2301871196

Matthew Brian

Computer Science Global Class

Web Programming

Before starting off with this project, we must first create a new Laravel project with the cmd command ‘Laravel new *projectName*’ in this case the project name is ‘uts’.

```
PS C:\Users\Matthew Munandar\onedrive\desktop\workspace> laravel new UTS

    Creating a "laravel/laravel" project at "./UTS"
    Installing laravel/laravel (v8.6.8)
  - Downloading laravel/laravel (v8.6.8)
      - Installing laravel/laravel (v8.6.8): Extracting archive
    Created project in C:\Users\Matthew Munandar\onedrive\desktop\workspace/UTS
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
    Loading composer repositories with package information
    Updating dependencies
    Lock file operations: 111 installs, 0 updates, 0 removals
  - Locking asm89/stack-cors (v2.0.3)
  - Locking brick/math (0.9.3)
  - Locking dflydev/dot-access-data (v3.0.1)
  - Locking doctrine/inflector (2.0.4)
```

Figure 1. New Laravel Project

Following this we then are required to edit the .env file inside the Laravel folder to help incorporate the database we are going to make in phpMyAdmin in this case the database name will be utsdatabase.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=utsdatabase
DB_USERNAME=root
DB_PASSWORD=
```

Figure 2.Editing env. File

After this we can then create the desired database in phpMyAdmin.

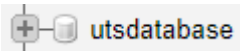

Figure 3. Database in phpMyAdmin

With all this done, we can then go forth with doing number one.

1. *Migration, Seeder and Data Insertion*

```
PS C:\Users\Matthew Munandar\onedrive\desktop\workspace\UTS> php artisan make:model Category -m -s
Created Migration: 2021_11_27_131230_create_categories_table
Seeder created successfully.
PS C:\Users\Matthew Munandar\onedrive\desktop\workspace\UTS> php artisan make:model Category -m -s
Model already exists!
PS C:\Users\Matthew Munandar\onedrive\desktop\workspace\UTS> php artisan make:model Book -m -s
Model created successfully.
Created Migration: 2021_11_27_131351_create_books_table
Seeder created successfully.
PS C:\Users\Matthew Munandar\onedrive\desktop\workspace\UTS> php artisan make:model Detail -m -s
Model created successfully.
Created Migration: 2021_11_27_131413_create_details_table
Seeder created successfully.
```

Figure 4. Model with Migration and Seeder

The following command will then be used to create each model starting off with the Category, followed by the book, then the details. These must be done in order due to how the relationship is between the tables. -m and -s will also be included to provide migration and seeder properties respectively.

After successfully creating the migration and seeders, we can then move on to create the tables. The snippets below will show case the tables made.

database > migrations > 2021_11_27_131230_create_categories_table.php > ...

```php
<?php

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

class CreateCategoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('category', 255);
        });
    }

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

Figure 5. Category Migration

database > migrations > 2021_11_27_131351_create_books_table.php > CreateBooksTable > up > #Fun

```php
<?php

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

class CreateBooksTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('books', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('category_id');
            $table->foreign('category_id')->references('id')->on('categories');
            $table->string('title', 255);
        });
    }

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

Figure 6. Book Migration

database > migrations > 2021_11_27_131413_create_details_table.php > ...

```php
<?php

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

class CreateDetailsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('details', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('book_id');
            $table->foreign('book_id')->references('id')->on('books');
            $table->string('author', 255);
            $table->string('publisher', 255);
            $table->integer('year');
            $table->longText('description');
        });
    }

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

Figure 7. Detail Migration

Following the graph provided by the question, we then link the tables with a foreign key with its respective references. After doing so we then edit the seeder files to proceed on with data insertion. The following snippets will then showcase the seeder files.

database > seeders > CategorySeeder.php > ...

```php

namespace Database\Seeders;

use App\Models\Category;
use Illuminate\Database\Seeder;

class CategorySeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $arr = [
            ['category' => 'Fiction'],
            ['category' => 'Science'],
            ['category' => 'Computer']
        ];

        Category::insert($arr);
    }
}
```

