UNIVERSITI TUNKU ABDUL RAHMAN

Lee Kong Chian Faculty of Engineering and Science

Bachelor of Science (Honours) Software Engineering

Academic Year 2023 /2024

UECS3294 Advanced Web Application Development

No.	Name	Student ID	Email
M1	E Zhong Lin	2001777	zonglin3634@1utar.my
M2	Chia Jia Shin	2102373	jiaxinnc@1utar,my
M3	Lee Boon Hao	2106860	bhao2002@1utar.my
M4	Choong Kah Yang	2005641	alvinkahyang@1utar.my

Table of Contents

Work Distribution 3

1. Part 1 4

1.1 Web Application Framework 4

1.1.1 Comparison of five popular web development frameworks 4

1.1.2 Key Capabilities of a modern web application 5

1.1.3 Personal Preferences and Perspectives 6

1.2 Software Design Pattern 6

1.2.1 Description of Five Software Design Patterns 6

1.2.2 Benefits and Drawbacks of using MVC architecture 7

1.2.3 Practical Scenarios 7

2.0 Part 2 9

2.1 Title and description of web application 9

2.2 Sample Screen 10

2.3 Code Listing 14

2.3.1 Database Migration 14

2.3.2 Route 16

2.3.3 Controllers 16

2.3.4 Models 17

2.3.5 Authentication Validation 19

2.3.6 Gate Implementation 19

2.3.7 Session Implementation 20

References 21

Work Distribution

	Chia Jia Shin	Choong Kah Yang	E Zhong Lin	Lee Boon Hao
Part 1				
Comparison of different web development frameworks			✓	
Key capabilities of modern web application				✓
Personal preference and perspective				✓
Software Design Pattern	✓			
Benefits and Drawbacks of MVC		✓		
Practical Scenarios		✓		
Part 2				
Database Migration	✓			
Model	✓			
Controller				✓
View	✓	✓	✓	
Route		✓		
Authentication Implementation			✓	
Authorization Implementation				✓
Session Implementation				✓

1. Part 1

1.1 Web Application Framework

1.1.1 Comparison of five popular web development frameworks

	Laravel	Django	Ruby on Rails	Spring Boot	Express.js
Advantages (Clark J., 2024)	• Open-source framework • MVC architecture pattern • Eloquent ORM system and database migration tools	• Fast development since it has ORM that make database management easy • Follow DRY principle, can reuse code.	• Suited with agile development • Good scalability and easy to add new functions. • Built-in testing functions.	• Rapid application development • Auto configuration • Embedded servers • Provides a set of defaults and conventions	• Powerful routing module • Easy integration with third-party libraries • Flexible and customizable
Disadvantages (Clark J., 2024)	• Slow development • Limited built-in support compared to Django and Ruby on Rails	• Steep learning curve • Slower than others • Not the best option to build highly customized web applications	• Steep learning curve • Slower performance than others • Updates cause issues on older versions of Rails	• Steep learning curve • Consume more memory • Limited flexibility	• Security issue • Manage database with third-party libraries • Not good for rich user interface
Suitability for Project Types	Modern and full-stack	Robust and scalable	Fast developmen	Complex, enterprise-	Small and medium web

(Johns R., 2024)	web applications for PHP	data-driven web application	t of high-quality, maintainabl e web applications with Ruby	level applications and microservices in Java	applications and Restful APIs using Node.js
Community Support & Ecosystem	Strong community, extensive packages	Large community, Django packages	Active community, Ruby gems	Large community, Spring ecosystem	Active community, NPM packages
Development Experience	Artisan command, easy to user	Django Admin, Pythonic syntax	Generators, convention-based	Maven/Gradl e build tools, annotations	Minimalistic , Flexible

1.1.2 Key Capabilities of a modern web application

Key capabilities of modern web application:

Cloud-Based and Scalable

- Ex: AWS, Azure, Google Cloud
- Offer unlimited bandwidth and storage through Virtual Machines, means can serve more users
- Improve security
- Access data and applications anytime at anywhere

Modular and Decentralized/Distributed

- Use microservices, breaks down entire applications into seperate smaller components or services
- These decentralized and loosely coupled services can be developed, tested, deployed and maintained independently.

