# BAB 3
# METODOLOGI PENELITIAN

## 3.1 Pengumpulan Kebutuhan (*Requirement Gathering*)

Tahap awal dalam perancangan sistem ini adalah pengumpulan kebutuhan sistem. Proses ini dilakukan melalui studi terhadap *workflow* dokumen skripsi yang sudah ada yaitu *platform* Akademik dan wawancara mendalam dengan pihak terkait di Universitas Multimedia Nusantara.

Wawancara dilakukan dengan Manajer BIA, Dosen Program Studi Informatika, dan Alumni Informatika Universitas Multimedia Nusantara 2025, untuk mendapatkan pemahaman yang jelas mengenai permasalahan yang ada, *workflow* yang diharapkan, dan kebutuhan pengguna dari sisi institusi. Hasil dari wawancara tersebut menjadi dasar utama dalam mendefinisikan kebutuhan dari sistem yang akan dibangun. Kebutuhan ini dikelompokkan berdasarkan peran pengguna:

1. Peran Mahasiswa:
    - Login ke sistem menggunakan akun mahasiswa.
    - Mengunggah file skripsi dengan format PDF.
    - Melihat riwayat dan status dokumen yang telah diunggah.
    - Logout dari sistem.
2. Peran Dosen:
    - Login ke sistem menggunakan akun dosen.
    - Melihat daftar dokumen mahasiswa yang menunggu persetujuan.
    - Membuka (preview) dan meninjau salah satu dokumen dari daftar.
    - Menyetujui dan memberikan tanda tangan digital pada dokumen.
    - Menolak dokumen skripsi dan memberikan revisi.
    - Logout dari sistem.
3. Peran Kepala Program Studi (Kaprodi):
    - Login ke sistem menggunakan akun Kaprodi.

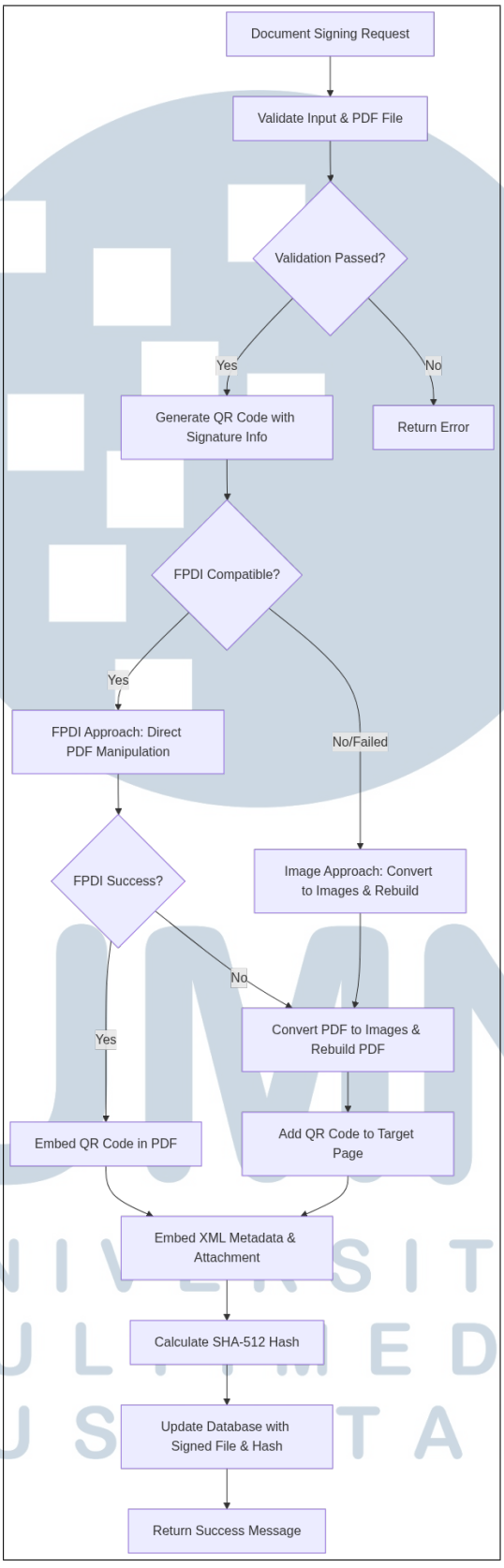
- Melihat daftar dokumen mahasiswa yang menunggu persetujuan.
- Membuka (preview) dan meninjau salah satu dokumen dari daftar.
- Menyetujui dan memberikan tanda tangan digital pada dokumen.
- Menolak dokumen skripsi dan memberikan revisi.
- Melakukan verifikasi dokumen melalui halaman verifikasi, untuk menguji keabsahannya (menggunakan dokumen valid, yang telah diubah, dan yang tidak ditandatangani).

4. Peran Admin (BIA):

- Login ke sistem dengan akun administrator.
- Melihat daftar akun mahasiswa.
- Menambahkan akun mahasiswa baru.
- Melihat daftar akun dosen.
- Menambahkan akun dosen baru.
- Mengganti periode tahun ajaran yang sedang aktif di sistem.
- Memantau status pengumpulan skripsi mahasiswa secara keseluruhan.
- Logout dari sistem.

## 3.2 Perancangan Alur Kerja Sistem

*Workflow* sistem secara keseluruhan, mulai dari penyerahan hingga verifikasi dokumen, dirancang untuk memastikan keamanan di setiap tahap. Proses ini digambarkan pada *flowchart* di Gambar 3.1.




Gambar 3.1. *Flowchart* Alur Kerja Sistem

*Workflow* sistem pada Gambar 3.1 diawali dengan *validation*, mencakup pemeriksaan format dan ukuran *file* PDF. Setelah validasi berhasil, sistem menghasilkan *QR code* yang berisi informasi penanda tangan berupa *signature info*. Signature info mencakup beberapa elemen kunci yang disematkan pada QR code dan metadata XML:

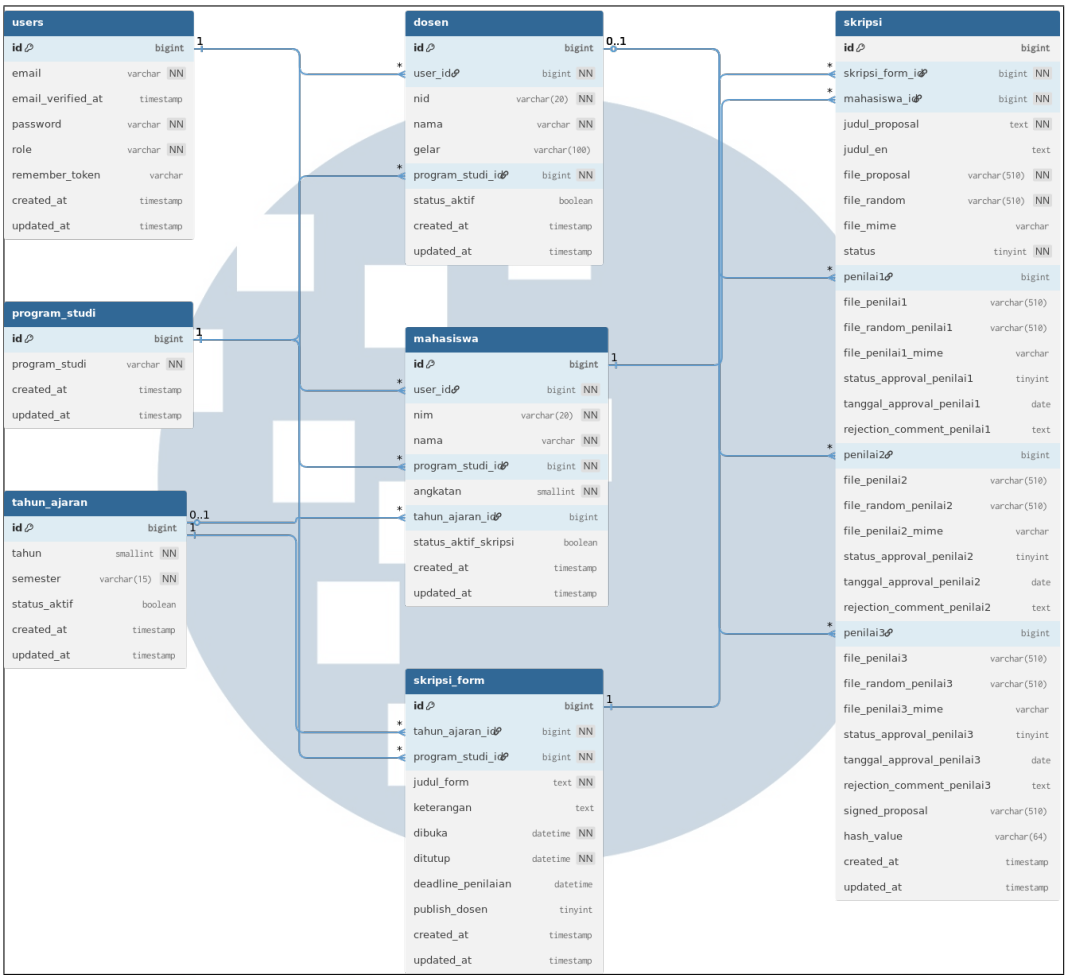
- Nama dan NIDN penandatangan
- Timestamp penandatanganan
- UUID dokumen sebagai pengenal unik
- Nilai hash SHA-512 untuk verifikasi integritas

Sistem kemudian memilih strategi penandatanganan melalui pendekatan dua lapis dengan *fallback* otomatis. Metode utama menggunakan *library* FPDI untuk memanipulasi PDF secara langsung. Pendekatan ini lebih cepat namun hanya kompatibel dengan format PDF 1.4 kebawah. Jika PDF tidak kompatibel atau proses gagal, sistem secara otomatis beralih ke metode sekunder berbasis gambar. Pada metode ini, setiap halaman PDF dikonversi menjadi gambar, kemudian sebuah PDF baru dibangun kembali menggunakan TCPDF dengan menyematkan gambar-gambar tersebut beserta *QR code* pada halaman yang telah ditentukan.

