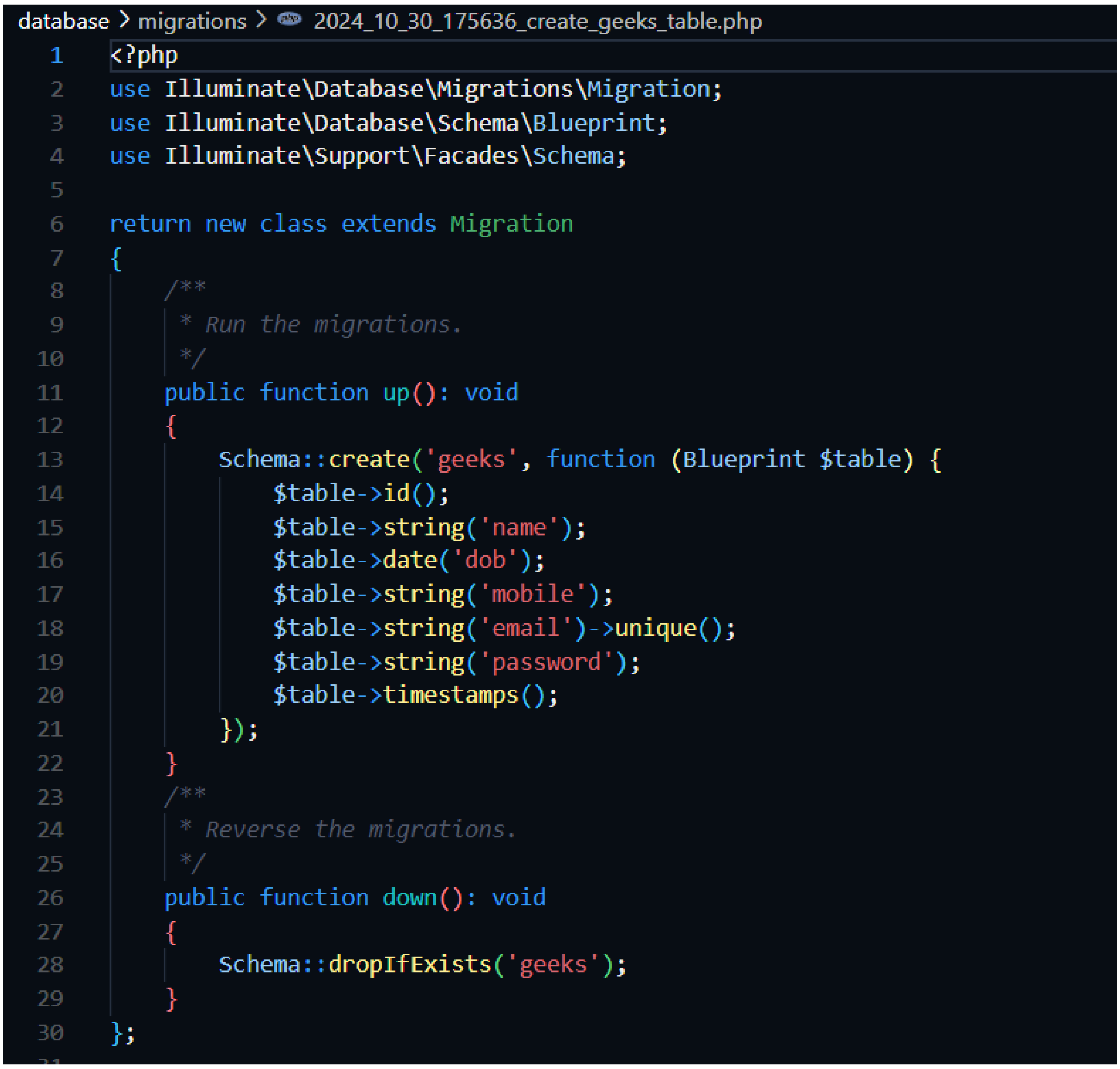
Name – Priyanshu Dubey Student No. - 2374690

Coursework 1 - Code Submission (CSCM48)

Geek Meet is a social media platform for tech enthusiasts to connect, share posts, comment, like, and build profiles. Users, or "Geeks," can interact around shared interests, creating a community space for discussions and connections.

Migration Files:

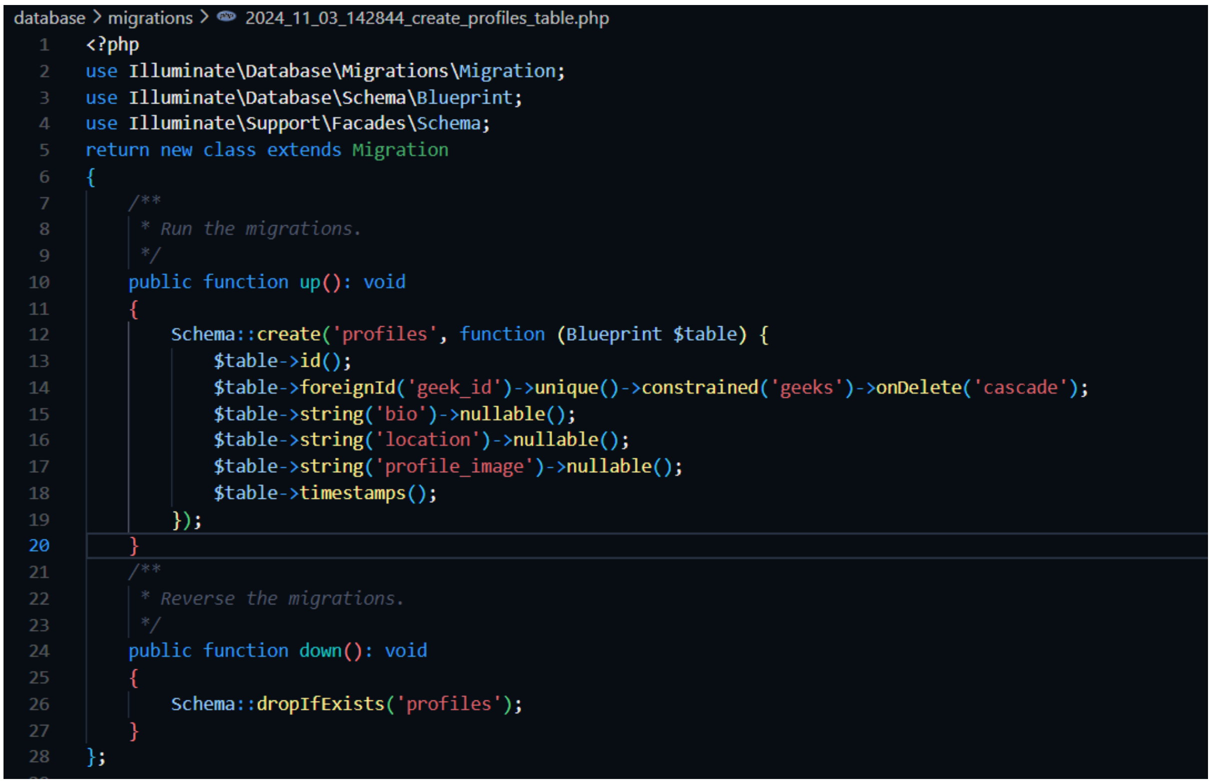
1. **create_geeks_table.php**: Migration file for the geeks table.



database > migrations > 2024_10_30_175636_create_geeks_table.php

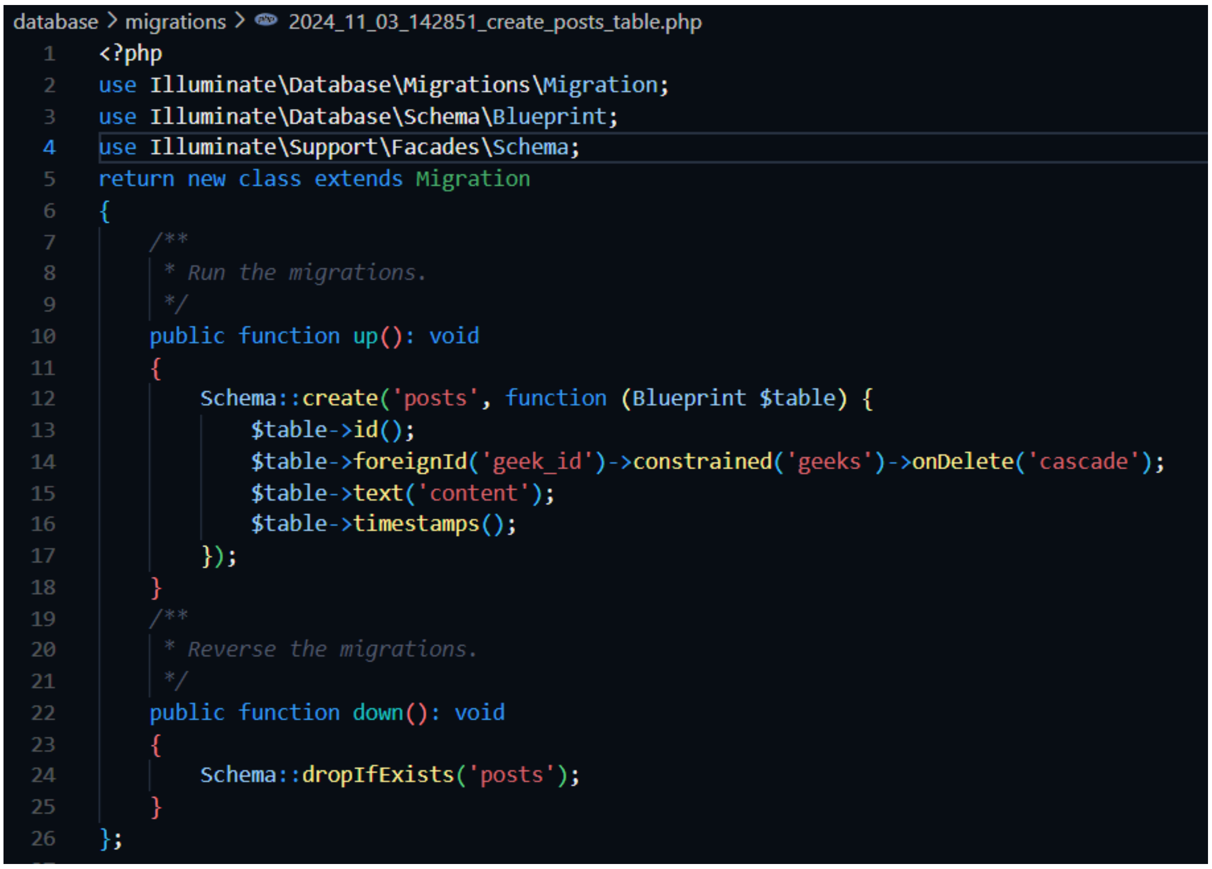
```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('geeks', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->date('dob');
            $table->string('mobile');
            $table->string('email')->unique();
            $table->string('password');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('geeks');
    }
};
```

