

# Web and Security Technologies

## Chapter 2: Authentication and Access Control





# Agenda

- ☐ Web Authentication
- ☐ Web Security Basics
- ☐ Web Access Control



# Web Authentication



# Modify Menu Bar: Add Web Authentication Links

resources\view\layouts\menu.blade.php

```
<nav class="navbar navbar-expand-sm bg-light">
  <div class="container-fluid">
    <ul class="navbar-nav">
      ...
    </ul>
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="{{route('login')}}">Login</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="{{route('register')}}">Register</a>
      </li>
    </ul>
  </div>
</nav>
```



# Add Web Authentication Routes

routes/web.php

```
use App\Http\Controllers\Web\UsersController;
```

```
Route::get('register', [UsersController::class, 'register'])->name('register');
```

```
Route::post('register', [UsersController::class, 'doRegister'])->name('do_register');
```

```
Route::get('login', [UsersController::class, 'login'])->name('login');
```

```
Route::post('login', [UsersController::class, 'doLogin'])->name('do_login');
```

```
Route::get('logout', [UsersController::class, 'doLogout'])->name('do_logout');
```



# Add Web Authentication Controller

app\Http\Controllers\Web\UsersController.php

```
<?php
namespace App\Http\Controllers\Web;
use Illuminate\Http\Request;
use DB;
use App\Http\Controllers\Controller;
use App\Models\User;

class UsersController extends Controller {
    public function register(Request $request) {
        return view('users.register');
    }
    public function doRegister(Request $request) {
        return redirect('/');
    }
    public function login(Request $request) {
        return view('users.login');
    }
    public function doLogin(Request $request) {
        return redirect('/');
    }
    public function doLogout(Request $request) {
        return redirect('/');
    }
}
```



```
...
<form action="{{route('do_register')}}" method="post">
    {{ csrf_field() }}
    <div class="form-group mb-2">
        <label for="code" class="form-label">Name:</label>
        <input type="text" class="form-control" placeholder="name" name="name" required>
    </div>
    <div class="form-group mb-2">
        <label for="model" class="form-label">Email:</label>
        <input type="email" class="form-control" placeholder="email" name="email" required>
    </div>
    <div class="form-group mb-2">
        <label for="model" class="form-label">Password:</label>
        <input type="password" class="form-control" placeholder="password" name="password" required>
    </div>
    <div class="form-group mb-2">
        <label for="model" class="form-label">Password Confirmation:</label>
        <input type="password" class="form-control" placeholder="Confirmation" name="password_confirmation" required>
    </div>
    <div class="form-group mb-2">
        <button type="submit" class="btn btn-primary">Register</button>
    </div>
</form>
```

## Register Page

resources\view\users\register.blade.php



# Register Action

app\Http\Controllers\Web\UsersController.php

```
public function doRegister(Request $request) {  
  
    $user = new User();  
    $user->name = $request->name;  
    $user->email = $request->email;  
    $user->password = $request->password; //Not Secure  
    $user->save();  
  
    return redirect("/");  
}
```





# Register Action

app\Http\Controllers\Web\UsersController.php

```
public function doRegister(Request $request) {  
  
    $user = new User();  
    $user->name = $request->name;  
    $user->email = $request->email;  
    $user->password = bcrypt($request->password); //Secure  
    $user->save();  
  
    return redirect("/");  
}
```



# Error Handling

app\Http\Controllers\Web\UsersController.php

```
public function doRegister(Request $request) {  
  
    if($request->password!=$request->password_confirmation)  
        return redirect()->route('register', ['error'=>'Confirm password not matched.']);  
  
    if(!$request->email || !$request->name || !$request->password)  
        return redirect()->route('register', ['error'=>'Missing registration info.']);  
  
    if(User::where('email', $request->email)->first()) //Not Secure  
        return redirect()->route('register', ['error'=>'User name already exist.']);  
  
    ...  
  
    return redirect("/");  
}
```



# Error Handling

app\Http\Controllers\Web\UsersController.php

```
public function doRegister(Request $request) {  
  
    if($request->password!=$request->confirm_password)  
        return redirect()->route('register', ['error'=>'Confirm password not matched.']);  
  
    if(!$request->email || !$request->name || !$request->password)  
        return redirect()->route('register', ['error'=>'Missing registration info.']);  
  
    if(User::where('email', $request->email)->first()) //Secure  
        return redirect()->route('register', ['error'=>'Missing registration info.']);  
  
    ...  
  
    return redirect("/");  
}
```



