Multi-Stage Approval Implementation Guide

Overview

This guide implements a multi-stage approval system where:

- Approvals must be sequential within a stage
- Only current approver can see and act on documents
- Template users can view approval status but cannot approve/reject
- Final approver triggers the external API

Step 1: Database Migration for Approval Tracking

Create a new migration to track approval progress:

```
php
```

```php
// database/migrations/create_approval_progress_table.php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateApprovalProgressTable extends Migration
{
    public function up()
    {
        Schema::create('approval_progress', function (Blueprint $table) {
            $table->id();
            $table->string('doc_num');
            $table->string('wdd_code');
            $table->string('draft_entry');
            $table->integer('stage_id');
            $table->integer('current_approver_user_id');
            $table->integer('current_approver_sequence')->default(1);
            $table->enum('status', ['pending', 'approved', 'rejected'])->default('pending');
            $table->text('remarks')->nullable();
            $table->integer('approved_by')->nullable();
            $table->timestamp('approved_at')->nullable();
            $table->json('approval_history')->nullable(); // Store approval chain
            $table->timestamps();

            $table->unique(['doc_num', 'wdd_code', 'draft_entry']);
            $table->foreign('stage_id')->references('id')->on('stages');
            $table->foreign('current_approver_user_id')->references('id')->on('user');
        });
    }

    public function down()
    {
        Schema::dropIfExists('approval_progress');
    }
}
```

Step 2: Create Approval Progress Model

php

```php
// app/Models/ApprovalProgress.php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class ApprovalProgress extends Model
{
    protected $table = 'approval_progress';

    protected $fillable = [
        'doc_num', 'wdd_code', 'draft_entry', 'stage_id',
        'current_approver_user_id', 'current_approver_sequence',
        'status', 'remarks', 'approved_by', 'approved_at', 'approval_history'
    ];

    protected $casts = [
        'approval_history' => 'array',
        'approved_at' => 'datetime'
    ];

    public function stage()
    {
        return $this->belongsTo(Stage::class, 'stage_id');
    }

    public function currentApprover()
    {
        return $this->belongsTo(User::class, 'current_approver_user_id');
    }

    public function approvedBy()
    {
        return $this->belongsTo(User::class, 'approved_by');
    }
}
```

Step 3: Create Required Models

```
php
```

```php
// app/Models/Stage.php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Stage extends Model
{
    protected $table = 'stages';

    protected $fillable = [
        'stage_name', 'stage_description', 'no_of_approval_required', 'no_of_rejection_required'
    ];

    public function authorizers()
    {
        return $this->hasMany(StageAuthorizer::class, 'stage_ID');
    }
}

// app/Models/StageAuthorizer.php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class StageAuthorizer extends Model
{
    protected $table = 'stages_authorizers';

    protected $fillable = ['stage_ID', 'user_ID'];

    public function stage()
    {
        return $this->belongsTo(Stage::class, 'stage_ID');
    }

    public function user()
    {
        return $this->belongsTo(User::class, 'user_ID');
    }
}

// app/Models/ApprovalTemplateUser.php
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class ApprovalTemplateUser extends Model
{
    protected $table = 'approval_template_users';

    protected $fillable = ['approval_template_ID', 'user_ID'];

    public function user()
    {
        return $this->belongsTo(User::class, 'user_ID');
    }
}
```

