OULU UNIVERSITY OF
APPLIED SCIENCES

Bishal Shah

DESIGNING OF WEBSITE ALONG WITH BLOG AND FORUM USING LARAVEL

DESIGNING OF WEBSITE ALONG WITH BLOG AND FORUM USING LARAVEL

Bishal Shah
Bachelor's Thesis
Spring 2019
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author: Bishal Shah
Title of the bachelor's thesis: Designing of Website along with Blog and Forum using Laravel
Supervisor: Pekka Alaluukas
Term and year of completion: Number of pages: 60

The main objective of this bachelor's thesis was to develop a user-friendly responsive website along with a forum, blog and donation platform. The website can be used to share the stories, experience and donating for the cause.

This project work was carried out in cooperation with Village Development Helping Organization which is one of the non-profit social organization working in Nepal to save lives, eradicate poverty, increase agricultural production and provide social justice to all people.

The web application was built using the PHP framework Laravel which eliminates the developer time and helps in deploying the application rapidly. The MAMP localhost was used for the development of the project locally. Admin panels were created to manage and make the website dynamic whereas the Stripe payment system was used for the handling of the donation system.

As a result of the work, a responsive website along with a blog and forum was created and handed over to the client. However, the website is still under the testing phase since there is some room for improvement.

Keywords: Laravel, PHP, Blog, Forum, MVC

CONTENTS

VOCABULARY

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

JS – JavaScript

PHP – Hypertext Pre-processor

API – Application Program Interface

UI – User Interface

MVC – Model View Controller

1 INTRODUCTION

In this digitalizing world, it is very important for any web developer to know at least one of the frameworks and Laravel is one of the most successful PHP frameworks based on reviews on Google trend comparison. A PHP framework eliminates the developer time and effort to write the repetitive code and helps in deploying the application rapidly. Thus, a reliable framework also helps to lower down the development cost. (Crettenand 2017, Date of retrieval 1 November 2018).

There are several advantages of using Laravel among others because of its features, specification and highly growing community of programming geeks and developer. Furthermore, using a robust framework like Laravel also helps to improve the security as a developer do not need to worry about the basic security threats like SQL injection, Cross-site scripting, and cross-site request forgery.

The Laravel Framework contains a collection class which outmost the PHP arrays by using it internally with the help of any helper function from the collection. The helper method helps in working with the data in any functional way. (Zaki 2017, Date of retrieval 1 November 2018)

The purpose of this thesis work was to learn a new framework and to use it efficiently to build a web application along with a blog and forum for a social organization named Village Development Helping Organization

The development of Laravel has come a long way since its release on June 9, 2011. Every new release has been able to fix the bug, add a new feature and improve the available package. The latest version Laravel 5.7 was released on September 4, 2018, it has been used for the development of the application for this thesis project

Village Development Helping Organization is a non-profit social organization working in Nepal to save lives, eradicate poverty, increase agricultural production and provide social justice to all people. This organization was established in

2009 A.D by Mrs. Pushpa Shah to reflect the family commitment of giving back to the society. Over the years they have used their investment to increase their network and experience in completing the mission to help the people out in society. They are doing multiple projects all along Nepal which is currently being funded by the Nepal Government. The number of employees and their community member is increasing day by day and they do not have any website or database to store their information.

Nowadays, all organizations have their website as well as the blog where they can convey the message easily to and from the community. To overcome this problem, this project along with a blog and forum is being implemented so that more people's problem can be brought in the limelight and be solved. (An Overview of PHP Framework Guides for Developers, Date of retrieval 1 November 2018)

2 REQUIREMENTS

2.1 System Feature

2.1.1 Description of the Organization

Any user can browse the main page of the website and get to know the fundamentals of the organization. They can see the latest Blog Posts, pictures from their latest work-related field, know the number of followers of their different social accounts and as well get information about the employees. The user can also subscribe to monthly newsletter from the main page of the website.

2.1.2 Registration and login

This feature enables any user to be the member of the organization and the registered member can read and comment on all the blog.

Similarly, In the forum section, the logged in user can ask any question to the community and as well as reply the others question. They will also be able to upvote the answers.

2.1.3 Feedback and Query

This feature enables any user to give feedback or ask any question by using the Contact Us page section on the website. The user has to fill in their details so that the organization can reply to their query. Both the admin and the user are notified by email about the feedback and query.

2.1.4 Blog

The user can view all the blog posted by the organization and can also comment on any specific blog by giving their email address in the comment section.

2.1.5 Admin Panel

This feature is intended for the admin handling the blog and overall setting of the website. This area will have the following sub-feature which are given below

2.1.5.1 Dashboard

The admin can view the number of the published post, the total number of users, number of trashed posts, categories and tags.

2.1.5.2 Category

The admin can view, create, edit and delete the category for the blog section.

2.1.5.3 Tags

The admin can view, create, edit and delete the Tags for the blog section. Besides this, the admin can also view the post associated with the specific tag and delete them instantly.

2.1.5.4 Posts

The admin can view, create, edit and delete the posts for the blog section. The post deleted from this section will be sent to trash which can later be retrieved or deleted permanently. Markdown editor which is quite popular is used to help the administration to create post description.

2.1.5.5 Site setting

The admin can change the name of the website, their contact information from this section. They can also add or delete new picture for their gallery and the

slider section from this feature. They can also update their employee information from this section.

2.1.5.6 User Setting

The admin can create a new user, delete the existing one or change the password of the user from this feature. They can also give admin right permission to other users. The super administration is created from the editor which cannot be deleted.

2.1.5.7 Comment

The admin can change the comment including deleting them but not alter the contact information of the commented user using this feature.

2.1.6 Profiles

This feature enables any logged in user to view their profile, change the default picture and update their information according to their needs.

2.1.7 Donate

Any user can donate $1 to the organization by filling in their card details on the donate page section. Both the admin and the user are notified by email about the transaction. The stripe payment system is used to complete the transaction where the user can directly pay through their card. After the payment is completed by the user, the fund is transferred first into the organization stripe account and later to the organization bank’s account by reducing the service charge for every transaction.

2.1.8 Forum

The logged in user can ask any question in the community as well as reply to other. They will also be able to upvote answers. This feature will be a real-time update so that the user does not have to refresh the page to get the notification. Pusher technology along with Laravel API and Laravel WebSockets is used to create the single page forum which is assisted by Vue JS and Vuetify for front end development.

3 IMPLEMENTATION

3.1 User Interface

In this chapter, the UI model is built which helps the reader demonstrate how the real application will be completed. The user interface model, which is shown in figure 1 is built with the help of cloud balsamiq.

3.1.1 Home page

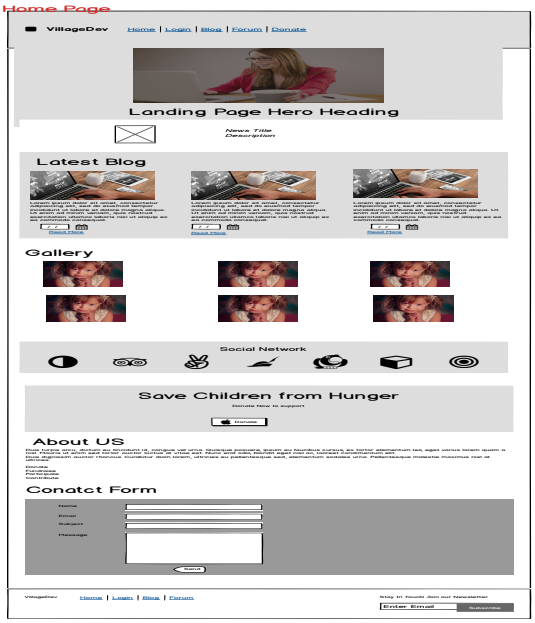


FIGURE 1. Design of Home Page for Village Development

The Figure 1 illustrates the appearance of the main page of the website. The page shows all the information about the organization such as the contact form, providing donation, gallery, latest blogs and their social networking sites.

On the main landing page, any user can make the donation by clicking the donate button either in the navigation bar or in the body section of the website.

When they click the button, the Figure 2 pops up on the screen.

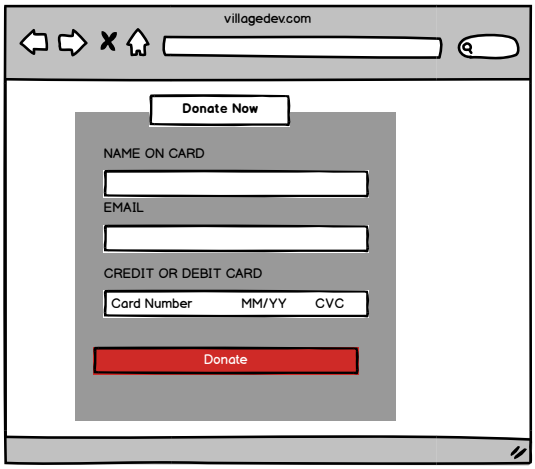


FIGURE 2. Donate Page

The user then needs to fill up their information. They will get an instant notification if they use the wrong email address or incorrect card details. If the payment is successful, both the user and the admin will get an email notifying about the transaction. The server-side handling is taken care by the Stripe where the money is stored for the time being which is later transferred to the organization bank account reducing the service fee.

As shown in the figure 1, the user can see the top latest blog of the organization and by clicking the read more button or clicking the link of blog in the navigation bar, they are directed to the blog page where they can see all the blog posts as well as make a comment on the specific blog which is more explained in the section 3.1.3.

The user can give any feedback or feel to have any query by filling their contact detail on the contact form of the Home page. The user email is saved in the database and both the admin and the user are notified by email. The user can also subscribe to the monthly newsletter where their data is saved in the database for a further notification.

The user can view the image posted by the organization and they can also view the image in the full screen by clicking the icon in the image. The user also has an option of downloading the image from the website.

The user can also view the information of the employee and connect with them by clicking the link on their image.

The user can also view the forum by clicking the link in the navigation bar. They can ask any question in the forum and as well as reply to other thread. They are able to upvote the answers of another user which is more explained in section 3.1.4.

From the other page, the user is directed to the main page of the website if they click the organization name or Home button section from the navigation bar.

