Web Programming Final Exam

Alfonsus Ardani Chendrawan
2301900555

Table of Content

1. Initialization

Requirement:

- Laravel 9
- XAMPP (MySQL + Apache)

Workspace Initialization

- **Create new workspace**

```
composer create-project laravel/laravel webproguas
```

- **Connect to MySQL**

```
xampp\mysql\bin\mysql.exe -h localhost -u root -p
```

- **Create Database**

```
MariaDB [(none)]> CREATE DATABASE webproguas;
```

- **Configure .env file**

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=webproguas
DB_USERNAME=root
DB_PASSWORD=
```

- **Serve the website**

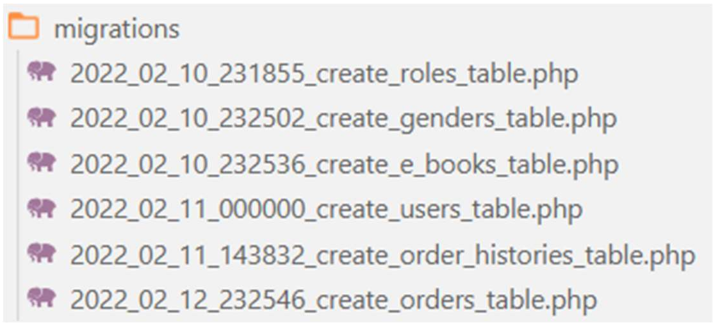
```
php artisan serve
```

2. Migration

Create Models, Migrations, and Seeders

```
php artisan make:model Gender -ms

php artisan make:model Role -ms

php artisan make:model EBook -ms

php artisan make:model Order -ms

php artisan make:model OrderHistory -ms
```

File Structure:



```
migrations
    2022_02_10_231855_create_roles_table.php
    2022_02_10_232502_create_genders_table.php
    2022_02_10_232536_create_e_books_table.php
    2022_02_11_000000_create_users_table.php
    2022_02_11_143832_create_order_histories_table.php
    2022_02_12_232546_create_orders_table.php
```

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

```php
// 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();
    });
}
```

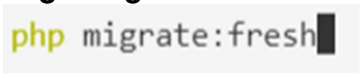
Order Histories Table

```php
// 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

```php
// 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



```
php migrate:fresh
```

3. Seeders

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

On DatabaseSeeder.php

```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

```php
public function run()
{
    EBook::factory()->count(10)->create();
}
```

RoleSeeder::class

```php
public function run()
{
    DB::table('roles')->insert([
        ['desc' => 'member'],
        ['desc' => 'admin']
    ]);
}
```

GenderSeeder::class

```php
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

```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

```php
public function definition()
{
    return [
        'title'=>$this->faker->sentence(3, true),
        'author'=>$this->faker->name,
        'description'=>$this->faker->paragraph(10),
    ];
}
```

Calling the factories

```php
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

```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

```php
class Gender extends Model
{
    use HasFactory;
    public function users(){
        return $this->hasMany(User::class);
    }
}
```

Order.php

```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

```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

```php
class Role extends Model
{
    use HasFactory;
    public function users()
    {
      return $this->hasMany(User::class);
    }
}
```

User.php

```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

```
php artisan make:Middleware Access
```

```
php artisan make:Middleware LocaleGlobal
```

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*

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

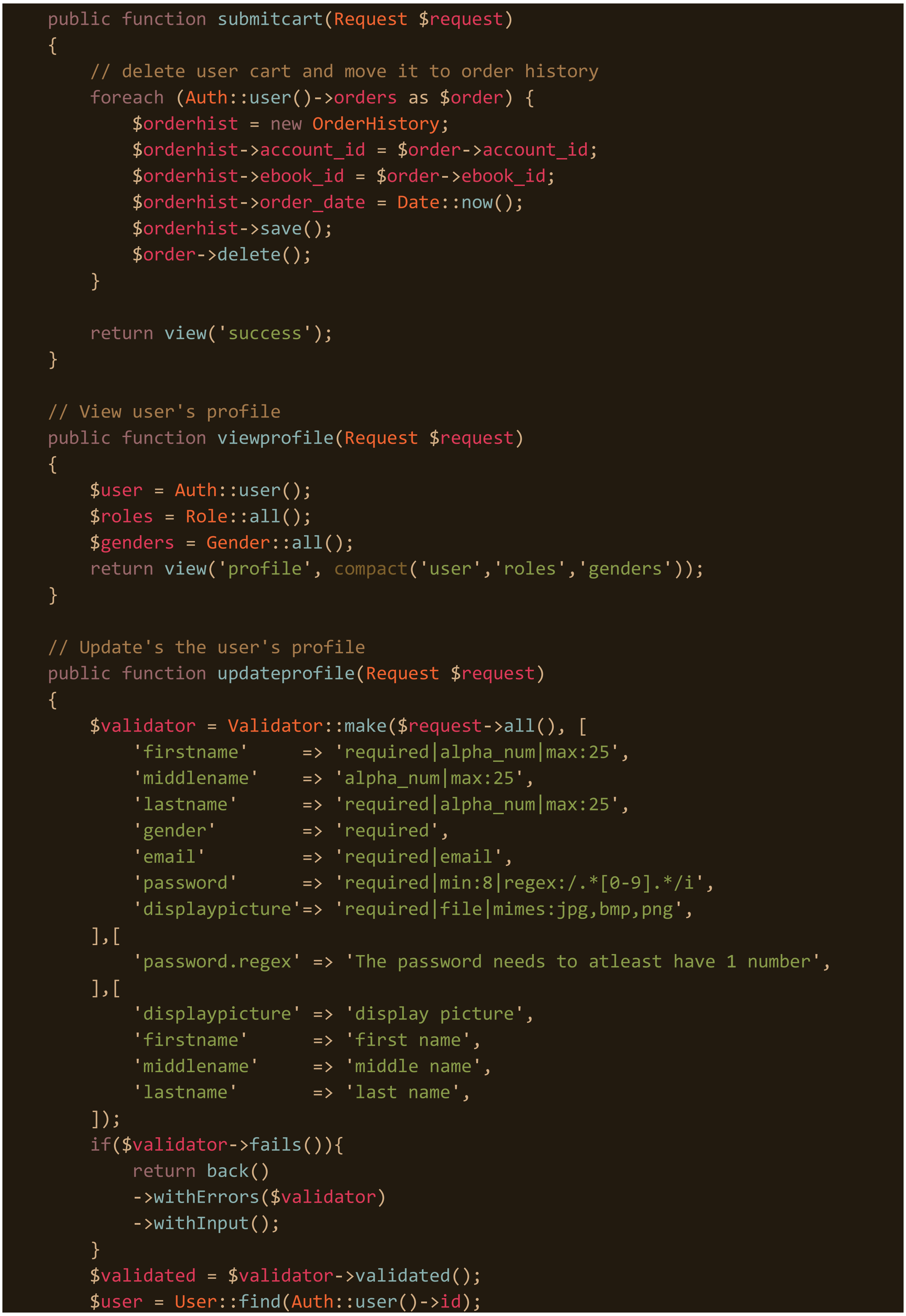
I use 2 controllers. One for general purpose controllers, the other one is for admin pages.

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'];
```

```php
        $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
```



```php
    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');
    }
```