2. **create_profiles_table.php**: Migration file for the profiles table.


database > migrations > 2024_11_03_142844_create_profiles_table.php

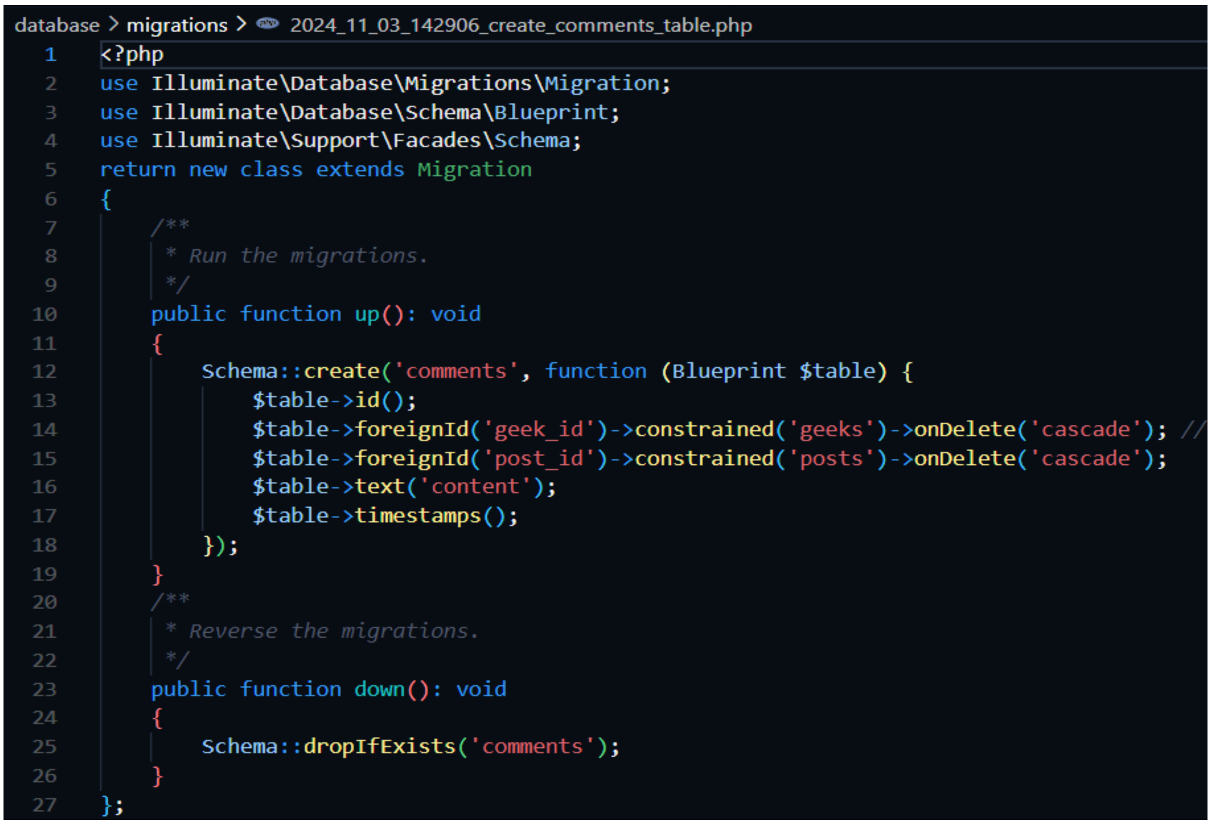
```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('profiles', function (Blueprint $table) {
            $table->id();
            $table->foreignId('geek_id')->unique()->constrained('geeks')->onDelete('cascade');
            $table->string('bio')->nullable();
            $table->string('location')->nullable();
            $table->string('profile_image')->nullable();
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('profiles');
    }
};
```

3. **create_posts_table.php**: Migration file for the posts table.


database > migrations > 2024_11_03_142851_create_posts_table.php

```php
<?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('posts', function (Blueprint $table) {
            $table->id();
            $table->foreignId('geek_id')->constrained('geeks')->onDelete('cascade');
            $table->text('content');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('posts');
    }
};
```

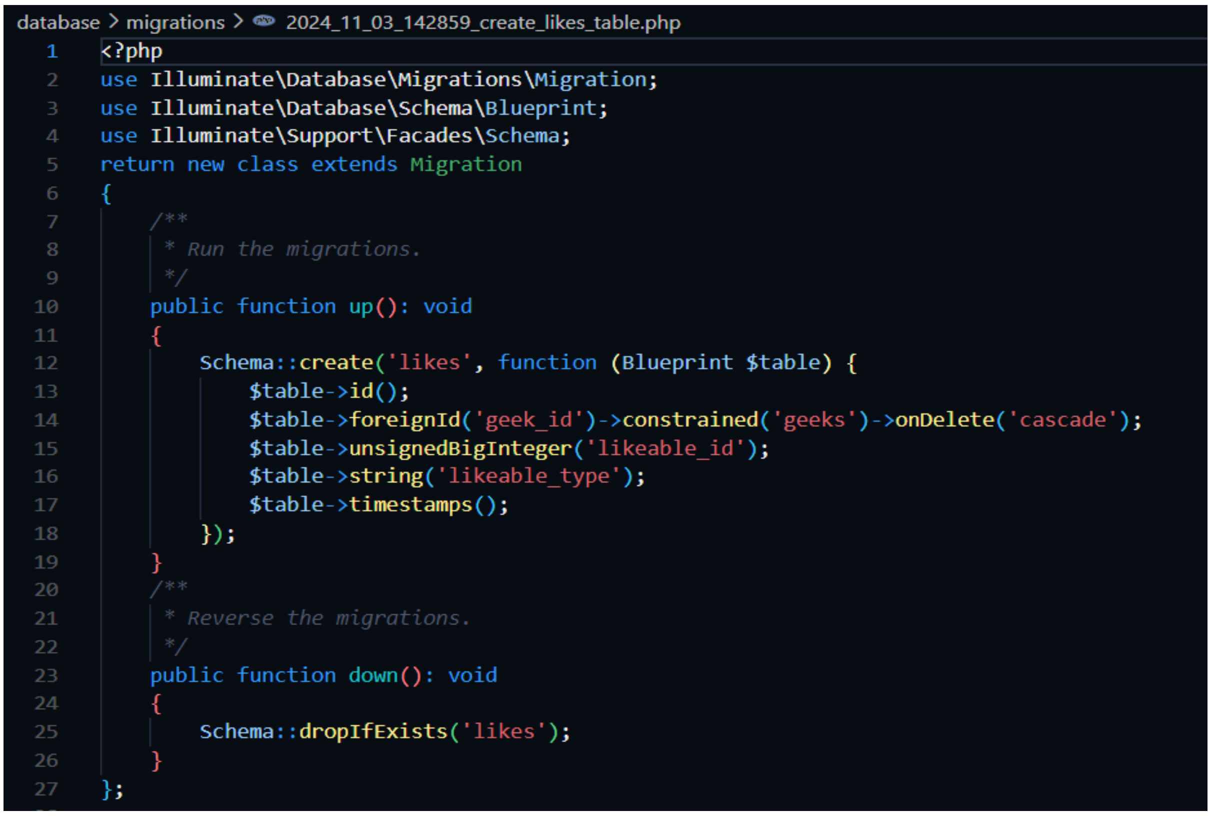
4. **create_comments_table.php**: Migration file for the comments table.



database > migrations > 2024_11_03_142906_create_comments_table.php

```php
<?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('comments', function (Blueprint $table) {
            $table->id();
            $table->foreignId('geek_id')->constrained('geeks')->onDelete('cascade'); //
            $table->foreignId('post_id')->constrained('posts')->onDelete('cascade');
            $table->text('content');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('comments');
    }
};
```

5. **create_likes_table.php**: Migration file for the likes table.

