



## Technical report - Practical training

Networking (University of Dar es Salaam)



Scan to open on Studocu

**MBEYA UNIVERSITY OF SCIENCE AND TECHNOLOGY**



**FIELD PRACTICAL TRAINING TECHNICAL REPORT.**

**COURSE NAME:** BACHELOR OF ENGINEERING IN DATA SCIENCE

**INDUSTRY/FIRM:** BIGBEE TECHNOLOGY LIMITED

**DEPARTMENT:** COMPUTER SCIENCE AND ENGINEERING.

**LEVEL** UQF8 FIRST YEAR

**NAME:** AMANI ELIA MACHANGE

**REGISTRATION NUMBER:** 22100533590034

**ACADEMIC YEAR:** 2022/2023.

**DURATION OF IPT:** 10<sup>th</sup> JULY 2023 TO 15<sup>th</sup> SEPTEMBER 2023.

## **PREFACE**

Industrial/field practical training IPT is an important crucial component of my data science program. It provides me with a unique opportunity to adopt theory with real experience in the real world.

The main objectives of field practical training skill development IPT helps me to acquire practical skills relevant to the field of data science. This means I will learn how to apply what I have studied in the real world setting.

Enhancing Abilities during this training, I am to develop my abilities in planning, organizing and effective communication. These skills are important for success in data science projects.

By the end of this training , my goal is to become an innovative, responsible, and capable data scientist.

## **ABSTRACT**

This report mostly explains

The Newsletter System which offers a comprehensive features made to meet the diverse needs of users across various sectors. It has a user-friendly interface that simplifies the newsletter creation process, allowing users to make visually informative content effortlessly. From customizable templates to rich multimedia integration, the system offers flexibility in design and content presentation.

Maintaining a consistent and organized newsletter schedule is crucial for the user. The Newsletter System includes a scheduling and automation feature, which the enabling users to plan their communication campaigns with precision. Furthermore, it supports multi-channel distribution, ensuring that newsletters reach audiences through preferred channels, email, social media.

Security and data privacy is a big concern in today's technological world. Our Newsletter System implements security measures to safeguard user data and protect against potential threats, ensuring compliance with data protection regulations.

In summary, the Newsletter System is a powerful solution designed to change the way individuals, businesses, and organizations communicate with their clients. By providing a user-friendly interface, automation, analytics, and strong security measures, this system offers a environment for enhancing engagement and achieving communication goals. System empowers users to connect with their audiences more effectively and efficiently.

### **ACKNOWLEDGEMENTS**

I would like to thank before the Almighty God for being with me during field practical training until now writing this report. Also, I would like to give much thanks to all, whose invaluable suggestions and contributions helped me so much when I was preparing this report.

For sure it is not possible to mention all, but there are some who were powerful contributors and played a great role to the completion of this report and with respect to their contributions I must mention few of them.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for

Educating and preparing me for my future.

I am grateful to the management of BIGBEE TECHNOLOGY LIMITED for giving us the opportunity to have an industrial practical training as a part of our curriculum.

I would be indebted if I did not express my gratitude to my industrial supervisor Vincent Simon who worked on us, He relentlessly guiding us throughout the field as He contributed a lot to develop our knowledge and technical skills.

Finally, my thanks go to all the people who have supported me to complete my IPT work directly or indirectly.

### **DECLARATION**

I Amani Elia Machange declare that this Technical Report submitted by me to MBEYA UNIVERSITY OF SCIENCE AND TECHNOLOGY, as fulfillment of the requirements for my studies for my Bachelor's degree in Data science Engineering is a record of the Field Practical Training work carried out by me. I further declare that the work reported in this report has never and will never be submitted, either in part or in full, for the same purpose in this University or any other university.

## TABLE OF CONTENT

<b>PREFACE.....</b>	<b>2</b>
<b>ABSTRACT.....</b>	<b>3</b>
<b>ACKNOWLEDGEMENTS. ....</b>	<b>4</b>
<b>DECLARATION.....</b>	<b>5</b>
<b>PART 1: GENERAL REPORT.....</b>	<b>8</b>
<b>HISTORICAL BACKGROUND OF BIGBEE TECHNOLOGY LIMITED .....</b>	<b>8</b>
<b>1.1: INTRODUCTION. ....</b>	<b>8</b>
<b>1.1.2: MISSION.....</b>	<b>8</b>
<b>1.1.3: VISION.....</b>	<b>8</b>
<b>1.1.4: SERVICES/ ACTIVITIES DONE BY THE FIRM.....</b>	<b>8</b>
<b>1.1.5: ADDRESS.....</b>	<b>9</b>
<b>1.2: ADMINISTRATIVE STRUCTURE.....</b>	<b>10</b>
<b>PART 2: TECHNICAL PART.....</b>	<b>11</b>
<b>ACTIVITIES PERFORMED.....</b>	<b>11</b>
<b>2.1: THE NEWSLETTER SYSTEM .....</b>	<b>11</b>
<b>2.1.1: INSTALLING LARAVEL.....</b>	<b>11</b>
<b>2.1.2: CONFIGURING THE DATABASE.....</b>	<b>12</b>
<b>2.1.3: CREATING AUTHENTICATION SYSTEM.....</b>	<b>13</b>
<b>2.1.4: SETTING UP THE ENVIRONMENT VARIABLE.....</b>	<b>15</b>
<b>2.2.1: DATABASE SCHEMA.....</b>	<b>16</b>
<b>2.2.2: CREATING THE NEWSLETTER MODEL.....</b>	<b>17</b>
<b>2.2.3: CREATING THE VIEWS FILES AND BLADES.....</b>	<b>18</b>
<b>2.3: CRUD OPERATION.....</b>	<b>19</b>
<b>2.3.1: CREATE OPERATION.....</b>	<b>19</b>
<b>2.3.2: READ OPERATION.....</b>	<b>21</b>
<b>2.3.3: UPDATE OPERATION.....</b>	<b>23</b>
<b>2.3.4: DELETE OPERATION.....</b>	<b>25</b>
<b>2.3.5: ROUTES.....</b>	<b>27</b>
<b>CONCLUSION ON NEWSLETTER.....</b>	<b>28</b>
<b>CHALLENGES FACED DURING IPT.....</b>	<b>29</b>
<b>RECOMMENDATIONS.....</b>	<b>29</b>
<b>REFERENCE.....</b>	<b>30</b>

## **TABLE OF FIGURES**

<b>Figure 1:</b> Laravel installation.....	12
<b>Figure 2:</b> Database configuration.....	13
<b>Figure 3:</b> Authentication file.....	14
<b>Figure 4:</b> Environment variable via mail.....	15
<b>Figure 5:</b> Database schema.....	16
<b>Figure 6:</b> Newsletter models .....	17
<b>Figure 7:</b> Create operation source codes.....	20
<b>Figures 8:</b> Output in create operation.....	20
<b>Figure 9:</b> Read operation source codes.....	22
<b>Figure 10:</b> Output in read operation .....	22
<b>Figure 11:</b> Update operation source code.....	24
<b>Figure 12:</b> Output in update operation .....	24
<b>Figure 13:</b> Delete operation source codes .....	26
<b>Figure 14:</b> Output in delete operation.....	26
<b>Figure 15:</b> Routes source codes.....	28



## **PART 1: GENERAL REPORT.**

### **HISTORICAL BACKGROUND OF BIGBEE TECHNOLOGY LIMITED**

#### **1.1: INTRODUCTION**

BIGBEE TECHNOLOGY LIMITED was established in 2014 as per NATIONAL CONSTITUTION ACT OF 2002 FOR COMPANIES WHICH ARE LIMITED AND UNLIMITED.

One of the functions of BIGBEE TECHNOLOGY LIMITED is “To provide And help clients make distinctive and substantial improvements in their business.”

BIGBEE TECHNOLOGY LIMITED has the following Mission and Vision

##### **1.1.2: MISSION:**

To help our clients make distinctive and substantial improvements in their business  
Through advanced information technology. For us, there is nothing to be happier  
About than satisfying our clients.

##### **1.1.3: VISION:**

To provide systems design and implementation services, to provide advanced  
IT consulting services, provide network design and installing services across all  
places in Tanzania and empowering Tanzania on technological advancement and  
empowering youth on learning different technological aspects.

