KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
POLITEKNIK NEGERI SRIWIJAYA
JURUSAN TEKNIK KOMPUTER
Jalan Srijaya Negara Bukit Besar Palembang 30139
Telepon 0711-353414 faksimil 0711-355918 E-mail : komputer@polsri.ac.id

Palembang, 17 April 2024

Hal : Permohonan Surat Pengantar Izin Pengambilan Data

KepadaYth.
Wakil Direktur I
Politeknik Negeri Sriwijaya

Dengan Hormat,
Berdasarkan kurikulum Jurusan Teknik Komputer Politeknik Negeri Sriwijaya bahwa Laporan Akhir merupakan mata kuliah wajib dilaksanakan pada akhir semester 6 (enam). Schubungan dengan itu, saya yang bertanda tangan di bawah ini:

No	Nama Mahasiswa	NPM	Kelas	Jurusan
1.	Afrilia Riyunsyah	062130701774	6 CN	Teknik Komputer

Dengan ini bermaksud mengajukan permohonan untuk dibuatkan Surat Izin Pengambilan Data yang diajukan kepada :

Yth.
Kepala Dinas Perpustakaan dan Kearsipan Kota Prabumulih
Jl. Jendral Sudirman, Prabumulih, Prabumulih Barat, Kota Prabumulih, Sumatera Selatan 31113

Demikianlah surat permohonan ini saya buat, atas bantuan dari Bapak saya ucapkan terima kasih.

Hormat Saya,

Afrilia Riyunsyah
NPM 062130701774

Mengetahui,
Ketua Jurusan Teknik Komputer

Azwardi, ST., MT.
NIP 197005232005011004

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara Bukit Besar – Palembang 30139 Telepon (0711) 353414
Laman : http://polsri.ac.id, Pos El : info@polsri.ac.id

Nomor : 3968/PL6.3.1/SP/2024
Perihal : Izin Pengambilan Data

18 April 2024

Yth. Kepala
Dinas Perpustakaan dan Kearsipan Kota Prabumulih
Jalan Jenderal Sudirman Prabumulih Kecamatan Prabumulih Barat
Kota Prabumulih 31113

Dengan hormat,

Sesuai dengan kurikulum Jurusan Teknik Komputer pada Politeknik Negeri Sriwijaya, Laporan Akhir merupakan mata kuliah wajib pada semester 6 (enam). Untuk itu mahasiswa kami memerlukan data untuk penyusunan Laporan Akhir tersebut.

Sehubungan dengan hal tersebut di atas, kami mohon Bapak/Ibu dapat memberikan izin dan membantu mahasiswa kami ini untuk melakukan penelitian.

Mahasiswa kami yang akan mengumpulkan data tersebut adalah :

No	Nama	NPM	Kelas	Jurusan / Program Studi
1	Afrilia Riyunsyah	0621 3070 1774	6 CN	Teknik Komputer

Atas perhatian dan bantuannya diucapkan terima kasih.

a.n. Plt. Direktur
Wakil Direktur I,

Carlos RS, S.T., M.T.
NIP 196403011989031003

Tembusan:
1. Plt. Direktur
2. Ketua Jurusan Teknik Komputer
3. Yang bersangkutan
4. Arsip

MS.Word/E/AD/Dw

PEMERINTAH KOTA PRABUMULIH

DINAS PERPUSTAKAAN DAN KEARSIPAN

Jl. Jend. Sudirman No. 011 (depan Kantor DPRD Kota Prabumulih)
Kel. Prabumulih Kec. Prabumulih Barat Kota Prabumulih Telp : (0713) 32000

Prabumulih, 3 Mei 2024

Nomor : 000.9/32/DPK/2024
Sifat : Biasa/Terbuka
Lampiran : Biasa
Hal : Izin Pengambilan Data Untuk Penelitian

Yth.
Wakil Direktur I Politeknik Negeri Sriwijaya
di
Palembang

Sehubungan dengan surat dari Politeknik Negeri Sriwijaya Nomor : 3968/PI.6.3.1/SP/2024 tanggal 18 April 2024 perihal Izin Pengambilan Data. Kami memberikan izin untuk melakukan penelitian guna penyusunan Laporan Akhir kepada :

Nama	:	Afrilia Riyunsyah
NPM	:	0621 3070 1774
Jurusan/Program Studi	:	Teknik Komputer

Selama penelitian mahasiswa tersebut dapat mengikuti serta mematuhi semua peraturan yang ada di Dinas Perpustakaan dan Kearsipan Kota Prabumulih.

Demikian disampaikan, atas perhatian dan kerjasamanya diucapkan terima kasih.

Kepala Dinas Perpustakaan dan
Kearsipan Kota Prabumulih

Drs. MAHPUZI, M.Si
Pembina Tk.I (IV/b)
196706112008011004

KEMENTERIAN PENDIDIKAN KEBUDAYAAN,
RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

LEMBAR KONSULTASI PROPOSAL LAPORAN AKHIR
TAHUN AKADEMIK 2024

Nama Mahasiswa : Afrilia Riyunsyah

NIM : 062130701774

Jurusan/Program Studi : DIII Teknik Komputer

Dosen Pembimbing : Herlambang Saputra, Ph.D

Judul : RANCANG BANGUN APLIKASI PERPUSTAKAAN BERBASIS *WEB* MENGGUNAKAN QR-CODE PADA PERPUSTAKAAN DAERAH KOTA PRABUMULIH

No	TANGGAL	URAIAN	PARAF PEMBIMBING
1	22-2-24.	Acc Judul.	
2.	22-3-24	Revisi Bab I	
3.	27-3-24.	Revisi Bab I.	
4.	18-4-24.	Revisi Bab I.	
5.	13-5-24	Revisi Bab I	
6.	17-5-24	Acc Bab I & Revisi Bab II.	
7.	10-5-24	Revisi Bab II	
8.	20-5-24.	Acc Bab II.	
9.	27-5-24.	Revisi Bab III.	
10.	31-5-24	Revisi Bab III.	
11.	5-6-24	Acc Bab III.	