Tahap finalisasi meliputi pembuatan metadata XML yang berisi informasi tanda tangan. Metadata ini kemudian disematkan sebagai lampiran di dalam dokumen PDF. Terakhir, sistem menghitung nilai *hash* SHA-512 dari keseluruhan dokumen untuk verifikasi integritas, memperbarui *database* dengan referensi *file* yang sudah ditandatangani beserta nilai *hash*-nya, dan mengirimkan konfirmasi keberhasilan. Sistem juga dirancang dengan penanganan *error* di setiap langkah dan pencatatan yang detail untuk memudahkan proses *debugging*.

## 3.3 Perancangan *Database*

Basis data dirancang untuk mendukung semua kebutuhan fungsional sistem, mencakup pengelolaan data pengguna, dokumen, dan alur persetujuan. Struktur basis data yang jelas dan terorganisir menjadi fondasi penting untuk memastikan integritas data dan kinerja sistem yang optimal. Rancangan *Entity-Relationship Diagram* (ERD) sistem dapat dilihat pada Gambar 3.2.




Gambar 3.2. Rancangan Skema ERD

Struktur *database* pada Gambar 3.2 terdiri dari beberapa tabel utama. Tabel `users` menyimpan informasi kredensial dasar seperti `email`, `password`, dan `role` untuk membedakan hak akses antara mahasiswa, dosen, dan BIA. Tabel ini terhubung ke tabel `mahasiswa` dan `dosen` yang menyimpan data lebih spesifik. Tabel `mahasiswa` mencatat data seperti `nim`, `nama`, dan `program_studi_id`, sementara tabel `dosen` mencatat `nid`, `nama`, dan `gelar`.

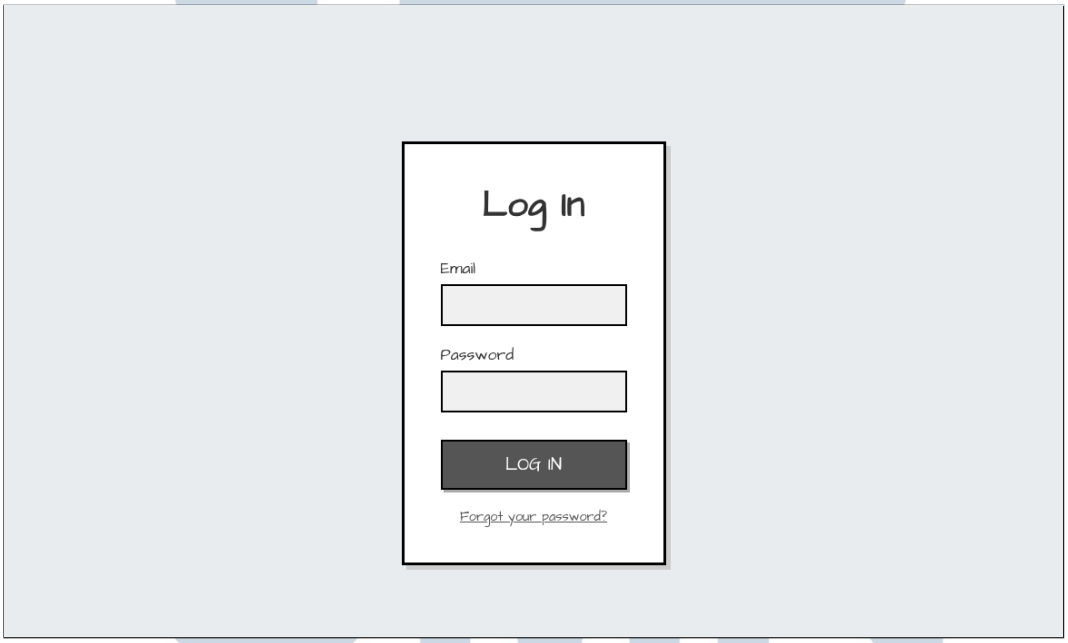
Tabel pendukung seperti `program_studi` dan `tahun_ajaran` digunakan untuk mengelola data administratif. Tabel `skripsi_form` berfungsi untuk mengatur periode pengumpulan skripsi, dengan atribut seperti tanggal `dibuka` dan `ditutup`.

Tabel `skripsi` merupakan tabel transaksi utama yang menghubungkan semua alur kerja. Tabel ini mencatat setiap dokumen yang diunggah oleh mahasiswa melalui `mahasiswa_id` dan terikat pada periode pengumpulan tertentu melalui `skripsi_form_id`. Atribut seperti `judul_proposal`,

`file_proposal`, dan `file_random` digunakan untuk menyimpan detail dan lokasi *file* dokumen. Alur persetujuan sekuensial dikelola melalui kolom `penilai1`, `penilai2`, dan `penilai3`, yang mereferensikan dosen penilai. Setiap penilai memiliki kolom statusnya sendiri, seperti `status_approval_penilai1` dan `tanggal_approval_penilai1`, untuk melacak proses persetujuan secara individual. Atribut `hash_value` menjadi kunci utama untuk verifikasi integritas dokumen, dimana nilai *hash* dari dokumen yang sudah ditandatangani disimpan untuk perbandingan di kemudian hari.

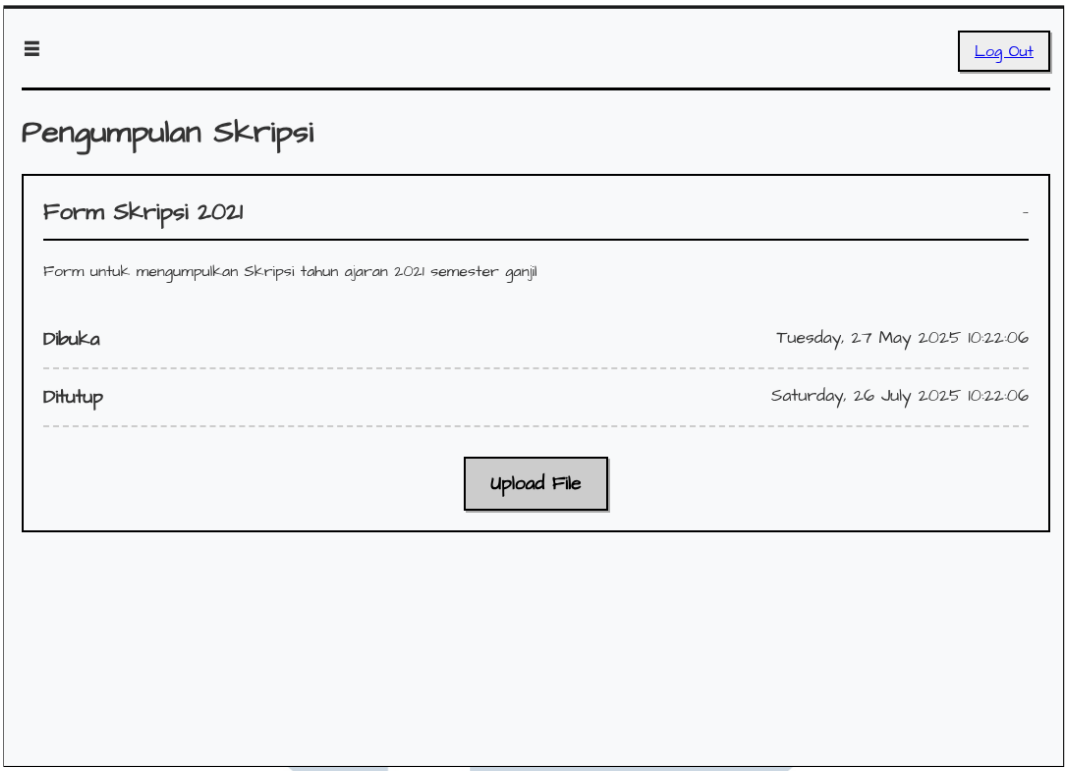
## 3.4 Desain *Prototype* Awal

Berikut merupakan desain pada prototype sistem awal.