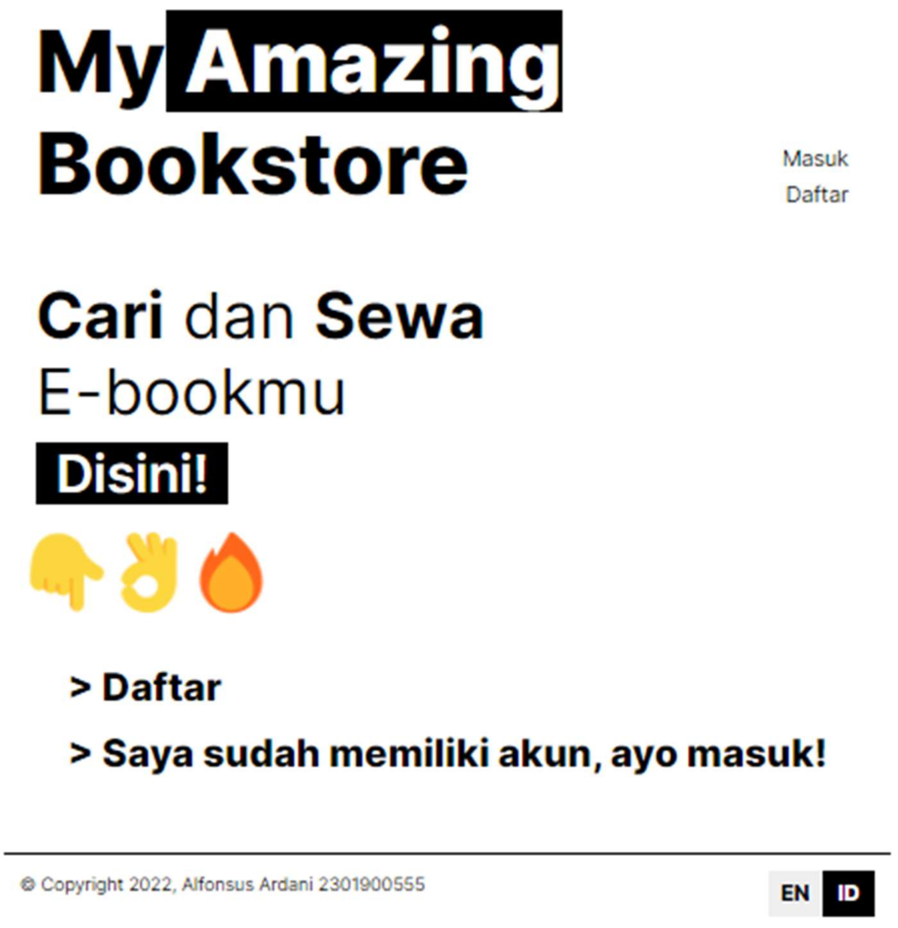
```php
// -------------------------
// CHANGE LOCALE
// -------------------------

public function changelocale(Request $request){
    Cookie::queue('locale',$request->locale);
    App::setlocale($request->locale);
    return redirect()->back();
}
```