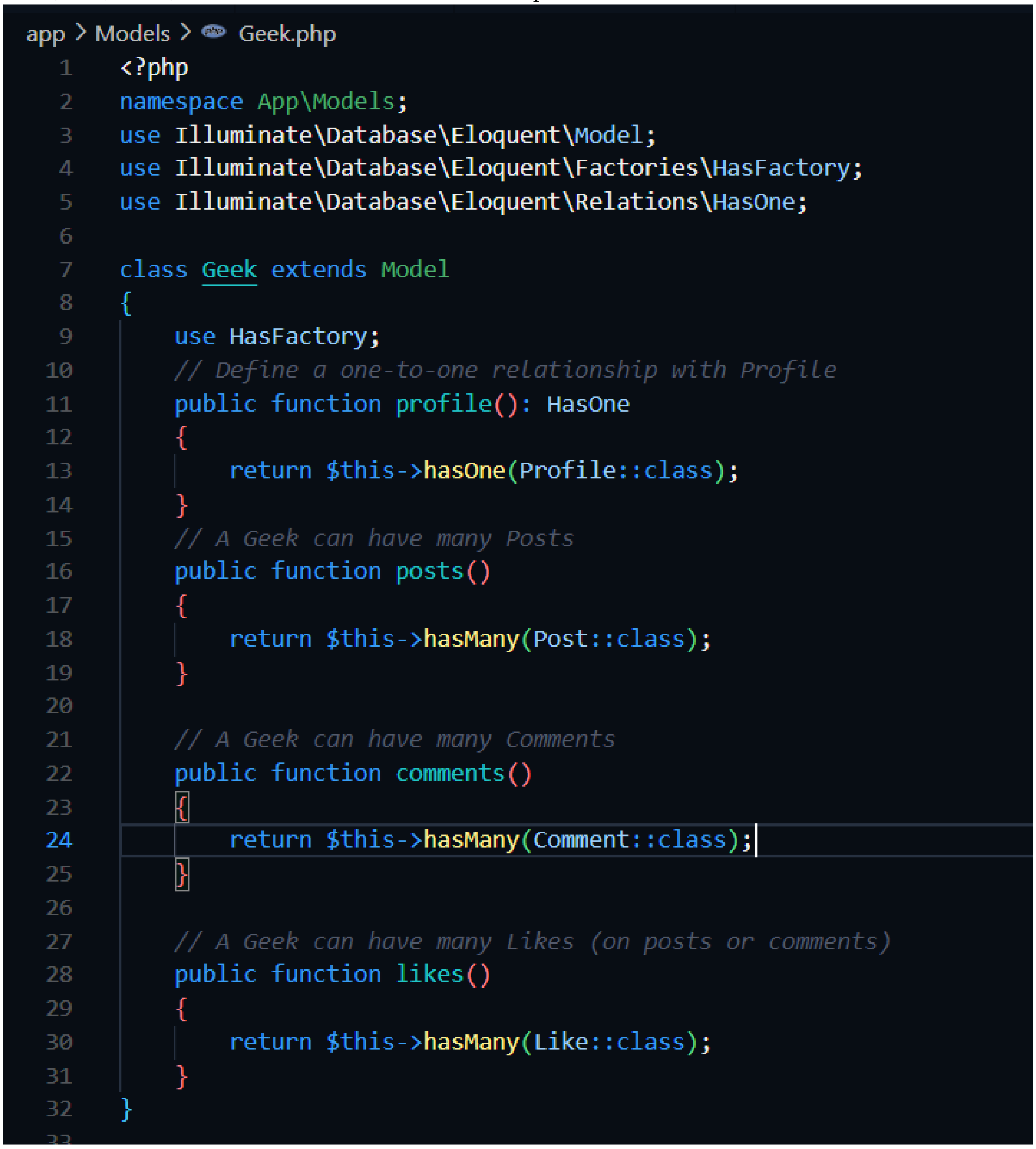


database > migrations > 2024_11_03_142859_create_likes_table.php

```php
<?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('likes', function (Blueprint $table) {
            $table->id();
            $table->foreignId('geek_id')->constrained('geeks')->onDelete('cascade');
            $table->unsignedBigInteger('likeable_id');
            $table->string('likeable_type');
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('likes');
    }
};
```

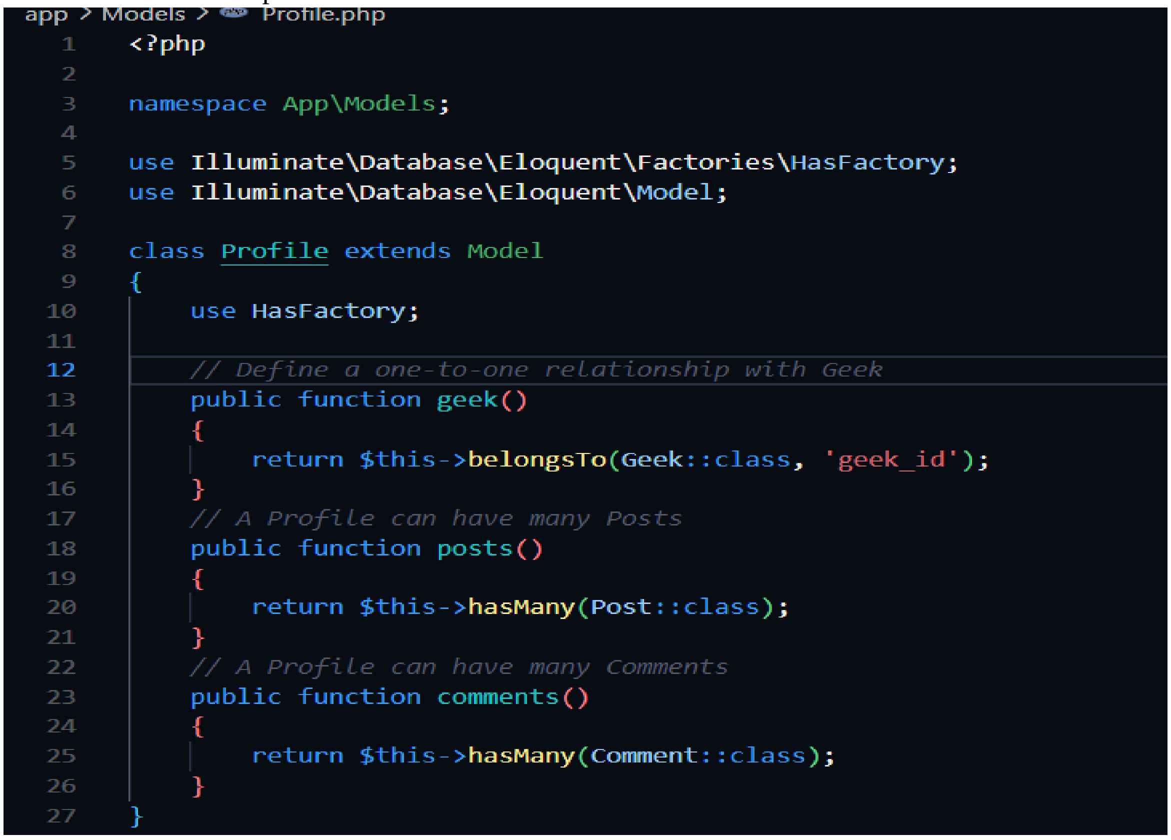
Models:

1. **Geek.php:** Defines the Geek model representing a user in the application, with attributes like name, dob, email, mobile and associated relationships.



```php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\HasOne;

class Geek extends Model
{
    use HasFactory;
    // Define a one-to-one relationship with Profile
    public function profile(): HasOne
    {
        return $this->hasOne(Profile::class);
    }
    // A Geek can have many Posts
    public function posts()
    {
        return $this->hasMany(Post::class);
    }

    // A Geek can have many Comments
    public function comments()
    {
        return $this->hasMany(Comment::class);
    }

    // A Geek can have many Likes (on posts or comments)
    public function likes()
    {
        return $this->hasMany(Like::class);
    }
}
```