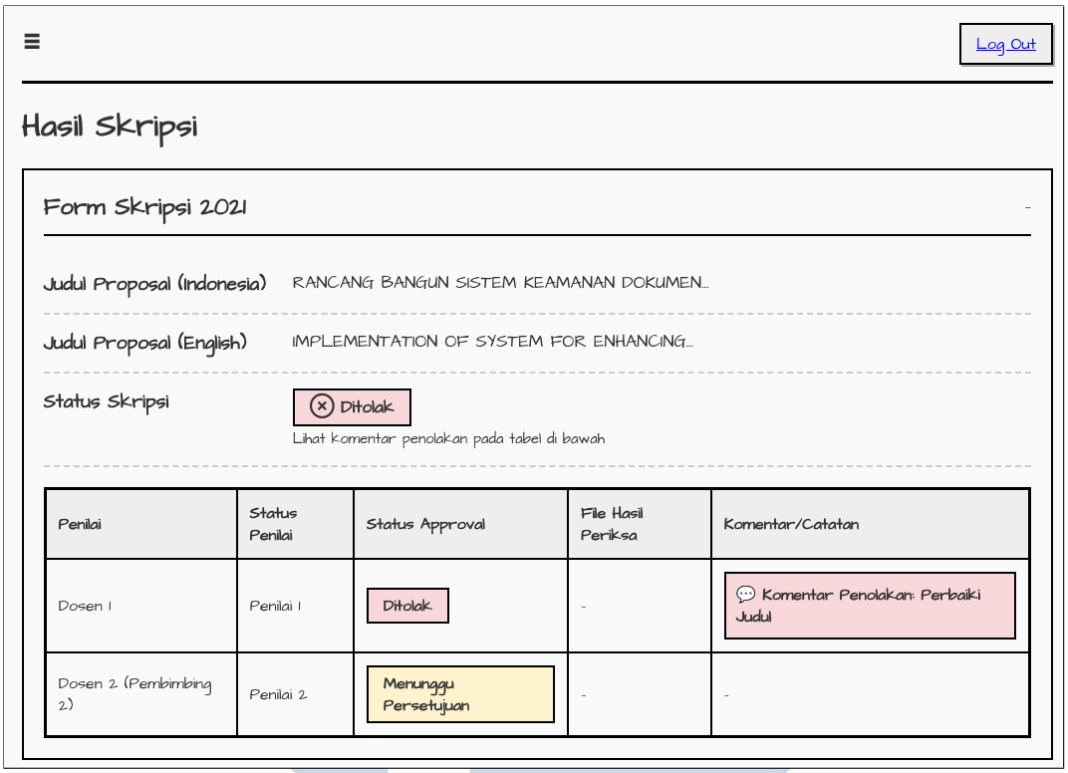

Gambar 3.3. *Prototype* Antarmuka Halaman Login

Halaman *Login* merupakan pintu awal pengguna untuk memasuki sistem seperti pada Gambar 3.3. Terhadap input *email* dan *password* yang perlu dimasukan oleh pengguna.



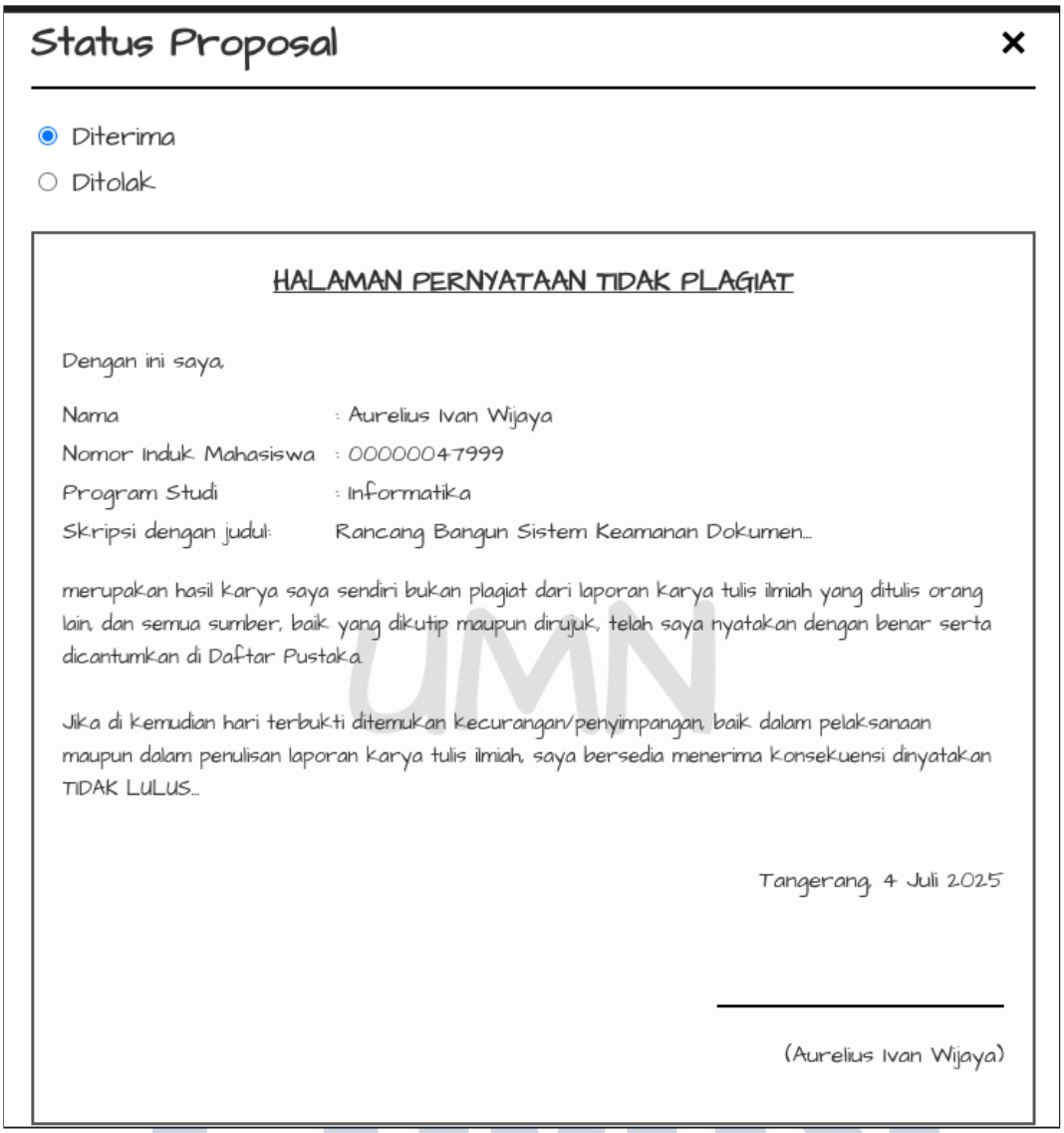

Gambar 3.4. *Prototype* Antarmuka Halaman Pengumpulan Skripsi

Mahasiswa dapat *login* dan kemudian mengumpulkan skripsi pada halaman ”Pengumpulan Skripsi” seperti pada Gambar 3.4.




Gambar 3.5. *Prototype* Antarmuka Halaman Status Skripsi

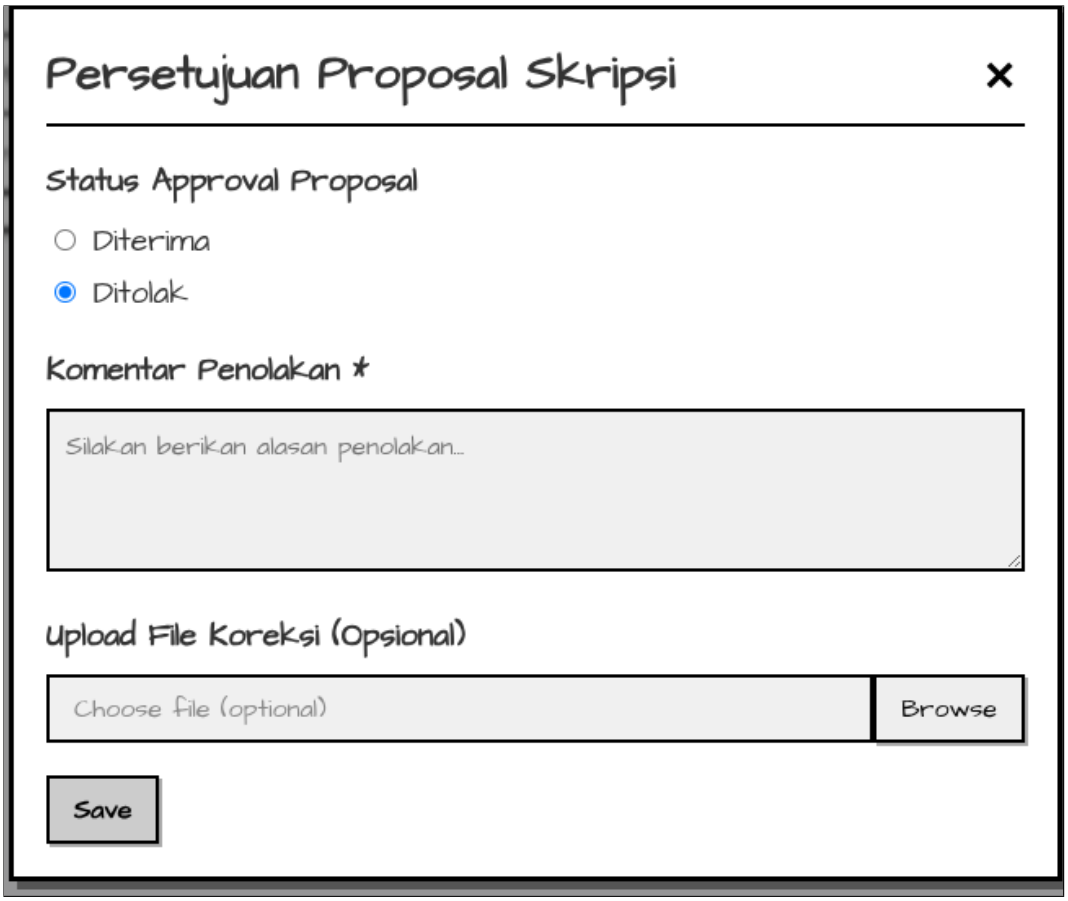
Mahasiswa kemudian dapat melihat status dan *feedback* dari skripsi yang telah dikumpulkan pada sistem seperti pada Gambar 3.5.