# Displaying Errors in Register Page

resources\view\users\register.blade.php

```
<form action="{{route('do_register')}}" method="post">
    {{ csrf_field() }}
    <div class="form-group">
        @if(request()->error)
            <div class="alert alert-danger">
                <strong>Error!</strong> {{request()->error}}
            </div>
        @endif
        ...
    </form>
```



# Error Handling with Laravel Validation

app\Http\Controllers\Web\UsersController.php

```
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Validation\Rules\Password;
..
class UsersController extends Controller {
    use ValidatesRequests;
    ...
    public function doRegister(Request $request) {

        $this->validate($request, [
            'name' => ['required', 'string', 'min:5'],
            'email' => ['required', 'email', 'unique:users'],
            'password' => ['required', 'confirmed',
                Password::min(8)->numbers()->letters()->mixedCase()->symbols()],
        ]);
        ...
        return redirect("/");
    }
}
```



# Displaying Validation Errors in Register Page

resources\view\users\register.blade.php

```
<form action="{{route('do_register')}}" method="post">
    {{ csrf_field() }}
    <div class="form-group">
        @foreach($errors->all() as $error)
            <div class="alert alert-danger">
                <strong>Error!</strong> {{$error}}
            </div>
        @endforeach
        ...
    </form>
```



```
...
<form action="{{route('do_login')}}" method="post">
    {{ csrf_field() }}
    <div class="form-group">
        @foreach($errors->all() as $error)
            <div class="alert alert-danger">
                <strong>Error!</strong> {{$error}}
            </div>
        @endforeach
    </div>
    <div class="form-group mb-2">
        <label for="model" class="form-label">Email:</label>
        <input type="email" class="form-control" placeholder="email" name="email" required>
    </div>
    <div class="form-group mb-2">
        <label for="model" class="form-label">Password:</label>
        <input type="password" class="form-control" placeholder="password" name="password" required>
    </div>
    <div class="form-group mb-2">
        <button type="submit" class="btn btn-primary">Login</button>
    </div>
</form>
```

# Login Page

resources\view\users\login.blade.php



# Login Action

app\Http\Controllers\Web\UsersController.php

```
use Illuminate\Support\Facades\Auth;
...
class UsersController extends Controller {
    ...
    public function doLogin(Request $request) {

        if(!Auth::attempt(['email' => $request->email, 'password' => $request->password]))
            return redirect()->back()->withInput($request->input())->withErrors('Invalid login information.');
```

\$user = User::where('email', \$request->email)->first();  
Auth::setUser(\$user);

```
        return redirect("/");
    }
    ...
}
```





# Modify Menu Bar: Add Logout Link

resources\view\layouts\menu.blade.php

```
...
<ul class="navbar-nav">
    @auth
        <li class="nav-item"><a class="nav-link">{{auth()->user()->name}}</a></li>
        <li class="nav-item"><a class="nav-link" href="{{route('do_logout')}}">Logout</a></li>
    @else
        <li class="nav-item"><a class="nav-link" href="{{route('login')}}">Login</a></li>
    @endauth
    <li class="nav-item"><a class="nav-link" href="{{route('register')}}">Register</a></li>
</ul>
...
```



# Make Logout Action

app\Http\Controllers\Web\UsersController.php

```
public function doLogout(Request $request) {  
  
    Auth::logout();  
  
    return redirect("/");  
}
```



# Protected Pages

- Problem
  - Some of pages are public, ex: Home Page
  - Some of pages are private, ex: Products Add/Edit
  - Some of pages are semi-private, ex: Products List
    - Public Part: Product listing and filter
    - Private Part: Add/Edit/Delete buttons
- Solution
  - Modify base controller to support auth check
  - Force page security check
  - Hide private parts in blade files



## EX: Manually Protecting Edit Page

app\Http\Controllers\Web\ProductsController.php

```
class ProductsController extends Controller {  
    ...  
    public function edit(Request $request, Product $product = null) {  
  
        if(!auth()->check()) return redirect()->route('login');  
  
        $product = $product??new Product();  
  
        return view('products.edit', compact('product'));  
    }  
    ...  
}
```



# Modify Base Controller

app\Http\Controllers\Web\Controller.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
abstract class Controller extends \Illuminate\Routing\Controller  
{  
}
```



# Protect ProductsController

app\Http\Controllers\Web\ProductsController.php