Compatible accross platforms

- Offer consistent, high-quality performance regardless of the screen size, pixel density and others.
- When system has responsive design and interaction, you get a superior user experience.

AI can definitely play a huge role in developing web applications since it can produce personalized design, layout, and content, analyse customer's habits and behaviours, provide recommendations by memorizing customer choices online, and develop a AI-tailored chatbot that can engage with a customer and adapt to the responses with the actions accordingly. (Kvernadze L., 2024).

Generative AI, which can create unique content, has revolutionized many aspects of web application development. It allows automatic generation of code, optimization of user interfaces, and creation of dynamic content that adapts to user needs. The breakthrough from generative AI shows how deeply this technology can change the landscape of web applications, making them smarter and more flexible. (Konarski M., 2024)

1.1.3 Personal Preferences and Perspectives

Based on the research above, we prefer Laravel because security features are easy to set up on any website to improve security and protect sites from hackers. For example, Laravel uses the Bcrypt hashing algorithm, which means it never stores real passwords in the database. Besides that, it is easy for the developers to lay out the logic for user authorization and authentication, since Laravel has its plug-and-play user authentication system.

Furthermore, Laravel supports an MVC (Model-View-Controller) structure that automatically provides a separation between logic and expression syntax. This separation makes it easier for frontend developers to improve the user interface without disrupting development or core functions. Moreover, Laravel integration makes life much easier for backend developers, especially its Artisan line tool. This tool allows you to generate base MVC files and generate custom commands while creating skeleton code, database architecture, and migrations.

1.2 Software Design Pattern

1.2.1 Description of Five Software Design Patterns

Design Pattern	Key Characteristics	Sample Project
Singleton pattern	A class only has one instance and provides a global point of access to it. It involves a private constructor, static method for instance retrieval, and a static instance variable (Shalinda, 2023).	Singleton Pattern
Observer Pattern	Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically (Shalinda, 2023).	Observer Pattern

Strategy Pattern	Define a family of algorithms, encapsulates each algorithm, and makes them interchangeable. It allows the appropriate algorithm at runtime (Shalinda, 2023).	Strategy Pattern
Decorator Pattern	Allow behaviour to be added to individual object dynamically, without affecting the behaviour of other objects from the same class.	Decorator Pattern
Factory Method Pattern	Defines an interface for creating objects but allow subclasses to alter the type of objects created. Provides a way to delegate instantiation logic to subclasses while ensuring clients work with an interface rather than concrete class (Shalinda, 2023).	Factory Method Pattern

1.2.2 Benefits and Drawbacks of using MVC architecture

Benefit (Charles N.C., 2023)	• Enhance maintainability. • Promote reusability. • Allow component to be employed across various sections
Drawback	• Add complexity to small project. • Require time to adapt to this architecture for new developers.
How MVC architecture separates and organizes concern (Charles N.C., 2023)	• Divide application into three interconnected components: Model, View, Controller. • Separate the application logic from business logic

1.2.3 Practical Scenarios

Design Pattern	Practical Scenario
Singleton pattern (Team T.E., 2019)	Manage a centralized configuration object for an application to avoid inconsistency.
Observer Pattern (Team T.E., 2019)	Implement an event-driven system for handling user interactions in GUI, where changes in a model state trigger updates in various UI components

Strategy Pattern (Team T.E., 2019)	Implement different payment methods in an e-commerce application, where each method follows a common interface but has distinct implementations for processing payments.
Decorator Pattern	Enhance input/output stream processing in a file handling system, such as adding buffering or encryption capabilities to a basic file input stream.
Factory Method Pattern (Team T.E., 2019)	Develop a report generation framework supporting various formats, where each report format is implemented as a subclass of a ReportFactory, enabling clients to create reports using different formats.