Gambar 3.6. *Prototype* Antarmuka Halaman *Approve* Skripsi

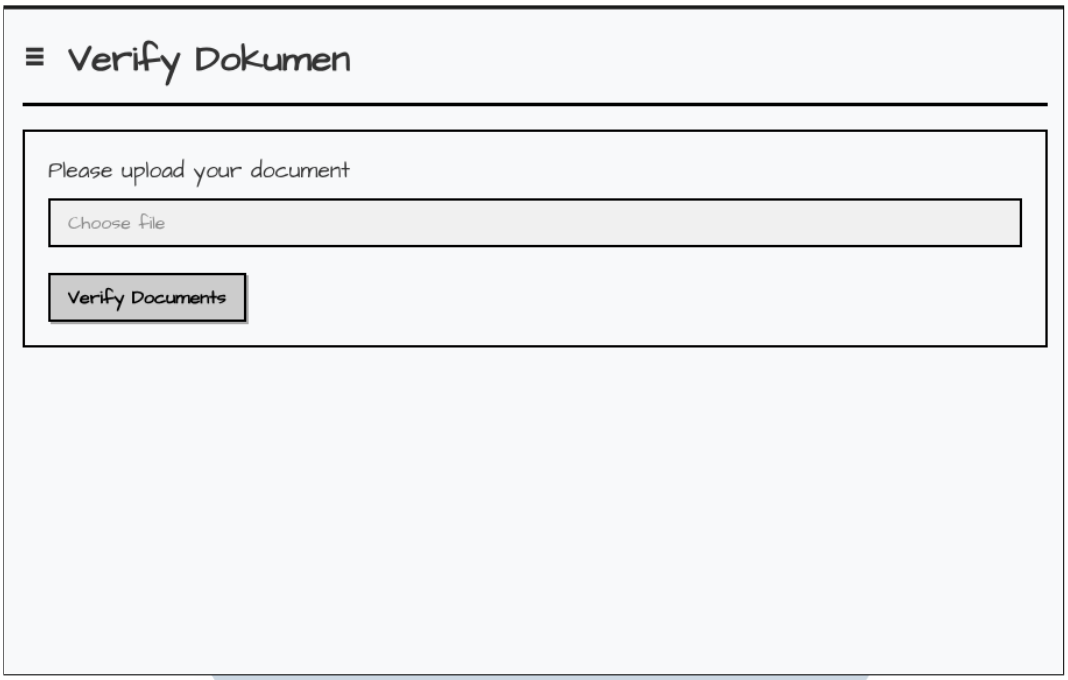
Dosen dan Kaprodi dapat melakukan *approval* pada dokumen skripsi seperti pada Gambar 3.7.

3.7.



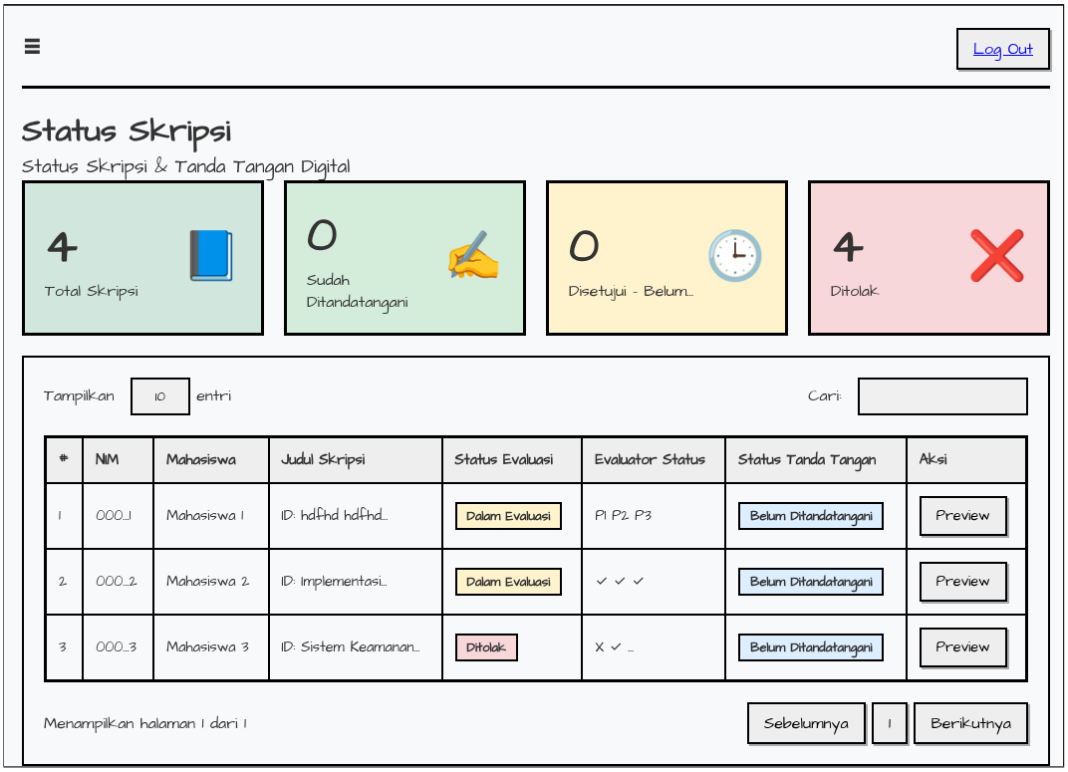

Gambar 3.7. *Prototype* Antarmuka Halaman *Reject* Skripsi

Dosen dan Kaprodi dapat melakukan penolakan pada dokumen skripsi seperti pada Gambar




Gambar 3.8. *Prototype* Antarmuka Halaman *Verify* Skripsi

BIA dan Kaprodi dapat melakukan verifikasi untuk mengecek keabsahan pada dokumen skripsi seperti pada Gambar 3.8.




Gambar 3.9. Rancangan Antarmuka Halaman Status Skripsi

BIA dan Kaprodi dapat melihat list dokumen skripsi pada halaman ”Status Skripsi” seperti pada Gambar 3.9. Pada halaman ”Status Skripsi”, BIA dan kaprodi bisa melakukan *preview*, sementara Kaprodi bisa menandatangani skripsi (pada jurusannya).

## 3.5 Pengembangan Sistem

Bagian ini menguraikan proses pengembangan sistem yang mengimplementasikan rancangan arsitektur dan antarmuka yang telah dijelaskan sebelumnya. Pengembangan sistem mencakup pemilihan teknologi dan implementasi modul-modul utama yaitu pengunggahan laporan, penandatanganan, dan verifikasi.

Sistem dikembangkan menggunakan *tech stack* yang dipilih berdasarkan kebutuhan fungsionalitas, skalabilitas, dan kemudahan pemeliharaan di UMN. *Environment* pengembangan utama yang dipilih sebagai berikut:

- *Framework Backend*: PHP-Laravel digunakan untuk membangun logika bisnis dan API.
- *Library* PDF: *setasign/fpdi* dan *tecnickcom/tcpdf* digunakan untuk manipulasi dan pembuatan PDF. *spatie/pdf-to-image* digunakan sebagai metode alternatif jika *fpdi* gagal memproses PDF.
- *Library QR Code*: *chillerlan/php-qrcode* untuk pembuatan *QR Code*.
- *Library* Ekstraksi: *smalot/pdfparser* untuk membaca lampiran *file* yang tersemat pada dokumen PDF saat verifikasi.
- *Database*: MySQL v8.0 digunakan untuk penyimpanan data dokumen dan metadata.

Sebelum dilakukan implementasi modul, dilakukan *generate* RSA *key pair* (*one-time*) pada terminal.

```
# Generate system-wide RSA key pair
openssl genrsa -out storage/app/keys/system_private.key 2048
# Creating the Public Certificate
openssl req -new -x509 -key storage/app/keys/system_private.key -
    out storage/app/keys/system_cert.crt -days 3650
```

Kode 3.1: Contoh Implementasi Penanaman QR Code (PHP-Laravel)

Dilakukan pembuatan RSA *key pair* satu kali pada terminal seperti pada Kode 3.1. *Private key* yang dihasilkan akan digunakan oleh sistem untuk menandatangani dokumen, sementara *public key* yang bersesuaian disematkan dalam sebuah *self-signed certificate*. Sertifikat ini berfungsi sebagai identitas kriptografis sistem dan akan digunakan dalam proses verifikasi tanda tangan digital nantinya.