```
class ProductsController extends Controller {  
  
    public function __construct()  
    {  
        $this->middleware('auth:web')->except('list');  
    }  
    ...  
}
```



# Hide Add/Edit/Delete buttons if no user authenticated

resources\products\list.blade.php

```
...  
<div class="row mt-2">  
  <div class="col col-10">  
    <h1>Products</h1>  
  </div>  
  <div class="col col-2">  
    @auth  
      <a href="{{route('products_edit')}}" class="btn btn-success form-control">Add Product</a>  
    @endauth  
  </div>  
</div>  
...
```



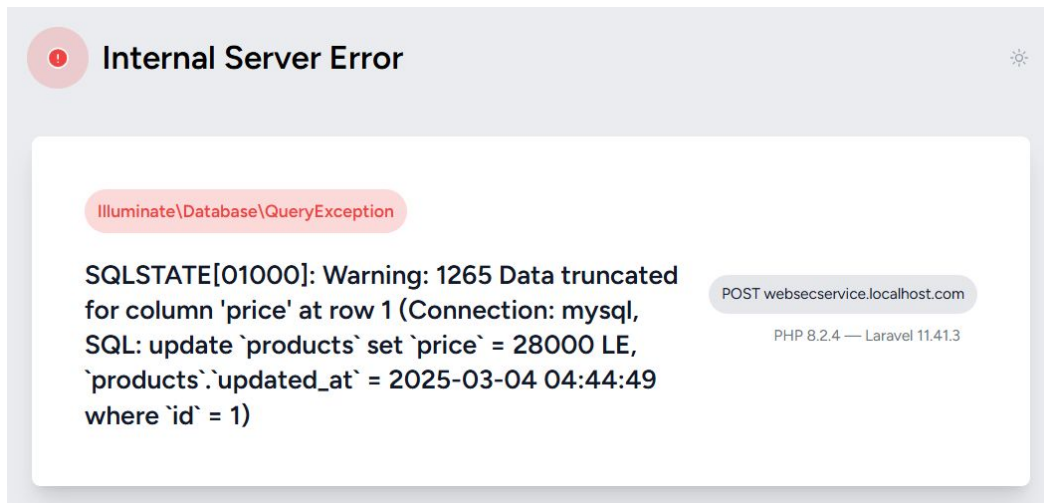
# Web Security Basics





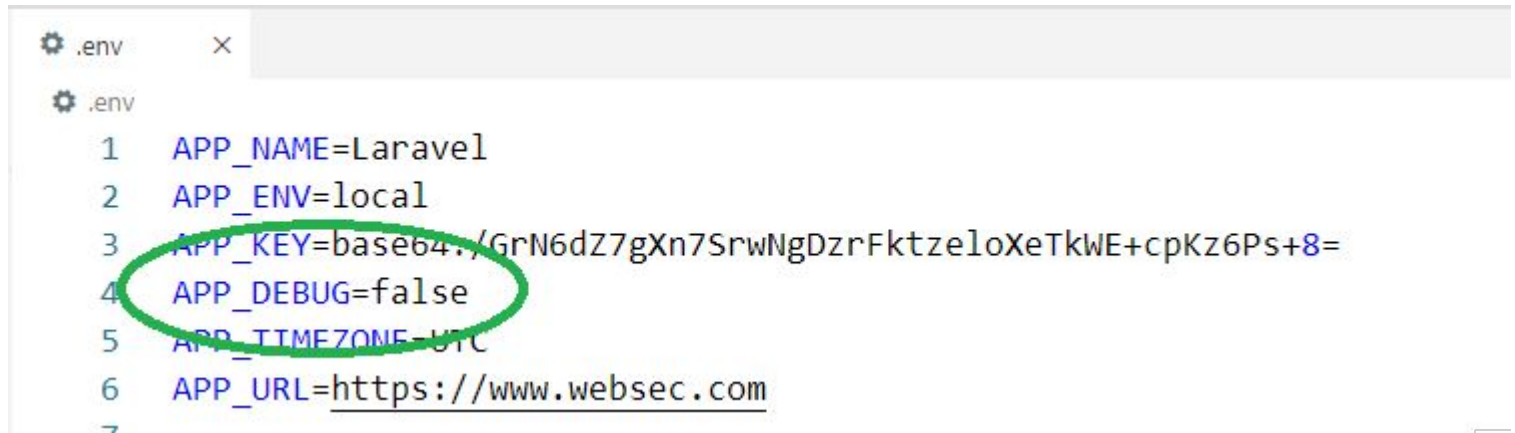
# 1. Turn off Debug in Production

- Try add product with price = “10000 LE”
- It gives error page with many information



# 1. Resolution

- Turn off Debug mode in production mode



A screenshot of a code editor showing a file named `.env`. The file contains several configuration lines. The line `APP_DEBUG=false` is highlighted with a green oval. The other lines are `APP_NAME=Laravel`, `APP_ENV=local`, `APP_KEY=base64:./GrN6dZ7gXn7SrwNgDzrFktzeloXeTkWE+cpKz6Ps+8=`, `APP_TIMEZONE=UTC`, and `APP_URL=https://www.websec.com`.