3.1.2 Login and Signup Page

FIGURE 3. Login Page

FIGURE 4. Signup Page

The figure 3 and 4 are the login and signup page respectively where the user can either sign in with their credential or create a new account so that they can get access to the organization forum and blog.

During the signup procedure, the user needs to give their information which is stored in the database. The user is then redirected to the main page of the website where the username is stored in the session and the session usually lasts for 60 minutes. During the signup, the user gets a default image, default description of their social account which they can later change by viewing their profile page.

During the logging, if there is a bad combination of password and user id, the user will get a flash message about their error which helps them in accessing the website. The user is normally blocked from accessing the website for 60 seconds after they have mismatched the combination 10 times. The throttle request is normally 10 times and it keeps on delaying the user again if they mismatched the combination. The request is usually stored in a session which

helps to count the number of login attempts made and respond accordingly with the time delay.

The user can request for the lost password by clicking the link on the login page. The user is directed to the other page and the link is sent from the database to the user's email so that they can reset their password. After they have reset their password using the link, they can normally access the website with their username and new password.

3.1.3 Blog Page

FIGURE 5. Single Blog page for Village Dev

The figure 5 illustrates the model for a blog post where the user can read the blog and as well as leave a comment to the specific blog post by giving their information. When the comment is created, the username is shown on the comment section along with their comment. The user can also read the comment by the other user but cannot reply to them.

The user can see all the tags associated with the current post and as well see the writer and time when the post was created. The user can also share the post by clicking on the social networking logo they want to share on.

The pagination and another post image along with description are listed on the right-hand side so that the user does not get diverted from the blog while accessing the website.

3.1.4 Forum

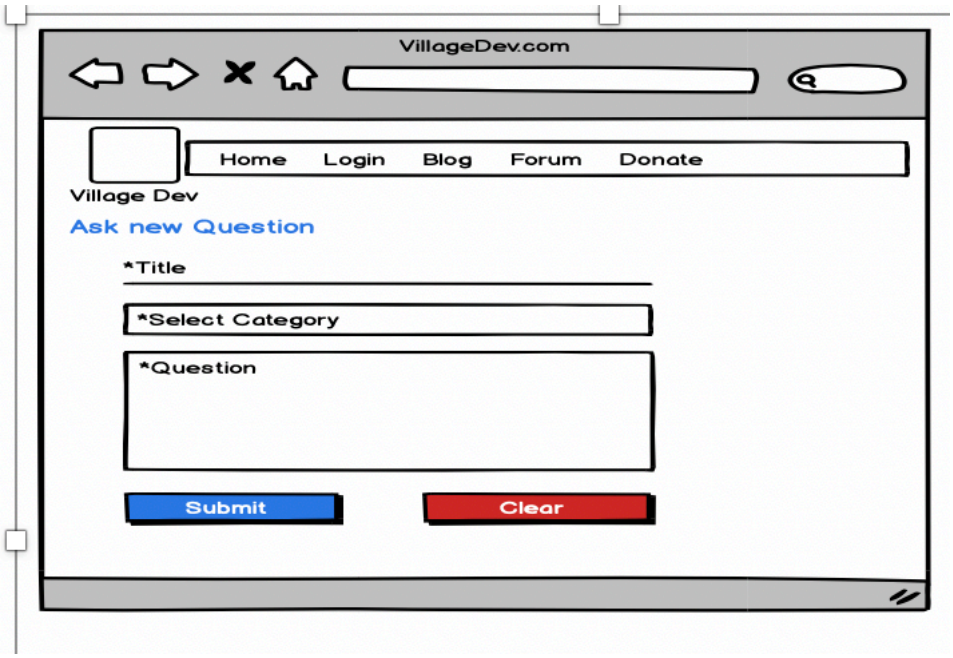


FIGURE 6. Forum Page

The logged in user can ask any question in the forum by filling all the required details. The user can also create a category in this section to elaborate on their topic. The user can reply to the other question and upvote their answer. When one user reply to another user they will get an instant notification about their reply. The user will be able to edit and update their answer as well as upvote and devote instantly.

3.1.5 Admin Dashboard

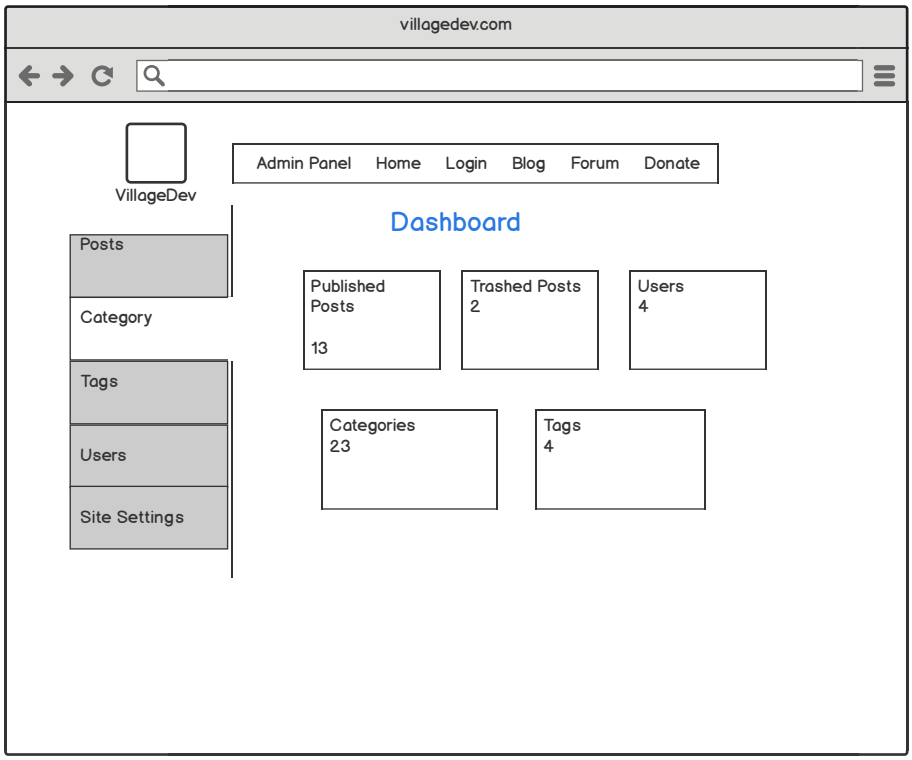


FIGURE 7. Admin Dashboard

The super admin is created from the terminal and is already active when the website is hosted over the internet. The super-admin cannot be deleted.

When the super admin is logged in, they can find the link of admin page in the home page of the website and when they click it, they are redirected to the page as shown in the figure 7. This page is protected by the middleware property of Laravel which means no guest can enter this property unless they have the admin right. The admin can see the basic information of the page e.g. the number of Posts, Category or Users.

3.1.5.1 Posts

When the admin clicks the posts link, they can see the image as illustrated in the figure 8.

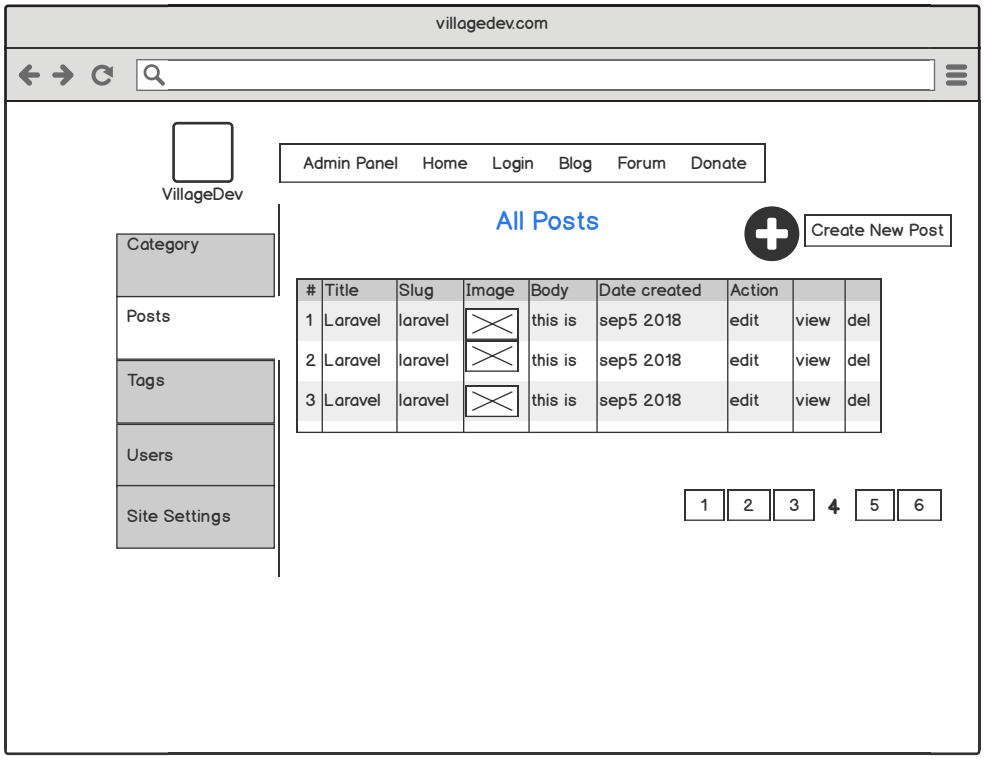


FIGURE 8. Admin post

The admin can see all the created post and other information regarding the post. They can also edit delete the post. The post deleted from this section is sent to trash where it can either be restored or deleted permanently.

The admin can navigate through different section using the pagination bar on the bottom of the page. They can also create a new post by clicking the icon of creating a new post. When they click the link, they are directed to the page as shown in the figure 9.

