با توجه به توضیحات ارائه شده جداول به صورت زیر طراحی شد:

جدول 'users' حاوی اطلاعات کاربران است.

```php
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('email')->unique();
    $table->string('username')->unique();
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->rememberToken();
    $table->string('token')->unique()->nullable();
    $table->timestamps();

    $table->unsignedInteger('coins')->default(0);
});
```

جدول 'orders' حاوی سفارشات کاربران است.

```php
Schema::create('orders', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger('user_id');
    $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
    $table->unsignedInteger('followers_count');
    $table->enum('status', ['ongoing', 'done'])->default('ongoing');
    $table->timestamps();
});
```

جدول 'followings' حاوی ارتباط فالو کردن کاربران است.

```php
Schema::create('followings', function (Blueprint $table) {
    $table->id();

    $table->unsignedBigInteger('user_id');
    $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');

    $table->unsignedBigInteger('follows_user_id');
    $table->foreign('follows_user_id')->references('id')->on('users')->onDelete('cascade');

    $table->unsignedBigInteger('order_id')->nullable();
    $table->foreign('order_id')->references('id')->on('orders')->onDelete('cascade');


    $table->timestamps();
});
```

لیست api ها: (در قسمت نکات و فرضیات به شماره route های زیر اشاره شده است.

```php
Route::post('/register', [AuthController::class, 'register'])->name('register'); ۱

Route::post('/login', [AuthController::class, 'login'])->name('login');۲


Route::group(['middleware' => ['auth:sanctum']], function (){
    Route::post('/logout', [AuthController::class, 'logout'])->name('logout'); ۳

    Route::post('/orders/create', [OrderController::class, 'store'])->name('create-order'); ۴

    Route::get('/orders/ongoing', [OrderController::class, 'getOngoingOrders'])->name('get-ongoing-orders'); ۵

    Route::post('/orders/follow/{orderToFollow}', [OrderController::class, 'followByOrder'])->name('follow-by-order'); ۶

    Route::post('/orders/buy-coins', [OrderController::class, 'buyCoins'])->name('buy-coins'); ۷

    Route::post('/following/follow/{userToFollow}', [FollowingController::class, 'follow'])->name('follow');۸

});
```

نکات و فرضیات:

- کاربر برای گرفتن توکن و فعالیت در شبکه اجتماعی باید ثبت نام کند یا وارد حساب کاربری خود بشود. (۱و۲)
- کاربر می‌تواند از حساب کاربری خود خارج شود (۳)
- در این پیاده سازی فرض شده که کاربر قبل از سفارش فالور باید میزان سکه مورد نیاز را داشته باشد، وگرنه امکان ثبت سفارش برای او وجود ندارد.
- پس از ثبت سفارش سکه ها به تناسب تعداد فالورهای درخواستی از حساب کاربر کسر می‌شود.
- کاربر با فالو کردن کاربرانی که سفارشات باز 'ongoing' دارند (هنوز تعداد افرادی که آن‌ها را فالو میکنند به میزان درخواستی شان نرسیده) می‌تواند ۲ سکه دریافت کند یا با خرید سکه به صورت فرضی (بدون پیاده سازی تراکنشات بانکی و پرداخت واقعی) سکه تهیه کند. (۶و۷و۸)
- کاربر می‌تواند لیست سفارشات کاربرانی که سفارش فالور دارند مشاهده کند، و با فالو کردن آنها سکه دریافت کند.(۵)
- کاربر می‌تواند هم براساس ای سفارش کاربر دیگری را فالو کند و هم بر اساس ای دی خود فرد (۶و ۸)
- اگر کاربر، کاربر دیگری که سفارش باز 'ongoing' ندارد (سفارش در حالت پایان یافته 'done' باشد) یا کلا سفارشی ندارد فالو کند. در جدول 'followings' ثبت می‌شود ولی سکه ای دریافت نمی کند. (فقط با فالو کردن کاربرانی که سفارش ثبت کرده اند، ۲ سکه دریافت می‌کنند و در صورت فالو کردن دوستان خود یا افراد دیگر سکه دریافت نمی کنند. (۶و۸)
- پس از این که تعداد فالورهای سفارش داده شده در سفارش کامل شد، سفارش به حالت 'done' در می‌آید و دیگر جزو لیست سفارشات باز به کاربران توصیه نمی‌شود، اما امکان فالو کردن بدون دریافت سکه امتیازی همچنان وجود خواهد داشت.

توسعه بیشتر:

- برای اضافه کردن ویژگی های بیشتر که ممکن است در آینده به آن‌ها نیاز پیدا کنیم. می‌توان از event ها و listener ها برای رخداد «فالو کردن» و «ثبت سفارش جدید» استفاده کرد تا هر گاه سفارش جدید ثبت شد بتوان اقدامات لازم را در سیستم برادکست کنیم، و با توجه به آن عمل مناسب را انجام دهیم.

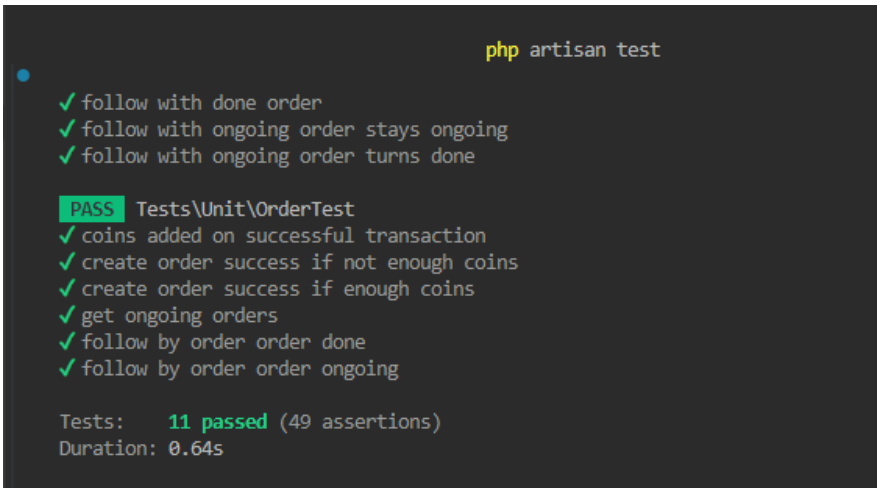
تست:

نوشتن تست‌های کارا و با پوشش مناسب می‌تواند در توسعه کار در آینده و دیباگ خطاهای احتمالی بسیار موثر باشد. به همین جهت نوشتن یونیت تیست یکی از توصیه‌های مهم است.

برای این پروژه چند تست نمونه برای تست کردن عملکرد های اصلی و سناریوهای مختلف سیستم نوشته شده است که در پوشه tests قرار گرفته اند و با دستور:

```
php artisan test
```

اجرا می شوند.



اجرا:

```
$ Composer install
```

Create .env file and add database and other configurations

$ php artisan migrate

$php artisan serve

کالکشن ریکوست‌ها در برنامه postman نیز با نام baham-test-project.postman_collection.json در پروژه موجود است.