Figure 8. Category Seeder

database > seeders > BookSeeder.php > BookSeeder > run

```php

namespace Database\Seeders;

use App\Models\Book;
use Illuminate\Database\Seeder;

class BookSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $arr = [
            ['category_id' => 1, 'title' => 'Mocking Jay'],
            ['category_id' => 1, 'title' => 'Percy Jackson'],
            ['category_id' => 1, 'title' => 'Harry Potter'],
            ['category_id' => 1, 'title' => 'Star Wars'],
            ['category_id' => 1, 'title' => 'Dune'],
            ['category_id' => 1, 'title' => '1984'],
            ['category_id' => 2, 'title' => 'Artificial Intelligence a Modern Approach'],
            ['category_id' => 2, 'title' => 'Big Data Architecture'],
            ['category_id' => 2, 'title' => 'Operating Systems 101'],
            ['category_id' => 2, 'title' => 'Web Programming']
        ];

        Book::insert($arr);
    }
}
```

Figure 9. Book Seeder

```
database > seeders > DetailSeeder.php > ...
namespace Database\Seeders;

use App\Models\Detail;
use Illuminate\Database\Seeder;

class DetailSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $arr = [
            ['book_id' => 1, 'author' => 'Jennifer Lawrence', 'publisher' => 'Gramedia', 'year' => 2010, 'description' => 'The book continues the story
            ['book_id' => 2, 'author' => 'Rick Riordan', 'publisher' => 'Gramedia', 'year' => 2013, 'description' => 'Twelve-year-old Percy Jackson is
            ['book_id' => 3, 'author' => 'J.K Rowling', 'publisher' => 'Warner Bros.', 'year' => 2007, 'description' => 'Harry Potter is a film series
            ['book_id' => 4, 'author' => 'George Lucas', 'publisher' => 'Disney', 'year' => 2001, 'description' => 'Star Wars is an American epic space
            ['book_id' => 5, 'author' => 'Zendaya', 'publisher' => 'Frank H.', 'year' => 1965, 'description' => 'Paul Atreides, a brilliant and gifted
            ['book_id' => 6, 'author' => 'George Orwell', 'publisher' => 'Secker', 'year' => 1949, 'description' => 'Nineteen Eighty-Four is a dystopia
            ['book_id' => 7, 'author' => 'Gavin Suhkwir', 'publisher' => 'G & S', 'year' => 2021, 'description' => 'Artificial Intelligence: A Guide fo
            ['book_id' => 8, 'author' => 'Faiz Arsalan', 'publisher' => 'Faisal', 'year' => 2018, 'description' => 'Extremely large data sets that may
            ['book_id' => 9, 'author' => 'Kenny Gilbert', 'publisher' => 'KenG', 'year' => 2019, 'description' => 'The software that supports a compute
            ['book_id' => 10, 'author' => 'Amadeo Willem', 'publisher' => 'Luigi', 'year' => 2009, 'description' => 'Web programming refers to the writ
        ];

        Detail::insert($arr);
    }
}
```

Figure 10. Detail Seeder

```
database > seeders > DatabaseSeeder.php > ...
namespace Database\Seeders;

use App\Models\Category;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this -> call([
            CategorySeeder::class,
            BookSeeder::class,
            DetailSeeder::class
        ]);
    }
}
```

