Laravel E-Sign Flow — One Document Drop (All Files)

Copy this single document into your project. Each section is labeled with its **file path**. Create any missing directories. After pasting, run the setup commands at the end.

```
// ==========================================
// file: composer.json (add required packages)
// ==========================================
{
  "require": {
    "php": "^8.2",
    "laravel/framework": "^11.0|^12.0",
    "setasign/fpdi": "^2.6",
    "setasign/fpdf": "^1.8",
    "symfony/process": "^7.0"
  }
}
```

```
// ================================================
// file: database/migrations/2025_09_18_000001_create_envelopes_table.php
// ================================================
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
  public function up(): void {
    Schema::create('envelopes', function (Blueprint $t) {
      $t->id();
      $t->string('title')->nullable();
      $t->string('original_filename');
      $t->string('pdf_path'); // always points to working PDF (docx converted
here)
      $t->enum('status',
['draft','sent','in_progress','completed','voided'])->default('draft');
      $t->foreignId('created_by')->constrained('users');
      $t->timestamp('completed_at')->nullable();
      $t->timestamps();
    });
  }
  public function down(): void { Schema::dropIfExists('envelopes'); }
};
```

```
// ================================================
// file: database/migrations/2025_09_18_000002_create_recipients_table.php
// ================================================
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
  public function up(): void {
    Schema::create('recipients', function (Blueprint $t) {
      $t->id();
      $t->foreignId('envelope_id')->constrained()->cascadeOnDelete();
      $t->string('name');
      $t->string('email');
      $t->unsignedInteger('signing_order')->default(1);
      $t->enum('status', ['pending','emailed','viewed','signed','declined'])-
>default('pending');
      $t->uuid('token')->unique();
      $t->timestamp('token_expires_at')->nullable();
      $t->timestamp('signed_at')->nullable();
      $t->json('meta')->nullable();
      $t->timestamps();
    });
  }
  public function down(): void { Schema::dropIfExists('recipients'); }
};
```

```
// ================================================
// file: database/migrations/2025_09_18_000003_create_fields_table.php
// ================================================
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
  public function up(): void {
    Schema::create('fields', function (Blueprint $t) {
      $t->id();
      $t->foreignId('envelope_id')->constrained()->cascadeOnDelete();
      $t->foreignId('recipient_id')->nullable()->constrained()-
>nullOnDelete();
      $t->enum('type', ['signature','initial','date','text','checkbox']);
      $t->unsignedInteger('page');   // 1-based page number
      $t->decimal('x', 10, 2);       // stored in PDF user units
      $t->decimal('y', 10, 2);
      $t->decimal('w', 10, 2);
      $t->decimal('h', 10, 2);
```

```
      $t->boolean('required')->default(true);
      $t->string('label')->nullable();
      $t->json('options')->nullable(); // font size, placeholder, etc.
      $t->timestamps();
    });
  }
  public function down(): void { Schema::dropIfExists('fields'); }
};
```

```
// ====================================================
// file: database/migrations/
2025_09_18_000004_create_envelope_files_table.php
// ====================================================
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
  public function up(): void {
    Schema::create('envelope_files', function (Blueprint $t) {
      $t->id();
      $t->foreignId('envelope_id')->constrained()->cascadeOnDelete();
      $t->enum('kind', ['original','intermediate','final']);
      $t->string('path');
      $t->timestamps();
    });
  }
  public function down(): void { Schema::dropIfExists('envelope_files'); }
};
```

```
// ====================================================
// file: database/migrations/
2025_09_18_000005_create_envelope_events_table.php
// ====================================================
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
  public function up(): void {
    Schema::create('envelope_events', function (Blueprint $t) {
      $t->id();
      $t->foreignId('envelope_id')->constrained()->cascadeOnDelete();
      $t->foreignId('recipient_id')->nullable()->constrained()-
>nullOnDelete();
      $t->string('event'); // created, recipients_added, fields_placed,
```

```
sent_to_signers, link_opened, signed, completed...
