Web Programming Final Exam

# Alfonsus Ardani Chendrawan

# 2301900555

# Table of Content

1. Initialization
2. Migrations
3. Seeders
4. Factories
5. Models
6. Middleware
7. Routing
8. Controller
9. View
10. Localization
11. Contact

1. Initialization

**Requirement:**

* Laravel 9
* XAMPP (MySQL + Apache)

**Workspace Initialization**

* Create new workspace



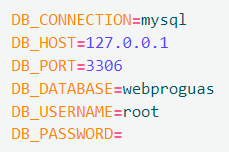
* Connect to MySQL



* Create Database



* Configure .env file

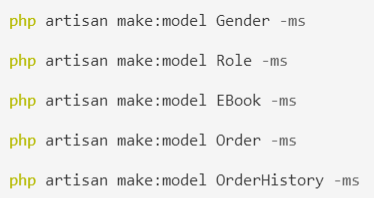


* Serve the website

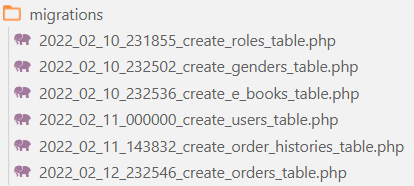


2. Migration

**Create Models, Migrations, and Seeders**

****

**File Structure:**

****

# Migration Files

**Roles Table**

*// 2022\_02\_10\_231855\_create\_roles\_table.php*

    public function **up**()

    {

**Schema**::**create**('roles', function (**Blueprint** $table) {

            $table->**id**();

            $table->**string**("desc",25);

            $table->**timestamps**();

        });

    }

**Genders Table**

*// 2022\_02\_10\_232502\_create\_genders\_table.php*

    public function **up**()

    {

**Schema**::**create**('genders', function (**Blueprint** $table) {

            $table->**id**();

            $table->**string**('desc',25);

            $table->**timestamps**();

        });

    }

**E-Books Table**

*// 2022\_02\_10\_232536\_create\_e\_books\_table.php*

    public function **up**()

    {

**Schema**::**create**('e\_books', function (**Blueprint** $table) {

            $table->**id**();

            $table->**string**('title',255);

            $table->**string**('author',255);

            $table->**longText**('description',255);

            $table->**timestamps**();

        });

    }

**Users Table**

*// 2022\_02\_11\_000000\_create\_users\_table.php*

    public function **up**()

    {

**Schema**::**create**('users', function (**Blueprint** $table) {

            $table->**id**();

            $table->**foreignId**('role\_id');

            $table->**foreign**('role\_id')->**references**('id')->**on**('roles')->**onDelete**('cascade');

            $table->**foreignId**('gender\_id');

            $table->**foreign**('gender\_id')->**references**('id')->**on**('genders')->**onDelete**('cascade');

            $table->**string**('first\_name',25);

            $table->**string**('middle\_name',25);

            $table->**string**('last\_name',25);

            $table->**string**('email',50)->**unique**();

            $table->**string**('password');

            $table->**string**('display\_picture\_link',255);

            $table->**integer**('deleted\_flag')->**default**(0);

            $table->**date**('modified\_at');

            $table->**string**('modified\_by',255)->**default**('migrated');

            $table->**timestamps**();

        });

    }

**Order Histories Table**

*// 2022\_02\_11\_143832\_create\_order\_histories\_table.php*

    public function **up**()

    {

**Schema**::**create**('order\_histories', function (**Blueprint** $table) {

            $table->**id**();

            $table->**foreignId**('account\_id');

            $table->**foreign**('account\_id')->**references**('id')->**on**('users')->**onDelete**('cascade');

            $table->**foreignId**('ebook\_id');

            $table->**foreign**('ebook\_id')->**references**('id')->**on**('e\_books')->**onDelete**('cascade');

            $table->**date**('order\_date');

            $table->**timestamps**();

        });

    }

**Orders Table**

*// 2022\_02\_12\_232546\_create\_orders\_table.php*

    public function **up**()

    {

**Schema**::**create**('orders', function (**Blueprint** $table) {

            $table->**id**();