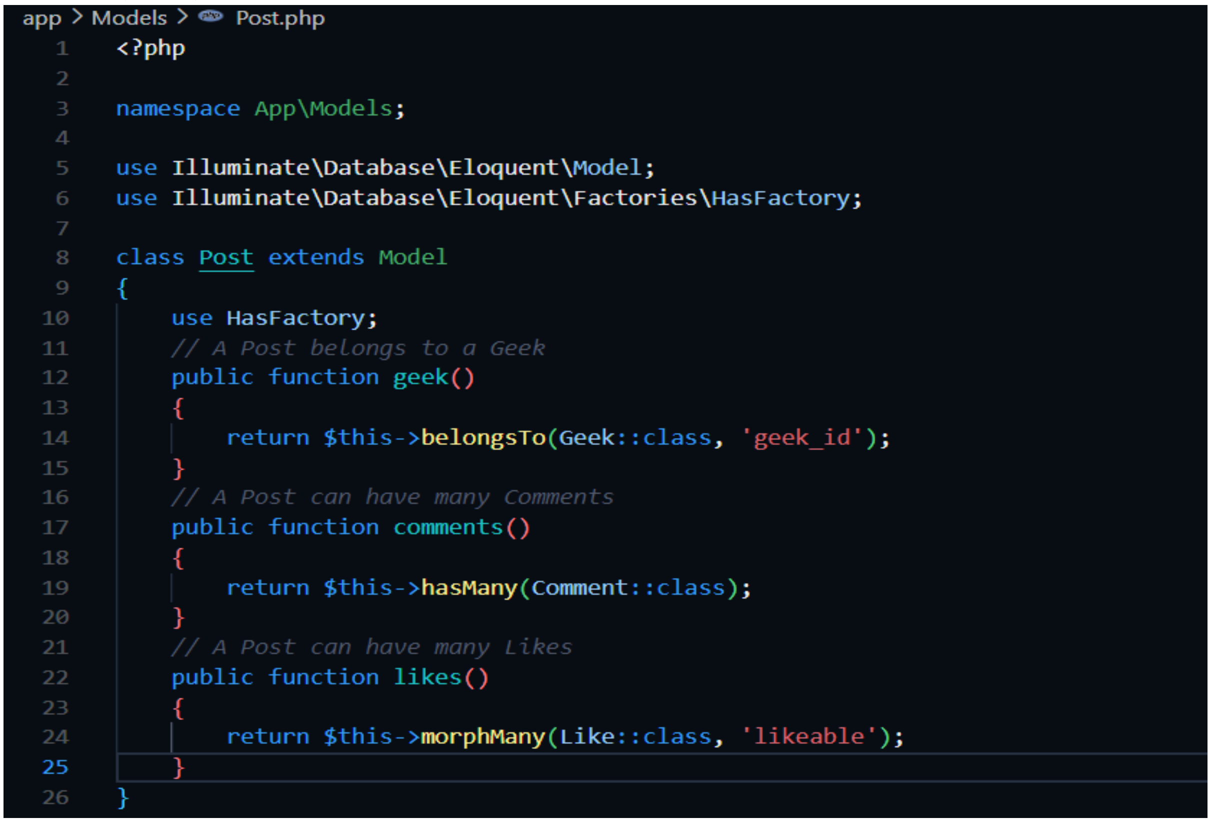
2. **Profile.php:** Represents user profile details in the application, linked to a Geek through a one-to-one relationship.



app > Models > Profile.php

```php
<?php

namespace App\Models;

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

class Profile extends Model
{
    use HasFactory;

    // Define a one-to-one relationship with Geek
    public function geek()
    {
        return $this->belongsTo(Geek::class, 'geek_id');
    }
    // A Profile can have many Posts
    public function posts()
    {
        return $this->hasMany(Post::class);
    }
    // A Profile can have many Comments
    public function comments()
    {
        return $this->hasMany(Comment::class);
    }
}
```

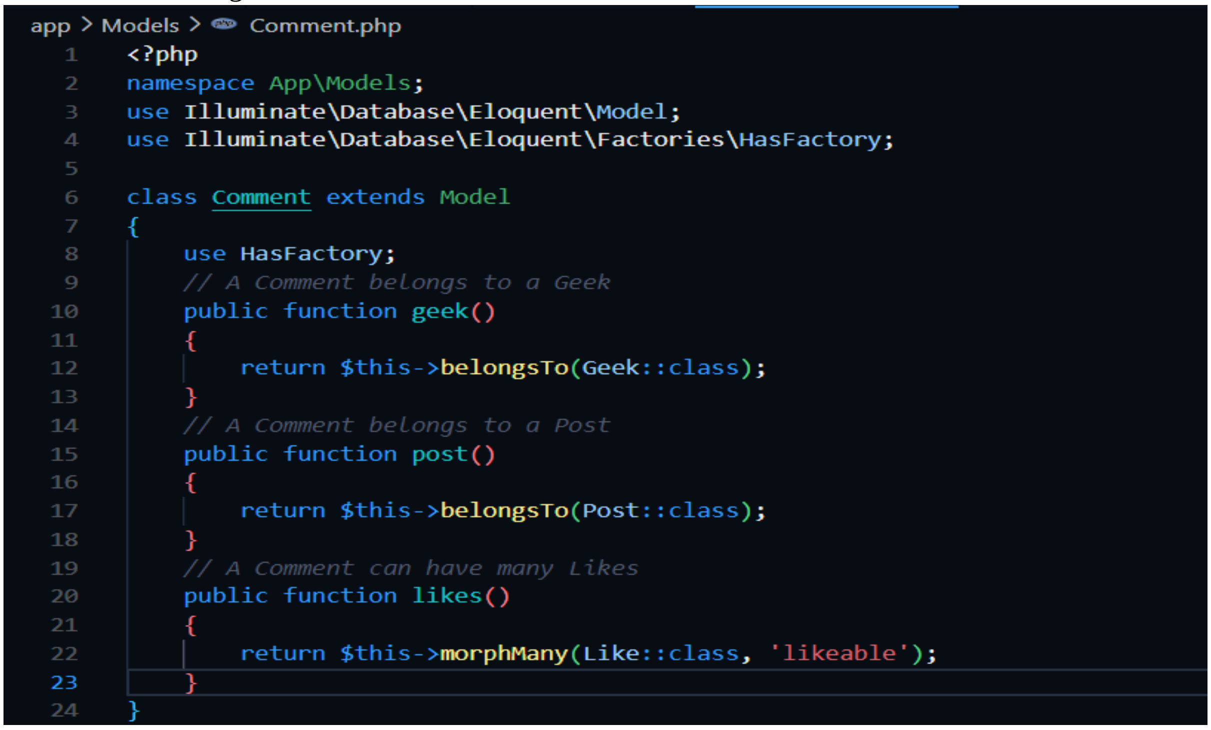
3. **Post.php:** Defines a Post model representing content shared by Geeks, with relationships to comments, likes, and the Geek who created it.



app > Models > Post.php

```php
<?php

namespace App\Models;

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

class Post extends Model
{
    use HasFactory;
    // A Post belongs to a Geek
    public function geek()
    {
        return $this->belongsTo(Geek::class, 'geek_id');
    }
    // A Post can have many Comments
    public function comments()
    {
        return $this->hasMany(Comment::class);
    }
    // A Post can have many Likes
    public function likes()
    {
        return $this->morphMany(Like::class, 'likeable');
    }
}
```

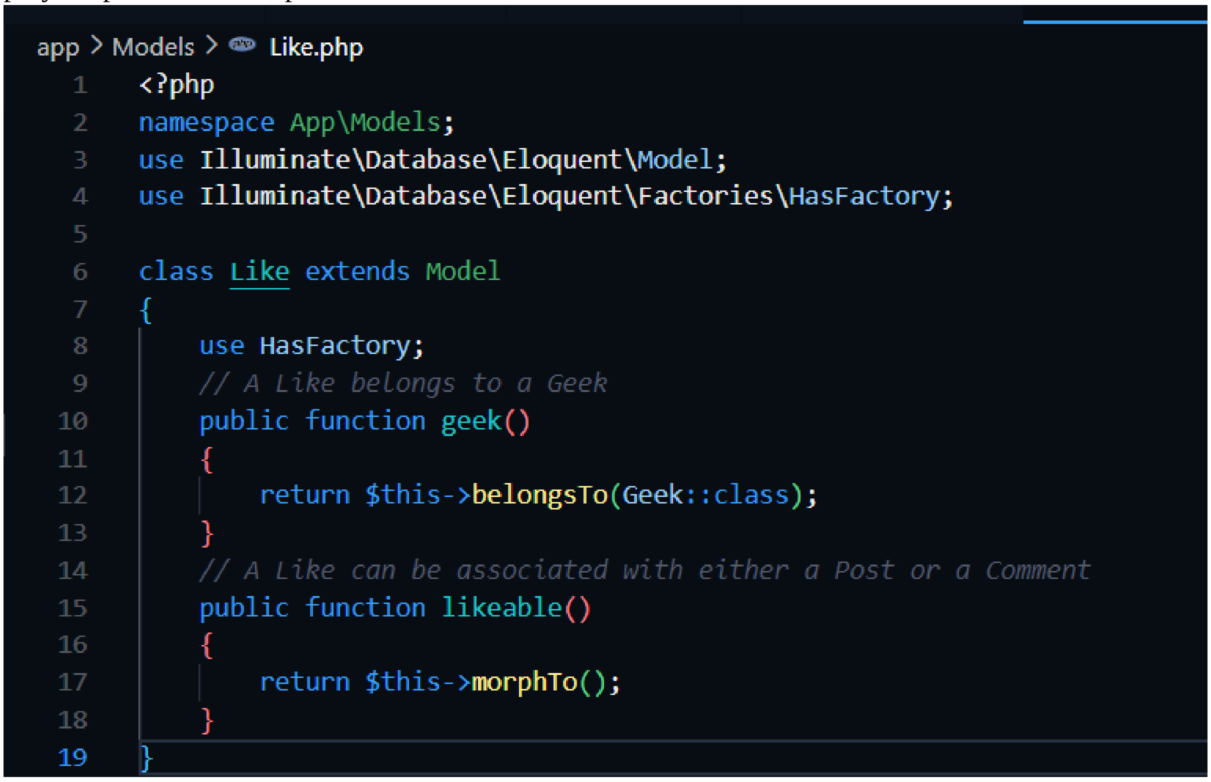
4. **Comment.php:** Represents comments made on posts, linking each comment to both a post and the commenting Geek.



app > Models > Comment.php

```php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class Comment extends Model
{
    use HasFactory;
    // A Comment belongs to a Geek
    public function geek()
    {
        return $this->belongsTo(Geek::class);
    }
    // A Comment belongs to a Post
    public function post()
    {
        return $this->belongsTo(Post::class);
    }
    // A Comment can have many Likes
    public function likes()
    {
        return $this->morphMany(Like::class, 'likeable');
    }
}
```

5. **Like.php:** Defines the Like model that allows Geeks to like posts or comments, supporting polymorphic relationships.