AdminController.php

```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**



My Amazing Bookstore

Masuk
Daftar

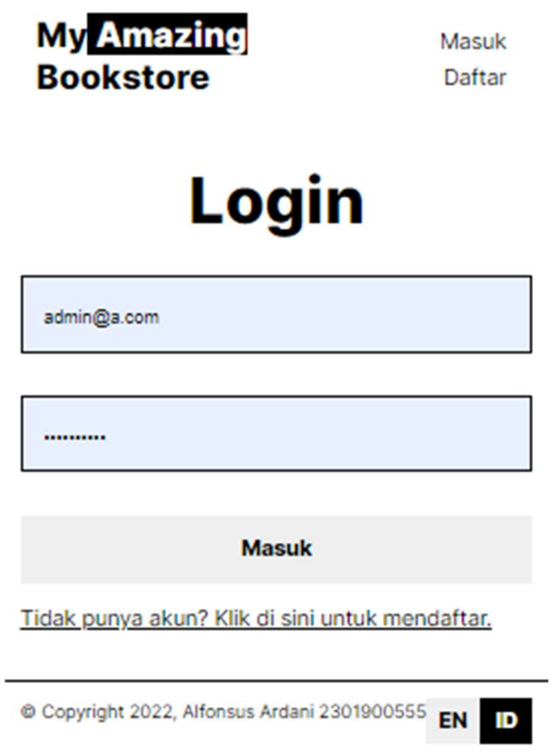
Cari dan Sewa E-bookmu Disini!

👇👌🔥

> Daftar
> Saya sudah memiliki akun, ayo masuk!

© Copyright 2022, Alfonsus Ardani 2301900555

- **Login Page**



My Amazing Bookstore

Masuk

Daftar

Login

admin@a.com

....

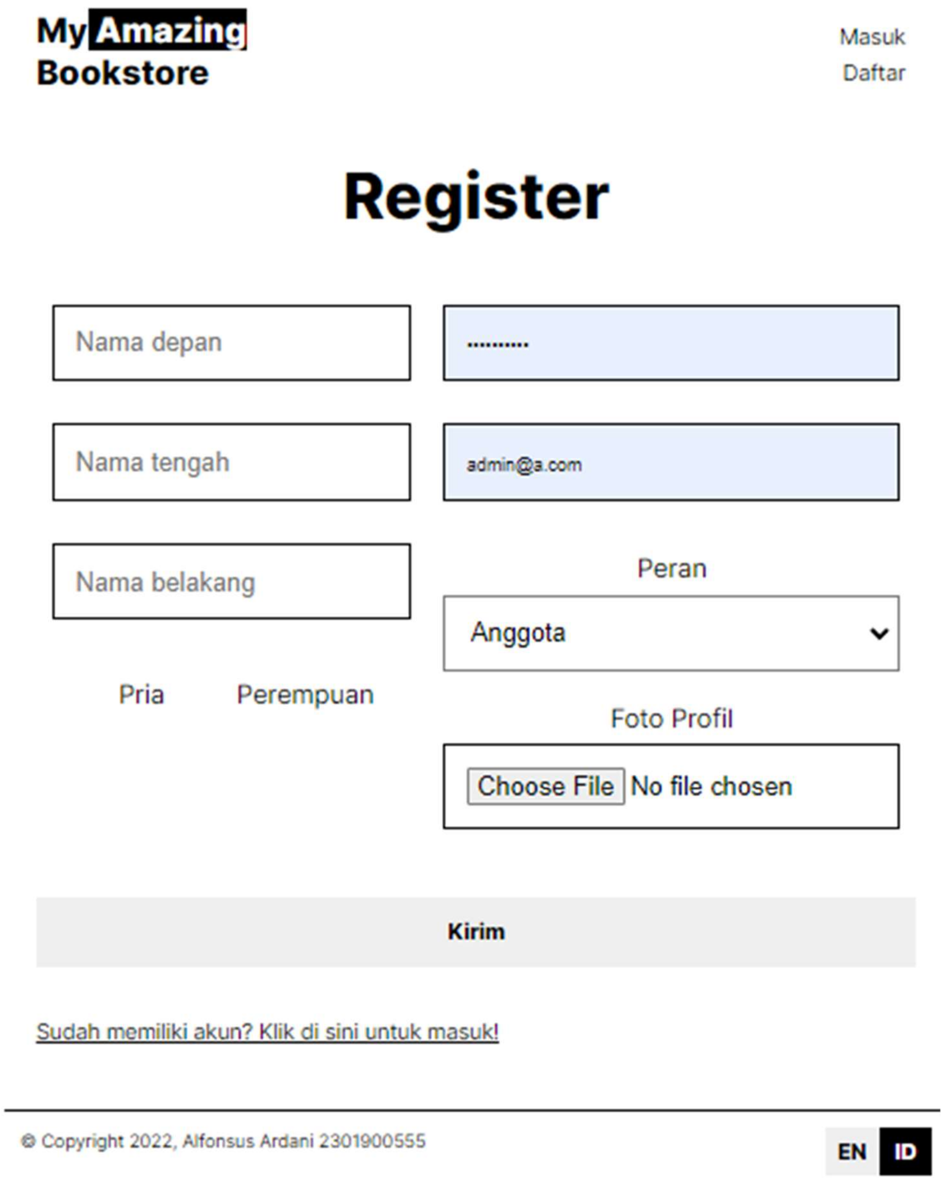
Masuk

Tidak punya akun? Klik di sini untuk mendaftar.

© Copyright 2022, Alfonsus Ardani 2301900555

EN ID

- **Register Page**



My Amazing Bookstore

Masuk

Daftar

Register

Nama depan

....

Nama tengah

admin@a.com

Nama belakang

Peran

Anggota

Pria Perempuan

Foto Profil

Choose File No file chosen

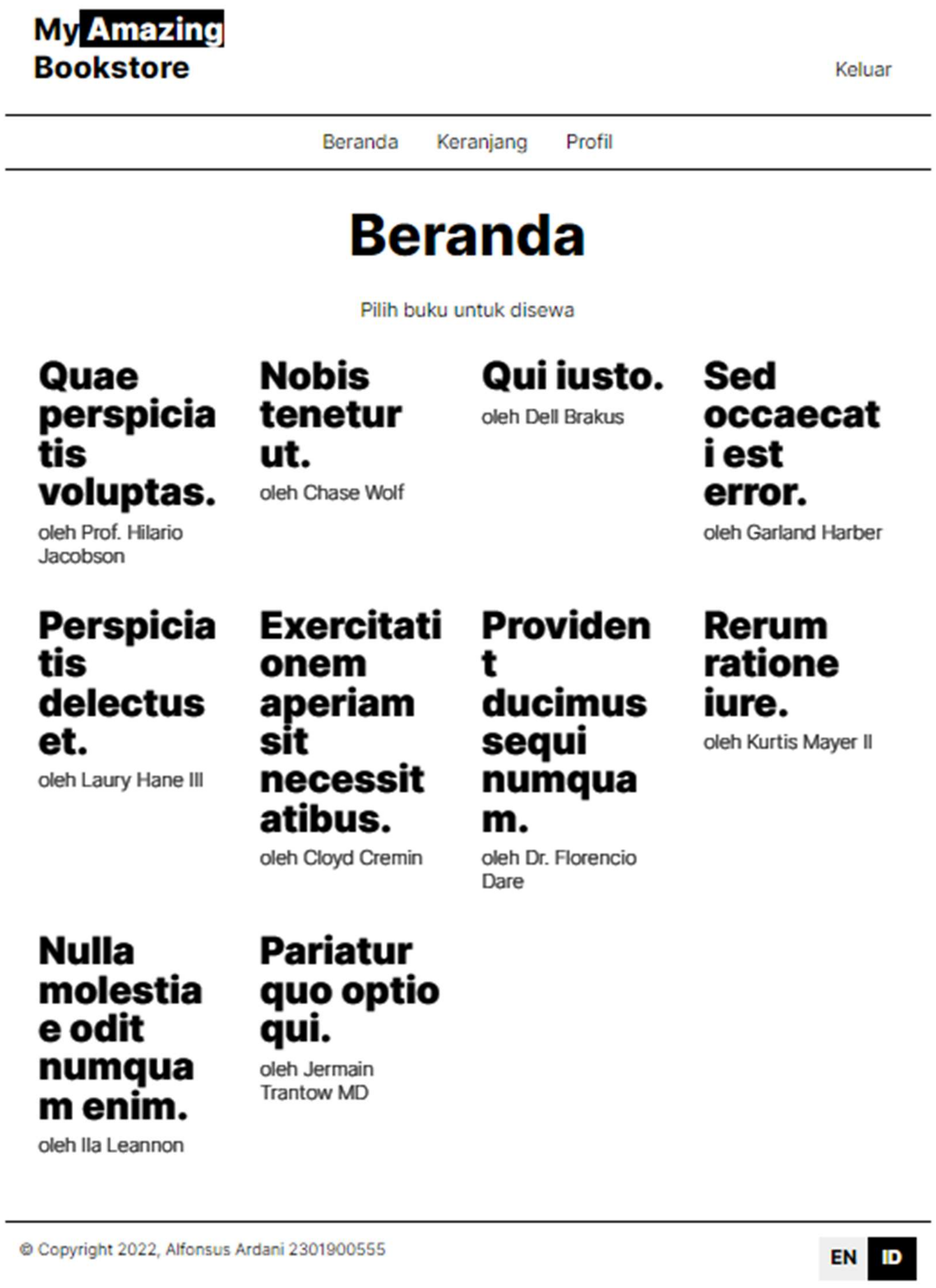
Kirim

Sudah memiliki akun? Klik di sini untuk masuk!

© Copyright 2022, Alfonsus Ardani 2301900555