##### **1.1.4: SERVICES/ ACTIVITIES DONE BY THE FIRM**

Bigbee technology limited offers software services including large scale systems and database design, web and mobile application development as well as data analytics and visualizations, it also provides networking planning, installation and security solutions and gives management services and custom solutions for automated infrastructure. Telephony technology from global technology leaders including but not limited to cisco, Huawei, Panasonic and Avaya they are able to deliver and support robust telephony systems tailor to their client's specific needs.

### **1.1.5 ADDRESS**

**Physical Address:**

Tanzania Tower 3<sup>rd</sup> Floor 3C, Sam Nujoma Road Dar es Salaam

**Postal Address:**

BIGBEE TECHNOLOGY LIMITED,

P.O.BOX 62910,

DAR ES SALAAM.

**Telephone:** +255 759 004 747

**Tel:** 0222221344

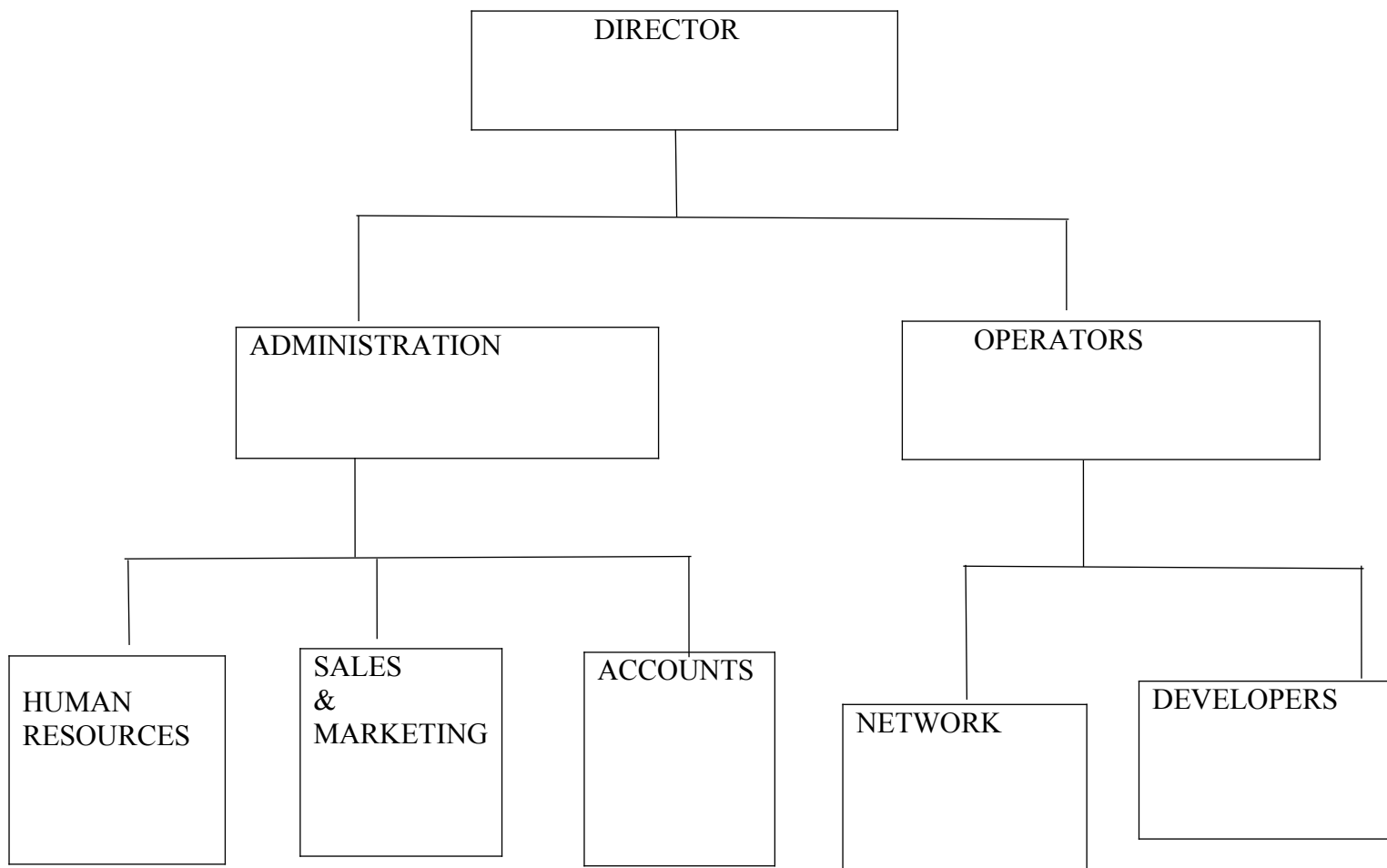
**Email:** [info@bigbee.io](mailto:info@bigbee.io)

**Web page:** bigbee.io

**Website:** www.bigbee.co.tz

## 1.2: ADMINISTRATIVE STRUCTURE

The BIGBEE TECHNOLOGY LIMITED is lead by director who is as the top and also the CEO of the company, Administration is lead by one person who is also the one concerned with human resources and accounts. Then at the operators side there is network and developers where in the developers side there is head of development and devops and in network is lead by head of Netsec and contains network engineers.



## PART 2

### 2: TECHNICAL PART

#### ACTIVITIES PERFORMED

##### 2.1: THE NEWSLETTER SYSTEM

Newsletter is a powerful tool for communication and marketing in the modern world. They allow businesses, organizations, and individuals to reach their audience directly through email. Laravel a php framework, makes an excellent platform for building a nice newsletter system. In the comprehensive guide, we will go through the process of creating a newsletter system in Laravel, covering everything from setting up the project to sending personalized newsletters and managing subscribers.

##### 2.1.1: INSTALLING LARAVEL

You can be able to install Laravel using Composer, a PHP dependency management tool. Open your terminal and run the following command.

**composer create-project Laravel/Laravel (project\_name )** this command helps to install all the files provided by Laravel which can be used to create any project of which you want to do with Laravel this command is executed in the gitbash which is a command line tool which the Laravel project is created and then it is opened in the visual code studio (vs code).

```
DELL USER@DESKTOP-1IFEU7Q MINGW64 ~/Desktop/project_44
composer create-project laravel/laravel project44
Creating a "laravel/laravel" project at "/project44"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v10.2.5)
- Installing laravel/laravel (v10.2.5): Extracting archive
Created project in C:\Users\DELL USER\Desktop\project_44\project44
@php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 109 installs, 0 updates, 0 removals
- Locking brick/math (0.11.0)
- Locking dflydev/dot-access-data (v3.0.2)
- Locking doctrine/inferno (2.0.8)
- Locking doctrine/lexer (3.0.0)
- Locking dragonmantank/cron-expression (v3.3.3)
- Locking egulias/email-validator (4.0.1)
- Locking fakerphp/faker (v1.23.0)
- Locking filp/whoops (2.15.3)
- Locking fruitcake/php-cors (v1.2.0)
- Locking graham-campbell/result-type (v1.1.1)
- Locking guzzlehttp/guzzle (7.7.0)
- Locking guzzlehttp/promises (2.0.1)
- Locking guzzlehttp/psr7 (2.6.0)
- Locking guzzlehttp/uri-template (v1.0.1)
- Locking hamcrest/hamcrest-php (v2.0.1)
- Locking laravel/framework (v10.18.0)
- Locking laravel/pint (v1.10.6)
- Locking laravel/prompts (v0.1.4)
- Locking laravel/sail (v1.23.2)
- Locking laravel/sanctum (v3.2.5)
- Locking laravel/serializable-closure (v1.3.1)
- Locking laravel/tinker (v2.8.1)
- Locking league/commonmark (2.4.0)
- Locking league/config (v1.2.0)
- Locking league/flysystem (3.15.1)
- Locking league/flysystem-local (3.15.0)
- Locking league/mime-type-detection (1.13.0)
- Locking mockery/mockery (1.6.6)
- Locking monolog/monolog (3.4.0)
- Locking myclabs/deep-copy (1.11.1)
- Locking nesbot/carbon (2.68.1)
- Locking nette/schema (v1.2.4)
- Locking nette/utils (v4.0.1)
- Locking nikic/php-parser (v4.17.1)
- Locking nunomaduro/collision (v7.8.1)
- Locking nunomaduro/termwind (v1.15.1)
- Locking phar-io/manifest (2.0.3)
```

**Figure 1:** Laravel installation

### **2.1.2: CONFIGURING THE DATABASE**

Configuring the database in a Laravel application involves setting up the database connection and defining the database-related settings in Laravel project. Laravel uses an '.env' file to manage environment-specific configurations including databases settings. configure database connection by editing the '.env' file with the database credentials. Inside the Laravel project root and configure database related settings.