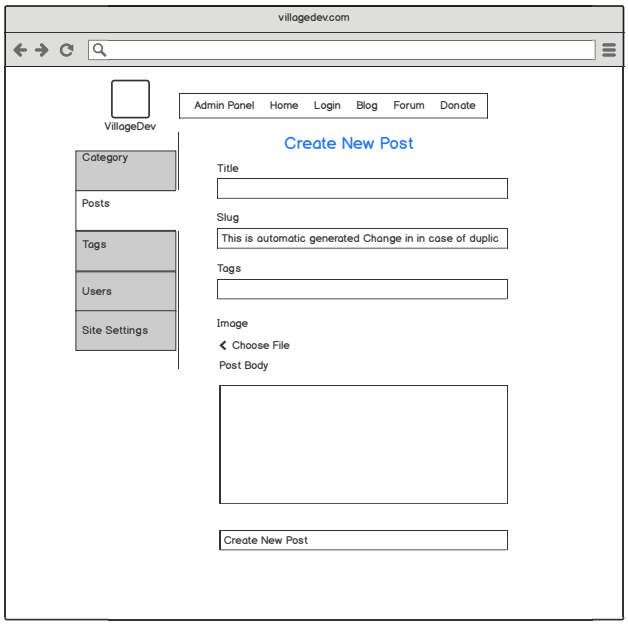


FIGURE 9. Admin Create New Post

Now the admin can create a new post by filling all the details. The slug is generated automatically from the title section but is kept here to avoid the duplicate. The admin can also view the specific post individually which layout is similar to this page.

3.1.5.2 Users

The super admin can create other admin and the normal user also using the backend panel. The admin has the right to delete any user and also give admin privilege to other users. The admin can also change the user email and password of another user.

3.1.5.3 Tags and Category

The post needs to have tags and the category section before they are created. When the admin clicks these links, the page looks similar to figure 10 as shown below.

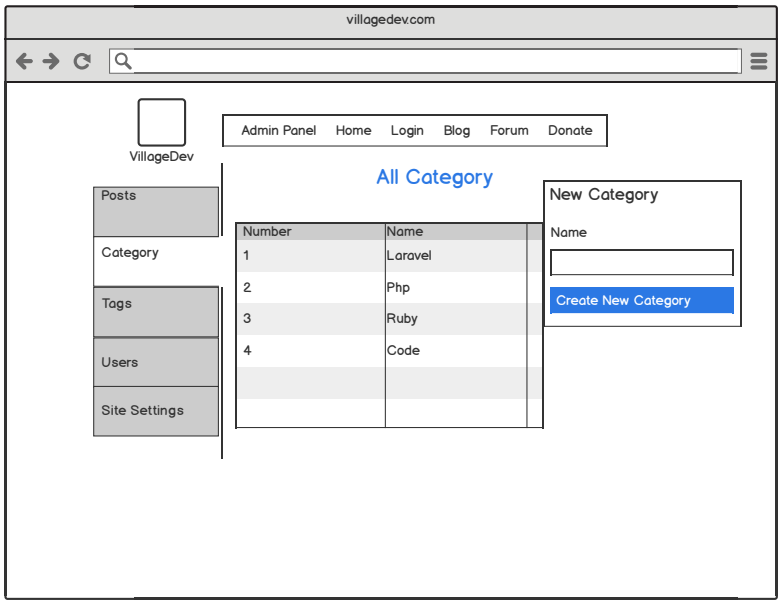


FIGURE 10. Category Section

The admin can create the new category and as well as view all the category in this page. They will also be able to edit and delete the category. The Tag section is similar to the category section.

3.1.5.4 Site Settings

The admin will be able to change the contact information of the main page, edit update and delete all the images for the website using this section. They can also add the employee details from this section. Overall, this section will help the webpage to become dynamic.

The UI (User Interface) also reflects the real-life situation, where different Laravel feature and technique are employed to create the web application. Although not all the UI page are shown here, the selected feature and technique shown in the above illustrations are enough for building all other feature.

4 FRONT END

4.1 HTML

HTML known as Hypertext Mark-up Language is a core language for creating any web pages and web application. This mark-up language informs web browser on how to display the content on the web pages. They are written in the format of tags. for instance, “body”, “head”, “section “.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

FIGURE 11. HTML syntax (W3schools HTML, Date of retrieval 9 October 2018)

In the above figure 11, the tags used in the HTML can be seen. Every tag starts with “<>” and closes with “</>”

4.2 CSS

CSS stands for Cascading Style Sheet which is used for formatting of the HTML pages. They provide the central location which defines various text style, back-ground color, table size and other related emphasization of the webpages. They can be written together in the same HTML file or in a separate file with the “.CSS” extension. The webpage can be linked to one or multiple CSS files by using the multiple link tag. (see the figure 12 below)

```
<link rel="stylesheet" type="text/css" href="what-is-css.css">
```

FIGURE 12. Link tag (The Server Side, Date of retrieval 13 October 2018)

4.3 JavaScript (JS)

JavaScript is a client-side programming language based on ECMAScript whose syntax is much more similar to C Programming language. This programming language proceeds on the browser which means any function of JavaScript can be loaded even without communicating with the server. The JavaScript code can be placed in HTML file within <script> tag or can also be used in a separate file with extension .JS. Below is an example which adds any two number.

```
<script>
  function sum(a,b)
  {
    return a + b;
  }
  var total = sum(7,11);
  alert(total);
</script>
```

FIGURE 13. JavaScript Function (TechTerms, Date of retrieval 15December 2018)

In the above example (the figure 13), the function named sub takes the two input from the user and returns the sum of them using the alert function displaying 18 on the browser. (Christensson, P.2014, Date of retrieval 15 December 2018)

4.4 Bootstrap

Bootstrap is a free and open source framework which is used in the creation of webpages and web application. It is used in coordination with HTML, CSS, and JavaScript for the creation of responsive websites and sites. The framework can either be downloaded directly from “getbootstrap.com” or be introduced to code as a CDN (Content Delivery Network). (WhatIs.com Bootstrap, Date of retrieval 20 December 2018)

4.5 Vuetify

Vuetify is an easy and reusable semantic component framework used in developing web application by providing a clean and reusable component. It supports all the modern browsers and it is easy to install and use in the app. Their component is based on the Google material design and is specially designed for Vue.js. Laravel out of the box gives us Vue.js. The Vuetify plugin can be injected into the Vue application by installing its npm package which is shown below in the figure 14.

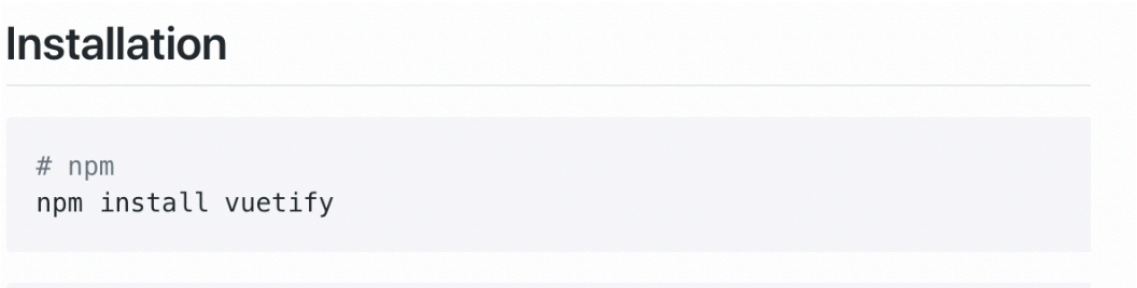


FIGURE 14. Vuetify Installation (Vuetify, Date of retrieval 21 December 2018)

The app.js files along with its other component are already created during the installation of the Laravel project. The installed Vuetify plugin can now be imported into the application by including the following code in app.js and app.scss file as shown in the figure 15.

```
window.Vue = require('vue');
import Vue from 'vue'
import Vuetify from 'vuetify'
```

.

FIGURE 15. Import of Vuetify in the app.js file

```
// Fonts
@import url('https://fonts.googleapis.com/css?family=Roboto:100,300,400,500,700,900|Materi

// Variables
@import 'variables';
// index.js or main.js
@import '~vuetify/dist/vuetify.min.css';
@import '~simplemde/dist/simplemde.min.css';


// Bootstrap
@import '~bootstrap/scss/bootstrap';

.navbar-laravel {
  background-color: #fff;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.04);
}
```

FIGURE 16. Import of Vuetify CSS file into the project from the app.scss file

The figure 16 shows the usage of the predefined layout of Vuetify component with Laravel application. (Vuetify, Date of retrieval 21 December 2018). The imported plugin now needs to be registered with the Vue.js which can be done by including the following code in the same app.js file.

- Vue.use (Vuetify)

Thus, imported Vuetify component can now be injected into the blade files of the Laravel application which is shown below

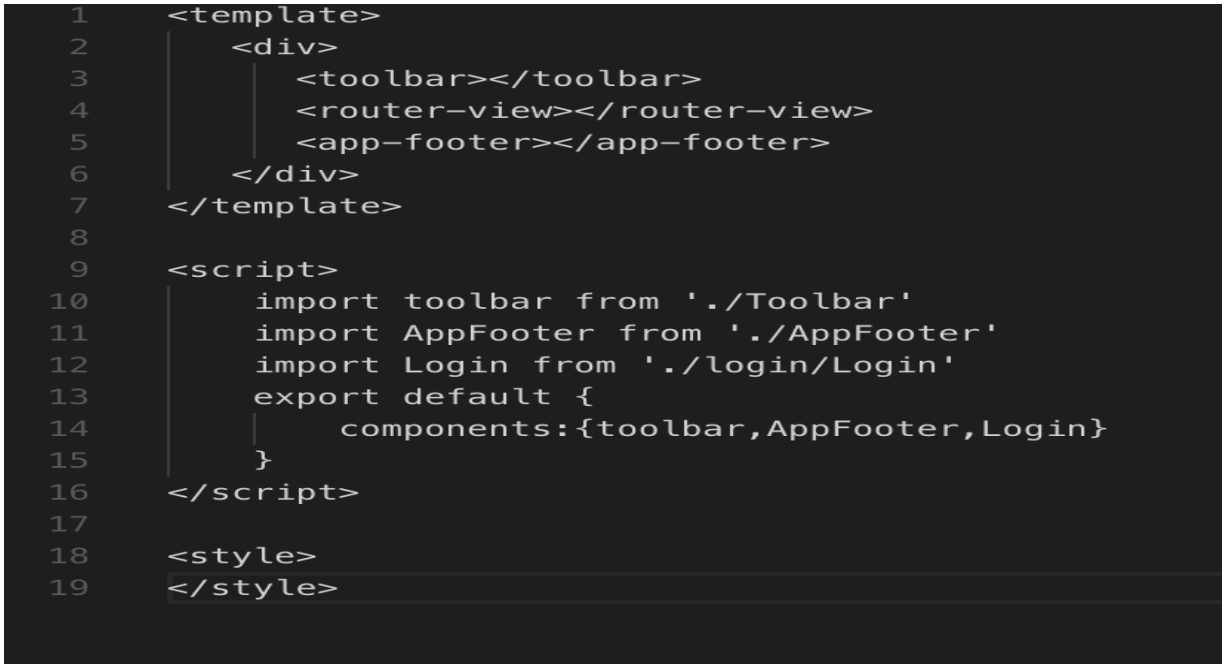
```
<body>
<div id="app">
    <v-app>
        <app-home></app-home>

            </v-app>
</div>
<script src="{{asset('js/app.js')}}"></script>
</body>
</html>
```