Modul pengunggahan laporan skripsi (mahasiswa) adalah pintu awal masuknya dokumen PDF skripsi.

```
// ... namespace dan uses
class LaporanAkhirController extends Controller
{
    /**
     * Store uploaded thesis PDF file
     */
    public function store(Request $request)
    {
        $request->validate([
            'revisi_proposal' => ['required'],
            'file' => ['required', 'file', 'mimes:pdf', 'max:30720'],
            'reupload' => ['required'],
            'laporan_id' => ['sometimes', 'required_if:reupload,1'],
            'id_form' => ['required'],
        ]);

        try {
            DB::transaction(function () use ($request) {
                $active = TahunAjaran::where('status_aktif', 1)->first();
                $user = Mahasiswa::where('user_id', Auth::user()->id)->firstOrFail();
                $form = LaporanAkhirForm::where('uuid', $request->id_form)->firstOrFail();
                $revisi = RevisiProposal::where('uuid', $request->revisi_proposal)->firstOrFail();

                $file = $request->file('file');
                $clientName = $file->getClientOriginalName();
                $fileNameRandom = date('YmdHis') . '_' . $file->hashName();

                // Create thesis record in database
```

```
                LaporanAkhir::create([
                    'laporan_akhir_form_id' => $form->id,
                    'revisi_proposal_id' => $revisi->id,
                    'mahasiswa_id' => $user->id,
                    'judul_laporan' => $revisi->judul_revisi_proposal,
                    'file_laporan' => $clientName,
                    'file_laporan_random' => $fileNameRandom,
                    'status' => 1,
                    'pembimbing1' => $pembimbingPertama->id,
                    'pembimbing2' => $pembimbingKedua->id,
                    'status_approval_pembimbing1' => 2,
                    'status_approval_pembimbing2' => 2,
                    'status_approval_kaprodi' => 2,
                ]);

                // Store PDF file to secure storage
                $file->storeAs('uploads/laporan-akhir', $fileNameRandom);
            });

            return redirect()->back()->with('success', 'Laporan berhasil diunggah!');
        } catch (Exception $e) {
            return redirect()->back()->with('error', 'Gagal mengunggah: ' . $e->getMessage());
        }
    }
}

// QR Code Implementation in SignatureController
// ... namespace dan uses
class SignatureController extends Controller
{
    /**
     * Sign thesis with QR code
     */
    public function signThesis(Request $request)
    {
        $validatedData = $request->validate([
            'id' => 'required|integer|exists:proposal_skripsi,id',
            'x' => 'required|numeric',
            'y' => 'required|numeric',
```

```
            'width' => 'required|numeric|min:1',
            'height' => 'required|numeric|min:1',
            'page_number' => 'required|integer|min:1',
        ]);

        // Retrieve proposal and lecturer data
        $proposal = ProposalSkripsi::with('mahasiswa')->findOrFail($validatedData['id']);
        $lecturer = Dosen::where('user_id', Auth::user()->id)->first();

        if (!$lecturer) {
            return redirect()->back()->with('error', 'Profil dosen tidak ditemukan.');
        }

        // Generate QR Code with signature data
        $qrData = "Signed by $lecturer->nama ($lecturer->nid) \nDate: " . now()->toDateTimeString();

        $options = new QROptions([
            'outputType' => QRCode::OUTPUT_IMAGE_PNG,
            'eccLevel' => QRCode::ECC_L,
            'scale' => 6,
            'imageBase64' => false,
        ]);

        $qrcode = new QRCode($options);
        $tempDir = storage_path('app/temp/');
        $qrCodePath = $tempDir . 'qr_code_' . time() . '.png';

        try {
            $qrcode->render($qrData, $qrCodePath);
            Log::info("QR code generated at: {$qrCodePath}");
        } catch (\Exception $e) {
            Log::error("Failed to generate QR code: " . $e->getMessage());
            return redirect()->back()->with('error', 'Gagal membuat kode QR.');
        }

        // Embed QR code into PDF
        return $this->embedQRCodeIntoPDF($validatedData, $proposal
```

```
, $qrCodePath);
105     }
106
107     /**
108      * Embed QR code into PDF document
109      */
110     private function embedQRCodeIntoPDF($validatedData, $proposal,
        $qrCodePath)
111     {
112         $fileName = $proposal->file_proposal_random;
113         $originalPdfPath = storage_path('app/uploads/proposal/' .
    $fileName);
114
115         // Initialize FPDI PDF processor
116         $pdf = new Fpdi();
117         $pageCount = $pdf->setSourceFile($originalPdfPath);
118
119         // Process each page
120         for ($pageNo = 1; $pageNo <= $pageCount; $pageNo++) {
121             $templateId = $pdf->importPage($pageNo);
122             $size = $pdf->getTemplateSize($templateId);
123
124             $pdf->AddPage($size['orientation'], [$size['width'],
    $size['height']]);
125             $pdf->useTemplate($templateId);
126
127             // Add QR code to specified page
128             if ($validatedData['page_number'] == $pageNo) {
129                 $x = floatval($validatedData['x']) / floatval(
    $validatedData['width']) * $size['width'] - 3;
130                 $y = floatval($validatedData['y']) / floatval(
    $validatedData['height']) * $size['height'] - 1;
131
132                 // Embed QR code image
133                 $qrWidth = 20; // mm
134                 $qrHeight = 20; // mm
135                 $pdf->Image($qrCodePath, $x, $y, $qrWidth,
    $qrHeight);
136
137                 Log::info("QR code embedded on page {$pageNo} at
    ({$x}mm, {$y}mm)");
138             }
139         }
```

```
140
141         // Save signed PDF
142         $pdf->Output($originalPdfPath, 'F');
143
144         // Generate metadata XML for authenticity
145         $this->embedMetadataXML($proposal, $originalPdfPath);
146
147         // Clean up temporary QR code file
148         if (file_exists($qrCodePath)) {
149             unlink($qrCodePath);
150         }
151
152         // Update database with signed status
153         $proposal->signed_proposal = $fileName;
154         $proposal->hash_value = hash('sha512', file_get_contents(
    $originalPdfPath));
155         $proposal->save();
156
157         return redirect()->back()->with('success', 'Proposal
    berhasil ditandatangani.');
158     }
159
160     /**
161      * Generate and embed XML metadata for document authenticity
162      */
163     private function embedMetadataXML($proposal, $originalPdfPath)
164     {
165         $data = [
166             'UUID' => $proposal->uuid,
167             'Tipe_Laporan' => 'Skripsi',
168             'Judul_Laporan' => $proposal->judul_proposal,
169             'Nama_Mahasiswa' => $proposal->mahasiswa->nama,
170             'NIM' => $proposal->mahasiswa->nim,
171             'Tanggal_Tanda_Tangan' => now()->toDateTimeString(),
172         ];
173
174         $xmlContent = $this->generateSecureXML($data);
175         $xmlPath = storage_path('app/temp/metadata_' . time() . '.
    xml');
176         file_put_contents($xmlPath, $xmlContent);
177
178         // Embed XML as attachment in PDF
179         $this->embedFilesInExistingPdf($originalPdfPath,
```

```
    $originalPdfPath, [$xmlPath]);

        // Clean up temporary XML file
        unlink($xmlPath);
    }
}
```

Kode 3.2: Contoh Implementasi Penanaman QR Code (PHP-Laravel)