DB\_CONNECTION: is used (mysql) database connection.

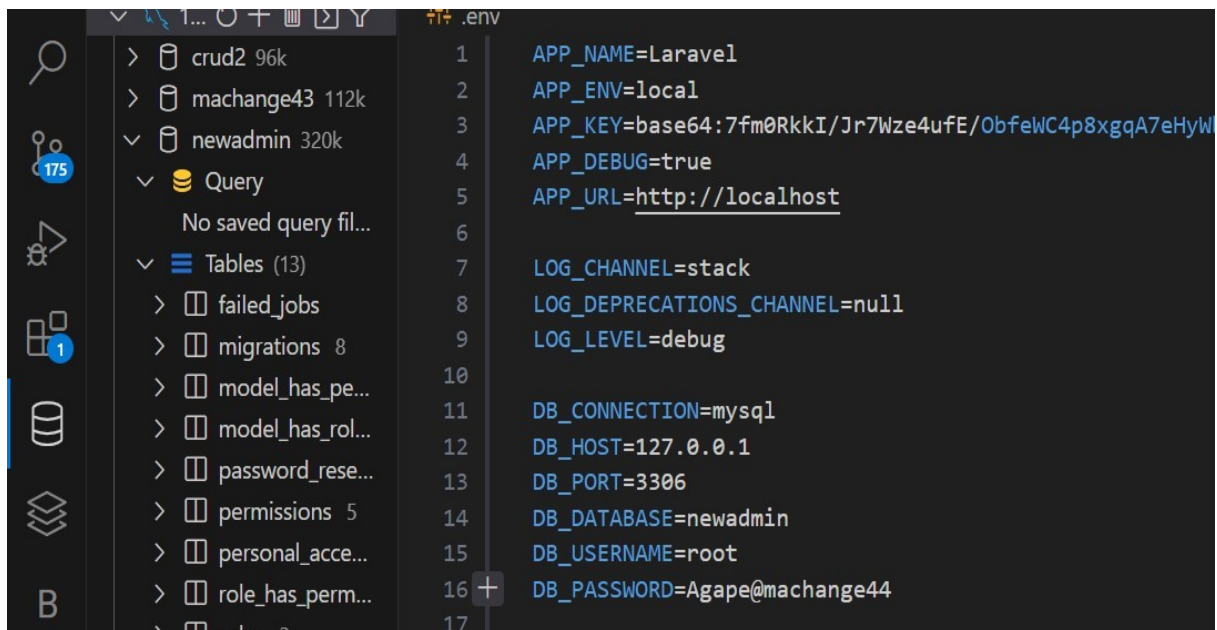
DB\_HOST: the host IP address of the database server

DB\_PORT: the port on which your database server listens

DB\_DATABASE: the name of the database

DB\_USERNAME: the username for accessing the database

DB\_PASSWORD: the password for the database user



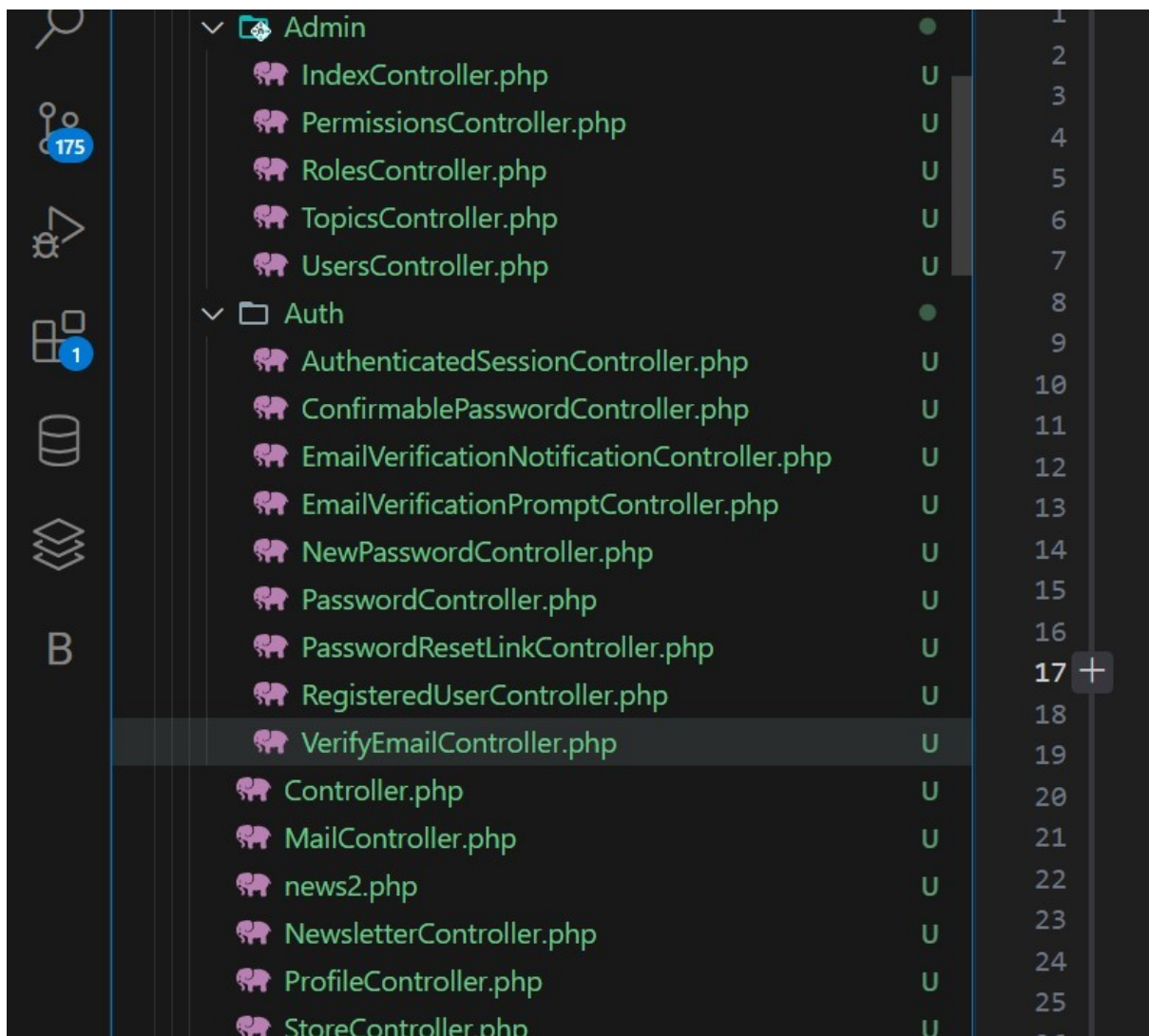
**Figure 2:** Database configuration

### 2.1.3: CREATING AUTHENTICATION SYSTEM

Authentication is a fundamental part of web applications, allowing users to securely access and manage their accounts. Laravel provides a comprehensive and easy to use authentication system right out of the box. Laravel provides built-in authentication which can be generated using Artisan commands.

#### **php artisan make:auth**

This command generates the necessary controllers, views and the routes for the user registration and login.



**Figure 3:** Authentication file

## 2.1.4: SETTING UP THE ENVIRONMENT VARIABLE

In the '.env' file environment variables for newsletter system, such as the sender's email address and email sending configuration. Which we connect it with an email delivery platform the Mailtrap so as we can test our emails when sending them to avoid using a normal gmail account because if u send roughly more than 5 email per second then scan block your account because it will be considered as spam email.