Palembang , Maret 2024

Menyetujui,

Ketua Jurusan Teknik Komputer

Azwardi, S.T., M.T

NIP : 19700523200501004

KEMENTERIAN PENDIDIKAN KEBUDAYAAN,
RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

LEMBAR KONSULTASI PROPOSAL LAPORAN AKHIR
TAHUN AKADEMIK 2024

Nama Mahasiswa : Afrilia Riyunsyah

NIM : 062130701774

Jurusan/Program Studi : DIII Teknik Komputer

Dosen Pembimbing : Herlambang Saputra, Ph.D

Judul : RANCANG BANGUN APLIKASI PERPUSTAKAAN BERBASIS WEB MENGGUNAKAN QR-CODE PADA PERPUSTAKAAN DAERAH KOTA PRABUMULIH

No	TANGGAL	URAIAN	PARAF PEMBIMBING
12.	6 - 6 - 2024.	Revisi DP	
13.	8 - 6 - 2024.	Revisi - DP.	
14.	11 - 6 - 2024.	Acc DP	
15.	8 - 7 - 2024	Revisi BAB III.	
16.	11 - 7 - 2024	Revisi BAB IV.	
17.	12 - 7 - 2024.	Acc BAB IV & Revisi BAB V	
18.	12 - 7 - 2024	Acc BAB V. Acc Ujian CP	

Palembang , Februari 2024
Menyetujui,
Ketua Jurusan Teknik Komputer

Azwardi, S.T., M.T
NIP : 19700523200501004

KEMENTERIAN PENDIDIKAN KEBUDAYAAN,
RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

LEMBAR KONSULTASI PROPOSAL LAPORAN AKHIR
TAHUN AKADEMIK 2024

Nama Mahasiswa : Afrilia Riyunsyah

NIM : 062130701774

Jurusan/Program Studi : DIII Teknik Komputer

Dosen Pembimbing : Ariansyah Saputra.,M.Kom.

Judul : RANCANG BANGUN APLIKASI PERPUSTAKAAN BERBASIS *WEB* MENGGUNAKAN QR-CODE PADA PERPUSTAKAAN DAERAH KOTA PRABUMULIH

No	TANGGAL	URAIAN	PARAF PEMBIMBING
1	01 maret 2024	Konsultasi judul	
2	17 april 2024	Acc judul & Revisi bab 1	
3	24 april 2024	revisi bab 1	
4.	14 mei 2024	revisi bab 1	
5.	16 mei 2024	revisi bab 1 & revisi bab 2	
6.	20 mei 2024	ACC 1 & 2, revisi bab 3	
7.	28 mei 2024	revisi bab 3	
8.	30 mei 2024	~~revisi~~ ACC bab 3	
9.	8 juli 2024	konsultasi bab 4 & bab 5	
10.	15 juli 2024	revisi bab 4 & bab 5	
11.	11 juli 2024	ACC bab 4 & Bab 5	

Palembang , Maret 2024
Menyetujui,
Ketua Jurusan Teknik Komputer

Azwardi, S.T., M.T

NIP : 19700523200501004

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

REKOMENDASI UJIAN TUGAS AKHIR

Pembimbing Laporan Tugas Akhir, memberikan rekomendasi ujian laporan tugas akhir kepada,

Nama Mahasiswa	:	Afrilia Riyunsyah
NIM	:	062130701774
Jurusan/Program Studi	:	Teknik Komputer/D3 Teknik Komputer
Judul Tugas Akhir	:	Rancang Bangun Aplikasi Perpustakaan Berbasis *Website* Pada Perpustakaan Daerah Kota Prabumulih

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti Ujian Laporan Tugas Akhir, pada Tahun Akademik 2024/[redacted]

Palembang, 2024

Pembimbing I

Herlambang Saputra, M.Kom, Ph.D
NIP. 198103182008121002

Pembimbing II

Ariansyah Saputra.,M.Kom
NIP. 198907122019031012

No.Dok. :	Tgl. Berlaku :	No. Rev. :
	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI **POLITEKNIK NEGERI SRIWIJAYA** Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414 **Website :** www.polisriwijaya.ac.id **E-mail :** info@polsri.ac.id	
	REVISI TUGAS AKHIR (TA)	

Dosen Penguji	:	Yulian Mirza, S.T., M.Kom
Nama Mahasiswa	:	Afrilia Riyunsyah
NIM	:	062130701774
Jurusan/Program Studi	:	Teknik Komputer/DIII-Teknik Komputer
Judul LA/Skripsi	:	Rancang Bangun Aplikasi Perpustakaan Berbasis Website Pada Perpustakaan Daerah Kota Prabumulih

NO	Uraian Revisi	Paraf
	tata tulis pembahasan kesimpulan	

Palembang, Januari 2024
Dosen Penguji,

Yulian Mirza, S.T., M.Kom
NIP. 196607121990031003

No Dok. :	Tgl. Berlaku :	No. Rev. :
	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI **POLITEKNIK NEGERI SRIWIJAYA** Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414 **Website :** www.polisriwijaya.ac.id **E-mail :** info@polsri.ac.id	
	REVISI TUGAS AKHIR (TA)	

Dosen Penguji	:	Alan Novi Tompunu, S.T., M.T
Nama Mahasiswa	:	Afrilia Riyunsyah
NIM	:	062130701774
Jurusan/Program Studi	:	Teknik Komputer/DIII-Teknik Komputer
Judul LA/Skripsi	:	Rancang Bangun Aplikasi Perpustakaan Berbasis Website Pada Perpustakaan Daerah Kota Prabumulih