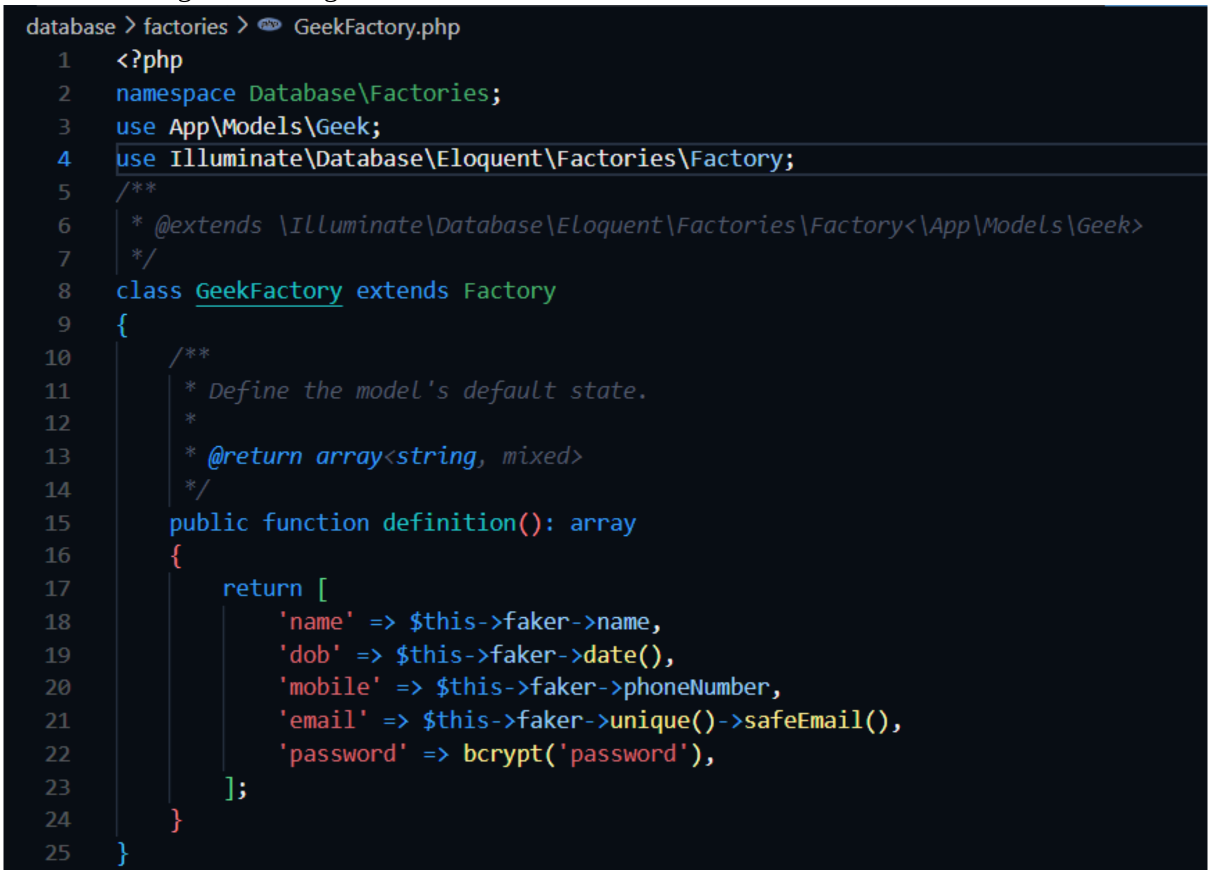


app > Models > Like.php

```php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class Like extends Model
{
    use HasFactory;
    // A Like belongs to a Geek
    public function geek()
    {
        return $this->belongsTo(Geek::class);
    }
    // A Like can be associated with either a Post or a Comment
    public function likeable()
    {
        return $this->morphTo();
    }
}
```

Factories:

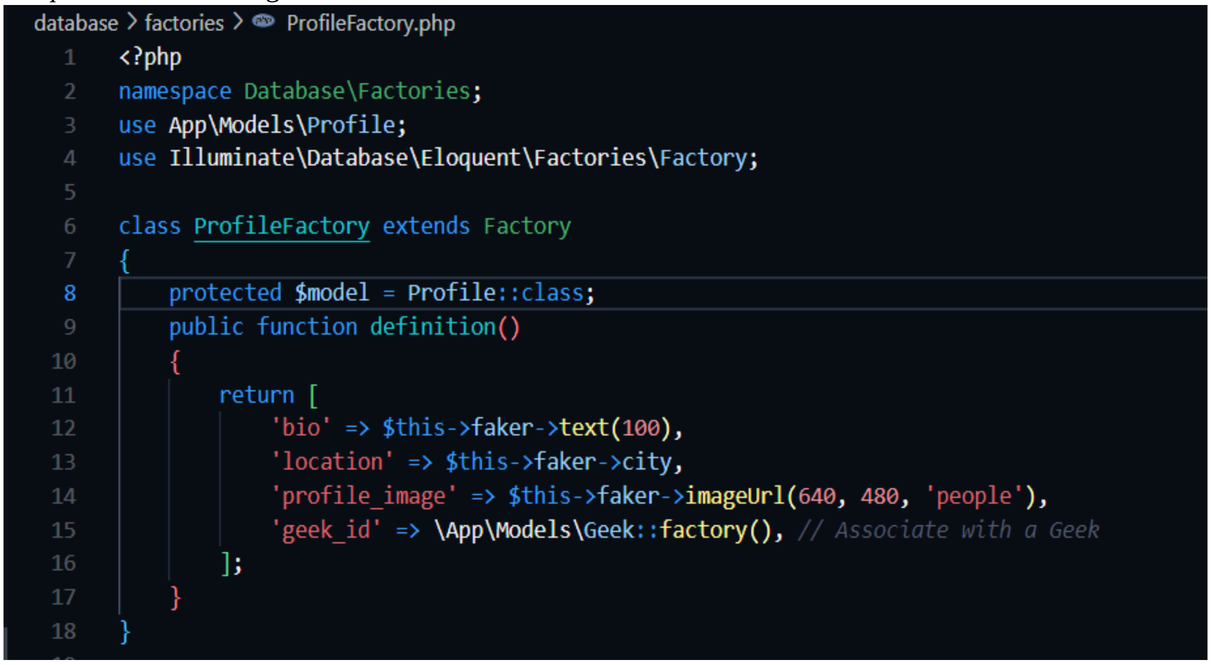
1. **GeekFactory.php:** Generates fake data for the Geek model, including unique emails, to facilitate testing and seeding.



database > factories > GeekFactory.php

```php
<?php
namespace Database\Factories;
use App\Models\Geek;
use Illuminate\Database\Eloquent\Factories\Factory;
/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Geek>
 */
class GeekFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        return [
            'name' => $this->faker->name,
            'dob' => $this->faker->date(),
            'mobile' => $this->faker->phoneNumber,
            'email' => $this->faker->unique()->safeEmail(),
            'password' => bcrypt('password'),
        ];
    }
}
```

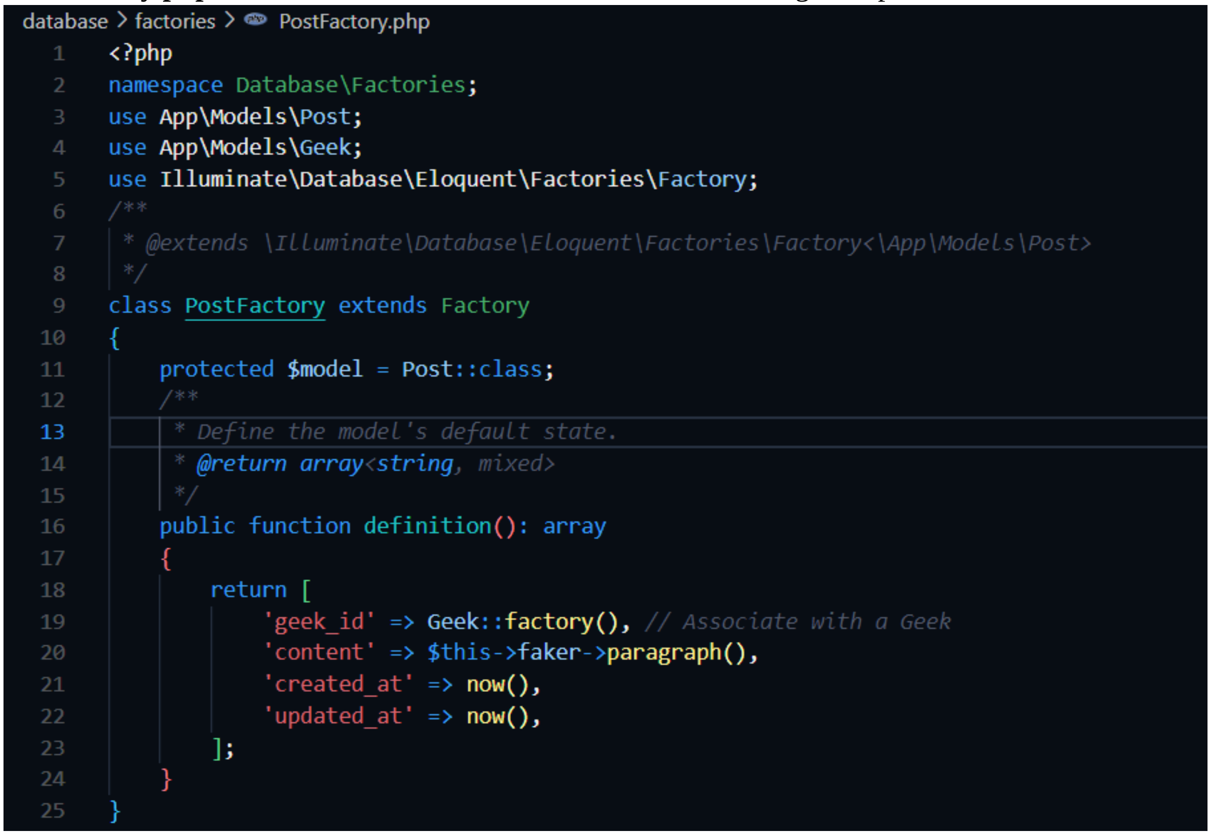
2. **ProfileFactory.php:** Creates dummy data for Profile instances, associating each with a unique Geek for testing.