Kode 3.2 Modul pengunggahan laporan skripsi mahasiswa merupakan komponen fundamental dalam sistem manajemen tugas akhir yang berfungsi sebagai pintu masuk utama dokumen PDF skripsi ke dalam sistem. Implementasi ini menggunakan framework Laravel dengan pendekatan yang komprehensif dalam menangani validasi, penyimpanan, dan digitalisasi dokumen. Proses pengunggahan dimulai dengan validasi ketat terhadap *file* yang diunggah. Sistem kemudian menghasilkan nama file acak menggunakan *timestamp* dan *hash* untuk mencegah konflik penamaan dan meningkatkan keamanan penyimpanan. Implementasi tanda tangan digital menggunakan teknologi QR Code yang terintegrasi dengan *metadata* dokumen. Sistem menggunakan library `chillerlan/qrcode` untuk menghasilkan QR Code yang berisi informasi penandatangan, termasuk nama dosen, NIDN, dan timestamp penandatanganan. QR *code* ini kemudian ditanamkan ke dalam dokumen PDF menggunakan *library* FPDI (PDF manipulation). Posisi QR *code* ditentukan secara dinamis berdasarkan koordinat yang dipilih pengguna melalui antarmuka web oleh penandatangan. Sistem melakukan konversi koordinat dari *pixel* ke milimeter untuk memastikan presisi penempatan dalam dokumen PDF. Proses ini disertai dengan pembuatan *metadata* XML yang berisi informasi autentisitas dokumen, termasuk UUID unik, judul laporan, data mahasiswa, dan *hash* SHA-512 untuk verifikasi integritas. Keamanan implementasi diperkuat dengan sanitasi konten PDF, validasi struktur file, dan pembersihan metadata berbahaya. Sistem juga mengimplementasikan mekanisme *fallback* untuk menangani berbagai format PDF yang mungkin tidak kompatibel dengan *processor* standar. Seluruh proses dilakukan dalam transaksi *database* untuk memastikan konsistensi data dan integritas sistem. Modul penandatanganan (dosen) adalah inti dari sistem yang dibangun. Modul ini bertanggung jawab untuk melakukan sanitasi, dan menanamkan QR *code* dan melakukan proses penyematan XML XAdES. Rangkaian proses dapat dilihat pada urutan kode berikut.

```
<?php
// ... namespace dan uses
```

```
class SignatureController extends Controller
{
    /**
     * XAdES Configuration following ETSI TS 101 903
     */
    private const XADES_CONFIG = [
        'enabled' => true,
        'algorithm' => OPENSSL_ALGO_SHA256,
        'signature_method' => 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha256',
        'canonicalization_method' => 'http://www.w3.org/2001/10/xml-exc-c14n#',
        'digest_method' => 'http://www.w3.org/2001/04/xmlenc#sha256',
        'namespaces' => [
            'ds' => 'http://www.w3.org/2000/09/xmldsig#',
            'xades' => 'http://uri.etsi.org/01903/v1.3.2#',
        ],
    ];

    public function signThesis(Request $request)
    {
        // 1. Validasi request dan ambil data proposal
        $validatedData = $request->validate([
            'id' => 'required|integer', 'x' => 'required|numeric',
            'y' => 'required|numeric', 'page_number' => 'required|integer'
        ]);
        $proposal = ProposalSkripsi::findOrFail($validatedData['id']);
        $lecturer = Dosen::where('user_id', Auth::user()->id)->first();

        // 2. Generate QR Code
        $qrData = "Signed by {$lecturer->nama}\nDate: " . now()->toDateTimeString();
        $options = new QROptions(['outputType' => QRCode::OUTPUT_IMAGE_PNG]);
        $qrCodePath = storage_path('app/temp/qr_code.png');
        (new QRCode($options))->render($qrData, $qrCodePath);

        // 3. Inisialisasi FPDI untuk memodifikasi PDF
        $pdf = new Fpdi();
```

```
38          $originalPdfPath = storage_path('app/uploads/proposal/' .
    $proposal->file_proposal_random);
39          $pageCount = $pdf->setSourceFile($originalPdfPath);
40
41          // 4. Loop semua halaman dan tambahkan QR Code
42          for ($pageNo = 1; $pageNo <= $pageCount; $pageNo++) {
43              $templateId = $pdf->importPage($pageNo);
44              $size = $pdf->getTemplateSize($templateId);
45              $pdf->AddPage($size['orientation'], [$size['width'],
    $size['height']]);
46              $pdf->useTemplate($templateId);
47
48              if ($pageNo == $validatedData['page_number']) {
49                  $x = floatval($validatedData['x']) / floatval(
    $validatedData['width']) * $size['width'] - 3;
50                  $y = floatval($validatedData['y']) / floatval(
    $validatedData['height']) * $size['height'] - 1;
51                  $pdf->Image($qrCodePath, $x, $y, 20, 20, 'PNG');
52              }
53          }
54
55          // 5. Simpan PDF yang sudah dimodifikasi
56          $pdf->Output($originalPdfPath, 'F');
57          unlink($qrCodePath); // Hapus file QR sementara
58
59          // 6. Generate metadata dengan XAdES signature
60          $data = [
61              'UUID' => $this->sanitizeXmlTextContent($proposal->
    uuid ?? ''),
62              'Tipe_Laporan' => 'Skripsi',
63              'Judul_Laporan' => $this->sanitizeXmlTextContent(
    $proposal->judul_proposal ?? ''),
64              'Judul_Laporan_EN' => $this->sanitizeXmlTextContent(
    $proposal->judul_proposal_en ?? ''),
65              'Prodi' => 'Teknik Informatika',
66              'Tahun' => date('Y'),
67              'Nama_Mahasiswa' => $this->sanitizeXmlTextContent(
    $proposal->mahasiswa->nama ?? ''),
68              'NIM' => $this->sanitizeXmlTextContent($proposal->
    mahasiswa->nim ?? ''),
69              'Dosen_Pembimbing_1__Nama' => $this->
    sanitizeXmlTextContent($proposal->penilaiPertama->nama ?? ''),
70              'Dosen_Pembimbing_1__NIDN' => $this->
```

```
    sanitizeXmlTextContent($proposal->penilaiPertama->nid ?? ''),
        ];

        // Generate XML with XAdES signature
        $xmlContent = self::generateXML($data);

        if ($xmlContent !== false) {
            // Create temporary XML file
            $xmlPath = storage_path('app/temp/data_' . time() . '_
' . uniqid() . '.xml');
            file_put_contents($xmlPath, $xmlContent);

            // 7. Embed XML in PDF
            self::embedFilesInExistingPdf(
                $originalPdfPath,
                $originalPdfPath,
                [$xmlPath]
            );

            // Clean up temporary file
            if (file_exists($xmlPath)) {
                unlink($xmlPath);
            }
        }

        // 8. Update hash in database
        $proposal->signed_proposal = $proposal->
file_proposal_random;
        $proposal->hash_value = substr(hash('sha512',
file_get_contents($originalPdfPath)), 0, 64);
        $proposal->save();

        return redirect()->back()->with('success', 'Dokumen
berhasil ditandatangani dengan signature XAdES.');
    }

    /**
     * Generate secure XML with integrated XAdES signature
     */
    private static function generateXML($object): bool|string
    {
        try {
            // Convert object to array
```

```
            $arrayData = json_decode(json_encode($object), true);

            // Create XML document with secure settings
            $xml = new \DOMDocument('1.0', 'UTF-8');
            $xml->formatOutput = true;

            // Create root element with ID for signature reference
            $rootElement = $xml->createElement('root');
            $xml->appendChild($rootElement);
            $rootElement->setAttribute('Id', 'DocumentRoot');

            // Add data to XML using secure method
            self::arrayToXMLSecureDOM($arrayData, $rootElement,
$xml);

            // Apply XAdES signature if configured
            if (self::XADES_CONFIG['enabled']) {
                // Load private key and certificate
                $privateKeyPath = config('app.xades.
private_key_path');
                $certificatePath = config('app.xades.
certificate_path');

                if (file_exists($privateKeyPath) && file_exists(
$certificatePath)) {
                    $privateKey = file_get_contents(
$privateKeyPath);
                    $certificate = file_get_contents(
$certificatePath);

                    // Generate unique IDs
                    $signatureId = 'Signature-' . uniqid();
                    $dataObjectId = 'DocumentRoot';
                    $signedPropertiesId = 'SignedProperties-' .
uniqid();

                    // Create XAdES signature instance
                    $instance = new self();
                    $signature = $instance->createETSISignature(
                        $xml,
                        $signatureId,
                        $dataObjectId,
                        $signedPropertiesId,
```

```
                        $privateKey,
                        $certificate
                    );

                    // Add signature to document
                    $rootElement->appendChild($signature);
                }
            }

            return $xml->saveXML();
        } catch (\Exception $e) {
            Log::error("Error generating XML with XAdES: " . $e->
    getMessage());
            return false;
        }
    }

    private function sanitizeXmlTextContent(string $content):
    string
    {
        // Remove control characters
        $content = preg_replace('/[\x00-\x08\x0B-\x0C\x0E-\x1F\x7F
    ]/', '', $content);

        // HTML encode to prevent XML injection
        $content = htmlspecialchars($content, ENT_QUOTES |
    ENT_XML1, 'UTF-8');

        return $content;
    }
}
```

Kode 3.3: Contoh Implementasi Penanaman QR Code (PHP-Laravel)

Kode 3.3 menunjukan bagaimana sistem memproses *request* tanda tangan dari dosen, dengan menanamkan QR *code* yang berisi informasi penanda tangan ke dalam dokumen PDF. Proses ini menggunakan *library FPDI* untuk memodifikasi *file* PDF yang ada, menempatkan gambar QR *code* pada posisi yang ditentukan, dan menyimpan hasilnya. Setelah itu, *metadata* XML disematkan dan nilai *hash* SHA-512 dari dokumen final dihitung dan disimpan untuk proses verifikasi nanti.

Modul verifikasi dirancang agar pengguna (BIA) dapat dengan mudah memeriksa keaslian dan integritas dokumen. Prosesnya adalah dengan

membandingkan *hash* dari dokumen yang diunggah dengan *hash* yang tersimpan di dalam *database*.

```php
<?php
// ... namespace and uses
class SignatureController extends Controller
{
    public function uploadVerifyThesis(Request $request)
    {
        // 1. Ambil konten PDF yang diunggah
        $pdfContent = $request->file('file')->get();

        // 2. Ekstrak file XML yang tersemat menggunakan Smalot/PdfParser
        $xmlContent = $this->extractEmbeddedFiles($pdfContent);
        if (!$xmlContent) {
            return view('...', ['status' => 'error', 'error' => 'Dokumen tidak valid atau rusak.']);
        }

        // 3. Parse XML untuk mendapatkan UUID
        $content = simplexml_load_string($xmlContent);
        $parsedArray = json_decode(json_encode($content), true);
        $uuid = $parsedArray['UUID'];

        // 4. Hitung hash dari PDF yang diunggah
        $calculatedHash = substr(hash('sha512', $pdfContent), 0, 64);

        // 5. Ambil hash asli dari database menggunakan UUID
        $proposal = ProposalSkripsi::where('uuid', $uuid)->first();
        if (!$proposal) {
            return view('...', ['status' => 'error', 'error' => 'Data proposal tidak ditemukan.']);
        }
        $storedHash = $proposal->hash_value;

        // 6. Bandingkan hash
        if ($calculatedHash !== $storedHash) {
            return view('...', ['status' => 'failed', 'error' => 'Hash tidak cocok, dokumen mungkin telah diubah.']);
        }

```

```
        // 7. Jika cocok, tampilkan data dari XML
        return view('...', ['status' => 'success', 'data' =>
    $parsedArray]);
    }
}
```

Kode 3.4: Contoh Implementasi Verifikasi Dokumen (PHP-Laravel)

Kode 3.4 menunjukkan bagaimana sistem memvalidasi dokumen yang telah ditandatangani. Sistem pertama-tama mengekstrak *file* XML yang tersemat untuk mendapatkan UUID dokumen. Kemudian, sistem menghitung *hash* SHA-512 dari *file* PDF yang diunggah dan membandingkannya dengan *hash* yang tersimpan di *database*. Jika kedua *hash* cocok, dokumen dinyatakan valid.