```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:./GrN6dZ7gXn7SrwNgDzrFktzeloXeTkWE+cpKz6Ps+8=
4 APP_DEBUG=false
5 APP_TIMEZONE=UTC
6 APP_URL=https://www.websec.com
7
```

## 2. Avoid Raw SQL Execution

//Secure

```
public function delete(Request $request, Product $product) {  
  
    $product->delete();  
  
    return redirect()->route('products_list');  
}
```

//Insecure

```
public function delete(Request $request, $product) {  
  
    DB::unprepared("DELETE FROM products WHERE id = $product;");  
  
    return redirect()->route('products_list');  
}
```

<http://websecservice.localhost.com/products/delete/99>

<http://websecservice.localhost.com/products/delete/99;DELETE%20FROM%20products;>



### 3. Validate Form Data

- Supplying wrong information may lead to security threat
- Ex: providing huge amount of text may lead to server malfunction



## 3. Modify Products Save with Laravel Validation

```
public function save(Request $request, Product $product = null) {  
  
    $this->validate($request, [  
        'code' => ['required', 'string', 'max:32'],  
        'name' => ['required', 'string', 'max:128'],  
        'model' => ['required', 'string', 'max:256'],  
        'description' => ['required', 'string', 'max:1024'],  
        'price' => ['required', 'numeric'],  
    ]);  
  
    $product = $product ?? new Product();  
    $product->fill($request->all());  
    $product->save();  
  
    return redirect()->route('products_list');  
}
```



### 3. Modify Products Edit to display Laravel Errors

```
<form action="{{route('products_save', $product->id)}}" method="post">
    {{ csrf_field() }}
    @foreach($errors->all() as $error)
        <div class="alert alert-danger">
            <strong>Error!</strong> {{$error}}
        </div>
    @endforeach
    ...
</form>
```



# More

- In change password, user must supply old password
- User email must be verified:
  - Send verification link to email
  - Activate the user if the user click on the link
  - Alternatively, send verification code and take it from user after login
- User mobile must be verified:
  - Send verification code to mobile via [SMS, Whatsapp, ...]
  - Take it from user after login



# More

- User alternative access methods rather than password:
  - Code sent to Email, Mobile SMS, Whatsapp, ...
  - OTP Code provided by Hardware or Software
  - Certificate installed on Smart Token or Smart Card
  - External global authentication provider like Google, Facebook, Microsoft
  - External proprietary authentication service provider





# Exercises 4



## Lab Exercises 4/1

- Make profile page if user click on “User Name”
- Show user information
- Allow user to change the password as if he/she supplied old one



## Lab Exercises 4/2

- In users CRUD in Exercise 3
- Set on of user as admin by adding “admin” field in users table
- Admin user can view all users profile page
- Admin user can edit/edit/change password of all users



## Lab Exercises 4/3

- Add security question to user information while registration
- Handle forget password by adding forget password link in login page
- The user should answer the security question correctly
- If so, the user is forwarded to change password page to make a new one
- Is this solution is secure?



## Lab Exercises 4/4

- Handle forget password by adding forget password link in login page
- User should provide email, the system should send email with temp password that can be used once



## Lab Exercises 4/5

- Add Mobile number to user information
- Handle forget password by adding forget password link in login page
- User should can optionally provide email or mobile number
- If the user provide the email, the system should send email with temp password that can be used once
- If the user provide the mobile, the system should send sms or whatsapp message with temp password that can be used once



# Web Access Control



# Installing Access Control Library “Spatie”

Command Line

