile:** A file input field with a "Pilih File" button and the selected file name "imunisasi-bayi-manfaat-jadwal-dan-efek-sampingnya.jpg".

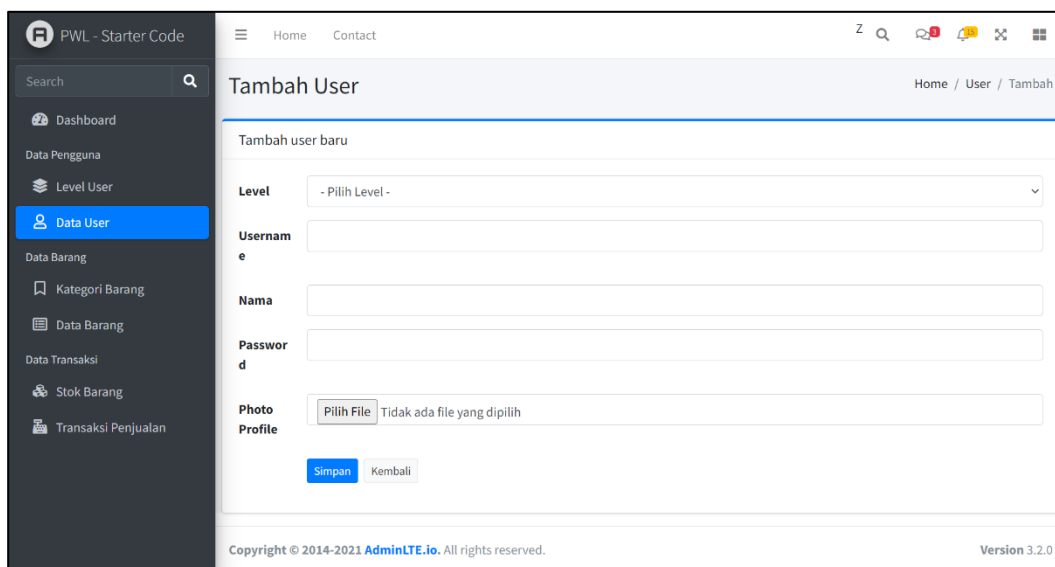
At the bottom of the form is a blue "Upload" button.



2. Tambahkan proses upload image dan application pada starter code

Jawab:

- **Menu User**



- **Menu Barang**

The screenshot shows the 'Tambah Barang' (Add Item) form in the AdminLTE application. The form is located in the main content area, titled 'Tambah barang baru'. It contains several input fields for adding a new item:

- Kategori:** A dropdown menu with the option '- Pilih Kategori -'.
- Kode Barang:** A text input field.
- Nama Barang:** A text input field.
- Harga Beli:** A text input field.
- Harga Jual:** A text input field.
- Foto Barang:** A file upload area with a 'Pilih File' button and the text 'Tidak ada file yang dipilih'.
- User:** A dropdown menu with the option '- Pilih User -'.
- Stok Jumlah:** A text input field.

At the bottom of the form, there are two buttons: 'Simpan' (Save) and 'Kembali' (Back). The footer of the page shows the copyright notice 'Copyright © 2014-2021 AdminLTE.io. All rights reserved.' and the version 'Version 3.2.0'.

The screenshot shows the 'Edit Barang' (Edit Item) form in the AdminLTE application. The form is located in the main content area, titled 'Edit barang'. It contains several input fields for editing an existing item:

- Kategori:** A dropdown menu with the option 'Elektronik'.
- Kode Barang:** A text input field with the value 'BRG001'.
- Nama Barang:** A text input field with the value 'Laptop'.
- Harga Beli:** A text input field with the value '40000'.
- Harga Jual:** A text input field with the value '7000000'.
- Foto Barang:** A file upload area with a 'Pilih File' button and the text 'xioami-mi-notebook-15-pro-foto-xiaomi-54.jpg'. Below the input field, there is a note: 'Abaikan (jangan diisi) jika tidak ingin mengganti foto barang.'

At the bottom of the form, there are two buttons: 'Simpan' (Save) and 'Kembali' (Back). The footer of the page shows the copyright notice 'Copyright © 2014-2021 AdminLTE.io. All rights reserved.' and the version 'Version 3.2.0'.

PWL - Starter Code

Search

Dashboard

Data Pengguna

Level User

Data User

Data Barang

Kategori Barang

Data Barang

Data Transaksi

Stok Barang

Transaksi Penjualan

Home

Contact


z

Q

Detail Barang

Home / Barang / Detail

Detail barang

ID	1
Kategori	Elektronik
Kode Barang	BRG001
Nama Barang	Laptop
Harga Beli	40000
Harga Jual	7000000
Photo Barang	

Kembali

Copyright © 2014-2021 AdminLTE.io. All rights reserved.

Version 3.2.0

*** Sekian, dan Terimakasih***

Jobsheet 12 - PWL 2023/2024

Hal. 27 / 27