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
              xmlns:xades="http://uri.etsi.org/01903/v1.3.2#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR
    /2001/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/
    xmldsig-more#ecdsa-sha512"/>
    <ds:Reference URI="">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig
    #enveloped-signature"/>
        <ds:Transform Algorithm="http://uri.etsi.org/01903/v1.3.2#
    detached-xades"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc
    #sha512"/>
      <ds:DigestValue>HASH_DOKUMEN_PDF</ds:DigestValue>
    </ds:Reference>
    <ds:Reference Type="http://www.w3.org/2000/09/xmldsig#
    SignatureProperties"
                  URI="#SignatureProperties">
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc
    #sha512"/>
      <ds:DigestValue>HASH_SIGNATURE_PROPERTIES</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>NILAI_TANDA_TANGAN_DIGITAL</ds:SignatureValue
    >
  <ds:KeyInfo>
    <ds:X509Data>
```

```
      <ds:X509Certificate>SERTIFIKAT_PENANDA_TANGAN</ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
  <ds:Object Id="SignatureProperties">
    <xades:QualifyingProperties Target="SIGNATURE_ID">
      <xades:SignedProperties Id="SignedProperties">
        <xades:SignedSignatureProperties>
          <xades:SigningTime>WAKTU_PENANDATANGANAN</xades:SigningTime>
          <xades:SigningCertificate>
            <xades:Cert>
              <xades:CertDigest>
                <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
                <ds:DigestValue>
                HASH_SERTIFIKAT
                </ds:DigestValue>
              </xades:CertDigest>
              <xades:IssuerSerial>
                <ds:X509IssuerName>
                NAMA_ISSUER
                </ds:X509IssuerName>
                <ds:X509SerialNumber>
                NOMOR_SERI_SERTIFIKAT
                </ds:X509SerialNumber>
              </xades:IssuerSerial>
            </xades:Cert>
          </xades:SigningCertificate>
        </xades:SignedSignatureProperties>
        <xades:SignedDataObjectProperties>
          <xades:DataObjectFormat ObjectReference="">
            <xades:MimeType>application/pdf</xades:MimeType>
          </xades:DataObjectFormat>
        </xades:SignedDataObjectProperties>
      </xades:SignedProperties>
      <xades:UnsignedProperties>
        <xades:UnsignedSignatureProperties>
          <xades:SignatureTimeStamp>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          </xades:SignatureTimeStamp>
        </xades:UnsignedSignatureProperties>
```

```
      </xades:UnsignedProperties>
    </xades:QualifyingProperties>
  </ds:Object>
</ds:Signature>
```

Kode 3.5: Contoh Struktur Metadata XML yang Disematkan

Contoh Kode 3.5 menunjukkan struktur dasar XML yang digunakan. Setiap elemen berisi data kunci yang relevan dengan dokumen, seperti UUID sebagai pengenal unik, judul, informasi mahasiswa, dan detail pembimbing. Informasi ini yang akan ditampilkan kepada verifikator setelah proses validasi *hash* berhasil.

## 3.6 Verifikasi Keaslian Tanda Tangan

Verifikasi tanda tangan dimulai dengan Kaprodi atau petugas BIA mengunggah kembali dokumen PDF yang sudah ditandatangani pada halaman verifikasi. Sistem mengekstrak lampiran XML XAdES dari berkas PDF kemudian mem-parsing elemen-elemen kunci dari XML tersebut, antara lain UUID dokumen dan nilai hash sertifikat penandatangan. Setelah XML berhasil diambil, sistem menghitung ulang hash SHA-512 dari keseluruhan konten PDF yang diunggah dan membandingkannya dengan hash yang telah tersimpan di basis data berdasarkan UUID. Jika kedua nilai hash tersebut sama, sistem membaca metadata dalam XML untuk menampilkan informasi penandatangan, seperti nama dosen, NIDN, dan timestamp penandatanganan, sebagai bukti keaslian. Sebaliknya, apabila nilai hash tidak cocok, verifikasi ditolak dengan status "*Failed*" dan pesan "*Hash* tidak cocok, dokumen mungkin telah diubah" ditampilkan. Dengan mekanisme ini, sistem memastikan bahwa dokumen tidak mengalami modifikasi setelah ditandatangani dan tanda tangan digital XAdES benar-benar valid sebelum dokumen dianggap sah.

## 3.7 *User Acceptance Testing* (UAT)

UAT dilakukan untuk memastikan sistem yang dibangun telah memenuhi kebutuhan fungsional pengguna dan dapat diterima untuk digunakan dalam alur kerja yang sesungguhnya.

Pengujian UAT melibatkan tiga partisipan yang merepresentasikan setiap peran utama dalam sistem:

- 1 (satu) orang Alumni Universitas Multimedia Nusantara Program Studi Informatika 2025.

- 1 (satu) orang Dosen Program Studi Informatika.
- 1 (satu) orang perwakilan dari Biro Informasi Akademik (BIA).

Perlu diakui bahwa jumlah partisipan yang terbatas ini menjadi sebuah batasan dalam penelitian. Hasil dari pengujian UAT dengan jumlah sampel dimaksudkan untuk memberikan indikasi awal mengenai fungsionalitas dan penerimaan sistem, namun tidak dapat dianggap representatif secara statistik untuk seluruh populasi pengguna di Universitas Multimedia Nusantara. Umpan balik yang diterima bersifat kualitatif dan berfokus pada validasi *workflow* inti.

Setiap partisipan diberikan serangkaian skenario tugas yang mencerminkan *workflow* nyata dari sistem, mencakup kasus penggunaan fungsional dan kasus negatif:

1. Peran Mahasiswa:
    - *Login* ke sistem menggunakan akun mahasiswa.
    - Mengunggah *file* skripsi dengan format PDF.
    - Melihat riwayat dan status dokumen yang telah diunggah.
    - *Logout* dari sistem.
2. Peran Dosen:
    - *Login* ke sistem menggunakan akun dosen.
    - Melihat daftar dokumen mahasiswa yang menunggu persetujuan.
    - Membuka (*preview*) dan meninjau salah satu dokumen dari daftar.
    - Menyetujui dan memberikan tanda tangan digital pada dokumen.
    - Menolak dokumen skripsi dan memberikan revisi.
    - *Logout* dari sistem.
3. Peran Kepala Program Studi (Kaprodi):
    - *Login* ke sistem menggunakan akun Kaprodi.
    - Melihat daftar dokumen mahasiswa yang menunggu persetujuan.
    - Membuka (*preview*) dan meninjau salah satu dokumen dari daftar.
    - Menyetujui dan memberikan tanda tangan digital pada dokumen.
    - Menolak dokumen skripsi dan memberikan revisi.

- Melakukan verifikasi dokumen melalui halaman verifikasi, untuk menguji keabsahannya (menggunakan dokumen valid, yang telah diubah, dan yang tidak ditandatangani).
- *Logout* dari sistem.

4. Peran Admin (BIA):
    - *Login* ke sistem dengan akun administrator.
    - Melihat daftar akun mahasiswa.
    - Menambahkan akun mahasiswa baru.
    - Melihat daftar akun dosen.
    - Menambahkan akun dosen baru.
    - Mengganti periode tahun ajaran yang sedang aktif di sistem.
    - Memantau status pengumpulan skripsi mahasiswa secara keseluruhan.
    - *Logout* dari sistem.

## 3.8 Evaluasi Keamanan Menggunakan CVSS

Evaluasi keamanan sistem dilakukan secara kuantitatif untuk mengukur tingkat risiko dari berbagai potensi kerentanan. Analisis ini menggunakan *framework Common Vulnerability Scoring System* (CVSS) dengan merujuk pada sepuluh risiko keamanan paling kritis yang diidentifikasi oleh *Open Web Application Security Project* (OWASP) *Top* 10. Setiap skenario ancaman dievaluasi untuk menghasilkan vektor CVSS yang terstandarisasi.

Berikut adalah daftar skenario serangan yang akan dianalisis dalam penelitian ini, beserta skema CVSS yang sesuai:

- A01:2021 - *Broken Access Control*: Skenario: Seorang penyerang dengan akun mahasiswa yang valid mencoba untuk mengakses langsung sebuah *endpoint* API yang hanya diperuntukkan bagi dosen, misalnya, `/api/proposal/convert/{filename}`. Tujuannya adalah untuk melewati alur persetujuan normal dan menyetujui skripsinya sendiri atau skripsi mahasiswa lain secara tidak sah. Serangan ini dilakukan dengan memanipulasi permintaan URL, dengan harapan sistem tidak menerapkan verifikasi peran (*role*) yang cukup ketat pada *endpoint* tersebut.

- A02:2021 - *Cryptographic Failures*: Skenario: *Private key* yang digunakan oleh sistem untuk menandatangani dokumen XAdES disimpan dengan tidak aman di dalam direktori *web root* yang dapat diakses publik, atau di-*hardcode* di dalam *source code* yang tersimpan di repositori publik. Seorang penyerang memindai repositori atau direktori server dan menemukan *private key* tersebut. Dengan *key* ini, penyerang dapat secara mandiri menandatangani dokumen skripsi palsu manapun, dan tanda tangan tersebut akan dianggap valid oleh sistem verifikasi, karena dibuat dengan *key* kriptografi yang sah.

- A03:2021 - Injection: Skenario: Seorang pelaku (misalnya, mahasiswa yang tidak bertanggung jawab) ingin mendapatkan pengesahan untuk dokumen skripsi yang isinya berbeda dari yang disetujui dosen. Pelaku tersebut kemudian menyisipkan sebuah struktur XML buatan di dalam dokumen PDF yang akan diunggah.

  Tujuannya adalah untuk mengelabui sistem agar menandatangani *hash* dari dokumen palsu, bukan dokumen asli yang dilihat oleh dosen.

  Muatan XML yang akan digunakan pada analisis XML *injection* tersebut dapat dilihat pada Kode 3.6. Pelaku dengan *role* mahasiswa mencoba meng-*embed* XML pada dokumen skripsi yang akan diunggah dengan muatan XML yang berisi XML yang menyerupai dokumen yang sudah di tandatangani.

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xades="http://uri.etsi.org/01903/v1.3.2#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n
    -20010315"/>
    <ds:SignatureMethod
    Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-
    sha512"/>
    <ds:Reference URI="">
      <ds:Transforms>
        <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
    signature"/>
        <ds:Transform
Algorithm="http://uri.etsi.org/01903/v1.3.2#detached-xades"/>
      </ds:Transforms>
      <ds:DigestMethod
```

```
Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
      <ds:DigestValue>
f4b5c6d7e8f9a0b1c2d3e4f5a6b7c8d9e0
f1a2b3c4d5e6f7a8b9c0d1e2f3a4b5
      </ds:DigestValue>
    </ds:Reference>
    <ds:Reference
Type="http://www.w3.org/2000/09/xmldsig#SignatureProperties"
URI="#SignatureProperties">
      <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
      <ds:DigestValue>
a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7
f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2
      </ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
MHcCAQEEICpbXztP5tbkOdE7IrQ5yD
UNFZAlBIpYFw2Eft8lRPDxoAoGCCqGSM49
AwEHoUQDQgAEhV5qZ9c8odRNiQrAAzTqPT
d2bC2Jei0oR5JqKhOY0npRMQRZ0ptA
iRQFgh9/Y8bG+EWr9KZz5FLzDN9cHFzKuw==
  </ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
MIIBhDCCASmgAwIBAgIUeFbgUWy8WybLn4
F0nTwzN34Bvj8wCgYIKoZIzj0EAwIw
FTETMBEGA1UEAwwKQ29udG9oIFNpZ24wHh
cNMjUwNzAxMTUxMTAwWhcNMjYwNzAx
MTUxMTAwWjAVMRMwEQYDVQQDDApDb250b2
ggU2lnbjBZMBMGByqGSM49AgEGCCqG
SM49AwEHA0IABIVeamfXPKHUTYkKwAM06j
03dmwtinotKEeSaihTmNJ6UTEEWdKb
QIkUBYIff2PGxvhFq/Smc+RS8wzfXBxcyr
swCgYIKoZIzj0EAwIDRwAwRAIgZpFB
NYRJ8ZQjxXK2kVAmfYFqkQbcDc2RxNZuFK
hRifgCIH3rbykF0w7Q6mOtAG/A7m1r
m+b5qkzEoJHt3IH9HU+0
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
```

```
  <ds:Object Id="SignatureProperties">
    <xades:QualifyingProperties Target="SIGNATURE_ID">
      <xades:SignedProperties Id="SignedProperties">
        <xades:SignedSignatureProperties>
          <xades:SigningTime>2025-07-01T15:11:00Z</
    xades:SigningTime>
          <xades:SigningCertificate>
            <xades:Cert>
              <xades:CertDigest>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
                <ds:DigestValue>
1a2b3c4d5e6f7a8b9c0d1e2f3a4b5
c6d7e8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b
                </ds:DigestValue>
              </xades:CertDigest>
              <xades:IssuerSerial>
                <ds:X509IssuerName>
                CN=Contoh Otoritas Sertifikat, O=Organisasi,
    C=ID
                </ds:X509IssuerName>
                <ds:X509SerialNumber>
                1234567890
                </ds:X509SerialNumber>
              </xades:IssuerSerial>
            </xades:Cert>
          </xades:SigningCertificate>
        </xades:SignedSignatureProperties>
        <xades:SignedDataObjectProperties>
          <xades:DataObjectFormat ObjectReference="">
            <xades:MimeType>application/pdf</xades:MimeType>
          </xades:DataObjectFormat>
        </xades:SignedDataObjectProperties>
      </xades:SignedProperties>
      <xades:UnsignedProperties>
        <xades:UnsignedSignatureProperties>
          <xades:SignatureTimeStamp>
            <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          </xades:SignatureTimeStamp>
        </xades:UnsignedSignatureProperties>
      </xades:UnsignedProperties>
    </xades:QualifyingProperties>
```

```
  </ds:Object>
</ds:Signature>