NO	Uraian Revisi	Paraf
	Acc	

Palembang, 18 Juli 2024
Dosen Penguji,

Alan Novi Tompunu, S.T., M.T
NIP. 197611082000031002

No.Dok. :	Tgl. Berlaku :	No. Rev. :
	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI **POLITEKNIK NEGERI SRIWIJAYA** Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414 **Website :** www.polisriwijaya.ac.id **E-mail :** info@polsri.ac.id	
	REVISI TUGAS AKHIR (TA)	

Dosen Penguji	:	Hartati Deviana, S.T., M.Kom
Nama Mahasiswa	:	Afrilia Riyunsyah
NIM	:	062130701774
Jurusan/Program Studi	:	Teknik Komputer/DIII-Teknik Komputer
Judul LA/Skripsi	:	Rancang Bangun Aplikasi Perpustakaan Berbasis Website Pada Perpustakaan Daerah Kota Prabumulih

NO	Uraian Revisi	Paraf
	Perbaiki aplikasi	

Palembang, Juli 2024
Dosen Penguji,

Hartati Deviana, S.T., M.Kom
NIP. 197405262008122001

No.Dok. :	Tgl. Berlaku :	No. Rev. :
	KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI **POLITEKNIK NEGERI SRIWIJAYA** Jalan Srijaya Negara, Palembang 30139 Telp. 0711-353414 **Website :** www.polisriwijaya.ac.id **E-mail :** info@polsri.ac.id	
	REVISI TUGAS AKHIR (TA)	

Dosen Penguji	:	Rian Rahmanda Putra, S.Kom., M.Kom
Nama Mahasiswa	:	Afrilia Riyunsyah
NIM	:	062130701774
Jurusan/Program Studi	:	Teknik Komputer/DIII-Teknik Komputer
Judul LA/Skripsi	:	Rancang Bangun Aplikasi Perpustakaan Berbasis Website Pada Perpustakaan Daerah Kota Prabumulih

NO	Uraian Revisi	Paraf
1.	Turnitin Similarity tidak lebih dari < 15%.	
2.	Selesaikan aplikasi sesuai dengan yang dijanjikan	

Palembang, Juli 2024
Dosen Penguji,

Rian Rahmanda Putra, S.Kom., M.Kom
NIP. 198901252019031013

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI SRIWIJAYA JURUSAN
TEKNIK KOMPUTER
Jalan Srijaya Negara, Palembang 30139. Telp. 0711-353414
Website : www.polsri.ac.id E-mail : info@polsri.ac.id

PELAKSANAAN REVISI UJIAN TUGAS AKHIR

Nama Mahasiswa : Afrilia Riyunsyah
NIM : 062130701774
Jurusan /Program Studi : DIII Teknik Komputer
Judul LA/ Skripsi : Rancang Bangun Aplikasi Perpustakaan Berbasis *Website* Pada Perpustakaan Daerah Kota Prabumulih

Telah melaksanakan revisi terhadap Laporan Tugas Akhir yang diujikan pada hari Kamis tanggal 18 bulan 7 tahun 2024 Pelaksanaan revisi terhadap Laporan Tugas Akhir tersebut telah disetujui oleh Dosen Penguji yang memberikan revisi:

No	Komentar	Nama Dosen Penguji	Tanggal/ bulan	Tanda Tangan
1.	ACC	Yulian Mirza, ST, M.Kom	28/7	
2.	ACC	Ir. Alan Novi Tompunu, ST, MT, IPM.,ASEAN Eng	26/7	
3.	ACC	Hartati Deviana, ST,M.Kom	30/7	
4.	ACC	Rian Rahmanda Putra, S.Kom., M.Kom	7/8 2024	

