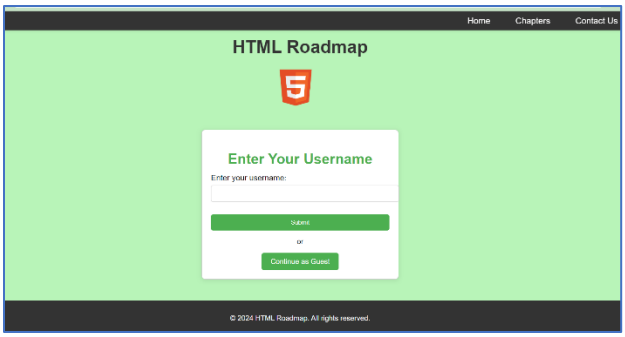
CACERES JELLA MARIE B.

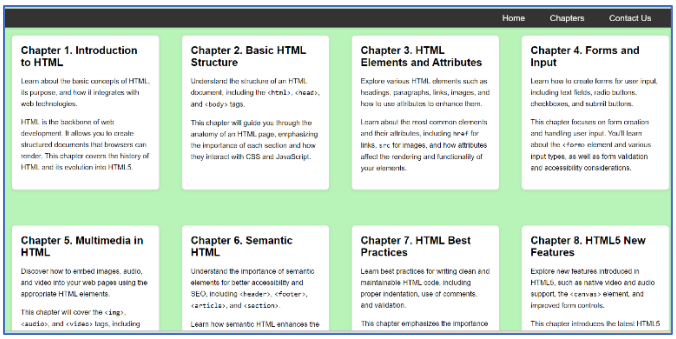
BSIT 3-C

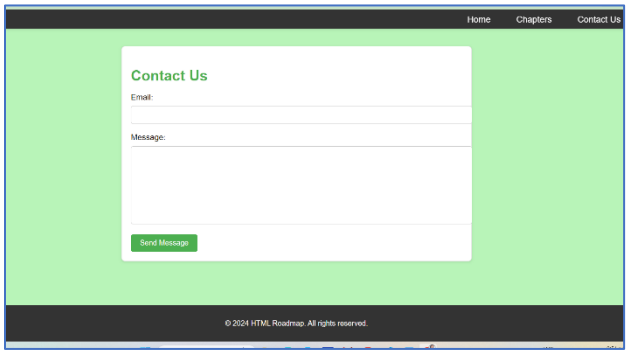
Web Development

Website Documentation

Web Pages:







Purpose of the Layout File and How It is Used:

The layout file (Layout.blade.php) serves as a reusable template for different views in the Laravel application. Instead of repeating common HTML elements (like the <head>, <header>, <footer>, and styles) in each view file, the layout centralizes these components.

Main Responsibilities of the Layout File:

1. **Defines the common structure** for the web pages, such as the navigation bar, header, footer, and other elements that will be present on all or most pages.
2. **Placeholders for dynamic content:** It defines where the content of specific pages (e.g., home, contact, chapters) will be injected using @yield or {{$slot}} directives.
3. **Styling and Asset Linking:** It links stylesheets and fonts globally, ensuring that all views have access to them.

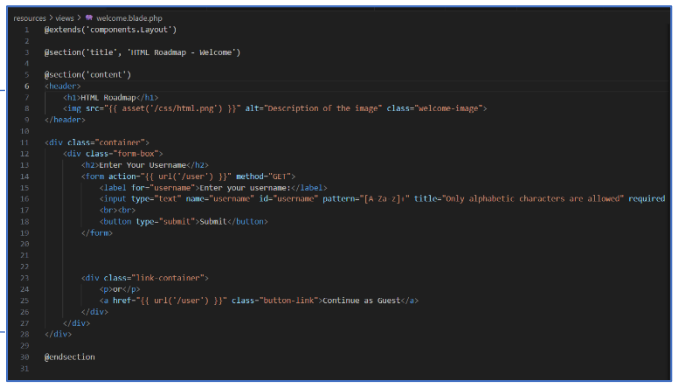
For instance:

- The navigation bar (<nav>) with links to the home, chapters, and contact pages is defined once in the layout file, and this structure will be shared across all views.
- The @yield('content') directive in the layout file is used as a placeholder where individual view content will be inserted.

How each view extends the layout and inserts specific content:

Each view (welcome.blade.php, chapters.blade.php, contact.blade.php) **extends the layout file** and **injects page-specific content** into it. This is achieved using the @extends ('components. Layout') directive, which means the view will inherit everything from the layout file.

Example:



<! -- Specific content for the Welcome page -->

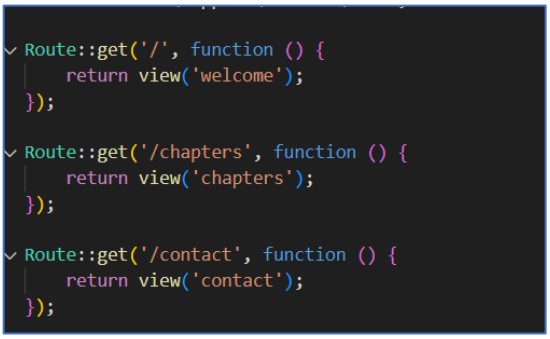
The @section('content') defines what content will be placed in the layouts @yield('content') section. Each view file can specify what content should be displayed for its respective page, while reusing the layout for the consistent structure.

The Routing Setup and How It Serves the Views:

Laravel routes are defined in the web.php file and control which view is shown for each URL.

The Route::get() method is used to define routes that respond to GET requests.
- **'/'** serves the welcome view.
- **'/chapters'** serves the chapters view.
- **'/contact'** serves the contact view.

- Each of these routes directly returns the associated view, which extends the layout file and fills in its specific content.



Challenges Faced and Resolutions

Positioning of Layout Elements:

- **Challenge:** Ensuring that the navbar and footer remain in place while allowing page-specific content to vary.
- **Resolution:** Flexbox was used in the layout's CSS (display: flex, flex-direction: column) to keep the header and footer fixed; while making sure the content section expands and adapts to the available space.

Handling Global Styles Across Multiple Pages:

- **Challenge:** Linking the right stylesheets and ensuring all pages inherit the styles consistently**.**
- **Resolution:** By including the CSS links in the layout file, all views automatically get the necessary styling without requiring individual links in each view.

Maintaining Consistent Navigation Across Pages:

- **Challenge:** Ensuring all views have the same navigation bar and footer without code duplication.
- **Resolution:** Using a layout file ensured that the navigation structure was shared across all pages, avoiding redundancy.

Difference Between {{$slot}} and @yield:
Both {{$slot}} and @yield are used to inject dynamic content in Blade templates, but they have different purposes:

1. **@yield:**

```
@yield('content')
```

- **Purpose**: It defines a section in a layout where content from child views can be injected.
- **How it Works**: It is used in the parent (layout) file to define a placeholder for content that will be provided by views that extend the layout. Each child view uses @section to define what goes into that specific @yield.

2. **{{$slot}}:**

```
<x-button>
    Click Me!
</x-button>
```

```
<button>{{$slot}}</button>
```

- **Purpose**: Primarily used in Blade components to allow dynamic content to be passed into a component.
- **How it Works**: It's commonly used in reusable components. A component can define one or more slot areas where dynamic content will be inserted.