MAIL\_DRIVER: the smtp

MAIL\_HOST: smtp.mailtrap.io

MAIL\_PORT: the port

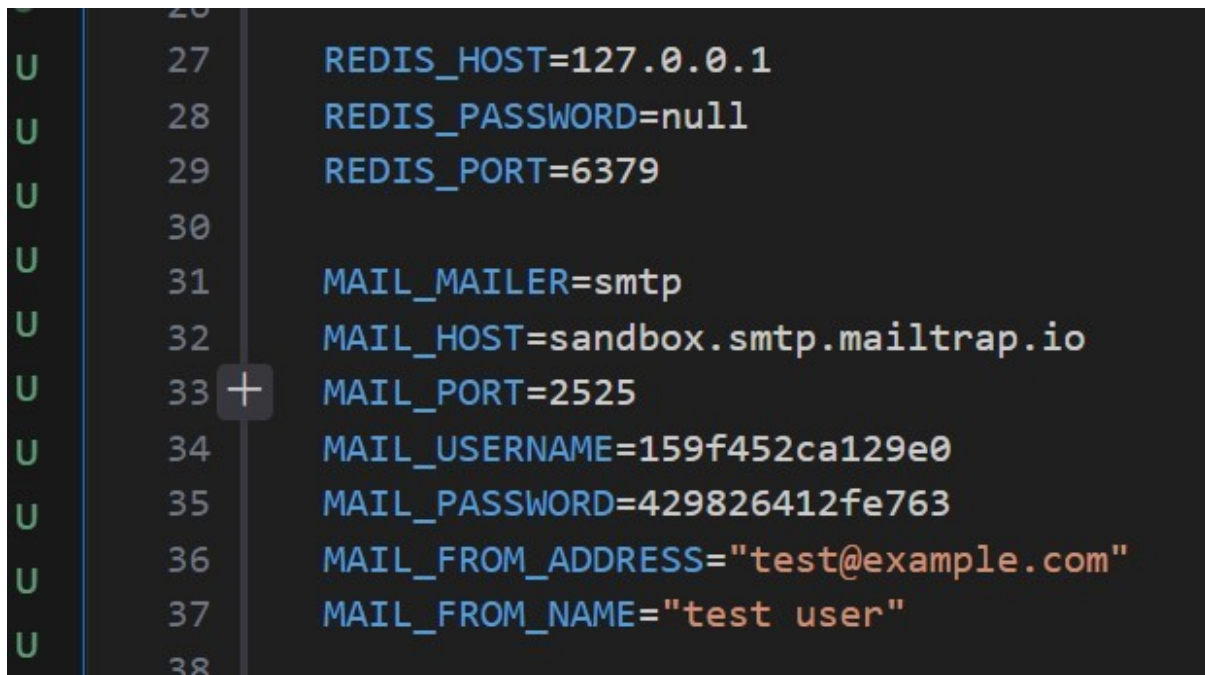
MAIL\_USERNAME: your mail username

MAIL\_PASSWORD: your mail password

MAIL\_ENCRYPTION: null

MAIL\_FROM\_ADDRESS: your email address

MAIL\_FROM\_NAME: "name"

A screenshot of a code editor showing a .env file. The file contains several environment variables for a newsletter system, including Redis and Mail configuration. The variables are listed with their corresponding values. A vertical line is visible on the left side of the editor, and a small '+' icon is next to line 33.

```
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=smtp
32 MAIL_HOST=sandbox.smtp.mailtrap.io
33 + MAIL_PORT=2525
34 MAIL_USERNAME=159f452ca129e0
35 MAIL_PASSWORD=429826412fe763
36 MAIL_FROM_ADDRESS="test@example.com"
37 MAIL_FROM_NAME="test user"
38
```

*Figure 4:* Environment variable via mail

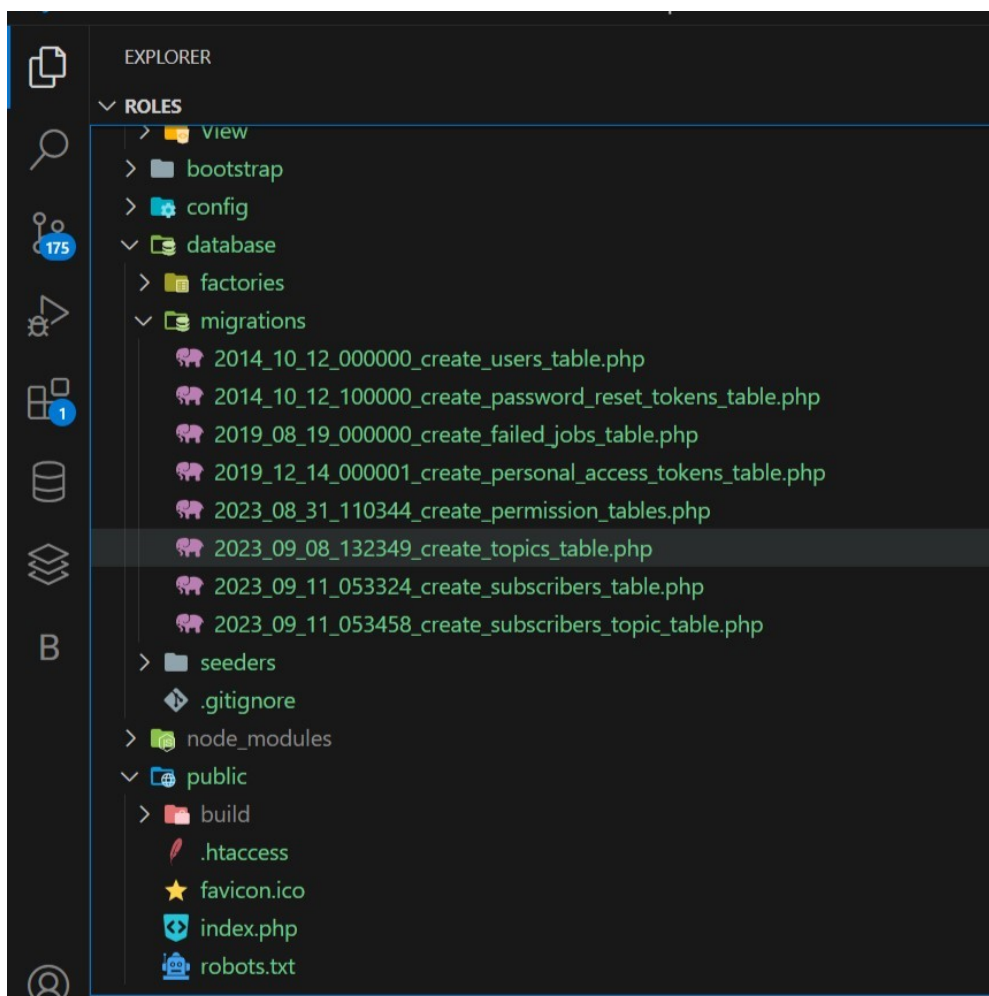


### 2.2.1: DATABASE SCHEMA

The database schema is the foundation of the newsletter system. Design the users table , roles table, permission table, topic table which will handle all information's which are kept inside the database and this tables are created using artisan command which is written inside the terminal

**php artisan make:migration create\_table\_(table name)**

In this newsletter project we have user table, roles table, permission table topics table, we have also the subscribers table subscribers\_topic table which helps to carry data entered by the user in the forms.