      $t->string('ip')->nullable();
      $t->text('user_agent')->nullable();
      $t->json('data')->nullable();
      $t->timestamps();
    });
  }
  public function down(): void { Schema::dropIfExists('envelope_events'); }
};
```

```
// ================================
// file: app/Models/Envelope.php
// ================================
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Envelope extends Model
{
  protected $fillable = [

'title','original_filename','pdf_path','status','created_by','completed_at'
  ];

  public function recipients(){ return $this->hasMany(Recipient::class); }
  public function fields(){ return $this->hasMany(Field::class); }
  public function files(){ return $this->hasMany(EnvelopeFile::class); }
  public function events(){ return $this->hasMany(EnvelopeEvent::class); }
}
```

```
// ================================
// file: app/Models/Recipient.php
// ================================
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Recipient extends Model
{
  protected $fillable = [

'envelope_id','name','email','signing_order','status','token','token_expires_at','signed_at','
  ];
  protected $casts =
['meta'=>'array','token_expires_at'=>'datetime','signed_at'=>'datetime'];
```

```
  public function envelope(){ return $this->belongsTo(Envelope::class); }
  public function fields(){ return $this->hasMany(Field::class); }
}
```

```
// ================================
// file: app/Models/Field.php
// ================================
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Field extends Model
{
  protected $fillable = [

'envelope_id','recipient_id','type','page','x','y','w','h','required','label','options'
  ];
  protected $casts = ['options'=>'array'];
  public function envelope(){ return $this->belongsTo(Envelope::class); }
  public function recipient(){ return $this->belongsTo(Recipient::class); }
}
```

```
// ====================================
// file: app/Models/EnvelopeFile.php
// ====================================
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class EnvelopeFile extends Model
{
  protected $fillable = ['envelope_id','kind','path'];
  public function envelope(){ return $this->belongsTo(Envelope::class); }
}
```

```
// ====================================
// file: app/Models/EnvelopeEvent.php
// ====================================
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class EnvelopeEvent extends Model
```

```
{
  protected $fillable =
['envelope_id','recipient_id','event','ip','user_agent','data'];
  protected $casts = ['data'=>'array'];
  public function envelope(){ return $this->belongsTo(Envelope::class); }
  public function recipient(){ return $this->belongsTo(Recipient::class); }
}
```

```
// ========================================
// file: app/Services/DocxToPdfService.php
// ========================================
// Converts .docx to .pdf using LibreOffice (headless)
namespace App\Services;

use Symfony\Component\Process\Process;
use Illuminate\Support\Facades\Storage;

class DocxToPdfService
{
  public function convertPublicDisk(string $docxRelPath): string
  {
    $docxAbs = Storage::disk('public')->path($docxRelPath);
    $outDir  = dirname($docxAbs);
    // soffice --headless --convert-to pdf --outdir <dir> <file.docx>
    $process = new Process(['soffice','--headless','--convert-to','pdf','--
outdir',$outDir,$docxAbs]);
    $process->setTimeout(120);
    $process->mustRun();

    $pdfAbs  = preg_replace('/\.docx?$/i','.pdf',$docxAbs);
    $pdfRel  = str_replace(Storage::disk('public')->path(''), '', $pdfAbs);
    return $pdfRel;
  }
}
```

```
// ====================================
// file: app/Services/PdfSignerService.php
// ====================================
namespace App\Services;

use setasign\Fpdi\Fpdi;
use Illuminate\Support\Facades\Storage;
use App\Models\{Envelope, Recipient, Field, EnvelopeFile};

class PdfSignerService
{
```

```
  /** Stamp PNG signatures at given x/y/w/h for fields of the provided
recipient */
  public function stamp(Envelope $envelope, Recipient $recipient, array
$sigImages): void
  {
    $src = Storage::disk('public')->path($envelope->pdf_path);
    $pdf = new Fpdi();
    $pageCount = $pdf->setSourceFile($src);

    $fields = $envelope->fields()->where('recipient_id',$recipient->id)-
>get()->keyBy('id');

    for ($pageNo = 1; $pageNo <= $pageCount; $pageNo++) {
      $tpl = $pdf->importPage($pageNo);
      $size = $pdf->getTemplateSize($tpl);
      $pdf->AddPage($size['orientation'], [$size['width'], $size['height']]);
      $pdf->useTemplate($tpl, 0, 0, $size['width'], $size['height']);

      foreach ($sigImages as $fieldId => $relPath) {
        $f = $fields[$fieldId] ?? null;
        if (!$f || (int)$f->page !== $pageNo) continue;
        $imgAbs = Storage::disk('public')->path($relPath);
        $pdf->Image($imgAbs, (float)$f->x, (float)$f->y, (float)$f->w,
(float)$f->h, 'PNG');
      }
    }

    $newPath = "envelopes/intermediate/{$envelope->id}_".time().".pdf";
    $pdf->Output(Storage::disk('public')->path($newPath), 'F');

    EnvelopeFile::create(['envelope_id'=>$envelope-
>id,'kind'=>'intermediate','path'=>$newPath]);
    $envelope->update(['pdf_path'=>$newPath]);
  }
}
```

```
// ================================================
// file: app/Http/Controllers/Admin/EnvelopeController.php
// ================================================
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;
use App\Models\{Envelope, EnvelopeEvent, EnvelopeFile, Recipient};
use App\Services\DocxToPdfService;
```

```
class EnvelopeController extends Controller
{