```
composer require spatie/laravel-permission
```

```
php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"
```

```
php artisan config:clear
```

```
php artisan migrate
```

app\Models\User.php

```
...  
use Spatie\Permission\Traits\HasRoles;  
class User extends Authenticatable  
{  
    use HasRoles;  
    ...  
}
```



# Spatie Database Models

websec model_has_permissions
permission_id : bigint(20) unsigned
model_type : varchar(255)
model_id : bigint(20) unsigned

websec permissions
id : bigint(20) unsigned
name : varchar(255)
guard_name : varchar(255)
created_at : timestamp
updated_at : timestamp

websec role_has_permissions
permission_id : bigint(20) unsigned
role_id : bigint(20) unsigned

websec model_has_roles
role_id : bigint(20) unsigned
model_type : varchar(255)
model_id : bigint(20) unsigned

websec roles
id : bigint(20) unsigned
name : varchar(255)
guard_name : varchar(255)
created_at : timestamp
updated_at : timestamp

Spatie allow creating jobs “**Roles**” with Responsibilities “**Permissions**”.

Users “**Models**” can have multiple “**Roles**”. Also they can have additional “**Permissions**” directly.



**Table:** roles

id	name	guard_name	created_at	updated_at
1	Admin	web	NULL	NULL

**Table:** permissions

id	name	guard_name	created_at	updated_at
1	add_products	web	NULL	NULL
2	edit_products	web	NULL	NULL
3	delete_products	web	NULL	NULL

**Table:** model\_has\_roles

role_id	model_type	model_id
1	App\Models\User	1

## Adding Roles and Permissions Manually

**Table:**  
roles\_has\_permissions

permission_id	role_id
1	1
2	1
3	1

# Activating Roles and Permissions Cash

```
php artisan cache:clear
```

OR

```
use Artisan;  
...  
Artisan::call('cache:clear');
```

Spatie library load roles/permissions into cash for faster access to avoid database queries every checking time inside code. Single page or service may have 10th of permissions inside backend functions or around every piece of UI.



# Protecting Frontend UI Elements

resources\view\products\list.blade.php

```
<div class="col col-2">
    @can('edit_products')
        <a href="{{route('products_edit')}}" class="btn btn-success form-control">Add Product</a>
    @endcan
</div>
```

```
<div class="col col-2">
    @can('edit_products')
        <a href="{{route('products_edit', $product->id)}}" class="btn btn-success form-control">Edit</a>
    @endcan
</div>
<div class="col col-2">
    @can('delete_products')
        <a href="{{route('products_delete', $product->id)}}" class="btn btn-danger form-control">Delete</a>
    @endcan
</div>
```



# Protecting Backend Services

app\Http\Controllers\Web\ProductsController.php

```
public function delete(Request $request, Product $product) {  
  
    if (!auth()->user()->hasPermissionTo('delete_products')) {  
        abort(401);  
    }  
  
    $product->delete();  
  
    return redirect()->route('products_list');  
}
```

401	Unauthorized
404	Page Not Found
402	Payment Required
500	Server Error

# Make User Profile Page

```
Route::get('profile/{user?}', [UsersController::class, 'profile'])->name('profile');
```

rote\web.php

```
public function profile(Request $request, User $user = null) {  
    $user = $user??auth()->user();  
    return view('users.profile', compact('user'));  
}
```

app\Http\Controllers\Web\UsersController.php



```

@extends('layouts.master')
@section('title', 'User Profile')
@section('content')
<div class="d-flex justify-content-center">
    <div class="m-4 col-sm-6">
        <table class="table table-striped">
            <tr>
                <th>Name</th><td>{{ $user->name }}</td>
            </tr>
            <tr>
                <th>Email</th><td>{{ $user->email }}</td>
            </tr>
            <tr>
                <th>Roles</th>
                <td>
                    @foreach($user->roles as $role)
                        <span class="badge bg-primary">{{ $role->name }}</span>
                    @endforeach
                </td>
            </tr>
        </table>
    </div>
</div>
@endsection

```

# Make User Profile Page

resources\view\users\profile.blade.php

<b>Name</b>	Mohamed Saleh
<b>Email</b>	mohamed.saleh@sut.edu.eg
<b>Roles</b>	<span>Admin</span>

# Securing Profile Page Access

id	name	display_name	guard_name	created
1	add_products	Add Products	web	NULL
2	edit_products	Edit Products	web	NULL
3	delete_products	Delete Products	web	NULL
4	show_users	Show Users	web	NULL
5	edit_users	Edit Users	web	NULL

**Table:**  
permissions

permission_id	model_type	model_id
4	App\Models\User	1
5	App\Models\User	1

**Table:** model\_has\_permissions

```
php artisan cache:clear
```

**Command Line**

```
public function profile(Request $request, User $user = null) {  
    $user = $user??auth()->user();  
    if(auth()->id()!=$user->id) {  
        if(!auth()->user()->hasPermissionTo('show_users')) abort(401);  
        return view('users.profile', compact('user'));  
    }  
}
```



# Getting all Permissions

app\Http\Controllers\Web\UsersController.php

```
public function profile(Request $request, User $user = null) {  
    ...  
  