FIGURE 17. Blade File

In the above figure 17, v-app is the root component of Vuetify where all other Vuetify components must be nested. Vuetify in its library has its own component and when the component is loaded onto the template, they render HTML which is seen in the web pages. Here the root element with the id of the “app” is created and the data is moved inside the object by creating a new Vue component “<app-home> </app-home>¨. The created Vue component must be registered in the app.js file which is done by adding the following code

- Vue.component(‘AppHome’, require(‘/components/AppHome.vue’));

The following figure 18 shows the AppHome.vue component which is used during the development of the application



```
<template>
    <div>
        <toolbar></toolbar>
        <router-view></router-view>
        <app-footer></app-footer>
    </div>
</template>

<script>
    import toolbar from './Toolbar'
    import AppFooter from './AppFooter'
    import Login from './login/Login'
    export default {
        components:{toolbar,AppFooter,Login}
    }
</script>

<style>
</style>
```

FIGURE 18. AppHome.vue

The above file contains the scaffolding for the Vue component which includes another component as well as the router tag. The different component can be switched in the Vue app by importing the vue-router which is done be installing its npm package as shown below.

- npm install vue-router

Thus, installed vue-router can be used in the application with the tag “<router-view></router-view>” which gives the routes where the view component must be loaded. The router.js file shown below in figure 19 is created where the vue-router is imported and registered. The file also includes the routes constant which defines the path and its component.

```
import Vue from 'vue'
import VueRouter from 'vue-router'
Vue.use(VueRouter)
import Login from '../components/login/Login'
import Singup from '../components/login/signup'
import Forum from '../components/forum/forum'
import Logout from '../components/login/Logout'
import Read from '../components/forum/read'
import Create from '../components/forum/create'
import CreateCategory from '../components/fcategory/CreateCategory'



const routes = [
    { path: '/thesis_new/public/try/flogin', component: Login },
    { path: '/thesis_new/public/try/flogout', component: Logout },
    { path: '/thesis_new/public/try/fsignup', component: Singup },
    { path: '/thesis_new/public/try/fcategory', component: CreateCategory },

    { path: '/thesis_new/public/try/forum', component: Forum , name:'forum'},
    { path: '/thesis_new/public/try/question/:slug', component: Read , name:'read'},
    { path: '/thesis_new/public/try/ask', component: Create},

]
const router = new VueRouter({
    routes, // short for `routes: routes`,
    hashbang : false,
    mode : 'history'
})
export default router
```

FIGURE 19. Router.js file

The other view component is created similarly as the component AppHome. The below picture shows the component AppFooter.vue which is shown below in figure 20.

```
1   <template>
2       <v-footer class="pa-3">
3           VillageDev
4           <v-spacer></v-spacer>
5           <div>&copy; {{ new Date().getFullYear() }}</div>
6       </v-footer>
7   </template>
8   <script>
9       export default {
10      }
11  </script>
12  <style>
13  </style>
```

FIGURE 20. AppFooter.vue component

The above file contains the basic scaffolding of the Vue file along with the Vuetify Components. Thus, all the created Vue components can now be used as a front end for the development of the application along with Laravel.

4.6 Vue

Vue is one of the progressive JavaScript frameworks used for building UI interface. It is one of the versatile, performant and maintainable JavaScript frameworks along with one of the two giant frameworks React and Angular. The Vue component can be injected into any part of the application by simply inserting a script tag and only those components are re-rendered during the change of the state. Below is an example of Vue along with its directive

Figure 1 Vue Example (Vue.js, Date of retrieval 21 December 2018)

In the figure 21, the Vue library is injected in the script source and a new Vue instance is created which is plugged into the root element with the id of #app and data is moved inside of an object. When the app is opened in the browser the

page shows an expression with any change in the value of the data inside the object. For the above example, the browser will display “Boots are in stock. “. (Copes,F..Date of retrieval 25 December 2018)

5 BACKEND

5.1 MAMP localhost

MAMP which stands for "Mac OS X, Apache, MySQL, and PHP." is a free local server environment which is mostly used for local testing and a web development purpose. This free operating server installs the AMP stacks on the computer which consists of the following package

- Apache = It is a web server application which allows the developer to view their application in the local browser without publishing into the external server.
- MySQL = It is a relational database management system which allows any webpage to access data from their system that contains PHP code. Therefore, it helps any developer to develop any dynamic website locally before publishing it.
- PHP= It is the most common server-side programming language which is used to create a webhost on the computer. (Christensson, P. 2013, Date of retrieval 15 December 2018)

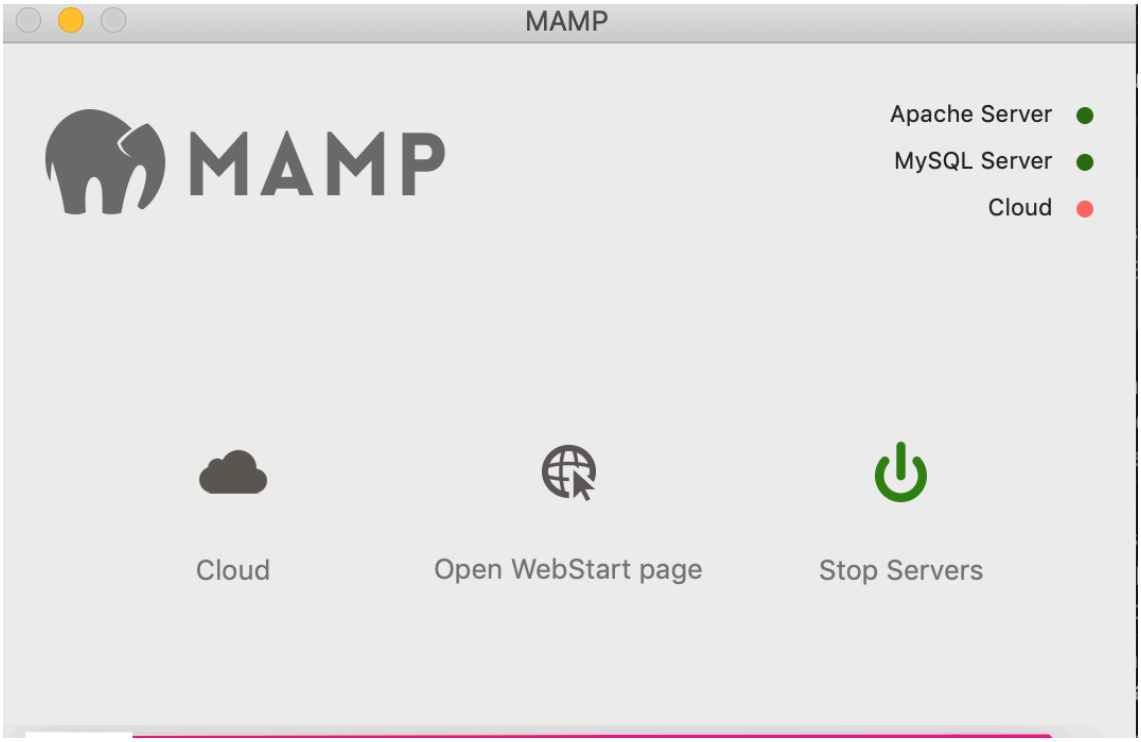


FIGURE 22. Appearance Of MAMP

The figure 22 shows the appearance of MAMP where the server can be started or stopped. When the server is started the default homepage of the MAMP is opened in the browser. It is shown below in the figure 23.

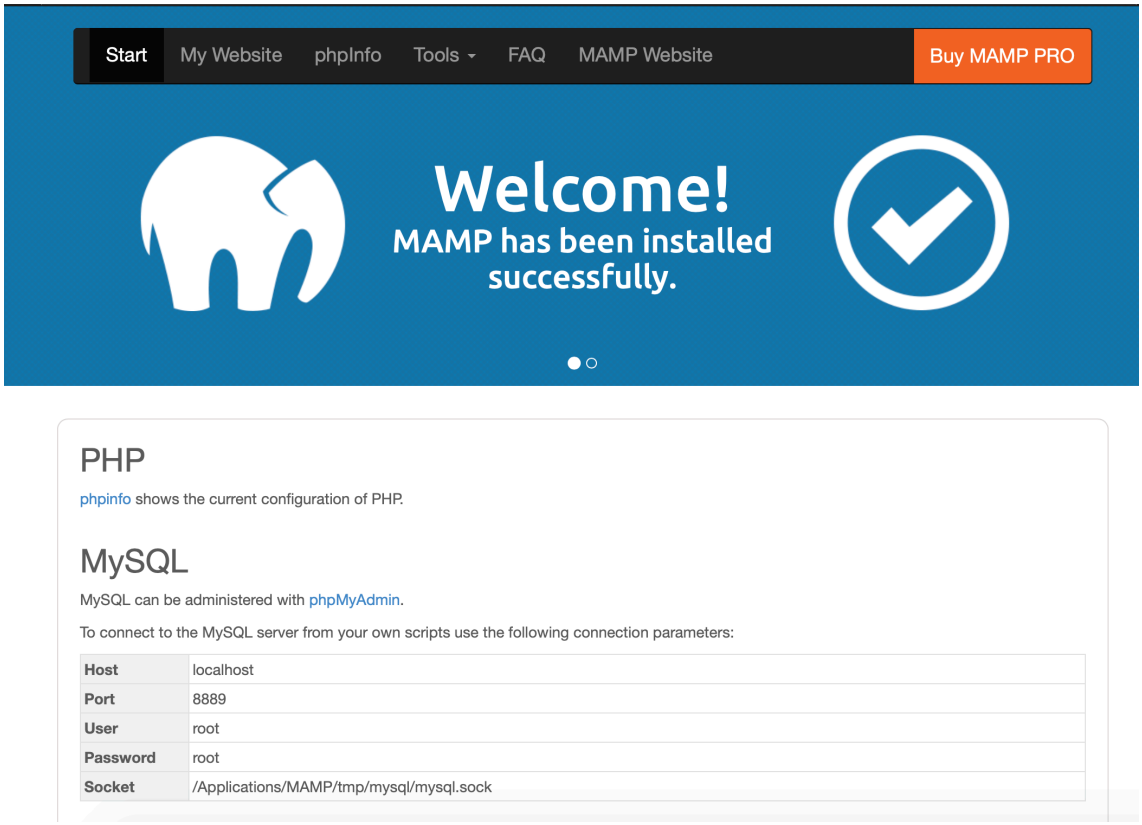


FIGURE 23. Main page of MAMP

The figure 23 is the home page of our local server which includes all the information from the database management system to all the web application saved in the server.