  public function index() {
    $rows = Envelope::withCount(['recipients','fields'])->latest()-
>paginate(20);
    return view('admin.envelopes.index', compact('rows'));
  }

  public function create() {
    return view('admin.envelopes.create');
  }

  public function store(Request $r, DocxToPdfService $converter) {
    $r->validate([
      'title' => ['nullable','string','max:255'],
      'file'  => ['required','file','mimes:pdf,doc,docx','max:256000'],
    ]);
    $uploaded = $r->file('file')->store('envelopes/original','public');
    $originalName = $r->file('file')->getClientOriginalName();

    // Convert doc/docx to PDF immediately so designer always previews PDF
    $pdfPath = $uploaded;
    if (preg_match('/\.(docx?|DOCX?)$/',$originalName)) {
      $pdfPath = $converter->convertPublicDisk($uploaded);
    }

    $env = Envelope::create([
      'title' => $r->title ?: pathinfo($originalName, PATHINFO_FILENAME),
      'original_filename' => $originalName,
      'pdf_path' => $pdfPath,
      'created_by' => auth()->id(),
    ]);
    EnvelopeFile::create(['envelope_id'=>$env-
>id,'kind'=>'original','path'=>$uploaded]);
    EnvelopeEvent::create(['envelope_id'=>$env->id,'event'=>'created']);

    return redirect()->route('admin.envelopes.designer', $env);
  }

  public function show(Envelope $envelope) {
    $envelope->load('recipients','fields','events','files');
    return view('admin.envelopes.show', compact('envelope'));
  }

  public function addRecipients(Request $r, Envelope $envelope) {
    $r->validate([
      'recipients' => ['required','array','min:1'],
      'recipients.*.name' => ['required','string'],
      'recipients.*.email'=> ['required','email'],
      'recipients.*.signing_order' => ['required','integer','min:1']
    ]);
```

```
    foreach ($r->recipients as $rec) {
      $envelope->recipients()->create([
        'name' => $rec['name'],
        'email'=> $rec['email'],
        'signing_order' => (int)$rec['signing_order'],
        'token' => (string) Str::uuid(),
        'token_expires_at' => now()->addDays(7),
      ]);
    }
    EnvelopeEvent::create(['envelope_id'=>$envelope-
>id,'event'=>'recipients_added']);
    return back()->with('success','Recipients added.');
  }

  public function send(Envelope $envelope) {
    if ($envelope->fields()->count() === 0) {
      return back()->with('error','Add fields first.');
    }
    $envelope->update(['status'=>'sent']);
    $first = $envelope->recipients()->min('signing_order');
    $batch = $envelope->recipients()->where('signing_order',$first)->get();
    foreach ($batch as $rec) {
      // Simple email via notification — replace with your Mailable if
desired
      \Mail::raw("Please sign: ".route('sign.show',$rec->token), function($m)
use($rec,$envelope){
        $m->to($rec->email,$rec->name)->subject('Signature Request: '.
$envelope->title);
      });
      $rec->update(['status'=>'emailed']);
    }
    EnvelopeEvent::create(['envelope_id'=>$envelope-
>id,'event'=>'sent_to_signers']);
    return redirect()->route('admin.envelopes.show',$envelope)-
>with('success','Sent to first signer(s).');
  }

  public function void(Envelope $envelope) {
    $envelope->update(['status'=>'voided']);
    EnvelopeEvent::create(['envelope_id'=>$envelope->id,'event'=>'voided']);
    return back()->with('success','Envelope voided.');
  }
}
```

```
// ================================================
// file: app/Http/Controllers/Admin/DesignerController.php
// ================================================
```

```
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\{Envelope, Field, EnvelopeEvent};

class DesignerController extends Controller
{
  public function edit(Envelope $envelope) {
    $envelope->load('recipients','fields');
    return view('admin.envelopes.designer', compact('envelope'));
  }

  public function saveFields(Request $r, Envelope $envelope) {
    $r->validate(['fields'=>'array']);
    // Replace existing fields with new set
    $envelope->fields()->delete();

    foreach ($r->fields ?? [] as $f) {
      $envelope->fields()->create([
        'recipient_id' => $f['recipient_id'] ?? null,
        'type'  => $f['type'],
        'page'  => (int)$f['page'],
        'x'     => (float)$f['x'],
        'y'     => (float)$f['y'],
        'w'     => (float)$f['w'],
        'h'     => (float)$f['h'],
        'required' => (bool)($f['required'] ?? true),
        'label' => $f['label'] ?? null,
        'options' => $f['options'] ?? null,
      ]);
    }
    EnvelopeEvent::create(['envelope_id'=>$envelope-
>id,'event'=>'fields_placed']);
    return back()->with('success','Fields saved.');
  }
}
```

```
// =======================================
// file: app/Http/Controllers/SigningController.php
// =======================================
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use App\Models\{Recipient, Envelope, EnvelopeEvent};
use App\Services\PdfSignerService;
```

```
class SigningController extends Controller
{
  public function show(string $token) {
    $recipient = Recipient::where('token',$token)->firstOrFail();
    if ($recipient->envelope->status === 'voided') abort(403,'This envelope
was voided.');
    if (optional($recipient->token_expires_at)->isPast()) abort(403,'Link
expired.');
    $env = $recipient->envelope()->with('fields')->first();
    $fields = $env->fields()->where('recipient_id',$recipient->id)->get();

    if ($recipient->status === 'pending') $recipient-
>update(['status'=>'viewed']);
    EnvelopeEvent::create(['envelope_id'=>$env-
>id,'recipient_id'=>$recipient->id,'event'=>'link_opened','ip'=>request()-