2.0 Part 2

2.1 Title and description of web application

The title of the web application is Laravel-based news portal system. It is built based on the open-source project with the link of https://github.com/devriazul/news-portal-php. The news portal project that has been created in this link is a PHP project created based on the raw PHP and MySQL. The author of this open-source project is Nirob Hassan who has vast amount of experience in development of web applications using PHP, JavaScript Library's, CMS, API's. He is an expert in transforming the business requirements into technical solutions.

There are three types of users in our user: Admin, Author, and User. The first-time user for the system needs to register an account first. After that, he or she can view all the post and its details. Furthermore, he or she can contact the admin by leaving a message in the 'Contact Us' tab. He or she can also check the status of the message that has been created. The message status will be changed to 'complete' when the admin has updated it. Our system allows the author to create a news post. Besides that, the author can also update the post that he or she has created. Meanwhile, admin in this system can manage category, district, and user table. Besides that, he or she can also delete and update feedback status in the feedback table as well as delete any post. The following table shows the functions of each type of users.

Function	Admin	Author	User
Login/Register			
Login	✓	✓	✓
Register	✓	✓	✓
Category			
Create	✓		
Read All	✓		
Update	✓		
Delete	✓		
District			
Create	✓		

Read All	✓		
Update	✓		
Delete	✓		
Feedback			
Create		✓	✓
View All	✓		
View Own		✓	✓
Update	✓		
Delete	✓		
Post			
Create		✓	
View All	✓	✓	✓
View Own Post		✓	
View Post Details		✓	✓
Update		✓	
Delete	✓		
User			
Create	✓		
Read All	✓		
Delete	✓		
Update	✓		

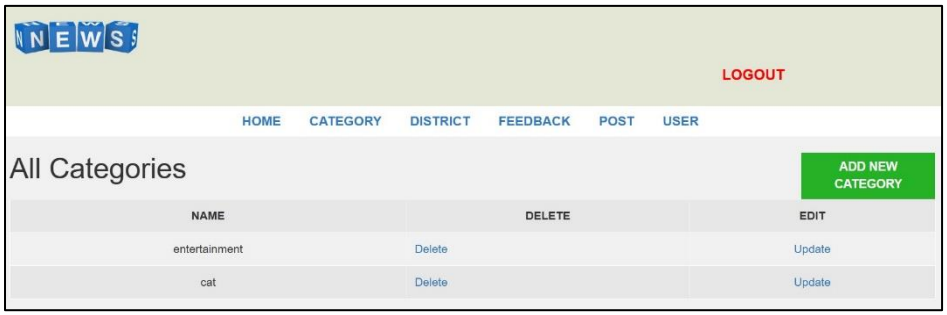
2.2 Sample Screen

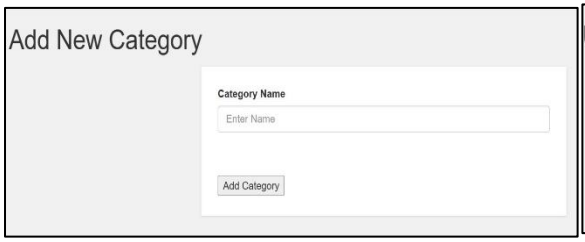
Login/Register

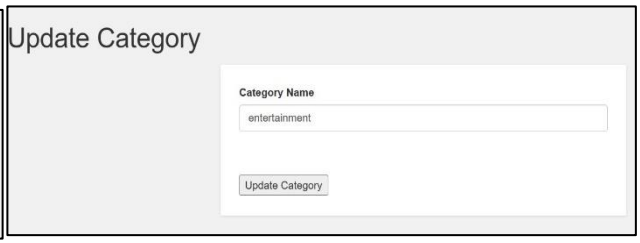

Laravel
Login
Register
Register
Name
E-Mail Address
Password
Confirm Password
Register

Admin


NEWS
LOGOUT
HOME CATEGORY DISTRICT FEEDBACK POST USER
Hello and welcome back, David
Dashboard
You have Admin Access
© Copyright 2024 News Site | Powered by Utar


NEWS
LOGOUT
HOME CATEGORY DISTRICT FEEDBACK POST USER
All Categories
ADD NEW CATEGORY
NAME
DELETE
EDIT
entertainment
Delete
Update
cat
Delete
Update