            $table->**foreignId**('account\_id');

            $table->**foreign**('account\_id')->**references**('id')->**on**('users')->**onDelete**('cascade');

            $table->**foreignId**('ebook\_id');

            $table->**foreign**('ebook\_id')->**references**('id')->**on**('e\_books')->**onDelete**('cascade');

            $table->**date**('order\_date');

            $table->**timestamps**();

        });

    }

**Migrating**

****

3. Seeders

**Only table genders, roles, and few records of users are generated manually. The rest is done by Factories.**

**On DatabaseSeeder.php**

    public function run()

    {

*$this*->call([

            EBookSeeder::class,

            GenderSeeder::class,

            OrderHistorySeeder::class,

            OrderSeeder::class,

            RoleSeeder::class,

        ]);

        DB::table('users')->insert([

            'role\_id'              => '1',

            'gender\_id'            => '1',

            'first\_name'           => 'Rock',

            'middle\_name'          => 'Dwayne',

            'last\_name'            => 'Johnson',

            'email'                => 'member@a.com',

            'password'             => Hash::make('password'),

            'display\_picture\_link' => 'images/person.jpg',

            'deleted\_flag'         => '0',

            'modified\_at'          => Date::now(),

        ]);

        DB::table('users')->insert([

            'role\_id'              => '2',

            'gender\_id'            => '1',

            'first\_name'           => 'Bill',

            'middle\_name'          => 'Microsoft',

            'last\_name'            => 'Gates',

            'email'                => 'admin@a.com',

            'password'             => Hash::make('password'),

            'display\_picture\_link' => 'images/person2.jpg',

            'deleted\_flag'         => '0',

            'modified\_at'          => Date::now(),

        ]);

        User::factory()->count(10)->create();

    }

**EBookSeeder::class**

    public function run()

    {

        EBook::factory()->count(10)->create();

    }

**RoleSeeder::class**

    public function run()

    {

        DB::table('roles')->insert([

            ['desc' => 'member'],

            ['desc' => 'admin']

        ]);

    }

**GenderSeeder::class**

    public function run()

    {

        DB::table('genders')->insert([

            ['desc' => 'male'],

            ['desc' => 'female']

        ]);

    }

4. Factories

**There are two factories that are used to generate the records. User factory, and EBook factory. Each of these factory uses the faker module (built-in) to generate datas automatically.**

**UserFactory.php**

*public* function definition()

    {

**return** [

            'role\_id'           **=>** '1',

            'gender\_id'         **=>** $*this***->***faker***->**randomElement(['1','2']),

            'first\_name'        **=>** $*this***->***faker***->**firstName(),

            'middle\_name'       **=>** $*this***->***faker***->**firstName(),

            'last\_name'         **=>** $*this***->***faker***->**lastName(),

            'email'             **=>** $*this***->***faker***->**unique()**->**safeEmail(),

            'password'          **=>** Hash**::**make($*this***->***faker***->**words(1,true)),

            'display\_picture\_link' **=>** 'images/Person-placeholder.jpg',

            'deleted\_flag'      **=>** '0',

            'modified\_at'       **=>** *Date***::**now(),

*// 'email' => $this->faker->unique()->safeEmail(),*

*// 'email\_verified\_at' => now(),*

*// 'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', // password*

*// 'remember\_token' => Str::random(10),*

        ];

    }

**EBookFactory.php**

*public* function definition()

    {

**return** [

            'title'**=>**$*this***->***faker***->**sentence(3, true),

            'author'**=>**$*this***->***faker***->***name*,

            'description'**=>**$*this***->***faker***->**paragraph(10),

        ];

    }

**Calling the factories**

  EBook**::**factory()**->**count(10)**->**create();

  User**::**factory()**->**count(10)**->**create();