>ip(),'user_agent'=>request()->userAgent()]);

    return view('signing.show', compact('env','recipient','fields'));
  }

  public function submit(Request $r, string $token, PdfSignerService $signer)
{
    $recipient = Recipient::where('token',$token)->firstOrFail();
    $env = $recipient->envelope;
    $r->validate(['signatures'=>['required','array','min:1']]);

    // Save dataURLs to PNG
    $paths = [];
    foreach ($r->signatures as $fieldId => $dataUrl) {
      $png = base64_decode(preg_replace('#^#i','',
$dataUrl));
      $rel = "envelopes/signatures/{$recipient->id}_{$fieldId}.png";
      Storage::disk('public')->put($rel, $png);
      $paths[$fieldId] = $rel;
    }

    $signer->stamp($env, $recipient, $paths);
    $recipient->update(['status'=>'signed','signed_at'=>now()]);
    EnvelopeEvent::create(['envelope_id'=>$env-
>id,'recipient_id'=>$recipient->id,'event'=>'signed','ip'=>request()-
>ip(),'user_agent'=>request()->userAgent()]);

    // Advance order
    $nextOrder = $env->recipients()->where('status','!=','signed')-
>min('signing_order');
    if ($nextOrder) {
      $batch = $env->recipients()->where('signing_order',$nextOrder)-
>where('status','pending')->get();
      foreach ($batch as $rec) {
        \Mail::raw("Please sign: ".route('sign.show',$rec->token),
function($m) use($rec,$env){
```

```
          $m->to($rec->email,$rec->name)->subject('Signature Request: '.$env->title);
        });
        $rec->update(['status'=>'emailed']);
      }
      $env->update(['status'=>'in_progress']);
    } else {
      $env->update(['status'=>'completed','completed_at'=>now()]);
      // Email final to all
      $final = "envelopes/final/{$env->id}_final.pdf";
      Storage::disk('public')->copy($env->pdf_path,$final);
      foreach ($env->recipients as $rec) {
        \Mail::raw("Completed: {$env->title}", function($m) use ($rec,$env,$final){
          $m->to($rec->email,$rec->name)->subject('Completed: '.$env->title)
            ->attach(Storage::disk('public')->path($final), ['as'=>$env->title.'.pdf','mime'=>'application/pdf']);
        });
      }
      EnvelopeEvent::create(['envelope_id'=>$env->id,'event'=>'completed']);
    }

    return view('signing.thanks');
  }
}
```

```
// ============================
// file: routes/web.php
// ============================
use App\Http\Controllers\Admin\EnvelopeController;
use App\Http\Controllers\Admin\DesignerController;
use App\Http\Controllers\SigningController;

Route::middleware(['auth']) // add 'can:admin' if only admins
  ->prefix('admin/envelopes')->name('admin.envelopes.')
  ->group(function () {
    Route::get('/', [EnvelopeController::class,'index'])->name('index');
    Route::get('/create', [EnvelopeController::class,'create'])->name('create');
    Route::post('/', [EnvelopeController::class,'store'])->name('store');
    Route::get('{envelope}', [EnvelopeController::class,'show'])->name('show');
    Route::post('{envelope}/recipients', [EnvelopeController::class,'addRecipients'])->name('recipients.add');

    Route::get('{envelope}/designer', [DesignerController::class,'edit'])->name('designer');
    Route::post('{envelope}/fields',
```

```
[DesignerController::class,'saveFields'])->name('fields.save');

    Route::post('{envelope}/send', [EnvelopeController::class,'send'])-
>name('send');
    Route::post('{envelope}/void', [EnvelopeController::class,'void'])-
>name('void');
  });

// Public signing
Route::get('/sign/{token}', [SigningController::class,'show'])-
>name('sign.show');
Route::post('/sign/{token}', [SigningController::class,'submit'])-
>name('sign.submit');
```

```
{{-- ========================================
file: resources/views/layouts/app.blade.php
Ultra design (Tailwind + glassmorphism)
======================================== --}}
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>{{ $title ?? 'E-Sign Admin' }}</title>
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <script src="https://cdn.tailwindcss.com"></script>
  <style>
    .glass { background: rgba(255,255,255,.06); backdrop-filter: blur(16px);
border: 1px solid rgba(255,255,255,.15); }
    .btn { @apply px-4 py-2 rounded-xl font-medium hover:scale-[1.01]
transition; }
    .btn-primary { @apply bg-indigo-600 text-white; }
    .btn-soft { @apply bg-white/10 text-white; }
    .chip { @apply px-2 py-1 text-xs rounded-full bg-white/10; }
  </style>
</head>
<body class="min-h-screen bg-gradient-to-br from-slate-950 via-slate-900 to-
indigo-950 text-slate-100">
  <div class="max-w-7xl mx-auto p-6">
    <div class="mb-6 flex items-center justify-between">
      <h1 class="text-2xl font-bold">{{ $title ?? 'E-Sign Admin' }}</h1>
      <div class="text-sm opacity-70">{{ auth()->user()->name ?? 'Admin' }}</
div>
    </div>

    @if(session('success'))
      <div class="glass mb-4 rounded-xl p-3 text-emerald-300 border border-
emerald-700/30">{{ session('success') }}</div>
    @endif
```

```
    @if(session('error'))
      <div class="glass mb-4 rounded-xl p-3 text-rose-300 border border-
rose-700/30">{{ session('error') }}</div>
    @endif

    <div class="glass rounded-2xl p-6">
      {{ $slot }}
    </div>
  </div>
</body>
</html>
```

```
{{-- ============================================
file: resources/views/admin/envelopes/create.blade.php
Upload PDF/Word
============================================ --}}
<x-layouts.app :title="'New Envelope'">
  <form class="space-y-6" method="POST"
action="{{ route('admin.envelopes.store') }}" enctype="multipart/form-data">
    @csrf
    <div class="grid md:grid-cols-2 gap-6">
      <div>
        <label class="block mb-2 text-sm">Title</label>