Step 4: Create Approval Service

php

```php
// app/Services/ApprovalService.php
<?php

namespace App\Services;

use App\Models\ApprovalProgress;
use App\Models\StageAuthorizer;
use App\Models\ApprovalTemplateUser;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Log;

class ApprovalService
{
    public function initializeApprovalProcess($documentData, $stageId = 1)
    {
        // Get first approver for the stage
        $firstApprover = StageAuthorizer::where('stage_ID', $stageId)
            ->orderBy('id')
            ->first();

        if (!$firstApprover) {
            throw new \Exception("No approvers found for stage {$stageId}");
        }

        return ApprovalProgress::updateOrCreate(
            [
                'doc_num' => $documentData['doc_num'],
                'wdd_code' => $documentData['wdd_code'],
                'draft_entry' => $documentData['draft_entry']
            ],
            [
                'stage_id' => $stageId,
                'current_approver_user_id' => $firstApprover->user_ID,
                'current_approver_sequence' => 1,
                'status' => 'pending',
                'approval_history' => []
            ]
        );
    }

    public function processApproval($documentData, $userId, $remarks = null)
    {
        DB::beginTransaction();

        try {
```

```php
$progress = ApprovalProgress::where('doc_num', $documentData['doc_num'])
    ->where('wdd_code', $documentData['wdd_code'])
    ->where('draft_entry', $documentData['draft_entry'])
    ->first();

if (!$progress) {
    // Initialize if not exists
    $progress = $this->initializeApprovalProcess($documentData);
}

// Check if current user is the expected approver
if ($progress->current_approver_user_id != $userId) {
    throw new \Exception("You are not authorized to approve this document at this stage");
}

// Add to approval history
$history = $progress->approval_history ?? [];
$history[] = [
    'user_id' => $userId,
    'user_name' => auth()->user()->userName,
    'sequence' => $progress->current_approver_sequence,
    'action' => 'approved',
    'remarks' => $remarks,
    'timestamp' => now()
];

// Get next approver in sequence
$nextApprover = StageAuthorizer::where('stage_ID', $progress->stage_id)
    ->orderBy('id')
    ->offset($progress->current_approver_sequence)
    ->first();

if ($nextApprover) {
    // Move to next approver
    $progress->update([
        'current_approver_user_id' => $nextApprover->user_ID,
        'current_approver_sequence' => $progress->current_approver_sequence + 1,
        'approval_history' => $history
    ]);
} else {
    // Last approver - mark as approved and call external API
    $progress->update([
        'status' => 'approved',
        'approved_by' => $userId,
        'approved_at' => now(),
        'approval_history' => $history
    ]);
```

```
            // Call external API for final approval
            $this->callExternalApprovalAPI($documentData);
        }

        DB::commit();
        return $progress;

    } catch (\Exception $e) {
        DB::rollback();
        Log::error("Approval process failed: " . $e->getMessage());
        throw $e;
    }
}

public function processRejection($documentData, $userId, $remarks)
{
    DB::beginTransaction();

    try {
        $progress = ApprovalProgress::where('doc_num', $documentData['doc_num'])
            ->where('wdd_code', $documentData['wdd_code'])
            ->where('draft_entry', $documentData['draft_entry'])
            ->first();

        if (!$progress) {
            throw new \Exception("Approval process not found");
        }

        // Check if current user is the expected approver
        if ($progress->current_approver_user_id != $userId) {
            throw new \Exception("You are not authorized to reject this document");
        }

        // Add to approval history
        $history = $progress->approval_history ?? [];
        $history[] = [
            'user_id' => $userId,
            'user_name' => auth()->user()->userName,
            'sequence' => $progress->current_approver_sequence,
            'action' => 'rejected',
            'remarks' => $remarks,
            'timestamp' => now()
        ];

        $progress->update([
            'status' => 'rejected',
```

```
            'approved_by' => $userId,
            'approved_at' => now(),
            'approval_history' => $history
        ]);

        // Call external API for rejection
        $this->callExternalRejectionAPI($documentData);

        DB::commit();
        return $progress;

    } catch (\Exception $e) {
        DB::rollback();
        Log::error("Rejection process failed: " . $e->getMessage());
        throw $e;
    }
}

private function callExternalApprovalAPI($documentData)
{
    $response = Http::withHeaders([
        'WddCode' => $documentData['wdd_code'],
        'DraftEntry' => $documentData['draft_entry'],
        'SAPID' => auth()->user()->sap_usercode,
        'SAPIDNAME' => auth()->user()->userName
    ])->post('http://103.73.190.42:7024/api/ApprovedRequestPostDoc');

    if (!$response->successful()) {
        throw new \Exception("External approval API call failed");
    }

    return $response->json();
}

private function callExternalRejectionAPI($documentData)
{
    $response = Http::withHeaders([
        'WddCode' => $documentData['wdd_code'],
        'DraftEntry' => $documentData['draft_entry']
    ])->post('http://103.73.190.42:7024/api/RejectRequest');

    if (!$response->successful()) {
        throw new \Exception("External rejection API call failed");
    }

    return $response->json();
}
```

```php
public function isTemplateUser($userId)
{
    return ApprovalTemplateUser::where('user_ID', $userId)->exists();
}

public function canUserApprove($documentData, $userId)
{
    $progress = ApprovalProgress::where('doc_num', $documentData['doc_num'])
        ->where('wdd_code', $documentData['wdd_code'])
        ->where('draft_entry', $documentData['draft_entry'])
        ->first();

    if (!$progress) {
        // Check if user is first approver
        $firstApprover = StageAuthorizer::where('stage_ID', 1)
            ->orderBy('id')
            ->first();
        return $firstApprover && $firstApprover->user_ID == $userId;
    }

    return $progress->current_approver_user_id == $userId && $progress->status == 'pending';
}

public function getApprovalStatus($documentData)
{
    $progress = ApprovalProgress::where('doc_num', $documentData['doc_num'])
        ->where('wdd_code', $documentData['wdd_code'])
        ->where('draft_entry', $documentData['draft_entry'])
        ->with('currentApprover')
        ->first();

    if (!$progress) {
        // Get first approver
        $firstApprover = StageAuthorizer::where('stage_ID', 1)
            ->with('user')
            ->orderBy('id')
            ->first();

        return [
            'current_approver_name' => $firstApprover->user->userName ?? 'Unknown',
            'status' => 'Pending'
        ];
    }

    return [
        'current_approver_name' => $progress->currentApprover->userName ?? 'Unknown',
```

```
            'status' => ucfirst($progress->status)
        ];
    }
}
```