Add New Category
Category Name
Enter Name
Add Category

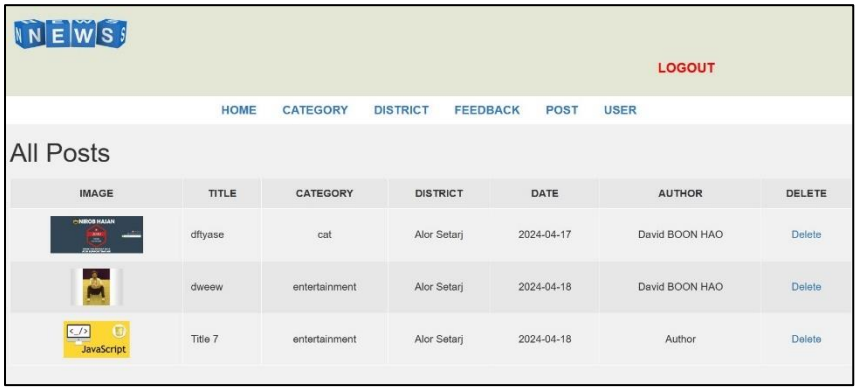

Update Category
Category Name
entertainment
Update Category


NEWS
LOGOUT
HOME CATEGORY DISTRICT FEEDBACK POST USER
All Districts
ADD NEW DISTRICT
Name
Delete
Edit
Malacca City
Delete
Update
Alor Setarj
Delete
Update

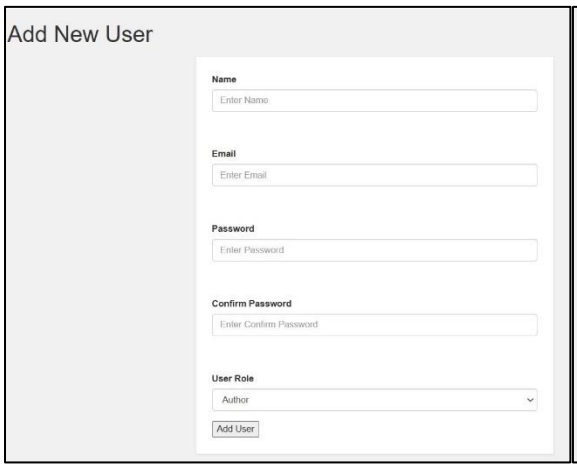

Add New District
District Name
Enter Name
Add District

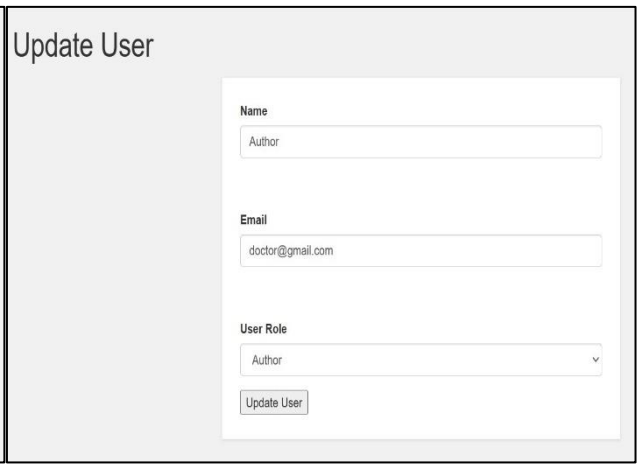

Update District
District Name
Malacca City
Update District


NEWS
LOGOUT
HOME CATEGORY DISTRICT FEEDBACK POST USER
All Feedback
USER EMAIL CONTACT MESSAGE CREATED AT STATUS UPDATED AT DELETE
dwede bhao2002@gmail.com 0182537228 efcvds asdcdd 2024-04-17 14:05:29 Completed 2024-04-18 22:19:46 Delete
David BOON HAO user1@gmail.com 0182537228 dwedw 2024-04-18 02:39:24 Pending Delete