Palembang, Juli 2024
Ketua Penguji,

Yulian Mirza, ST, M.Kom
NIP. 196607121990031003

```
<?= $this->extend('layouts/home_layout') ?>

<?= $this->section('head') ?>
<title>Home</title>
<?= $this->endSection() ?>

<?= $this->section('content') ?>
<div class="px-4 pt-5 my-5 text-center border-bottom">
  <h1 class="display-4 fw-bold text-body-emphasis">Perpustakaan<span class="text-primary"> Kota Prabumulih</span></h1>
  <div class="col-lg-6 mx-auto">
    <p class="lead mb-4">Temukan buku-buku menarik untuk memperluas pengetahuan dan imajinasi Anda. Perpustakaan Kota Prabumulih adalah teman setia pencinta buku dan pembelajar di mana saja, kapan saja.</p>
    <div class="d-grid gap-2 d-sm-flex justify-content-sm-center mb-5">
      <a href="<?= base_url('login'); ?>" class="btn btn-primary btn-lg px-4 me-sm-3">Login</a>
      <a href="<?= base_url('register_pengunjung'); ?>" class="btn btn-primary btn-lg px-4 me-sm-3">Register</a>
      <a href="<?= base_url('book'); ?>" class="btn btn-outline-secondary btn-lg px-4">Daftar buku</a>
    </div>
  </div>
  <div class="overflow-hidden" style="max-height: 45vh;">
    <div class="container px-5">
      <img src="<?= base_url('assets/images/dashboard.png'); ?>" class="img-fluid border rounded-3 " alt="Example image" width="200" height="500" loading="lazy">
    </div>
  </div>
</div>
<?= $this->endSection() ?>

<?php

namespace App\Controllers\Books;

use App\Models\BookModel;
use App\Libraries\QRGenerator;
use App\Models\BookStockModel;
use App\Models\CategoryModel;
use App\Models\LoanModel;
use App\Models\RackModel;
use CodeIgniter\Exceptions\PageNotFoundException;
use CodeIgniter\RESTful\ResourceController;

class BooksController extends ResourceController
{
    protected BookModel $bookModel;
    protected CategoryModel $categoryModel;
    protected RackModel $rackModel;
    protected BookStockModel $bookStockModel;
    protected LoanModel $loanModel;

    public function __construct()
    {
        $this->bookModel = new BookModel;
```

```
        $this->categoryModel = new CategoryModel;
        $this->rackModel = new RackModel;
        $this->bookStockModel = new BookStockModel;
        $this->loanModel = new LoanModel;

        helper('upload');
    }

    /**
     * Return an array of resource objects, themselves in array format
     *
     * @return mixed
     */
    public function index()
    {
        $itemPerPage = 20;

        if ($this->request->getGet('search')) {
            $keyword = $this->request->getGet('search');
            $books = $this->bookModel
                ->select('books.*, book_stock.quantity, categories.name as category, racks.name as rack,
racks.floor')
                ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
                ->join('categories', 'books.category_id = categories.id', 'LEFT')
                ->join('racks', 'books.rack_id = racks.id', 'LEFT')
                ->like('title', $keyword, insensitiveSearch: true)
                ->orLike('slug', $keyword, insensitiveSearch: true)
                ->orLike('author', $keyword, insensitiveSearch: true)
                ->orLike('publisher', $keyword, insensitiveSearch: true)
                ->paginate($itemPerPage, 'books');

            $books = array_filter($books, function ($book) {
                return $book['deleted_at'] == null;
            });
        } else {
            $books = $this->bookModel
                ->select('books.*, book_stock.quantity, categories.name as category, racks.name as rack,
racks.floor')
                ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
                ->join('categories', 'books.category_id = categories.id', 'LEFT')
                ->join('racks', 'books.rack_id = racks.id', 'LEFT')
                ->paginate($itemPerPage, 'books');
        }

        $data = [
            'books'        => $books,
            'pager'        => $this->bookModel->pager,
            'currentPage'   => $this->request->getVar('page_books') ?? 1,
            'itemPerPage'   => $itemPerPage,
            'search'       => $this->request->getGet('search')
        ];

        return view('books/index', $data);
    }
```

```
    /**
     * Return the properties of a resource object
     *
     * @return mixed
     */
    public function show($uid = null)
    {
        $book = $this->bookModel
            ->select('books.*, book_stock.quantity, categories.name as category, racks.name as rack,
racks.floor')
            ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
            ->join('categories', 'books.category_id = categories.id', 'LEFT')
            ->join('racks', 'books.rack_id = racks.id', 'LEFT')
            ->where('uid', $uid)->first();

        if (empty($book)) {
            throw new PageNotFoundException('Book with slug \'' . $uid . '\' not found');
        }

        $loans = $this->loanModel->where([
            'book_id' => $book['id'],
            'return_date' => null
        ])->findAll();

        $loanCount = array_reduce(
            array_map(function ($loan) {
                return $loan['quantity'];
            }, $loans),
            function ($carry, $item) {
                return ($carry + $item);
            }
        );

        $bookStock = $book['quantity'] - $loanCount;

        $data = [
            'book'      => $book,
            'loanCount' => $loanCount ?? 0,
            'bookStock' => $bookStock
        ];

        return view('books/show', $data);
    }

    /**
     * Return a new resource object, with default properties
     *
     * @return mixed
     */
    public function new()
    {
        $categories = $this->categoryModel->findAll();
        $racks = $this->rackModel->findAll();

        $data = [
```

```
            'categories' => $categories,
            'racks'      => $racks,
            'validation' => \Config\Services::validation(),
        ];

        return view('books/create', $data);
    }

    /**
     * Create a new resource object, from "posted" parameters
     *
     * @return mixed
     */
    public function create()
    {
        if (!$this->validate([
            'cover'     =>
'is_image[cover]|mime_in[cover,image/jpg,image/jpeg,image/gif,image/png,image/webp]|max_size[c
over,5120]',
            'title'     => 'required|string|max_length[127]',
            'author'    => 'required|alpha_numeric_punct|max_length[64]',
            'publisher' => 'required|string|max_length[64]',
            'isbn'      => 'required|numeric|min_length[10]|max_length[13]',
            'year'      => 'required|numeric|min_length[4]|max_length[4]|less_than_equal_to[2100]',
            'rack'      => 'required|numeric',
            'category'  => 'required|numeric',
            'stock'     => 'required|numeric|greater_than_equal_to[1]',