Step 5: Update Controller

php

```php
// Update your existing controller method
public function viewApInvoice()
{
    $apiUrl = 'http://103.73.190.42:7024/api/GetApprovalDetails';
    $headers = ['DocType: 18'];

    $ch = curl_init($apiUrl);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

    $response = curl_exec($ch);
    curl_close($ch);

    $data['content'] = [];
    $user = Auth::user();
    $approvalService = new \App\Services\ApprovalService();

    if ($response) {
        $decoded = json_decode($response, true);
        $allData = isset($decoded['data']) ? $decoded['data'] : $decoded;

        if ($user && $user->role_ID == 1) {
            $data['content'] = $allData;
        } else {
            // Filter data based on user permissions
            $filteredData = [];

            foreach ($allData as $item) {
                $documentData = [
                    'doc_num' => $item['DocNum'],
                    'wdd_code' => $item['WddCode'],
                    'draft_entry' => $item['DraftEntry']
                ];

                // Initialize approval process if not exists
                $approvalService->initializeApprovalProcess($documentData);

                // Check if user can see this document
                if ($approvalService->isTemplateUser($user->id)) {
                    // Template users can see all documents
                    $item['approval_status'] = $approvalService->getApprovalStatus($documentData);
                    $filteredData[] = $item;
                } else if ($approvalService->canUserApprove($documentData, $user->id)) {
                    // Only current approver can see and act
                    $filteredData[] = $item;
                } else if (isset($item['SAPID']) && $item['SAPID'] == $user->sap_usercode) {
```

```
                // Original SAP user can always see their documents
                $filteredData[] = $item;
            }
        }

        $data['content'] = $filteredData;
    }
}

$data['is_template_user'] = $approvalService->isTemplateUser($user->id);

return view('apmanagement::view-ap-invoice', compact('data'));
}

// Add new methods for approval/rejection
public function approveDocument(Request $request)
{
$approvalService = new \App\Services\ApprovalService();

try {
    $documentData = [
        'doc_num' => $request->doc_no,
        'wdd_code' => $request->wdd_code,
        'draft_entry' => $request->draft_entry
    ];

    $progress = $approvalService->processApproval(
        $documentData,
        auth()->user()->id,
        $request->remark
    );

    return response()->json([
        'success' => true,
        'message' => 'Document approved successfully',
        'data' => $progress
    ]);

} catch (\Exception $e) {
    return response()->json([
        'success' => false,
        'message' => $e->getMessage()
    ], 400);
}
}

public function rejectDocument(Request $request)
```

```
{
    $approvalService = new \App\Services\ApprovalService();

    try {
        $documentData = [
            'doc_num' => $request->doc_no,
            'wdd_code' => $request->wdd_code,
            'draft_entry' => $request->draft_entry
        ];

        $progress = $approvalService->processRejection(
            $documentData,
            auth()->user()->id,
            $request->remark
        );

        return response()->json([
            'success' => true,
            'message' => 'Document rejected successfully',
            'data' => $progress
        ]);

    } catch (\Exception $e) {
        return response()->json([
            'success' => false,
            'message' => $e->getMessage()
        ], 400);
    }
}
```