5.2 PHP

PHP stands for “PHP Hypertext Preprocessor” which is the most common server-side programming language for creating and managing dynamic content and its database. PHP Runs on the various OS platform and is also compatible with all the servers of the modern era. PHP files have .php extension and may contain any plain text, HTML or CSS and after being executed on the server it is loaded in the browser as a plain HTML. (w3schools.com, Date of retrieval 20 December 2018).

A PHP script can be written anywhere between the document within the <?PHP and?> tag which is shown below in the picture.

```
<html>

   <head>
      <title>Hello World</title>
   </head>

   <body>
      <?php echo "Hello, World!";?>
   </body>

</html>
```

FIGURE 24. PHP script demo (tutorialspoint, Date of retrieval 13 October 2018)

In the above figure 24, the PHP code is set between the HTML document and produces the result “Hello, World!”.

Some of the common uses of PHP are given below

- Performing system function, such as CRUD (Create, Read, Update, Delete).
- Handling forms from gathering information to save the data into the database.
- Restricting a user based on their privileges.
- Setting cookies variable and encrypting the data.

5.3 Stripe

Stripe is a payment processing platform which allows the apps and website to accept credit card payment. It helps in collecting the card details using their pre-built UI components. The obtained card information is then converted into the token and sent to their server where all the server-side handling is done. There are four steps for injecting this platform into the application which are given below

- Setting up stripe elements = To prevent from mishandling the data the script of the stripe must be loaded on every page. The stripe elements help in creating the instance of an element along with the personal publishable API key.
- Creation of a payment form = The created stripe element is thus placed in the payment form where the element is mounted to the element container after the form has loaded. The figure 25 shows the pre-built payment form of stipe which can be directly loaded into the application.

Figure 25. Stripe Payment Form

- Creation of token for secure transmission of data = The payment detail is collected from the payment form using the elements and then is converted to the token. An event handler is created which sends the data to the stripe server for tokenization.
- Submission of the token = The data is submitted along with any additional information to the stripe server which takes care of all the server-side handling.

5.4 Laravel

Laravel is an open-source MVC based PHP framework which uses a component from another existing framework to design the web application in a structured and rational way. This framework was developed by Taylor Otwell with the intention to reduce the development cost and enhancing the quality of the coding industry. (tutorialspoint, Date of retrieval 20th December 2018).

5.4.1 Server Requirement

Like every framework, Laravel has its own system requirement. The requirement can be met by using the Laravel Homestead virtual machine. The Laravel Homestead virtual machine can be omitted by fulfilling the following server requirement as shown in the figure 26.

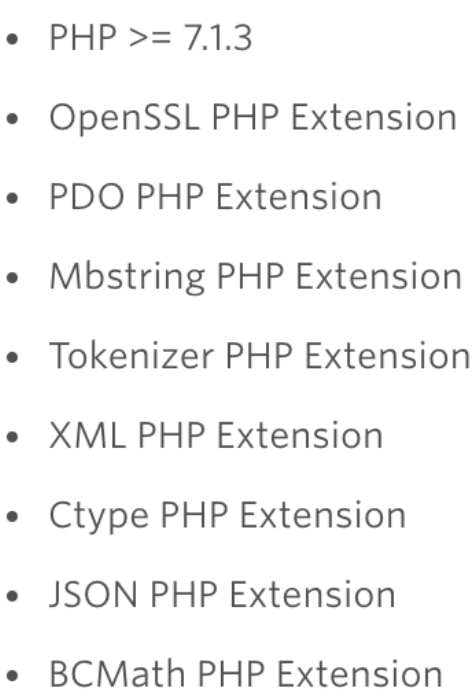


FIGURE 26. Server Requirement (Laravel 5.7 documentation Date of retrieval 20th December 2018)

5.4.2 Installation

Laravel use composer as a dependency management tool. So, a composer must be installed before installing Laravel. After the composer has been installed, it is possible to check it by typing “composer” in the terminal which gives the following screenshot. (See the figure 27 below).

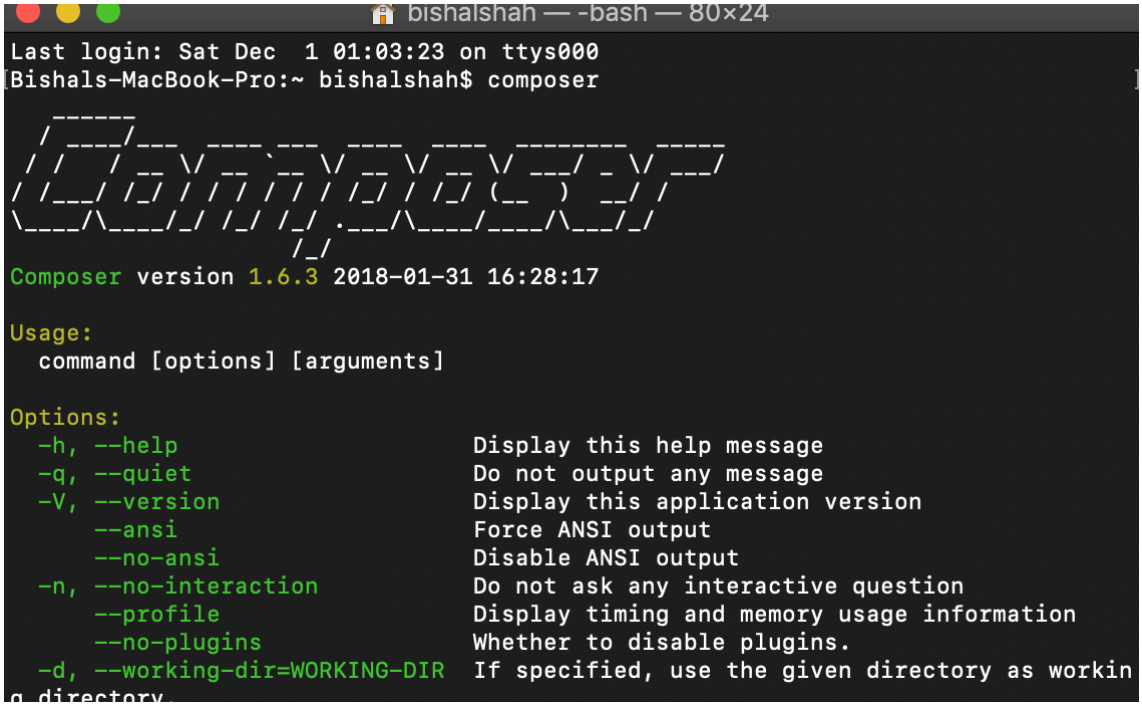


FIGURE 27. Composer installation

Then the Laravel installer can be downloaded using the following command as shown below

- composer global require laravel/installer

Similarly, a Laravel Project can be created and started with the command shown below

- composer create-project -–prefer-dist laravel/laravel blog
- php artisan serve

5.4.3 Application Structure

The figure 28 shows all the folders and subfolders of the Laravel application.

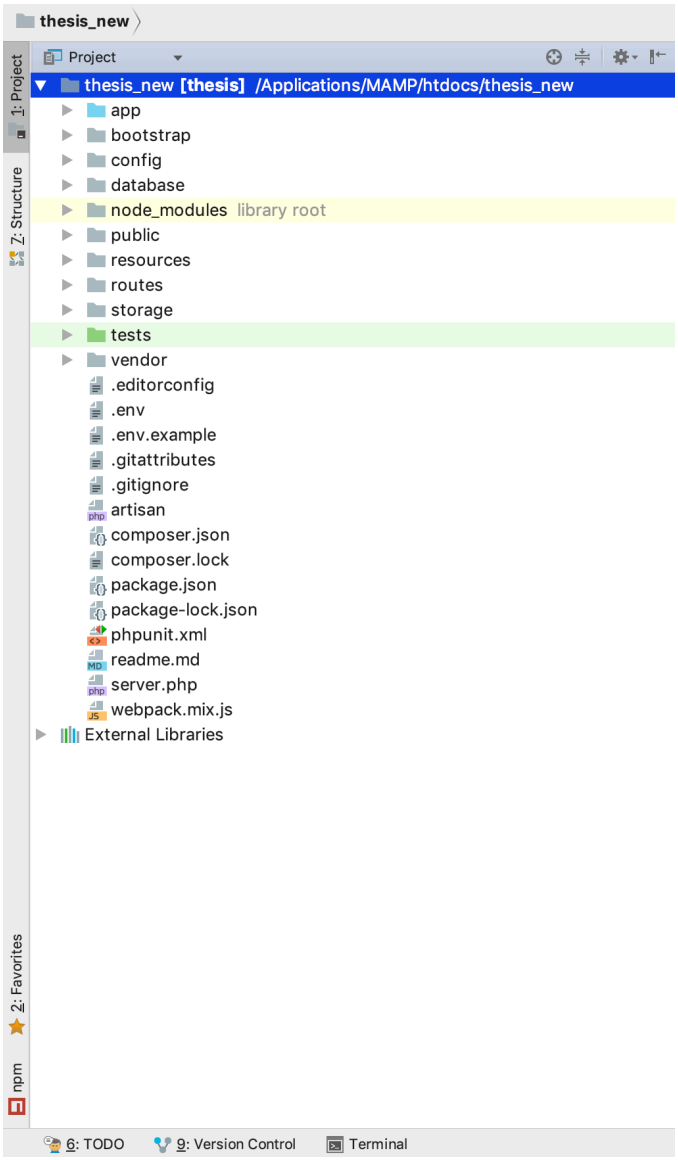


FIGURE 28. Application Structure

- app directory is the core of the application which includes all the event handler, exception handler, controllers, models, policies, providers and console which includes all the necessary artisan command.
- Bootstrap folder encloses all the files necessary for web caching and initializing the script necessary for Bootstrap.
- config folder contains all the configuration files for the smooth running of the application.
- database folder includes all the seeding, factory and migration parameter for database functionality.