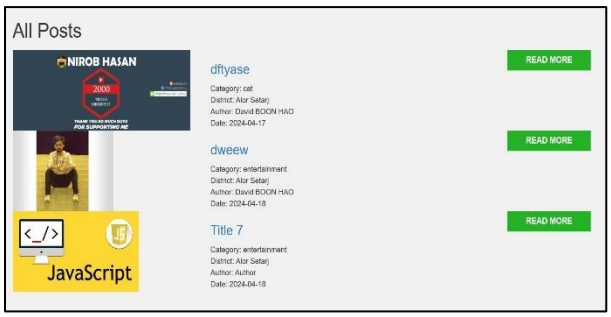

NEWS
LOGOUT
HOME CATEGORY DISTRICT FEEDBACK POST USER
All Posts
IMAGE TITLE CATEGORY DISTRICT DATE AUTHOR DELETE
dftyase cat Alor Setarj 2024-04-17 David BOON HAO Delete
dweew entertainment Alor Setarj 2024-04-18 David BOON HAO Delete
JavaScript Title 7 entertainment Alor Setarj 2024-04-18 Author Delete

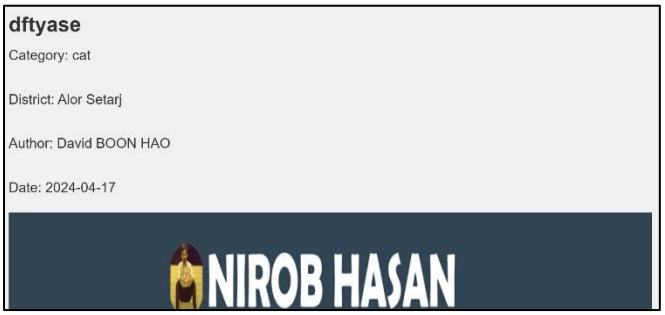

NEWS
LOGOUT
HOME CATEGORY DISTRICT FEEDBACK POST USER
All Users
ADD NEW USER
Name Email Role Delete Edit
Author doctor@gmail.com author Delete Update
dwede bhao2002@gmail.com user Delete Update
David BOON HAO user1@gmail.com author Delete Update
David bhao2002@gmail123.com admin Delete Update

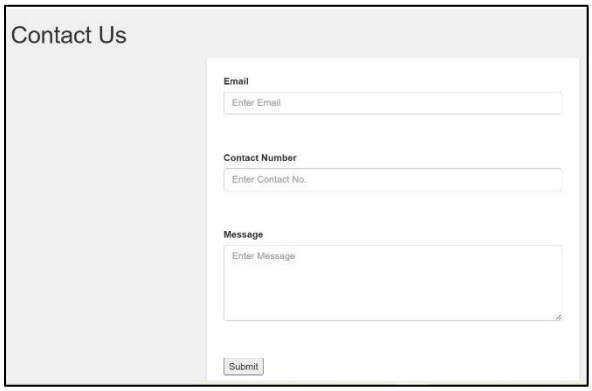

Add New User
Name
Enter Name
Email
Enter Email
Password
Enter Password
Confirm Password
Enter Confirm Password
User Role
Author
Add User