5. Model

**After setting up the database, we also have to setup the eloquent ORM using HasFactory to make programming more elegant.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EBook.php**    class EBook *extends* *Model*    {  **use** HasFactory;  *public* function orders(){  **return** $*this***->**hasMany(Order**::class**);        }  *public* function orderhistories(){  **return** $*this***->**  hasMany(OrderHistory**::class**,'ebook\_id');        }    } | | | **Gender.php**      class Gender *extends* *Model*      {  **use** HasFactory;  *public* function users(){  **return** $*this***->**hasMany(User**::class**);          }      } | |
| **Order.php**    class Order *extends* *Model*    {  **use** HasFactory;  *public* function user(){  **return** $*this*  **->**belongsTo(User**::class**,'account\_id');        }  *public* function book(){  **return** $*this*  **->**belongsTo(EBook**::class**,'ebook\_id');        }    } | | **OrderHistory.php**    class OrderHistory *extends* *Model*    {  **use** HasFactory;  *public* function user(){  **return** $*this*  **->**belongsTo(User**::class**,'account\_id');        }  *public* function book(){  **return** $*this*  **->**belongsTo(EBook**::class**,'ebook\_id');        }  } | |
| **Role.php**    class Role *extends* *Model*    {  **use** HasFactory;  *public* function users()        {  **return** $*this***->**hasMany(User**::class**);        }    } | **User.php**    *public* function role()      {  **return** $*this***->**belongsTo(Role**::class**);      }  *public* function gender()      {  **return** $*this***->**belongsTo(Gender**::class**);      }  *public* function orders()      {  **return** $*this***->**hasMany(Order**::class**,'account\_id');      }  *public* function orderhistories()      {  **return** $*this***->**hasMany(OrderHistory**::class**);      } | | |

6. Middleware

**There will be two middleware used for this project.** One is for role access and authoring user to pages**.** Second one is to handle localization for all requests

**Creating Middleware**

**  
**

**For the Access Middleware, first we want to know what it does and the logic can be determined by the overall requirement from the question.**

**The question wants to add authorization system which required the website to do the following:**

* Redirect User to Home if user is authenticated but not authorized to certain page
* Redirect User to Index if user is not authenticated to certain page
* Continue if User is authenticated to a certain page that needs authentication

**Hence I came into the following logic for Access.php. It acts as a route middleware hence I also add the middleware in $routeMiddleware in Kernel.php**

    public function handle(Request $request, Closure $next, ... $roles)

    {

        $user = Auth::user();

        // *if its meant for people who are not authorized*

        if( (empty($roles) && $user==null) ){

            return $next($request);

        // *if its meant for authorized page*

        }elseif($user != null){

            // *if a page is meant for someone*

            foreach($roles as $role){

                if($role == $user->role->deAsc){

                    return $next($request);

                }

            }

            // *if a page is not meant for someone*

            return redirect('home');

        }

        // *if not authorized*

        return redirect('/');

    }

}

*Source:* [*https://stackoverflow.com/questions/43901719/laravel-middleware-with-multiple-roles*](https://stackoverflow.com/questions/43901719/laravel-middleware-with-multiple-roles)

**And then I can use the following syntax to group the routing with middleware.**

 Route::middleware('access:member,admin')->group(function () {

      ...

  });

**For Localization, It will be discussed in the Localization Section**

7. Routing

**For the Routing, it is divided into controllers and each access types.**

  Route::controller(PageController::class)->group(function (){

      Route::middleware('access')->group(function () {

          Route::get   ('/',               'viewindex');

          Route::get   ('/signup',         'viewsignup');

          Route::get   ('/login',          'viewlogin');

          Route::post  ('/signup',         'signup');

          Route::post  ('/login',          'login');

      });

      Route::middleware('access:member,admin')->group(function () {

          Route::get   ('/home',           'viewhome');

          Route::get   ('/book/{id}',      'viewbookdetail');

          Route::get   ('/cart',           'viewcart');

          Route::get   ('/success',        'viewsuccess');

          Route::get   ('/profile',        'viewprofile');

          Route::get   ('/saved',          'viewsaved');

          Route::get   ('/logout',         'logout');

          Route::post  ('/book/{id}/rent', 'rent');

          Route::delete('/cart/{id}',      'deletecartitem');

          Route::post  ('/cart/submit',    'submitcart');

          Route::post  ('/profile',        'updateprofile');

      });