        <input name="title" class="w-full rounded-xl bg-white/5 border
border-white/10 p-3" placeholder="Contract – Sept"/>
      </div>
      <div>
        <label class="block mb-2 text-sm">PDF / DOCX</label>
        <input type="file" name="file" accept=".pdf,.doc,.docx" required
class="w-full rounded-xl bg-white/5 border border-white/10 p-3"/>
        <p class="text-xs opacity-70 mt-1">.docx auto-converts to PDF via
LibreOffice</p>
      </div>
    </div>
    <div class="flex gap-3">
      <button class="btn btn-primary">Upload & Continue</button>
      <a href="{{ route('admin.envelopes.index') }}" class="btn btn-
soft">Cancel</a>
    </div>
  </form>
</x-layouts.app>
```

```
{{-- ============================================
file: resources/views/admin/envelopes/index.blade.php
List envelopes
============================================ --}}
```

```
<x-layouts.app :title="'Envelopes'">
  <div class="flex justify-between mb-4">
    <a href="{{ route('admin.envelopes.create') }}" class="btn btn-primary">+
New Envelope</a>
  </div>
  <div class="overflow-x-auto">
    <table class="w-full text-sm">
      <thead class="text-left opacity-70">
        <tr><th class="py-2">Title</th><th>Status</th><th>Recipients</
th><th>Updated</th></tr>
      </thead>
      <tbody>
        @foreach($rows as $e)
          <tr class="border-t border-white/10">
            <td class="py-3"><a class="underline"
href="{{ route('admin.envelopes.show',$e) }}">{{ $e->title }}</a></td>
            <td><span class="chip">{{ $e->status }}</span></td>
            <td>{{ $e->recipients_count }}</td>
            <td>{{ $e->updated_at->diffForHumans() }}</td>
          </tr>
        @endforeach
      </tbody>
    </table>
  </div>
  <div class="mt-4">{{ $rows->links() }}</div>
</x-layouts.app>
```

```
{{-- ============================================
file: resources/views/admin/envelopes/show.blade.php
Envelope detail: add recipients, send
============================================ --}}
<x-layouts.app :title="$envelope->title">
  <div class="grid md:grid-cols-3 gap-6">
    <div class="md:col-span-2 space-y-4">
      <div class="glass rounded-xl p-4">
        <div class="flex items-center justify-between">
          <div class="font-semibold">Preview</div>
          <a href="{{ route('admin.envelopes.designer',$envelope) }}"
class="btn btn-soft">Open Designer</a>
        </div>
        <iframe src="{{ asset('storage/'.$envelope->pdf_path) }}" class="w-
full h-[600px] rounded-xl mt-3 bg-white"></iframe>
      </div>
      <div class="glass rounded-xl p-4">
        <div class="font-semibold mb-3">Events</div>
        <div class="space-y-2 text-sm">
          @foreach($envelope->events()->latest()->take(20)->get() as $ev)
            <div class="flex justify-between border-b border-white/10 pb-1">
```

```
              <div>{{ $ev->event }}</div>
              <div class="opacity-60">{{ $ev->created_at->toDayDateTimeString() }}</div>
            </div>
          @endforeach
        </div>
      </div>
    </div>
    <div class="space-y-6">
      <form class="glass rounded-xl p-4" method="POST" action="{{ route('admin.envelopes.recipients.add',$envelope) }}">
        @csrf
        <div class="font-semibold mb-3">Recipients</div>
        <div id="rec-list" class="space-y-3"></div>
        <button type="button" id="add-rec" class="btn btn-soft w-full">+ Add recipient</button>
        <button class="btn btn-primary w-full mt-3">Save Recipients</button>
      </form>

      <form method="POST" action="{{ route('admin.envelopes.send', $envelope) }}" class="glass rounded-xl p-4">
        @csrf
        <div class="flex items-center justify-between">
          <div class="font-semibold">Send for signature</div>
          <button class="btn btn-primary">Send</button>
        </div>
      </form>

      <form method="POST" action="{{ route('admin.envelopes.void', $envelope) }}" class="glass rounded-xl p-4">
        @csrf
        <div class="flex items-center justify-between">
          <div class="font-semibold text-rose-300">Void</div>
          <button class="btn bg-rose-700/80">Void Envelope</button>
        </div>
      </form>
    </div>
  </div>

  <script>
    const container = document.getElementById('rec-list');
    document.getElementById('add-rec').onclick = () => addRow();
    // starter one row
    addRow();

    function addRow() {
      const wrap = document.createElement('div');
      wrap.className = "grid grid-cols-6 gap-2";
      wrap.innerHTML = `
        <input name="recipients[][name]" placeholder="Name" class="col-span-2 p-2 rounded bg-white/5 border border-white/10">
```

```
        <input name="recipients[][email]" placeholder="Email" class="col-
span-3 p-2 rounded bg-white/5 border border-white/10">
        <input name="recipients[][signing_order]" type="number" value="1"
min="1" class="col-span-1 p-2 rounded bg-white/5 border border-white/10">
      `;
      container.appendChild(wrap);
    }
  </script>
</x-layouts.app>
```

```
{{-- ============================================================
file: resources/views/admin/envelopes/designer.blade.php
Zoho-style Designer: PDF.js + interact.js drag/resize + ultra UI
============================================================ --}}
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Designer – {{ $envelope->title }}</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <script src="https://cdn.jsdelivr.net/npm/interactjs/dist/
interact.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/signature_pad@5.0.4/dist/
signature_pad.umd.min.js"></script>
  <!-- PDF.js -->