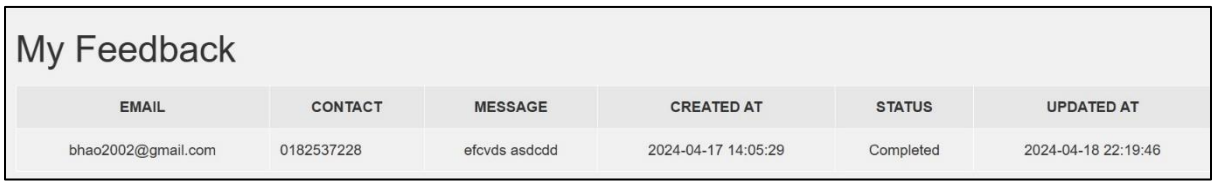

Update User
Name
Author
Email
doctor@gmail.com
User Role
Author
Update User

User





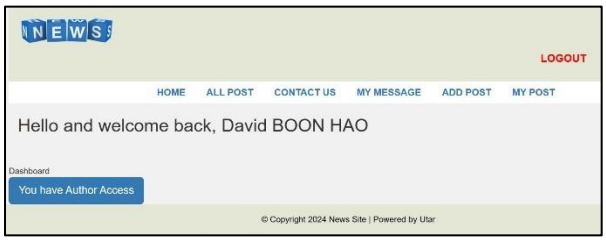


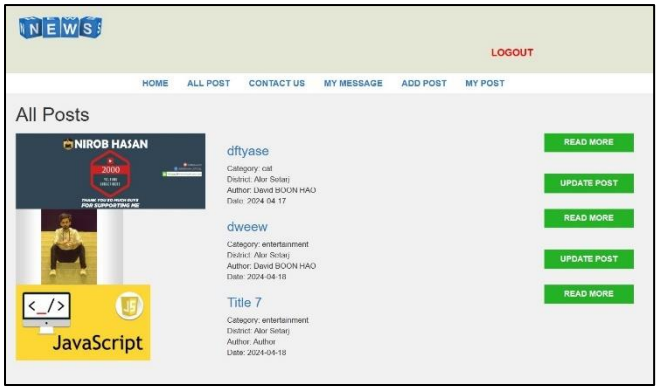


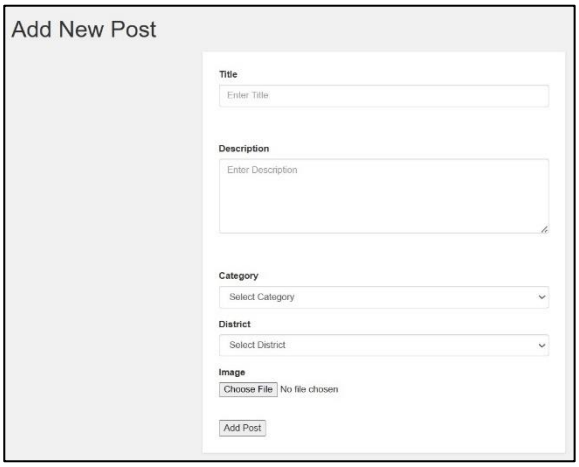
My Feedback

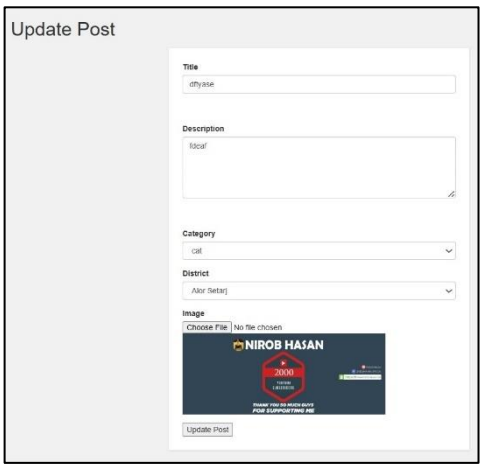
EMAIL	CONTACT	MESSAGE	CREATED AT	STATUS	UPDATED AT
bhao2002@gmail.com	0182537228	efcvds asdcdd	2024-04-17 14:05:29	Completed	2024-04-18 22:19:46

Author









2.3 Code Listing

2.3.1 Database Migration

create_users_table.php

```php
class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

Create_admins_table.php

```php
class CreateAdminsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('admins', function(Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->boolean('is_super')->default(false);
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

Create_authors_table.php

```php
class CreateAuthorsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('authors', function(Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->boolean('is_editor')->default(false);
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

Add_role_columns_to_users_table.php

```php
class AddRoleColumnToUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->enum('role', ['user','author','admin'])->default('user');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
```

Create_categories_table.php

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

Create_districts_table.php

```php
class CreateDistrictsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('districts', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }
```

Create_feedbacks_table.php

```php
class CreateFeedbacksTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('feedbacks', function (Blueprint $table) {
            $table->id();
            $table->string('email');
            $table->string('contact');
            $table->string('message');
            $table->integer('user_id');
            $table->integer('status');
            $table->timestamps();
        });
    }
```

Create_posts_table.php