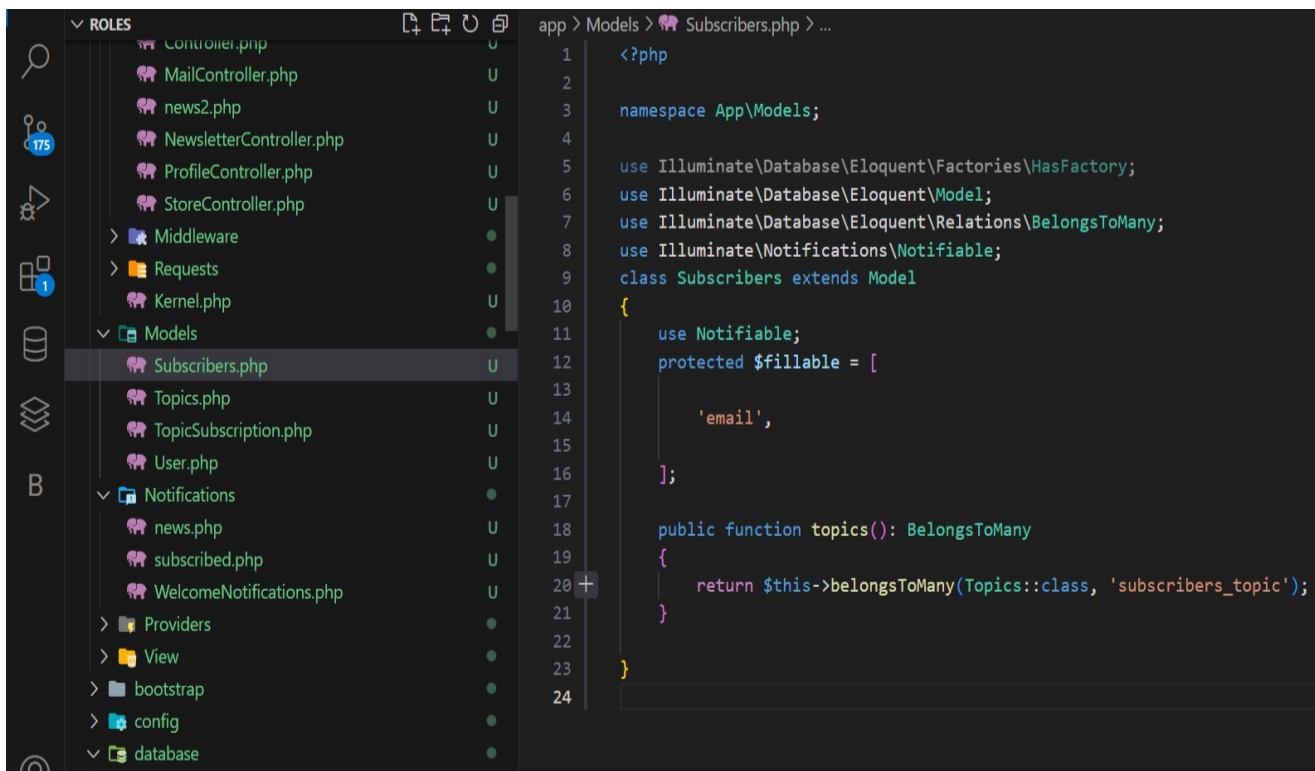
*Figure 5: Database schema*

## 2.2.2: CREATING THE NEWSLETTER MODELS

Using Laravel, models represents databases tables and the migrations define the structure of those tables. And the model contain methods for interacting with the databases. Model simplify database operations like creating, reading updating and deleting records. They also facilitate date validation and retrieval. By encapsulating database logic, models help to make it easy to maintain and organize your application's data, ensuring efficient and structured database management within laravel's powerful framework. Model can be made using the following command:

**php artisan make:model (model\_name) -m**

This command creates a model in the 'app' directory and generates a corresponding migration file inside the 'database/migrations' directory.



*Figure 6: Newsletter models*

## 2.2.3: CREATING THE VIEWS FILES AND BLADES

Views in Laravel refers to the templates or HTML files that define the structure and layout of web pages. They serve as the presentation layer of your application, responsible for rendering dynamic content and displaying it to the users. In Laravel, views are typically written using the blade templating engine which offers a convenient way to work with the php code inside the html templates.

## **THE ROLE OF VIEWS IN MVC**

In the MVC architectural pattern, views are responsible for presenting data to users in a readable and user-friendly format. They receive data from the controller and use it to generate the final HTML that users see in their web browsers. Views ensure a clear separation of concerns, as they focus mainly on the presentation logic while binding up business logic to controllers and data storage to models.

## **BLADE TEMPLATING ENGINE**

Laravel's Blade templating engine is a powerful tool for creating dynamic and reusable views. Blade provides a nice syntax for common tasks like displaying data, looping through arrays, and including subviews. It also offers features for extending layouts and creating custom directives, making it a big choice for web development.

The views is where we implement the crud operation which helps us to delete, create update and read which helps the users to operate the and this helps the user to see or this is what is displayed to be seen by the user in the application and is where the forms are been created in Laravel.

## 2.3: CRUD OPERATION

CRUD operations refer to the basic operations that can be performed on data in a database or data storage system. The acronym CRUD stands for Create, Read, Update, and Delete. These operations are fundamental in database management and are often used in web applications, APIs, and software systems to interact with and operate data. Here is explanation of each CRUD operation.

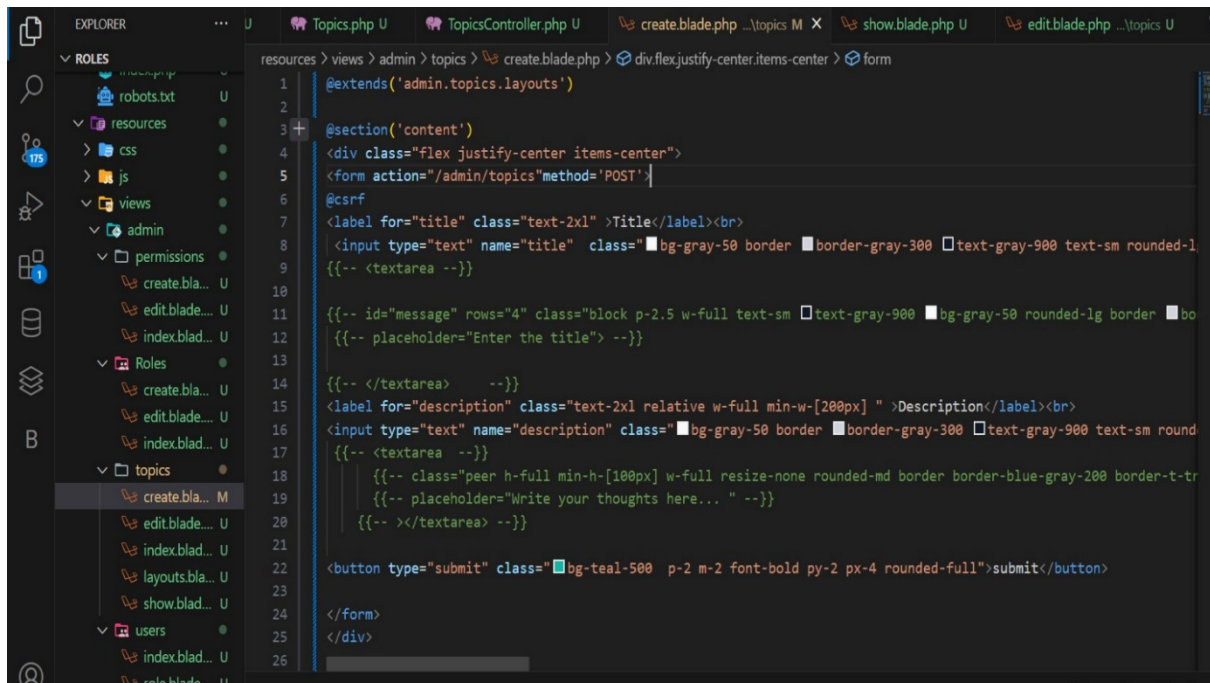
### 2.3.1: CREATE OPERATION

The purpose of the Create operation is to add new data or records to a database or data store. It enables users or applications to input fresh information and store it for future use. This operation is fundamental for applications because it allows them to grow and adapt as new data is generated or entered.

**Action in the Create Operation:** The core action in the Create operation involves inserting a new row (in the case of a relational database) or a new document (in the case of NoSQL databases like MongoDB) into a database table . When performing this action, several steps are involved:

1. **User Interaction:** In a web application, the process often begins with user interaction. For example, a user might want to create a new account on a social media platform, so they fill out a registration form with details like their username, email address, password, and other related information.
2. **Data Validation:** Once the user submits the form, the application performs data validation to ensure that the entered information is correct, complete, and follows some rules. This validation helps maintain data integrity.
3. **Data Processing:** After validation, the application processes the data to prepare it for storage. This might involve encrypting sensitive information like passwords, generating unique identifiers, or formatting data as required by the database.
4. **Database Interaction:** The application communicates with the database to insert the newly prepared data.

5. **Feedback to User:** Once the data is successfully stored in the database, the application provides feedback to the user, confirming that their action was successful. This feedback could be a confirmation message or a redirection to a relevant page.