            'file'      => 'permit_empty|mime_in[file,application/pdf]|max_size[file,10240]',
        ])) {
            $categories = $this->categoryModel->findAll();
            $racks = $this->rackModel->findAll();

            $data = [
                'categories' => $categories,
                'racks'      => $racks,
                'validation' => \Config\Services::validation(),
                'oldInput'   => $this->request->getVar(),
            ];

            return view('books/create', $data);
        }

        $coverImage = $this->request->getFile('cover');
        $pdfFile = $this->request->getFile('file');

        if ($coverImage->getError() != 4) {
            $coverImageFileName = uploadBookCover($coverImage);
        }

        if ($pdfFile->getError() != 4) {
            $pdfFileName = uploadBookCover($pdfFile);
        }

        $slug = url_title($this->request->getVar('title') . ' ' . rand(0, 1000), '-', true);
        $uid = sha1(
```

```
            $this->request->getVar('first_name')
                . $this->request->getVar('email')
                . $this->request->getVar('phone')
                . rand(0, 1000)
                . md5($this->request->getVar('gender'))
        );

        $qrGenerator = new QRGenerator();
        $qrCodeLabel = $this->request->getVar('slug')
            . ($this->request->getVar('isbn') ? ' ' . $this->request->getVar('isbn') : '');
        $qrCode = $qrGenerator->generateQRCode(
            data: $uid,
            labelText: $qrCodeLabel,
            dir: BOOKS_QR_CODE_PATH,
            filename: $qrCodeLabel
        );

        if (!$this->bookModel->save([
            'uid'  => $uid,
            'slug' => $slug,
            'title' => $this->request->getVar('title'),
            'author' => $this->request->getVar('author'),
            'publisher' => $this->request->getVar('publisher'),
            'isbn' => $this->request->getVar('isbn'),
            'year' => $this->request->getVar('year'),
            'rack_id' => $this->request->getVar('rack'),
            'category_id' => $this->request->getVar('category'),
            'book_cover' => $coverImageFileName ?? null,
            'qr_code'  => $qrCode,
            'file' => $pdfFileName ?? null,
        ]) || !$this->bookStockModel->save([
            'book_id' => $this->bookModel->getInsertID(),
            'quantity' => $this->request->getVar('stock'),

        ])) {
            $categories = $this->categoryModel->findAll();
            $racks = $this->rackModel->findAll();

            $data = [
                'categories' => $categories,
                'racks'      => $racks,
                'validation' => \Config\Services::validation(),
                'oldInput'   => $this->request->getVar(),
            ];

            session()->setFlashdata(['msg' => 'Insert failed']);
            return view('books/create', $data);
        }

        session()->setFlashdata(['msg' => 'Insert new book successful']);
        return redirect()->to('admin/books');
    }


    /**
```

```
     * Return the editable properties of a resource object
     *
     * @return mixed
     */
    public function edit($slug = null)
    {
        $book = $this->bookModel
            ->select('books.*, book_stock.quantity')
            ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
            ->where('slug', $slug)->first();

        if (empty($book)) {
            throw new PageNotFoundException('Book with slug \" . $slug . '\' not found');
        }

        $categories = $this->categoryModel->findAll();
        $racks = $this->rackModel->findAll();

        $data = [
            'book'       => $book,
            'categories' => $categories,
            'racks'      => $racks,
            'validation' => \Config\Services::validation(),
        ];

        return view('books/edit', $data);
    }

    /**
     * Add or update a model resource, from "posted" properties
     *
     * @return mixed
     */
    public function update($slug = null)
    {
        $book = $this->bookModel->where('slug', $slug)->first();

        if (empty($book)) {
            throw new PageNotFoundException('Book with slug \" . $slug . '\' not found');
        }

        if (!$this->validate([
            'cover'     =>
'is_image[cover]|mime_in[cover,image/jpg,image/jpeg,image/gif,image/png,image/webp]|max_size[c
over,5120]',
            'title'     => 'required|string|max_length[127]',
            'author'    => 'required|alpha_numeric_punct|max_length[64]',
            'publisher' => 'required|string|max_length[64]',
            'isbn'      => 'required|numeric|min_length[10]|max_length[13]',
            'year'      => 'required|numeric|min_length[4]|max_length[4]|less_than_equal_to[2100]',
            'rack'      => 'required|numeric',
            'category'  => 'required|numeric',
            'stock'     => 'required|numeric|greater_than_equal_to[1]',
        ])) {
            $categories = $this->categoryModel->findAll();
```

```
    $racks = $this->rackModel->findAll();

    $data = [
        'book'       => $book,
        'categories' => $categories,
        'racks'      => $racks,
        'validation' => \Config\Services::validation(),
        'oldInput'   => $this->request->getVar(),
    ];

    return view('books/edit', $data);
}

$bookStock = $this->bookStockModel->where('book_id', $book['id'])->first();

$coverImage = $this->request->getFile('cover');

if ($coverImage->getError() == 4) {
    $coverImageFileName = $book['book_cover'];
} else {
    $coverImageFileName = updateBookCover(
        newCoverImage: $coverImage,
        formerCoverImageFileName: $book['book_cover']
    );
}

$slug = $this->request->getVar('title') != $book['title']
    ? url_title($this->request->getVar('title') . ' ' . rand(0, 1000), '-', true)
    : $book['slug'];

if (!$this->bookModel->save([
    'id'  => $book['id'],
    'slug' => $slug,
    'title' => $this->request->getVar('title'),
    'author' => $this->request->getVar('author'),
    'publisher' => $this->request->getVar('publisher'),
    'isbn' => $this->request->getVar('isbn'),
    'year' => $this->request->getVar('year'),
    'rack_id' => $this->request->getVar('rack'),
    'category_id' => $this->request->getVar('category'),
    'book_cover' => $coverImageFileName ?? null,
]) || !$this->bookStockModel->save([
    'id' => $bookStock['id'],
    'book_id' => $book['id'],
    'quantity' => $this->request->getVar('stock')
])) {
    $categories = $this->categoryModel->findAll();
    $racks = $this->rackModel->findAll();

    $data = [
        'book'       => $book,
        'categories' => $categories,
        'racks'      => $racks,
        'validation' => \Config\Services::validation(),
        'oldInput'   => $this->request->getVar(),
```

```
            ];

            session()->setFlashdata(['msg' => 'Update failed']);
