CSC348 Assignment 1

Models

User.php

```php
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use App\Models\Post;
use App\Models\Comment;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

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

    /**
     * Retrieve all posts associated with this user.
     *
     * @return One-To-Many relationship between the one user and their many posts.
     */
    public function posts() {
        return $this -> hasMany(Post::class);
    }

    /**
     * Retrieve all comments commented by this user.
     *
     * @return One-To-Many relationship between the one user and their many comments on posts.
     */
    public function comments() {
        return $this -> hasMany(Comment::class);
    }

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
}
```

Post.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\User;
use App\Models\Comment;

class Post extends Model
{
    use HasFactory;

    /**
     * Retrieve the user who posted this item.
     *
     * @return One-To-One relationship between the one user who owns this one post.
```

```php
     */
    public function user() {
        return $this->belongsTo(User::class);
    }

    /**
     * Retrieve the comments on this post.
     *
     * @return One-To-Many relationship between many comments commented on this one post.
     */
    public function comments() {
        return $this->hasMany(Comment::class);
    }
}
```

Comment.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Comment extends Model
{
    use HasFactory;

    /**
     * Retrieve the post to which this comment belongs.
     *
     * @return One-To-One relationship between one comment on one post.
     */
    public function post() {
        return $this -> belongsTo(Post::class);
    }

    /**
     * Retrieve the user who commented on this comment.
     *
     * @return One-To-One relationship between one comment from one user.
     */
    public function user() {
        return $this -> belongsTo(User::class);
    }

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'user_id',
        'comment',
        'post_id'
    ];
}
```

Migrations

User Table

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('password');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->rememberToken();
            $table->boolean('isAdmin')->default(false);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
};
```

Posts Table

```
<?php

```

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use App\Models\User;
use App\Models\Comment;
use App\Models\Like;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('image_path');
            $table->foreignId('user_id')->constrained()
            ->onDelete('cascade')->onUpdate('cascade');
            $table->timestamps();
        });

    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('posts');
    }
};
```

Comments Table

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use App\Models\User;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        //Pivot table between many user's comments, on many posts.
        Schema::create('comments', function (Blueprint $table) {
            $table->id();
            $table->string('comment');
            $table->foreignId('post_id')->constrained()
            ->onDelete('cascade')->onUpdate('cascade');
            $table->foreignId('user_id')->constrained()
            ->onDelete('cascade')->onUpdate('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('comments');
    }
};
```

Seeding

UserTableSeeder.php

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\User;
use Carbon\Carbon;
use Illuminate\Support\Str;

class UserTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //Statically create two admin users.
        $user = new User;
        $user->name = "linus";
        $user->password = bcrypt("windows");
        $user->email = "linus.torvalds@linux.com";
        $user->email_verified_at = Carbon::now();
        $user->remember_token = Str::random(10);
        $user->isAdmin = true;
        $user->save();

        $user = new User;
        $user->name = "Gosling";
        $user->password = bcrypt("java");
        $user->email = "james.gosling@oracle.com";
        $user->email_verified_at = Carbon::now();
        $user->remember_token = Str::random(10);
        $user->isAdmin = true;
        $user->save();

        //Factory creates 50 random non-admin users.
        User::factory()->count(50)->create();
    }
}
```

PostsTableSeeder.php

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Post;

class PostsTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //Statically create two posts for admin users.
        $post = new Post;
        $post->image_path = asset("images/108706.JPG");
        $post->user_id = 1;
        $post->save();
        $post = new Post;
        $post->image_path = asset("images/142266.JPG");
        $post->user_id = 2;
        $post->save();

        //Create 10 random posts for admin user 1
        Post::factory()->count(10)->create(['user_id' => 1]);

        //Create 10 random posts for admin user 2
        Post::factory()->count(10)->create(['user_id' => 2]);

        //Create 50 posts for random users
        Post::factory()->count(50)->create();
    }
}
```

CommentsTableSeeder.php

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Comment;

class CommentsTableSeeder extends Seeder
{
    /**
     * Run the Comment seeding.
     */
    public function run(): void
    {
        //Two static comments for admin users.
        $comment = new Comment;
        $comment->comment = "What a cute dog!";
        $comment->user_id = 1;
        $comment->post_id = 1;
        $comment->save();

        $comment = new Comment;
        $comment->comment = "What breed of dog is that?";
        $comment->user_id = 2;
        $comment->post_id = 2;
        $comment->save();

        //Factory generates 25 random comments, between random posts and random users.
        Comment::factory()->count(25)->create();
    }
}
```

DatabaseSeeder.php

```php
<?php

namespace Database\Seeders;

use App\Models\User;
// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
```

```php
{
    /**
     * Seed the application's database.
     * Note: The order is important for referential integrity.
     */
    public function run(): void
    {
        $this->call(UserTableSeeder::class);
        $this->call(PostsTableSeeder::class);
        $this->call(CommentsTableSeeder::class);
    }
}
```

Factories

UserFactory.php

```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\User>
 */
class UserFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        return [
            'name' => fake()->name(),
            'email' => fake()->unique()->safeEmail(),
            'email_verified_at' => now(),
            'password' => Hash::make(fake()->password()),
            'remember_token' => Str::random(10),
            'isAdmin' => false,
        ];
    }
}
```

PostFactory.php

```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;
use Illuminate\Support\Str;
use App\Models\User;
use App\Models\Comment;
use App\Models\Like;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Post>
 */
class PostFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        //Check if a user is passed into the post factory.
        $defaultUserId = User::first()->id ?? null;

        //Get a random file the post will have.
        $files = glob(public_path('images/*.*'));
        $key = array_rand($files);
        $file = $files[$key];

        // Convert the system path to a web-accessible path.
        $relativePath = str_replace(public_path(), '', $file);

        $image_path = asset($relativePath);

        //Assign the post to the user passed in, or find a random user if null.
        return [
             'image_path' => $image_path,
             'user_id' => fake()->randomElement(User::pluck('id')->toArray()) ?: $defaultUserId,
             'created_at' => fake()->dateTimeBetween('-1 year', 'now'),
        ];
    }
}
```

CommentFactory.php

```php
<?php

namespace Database\Factories;
```

```php
use Illuminate\Database\Eloquent\Factories\Factory;
use App\Models\User;
use App\Models\Post;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Comment>
 */
class CommentFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        return [
            'comment' => fake()->sentence,
            'user_id' => fake()->randomElement(User::pluck('id')->toArray()),
            'post_id' => fake()->randomElement(Post::pluck('id')->toArray()),
        ];
    }
}
```