```

Kode 3.6: Skema XML pada skenaro serangan XML *injection*

Dokumen PDF tersebut akan diunggah pelaku dengan harapan sistem tidak akan memvalidasi input dan menggangap bahwa *file* sudah di tandatangani oleh dosen pembimbing, penguji, dan kepala program studi.

- A04:2021 - *Insecure Design*: Skenario yang coba di uji: mahasiswa mengunggah skripsi, disetujui oleh Dosen Pembimbing 1. Kemudian, mahasiswa mengunggah versi dokumen yang berbeda untuk Dosen Pembimbing 2. Jika sistem tidak dirancang untuk me-*reset* semua status persetujuan setiap kali *file* baru diunggah, maka ada potensi dokumen yang disetujui oleh Pembimbing 2 berbeda dari yang dilihat oleh Pembimbing 1. Ini adalah kelemahan logika bisnis yang dapat dieksploitasi tanpa meretas sistem secara teknis.

- A05:2021 - *Security Misconfiguration*: Skenario serangan berfokus pada *server* yang berjalan dalam mode *debug* di *environment production*. Penyerang dengan sengaja memasukkan data yang tidak valid pada formulir unggah untuk memicu *exception*. Karena mode *debug* aktif, sistem menampilkan halaman *stack trace* yang rinci, yang membocorkan informasi sensitif seperti versi *framework* (PHP Laravel), nama-nama tabel database, path file di server, dan variabel *environment*. Informasi ini sangat berharga bagi penyerang untuk merencanakan serangan lebih lanjut.

- A06:2021 - *Vulnerable and Outdated Components*: Scenarionya: Seorang mahasiswa (penyerang) membuat dan mengunggah sebuah *file* PDF yang dibuat secara khusus (*crafted*) untuk mengeksploitasi kerentanan ini. Ketika *server* mencoba memproses PDF tersebut untuk menambahkan QR *code*, *payload* RCE terpicu, memberikan penyerang kendali penuh atas *server*.

- A07:2021 - *Identification and Authentication Failures*: Skenarionya: Penyerang, yang mengetahui email seorang Ketua Program Studi,

menggunakan sebuah skrip otomatis untuk mencoba ribuan kombinasi kata sandi umum. Jika sistem tidak memiliki mekanisme *rate limiting* atau penguncian akun sementara setelah beberapa kali gagal login, skrip tersebut dapat terus berjalan hingga berhasil menebak kata sandi. Setelah berhasil, penyerang dapat *login* sebagai Kaprodi dan menyetujui dokumen skripsi secara tidak sah.

- A08:2021 - *Software and Data Integrity Failures*: Skenarionya: Mahasiswa mengunggah dokumen skripsi dan disetujui secara sah oleh dosen pembimbing. Sistem menandatangani dokumen dan menghasilkan PDF yang sudah tertanam QR code dan metadata XML. Mahasiswa kemudian menggunakan editor PDF untuk mengubah konten di dalam dokumen (misalnya, mengubah beberapa paragraf atau data hasil penelitian) tanpa merusak struktur PDF atau tanda tangan yang ada. Jika sistem verifikasi hanya memeriksa keberadaan tanda tangan digital, tetapi gagal membandingkan *hash* dari konten dokumen saat ini dengan *hash* asli yang disimpan di database saat penandatanganan, maka integritas data telah gagal dan dokumen yang telah dimodifikasi akan dianggap sah.

- A09:2021 - *Security Logging and Monitoring Failures*: Skenarionya: Penyerang berhasil melakukan *brute-force* (A07), lalu mencoba mengakses beberapa halaman admin dan gagal karena *Broken Access Control* (A01), namun akhirnya berhasil menemukan satu dokumen yang bisa disetujui. Jika sistem gagal mencatat (*log*) upaya *login* yang gagal, upaya akses yang ditolak, dan aktivitas persetujuan dokumen (siapa, apa, kapan, dari mana), maka ketika terjadi insiden, tim keamanan tidak memiliki jejak audit untuk melakukan investigasi, memahami cakupan kerusakan, dan mencegah serangan serupa di masa depan.

- A10:2021 - *Server-Side Request Forgery* (SSRF): Skenario untuk aplikasi ini sulit terjadi karena desainnya tidak memiliki fungsionalitas di mana server mengambil sumber daya dari URL yang diberikan pengguna. Namun, jika ada fitur imajiner seperti ”Impor profil dosen dari LinkedIn”, dan pengguna dapat memasukkan URL, penyerang dapat memasukkan URL internal seperti

`http://localhost/server-status`. Server, saat mencoba mengambil data dari URL tersebut, justru akan menyerang atau membocorkan data dari jaringannya sendiri.