            return view('books/edit', $data);
        }

        session()->setFlashdata(['msg' => 'Update book successful']);
        return redirect()->to('admin/books');
    }

    /**
     * Delete the designated resource object from the model
     *
     * @return mixed
     */
    public function delete($slug = null)
    {
        $book = $this->bookModel->where('slug', $slug)->first();

        if (empty($book)) {
            throw new PageNotFoundException('Book with slug \" . $slug . '\' not found');
        }

        $bookStock = $this->bookStockModel->where('book_id', $book['id'])->first();

        if (!$this->bookModel->delete($book['id']) || !$this->bookStockModel-
>delete($bookStock['id'])) {
            session()->setFlashdata(['msg' => 'Failed to delete book', 'error' => true]);
            return redirect()->back();
        }

        // delete former image file
        deleteBookCover($book['book_cover']);

        session()->setFlashdata(['msg' => 'Book deleted successfully']);
        return redirect()->to('admin/books');
    }
}
```

```
<?php

namespace App\Controllers\Loans;

use App\Libraries\QRGenerator;
use App\Models\BookModel;
use App\Models\LoanModel;
use App\Models\MemberModel;
use CodeIgniter\Exceptions\PageNotFoundException;
use CodeIgniter\I18n\Time;
use CodeIgniter\RESTful\ResourceController;

class LoansController extends ResourceController
{
    protected LoanModel $loanModel;
    protected MemberModel $memberModel;
```

```
protected BookModel $bookModel;

public function __construct()
{
    $this->loanModel = new LoanModel;
    $this->memberModel = new MemberModel;
    $this->bookModel = new BookModel;

    helper('upload');
}

/**
 * Return an array of resource objects, themselves in array format
 *
 * @return mixed
 */
public function index()
{
    $itemPerPage = 20;
    $userGroup = auth()->user()->getGroups()[0];
    if ($this->request->getGet('search')) {
        $keyword = $this->request->getGet('search');
        $loans = $this->loanModel
            ->select('members.*, members.uid as member_uid, books.*, loans.*')
            ->join('members', 'loans.member_id = members.id', 'LEFT')
            ->join('books', 'loans.book_id = books.id', 'LEFT')
            ->like('first_name', $keyword, insensitiveSearch: true)
            ->orLike('last_name', $keyword, insensitiveSearch: true)
            ->orLike('email', $keyword, insensitiveSearch: true)
            ->orLike('title', $keyword, insensitiveSearch: true)
            ->orLike('slug', $keyword, insensitiveSearch: true)
            ->paginate($itemPerPage, 'loans');
    } else {
        if ($userGroup == 'pengunjung') {
            $userEmail = auth()->user()->email;
            $loans = $this->loanModel
                ->select('members.*, members.uid as member_uid, books.*, loans.*')
                ->join('members', 'loans.member_id = members.id', 'LEFT')
                ->join('books', 'loans.book_id = books.id', 'LEFT')
                ->where('members.email', $userEmail)
                ->paginate($itemPerPage, 'loans');
        }
        else{
            $loans = $this->loanModel
            ->select('members.*, members.uid as member_uid, books.*, loans.*')
            ->join('members', 'loans.member_id = members.id', 'LEFT')
            ->join('books', 'loans.book_id = books.id', 'LEFT')
            ->paginate($itemPerPage, 'loans');
        }
    }

    $loans = array_filter($loans, function ($loan) {
        return $loan['deleted_at'] == null && $loan['return_date'] == null;
    });
```

```
        $data = [
            'loans'         => $loans,
            'pager'         => $this->loanModel->pager,
            'currentPage'   => $this->request->getVar('page_loans') ?? 1,
            'itemPerPage'   => $itemPerPage,
            'search'        => $this->request->getGet('search')
        ];

        return view('loans/index', $data);
    }

    /**
     * Return the properties of a resource object
     *
     * @return mixed
     */
    public function show($uid = null)
    {
        $loan = $this->loanModel
            ->select('members.*, members.uid as member_uid, books.*, loans.*, loans.qr_code as
loan_qr_code, book_stock.quantity as book_stock, racks.name as rack, categories.name as category')
            ->join('members', 'loans.member_id = members.id', 'LEFT')
            ->join('books', 'loans.book_id = books.id', 'LEFT')
            ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
            ->join('racks', 'books.rack_id = racks.id', 'LEFT')
            ->join('categories', 'books.category_id = categories.id', 'LEFT')
            ->where('loans.uid', $uid)
            ->first();

        if (empty($loan)) {
            throw new PageNotFoundException('Loan not found');
        }

        if ($this->request->getGet('update-qr-code') && $loan['return_date'] == null) {
            $qrGenerator = new QRGenerator();
            $qrCodeLabel = substr($loan['first_name'] . ($loan['last_name'] ? " {$loan['last_name']}" : ''),
0, 12) . '_' . substr($loan['title'], 0, 12);
            $qrCode = $qrGenerator->generateQRCode(
                $loan['uid'],
                labelText: $qrCodeLabel,
                dir: LOANS_QR_CODE_PATH,
                filename: $qrCodeLabel
            );

            // delete former qr code
            deleteLoansQRCode($loan['qr_code']);

            $this->loanModel->update($loan['id'], ['qr_code' => $qrCode]);

            $loan = $this->loanModel
                ->select('members.*, members.uid as member_uid, books.*, loans.*, loans.qr_code as
loan_qr_code, book_stock.quantity as book_stock, racks.name as rack, categories.name as category')
                ->join('members', 'loans.member_id = members.id', 'LEFT')
                ->join('books', 'loans.book_id = books.id', 'LEFT')
                ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
```

```
            ->join('racks', 'books.rack_id = racks.id', 'LEFT')
            ->join('categories', 'books.category_id = categories.id', 'LEFT')
            ->where('loans.uid', $uid)
            ->first();

        return redirect()->to("admin/loans/{$loan['uid']}");
    }

    $data = [
        'loan'         => $loan,
    ];

    return view('loans/show', $data);
}

public function searchMember()
{
    if ($this->request->isAJAX()) {
        $param = $this->request->getVar('param');

        if (empty($param)) return;

        $members = $this->memberModel
            ->like('first_name', $param, insensitiveSearch: true)
            ->orLike('last_name', $param, insensitiveSearch: true)
            ->orLike('email', $param, insensitiveSearch: true)
            ->orWhere('uid', $param)
            ->findAll();

        $members = array_filter($members, function ($member) {
            return $member['deleted_at'] == null;