database > factories > ProfileFactory.php

```php
<?php
namespace Database\Factories;
use App\Models\Profile;
use Illuminate\Database\Eloquent\Factories\Factory;

class ProfileFactory extends Factory
{
    protected $model = Profile::class;
    public function definition()
    {
        return [
            'bio' => $this->faker->text(100),
            'location' => $this->faker->city,
            'profile_image' => $this->faker->imageUrl(640, 480, 'people'),
            'geek_id' => \App\Models\Geek::factory(), // Associate with a Geek
        ];
    }
}
```

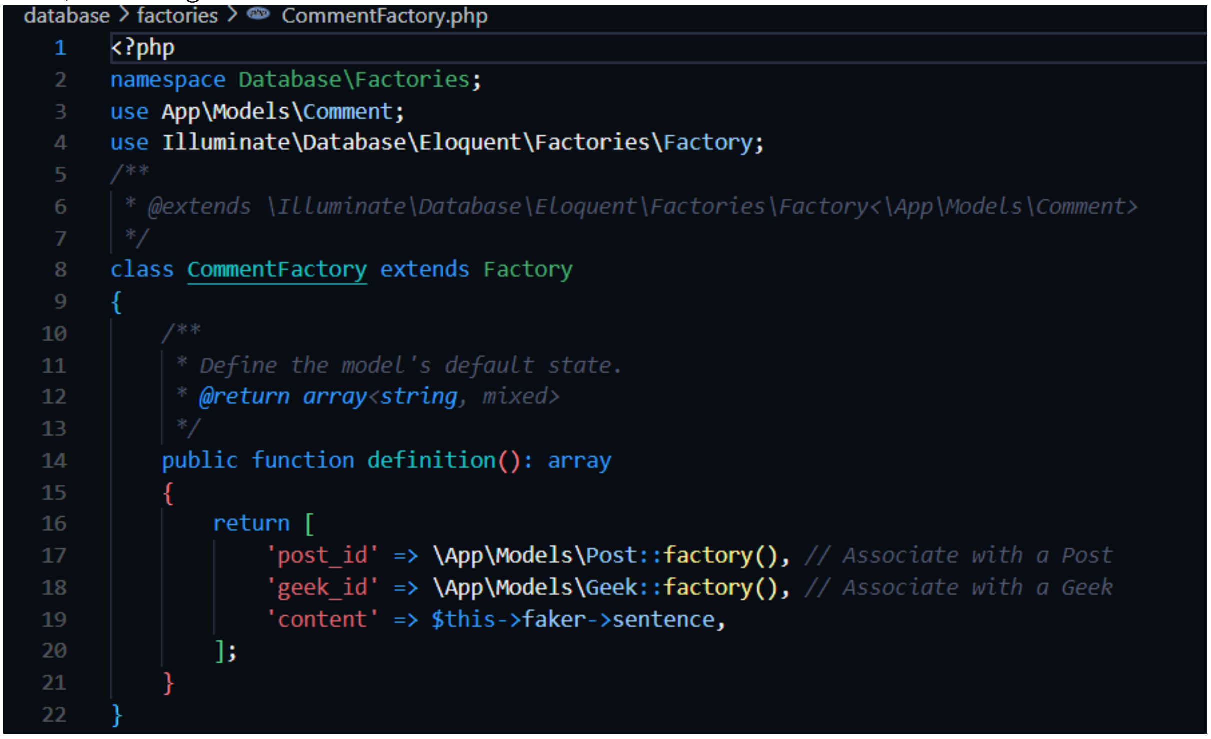
3. **PostFactory.php:** Generates test data for Post instances, associating each post with a Geek.



database > factories > PostFactory.php

```php
<?php
namespace Database\Factories;
use App\Models\Post;
use App\Models\Geek;
use Illuminate\Database\Eloquent\Factories\Factory;
/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Post>
 */
class PostFactory extends Factory
{
    protected $model = Post::class;
    /**
     * Define the model's default state.
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        return [
            'geek_id' => Geek::factory(), // Associate with a Geek
            'content' => $this->faker->paragraph(),
            'created_at' => now(),
            'updated_at' => now(),
        ];
    }
}
```

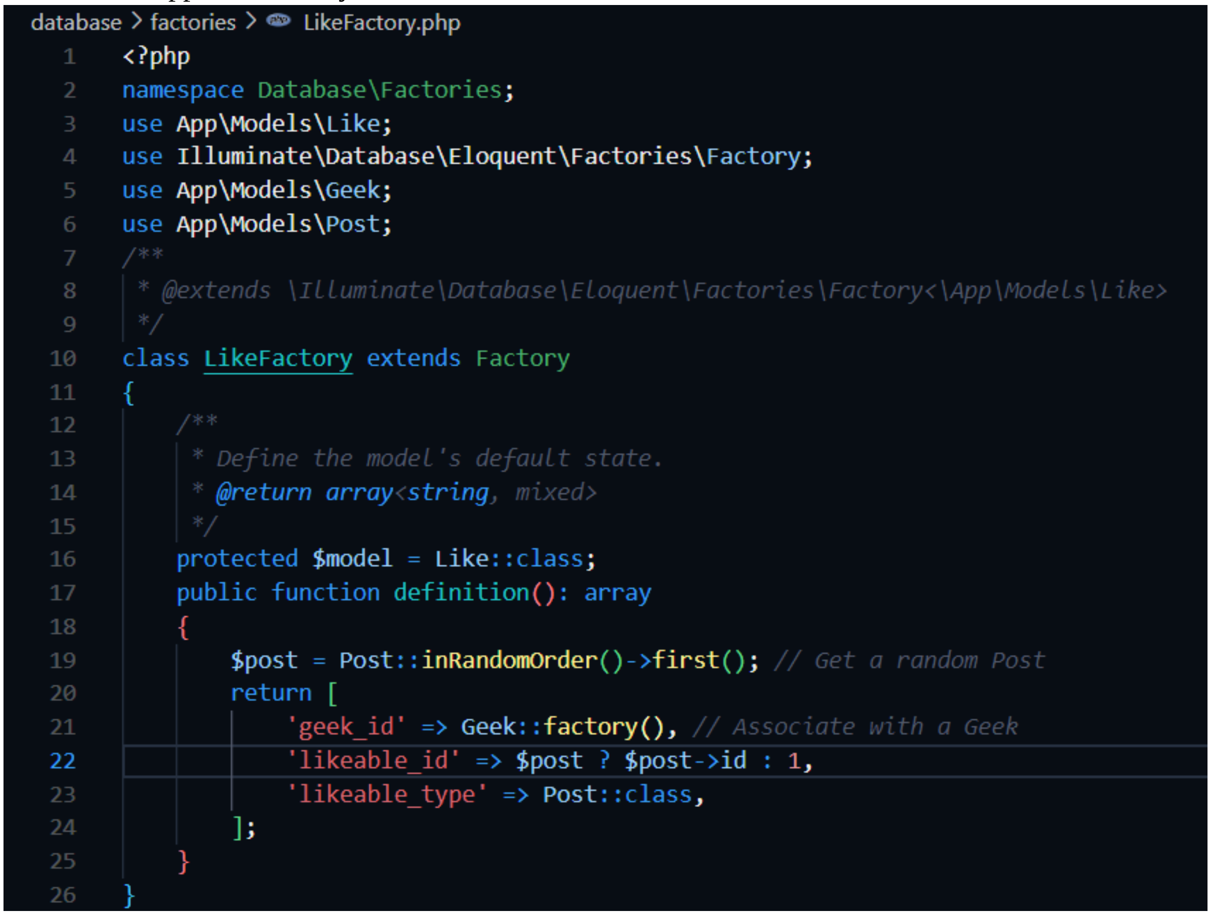
4. **CommentFactory.php:** Creates random comments for posts, each tied to a specific post and Geek, for testing interactions.



database > factories > CommentFactory.php

```php
<?php
namespace Database\Factories;
use App\Models\Comment;
use Illuminate\Database\Eloquent\Factories\Factory;
/**
 * @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 [
            'post_id' => \App\Models\Post::factory(), // Associate with a Post
            'geek_id' => \App\Models\Geek::factory(), // Associate with a Geek
            'content' => $this->faker->sentence,
        ];
    }
}
```

5. **LikeFactory.php:** Generates test data for likes on posts or comments, with polymorphic relations to support flexibility.