Step 6: Update Blade Template

blade

```
{{-- Update the table header --}}
<thead>
    <tr>
        <th data-toggle="true">Sr.No</th>
        <th data-toggle="true">Doc. No</th>
        <th data-toggle="true">User Code</th>
        <th data-toggle="true">Doc. Date</th>
        <th data-toggle="true">Customer/Vendor Code</th>
        <th data-toggle="true">Total Amount</th>
        <th data-toggle="true">Approval Type</th>

        @if($data['is_template_user'])
            <th data-toggle="true">Approver Name</th>
            <th data-toggle="true">Status</th>
        @endif

        @if(!$data['is_template_user'])
            <th data-toggle="true">Action</th>
        @endif
    </tr>
</thead>

{{-- Update the table body --}}
<tbody id='assignUserData'>
    @foreach($data['content'] as $cnt)
        @php
            // Your existing PDF and attachment logic here
            $pdfUrl = null;
            $attachments = [];
            // ... existing logic ...
        @endphp

        <tr>
            <td class="srNoteditable">{{ $loop->iteration }}</td>
            <td>{{ $cnt['DocNum'] }}</td>
            <td>{{ $cnt['SAPIDNAME'] ?? 'NA' }}</td>
            <td>{{ date('d-m-Y', strtotime(str_replace('/', '-', $cnt['Creation_Date']))) }}</td>
            <td>{{ $cnt['CardName'] . ' - ' . $cnt['CardCode'] }}</td>
            <td>{{ number_format((float)$cnt['DocTotal'], 2) }}</td>
            <td>{{ $cnt['ApprovalType'] ?? 'NA' }}</td>

            @if($data['is_template_user'])
                <td>{{ $cnt['approval_status']['current_approver_name'] ?? 'N/A' }}</td>
                <td>
                    <span class="badge badge-{{ $cnt['approval_status']['status'] == 'Approved' ? 'success' : ($cnt['approval_sta
                        {{ $cnt['approval_status']['status'] ?? 'Pending' }}
```

```
        </span>
    </td>
@endif

@if(!$data['is_template_user'])
    <td class="d-flex align-items-center gap-1">
        {{-- Your existing PDF and attachment icons --}}
        @if ($pdfUrl)
            <a href="{{ $pdfUrl }}" target="_blank" title="Open PDF">
                <button type="button" class="btn btn-sm btn-outline-secondary">
                    <i class="fas fa-file-alt text-danger"></i>
                </button>
            </a>
        @else
            <button type="button" class="btn btn-sm btn-outline-secondary" disabled title="No PDF Available">
                <i class="fas fa-file-alt text-muted"></i>
            </button>
        @endif

        {{-- Attachment Icon --}}
        <div class="position-relative attachment-icon-wrapper">
            <button type="button" class="btn btn-sm btn-outline-secondary toggle-attachments" title="Attachmen
                <i class="fas fa-paperclip text-secondary"></i>
            </button>
            <div class="hover-icons text-center">
                @forelse ($attachments as $url)
                    <a href="{{ $url }}" target="_blank" title="Download Attachment">
                        <i class="fas fa-file-archive text-warning"></i>
                    </a>
                @empty
                    <small class="text-muted">No Attachments</small>
                @endforelse
            </div>
        </div>

        {{-- Approve & Reject Buttons --}}
        <button type="button" class="btn btn-sm btn-outline-success m-1 approve-btn" title="Approve"
                data-docno="{{ $cnt['DocNum'] }}"
                data-wddcode="{{ $cnt['WddCode'] }}"
                data-draftentry="{{ $cnt['DraftEntry'] }}">
            <i class="fas fa-thumbs-up"></i>
        </button>
        <button type="button" class="btn btn-sm btn-outline-danger m-1 reject-btn" title="Reject"
                data-docno="{{ $cnt['DocNum'] }}"
                data-wddcode="{{ $cnt['WddCode'] }}"
                data-draftentry="{{ $cnt['DraftEntry'] }}">
            <i class="fas fa-thumbs-down"></i>
```

```
                </button>
            </td>
        @endif
    </tr>
@endforeach
</tbody>
```