      Route::get('{locale}/locale', 'changelocale');

  });

  Route::controller(AdminController::class)->group(function (){

      Route::middleware('access:admin')->group(function () {

          Route::get   ('/maccount',            'viewmanageaccount');

          Route::get   ('/user/{id}/updaterole','viewupdaterole');

          Route::delete('/user/{id}/delete',    'deleteuser');

          Route::put   ('/user/{id}/updaterole','updateroleuser');

      });

  });

  Route::redirect('/', '/en/');

**The routes that is using middleware ‘access’ are only checking if the user is NOT authenticated to access the page. The routes that is using middleware ‘access:member,admin’ are for the pages that can be accessed by members and admin. The routes that is using middleware ‘access:admin’ are for pages that can be accessed ONLY by admins.**

8. Controllers

**I use 2 controllers. One for general purpose controllers, the other one is for admin pages.**

# PageController.php

    // shows the index page

    public function viewindex()

    {

        return view('index');

    }

    // shows the signup page

    public function viewsignup()

    {

        $roles = Role::all();

        $genders = Gender::all();

        return view('signup',compact('roles','genders'));

    }

    // attempt to sign up the user

    public function signup(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'firstname'     => 'required|alpha\_num|max:25',

            'middlename'    => 'required|alpha\_num|max:25',

            'lastname'      => 'required|alpha\_num|max:25',

            'gender'        => 'required',

            'email'         => 'required|email|unique:users,email',

            'role'          => 'required|exists:roles,id',

            'password'      => 'required|min:8|regex:/.\*[0-9].\*/i',

            'displaypicture'=> 'required|file|mimes:jpg,bmp,png',

        ],[

            'password.regex' => 'The password needs to atleast have 1 number',

        ],[

            'displaypicture' => 'display picture',

            'firstname'      => 'first name',

            'middlename'     => 'middle name',

            'lastname'       => 'last name',

        ]);

        if($validator->fails()){

            return back()

            ->withErrors($validator)

            ->withInput();

        }

        $validated = $validator->validated();

        $user = new User;

        $user->first\_name   = $validated['firstname'];

        $user->middle\_name  = $validated['middlename'];

        $user->last\_name    = $validated['lastname'];

        $user->gender\_id    = $validated['gender'];

        $user->email        = $validated['email'];

        $user->password     = Hash::make($request->password);

        $user->role\_id      = $validated['role'];

        $user->modified\_at  = Date::now();

        $user->modified\_by  = "user";

        $file = $request->file('displaypicture');

        $imageName = time().".".$file->getClientOriginalExtension();

        Storage::putFileAs('public/images',$file, $imageName);

        $user->display\_picture\_link = 'images/'.$imageName;

        $user->save();

        return redirect('/');

    }

    // shows the login page

    public function viewlogin()

    {

        return view('login');

    }

    // attempt to authenticate the user

    public function login(Request $request)

    {

        // TODO:

        // validation

        // attempt

        $validator = Validator::make($request->all(), [

            'email' => 'required|email',

            'password' => 'required'

        ]);

        if($validator->fails()){

            return back()->withErrors($validator)->withInput();

        }

        if(Auth::attempt([

            'email' => $request->email, 'password' => $request->password

        ])){

            $request->session()->regenerate();

            return redirect()->intended('home');

        }

        return back()->withErrors([

            'email' => 'The provided credentials do not match our records.'

        ]);

    }

    // -------------------------

    // PAGES FOR REGISTERED MEMBER

    // -------------------------

    // show the home page as a registered user

    public function viewhome()

    {

        $books = EBook::all();

        return view('home',compact('books'));

    }

    // Show the detail of the book

    public function viewbookdetail(Request $request)

    {

        // TODO:

        // get book from model

        // pass book model to view

        $book = EBook::find($request->id);

        return view('bookdetail', compact('book'));

    }

    // Rent the selected book

    public function rent(Request $request)

    {

        // TODO:

        // add selected book to the current order

        $order = new Order;

        $order->account\_id = Auth::user()->id;

        $order->ebook\_id = $request->id;

        $order->order\_date = Date::now();