Figure 11. Database Seeder

This will show case all the data we would like to insert to each table. This is then followed by editing the DatabaseSeeder by calling the functions made in each Seeder. With the seeder, since 10 data rows are required, I have written 10 different types of books to use for the insertion with 6 belonging to the fiction category, 4 belonging to science, and I am leaving the computer category empty to show case what it will look like without data. We can then use the command done below to refresh the migration. This will insert all data into the tables.

```
PS C:\Users\Matthew Munandar\onedrive\desktop\workspace\uts> php artisan migrate:refresh --seed
Rolling back: 2021_11_27_131413_create_details_table
Rolled back:  2021_11_27_131413_create_details_table (7.65ms)
Rolling back: 2021_11_27_131351_create_books_table
Rolled back:  2021_11_27_131351_create_books_table (7.48ms)
Rolling back: 2021_11_27_131230_create_categories_table
Rolled back:  2021_11_27_131230_create_categories_table (7.31ms)
Rolled back:  2019_12_14_000001_create_personal_access_tokens_table (3.98ms)
Rolled back:  2019_08_19_000000_create_failed_jobs_table (4.82ms)
Rolling back: 2014_10_12_100000_create_password_resets_table
Rolled back:  2014_10_12_100000_create_password_resets_table (5.79ms)
Rolling back: 2014_10_12_000000_create_users_table
Rolled back:  2014_10_12_000000_create_users_table (4.93ms)
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (44.28ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (48.17ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (42.51ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (39.93ms)
Migrating: 2021_11_27_131230_create_categories_table
Migrated:  2021_11_27_131230_create_categories_table (25.59ms)
Migrating: 2021_11_27_131351_create_books_table
Migrated:  2021_11_27_131351_create_books_table (62.57ms)
Migrating: 2021_11_27_131413_create_details_table
Migrated:  2021_11_27_131413_create_details_table (59.96ms)
Seeding: Database\Seeders\CategorySeeder
Seeded:  Database\Seeders\CategorySeeder (11.72ms)
Seeding: Database\Seeders\BookSeeder
Seeded:  Database\Seeders\BookSeeder (7.79ms)
Seeding: Database\Seeders\DetailSeeder
Seeded:  Database\Seeders\DetailSeeder (2.76ms)
Database seeding completed successfully.
```

Figure 12. Data Insertion

2. *Model and Table Relations*

After successfully seeding the database, we can then move on with the table models. Preparing for the creation of thee table models, we are required to first understand the relationships between each table. For this I will be using the demonstration ERD below.

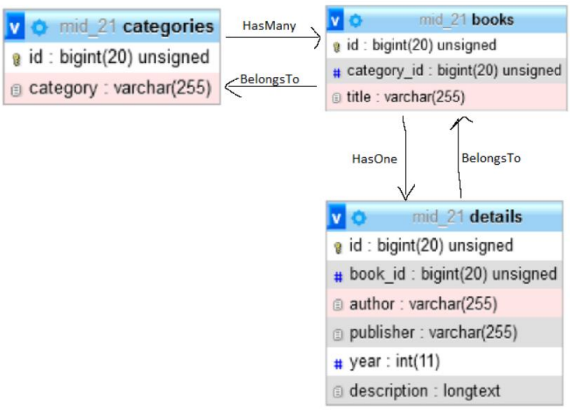

Figure 13. Table Relations

With this done, we can then reference this to make the models for each table. This step is important due to the sole reason that we need to link each table for us to use in the controller. This involves creating new functions for each, and defining the function based on the relationships towards the linked tables.

app > Models > Category.php > Category

```php
<?php

namespace App\Models;

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

class Category extends Model
{
    use HasFactory;

    public function book() {
        return $this -> hasMany(Book::class);
    }

}
```

Figure 14. Category Model

app > Models > Book.php > Book

```php
<?php

namespace App\Models;

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

class Book extends Model
{
    use HasFactory;

    public function category(){
        return $this -> belongsTo(Category::class);
    }

    public function detail(){
        return $this -> hasOne(Detail::class);
    }

}
```

Figure 15. Book Model

app > Models > Detail.php > ...

```php
<?php

namespace App\Models;

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

class Detail extends Model
{
    use HasFactory;

    public function book() {
        return $this -> belongsTo(Book::class);
    }

}
```

Figure 16. Detail Model

3. & 4. Application Controller and Routes for Access

With this done we can then proceed to move on with the PageController. We can then use the cmd command ‘php artisan make:controller PageController’ to make the controller and all we have to do left is create the functions we are going to use to display the pages.