- public folder includes all files and folders required for the configuration and initializing of the web application.
- resource folder contains all the blade files for Front end development as well as also other folders for localization and styling the web application.
- storage folder contains all the files which store the session, cache and error logs while the application is running
- tests folder contains all the automated test.
- vendor folder contains all the composer dependency.

5.4.4 Routes

The routes for the web application are registered in the web.php file for handling all the requests from the normal web application and must be registered to api.php for handling the request from a mobile app. The routes are registered in the api.php file prefixes with the route 'api' and use api middleware that provides an extra layer of security for handling all the ajax requests. The basic method for defining the routes is shown in the figure 29.

FIGURE 29. Defining Routes

The other available router method is shown in the figure 30.

```
Route::get($uri, $callback);
Route::post($uri, $callback);
Route::put($uri, $callback);
Route::patch($uri, $callback);
Route::delete($uri, $callback);
Route::options($uri, $callback);
```

FIGURE 30. Routes Method (Laravel routing, Date of retrieval 8 September 2018)

5.4.5 Middleware

Middleware acts as a bridge between the client request and the backend data. Laravel uses the middleware to check if the user has logged into the system and redirects the user based on their authentication. The application used for this project also includes a middleware named admin to differentiate the user based on their admin permission. The admin middleware is created and is thus registered in the kernel file. The admin middleware thus created contains the following function that is passed to the custom argument.

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Auth;

class Admin
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if(!Auth::user()->admin)
        {
            return redirect()->back();
        }


        return $next($request);
    }
}
```

FIGURE 31. Admin.php file

In the figure 31 a closure is used. It is an anonymous function and can be used as a callback method or even as a parameter in the function. The Closure provides the user to pass their own custom logic to the actual function. Closure function is started by adding the handler function to the closure parameter. Then the handle function is called where the function is passed as a first parameter. Here in the figure 31, the term $next is used as a callback with the $request to pass the request into the application. If the authenticated user is not an admin, the middleware will return an HTTP redirect to the user. Otherwise, the request is passed into the application.

5.4.6 Controller

The root directory for the controller is “app/Http/Controllers” and all newly created controller in Laravel extends the base controller so that the request can be handled in the same class. The controller acts as the traffic between the Models and the Views. The Controller can be created by passing an artisan command as shown in the figure 32

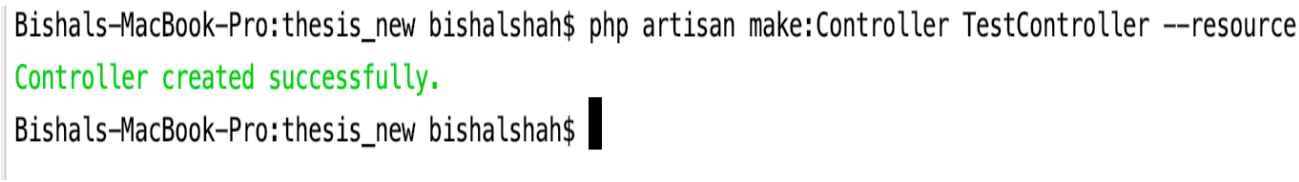


FIGURE 32. Creating a controller

Some of the Controller used during the development of this application are given below

5.4.6.1 Authentication Controller

This consists of a group of controllers which includes Login, Register and Forget Password Controller. The following figure 33 shows the Register controller

where the Profile of the user is created when a user registers for the application.

```
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
       $user=  User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
        ]);
        Profile::create([
            'user_id'=>$user->id

        ]);
        return $user;
    }
}
```

FIGURE 33. Register Controller

5.4.6.2 Donate Controller

```
class DonateController extends Controller
{
    public function storePayment(Request $request)
    {
        $this->validate($request,[
            'email'=>'required|email',
            'name' => 'min:3'


        ]);
        $data = array(
            'email' => $request->email,
            'name' => $request->name
        );



        // Set your secret key: remember to change this to your live secret key in production
// See your keys here: https://dashboard.stripe.com/account/apikeys
        \Stripe\Stripe::setApiKey( apiKey: "sk_test_Tz2zoQSqOfQp27Qk0aMOaa4X");

// Token is created using Checkout or Elements!
// Get the payment token ID submitted by the form:
        $token = $_POST['stripeToken'];
        $charge = \Stripe\Charge::create([
            'amount' => 100,
            'currency' => 'usd',
            'description' => 'bishal',
            'source' => $token,
        ]);
        Mail::send( view: 'emails.thankdonate', $data, function($message) use ($data){
            $message->to($data['email']);
            $message->from('great.shah222@gmail.com');
        });
        Mail::send( view: 'emails.admindonate', $data, function($message) use ($data){
            $message->to('t5shbi00@students.oamk.fi');
            $message->from('great.shah222@gmail.com');
        });

        return redirect()->route( route: 'name')  ;  }
```

FIGURE 34. Donate Controller

This controller shown in the figure 34 is responsible for handling the payment by using Stripe payment system. Beside this, it also sends emails to both the user and the admin notifying them about the successful donation.

5.4.6.3 Front Controller

```
class FrontController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $photos=Photo::orderByRaw('RAND()')->limit(9)->get();
        $personals=Personal::orderBy('id','desc')->limit(4)->get();


        return view( view: 'front.home')->withPhotos($photos)->with('settings',Setting::first())->withPersonals($personals);
    }


    public function postContactme(Request $request) {
        $this->validate($request, [
            'email' => 'required|email',
            'subject' => 'min:3',
            'name'=>'required',
            'message' => 'min:3']);

        $data = array(
            'email' => $request->email,
            'subject' => $request->subject,
            'name'=> $request->name,
            'bodyMessage' => $request->message
        );

        Mail::send( view: 'emails.contact', $data, function($message) use ($data){
            $message->from($data['email']);
            $message->to('great.shah222@gmail.com');
            $message->subject($data['subject']);
        });
        Mail::send( view: 'emails.admincontact', $data, function($message) use ($data){
            $message->to($data['email']);
            $message->from('great.shah222@gmail.com');
            $message->subject('Thankyou for contacting');
        });

        return redirect()->back();
    }
}
```

FIGURE 35. Front Controller

This controller shown in the figure 35 renders the front-end view of the application and also handles the Contact us section as shown below in the figure 36.

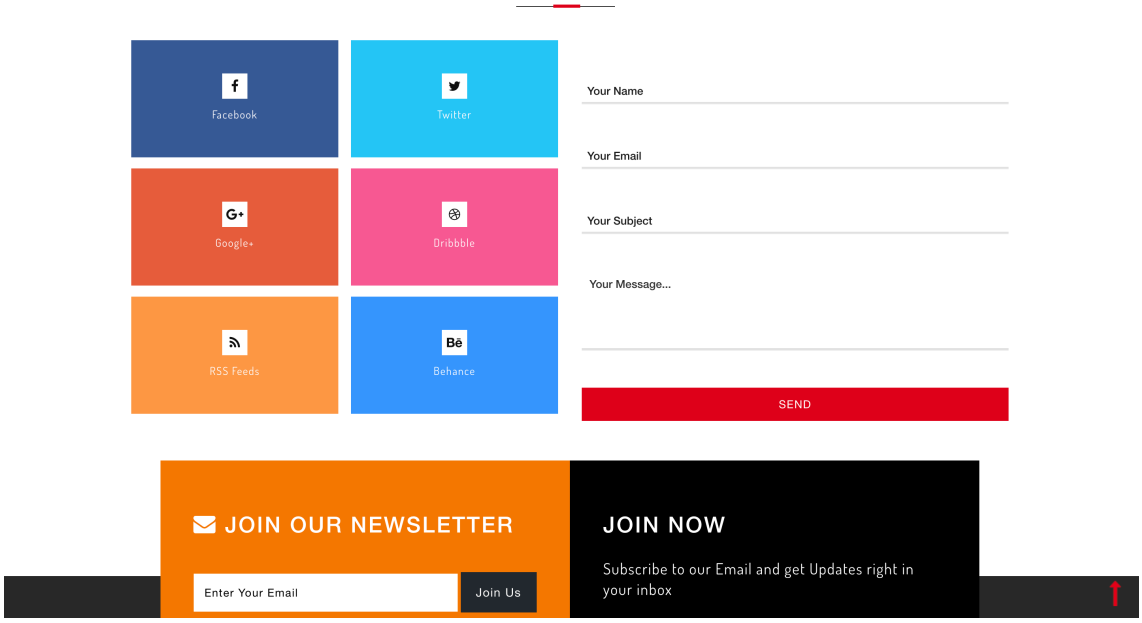


FIGURE 36. Contact Section

After the successful validation of the form by the controller, the message from the user is sent to the admin. The user also gets the default follow-up message as written in the blade files.

5.4.7 Views

The root directory for the views is “resource/views” which stores all the template file required for serving the application. Views act as a mediator between the application and the presentation logic. One of the blade files used in this application is described below

5.4.7.1 main.blade.php

This blade file is the home page of the application which is rendering the data as well as extending its layout to the child element. The following figure 37 shows that the layout is extended to the child element ‘”value” which is later then injected in the other template file as shown in the figure 38.

