The five code review has been done by following the below given rubrics. Based on this rubric, the code has been reviewed.

Code will be marked by me using the following rubric.

10%	10%	10%	20%	10%	20%	20%
The student has submitted model definitions, along with migration files (all in pdf). With no obvious errors. If student has just submitted all files to be safe do not award these marks.	The model definitions include examples of relationship methods implemented correctly.	The student has submitted source files for database seeding. With no obvious errors.	Database seeding seeds data with relationships. With no obvious errors.	Factories are used to seed random, realistic, data. With no obvious errors.	Factories are used to seed random, realistic, relationships between data. With no obvious errors.	All of the student's code follows correct naming conventions and is easy to follow.

Name – OSIAN JONES **Student No. – Not Given**

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;
```

1. Model Definitions and Migration Files

- **Models**: The User, Post, and Comment models are well-structured, with clear relationships using hasMany and belongsTo. Docblocks are used to document relationships, following best practices.
- **Migrations**: Migration files are well-written, with proper foreign key constraints and onDelete('cascade')/onUpdate('cascade') for referential integrity. Laravel conventions are followed.
- Rating: 20%

2. Database Seeding

- **Seeders**: The UserTableSeeder, PostsTableSeeder, and CommentsTableSeeder correctly generate static and random data, linking records properly. The DatabaseSeeder` ensures the correct order for seeding.
- Rating: 20%

3. Factories

- **Factories**: The UserFactory, PostFactory, and CommentFactory generate realistic, random data, linking records as required. The use of Laravel conventions for generating related data is well executed.
- Rating: 20%

4. Overall Code Quality and Naming Conventions

- **Naming**: Consistent use of Laravel naming conventions. Relationship method names are intuitive.
- **Code Quality**: The code is clean, well-organized, and maintainable. Docblocks and clear structure enhance readability.
- Rating: 20%

5. Extra Files

- The submission contains no unnecessary files. All the included files (models, migrations, seeders, and factories) are relevant to the task and contribute to setting up database relationships and seeding.
- Rating: 20%

Name – Zichun Liu **Student No. – Not given**

SQL code

Complete sql code

```
/*
 Navicat Premium Data Transfer

 Source Server         : php
 Source Server Type    : MySQL
 Source Server Version : 50726
 Source Host           : localhost:3306
 Source Schema         : blogs

 Target Server Type    : MySQL
 Target Server Version : 50726
 File Encoding         : 65001