        $order->save();

        return redirect('/cart');

    }

    // View the cart of the user

    public function viewcart(Request $request)

    {

        return view('cart');

    }

    // Delete cart item from user's cart

    public function deletecartitem(Request $request)

    {

        Order::destroy($request->id);

        return redirect()->back();

    }

    // Proceed user's cart to checkout

    public function submitcart(Request $request)

    {

        // delete user cart and move it to order history

        foreach (Auth::user()->orders as $order) {

            $orderhist = new OrderHistory;

            $orderhist->account\_id = $order->account\_id;

            $orderhist->ebook\_id = $order->ebook\_id;

            $orderhist->order\_date = Date::now();

            $orderhist->save();

            $order->delete();

        }

        return view('success');

    }

    // View user's profile

    public function viewprofile(Request $request)

    {

        $user = Auth::user();

        $roles = Role::all();

        $genders = Gender::all();

        return view('profile', compact('user','roles','genders'));

    }

    // Update's the user's profile

    public function updateprofile(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'firstname'     => 'required|alpha\_num|max:25',

            'middlename'    => 'alpha\_num|max:25',

            'lastname'      => 'required|alpha\_num|max:25',

            'gender'        => 'required',

            'email'         => 'required|email',

            'password'      => 'required|min:8|regex:/.\*[0-9].\*/i',

            'displaypicture'=> 'required|file|mimes:jpg,bmp,png',

        ],[

            'password.regex' => 'The password needs to atleast have 1 number',

        ],[

            'displaypicture' => 'display picture',

            'firstname'      => 'first name',

            'middlename'     => 'middle name',

            'lastname'       => 'last name',

        ]);

        if($validator->fails()){

            return back()

            ->withErrors($validator)

            ->withInput();

        }

        $validated = $validator->validated();

        $user = User::find(Auth::user()->id);

        $user->first\_name   = $validated['firstname'];

        $user->middle\_name  = $validated['middlename'];

        $user->last\_name    = $validated['lastname'];

        $user->gender\_id    = $validated['gender'];

        if(Auth::user()->email != $request->email){

            $validator = Validator::make($request->all(), [

                'email' => 'required|email|unique:users,email'

            ]);

            if($validator->fails()){

                return back()

                ->withErrors($validator)

                ->withInput();

            }

            $user->email        = $validated['email'];

        }

        $user->password     = Hash::make($request->password);

        $user->modified\_at  = Date::now();

        $user->modified\_by  = "user";

        $file = $request->file('displaypicture');

        $imageName = time().".".$file->getClientOriginalExtension();

        Storage::putFileAs('public/images',$file, $imageName);

        $user->display\_picture\_link = 'images/'.$imageName;

        $user->save();

        return view('saved');

    }

    // Log the authenticated user out

    public function logout(Request $request)

    {

        Auth::logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return view('logout');

    }

    // -------------------------

    // CHANGE LOCALE

    // -------------------------

    public function changelocale(Request $request){

        Cookie::queue('locale',$request->locale);

        App::setlocale($request->locale);

        return redirect()->back();

    }

# AdminController.php

    // Show the page to manage user accounts

    public function viewmanageaccount(Request $request)

    {

        $users = User::all();

        return view('admin.manageaccount',compact('users'));

    }

    // Delete the user

    public function deleteuser(Request $request)

    {

        $user = User::find($request->id);

        $user->delete();

        return redirect()->back();

    }

    // Show the page to update the selected user's role

    public function viewupdaterole(Request $request)

    {

        $user = User::find($request->id);

        $roles = Role::all();

        return view('admin.changerole',compact('user','roles'));

    }

    // Update the selected user's role

    public function updateroleuser(Request $request)

    {

        $user = User::find($request->id);

        $user->role\_id = $request->role;

        $user->save();

        return redirect()->back();

