Setting up Laravel Project

1. Git Clone this repository: git@github.com:Kachenas/tdd-laravel.git
2. Create a Database
   - Read the manual on how to create a database in setting up Laragon Project
3. Make a copy of .env, rename .env.example to .env
4. Update the .env to point to newly created database on part 1

   DB_CONNECTION=mysql
   DB_HOST=127.0.0.1
   DB_PORT=3306
   DB_DATABASE=<the name of your database>
   DB_USERNAME=root
   DB_PASSWORD=
5. Run composer install on your laragon terminal
   - This will install composer on your project folder
6. Run php artisan optimize:clear on your laragon terminal
7. Run php artisan serve on your terminal
8. Be sure to enable PHP Extensions
9. Find phpunit.xml and comment out DB_CONNECTION and DB Database
10. Close php artisan serve and run php artisan migrate
    - This will migrate the migrations file made for this repo.

Running Test

1. Run php artisan test
   - This will run the artisan test made for this project. Since the routes/api with group product is commented, this will cause the test to fail.
2. Comment out these line of code found on routes/api

```
3.  Route::group(
4.      ['prefix' => 'products'],
5.      function () {
6.          Route::get('index', 'ProductController@index');
7.          Route::post('create','ProductController@create');
8.          Route::post('show', 'ProductController@show');
9.          Route::put('update','ProductController@update');
10.         Route::delete('delete','ProductController@delete');
11.     }
12. );
```

3. After that, go to productController, and go to index function. Comment the line of code below 1st comment and uncomment code below the 2nd Comment.

```
//1st Comment
      //return response()->json(['message' => 'Index found!'], 200);

      //2nd Comment
      $products = Product::all();
      return response()->json($products);
```

4. Also comment the line of code below 1st comment and uncomment code below the 2nd Comment of the create function. And run php artisan test on your console.

```
//1st Comment
       //return response()->json(['message' => 'This is for create'], 200);

       //2nd Comment
       return Product::create($request->all());
```

- The create function will fail because no data is being inserted.

- Now go to test_endpoint_create and comment out 2nd comment and comment 1st comment.

```
//1st comment
       // $response = $this->post(self::PRODUCTS_URL."/create");
       // $response->assertStatus(200);

       //2nd comment
       $data = [
           'product_name' => $this->faker->randomElement(["Addidas", "Nike", "Reebok", "Pony"]),
           'product_price' => $this->faker->randomFloat(),
           'quantity' => $this->faker->randomDigit(),
           'product_status' => $this->faker->randomElement(["Active", "InActive"]),
       ];

       $response = $this->post(self::PRODUCTS_URL."/create", $data);
       $response->assertStatus(201);
```

- Hit save and run php artisan test.

5. Now go to Product Controller under show function. Comment the line of code below 1st comment and uncomment code below the 2nd Comment of the show function. And run php artisan test on your console.

```
//1st Comment
       //return response()->json(['message' => 'This is for show'], 200);

       //2nd Comment
       if ($product  = Product::find($request->id)) {
           return $product;
       }
       return response()->json(['message' => 'No Product found!'], 404);
```

- The show function will fail because no product is found.

```
//1st comment
       // $response = $this->post(self::PRODUCTS_URL."/show");
       // $response->assertStatus(200);

       //2nd comment
       $data = [
           'id' => $this->faker->randomDigit()
       ];

       $response = $this->post(self::PRODUCTS_URL."/show",$data);

       $response->assertStatus(404);
```

- Now go to test_endpoint_show_specific_product and comment out 2nd comment and comment 1st comment.

- Hit save and run php artisan test.

6. Now go to Product Controller under update function. Comment the line of code below 1st comment and uncomment code below the 2nd Comment of the update function. And run php artisan test on your console.

```
//1st Comment
       //return response()->json(['message' => 'This is for update'], 200);

       //2nd Comment
       $user = Product::findOrFail($request->id)->update($request->all());
       return response()->json(['message' => 'Product is Updated!'], 200);
```

- The update function will fail because no data is being created and updated.

- Now go to test_endpoint_update and comment out 2nd comment and comment 1st comment.

```
//1st comment
       // $response = $this->put(self::PRODUCTS_URL."/update");
       // $response->assertStatus(200);

       //2nd comment
       $product = Product::factory()->create();

       $updatedData = [
           'id' => $product->id,
           'product_price' => $this->faker->randomNumber(2),
           'quantity' => $this->faker->randomDigit(),
           'product_status' => $this->faker->randomElement(["Active", "InActive"])
       ];

       $response = $this->put(self::PRODUCTS_URL."/update", $updatedData);
       $response->assertStatus(200);
       $this->assertDatabaseHas('products', $updatedData);
```

- Hit save and run php artisan test.

7. Now go to Product Controller under delete function. Comment the line of code below 1st comment and uncomment code below the 2nd Comment of the delete function. And run php artisan test on your console.

```
//1st Comment
       //return response()->json(['message' => 'This is for destroy'], 200);

       //2nd Comment
       $user = Product::findOrFail($request->id)->delete();
       return response()->json(['message' => 'Product is deleted!'], 204);
```

- The delete function will fail because no data is being deleted.

- Now go to test_endpoint_delete and comment out 2nd comment and comment 1st comment.

```
//1st comment
       // $response = $this->delete(self::PRODUCTS_URL."/delete");
       // $response->assertStatus(200);

       //2nd comment
       $product = Product::factory()->create();

       $deleteData = [
           'id' => $product->id
       ];

       $response = $this->delete(self::PRODUCTS_URL."/delete", $deleteData);
       $response->assertStatus(204);
       $this->assertDatabaseMissing('products', ['id' => $product->id]);
```

- Hit save and run php artisan test.

8. Now under test_invalid_input_create. We purposely entered an invalid input, this will cause the test to fail because of the wrong data type and null data being pass. To pass this test, uncomment //'product_name' => $this->faker->randomElement(["Addidas", "Nike", "Reebok", "Pony"]), and //'quantity' => $this->faker->randomDigit(), then comment 'product_name' => $this->faker->randomFloat(), and 'quantity' => $this->faker->randomElement(["", null]),