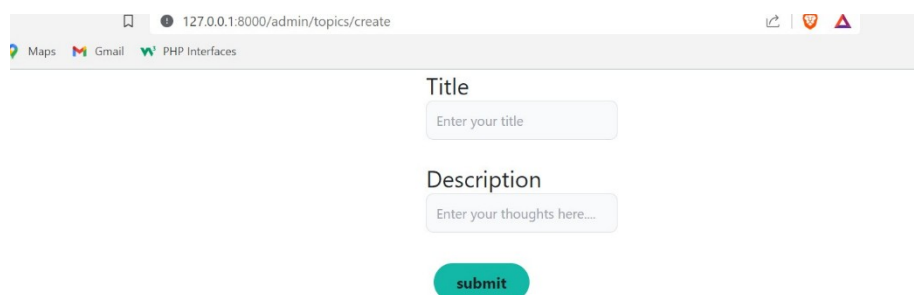


```

1 @extends('admin.topics.layouts')
2
3 @section('content')
4 <div class="flex justify-center items-center">
5 <form action="/admin/topics" method="POST">
6 @csrf
7 <label for="title" class="text-2xl">Title</label><br>
8 <input type="text" name="title" class="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-l
9 {{-- <textarea --}}
10
11 {{-- id="message" rows="4" class="block p-2.5 w-full text-sm text-gray-900 bg-gray-50 rounded-lg border bo
12 {{-- placeholder="Enter the title"> --}}
13
14 {{-- </textarea --}}
15 <label for="description" class="text-2xl relative w-full min-w-[200px]">Description</label><br>
16 <input type="text" name="description" class="bg-gray-50 border border-gray-300 text-gray-900 text-sm round
17 {{-- <textarea --}}
18 {{-- class="peer h-full min-h-[100px] w-full resize-none rounded-md border border-blue-gray-200 border-t-tr
19 {{-- placeholder="Write your thoughts here... "> --}}
20 {{-- ></textarea --}}
21
22 <button type="submit" class="bg-teal-500 p-2 m-2 font-bold py-2 px-4 rounded-full">submit</button>
23
24 </form>
25 </div>
26

```

**Figure 7:** Create operation source codes



127.0.0.1:8000/admin/topics/create

Title

Enter your title

Description

Enter your thoughts here...

submit

**Figure 8:** Output in create operation

## 2.3.2: READ OPERATION

The primary purpose of the Read operation is to retrieve and view data from a database or data store. It allows users or applications to access information that has been previously stored. This operation is important for presenting data to users, making it one of the basic aspects of any application.

**Action in the Read Operation:** The core action in the Read operation involves querying and fetching data from the database without making any modifications to it. When performing this action, several steps are typically involved:

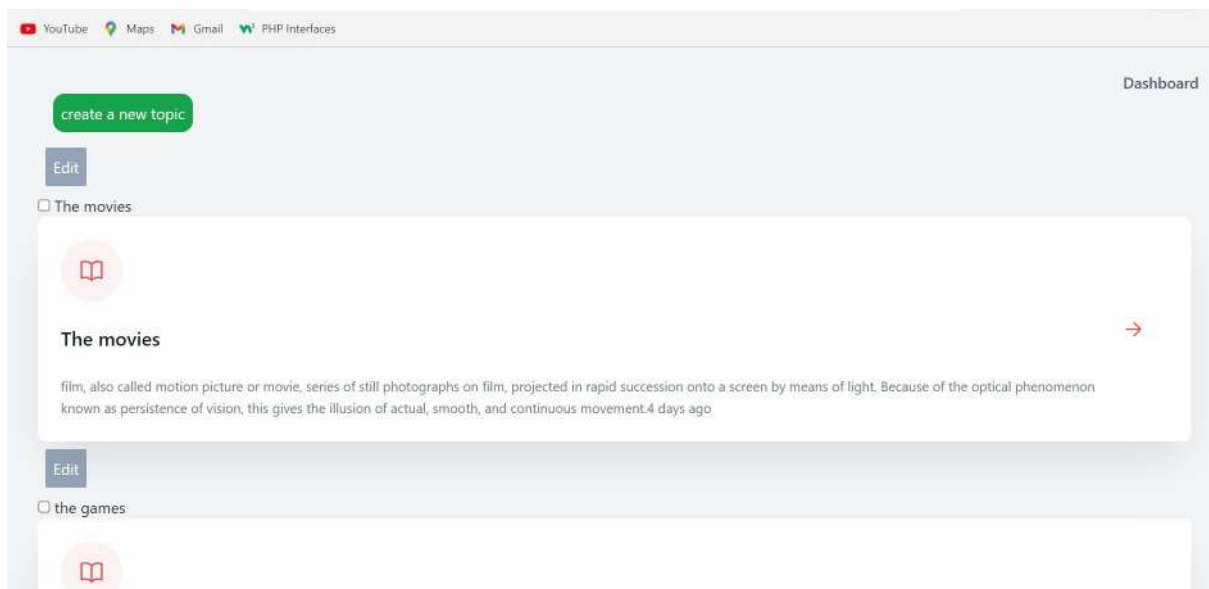
1. **User Request:** The Read operation often begins with a user request. For instance, a user visits an e-commerce website and navigates to the product catalog page. They want to view a list of available products.
2. **Database Query:** In response to the user request, the application interacts with the database by executing a query. This query specifies what data should be retrieved. In SQL databases, this might be a SELECT statement.
3. **Data Retrieval:** The database processes the query and retrieves the requested data from its tables, collections, or documents. This data is then sent back to the application.
4. **Data Presentation:** Once the data is fetched from the database, the application formats and presents it to the user. This presentation could take the form of a webpage, a mobile app screen, an API response, or any other appropriate medium.
5. **User Interaction:** The user interacts with the presented data. They can view, analyze, or take further actions based on the information displayed.

```

1 @extends('admin.topics.layouts')
2 @section('content')
3     <div class="row">
4         <div class="col-lg-12 margin-tb">
5             <div class="pull-left">
6                 <h2 class="flex justify-center text-2xl"> Show Topics</h2>
7             </div>
8             <div class="pull-right">
9                 <a class="btn btn-primary" href="{{ route('admin.topics.index') }}"> Back</a>
10            </div>
11        </div>
12    </div>
13
14    <div class="row">
15        <div class="col-xs-12 col-sm-12 col-md-12">
16            <div class="form-group">
17                <strong>Title:</strong>
18                <a href="{{ $topics->title }}"></a>
19            </div>
20        </div>
21        <div class="col-xs-12 col-sm-12 col-md-12">
22            <div class="form-group">
23                <strong>Description:</strong>
24                <a href="{{ $topics->Description }}"></a>
25            </div>
26        </div>

```

**Figure 9:** Read operation source codes



**Figure 10:** Output in read operation

### 2.3.3: UPDATE OPERATION

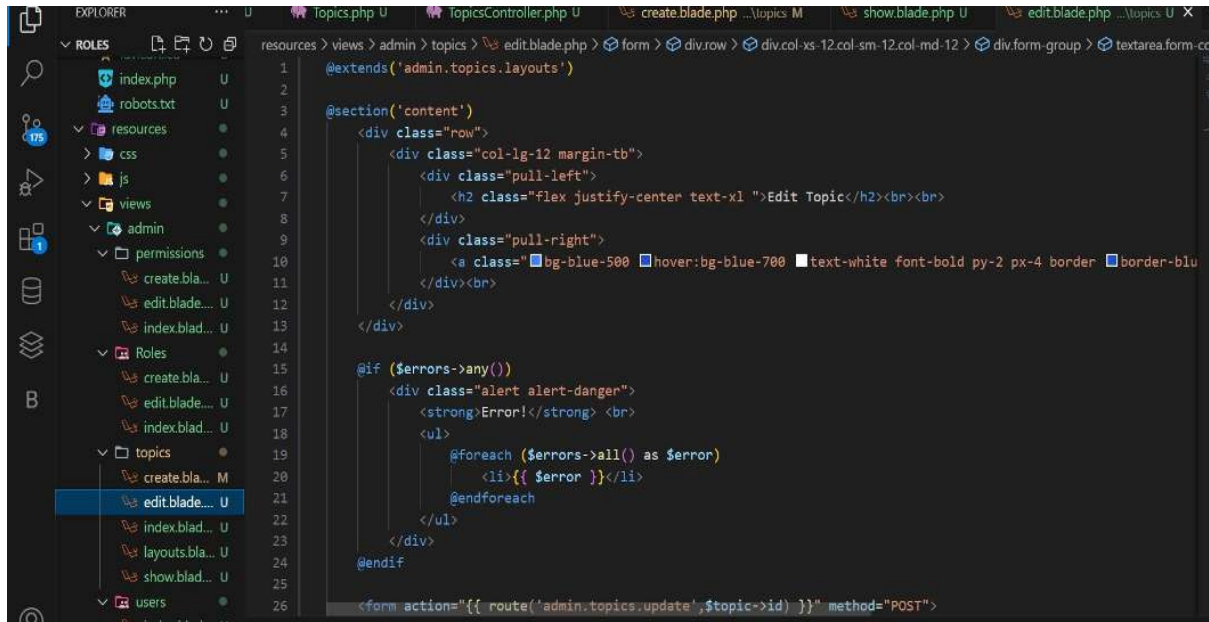
The general purpose of the Update operation is to modify existing data in a database or data store. It allows users or applications to change the values of one or more fields in an existing record without creating a new one. This operation is essential for maintaining data accuracy and reflecting real-world changes.