EN ID

- **Home Page**



My Amazing Bookstore

Keluar

Beranda Keranjang Profil

Beranda

Pilih buku untuk disewa

Quae perspiciatis voluptas.
oleh Prof. Hilario Jacobson

Nobis tenetur ut.
oleh Chase Wolf

Qui iusto.
oleh Dell Brakus

Sed occaecati est error.
oleh Garland Harber

Perspiciatis delectus et.
oleh Laury Hane III

Exercitationem aperiam sit necessitatibus.
oleh Cloyd Cremin

Provident ducimus sequi numquam.
oleh Dr. Florencio Dare

Rerum ratione iure.
oleh Kurtis Mayer II

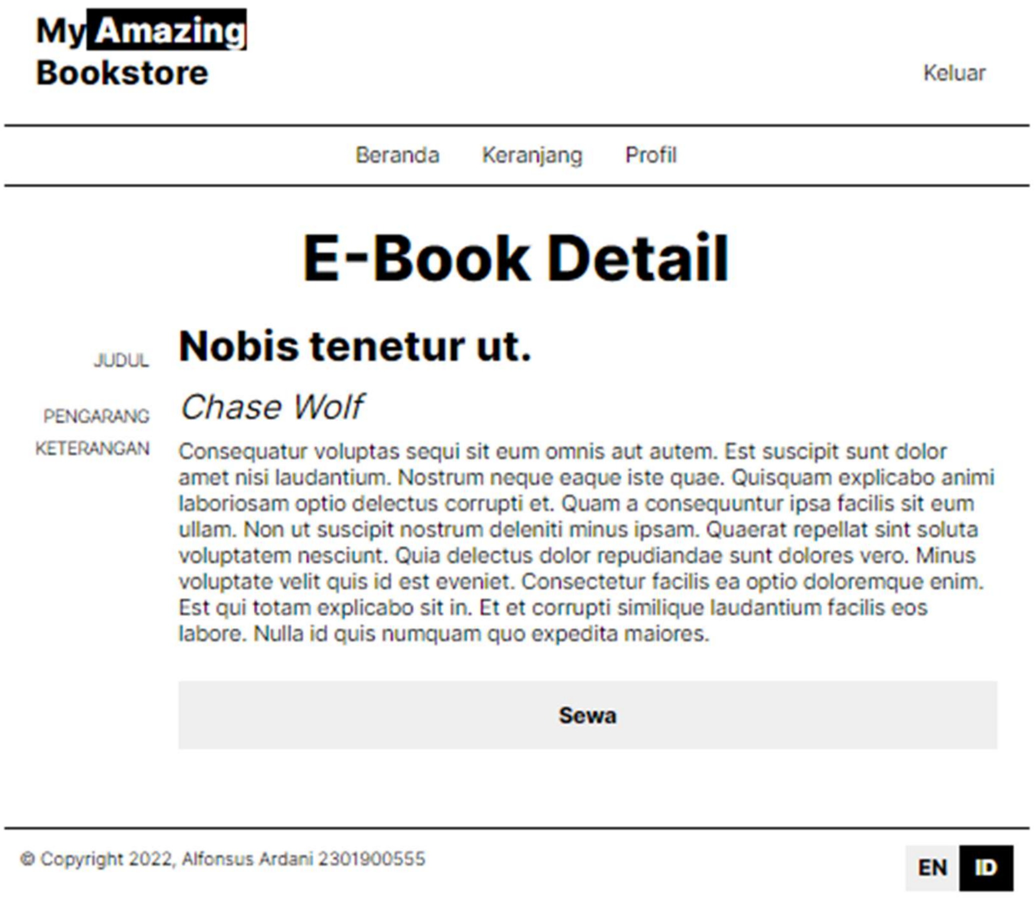
Nulla molestiae odit numquam enim.
oleh Ila Leannon

Pariatur quo optio qui.
oleh Jermain Trantow MD

© Copyright 2022, Alfonsus Ardani 2301900555

EN ID

- **Book Detail Page**



My Amazing Bookstore

Keluar

Beranda Keranjang Profil

E-Book Detail

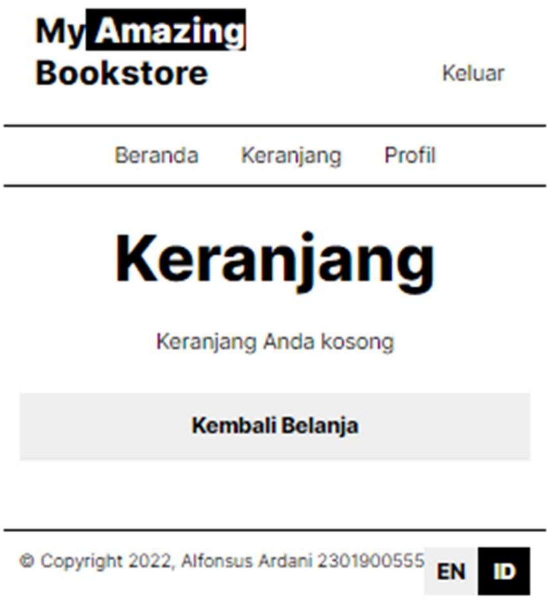
JUDUL	**Nobis tenetur ut.**
PENGARANG	*Chase Wolf*