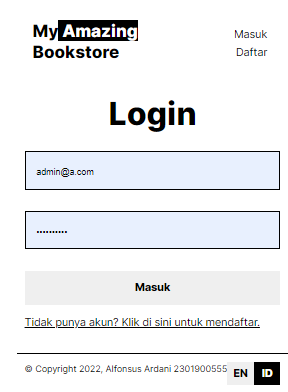
    }

9. View

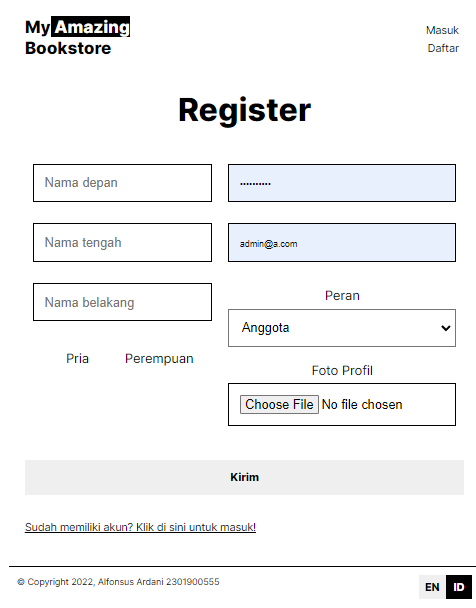
* Index Page



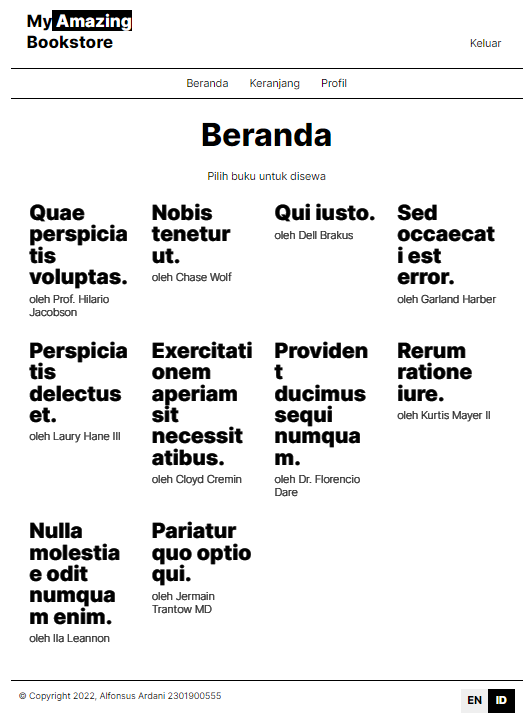
* Login Page



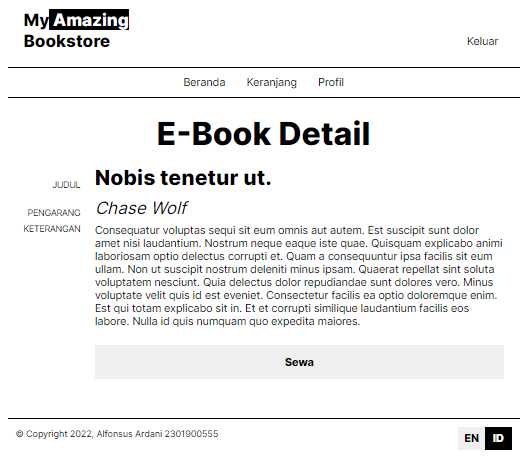
* Register Page



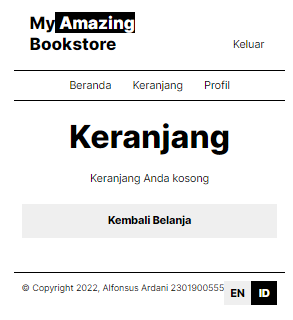
* Home Page



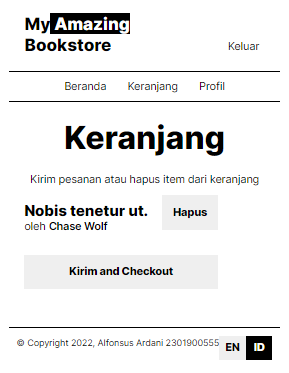
* Book Detail Page



* Cart Page



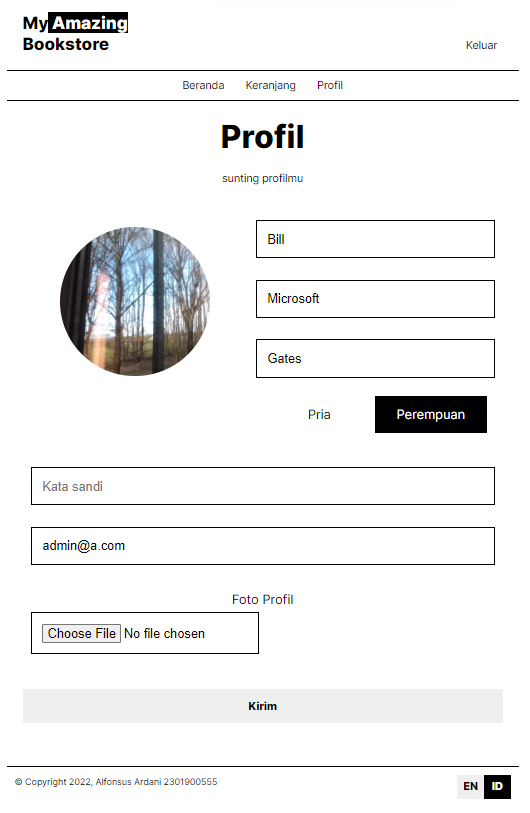
* Cart Page when rented



* Cart Checkout Confirmation Page



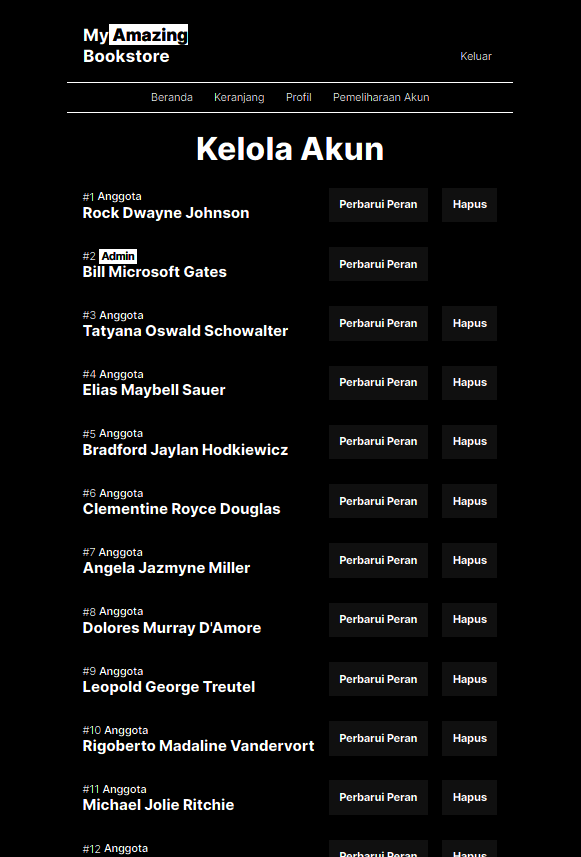
* Profile Page



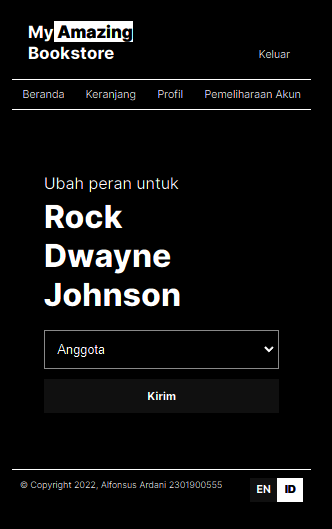
* Profile Edit Confirmation



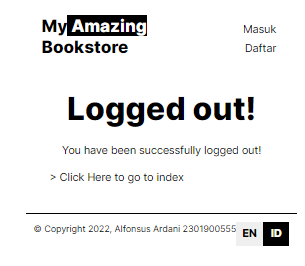
* Account Maintenance Page



* Update Role Page



* Log out Confirmation



10. Localization

**For Localization, you need the language file, the middleware and the button to change the locale.**

**lang/id.json (The Language File)**

{

    "Log-in":"Masuk",

    "Register":"Daftar",

    "Find":"Cari",

    "and":"dan",

    "Rent":"Sewa",

    "your E-Book":"E-bookmu",

    "Here!":"Disini!",

    "I already have an account, sign me in!":"Saya sudah memiliki akun, ayo masuk!",

    "First Name":"Nama depan",

    "Middle Name":"Nama tengah",

    "Last Name":"Nama belakang",

    "Password":"Kata sandi",

    "E-mail":"Surel",

    "Email Address":"Alamat email",

    "Login":"Masuk",

    "Don't have an account? Click here to sign up.":"Tidak punya akun? Klik di sini untuk mendaftar.",

    "Role":"Peran",