        });

        if (empty($members)) {
            return view('loans/member', ['msg' => 'Member not found']);
        }

        return view('loans/member', ['members' => $members]);
    }

    return view('loans/search_member');
}

public function searchBook()
{
    if ($this->request->isAJAX()) {
        $param = $this->request->getVar('param');
        $memberUid = $this->request->getVar('memberUid');

        if (empty($param)) return;

        if (empty($memberUid)) {
            return view('loans/book', ['msg' => 'Member UID is empty']);
        }
```

```
            $books = $this->bookModel
                ->select('books.*, book_stock.quantity, categories.name as category, racks.name as rack,
racks.floor')
                ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
                ->join('categories', 'books.category_id = categories.id', 'LEFT')
                ->join('racks', 'books.rack_id = racks.id', 'LEFT')
                ->like('title', $param, insensitiveSearch: true)
                ->orLike('slug', $param, insensitiveSearch: true)
                ->orLike('author', $param, insensitiveSearch: true)
                ->orLike('publisher', $param, insensitiveSearch: true)
                ->orWhere('isbn', $param)
                ->findAll();

            $books = array_filter($books, function ($book) {
                return $book['deleted_at'] == null;
            });

            if (empty($books)) {
                return view('loans/book', ['msg' => 'Book not found']);
            }

            $books = array_map(function ($book) {
                $newBook = $book;
                $newBook['stock'] = $this->getRemainingBookStocks($book);
                return $newBook;
            }, $books);

            return view('loans/book', ['books' => $books, 'memberUid' => $memberUid]);
        }

        $memberUid = $this->request->getVar('member-uid');

        if (empty($memberUid)) {
            session()->setFlashdata(['msg' => 'Select member first', 'error' => true]);
            return redirect()->to('admin/loans/new/members/search');
        }

        $member = $this->memberModel->where('uid', $memberUid)->first();

        if (empty($member)) {
            session()->setFlashdata(['msg' => 'Member not found', 'error' => true]);
            return redirect()->to('admin/loans/new/members/search');
        }

        return view('loans/search_book', ['member' => $member]);
    }

    public function searchBook2()
    {
        if ($this->request->isAJAX()) {
            $param = $this->request->getVar('param');
            $memberUid = $this->request->getVar('memberUid');

            if (empty($param)) return;
```

```
            if (empty($memberUid)) {
                return view('loans/book', ['msg' => 'Member UID is empty']);
            }

            $books = $this->bookModel
                ->select('books.*, book_stock.quantity, categories.name as category, racks.name as rack,
racks.floor')
                ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
                ->join('categories', 'books.category_id = categories.id', 'LEFT')
                ->join('racks', 'books.rack_id = racks.id', 'LEFT')
                ->like('title', $param, insensitiveSearch: true)
                ->orLike('slug', $param, insensitiveSearch: true)
                ->orLike('author', $param, insensitiveSearch: true)
                ->orLike('publisher', $param, insensitiveSearch: true)
                ->orWhere('isbn', $param)
                ->findAll();

            $books = array_filter($books, function ($book) {
                return $book['deleted_at'] == null;
            });

            if (empty($books)) {
                return view('loans/book', ['msg' => 'Book not found']);
            }

            $books = array_map(function ($book) {
                $newBook = $book;
                $newBook['stock'] = $this->getRemainingBookStocks($book);
                return $newBook;
            }, $books);

            return view('loans/book', ['books' => $books, 'memberUid' => $memberUid]);
        }


        return view('loans/search_book2');
    }


    protected function getRemainingBookStocks($book)
    {
        $loans = $this->loanModel->where([
            'book_id' => $book['id'],
            'return_date' => null
        ])->findAll();

        $loanCount = array_reduce(
            array_map(function ($loan) {
                return $loan['quantity'];
            }, $loans),
            function ($carry, $item) {
                return ($carry + $item);
            }
        );
```

```
    return $book['quantity'] - $loanCount;
}

/**
 * Return a new resource object, with default properties
 *
 * @return mixed
 */
public function new($validation = null, $oldInput = null)
{
    if ($this->request->getMethod() === 'post') {
        return $this->create();
    }

    $member = $this->memberModel
        ->where('uid', $this->request->getVar('member_uid'))
        ->first();

    $books = [];

    $bookSlugs = $this->request->getVar('slugs');

    if (empty($bookSlugs)) {
        return redirect()->back();
    }

    foreach ($bookSlugs as $slug) {
        $book = $this->bookModel
            ->join('book_stock', 'books.id = book_stock.book_id', 'LEFT')
            ->where('books.slug', $slug)->first();

        if (!empty($book)) {
            $book['stock'] = $this->getRemainingBookStocks($book);
            array_push($books, $book);
        }
    }

    $data = [
        'books'      => $books,
        'member'     => $member,
        'validation' => $validation ?? \Config\Services::validation(),
        'oldInput'   => $oldInput,
    ];

    return view('loans/create', $data);
}

/**
 * Create a new resource object, from "posted" parameters
 *
 * @return mixed
 */
public function create()
{
    $validation = [
```

```
            'member_uid' => 'required|string|max_length[64]',
        ];

        $bookSlugs = $this->request->getVar('slugs') or die();

        foreach ($bookSlugs as $slug) {
            $validation['quantity-' . $slug] =
'required|numeric|integer|greater_than[0]|less_than_equal_to[10]';
            $validation['duration-' . $slug] =
'required|numeric|integer|greater_than[0]|less_than_equal_to[30]';
        }

        if (!$this->validate($validation)) {
            return $this->new(\Config\Services::validation(), $this->request->getVar());
        }

        $memberUid = $this->request->getVar('member_uid') or die();

        $member = $this->memberModel->where('uid', $memberUid)->first();

        if (empty($member)) {
            session()->setFlashdata(['msg' => 'Member not found']);
            return redirect()->to('admin/loans/new/members/search');
        }

        $newLoanIds = [];

        foreach ($bookSlugs as $slug) {