KETERANGAN	Consequatur voluptas sequi sit eum omnis aut autem. Est suscipit sunt dolor amet nisi laudantium. Nostrum neque eaque iste quae. Quisquam explicabo animi laboriosam optio delectus corrupti et. Quam a consequuntur ipsa facilis sit eum ullam. Non ut suscipit nostrum deleniti minus ipsam. Quaerat repellat sint soluta voluptatem nesciunt. Quia delectus dolor repudiandae sunt dolores vero. Minus voluptate velit quis id est eveniet. Consectetur facilis ea optio doloremque enim. Est qui totam explicabo sit in. Et et corrupti similique laudantium facilis eos labore. Nulla id quis numquam quo expedita maiores.

Sewa

© Copyright 2022, Alfonsus Ardani 2301900555

EN ID

- **Cart Page**


My Amazing Bookstore

Keluar

Beranda Keranjang Profil

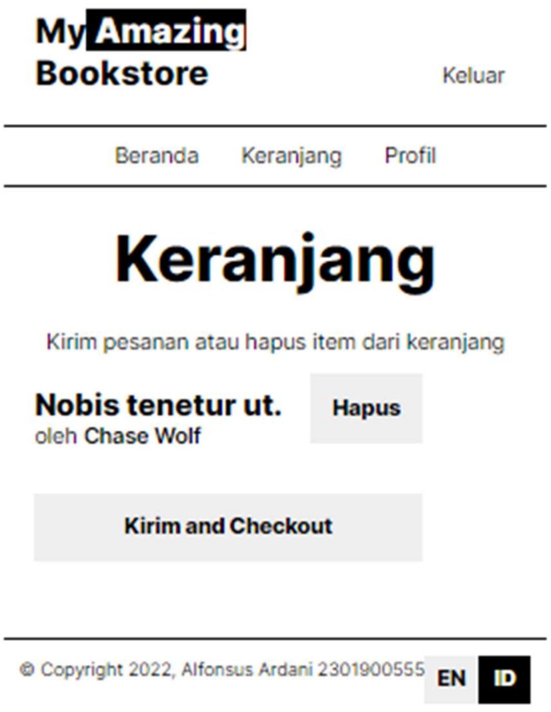
Keranjang

Keranjang Anda kosong

Kembali Belanja