```
</head>
<body>
@include('_includes.navbar.head')

@yield('value')

@include('_includes.navbar.footer')

<!-- header -->
```

FIGURE 37. View File

```
@extends('layout.main')

@section('value')
<!-- banner -->
<div class="banner-w3ls">
    <div class="container">

        <h2 class="wthree-tittle">We Need Your Help</h2>
        <p class="subtitle-agileinfo">Give a little. Change a lot.</p>
        <div class="banner-info">
            <div  id="top" class="callbacks_container">
                <ul class="rslides" id="slider3">
                    <li>
                        <div class="col-md-5 ban-left">
                            <img src="{{asset('main/images/v1.jpg')}}" alt=" "/>
                        </div>
```

FIGURE 38. Template File

This template file also consists of all the CSS files, JS files, fonts and the stripe element necessary for the smooth running of the application which is shown below in the figure 39, 40 and 41.

```
<script type="application/x-javascript"> addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false);
    function hideURLbar(){ window.scrollTo(0,1); } </script>
<!-- //for-mobile-apps -->
<link href="{{asset('main/css/bootstrap.css')}}" rel="stylesheet" type="text/css" media="all" />

<link href="{{asset('main/css/lsb.css')}}" rel="stylesheet" type="text/css">
<link href="{{asset('main/css/style.css')}}" rel="stylesheet" type="text/css" media="all" />
<link href="{{asset('main/css/font-awesome.css')}}" rel="stylesheet">
<link href="//fonts.googleapis.com/css?family=Noto+Sans:400,400i,700,700i" rel="stylesheet">
<link href='//fonts.googleapis.com/css?family=Lato:400,100,100italic,300,300italic,400italic,700,700italic,900,900italic'
<script src="https://js.stripe.com/v3/"></script>
```

FIGURE 39. CSS files and Fonts

```
<script src="{{asset('main/js/responsiveslides.min.js')}}"></script>
<script>
    // You can also use "$(window).load(function() {"
    $(function () {
        // Slideshow 4
        $("#slider3").responsiveSlides({
            auto: true,
            pager: false,
            nav: true,
            speed: 500,
            namespace: "callbacks",
            before: function () {
                $('.events').append("<li>before event fired.</li>");
            },
            after: function () {
                $('.events').append("<li>after event fired.</li>");
            }
        });
    });
</script>

<script type="text/javascript">
```

FIGURE 40. JavaScript Files

```
<script>
    var stripe = Stripe('pk_test_FZz1641WpWy8vHCcYoxhtmVL');

    // Create an instance of Elements.
    var elements = stripe.elements();

    // Custom styling can be passed to options when creating an Element.
    // (Note that this demo uses a wider set of styles than the guide below.)
    var style = {
        base: {
            color: '#32325d',
            lineHeight: '18px',
            fontFamily: '\'"Josefin Sans", sans-serif\'',
            fontSmoothing: 'antialiased',
            fontSize: '16px',
            '::placeholder': {
                color: '#aab7c4'
            }
        },
        invalid: {
            color: '#fa755a',
            iconColor: '#fa755a'
        }
    };

    // Create an instance of the card Element.
    var card = elements.create('card',
        {style: style,
            hidePostalCode: true});

    // Add an instance of the card Element into the `card-element` <div>.
    card.mount('#card-element');

    // Handle real-time validation errors from the card Element.
    card.addEventListener('change', function(event) {
        var displayError = document.getElementById('card-errors');
        if (event.error) {
            displayError.textContent = event.error.message;
        } else {
            displayError.textContent = '';
        }
    });

    // Handle form submission.
    var form = document.getElementById('payment-form');
    form.addEventListener('submit', function(event) {
```

FIGURE 41. Stripe Element

5.4.8 Model

The typical location for the Model is the app directory but it can be placed anywhere which can be auto-loaded by the file composer.json. The model manages the data of an application and is used to interact with the database. The simplest way to create a model is shown in the figure 42.

```
1 php artisan make:model <model name>
```

FIGURE 42. Model Creation

Migration along with the model can be created by using the following command. (See the figure 43 below).

```
php artisan make:model Flight --migration

php artisan make:model Flight -m
```

FIGURE 43. Migration Creation

One of the Models used in the application is shown below in the figure 44.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;
use App\Comment;

class Post extends Model
{
    use SoftDeletes;


    protected $dates =['deleted_at'];



    public function category()
    {
        return $this->belongsTo('App\Category');
    }

    public function tags()
    {
        return $this->belongsToMany('App\Tag');
    }
    public function author()
    {
        return $this->belongsTo('App\User');
    }
    public function comments()
    {
        return $this->hasMany('App\Comment');
    }
}
```

FIGURE 44. Post Model

Here the $dates property represents the mass assignment attribute for this model which is used to alter their property in the database. The property softDeletes allow model from not being deleted from the database. Instead, their value is set to deleted_at attribute in the database. The above figure 44 shows the model along with its attribute and relation to other models. For instance, the category is linked with the post model whereas one post may have many comments.

5.4.9 Database Migration

Migration is easy to use and is like the version control system for the database. They allow to modify and manipulate the content of the project database with the help of their Laravel Schema builder. This project uses the MYSQL database which consists of following tables as shown in the figure 45.

FIGURE 45. Database Content

Laravel along with its artisan command “migration” allows a simpler way to create a migration instead of creating the SQL code manually for all the tables for

the project. The migration for the profile is created by executing the following statement as in the figure 46.

```
Bishals-MacBook-Pro:~ bishalshah$ $ php artisan make:migration create_profiles_table.php
```

FIGURE 46. Migration for Profile Table

After the successful execution of the command, a new *_create_profiles_table.php file can be found in the database/migrations directory which is edited to have the profiles table schema shown below in the figure 47.

```php
class CreateProfilesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('profiles', function (Blueprint $table) {
            $table->increments('id');
            $table->string('avatar')->default('galleryimages/2.png');
            $table->integer('user_id');
            $table->text('about')->nullable();
            $table->string('facebook')->nullable();
            $table->string('youtube')->nullable();


            $table->timestamps();
        });
    }

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

FIGURE 47. Profiles table Schema

The database needs to be connected to the Laravel application before the migration can be run. All the necessary setup for the database can be done in the .env file which is shown below in the figure 48.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=8889
DB_DATABASE=thesis
DB_USERNAME=Bishal
DB_PASSWORD=Bishal
```

FIGURE 48. Setup in .env file

After the database setup, the final step for the migration can be done by running the following command as shown in the figure 49.

```
Bishals-MacBook-Pro:~ bishalshah$ $ php artisan migrate
```

FIGURE 49. Migrating files using artisan command

5.4.10 Laravel WebSocket

Laravel WebSocket can be used as a replacement for Pusher. This WebSocket has a similar feature like the Pusher socket but it can be implemented in any application without any charges. Besides this, with the help of this socket, it is possible to debug all the incoming and outgoing WebSocket request in the debugging dashboard. This package can be simply injected into the application by installing the Pusher PHP SDK which is given below in the figure 50 and then using the Pusher as a broadcasting driver in the .env file.

```sh
composer require pusher/pusher-php-server "~3.0"
```

FIGURE 50. Pusher PHP SDK

Thus, installed Laravel WebSocket can be now integrated with the frontend application and receive the broadcasted event with the help of the package Laravel Echo.

6 WEBSITE

6.1 Header and Footer

This tool allows the user to access the content from different pages on the website. The navigation shown in the figure 51 also allows the user to connect with different social accounts, join the monthly newsletter and log into the website. Similarly, the figure 52 shows the footer of the website.