**Action in the Update Operation:** The core action in the Update operation involves identifying a specific record in the database that needs to be modified and then changing the values of the desired fields within that record. When performing this action, several steps are typically involved:

1. **User Request:** The Update operation is often initiated by a user's request to modify data. For instance, a user on a social media platform may want to update their profile information, such as changing the roles and permissions.
2. **Data Retrieval:** The application retrieves the existing data from the database to display it to the user. This data serves as the initial values that the user can modify.
3. **User Interaction:** The user interacts with a user interface element that allows them to make changes. This could be a form or a set of input fields pre-populated with the existing data.
4. **Data Modification:** When the user submits the updated data, the application processes the changes. It identifies the specific record in the database that corresponds to the user and updates the selected fields with the new values.
5. **Database Update:** The application sends an update query to the database, specifying which record to modify and what changes to apply. In SQL databases, this is typically done using an UPDATE statement.

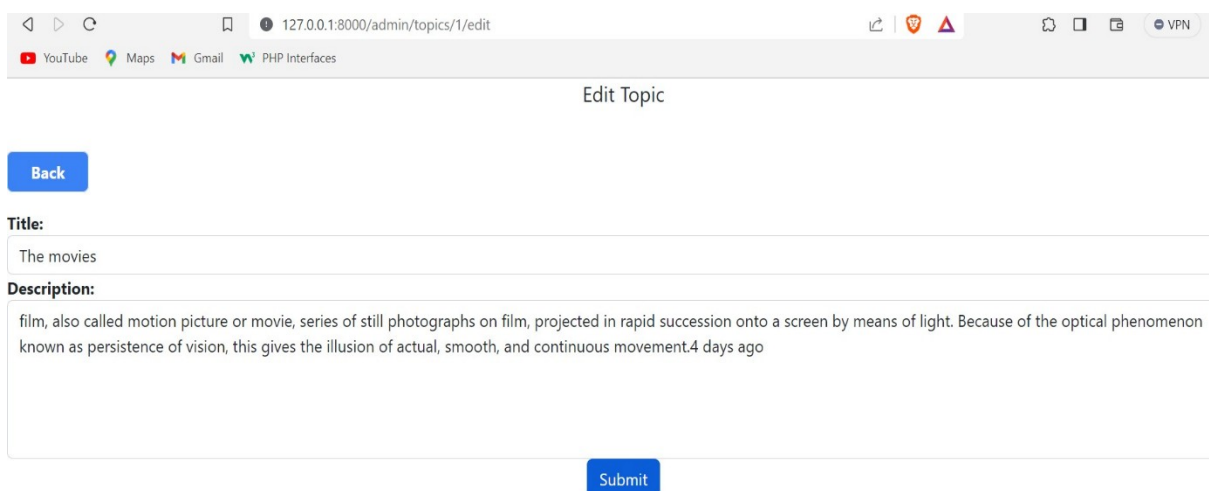


6. **Feedback to User:** After successfully updating the data, the application provides feedback to the user, confirming that the changes have been saved. This feedback may include a success message or a redirection to a relevant page.



```
1 @extends('admin.topics.layouts')
2
3 @section('content')
4     <div class="row">
5         <div class="col-lg-12 margin-tb">
6             <div class="pull-left">
7                 <h2 class="flex justify-center text-xl">Edit Topic</h2><br><br>
8             </div>
9             <div class="pull-right">
10                 <a class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 border border-bl
11             </div><br>
12         </div>
13     </div>
14
15     @if ($errors->any())
16         <div class="alert alert-danger">
17             <strong>Error!</strong> <br>
18             <ul>
19                 @foreach ($errors->all() as $error)
20                     <li>{{ $error }}</li>
21                 @endforeach
22             </ul>
23         </div>
24     @endif
25
26     <form action="{{ route('admin.topics.update',$topic->id) }}" method="POST">
```

**Figure 11:** Update operation Source code



127.0.0.1:8000/admin/topics/1/edit

YouTube Maps Gmail PHP Interfaces

### Edit Topic

[Back](#)

**Title:**  
The movies

**Description:**  
film, also called motion picture or movie, series of still photographs on film, projected in rapid succession onto a screen by means of light. Because of the optical phenomenon known as persistence of vision, this gives the illusion of actual, smooth, and continuous movement.4 days ago

[Submit](#)

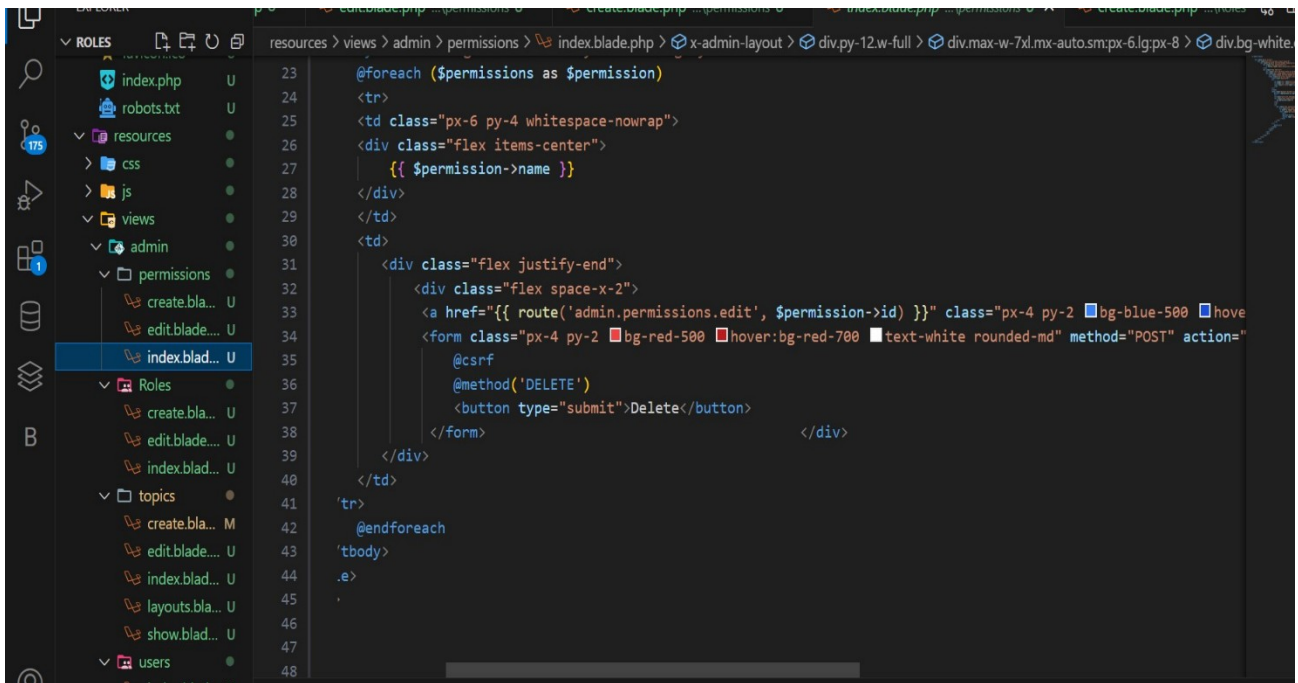
**Figure 12:** Output in Update operation