© Copyright 2022, Alfonsus Ardani 2301900555 EN ID

- **Cart Page when rented**


My Amazing Bookstore

Keluar

Beranda Keranjang Profil

Keranjang

Kirim pesanan atau hapus item dari keranjang

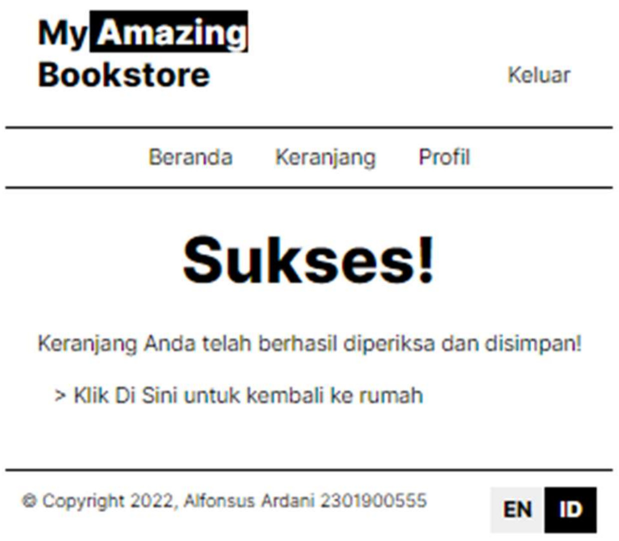
Nobis tenetur ut.
oleh Chase Wolf

Hapus

Kirim and Checkout

© Copyright 2022, Alfonsus Ardani 2301900555 EN ID

- **Cart Checkout Confirmation Page**


My Amazing Bookstore

Keluar

Beranda Keranjang Profil

Sukses!

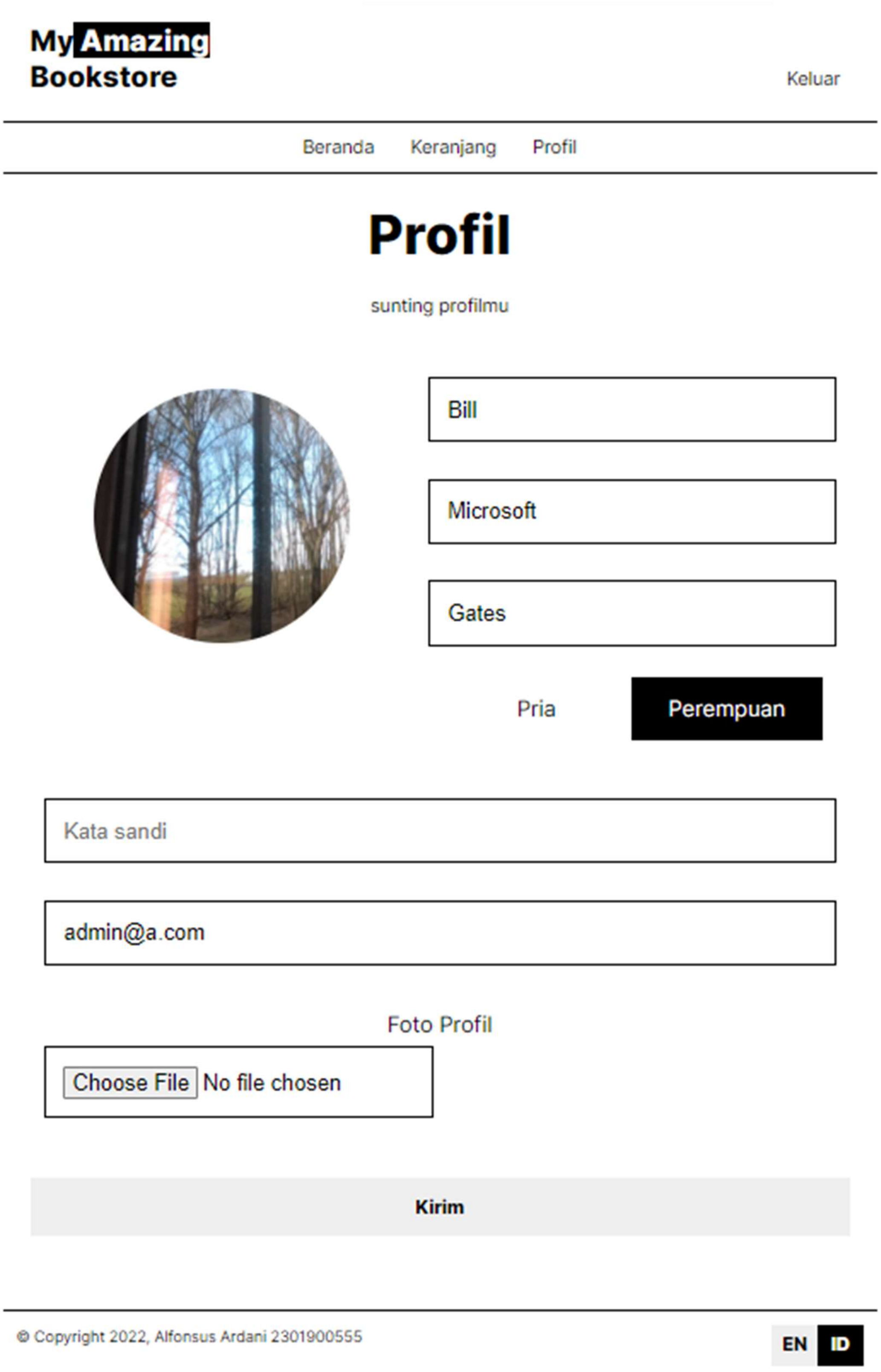
Keranjang Anda telah berhasil diperiksa dan disimpan!

> Klik Di Sini untuk kembali ke rumah

© Copyright 2022, Alfonsus Ardani 2301900555

EN ID

- **Profile Page**


My Amazing Bookstore

Keluar

Beranda Keranjang Profil

Profil

sunting profilmu

Bill

Microsoft

Gates

Pria [✓ Perempuan]

Kata sandi

admin@a.com

Foto Profil

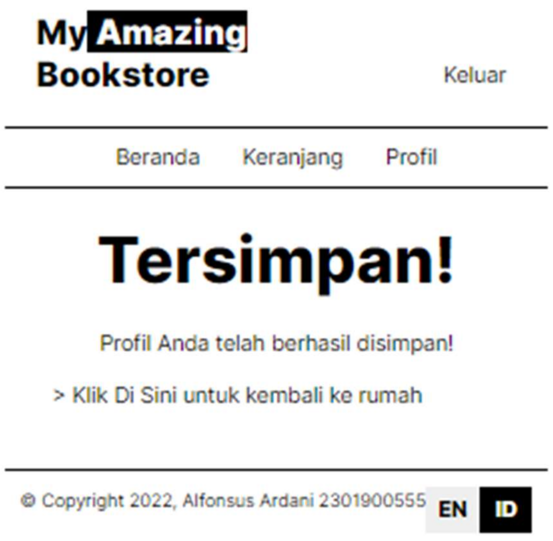