app > Http > Controllers > PageController.php > PageController > displayHome

```php
<?php

namespace App\Http\Controllers;

use App\Models\Book;
use App\Models\Category;
use App\Models\Detail;
use Illuminate\Http\Request;

class PageController extends Controller
{
    public function displayHome(){
        $book = Book::all();

        $data = [
            'book' => $book
        ];

        return view('home', $data);
    }

    public function displayCategory($id, $name){

        $book = Book::where('category_id', $id) -> get();
        $category = Category::where('category', $name)->get();

        $data = [
            'book' => $book
        ];

        $catName = [
            'category' => $category
        ];

        return view('category', $data, $catName);
    }

    public function displayContact(){
        return view('contact');
    }

    public function displayDetail($title){

        $book = Book::where('id', $title) -> first();

        $data = [
            'book' => $book
        ];

        return view('detail', $data);
    }

}
```

Figure 17. PageController

With the PageController shown and after implementing the model Book, I will explain what each function does and why I have chosen that specific way of programming.

For the displayHome() function, I have opted to use the 'Book' model to call all books in the database, this allows me to display the list of books on the homepage whilst storing them in an array for ease when passing parameters.

The displayCategory() function may seem tricky but is quite simple in the nature that I will pass the name of the category and the category id. This will allow me to filter out the collection of books based on the category. The Category model is used to display the title of the page based on what category is being displayed.

The displayContact() function will display the contact view.blade.php page and displayDetail() will display the details of a book when the book is clicked based on the book's id which we will pass through the route.

On that note the picture below will show case the routing I have made which includes the parameters needed to be passed to the displayCategory() and displayDetail() function.

routes > web.php

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PageController;

Route::get('/', [PageController::class, 'displayHome']);

Route::get('/category/{id}/{name}', [PageController::class, 'displayCategory']);

Route::get('/contact', [PageController::class, 'displayContact']);

Route::get('/detail/{title}', [PageController::class, 'displayDetail']);
```

Figure 18. Routing

As you can see the PageController is being implemented in this routing page and due to that the views are all controlled by the route. This allows us to edit everything through the controller.

5. Views

With all being said I then finish off the project by completing the views for the website by passing the parameters through the controller.

resources > views > layout.blade.php > html > body > div.container.mt-5 > div.row. > div.p-3.col-2 > a.dropdown-item.text-primary

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>2301871196</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="{{ asset('bootstrap/css/bootstrap.min.css') }}" rel="stylesheet">
</head>
<body>

  <div class="container-fluid p-4 bg-primary text-white text-center">
    <h1>HAPPY BOOK STORE</h1>
  </div>
  <div class="container">
    <ul class="nav  justify-content-center">
      <li class="nav-item">
        <a class="nav-link" href="{{ url('/') }}">Home</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link  dropdown-toggle" data-bs-toggle="dropdown" href="#">Category</a>
        <ul class="dropdown-menu">
          <li><a class="dropdown-item" href="{{ url('/category/1/Fiction') }}">Fiction</a></li>
          <li><a class="dropdown-item" href="{{ url('/category/2/Science') }}">Science</a></li>
          <li><a class="dropdown-item" href="{{ url('/category/3/Computer') }}">Computer</a></li>
        </ul>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="{{ url('/contact') }}">Contact</a>
      </li>
    </ul>
  </div>
  <div class="container mt-5">
    <div class="row ">

      <div class="col-10 p-3">
        @yield('content')
      </div>

      <div class = "p-3 col-2">
        <div class="container-fluid p-1 bg-warning">
          <h4>Category</h4>
        </div>
        <br>
        <a class="dropdown-item text-primary" href="{{ url('/category/1/Fiction') }}">Fiction</a>
        <a class="dropdown-item text-primary" href="{{ url('/category/2/Science') }}">Science</a>
        <a class="dropdown-item text-primary" href="{{ url('/category/3/Computer') }}">Computer</a>
      </div>

  </div>
  <div class="container p-1 mt-5 bg-primary text-white text-center">
    <small>&copy Happy Book Store {{ date("Y") }}</small>
  </div>
  <script src="{{ asset('bootstrap/js/bootstrap.bundle.min.js') }}"></script>

</body>
</html>
```