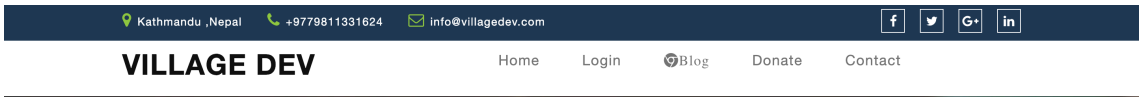


FIGURE 51. Header of the Website

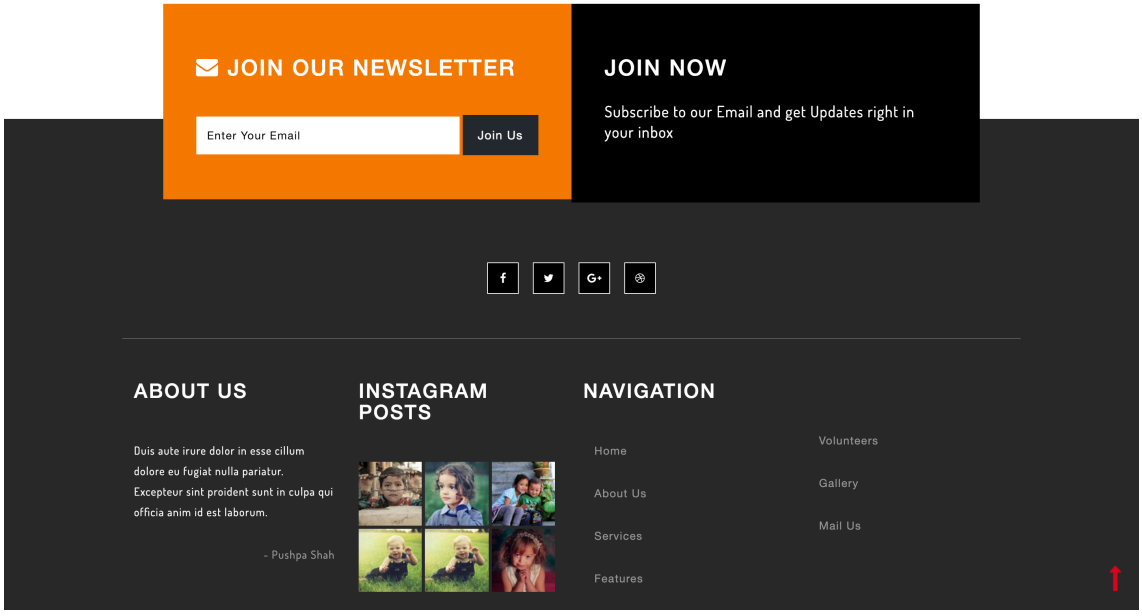


FIGURE 52. Footer of the Website

6.2 Contact form

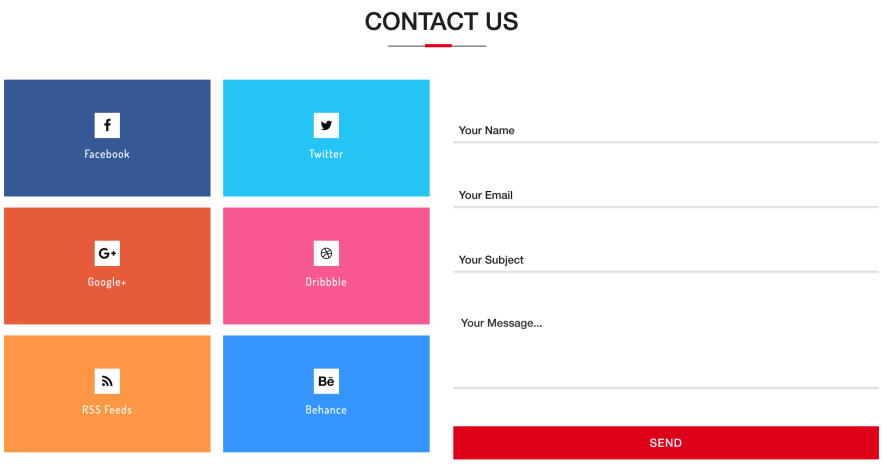


FIGURE 53. Contact form

This feature shown in the figure 53 enables any user to give feedback or ask any question. The user has to fill in their details so that the organization can reply to their query. Both the admin and the user will be notified by email about the feedback and query. The below figure 54 shows the partial code used during the development of this contact form.

```
<!-- Contact-form -->
<div class="w3layouts_mail_grid_right">
    <div class="container">
        <form action="#" method="post">
            <div class="col-md-6 col-xs-6 wthree_contact_left_grid">
                <input type="text" name="Name" placeholder="Name" required="">
                <input type="email" name="Email" placeholder="Email" required="">
            </div>
            <div class="col-md-6 col-xs-6 wthree_contact_left_grid">
                <input type="text" name="Telephone" placeholder="Telephone" required="">
                <input type="text" name="Subject" placeholder="Subject" required="">
            </div>
            <div class="clearfix"> </div>
            <textarea name="Message" placeholder="Message..." required=""></textarea>
            <input type="submit" value="Submit">
            <input type="reset" value="Clear">
        </form>
    </div>
</div>
```

FIGURE 54. View File for Contact form

6.3 BLOG

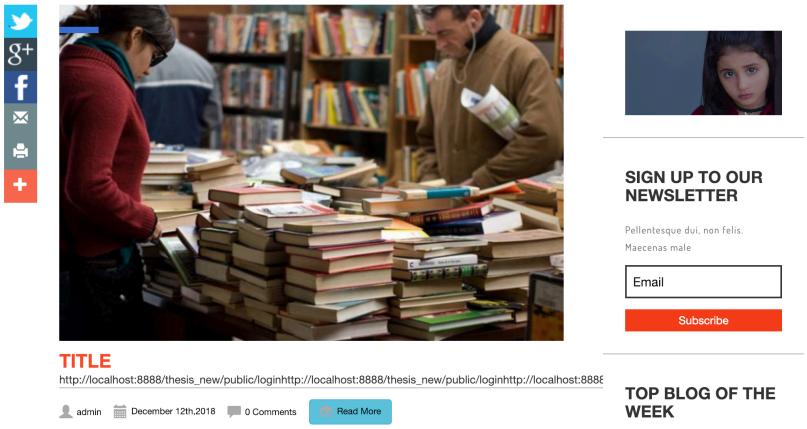


FIGURE 55. Blog Page

The figure 55 illustrates the model for a blog post where the user can read the blog as well as leave a comment to the specific blog post by giving their information. The below figure 56 shows some part of the code used during the development of this blog

```
<div class="technology">

    <div class="container">
    <div class="col-md-7  technology-left">

        <div class="tech-no">

        @foreach($posts as $post)

            <!-- technology-top -->
            <div class="soci">
                <ul>
                    <li><a href="#" class="facebook-1"> </a></li>
                    <li><a href="#" class="facebook-1 twitter"> </a></li>
                    <li><a href="#" class="facebook-1 chrome"> </a></li>
                    <li><a href="#"><i class="glyphicon glyphicon-envelope"> </i></a></li>
                    <li><a href="#"><i class="glyphicon glyphicon-print"> </i></a></li>
                    <li><a href="#"><i class="glyphicon glyphicon-plus"> </i></a></li>
                </ul>
            </div>

            <div class="tc-ch">

                <div class="tch-img">
                    <a href="{{asset('postimages/'.$post->featured)}}"><img src="{{asset('postimages/'.$post->featured)}}" class="img-responsive" alt="djsb"/></a>
                </div>
                <a class="blog blue" href="#"></a>
                <h3><a href="{{url('blog/'.$post->slug)}}">{{$post->title}}</a></h3>
                {{substr(strip_tags($post->body) ,0,200)}} {{strlen(strip_tags($post->body) ) > 200 ? " ..." : " "}}
                <div class="blog-poast-info">
                    <ul>
                        <li><i class="glyphicon glyphicon-user"> </i><a class="admin" href="#">admin</a></li>

                        <li><i class="glyphicon glyphicon-calendar"> </i>{{ date('F nS,Y',strtotime($post->created_at))  }}</li>
                        <li><i class="glyphicon glyphicon-comment"> </i><a class="p-blog" href="#">{{$post->comments()->count()}} Comments </a></li>

                     <a href="{{url('blog/'.$post->slug)}}"> <li class="btn btn-info btn-group-sm"><i class="glyphicon glyphicon-eye-open"> </i>Read More</li></a>
                    </ul>
                </div>
            </div>
            @endforeach
```

FIGURE 56. Code for Blog Page

6.4 FORUM

FIGURE 57. Forum Page

This feature shown in the figure 57 enables logged in user to ask any question by filling all the required details. The user can also create a category in this section to elaborate on their topic. The user can reply to the other question and upvote their answer. Vuetify along with Vue JS has been used for the development of this section and some parts of the development are shown below in the figure 58.

```
<template>
    <v-container fluid grid-list-md>
        <v-layout row wrap>
            <v-flex xs8>
                <question
                        v-for="question in questions"
                        :key="question.path"
                        :data=question
                ></question>
            </v-flex>
            <v-flex xs4>
                <app-sidebar></app-sidebar>
            </v-flex>        </v-layout>
    </v-container>
</template>
<script>
    import AppSidebar from './AppSidebar'

    import question from './question'
    export default {
        data(){
            return {
                questions:{}
            }
        },
        components:{question,AppSidebar},
        created(){
            axios.get('/thesis_new/public/api/question/')
                .then(res => this.questions = res.data.data)
        .catch(error => console.log(error.response.data))
        }
    }
</script>
```

FIGURE 58. Code for Forum Page

7 CONCLUSION

In this growing technology, a user-friendly website with an enhanced user experience is very important for bringing high volume traffic. Blogs and the forum are one of the powerful ways for any individual or organization to convey their views and share their experience. The main objective of this project was to develop a responsive and client friendly website along with blog, forum and a donation platform.

The project gave me an opportunity to learn new skills and work with a real environment. The biggest challenge during this project was designing and implementing both the front-end and back-end development. Since I had less knowledge about handling the PHP framework, I had a problem in most of the part and it took me much more time than anticipated to complete the work. With the help of my teachers and the Laravel Community forum, I successfully managed to overcome all the problem.

In sum, the website is still under the testing phase before it is launched. This project work shows how to plan, design and implement a PHP framework-based website.

8 REFERENCES

Christensson, P. 2013, TechTerms, Date of retrieval 15 December 2018, https://techterms.com/definition/mamp

Christensson, P. 2014, TechTerms, Date of retrieval 15 December 2018, https://techterms.com/definition/javascript

Copes, F. The Vue Handbook: a thorough introduction to Vue.js, Date of retrieval 25 December 2018, https://medium.freecodecamp.org/the-vue-handbook-a-thorough-introduction-to-vue-js-1e86835d8446

Crettenand, G. 2017,Functional PHP,Date of retrieval 1 November 2018, https://proquest-safaribooksonline-com.ezp.oamk.fi:2047/book/programming/php/9781785880322/immutability/ch02lvl2sec18_html?query=((laravel+5))+AND+(PUBDATEYEAR%3d2017)

Oneextrapixel, An Overview of PHP Framework Guides for Developers, Date of retrieval 1 November 2018, https://onextrapixel.com/an-overview-of-php-framework-guides-for-developers/

The Server Side CSS, Date of retrieval 13 October 2018, https://www.theserverside.com/definition/cascading-style-sheet-CSS

The Server Side HTML, Date of retrieval 13 October 2018, https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language

tutorials point PHP-Introduction, Date of retrieval 13 October 2018, https://www.tutorialspoint.com/php/php_introduction.htm

tutorialspoint, Date of retrieval 20 December 2018, https://www.tutorialspoint.com/laravel/laravel_overview.htm

Vue.js Date of retrieval 21 December 2018, https://vuejs.org/

Vuetify vuetifyjs/vuetify, Date of retrieval 21 December 2018, https://github.com/vuetifyjs/vuetify

WhatIs.com Bootstrap, Date of retrieval 20 December 2018, https://whatis.tech-target.com/definition/bootstrap

W3schools HTML, Date of retrieval 9 October 2018,https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default,

w3schools.com PHP 5 Introduction, Date of retrieval 20 December 2018, https://www.w3schools.com/php/php_intro.asp

Zaki, E. 2017, Laravel, Date of retrieval 1 November 2018, https://www.linkedin.com/pulse/why-laravel-best-php-framework-2017-emad-zaki/