Choose File No file chosen

Kirim

© Copyright 2022, Alfonsus Ardani 2301900555

EN [✓ ID]

- **Profile Edit Confirmation**



My Amazing Bookstore

Keluar

Beranda Keranjang Profil

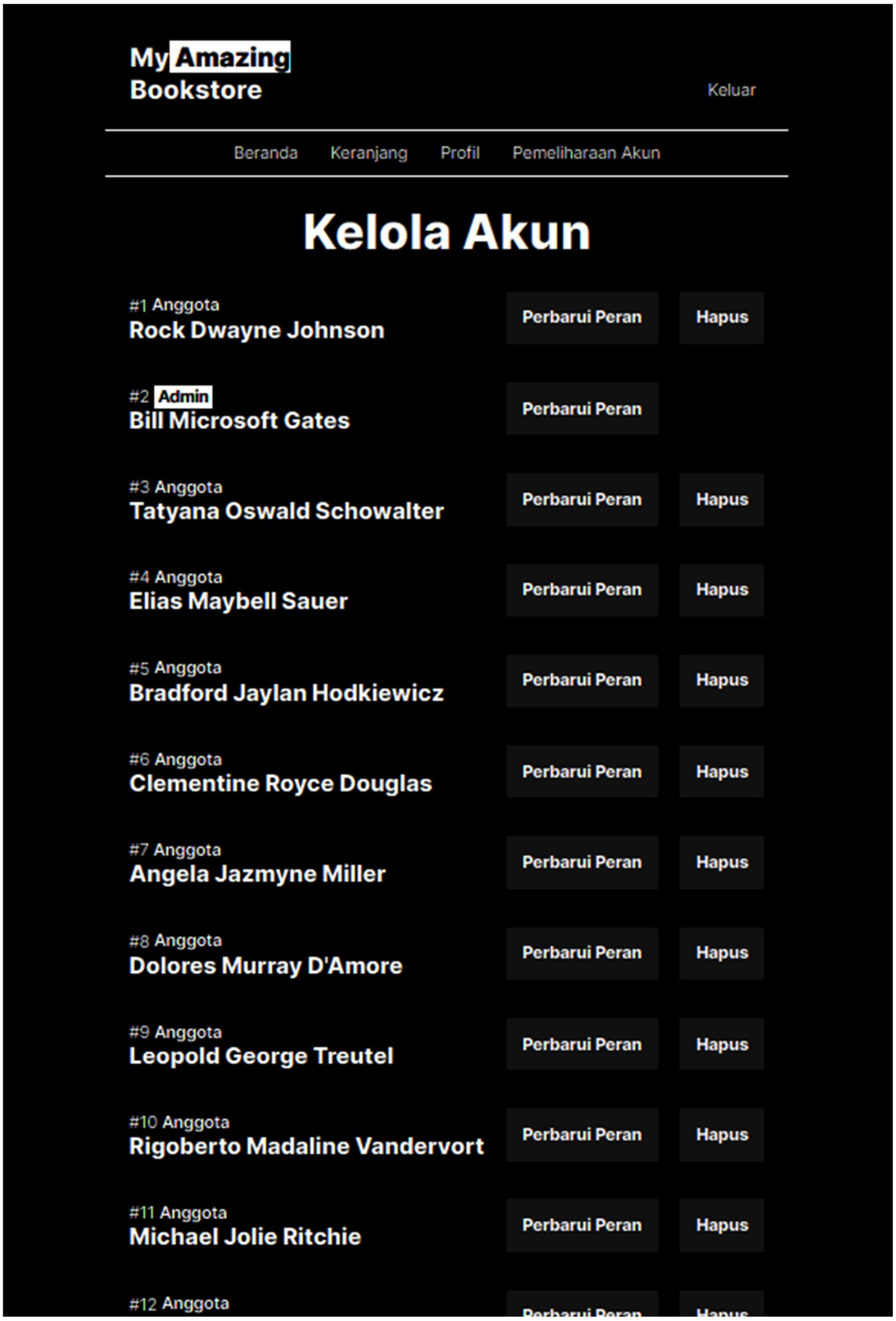
Tersimpan!

Profil Anda telah berhasil disimpan!

> Klik Di Sini untuk kembali ke rumah

© Copyright 2022, Alfonsus Ardani 2301900555

- **Account Maintenance Page**



My Amazing Bookstore

Keluar

Beranda Keranjang Profil Pemeliharaan Akun

Kelola Akun

#1 Anggota
Rock Dwayne Johnson
Perbarui Peran | Hapus

#2 Admin
Bill Microsoft Gates
Perbarui Peran

#3 Anggota
Tatyana Oswald Schowalter
Perbarui Peran | Hapus

#4 Anggota
Elias Maybell Sauer
Perbarui Peran | Hapus

#5 Anggota
Bradford Jaylan Hodkiewicz
Perbarui Peran | Hapus

#6 Anggota
Clementine Royce Douglas
Perbarui Peran | Hapus

#7 Anggota
Angela Jazmyne Miller
Perbarui Peran | Hapus