 Date: 30/10/2024 09:50:02
*/
```

1. Model Definitions and Migration Files

- Model relationships are well defined with correct relationships, such as hasMany, belongsTo, and belongsToMany. Each model uses the naming convention given by Laravel for the relationship.
- The migrations are set up with appropriate foreign key constraints, timestamps, and default values. Relationships among the models should be respected, and referential integrity should be maintained.
- Rating: 20%

2. Database Seeding

- Databases are seeded correctly, creating the database with static records, using factories for the creation of random data. The seeding is created following the model relationships in order to create realistic test data.
- Rating: 20%

3. Factories

- The factories, like AdminRoleFactory, AdminUserFactory, and BlogFactory, are generating random data appropriately, and that data should be realistic. Use of Faker will provide varied data for testing; relationships are set up properly.
- Rating: 20%

4. Overall Code Quality and Naming Conventions

- The code in general follows the naming conventions for model, table, and relation names according to Laravel. Code is clean, well-organised, readable, and comments or docblocks are used where necessary.
- Rating: 20%

5. Extra Files

- The SQL code files are unnecessary for this assignment. The focus should be on Laravel migrations, models, and seeders. These files should be excluded to avoid possible deduction of marks.
- Recommendation: Only include Laravel source files as per the assignment requirements.
- Rating: 10%

Name - Meihuan Li **Student No. – 2311544**

1. Model Definitions and Migration Files

- **Models**: The models (Cat, Dog, Breeds, PetSupplies, Comment, Post, User) are well-structured and follow Laravel conventions for defining relationships, including hasMany, belongsTo, and belongsToMany. The use of fillable ensures that only the intended attributes are mass-assigned.
- **Migrations**: The migrations for creating tables such as cats, dogs, breeds, and others are correctly implemented. Foreign key constraints are set up properly with onDelete('cascade') and onDelete('set null'), ensuring referential integrity.
- Rating: 20%

2. Database Seeding

- **Seeders**: Seeders (BreedsTableSeeder, CatTableSeeder, DogTableSeeder, etc.) are well-organized and correctly insert both static and random data. The use of DB::statement('SET FOREIGN_KEY_CHECKS=0;') ensures that foreign key constraints don't interfere during seeding.
- Rating: 20%

3. Factories

- **Factories** (BreedsFactory, CatFactory, DogFactory, CommentFactory, etc.) are defined to generate random data, using Faker for attributes such as name, country, color, etc. The use of inRandomOrder()->first() to link related models (e.g., breed_id for Cat) is a good practice.
- Rating: 20%

4. Overall Code Quality and Naming Conventions

- **Naming Conventions**: The code follows Laravel's conventions, with model names in singular and table names in plural. The relationship methods are named intuitively (e.g., petSupplies(), posts(), comments()).
- **Code Quality**: The code is clean and well-structured. The use of docblocks for methods and clear variable names ensures the code is readable and maintainable.
- Rating: 20%

5. Extra Files

- The submission contains no unnecessary files. All the included files (models, migrations, seeders, and factories) are relevant to the task and contribute to setting up database relationships and seeding.
- Rating: 20%

Name – Annabel Ryan **Student No. – 2215461**

1. Model Definitions and Migration Files

- **Models** (Pokemon, Trainer):
  - Models are well-defined, following Laravel conventions. The relationships between Trainer and Pokemon are set by the hasMany and belongsTo methods.
- **Migrations**:
  - As can be seen, the migrations for the trainers and pokemon tables are created in a neat way, considering the fields that are necessary and foreign key constraints across related tables for maintaining relational integrity.
- Rating: 20%

2. Database Seeding

- **Seeders** (DatabaseSeeder, PokemonTableSeeder, TrainerTableSeeder):
  - Seeders are properly implemented to insert both static and random data. The use of factories in PokemonTableSeeder and TrainerTableSeeder ensures realistic and varied data.
- Rating: 20%

3. Factories

- **Factories** (PokemonFactory, TrainerFactory):
  - The factories are defined to generate random and realistic data, using Faker for attributes such as name, level, etc. The use of relationships in the factories (e.g., linking Trainer with Pokemon) is well executed.
- Rating: 20%

4. Overall Code Quality and Naming Conventions

- **Naming Conventions**:
  - The code follows Laravel's naming conventions, with clear, descriptive names for models, migrations, and factories. The naming of relationship methods (e.g., pokemons() in Trainer) is intuitive and adheres to best practices.
- **Code Quality**:
  - The code is well-organized, clean, and easy to follow. The use of docblocks and meaningful variable names enhances readability and maintainability.
- Rating: 20%

5. Extra Files

- The submission contains no unnecessary files. All the included files (models, migrations, seeders, and factories) are relevant to the task and contribute to setting up database relationships and seeding.
- Rating: 20%

Name – Sam Davies **Student No. – 2214477**

1. Model Definitions and Migration Files

- **Models** (User, Profile, Post, Comment, Notification):
  - The models are well-structured with clear relationships. The User model has many-to-many and one-to-many relationships with Post, Comment, and Notification.
- **Migrations**:
  - The migrations for creating tables (users, profiles, posts, comments, post_user_like, notifications) are implemented correctly with foreign key constraints and referential integrity.
- Rating: 20%

2. Database Seeding

- **Seeders** (DatabaseSeeder, UserTableSeeder, ProfileTableSeeder, PostTableSeeder, LikeTableSeeder, NotificationTableSeeder):
  - Seeders are implemented effectively, with factories used to generate realistic data. The use of hasPosts() in the UserTableSeeder and random attachment of posts to users in LikeTableSeeder is a good approach for testing.
- Rating: 20%

3. Factories

- **Factories** (UserFactory, ProfileFactory, PostFactory, CommentFactory, NotificationFactory):
  - The factories are correctly defined, using Faker to generate realistic test data for attributes like name, email, comment_text, etc. The use of random associations (e.g., User::inRandomOrder()) is well implemented.
- Rating: 20%

4. Overall Code Quality and Naming Conventions

- **Naming Conventions**:
  - The code follows Laravel's naming conventions. The relationship method names (likedPosts(), posts(), etc.) are descriptive and adhere to best practices.
- **Code Quality**:
  - The code is clean, well-commented, and easy to follow. The use of docblocks for methods helps clarify the relationships and purpose of each method.
- Rating: 20%

5. Extra Files

- The submission contains no unnecessary files. All the included files (models, migrations, seeders, and factories) are relevant to the task and contribute to setting up database relationships and seeding.
- Rating: 20%