    $permissions = [];  
    foreach($user->permissions as $permission) {  
        $permissions[] = $permission;  
    }  
    foreach($user->roles as $role) {  
        foreach($role->permissions as $permission) {  
            $permissions[] = $permission;  
        }  
    }  
  
    return view('users.profile', compact('user', 'permissions'));  
}
```



# Displaying all Permissions

resources\view\users\profile.blade.php

```
...
<tr>
    <th>Direct Permissions</th>
    <td>
        @foreach($permissions as $permission)
            <span class="badge bg-success">{{$permission->display_name}}</span>
        @endforeach
    </td>
</tr>
...
```

Name	Mohamed Saleh Ahmed
Email	mohamed.saleh@sut.edu.eg
Roles	<a href="#">Admin</a>
Direct Permissions	<a href="#">Show Users</a> <a href="#">Edit Users</a> <a href="#">Add Products</a> <a href="#">Edit Products</a> <a href="#">Delete Products</a>

# Make Edit User Page

```
Route::get('users/edit/{user?}', [UserController::class, 'edit'])->name('users_edit');
```

rote\web.php

```
public function edit(Request $request, User $user = null) {  
  
    $user = $user??auth()->user();  
    if(auth()->id()!=$user->id) {  
        if(!auth()->user()->hasPermissionTo('edit_users')) abort(401);  
    }  
  
    return view('users.edit', compact('user'));  
}
```

app\Http\Controllers\Web\UserController.php

# Make Edit User Page

resources\view\users\profile.blade.php

```
...
<div class="row">
    <div class="col col-10">
    </div>
    <div class="col col-2">
        @if(auth()->user()->hasPermissionTo('edit_users') || auth()->id()==$user->id)
            <a href="{{route('users_edit')}}" class="btn btn-success form-control">Edit</a>
        @endif
    </div>
</div>
...
```



# Make Edit User Page

resources\view\users\edit.blade.php

```
<div class="d-flex justify-content-center">
  <div class="row m-4 col-sm-8">
    <form action="{{route('users_save', $user->id)}}" method="post">
      {{ csrf_field() }}
      @foreach($errors->all() as $error)
        <div class="alert alert-danger">
          <strong>Error!</strong> {{$error}}
        </div>
      @endforeach
      <div class="row mb-2">
        <div class="col-12">
          <label for="code" class="form-label">Name:</label>
          <input type="text" class="form-control" placeholder="Name" name="name" required value="{{ $user->name }}">
        </div>
      </div>
      <button type="submit" class="btn btn-primary">Submit</button>
    </form>
  </div>
</div>
```



# Make Edit User Page

```
Route::post('users/save/{user}', [UsersController::class, 'save'])->name('users_save');
```

rote\web.php

```
public function save(Request $request, User $user) {  
  
    if(auth()->id()!=$user->id) {  
        if(!auth()->user()->hasPermissionTo('show_users')) abort(401);  
    }  
  
    $user->name = $request->name;  
    $user->save();  
  
    return redirect(route('profile', ['user'=>$user->id]));  
}
```

app\Http\Controllers\Web\UsersController.php

# Edit User Roles and Permissions

app\Http\Controllers\Web\UsersController.php

```
public function edit(Request $request, User $user = null) {  
    ...  
    $roles = [];  
    foreach(Role::all() as $role) {  
        $role->taken = ($user->hasRole($role->name));  
        $roles[] = $role;  
    }  
  
    $permissions = [];  
    $directPermissionsIds = $user->permissions()->pluck('id')->toArray();  
    foreach(Permission::all() as $permission) {  
        $permission->taken = in_array($permission->id, $directPermissionsIds);  
        $permissions[] = $permission;  
    }  
  
    return view('users.edit', compact('user', 'roles', 'permissions'));  
}
```



# Edit User Roles and Permissions

resources\view\users\edit.blade.php

