Name: **Oliver Harris**
Student id: **952283**

Models

Comment.php

app > Comment.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Comment extends Model
{
    public function user()
    {
        return $this->belongsTo("App\User");
    }


    public function post()
    {
        return $this->belongsTo("App\Post");
    }
}
```

Notepad.php

app > Notepad.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Notepad extends Model
{

    public function user()
    {
        return $this->belongsTo("App\User");
    }
}
```

Post.php

app > Post.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    public function user()
    {
        return $this->belongsTo("App\User");
    }
    public function tags()
    {
        return $this->belongsToMany("App\Tag");
    }
    public function comments()
    {
        return $this->hasMany("App\Comment");
    }
}

```

Tag.php

app > Tag.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Tag extends Model
{
    public function posts()
    {
        return $this->belongsToMany("App\Post");
    }
}
```

User.php

User.php app/User.php

```php
<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use Notifiable;


    public function posts()
    {
        return $this->hasMany("App\Post");
    }

    public function notepad()
    {
        return $this->hasOne("App\Notepad");
    }

    public function comments()
    {
        return $this->hasMany("App\Comment");
    }

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

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```

Migrations

(in order of creation and omitting password_rests and failed_jobs table)

2014_10_12_000000_create_users_table.php

database > migrations > 2014_10_12_000000_create_users_table.php

```php
<?php

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

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

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

```

2019_11_04_192208_create_tags_table.php

database > migrations > 2019_11_04_192208_create_tags_table.php

```php
<?php

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

class CreateTagsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('tags', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string("tag")->unique();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('tags');
    }
}

```

2019_11_04_192214_create_posts_table.php

database > migrations > 2019_11_04_192214_create_posts_table.php

```php
<?php

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

class CreatePostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string("title");
            $table->longText("content");
            $table->unsignedBigInteger("user_id");
            $table->timestamps();

            $table->foreign("user_id")->references("id")->on("users")->onDelete("cascade")->onUpdate("cascade");
        });
    }

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

2019_11_04_192220_create_comments_table.php

database > migrations > 2019_11_04_192220_create_comments_table.php

```php
<?php

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

class CreateCommentsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('comments', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string("comment");
            $table->unsignedBigInteger("user_id");
            $table->unsignedBigInteger("post_id");
            $table->timestamps();

            $table->foreign("user_id")->references("id")->on("users")->onDelete("cascade")->onUpdate("cascade");

            $table->foreign("post_id")->references("id")->on("posts")->onDelete("cascade")->onUpdate("cascade");

        });
    }

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

```

2019_11_04_192226_create_notepads_table.php

database > migrations > 2019_11_04_192226_create_notepads_table.php

```php
<?php

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

class CreateNotepadsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('notepads', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->longText("content");
            $table->unsignedBigInteger("user_id")->unique();
            $table->timestamps();


            $table->foreign("user_id")->references("id")->on("users")->onDelete("cascade")->onUpdate("cascade");
        });
    }

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

2019_11_04_192939_create_post_tags.php

database > migrations > 2019_11_04_192939_create_post_tags.php

```php
<?php

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

class CreatePostTags extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('post_tag', function (Blueprint $table) {
            $table->primary(["post_id","tag_id"]);
            $table->unsignedBigInteger("post_id");
            $table->unsignedBigInteger("tag_id");
            $table->timestamps();

            $table->foreign("post_id")->references("id")->on("posts")->onDelete("cascade")->onUpdate("cascade");
            $table->foreign("tag_id")->references("id")->on("tags")->onDelete("cascade")->onUpdate("cascade");
        });
    }

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

Factories

CommentFactory.php

database > factories > CommentFactory.php

```php
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */

use App\Comment;
use Faker\Generator as Faker;

$factory->define(Comment::class, function (Faker $faker) {
    return [
        "comment" => $faker->realText(100,3),
    ];
});

```

NotepadFactory.php

database > factories > NotepadFactory.php

```php
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */

use App\Notepad;
use Faker\Generator as Faker;

$factory->define(Notepad::class, function (Faker $faker) {
    return [
        "content" =>  $faker->realText(500,3),
    ];
});