Figure 19. View Layout

resources > views > home.blade.php > table.table

```
@extends('layout')
@section('content')
  <div class="container-fluid p-1 bg-warning">
    <h4>Book List</h4>
  </div>
  <br>
  <table class="table">
    <thead>
      <tr>
        <th>Title</th>
        <th>Author</th>
      </tr>
    </thead>
    <tbody>
      @foreach ($book as $item)
      <tr>
        <td>
          <a href="/detail/{{$item -> id}}" class="text-decoration-none text-black">
            {{$item -> title}}
          </a>
        </td>
        <td>
          {{$item -> detail -> author}}
        </td>
      </tr>
      @endforeach
    </tbody>
  </table>
@endsection
```

Figure 20. View Home

```
resources > views > category.blade.php > ...
@extends('layout')
@section('content')
  <div class="container-fluid p-1 bg-warning">
    <h4>
      @foreach ($category as $item)
      {{$item->category}}
      @endforeach
    </h4>
  </div>
  <table class="table">
    <thead>
      <tr>
        <th>Title</th>
        <th>Author</th>
      </tr>
    </thead>
    <tbody>
      @forelse ($book as $item)

        <tr>
          <td>
            <a href="/detail/{{$item -> id}}" class="text-decoration-none text-black">
              {{$item -> title}}
            </a>
          </td>
          <td>
            {{$item -> detail -> author}}
          </td>
        </tr>

      @empty
      <tr>
       <td colspan = 2>
       <p class="container-fluid p-1 bg-warning">No Data...</p>
       </td>
      </tr>
      @endforelse
    </tbody>
  </table>
@endsection
```

Figure 21. View Category

```
resources > views > contact.blade.php > ...
@extends('layout')
@section('content')
<div class="container-fluid p-1 bg-warning">
    <h4>Contact</h4>
</div>

    <br>
    <h2>Store Address : </h2>
    <p>Jalan Pembangunan Baru Raya,</p>
    <p>Kompleks Pertokoan Emrald Blok III/12</p>
    <p>Bintaro, Tangerang Selatan</p>
    <p>Indonesia</p>

    <h4>Open Daily : </h4>
    <p>08.00 - 20.00</p>

    <h4>Contact:</h4>
    <p>Phone : 021-08899776655</p>
    <p>Email : happybookstore@happy.com</p>

@endsection
```

Figure 22. View Contact

```
resources > views > detail.blade.php > ...
@extends('layout')
@section('content')
  <div class="container-fluid p-1 bg-warning">
    <h4>Book Detail</h4>
  </div>
  <br>
  <p>Title : {{$book -> title}}</p>
  <p>Author : {{$book -> detail -> author}}</p>
  <p>Publisher : {{$book -> detail -> publisher}}</p>
  <p>Year : {{$book -> detail -> year}}</p>
  <p>Description : </p>
  <p>{{$book -> detail -> description}}</p>
@endsection
```

Figure 23. View Detail

With all this done we have come down to 5 view pages, the main layout of each page being the layout.blade, the home page where all books are displayed being the home.blade file, contact page in the contact.blade file, and due to our format of our displayCategory() and displayDetail() functions located in our PageController we are able to create only one view file for each category and book detail by passing the model collection as individual items.

With all this done we can then type in the cmd the command to host a local server with the command 'php artisan serve'. This will allow us to view the website we have created.