    "Male":"Pria",

    "Female":"Perempuan",

    "Member":"Anggota",

    "Admin":"Admin",

    "Profile Picture":"Foto Profil",

    "Choose File":"Pilih File",

    "No file chosen":"No file chosen",

    "Submit":"Kirim",

    "Already have an account? Click here to log in!":"Sudah memiliki akun? Klik di sini untuk masuk!",

    "Home":"Beranda",

    "Cart":"Keranjang",

    "Profile":"Profil",

    "Account Maintenance":"Pemeliharaan Akun",

    "Log-out":"Keluar",

    "Select a book for rent":"Pilih buku untuk disewa",

    "Title":"Judul",

    "Author":"Pengarang",

    "Description":"Keterangan",

    "Submit order or delete item from cart":"Kirim pesanan atau hapus item dari keranjang",

    "Delete":"Hapus",

    "Submit and Checkout":"Kirim dan Checkout",

    "Success!":"Sukses!",

    "Your cart has been successfully checked out and saved!":"Keranjang Anda telah berhasil diperiksa dan disimpan!",

    "Click Here to go back home":"Klik Di Sini untuk kembali ke rumah",

    "Edit your profile":"sunting profilmu",

    "Saved!":"Tersimpan!",

    "Your profile has been successfully saved!":"Profil Anda telah berhasil disimpan!"

    ,"Manage Account":"Kelola Akun"

    ,"Update Role":"Perbarui Peran"

    ,"Change the role for":"Ubah peran untuk"

    ,"by":"oleh"

    ,"Your cart is empty":"Keranjang Anda kosong"

    ,"Back to Shopping":"Kembali Belanja"

}

**The Button, the language selector**

        <a href="{{url('en/locale')}}"><div class="localeselector">