  <script src="https://cdn.jsdelivr.net/npm/pdfjs-dist@4.4.168/build/
pdf.min.js"></script>
  <script>pdfjsLib.GlobalWorkerOptions.workerSrc="https://cdn.jsdelivr.net/
npm/pdfjs-dist@4.4.168/build/pdf.worker.min.js";</script>
  <style>
    body{background:radial-gradient(1200px 600px at 70% -10%, #3e2a8a33,
transparent), #0b0f1b;}
    .glass{background:rgba(255,255,255,.06);backdrop-
filter:blur(18px);border:1px solid rgba(255,255,255,.1);}
    .field{border:2px dashed rgba(99,102,241,.9);
background:rgba(99,102,241,.12); border-radius:10px; cursor:move;
position:absolute; display:flex; align-items:center; justify-content:center;
color:#e0e7ff; font-size:12px;}
    .recipient-dot{width:10px;height:10px;border-radius:
9999px;display:inline-block;margin-right:6px;}
  </style>
</head>
<body class="text-slate-100">
  <div class="p-4 grid grid-cols-12 gap-4">
    <!-- Sidebar -->
    <div class="col-span-3 glass rounded-2xl p-4">
      <div class="text-lg font-semibold mb-4">Designer</div>
```

```
      <div class="mb-6">
        <div class="text-sm opacity-80 mb-2">Recipients</div>
        @foreach($envelope->recipients as $r)
          <div class="flex items-center justify-between glass rounded-xl p-2
mb-2">
            <div><span class="recipient-dot" style="background:
{{ ['#4ade80','#60a5fa','#f472b6','#facc15'][$loop->index%4] }}"></
span>{{ $r->name }}</div>
            <div class="text-xs opacity-70">Order {{ $r->signing_order }}</
div>
          </div>
        @endforeach
      </div>

      <div class="mb-6">
        <div class="text-sm opacity-80 mb-2">Fields</div>
        <div class="grid grid-cols-2 gap-2">
          <button data-type="signature" class="glass p-2 rounded-lg hover:bg-
white/10">Signature</button>
          <button data-type="initial" class="glass p-2 rounded-lg hover:bg-
white/10">Initial</button>
          <button data-type="date" class="glass p-2 rounded-lg hover:bg-
white/10">Date</button>
          <button data-type="text" class="glass p-2 rounded-lg hover:bg-
white/10">Text</button>
          <button data-type="checkbox" class="glass p-2 rounded-lg hover:bg-
white/10">Checkbox</button>
        </div>
        <p class="text-xs opacity-70 mt-2">Click a field, then click on the
page to place it. Drag edges to resize.</p>
      </div>

      <form id="saveFields" method="POST"
action="{{ route('admin.envelopes.fields.save',$envelope) }}">
        @csrf
        <input type="hidden" name="fields" id="fieldsInput">
        <div class="text-sm opacity-80 mb-2">Assign To</div>
        <select id="recipientSelect" class="w-full p-2 rounded bg-white/5
border border-white/10 mb-4">
          @foreach($envelope->recipients as $r)
            <option value="{{ $r->id }}">{{ $r->name }} (Order {{ $r-
>signing_order }})</option>
          @endforeach
        </select>
        <button class="w-full p-3 rounded-xl bg-indigo-600
hover:opacity-95">Save Fields</button>
        <a href="{{ route('admin.envelopes.show',$envelope) }}" class="block
text-center mt-3 underline">Back</a>
      </form>
    </div>
```

```
    <!-- Canvas Area -->
    <div class="col-span-9">
      <div class="glass rounded-2xl p-4">
        <div class="flex items-center justify-between mb-3">
          <div class="font-semibold">Preview: {{ $envelope-
>original_filename }}</div>
          <div class="text-xs opacity-70">Click to place fields. Zoom:
            <button id="zoomOut" class="px-2">–</button> <span
id="zoomLbl">100%</span> <button id="zoomIn" class="px-2">+</button>
          </div>
        </div>
        <div id="pages" class="space-y-4"></div>
      </div>
    </div>
  </div>

  <script>
    // --- PDF render ---
    const pdfUrl = "{{ asset('storage/'.$envelope->pdf_path) }}";
    let PDF=null, scale=1.0, placed=[];
    const pagesDiv = document.getElementById('pages');
    const zoomLbl = document.getElementById('zoomLbl');
    document.getElementById('zoomIn').onclick = ()=>{scale=Math.min(2.5,
scale+0.1); render();};
    document.getElementById('zoomOut').onclick= ()=>{scale=Math.max(.5,
scale-0.1); render();};

    pdfjsLib.getDocument(pdfUrl).promise.then(pdf=>{ PDF=pdf; render(); });

    async function render(){
      pagesDiv.innerHTML='';
      zoomLbl.textContent = Math.round(scale*100)+'%';
      for(let p=1;p<=PDF.numPages;p++){
        const page = await PDF.getPage(p);
        const viewport = page.getViewport({scale});
        const wrap = document.createElement('div');
        wrap.className="relative mx-auto bg-white rounded-xl overflow-
hidden";
        wrap.style.width=viewport.width+'px';
        wrap.style.height=viewport.height+'px';
        wrap.dataset.page = p;

        const canvas=document.createElement('canvas');
        canvas.width=viewport.width;
        canvas.height=viewport.height;
        wrap.appendChild(canvas);

        const ctx=canvas.getContext('2d');
        await page.render({canvasContext:ctx, viewport}).promise;

        // click to place active field
```

```
        wrap.addEventListener('click', e=>{
          if (!activeType) return;
          const rect=wrap.getBoundingClientRect();
          const x=e.clientX-rect.left, y=e.clientY-rect.top;
          const w=150, h=40; // default size
          addFieldDiv(p, x-w/2, y-h/2, w, h, activeType);
        });

        pagesDiv.appendChild(wrap);
      }
      // rehydrate existing (server) fields
      existingFields.forEach(f=>{
        if (f.page<=PDF.numPages) addFieldDiv(f.page, f.x*scale, f.y*scale,
f.w*scale, f.h*scale, f.type, f.recipient_id, f.id);
      })
    }

    // --- existing fields from server (converted already in PDF units).