```

PostFactory.php

database > factories > PostFactory.php

```php
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */

use App\Post;
use Faker\Generator as Faker;

$factory->define(Post::class, function (Faker $faker) {
    return [
        "title" =>  $faker->realText(30,1),
        "content" =>  $faker->realText(850,2),
    ];
});

```

TagFactory.php

database > factories > TagFactory.php

```php
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */

use App\Tag;
use Faker\Generator as Faker;

$factory->define(Tag::class, function (Faker $faker) {
    return [
        "tag"=> $faker->realText(10,1),
    ];
});

```

UserFactory.php

database > factories > UserFactory.php

```php
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */
use App\User;
use Faker\Generator as Faker;
use Illuminate\Support\Str;

/*
|--------------------------------------------------------------------------
| Model Factories
|--------------------------------------------------------------------------
|
| This directory should contain each of the model factory definitions for
| your application. Factories provide a convenient way to generate new
| model instances for testing / seeding your application's database.
|
*/

$factory->define(User::class, function (Faker $faker) {
    return [
        'name' => $faker->name,
        'email' => $faker->unique()->safeEmail,
        'email_verified_at' => now(),
        'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', // password
        'remember_token' => Str::random(10),
    ];
});

```

Seeders

DatabaseSeeder.php

database > seeds > DatabaseSeeder.php

```php
<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        //Tags are required before posts
        $this->call(TagsTableSeeder::class);
        //Users are required before posts, comments and notepads
        $this->call(UsersTableSeeder::class);
        //Posts are required before comments
        $this->call(PostsTableSeeder::class);
        $this->call(CommentsTableSeeder::class);
        $this->call(NotepadsTableSeeder::class);
    }


}
```

CommentsTableSeeder.php

database > seeds > CommentsTableSeeder.php

```php
<?php

use Illuminate\Database\Seeder;
use App\Post;
class CommentsTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $posts = Post::get();
        $users = App\User::get();
        foreach($posts as $post )
        {
            $numberOfComments = rand(0,25);
            $comments = factory(App\Comment::class,$numberOfComments )->make()->each(function ($comment)
            {
                //For each comment give it a creator (a user)
                $user = App\User::inRandomOrder()->first();
                $comment->user()->associate($user);

            });
            //Add the comments to the post
            $post->comments()->saveMany($comments);

        }
    }
}
```

NotepadsTableSeeder.php

database > seeds > NotepadsTableSeeder.php

```php
<?php

use Illuminate\Database\Seeder;

class NotepadsTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $users = App\User::get();
        //Each user will have one notepad
        foreach($users as $user)
        {
            $notepad =  factory(App\Notepad::class)->make();
            $user->notepad()->save($notepad);
        }
    }
}
```

PostsTableSeeder.php

database > seeds > PostsTableSeeder.php

```php
<?php

use Illuminate\Database\Seeder;

class PostsTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //Create the posts
        factory(App\Post::class,100)->make()->each(function ($post)
        {
            //For each post give it a creator (a user)
            $user = App\User::inRandomOrder()->first();
            $user->posts()->save($post);

            //Add tags to this post
            $numberOfTags = rand(1,10); //Between 1 and 10 tags
            $tags = App\Tag::get()->take($numberOfTags);
            $post->tags()->saveMany($tags);
        });

    }
}

```

TagsTableSeeder.php

database > seeds > TagsTableSeeder.php

```php
<?php

use Illuminate\Database\Seeder;

class TagsTableSeeder extends Seeder
{



    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $tags = factory(App\Tag::class,50)->make();
        //Each tag must be unique. If it's not unique, we'll make a new one and try again.
        foreach($tags as $tag){
            $saved = false;
            while (!$saved)
            {
                try
                {
                    $tag->save();
                    $saved = true;
                }
                catch (\Illuminate\Database\QueryException $e) {
                    $tag = factory(App\Tag::class)->make();
                }
            }
        }
    }

}
```

UsersTableSeeder.php

database > seeds > UsersTableSeeder.php

```php
<?php

use Illuminate\Database\Seeder;

class UsersTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        factory(App\User::class,50)->create();
    }
}
```