### 2.3.4: DELETE OPERATION

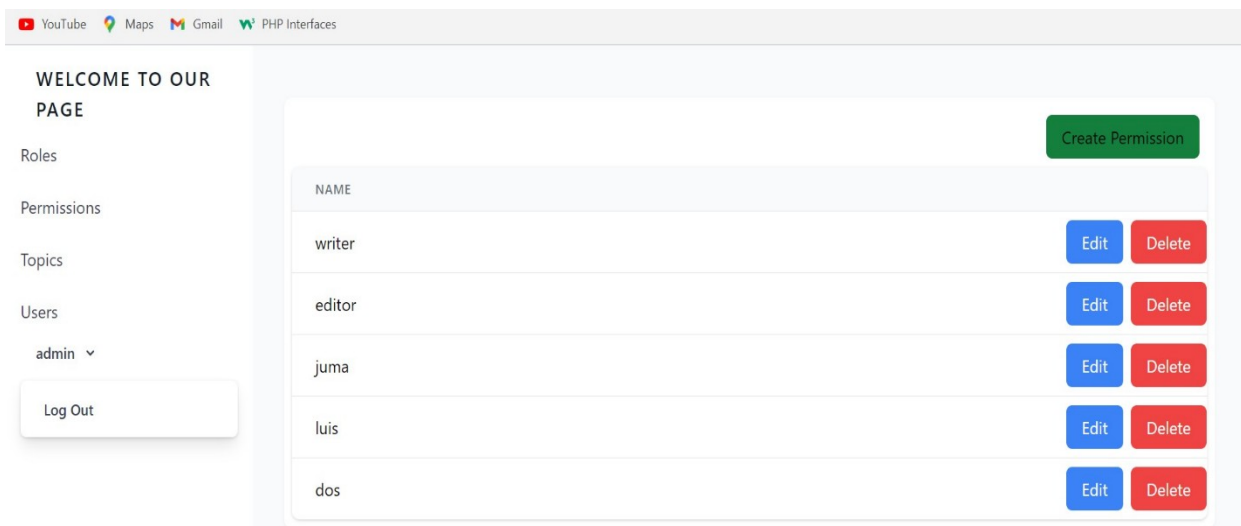
The goal of Delete operation is to remove data or records from a database or data store. It allows users or applications to eliminate information that is no longer needed or should not be reserved. This operation is crucial for maintaining data cleanliness and ensuring that only relevant and up-to-date information is stored.

**Action in the Delete Operation:** The core action in the Delete operation involves specifying a condition or criteria to identify the data or records that should be deleted and then removing them from the database. When performing this action, several steps are typically involved:

1. **User Request:** The Delete operation is often initiated by a user's request to remove specific data. For example, a user may want to delete a post they previously published on a social media platform.
2. **Identification of Data:** The user or application identifies the data or record(s) that should be deleted. This could be done through a user interface element like a "Delete" button or a selection process.
3. **Data Removal:** Upon confirmation, the application processes the deletion request. It uses the specified criteria or identifier to locate and delete the selected data from the database.
4. **Database Deletion:** The application sends a delete query to the database, specifying which records should be removed. In SQL databases, this is typically done using a DELETE statement.
5. **Feedback to User:** After successfully deleting the data, the application provides feedback to the user, confirming that the data has been removed. This feedback may include a success message or a redirection to a relevant page.



**Figure 13:** Delete operation source codes



**Figure 14:** Output in delete operation

### 2.3.5: ROUTES

Laravel's **web.php** file is a fundamental component of the Laravel framework responsible for defining web routes. These routes determine how HTTP requests are handled and which controllers or actions are executed in response to those requests. In this comprehensive details, this **web.php** file in Laravel I will explain covering various aspects and use cases.

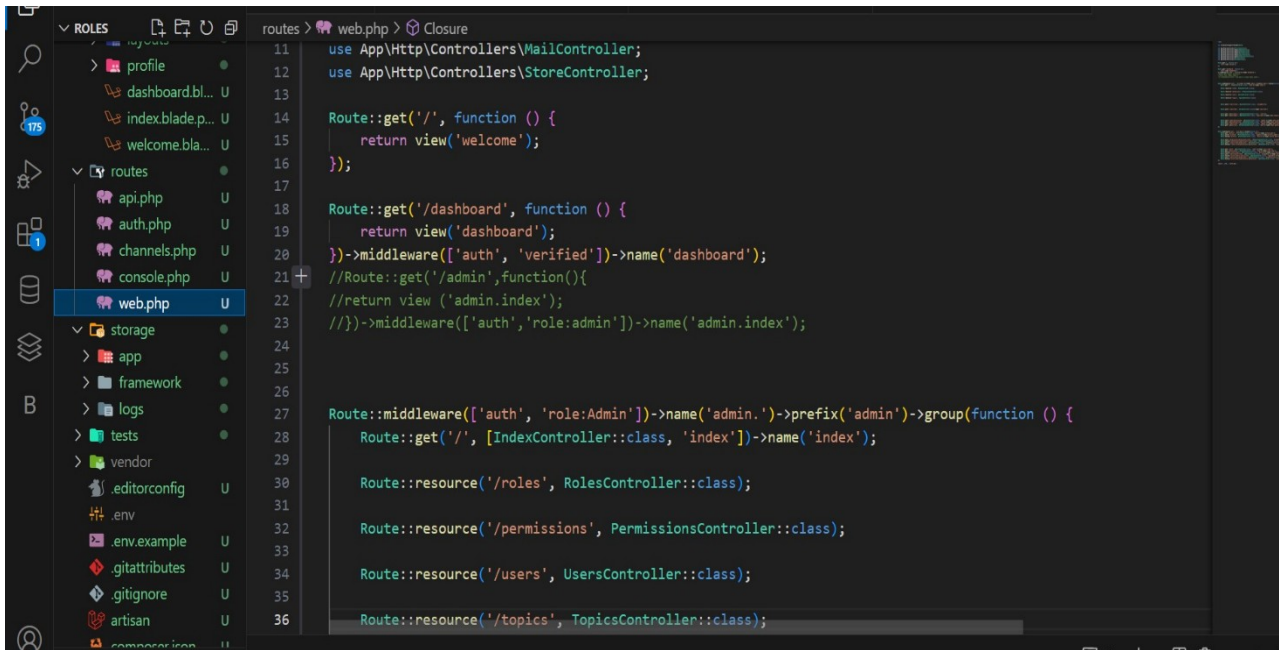
The **web.php** file, located in the **routes** directory, is the primary location for defining web routes. These routes are intended to handle HTTP requests coming from web browsers and web clients. The **web.php** file uses Laravel's expressive routing system, which makes it easy to define and manage routes for your web application.

### Basic Route Definition

```
Route::get('/home', function () {  
    return view('home');  
});
```

In this example, we define a route using the **Route::get** method. This method creates a route that responds to HTTP GET requests to the **/home** URL. When this route is accessed, it executes an anonymous function that returns a view named 'home'. This is a typical way to define routes that render views.

The **web.php** file is where web routes are defined to handle HTTP requests from browsers. These routes are key to mapping URLs to specific actions or views within the application. Used to define routes with various HTTP methods, name them for easy URL generation, and utilize parameters to get changing values from URLs. Middleware can be applied to enforce authentication, authorization, and other request processing tasks. Route groups help organize related routes, while model binding simplifies working with database records. Overall, **web.php** serves as the foundation for structuring how your Laravel application responds to web requests, offering flexibility and control.



*Figure 15: Routes source codes*

## **CONCLUSION ON NEWSLETTER**

In conclusion, creating a newsletter system using Laravel is a powerful and adaptable solution for businesses and organizations aiming to engage their audience through email marketing. Laravel's MVC architecture, Blade templating engine, database management capabilities, and enhanced security features provide a solid foundation for building and managing a newsletter system efficiently.

By focusing on key components such as subscriber management, email templates, analytics, compliance, and scalability, one can create a newsletter system that not only delivers content effectively but also respects user preferences and legal requirements. Laravel's flexibility, joined with its extensive ecosystem of packages and libraries, empowers developers to build newsletter systems that can adapt and grow with the needs of the organization, ultimately leads to engagement and making valuable relationships with subscribers.

## **CHALLENGES FACED DURING IPT**

There was a big challenge of getting a place to conduct the industrial practical training. Shorter time of IPT which led to study partial some of the concepts in order so as to save time and a binding with the speed of the company. Also at the firm there was an electricity shortage some of the time and this led to delay in accomplishing of the task which you have been assigned with the supervisor who is responsible with you.

### **RECOMMENDATIONS**

My recommendations are the time allocated for industrial practical training should be increased so a person can grasp a lot of knowledge. Also a challenge of finding a place to pursue your industrial practical training should be allocated by the university.

### **REFERENCES**

W3Schools. (2021, June 15). HTML Tutorial.

Laravel application development by Terry Matula year 2013

Laravel: Up & Running by Matt Stauffer year 2023

Battle ready Laravel: by Ashley Allen year 2022

Koleksi Aplikasi Web Laravel: by Ir. Yuniar Supardi, Yogi Syarif year 2023

Laravel Documentation by Taylor Otwell copyright 2011-2023