We'll place scaled.
    const existingFields = @json($envelope->fields->map(fn($f)=>[
      'id'=>$f->id,'type'=>$f->type,'page'=>$f->page,
      'x'=>(float)$f->x,'y'=>(float)$f->y,'w'=>(float)$f->w,'h'=>(float)$f-
>h,'recipient_id'=>$f->recipient_id
    ]));

    // --- Place fields & interact.js for drag/resize ---
    let activeType=null;
    document.querySelectorAll('[data-type]').forEach(b=>{
      b.onclick=()=>{ activeType=b.dataset.type;
document.querySelectorAll('[data-
type]').forEach(x=>x.classList.remove('ring-2'));
b.classList.add('ring-2'); }
    });

    function addFieldDiv(page, x, y, w, h, type, recipientId=null, id=null){
      const pageEl = [...pagesDiv.children][page-1];
      const el = document.createElement('div');
      el.className = 'field';
      el.style.left=x+'px'; el.style.top=y+'px'; el.style.width=w+'px';
el.style.height=h+'px';
      el.dataset.page = page; el.dataset.type = type; el.dataset.id = id ||
'';
      el.dataset.recipientId = recipientId ||
document.getElementById('recipientSelect').value;
      el.innerHTML = `<div>${type.toUpperCase()}</div>`;
      pageEl.appendChild(el);

      interact(el).draggable({
        listeners:{
          move(ev){
            const target = ev.target;
```

```
            const tx = (parseFloat(target.style.left)||0)+ev.dx;
            const ty = (parseFloat(target.style.top)||0)+ev.dy;
            target.style.left = tx+'px';
            target.style.top  = ty+'px';
          }
        }
      }).resizable({
        edges:{left:true,right:true,top:true,bottom:true}
      }).on('resizemove', function (ev) {
        let {x, y} = ev.target.dataset;
        x = (parseFloat(ev.target.style.left)||0) + ev.deltaRect.left;
        y = (parseFloat(ev.target.style.top)||0) + ev.deltaRect.top;
        Object.assign(ev.target.style, {
          width: ev.rect.width + 'px',
          height: ev.rect.height + 'px',
          left: x + 'px',
          top: y + 'px'
        });
      });

      placed.push(el);
    }

    // Save fields: convert canvas px → PDF units using page scale
    document.getElementById('saveFields').addEventListener('submit', e=>{
      const out=[];
      placed.forEach(el=>{
        const page = parseInt(el.dataset.page);
        const pdfX = parseFloat(el.style.left)/scale;
        const pdfY = parseFloat(el.style.top)/scale;
        const pdfW = parseFloat(el.style.width)/scale;
        const pdfH = parseFloat(el.style.height)/scale;
        out.push({
          id: el.dataset.id || null,
          type: el.dataset.type,
          page,
          x: pdfX, y: pdfY, w: pdfW, h: pdfH,
          recipient_id: el.dataset.recipientId ||
document.getElementById('recipientSelect').value,
          required: true
        });
      });
      document.getElementById('fieldsInput').value = JSON.stringify(out);
    });
  </script>
</body>
</html>
```

```
{{-- ========================================
file: resources/views/signing/show.blade.php
Signer page with SignaturePad on required fields
========================================== --}}
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Sign – {{ $env->title }}</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <script src="https://cdn.jsdelivr.net/npm/pdfjs-dist@4.4.168/build/
pdf.min.js"></script>
  <script>pdfjsLib.GlobalWorkerOptions.workerSrc="https://cdn.jsdelivr.net/
npm/pdfjs-dist@4.4.168/build/pdf.worker.min.js";</script>
  <script src="https://cdn.jsdelivr.net/npm/signature_pad@5.0.4/dist/
signature_pad.umd.min.js"></script>
  <style>
    body{background:linear-gradient(120deg,#0b1020,#0a0920 60%,#121d3a);}
    .glass{background:rgba(255,255,255,.06);backdrop-
filter:blur(18px);border:1px solid rgba(255,255,255,.1);}
    .box{position:absolute;border:2px dashed
#22c55e;background:#22c55e22;border-radius:10px;}
  </style>
</head>
<body class="text-slate-100">
  <div class="max-w-6xl mx-auto p-4">
    <div class="mb-4 flex items-center justify-between">
      <div>
        <div class="text-xl font-bold">{{ $env->title }}</div>
        <div class="text-sm opacity-70">Hello {{ $recipient->name }}, please
sign the highlighted fields.</div>
      </div>
      <form id="submitForm" method="POST" action="{{ route('sign.submit',
$recipient->token) }}">
        @csrf
        <input type="hidden" name="signatures" id="signaturesInput">
        <button class="px-4 py-2 rounded-xl bg-emerald-600
hover:opacity-95">Submit Signatures</button>
      </form>
    </div>

    <div id="pages" class="space-y-6"></div>