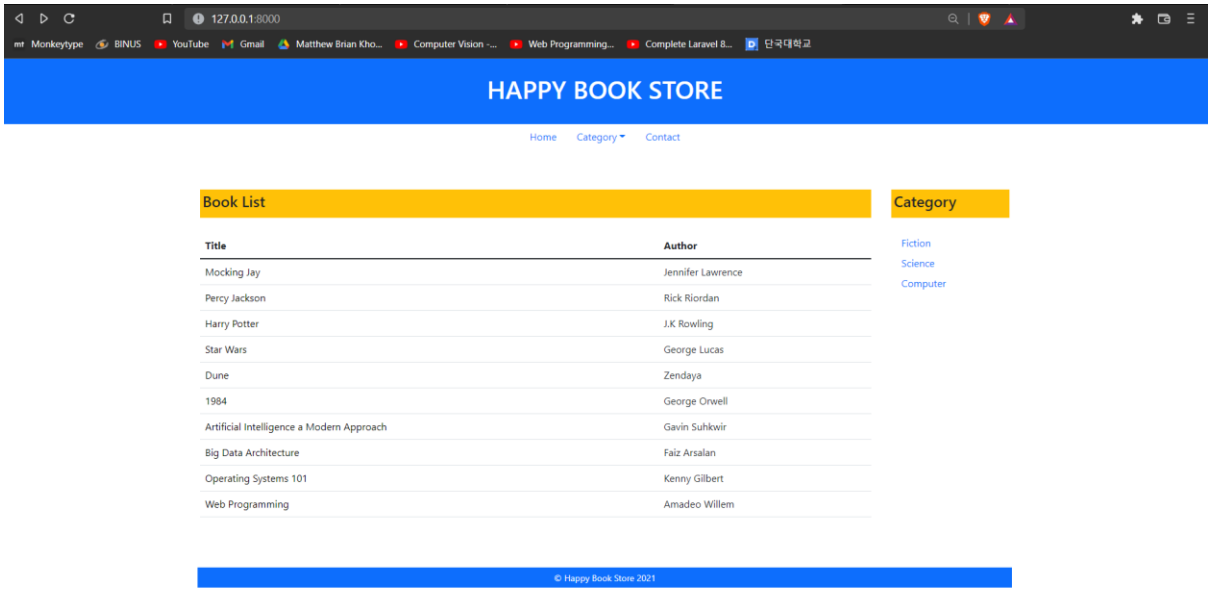

Figure 24. Home Page

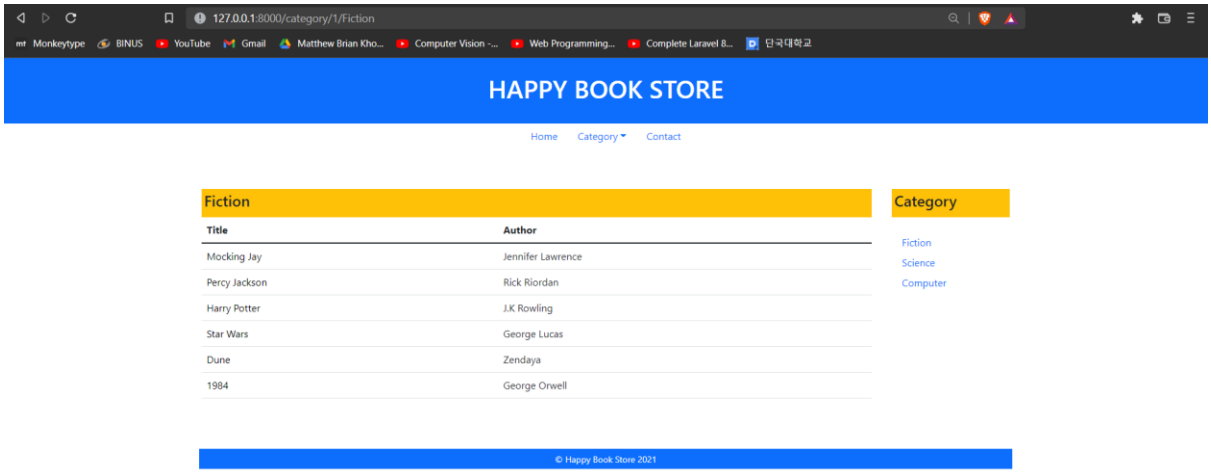

Figure 25. Fiction Page

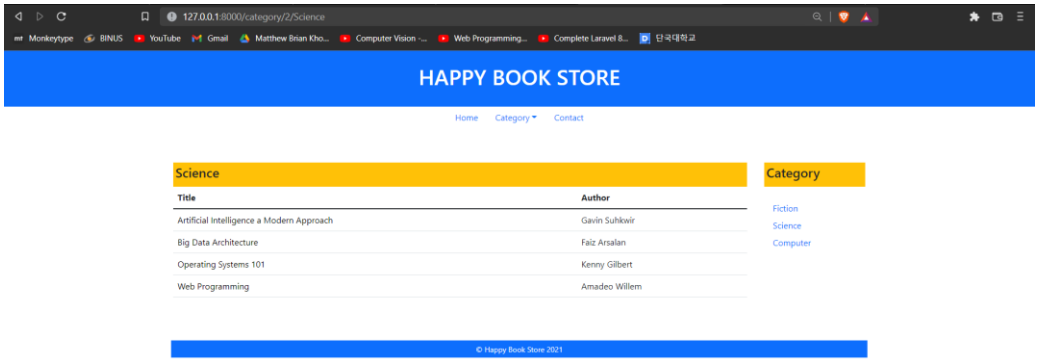

