1) Login and Register

Built in auth in laravel : php artisan make:auth

Controllers> auth

Model> user.php

Views> auth

Route> web.php

2) Encrypt/Decrypt User Information

php artisan key:generate

.env >

```
APP_KEY=base64:BQoxZC7gwUhY9BnXOfOcK4zN0pmc20L577G+sBMjr+s=
```

Registeredusercontroller > encryption

app > Http > Controllers > Auth > RegisteredUserController.php > RegisteredUserController > create

```php
class RegisteredUserController extends Controller
    public function store(Request $request): RedirectResponse
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email', 'max:255', 'unique:'.User::class],
            'password' => ['required', 'confirmed', Rules\Password::defaults()],
        ]);

        // $user = User::create([
        //     'name' => $request->name,
        //     'email' => $request->email,
        //     'password' => Hash::make($request->password),
        // ]);

        $encryptedData = [
            'name' => Crypt::encryptString($request->name),
            'email' => Crypt::encryptString($request->email),
            'password' => Hash::make($request->password),
        ];

        $user = User::create($encryptedData);

        event(new Registered($user));

        Auth::login($user);

        return redirect(RouteServiceProvider::HOME);
    }
```

3. Password should be hashed and salted before storing in the database.

To hash and salt passwords in Laravel, we don't need to manually handle the hashing process. Laravel automatically hashes and salts passwords using the Bcrypt algorithm. Laravel's built-in authentication system takes care of it .

Passwordcontroller.php

```php
class PasswordController extends Controller
{
    /**
     * Update the user's password.
     */
    public function update(Request $request): RedirectResponse
    {
        $validated = $request->validateWithBag('updatePassword', [
            'current_password' => ['required', 'current_password'],
            'password' => ['required', Password::defaults(), 'confirmed'],
        ]);

        $request->user()->update([
            'password' => Hash::make($validated['password']),
        ]);

        return back()->with('status', 'password-updated');
    }
}
```

Model > user,php

```php
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Illuminate\Support\Facades\Crypt;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];

    /**
     * Decrypt the name attribute when retrieving it.
     *
     * @param  string  $value
     * @return string
     */
```

4. Separate function for credential check.

Registeredusercontroller.php

```php
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email', 'max:255', 'unique:'.User::class],
            'password' => ['required', 'confirmed', Rules\Password::defaults()],
        ]);
```

5. A key management module should be defined.

Config > app.php

```
AES-256-CBC
```

```
|--------------------------------------------
| Encryption Key
|--------------------------------------------
|
| This key is used by the Illuminate encrypter
| to a random, 32 character string, otherwise t
| will not be safe. Please do this before deplo
|
*/

'key' => env('APP_KEY'),

'cipher' => 'AES-256-CBC',

/*
|--------------------------------------------
| Maintenance Mode Driver
```

php artisan key:generate

```
APP_KEY=base64:BQoxZC7gwUhY9BnXOfOcK4zN0pmc20L577G+sBMjr+s=
```

6. Users can post/view using encryption and decryption.