```
resources > views > Components > 🐘 layout.blade.php
  1    <!DOCTYPE html>
  2  ∨ <html lang="en">
  3  ∨ <head>
  4        <meta charset="UTF-8">
  5        <meta name="viewport" content="width=device-width, initial-scale=1.0">
  6        <title>@yield(section: 'title') - Scrolling Blog - Modern Design</title>
  7        @yield(section: 'styles')
  8    </head>
  9  ∨ <body>
 10
 11        @yield(section: 'content')
 12
 13  ∨     <footer class="footer">
 14            <p>&copy; 2024 JDM. All rights reserved.</p>
 15        </footer>
 16        @yield(section: 'scripts')
 17    </body>
 18    </html>
 19
```
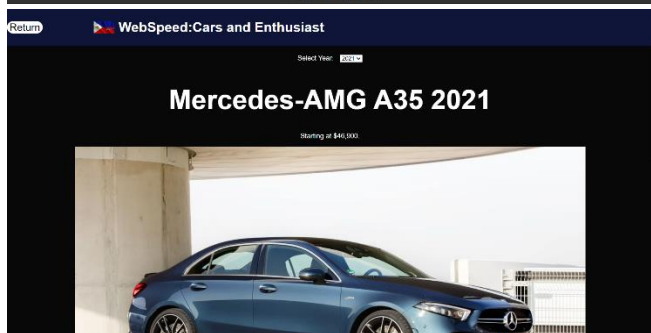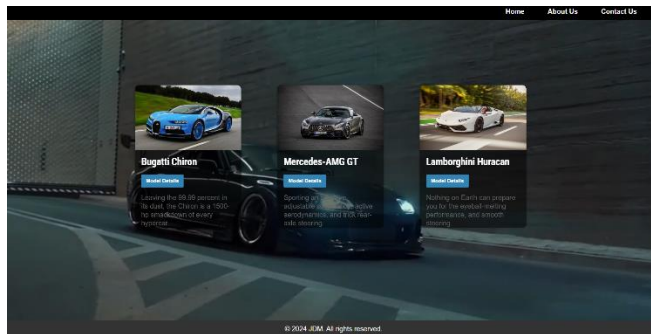
- This Blade layout file serves as the foundation for all the pages that extend it. using a layout simplifies maintaining consistent design and structure across multiple views.
- **Purpose of the Layout File -** The layout file is a reusable template that contains the common HTML structure shared across multiple pages. creating a base layout, you avoid writing the same HTML code repeatedly in each page.

1. **Extending the Layout -** The @extends directive is used to tell Laravel that this view file (page1.blade.php) will inherit the base structure defined in the layout.blade.php file.
2. **Inserting the Title -** This section inserts the page-specific title ("Page 1") into the @yield('title') placeholder in the layout.
3. **Adding Custom Styles -** The @section('styles') block is used to inject custom CSS files into the layout's @yield('styles') directive.
4. **Defining Page-Specific Content -** The @section('content') block is where the main content for this view is defined. This content will be injected into the layout's @yield('content') placeholder.
5. **Including JavaScript Files -** The @section('scripts') block is used to insert any JavaScript files or inline scripts specific to this view.

```php
routes >  web.php
  1   <?php
  2
  3   use Illuminate\Support\Facades\Route;
  4   use Illuminate\Support\Facades\Validator;
  5
  6
  7   Route::get(uri: '/', action: function (): Factory|View {
  8       return view(view: 'blog');
  9   })->name(name: 'blog');
 10
 11   Route::get(uri: '/about', action: function (): Factory|View {
 12       return view(view: 'about');
 13   })->name(name: 'about');
 14
 15   Route::get(uri: '/contact', action: function (): Factory|View {
 16       return view(view: 'contact');
 17   })->name(name: 'contact');
 18
 19   Route::get(uri: '/post1', action: function (): Factory|View {
 20       return view(view: 'post1');
 21   })->name(name: 'post1');
 22
 23   Route::get(uri: '/post2', action: function (): Factory|View {
 24       return view(view: 'post2');
 25   })->name(name: 'post2');
 26
 27   Route::get(uri: '/post3', action: function (): Factory|View {
 28       return view(view: 'post3');
 29   })->name(name: 'post3');
 30
```

- Each route is defined using **Route::get()** method in which it specifies the URL that corresponds to a get request.
- The **Route::get()** accepts two arguments, the URL path (ex. \name) or the closure function using view() function. Each route also has a name defined using the ->name() method, which assigns a name to the route.
- By naming routes (ex. ->name('post1')) we can easily reference it to other view files by using it like this (Model Details) that we used on the landing page of the views.

CHALLENGES:

- The challenges that I encountered is when I tried to reference different css files to different view files. I first tried to compile all the css into one single file so that I can put it in the layout file and just extend it at once to the different views, but it got complicated once the line of codes started to get a lot and that's when I tried using @section and just referenced the specific style for each view while extending the layout so that only the content is present in the file.

_____

- @yields are used in layouts so that we can define a section where the child view can fill or put its content using @section
- {{$slot}} is used when we want to pass components in parent views in its specific content. (ex. If we want to pass a message we use {{$slot}} and when we invoke the component the message will be shown there instead of us hard coding it in the specific line.