Figure 26. Science Page

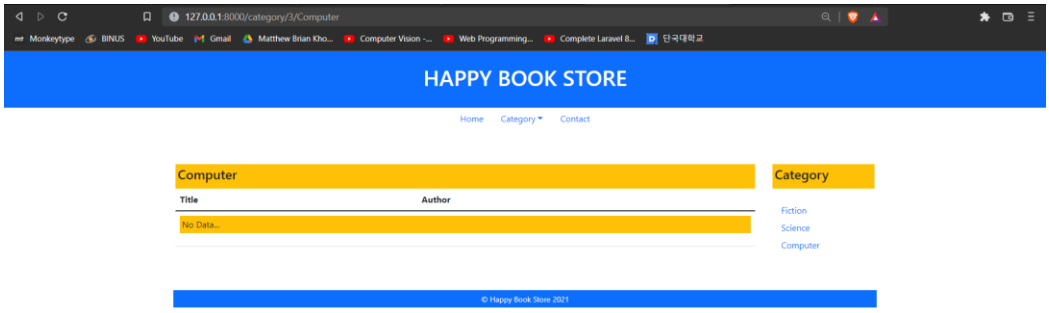

Figure 27. Computer Page

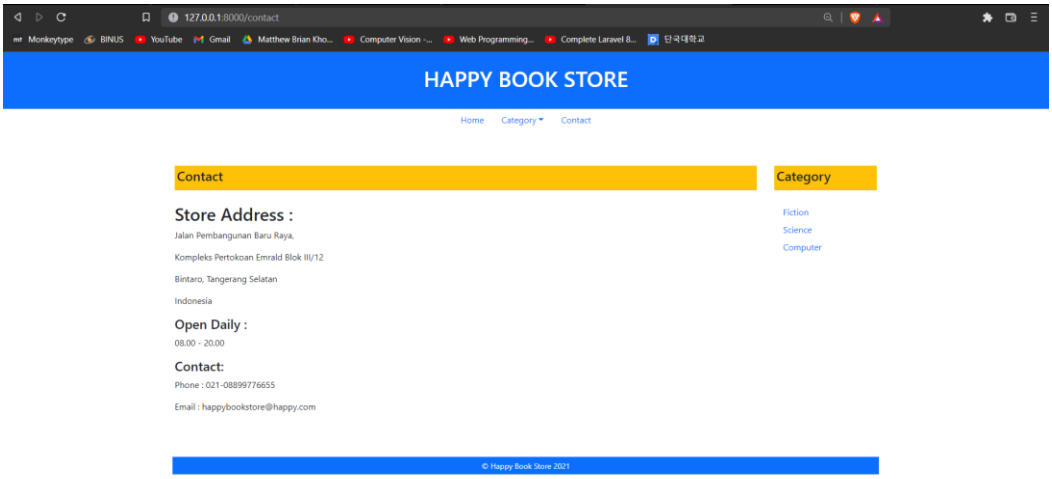

Figure 28. Contact Page