Step 7: Update JavaScript

javascript

```
// Update the approve form submit handler
$('#approveForm').submit(function (e) {
    e.preventDefault();

    var button = $('#approveForm button[type="submit"]');
    button.prop('disabled', true).text('Approving...');

    const payload = {
        doc_no: $('#approveDocNo').val(),
        wdd_code: $('#approveWddCode').val(),
        draft_entry: $('#approveDraftEntry').val(),
        remark: $('#approveRemark').val(),
        _token: '{{ csrf_token() }}'
    };

    $.ajax({
        url: "{{ route('approve.document') }}", // Add this route
        type: 'POST',
        data: payload,
        success: function (response) {
            if (response.success) {
                alert('Document processed successfully!');
                $('#approveRemarkModal').modal('hide');
                location.reload();
            } else {
                alert('Error: ' + response.message);
            }
        },
        error: function (xhr) {
            alert('Error: ' + xhr.responseJSON.message);
        },
        complete: function () {
            button.prop('disabled', false).text('Approve');
        }
    });
});

// Update the reject form submit handler
$('#rejectForm').submit(function (e) {
    e.preventDefault();

    var button = $('#rejectForm button[type="submit"]');
    button.prop('disabled', true).text('Rejecting...');

    const payload = {
        doc_no: $('#rejectDocNo').val(),
```

```
        wdd_code: $('#rejectWddCode').val(),
        draft_entry: $('#rejectDraftEntry').val(),
        remark: $('#rejectRemark').val(),
        _token: '{{ csrf_token() }}'
    };

    $.ajax({
        url: "{{ route('reject.document') }}", // Add this route
        type: 'POST',
        data: payload,
        success: function (response) {
            if (response.success) {
                alert('Document rejected successfully!');
                $('#rejectRemarkModal').modal('hide');
                location.reload();
            } else {
                alert('Error: ' + response.message);
            }
        },
        error: function (xhr) {
            alert('Error: ' + xhr.responseJSON.message);
        },
        complete: function () {
            button.prop('disabled', false).text('Reject');
        }
    });
});
```

Step 8: Add Routes

php

```
// Add these routes to your routes file
Route::post('/approve-document', [YourController::class, 'approveDocument'])->name('approve.document');
Route::post('/reject-document', [YourController::class, 'rejectDocument'])->name('reject.document');
```

Step 9: CSS for Template Users

css

```
.badge-success { background-color: #28a745; }
.badge-danger { background-color: #dc3545; }
.badge-warning { background-color: #ffc107; color: #212529; }
```

Key Features Implemented:

1. **Sequential Approval**: Users must approve in order within each stage
2. **Visibility Control**: Only current approver can see/act on documents
3. **Template User View**: Special columns for monitoring users
4. **External API Integration**: Calls approval/rejection APIs at appropriate times
5. **Audit Trail**: Maintains history of all approval actions
6. **Flexible Stage Management**: Easily configure different approval stages

Testing Steps:

1. Run migrations to create the approval_progress table
2. Test with Vaibhav login - should see documents requiring approval
3. Approve as Vaibhav - document should move to Kalpak
4. Login as Kalpak - should see the document
5. Login as Kedar/Nitin - should see status columns
6. Test rejection at any stage - should call rejection API immediately
7. Test final approval - should call approval API

This implementation provides a robust multi-stage approval system that meets all your requirements while maintaining data integrity and proper API integration.