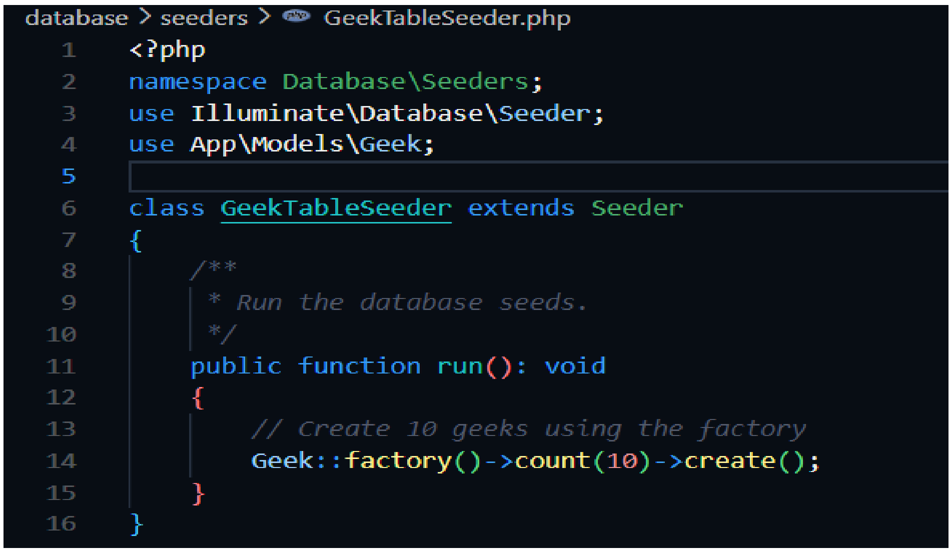


database > factories > LikeFactory.php

```php
<?php
namespace Database\Factories;
use App\Models\Like;
use Illuminate\Database\Eloquent\Factories\Factory;
use App\Models\Geek;
use App\Models\Post;
/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Like>
 */
class LikeFactory extends Factory
{
    /**
     * Define the model's default state.
     * @return array<string, mixed>
     */
    protected $model = Like::class;
    public function definition(): array
    {
        $post = Post::inRandomOrder()->first(); // Get a random Post
        return [
            'geek_id' => Geek::factory(), // Associate with a Geek
            'likeable_id' => $post ? $post->id : 1,
            'likeable_type' => Post::class,
        ];
    }
}
```

Seeders:

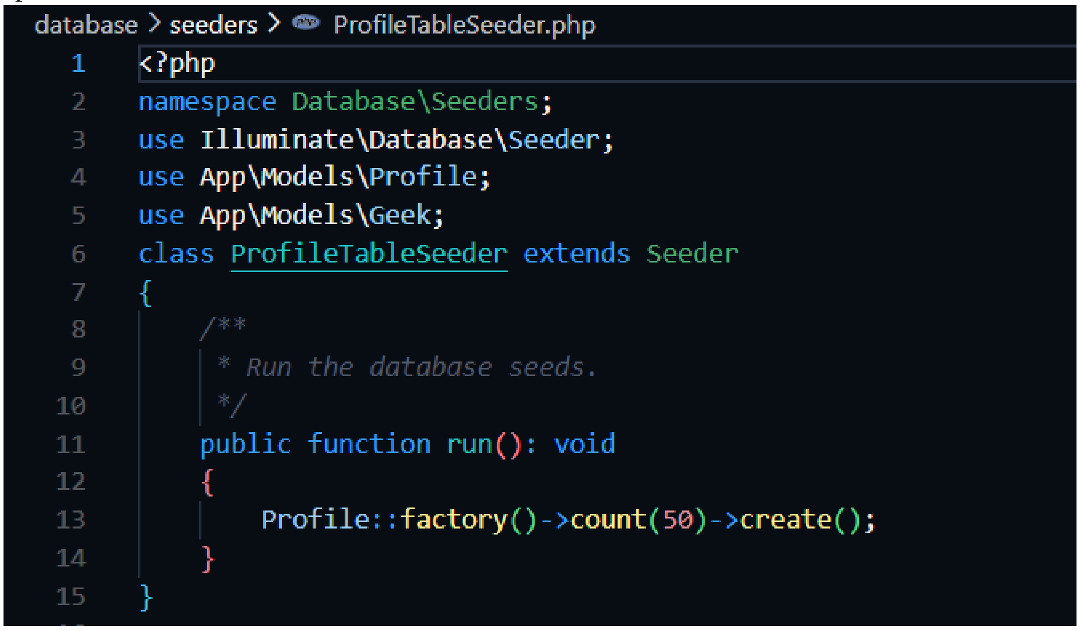
1. **GeekTableSeeder.php:** Seeds the database with an initial Geek instance, manually setting attributes like name and email.



database > seeders > GeekTableSeeder.php

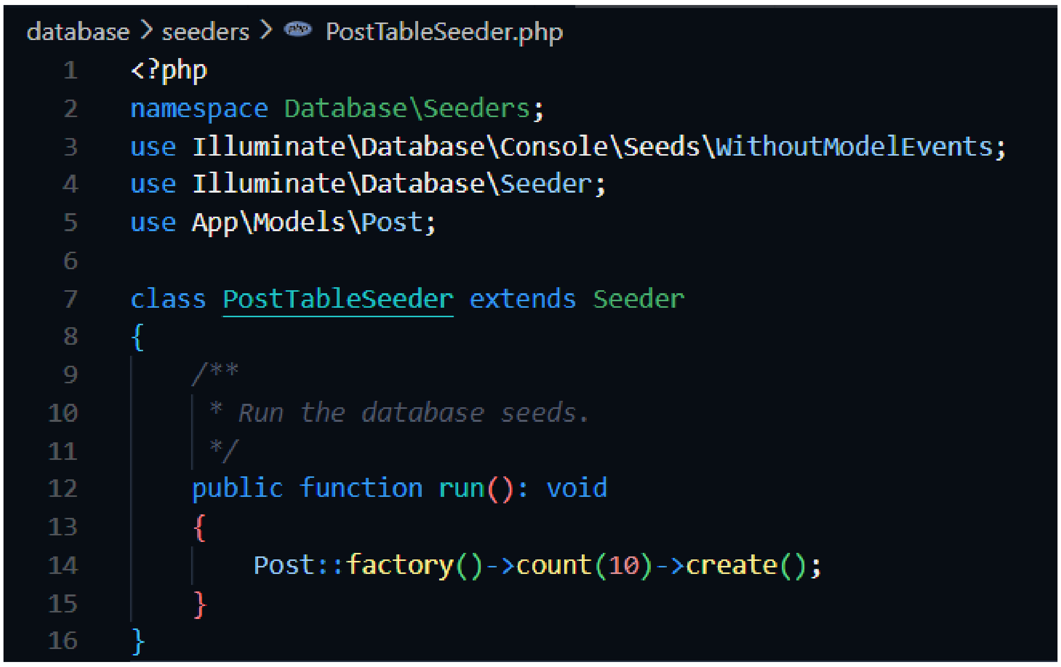
```php
<?php
namespace Database\Seeders;
use Illuminate\Database\Seeder;
use App\Models\Geek;

class GeekTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        // Create 10 geeks using the factory
        Geek::factory()->count(10)->create();
    }
}
```

2. **ProfileTableSeeder.php:** Populates the database with Profile records, each linked to a specific Geek instance.



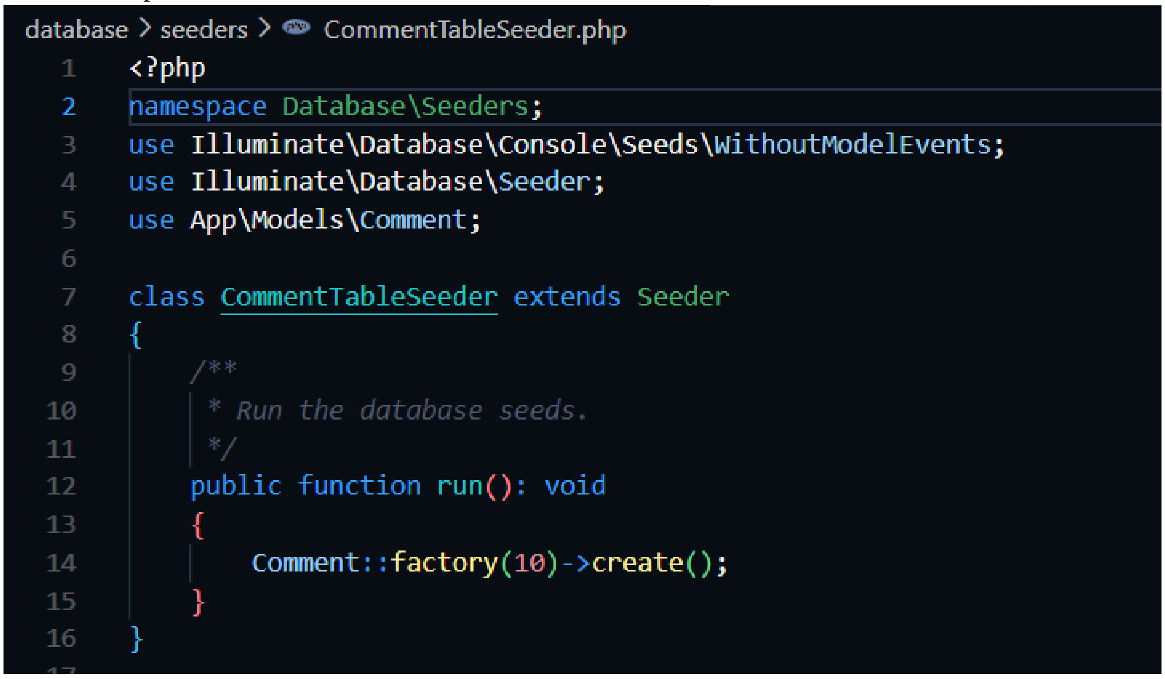
```php
<?php
namespace Database\Seeders;
use Illuminate\Database\Seeder;
use App\Models\Profile;
use App\Models\Geek;
class ProfileTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        Profile::factory()->count(50)->create();
    }
}
```