            $duration = $this->request->getVar('duration-' . $slug);
            $quantity = $this->request->getVar('quantity-' . $slug);

            $book = $this->bookModel->where('slug', $slug)->first();

            if (empty($duration) || empty($quantity) || empty($book)) {
                continue;
            }

            $loanUid = sha1($book['slug'] . $member['uid'] . time());

            $qrGenerator = new QRGenerator();

            $qrCodeLabel = substr($member['first_name'] . ($member['last_name'] ? "
{$member['last_name']}" : ''), 0, 12) . '_' . substr($book['title'], 0, 12);

            $qrCode = $qrGenerator->generateQRCode(
                data: $loanUid,
                labelText: $qrCodeLabel,
                dir: LOANS_QR_CODE_PATH,
                filename: $qrCodeLabel
            );

            $newLoan = [
                'book_id' => $book['id'],
                'quantity' => $quantity,
                'member_id' => $member['id'],
```

```
            'uid' => $loanUid,
            'loan_date' => Time::now()->toDateTimeString(),
            'due_date' => Time::now()->addDays(intval($duration))->toDateTimeString(),
            'qr_code' => $qrCode,
        ];

        $this->loanModel->insert($newLoan);

        array_push($newLoanIds, $this->loanModel->getInsertID());
    }

    $newLoans = array_map(function ($id) {
        return $this->loanModel->select('members.*, members.uid as member_uid, books.*, loans.*')
            ->join('members', 'loans.member_id = members.id', 'LEFT')
            ->join('books', 'loans.book_id = books.id', 'LEFT')
            ->where('loans.id', $id)->first();
    }, $newLoanIds);

    return view('loans/result', [
        'newLoans'  => $newLoans
    ]);
}

/**
 * Return the editable properties of a resource object
 *
 * @return mixed
 */
// public function edit($uid = null)
// {
//! Not implemented
// }

/**
 * Add or update a model resource, from "posted" properties
 *
 * @return mixed
 */
// public function update($uid = null)
// {
//! Not implemented
// }

/**
 * Delete the designated resource object from the model
 *
 * @return mixed
 */
public function delete($uid = null)
{
    $loan = $this->loanModel->where('uid', $uid)->first();

    if (empty($loan)) {
        throw new PageNotFoundException('Loan not found');
    };
```

```
        if (!$this->loanModel->delete($loan['id'])) {
            session()->setFlashdata(['msg' => 'Failed to delete loan', 'error' => true]);
            return redirect()->back();
        }

        deleteLoansQRCode($loan['qr_code']);

        session()->setFlashdata(['msg' => 'Loan deleted successfully']);
        return redirect()->to('admin/loans');
    }
}

<?php

namespace App\Controllers\Users;

use CodeIgniter\Exceptions\PageNotFoundException;
use CodeIgniter\RESTful\ResourceController;
use CodeIgniter\Shield\Authentication\Passwords;
use CodeIgniter\Shield\Entities\User;
use CodeIgniter\Shield\Models\UserModel;

class UsersController extends ResourceController
{
    protected UserModel $userModel;

    public function __construct()
    {
        $this->userModel = new UserModel;
    }

    /**
     * Return an array of resource objects, themselves in array format
     *
     * @return mixed
     */
    public function index()
    {
        $itemPerPage = 20;

        $users = $this->userModel->withIdentities()->paginate($itemPerPage, 'users');

        $data = [
            'users'             => $users,
            'pager'             => $this->userModel->pager,
            'currentPage'       => $this->request->getVar('page_users') ?? 1,
            'itemPerPage'       => $itemPerPage,
        ];

        return view('users/index', $data);
    }


    /**
```

```
     * Return the editable properties of a resource object
     *
     * @return mixed
     */
    public function edit($id = null)
    {
        $user = $this->userModel->withIdentities()->find($id);

        if (empty($user)) {
            throw new PageNotFoundException('User not found');
        }

        $data = [
            'user'          => $user,
            'validation'    => \Config\Services::validation(),
        ];

        return view('users/edit', $data);
    }

    /**
     * Add or update a model resource, from "posted" properties
     *
     * @return mixed
     */
    public function update($id = null)
    {
        $user = $this->userModel->withIdentities()->find($id);

        if (empty($user)) {
            throw new PageNotFoundException('User not found');
        }

        $username = $user->toArray()['username'];

        $usernameChanged = $username != $this->request->getVar('username');

        if (!$this->validate([
            'username'      => $usernameChanged ? 'required|string|is_unique[users.username]' :
'required|string',
            'email'         => 'required|valid_email|max_length[255]',
            'password' => [
                'label'  => 'Auth.password',
                'rules'  => 'permit_empty|' . Passwords::getMaxLengthRule() . '|strong_password',
                'errors' => [
                    'max_byte' => 'Auth.errorPasswordTooLongBytes',
                ],
            ],
            'password_confirm' => [
                'label' => 'Auth.passwordConfirm',
                'rules' => 'permit_empty|matches[password]',
            ],
        ])) {
            $data = [
                'user'       => $user,
```

```
            'validation' => \Config\Services::validation(),
            'oldInput'   => $this->request->getVar(),
        ];

        return view('users/edit', $data);
    }

    if (!$this->userModel->save(new User([
        'id'       => $id,
        'username' => $this->request->getVar('username'),
        'email'    => $this->request->getVar('email'),
        'password' => $this->request->getVar('password') ?? null,
    ]))) {
        $data = [
            'user'     => $user,
            'validation' => \Config\Services::validation(),
            'oldInput'   => $this->request->getVar(),
        ];

        session()->setFlashdata(['msg' => 'Insert failed']);
        return view('users/create', $data);
    }

    session()->setFlashdata(['msg' => 'Update user successful']);
    return redirect()->to('admin/users');
}

/**
 * Delete the designated resource object from the model
 *
 * @return mixed
 */
public function delete($id = null)
{
    $user = $this->userModel->where('id', $id)->first();

    if (empty($user)) {
        throw new PageNotFoundException('User not found');
    }

    if (!$this->userModel->delete($id)) {
        session()->setFlashdata(['msg' => 'Failed to delete user', 'error' => true]);
        return redirect()->back();
    }

    session()->setFlashdata(['msg' => 'User deleted successfully']);
    return redirect()->to('admin/users');
}
}
```