```
...
@can('edit_users')
<div class="col-12 mb-2">
    <label for="model" class="form-label">Roles:</label>
    <select multiple class="form-select" name="roles[]">
        @foreach($roles as $role)
            <option value="{{ $role->name }}" {{ $role->taken?'selected':'' }}>
                {{ $role->name }}
            </option>
        @endforeach
    </select>
</div>
<div class="col-12 mb-2">
    <label for="model" class="form-label">Direct Permissions:</label>
    <select multiple class="form-select" name="permissions[]">
        @foreach($permissions as $permission)
            <option value="{{ $permission->name }}" {{ $permission->taken?'selected':'' }}>
                {{ $permission->display_name }}
            </option>
        @endforeach
    </select>
</div>
@endcan
...
```

Name:

Mohamed Saleh Ahmed

Roles:

Admin

Direct Permissions:

Add Products  
Edit Products  
Delete Products  
Show Users

Submit



# Edit User Roles and Permissions

app\Http\Controllers\Web\UsersController.php

```
use Artisan;

...
public function save(Request $request, User $user) {
    ...
    if(auth()->user()->hasPermissionTo('edit_users')) {

        $user->syncRoles($request->roles);
        $user->syncPermissions($request->permissions);

        Artisan::call('cache:clear');
    }

    return redirect(route('profile', ['user'=>$user]));
}
```



# Exercises 5



## Lab Exercises 5/1

- Modify users CRUD to support access control
- Allow admin with edit\_users to view, edit, change password, delete any users.
- Allow normal users to view his/her profile and edit general information only.
- Make another role “Employee” that able to edit general user information only.



## Lab Exercises 5/2

- Make CRUD for roles
- Allow admin only to manage roles from web without a need to access database directly.
- Admin can view/add/edit/remove roles.
- Admin can change permissions per roles.



## Lab Exercises 5/3

- Make CRUD for permissions
- Do you think this feature is valid?



# Secure Email Verification



# Generation Mail Wrapper

php artisan make:mail VerificationEmail

app\Models\User.php

```
...  
MAIL_DRIVER=smtp  
MAIL_HOST=outlook.office.com  
MAIL_PORT=587  
MAIL_USERNAME=????@sut.edu.eg  
MAIL_PASSWORD=????  
MAIL_ENCRYPTION=tls  
MAIL_FROM_ADDRESS=????@sut.edu.eg  
MAIL_FROM_NAME=SUT
```

Page - 63

.env

```
...  
class VerificationEmail extends Mailable  
{  
    ...  
}
```

app\Mail\VerificationEmail.php

# Modify Mail Wrapper

app\Mail\VerificationEmail.php

```
class VerificationEmail extends Mailable
{
    private $link = null;
    private $name = null;
    ...
    public function __construct($link, $name) {
        $this->link = $link; $this->name = $name;
    }
    ...
    public function content(): Content
    {
        return new Content(
            view: 'emails.verification',
            with: [ 'link' => $this->link, 'name' => $this->name ],
        );
    }
}
```





# Make Email Template

resources\view\email\activation.blade.php

```
<!DOCTYPE html>
<html lang="en">
<body>
    <p>Dear {{$name}},</p>
    <p>Click on the following link to verify your account:</p>
    <p><a href="{{$link}}" target='_blank'>Verification Link</a></p>
</body>
</html>
```



# Modify Registration Sequence

```
Route::get('verify', [UsersController::class, 'verify'])->name('verify');
```

rote\web.php

```
use Illuminate\Support\Facades\Crypt;
use Illuminate\Support\Facades\Mail;
use App\Mail\VerificationEmail;
...
public function doRegister(Request $request) {
    ...
    $title = "Verification Link";
    $token = Crypt::encryptString(json_encode(['id' => $user->id, 'email' => $user->email]));
    $link = route("verify", ['token' => $token]);
    Mail::to($user->email)->send(new VerificationEmail($link, $user->name));
    return redirect('/');
}
```

# Verify the Verification Link

app\Http\Controllers\Web\UsersController.php

```
use Carbon\Carbon;
...
public function verify(Request $request) {

    $decryptedData = json_decode(Crypt::decryptString($request->token), true);
    $user = User::find($decryptedData['id']);
    if(!$user) abort(401);
    $user->email_verified_at = Carbon::now();
    $user->save();

    return view('users.verified', compact('user'));
}
```