#8 Anggota
Dolores Murray D'Amore
Perbarui Peran | Hapus

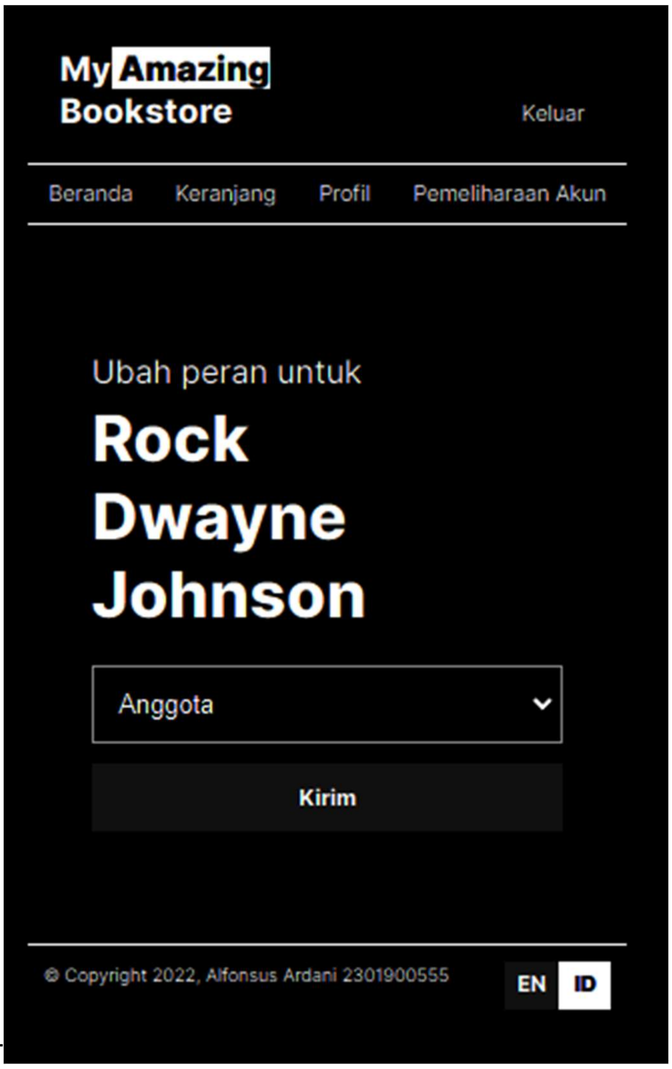
#9 Anggota
Leopold George Treutel
Perbarui Peran | Hapus

#10 Anggota
Rigoberto Madaline Vandervort
Perbarui Peran | Hapus

#11 Anggota
Michael Jolie Ritchie
Perbarui Peran | Hapus

#12 Anggota

- **Update Role Page**



My Amazing Bookstore

Keluar

Beranda Keranjang Profil Pemeliharaan Akun

Ubah peran untuk

Rock Dwayne Johnson

Anggota

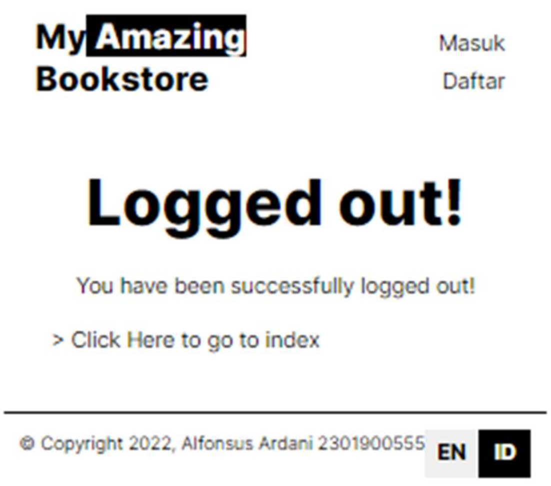
Kirim

© Copyright 2022, Alfonsus Ardani 2301900555

EN ID

T

- **Log out Confirmation**



My Amazing Bookstore

Masuk

Daftar

Logged out!

You have been successfully logged out!

> Click Here to go to index

© Copyright 2022, Alfonsus Ardani 2301900555

EN ID

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

Binusian Email: 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