  </div>

  <script>
    const pdfUrl = "{{ asset('storage/'.$env->pdf_path) }}";
    const myFields = @json($fields->map(fn($f)=>[
      'id'=>$f->id,'type'=>$f->type,'page'=>$f->page,'x'=>(float)$f-
>x,'y'=>(float)$f->y,'w'=>(float)$f->w,'h'=>(float)$f->h
    ]));
```

```
    let pads = {}; // fieldId => SignaturePad

    pdfjsLib.getDocument(pdfUrl).promise.then(async pdf=>{
      for (let p=1;p<=pdf.numPages;p++){
        const page = await pdf.getPage(p);
        const viewport = page.getViewport({scale:1.2});
        const wrap = document.createElement('div');
        wrap.className = "relative mx-auto bg-white rounded-xl overflow-
hidden glass p-2";
        wrap.style.width = viewport.width+'px';

        const canvas = document.createElement('canvas');
        canvas.width=viewport.width; canvas.height=viewport.height;
        wrap.appendChild(canvas);
        const ctx=canvas.getContext('2d');
        await page.render({canvasContext:ctx, viewport}).promise;

        // Overlay fields for this page
        myFields.filter(f=>f.page===p).forEach(f=>{
          const div = document.createElement('div');
          div.className='box';
          div.style.left=(f.x*1.2)+'px';
          div.style.top =(f.y*1.2)+'px';
          div.style.width=(f.w*1.2)+'px';
          div.style.height=(f.h*1.2)+'px';
          wrap.appendChild(div);

          if (f.type==='signature' || f.type==='initial'){
            const c = document.createElement('canvas');
            c.width = Math.max(200, f.w*1.2);
            c.height= Math.max(80, f.h*1.2);
            c.style.background='#fff';
            c.style.borderRadius='10px';
            c.style.boxShadow='inset 0 0 0 1px #0002';
            div.appendChild(c);
            pads[f.id] = new SignaturePad(c, {minWidth:1, maxWidth:2.5,
penColor:"#0a0a0a"});
          } else if (f.type==='checkbox'){
            const cb = document.createElement('input'); cb.type='checkbox';
            cb.style.transform='scale(1.5)'; cb.style.accentColor='#22c55e';
            div.appendChild(cb);
            // Non-signature fields aren't stamped in this demo — extend as
needed.
          } else if (f.type==='date' || f.type==='text'){
            const inp = document.createElement('input'); inp.className='p-2
text-black rounded';
            inp.placeholder = f.type==='date' ? 'YYYY-MM-DD' : 'Type here';
            inp.style.width='100%'; inp.style.height='100%';
div.appendChild(inp);
          }
        });
```

```
        document.getElementById('pages').appendChild(wrap);
      }
    });

    document.getElementById('submitForm').addEventListener('submit', e=>{
      const payload = {};
      Object.entries(pads).forEach(([fieldId, pad])=>{
        if (!pad.isEmpty()) payload[fieldId] = pad.toDataURL('image/png');
      });
      document.getElementById('signaturesInput').value =
JSON.stringify(payload);
    });
  </script>
</body>
</html>
```

```
{{-- ================================
file: resources/views/signing/thanks.blade.php
================================ --}}
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Thank you</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="min-h-screen grid place-items-center bg-slate-950 text-
slate-100">
  <div class="p-10 rounded-2xl bg-white/5 border border-white/10">
    <h1 class="text-2xl font-bold mb-2">Thanks! 🎉</h1>
    <p class="opacity-80">Your signatures were submitted successfully.</p>
  </div>
</body>
</html>
```

🛠️Setup & Run

```
# Install deps
composer update

# Migrations + queue
php artisan migrate
php artisan queue:table && php artisan migrate
php artisan queue:work
```

```
# Storage symlink
php artisan storage:link
```

.env (mail)

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.yourhost.com
MAIL_PORT=587
MAIL_USERNAME=...
MAIL_PASSWORD=...
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=no-reply@yourapp.com
MAIL_FROM_NAME="Your App"
```

LibreOffice - Linux: `sudo apt-get install libreoffice` - Windows: install LibreOffice, ensure `soffice` in PATH.

Permissions - Ensure `storage/app/public/envelopes/*` is writable.

Notes

- Designer uses **PDF.js** + **interact.js** for drag/resize; fields saved in **PDF user units**.
- Signing uses **Signature Pad**; server stamps PNGs into the PDF with **FPDI**.
- Supports **.doc/.docx** upload via headless conversion to **PDF**.
- Ordered signing: same `signing_order` = parallel batch.
- Extend to stamp text/date/checkbox by drawing text or marks in `PdfSignerService`.