3. **PostTableSeeder.php:** Seeds posts in the database, associating each post with a specific Geek.



```php
<?php
namespace Database\Seeders;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Post;

class PostTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        Post::factory()->count(10)->create();
    }
}
```

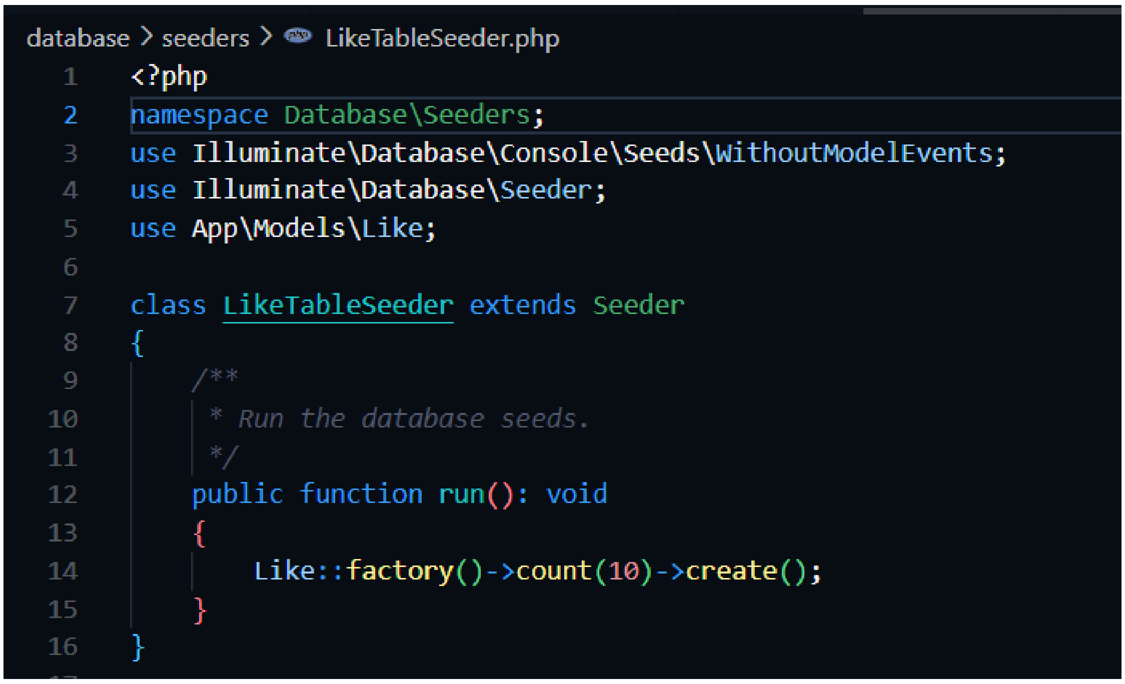
4. **CommentTableSeeder.php:** Add sample comments to posts, linking each comment to a Geek and a post.



database > seeders > CommentTableSeeder.php

```php
<?php
namespace Database\Seeders;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Comment;

class CommentTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        Comment::factory(10)->create();
    }
}
```

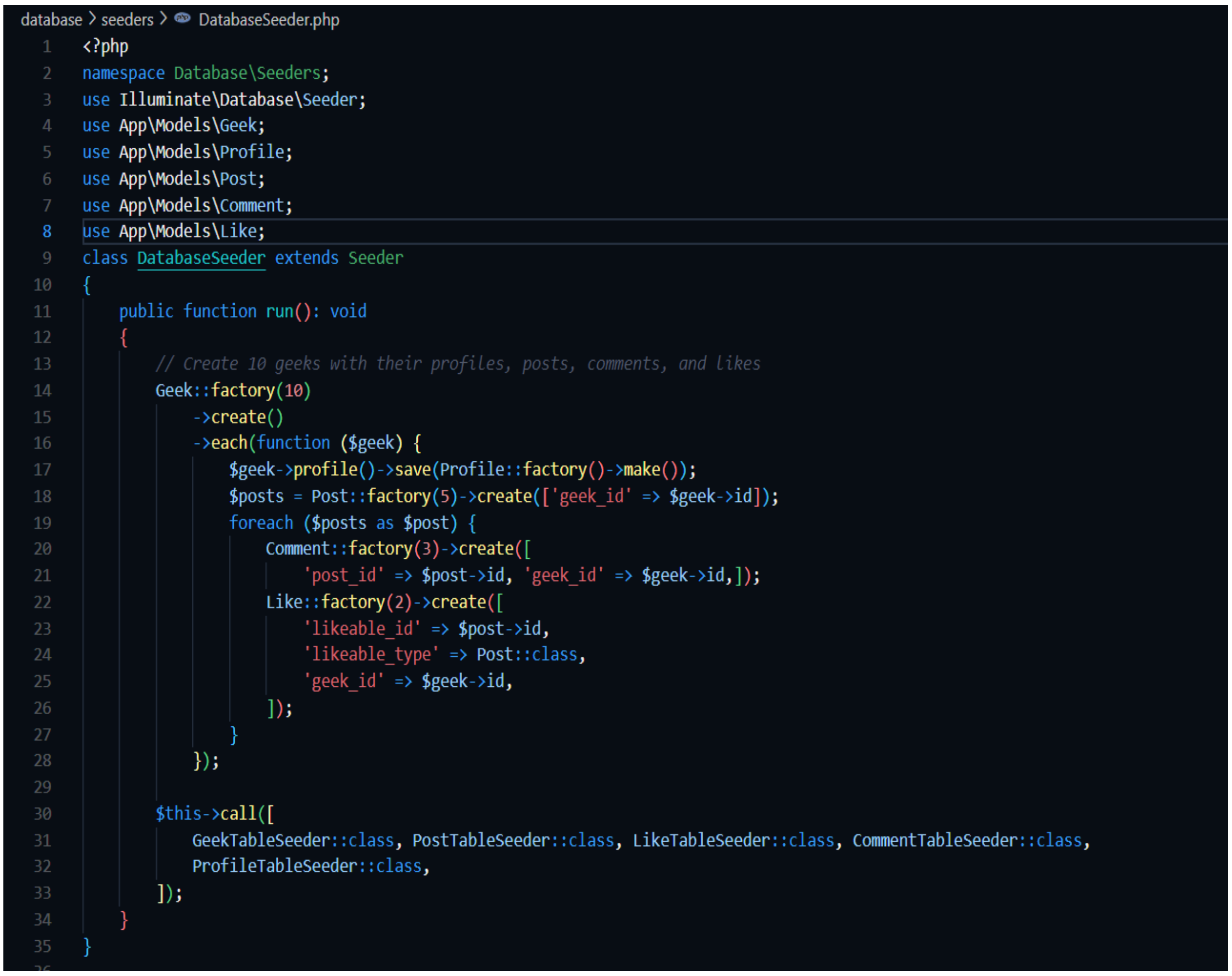
5. **LikeTableSeeder.php:** Inserts likes into the database, associating each like with a post or comment and a Geek.



database > seeders > LikeTableSeeder.php

```php
<?php
namespace Database\Seeders;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Like;

class LikeTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        Like::factory()->count(10)->create();
    }
}
```

DatabaseSeeder.php: Coordinates the seeding of all models, calling individual seeders to populate the database with sample data for testing.



```php
<?php
namespace Database\Seeders;
use Illuminate\Database\Seeder;
use App\Models\Geek;
use App\Models\Profile;
use App\Models\Post;
use App\Models\Comment;
use App\Models\Like;
class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        // Create 10 geeks with their profiles, posts, comments, and likes
        Geek::factory(10)
            ->create()
            ->each(function ($geek) {
                $geek->profile()->save(Profile::factory()->make());
                $posts = Post::factory(5)->create(['geek_id' => $geek->id]);
                foreach ($posts as $post) {
                    Comment::factory(3)->create([
                        'post_id' => $post->id, 'geek_id' => $geek->id,]);
                    Like::factory(2)->create([
                        'likeable_id' => $post->id,
                        'likeable_type' => Post::class,
                        'geek_id' => $geek->id,
                    ]);
                }
            });

        $this->call([
            GeekTableSeeder::class, PostTableSeeder::class, LikeTableSeeder::class, CommentTableSeeder::class,
            ProfileTableSeeder::class,
        ]);
    }
}
```