            <button @if (App::currentlocale()=='en')

                class="selected"

            @endif>EN</button></a>

        <a href="{{url('id/locale')}}"><button @if (App::currentlocale()=='id')

            class="selected"

            @endif>ID</button></a>

**Which sends request to route**

      Route::get('{locale}/locale', 'changelocale');

**Which calls the changelocale function**

    public function changelocale(Request $request){

        Cookie::queue('locale',$request->locale);

        App::setlocale($request->locale);

        return redirect()->back();

    }

**The Middleware that changes the locale everytime a route actions is called**

    public function handle(Request $request, Closure $next)

    {

        if(Cookie::has('locale')){

            App::setLocale(Cookie::get('locale'));

        }else{

            App::setLocale(App::currentLocale());

            Cookie::queue('locale',App::currentLocale(), 2147483647);

        }

        return $next($request);

    }

11. Contact

Private Email: [Alfonsusac@gmail.com](mailto:Alfonsusac@gmail.com)

Binusian Email: [Alfonsus.chendrawan@binus.ac.id](mailto:Alfonsus.chendrawan@binus.ac.id)

Phone/Whatsapp: +62 812 600 3290

LineID: alfonsusac

LINK:

<https://binusianorg-my.sharepoint.com/personal/alfonsus_chendrawan_binus_ac_id/_layouts/15/guestaccess.aspx?docid=0620f698c89134abbbb93f5419a98d867&authkey=AbbkSobfbfo1RIQcVanhAoE&e=3CTbfw>