```php
class CreatePostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->text('description');
            $table->integer('category_id');
            $table->date('post_date');
            $table->integer('user_id');
            $table->integer('district_id');
            $table->string('post_img');
            $table->timestamps();
        });
    }
```

2.3.2 Route

```php
web.php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Auth\LoginController;
use App\Http\Controllers\Auth\RegisterController;
use App\Http\Controllers\CategoryController;
use App\Http\Controllers\DistrictController;
use App\Http\Controllers\FeedbackController;
use App\Http\Controllers\PostController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Gate;

Auth::routes();

Route::get('/login/admin', [LoginController::class, 'showAdminLoginForm']);
Route::get('/login/author', [LoginController::class,'showAuthorLoginForm']);
Route::get('/register/admin', [RegisterController::class,'showAdminRegisterForm']);
Route::get('/register/author', [RegisterController::class,'showAuthorRegisterForm']);

Route::post('/login/admin', [LoginController::class,'adminLogin']);
Route::post('/login/author', [LoginController::class,'authorLogin']);
Route::post('/register/admin', [RegisterController::class,'createAdmin']);
Route::post('/register/author', [RegisterController::class,'createAuthor']);

Route::group(['middleware' => 'auth:author'], function () {
    Route::view('/author', 'author');
});

Route::group(['middleware' => 'auth:admin'], function () {
    Route::view('/admin', 'admin');
});

Route::get('logout', [LoginController::class,'logout']);

Route::get('/', [App\Http\Controllers\HomeController::class, 'index'])->name('home');

//Category Controller
```

```php
//Category Controller
Route::get('/category', [CategoryController::class, 'index'])->middleware('can:isAdmin');
Route::view("/addCategory", "addCategory")->middleware('can:isAdmin');
Route::post("addCategory", [CategoryController::class, 'addCategory'])->middleware('can:isAdmin');
Route::get("deleteCategory/{id}", [CategoryController::class, "deleteCategory"])->middleware('can:isAdmin');
Route::get("updateCategory/{id}", [CategoryController::class, 'showUpdate'])->middleware('can:isAdmin');
Route::post("updateCategory/{id}", [CategoryController::class, 'updateCategory'])->middleware('can:isAdmin');

//User Controller
Route::get('/user', [UserController::class, 'index'])->middleware('can:isAdmin');
Route::view("/addUser", "addUser")->middleware('can:isAdmin');
Route::post("addUser", [UserController::class, 'addUser'])->middleware('can:isAdmin');
Route::get("deleteUser/{id}", [UserController::class, "deleteUser"])->middleware('can:isAdmin');
Route::get("updateUser/{id}", [UserController::class, 'showUser'])->middleware('can:isAdmin');
Route::post("updateUser/{id}", [UserController::class, 'updateUser'])->middleware('can:isAdmin');

//Post Controller
Route::get('/post', [PostController::class, 'index'])->middleware('can:isAdmin');
Route::get('/allPost', [PostController::class, 'allPost'])->middleware('isUserOrAuthor');
Route::get('/singlePost/{id}', [PostController::class, 'singlePost'])->middleware('isUserOrAuthor');
Route::get('/myPost', [PostController::class, 'myPost'])->middleware('can:isAuthor');
Route::get("/addPost", [PostController::class, 'showAddPost'])->middleware('can:isAuthor');
Route::post("addPost", [PostController::class, 'addPost'])->middleware('can:isAuthor');
Route::get("deletePost/{id}", [PostController::class, "deletePost"])->middleware('can:isAdmin');
Route::get("updatePost/{id}", [PostController::class, 'showPost'])->middleware('can:isAuthor');
Route::post("updatePost/{id}", [PostController::class, 'updatePost'])->middleware('can:isAuthor');

//Feedback Controller
Route::get('/feedback', [FeedbackController::class, 'index'])->middleware('can:isAdmin');
Route::get('/viewUserFeedback', [FeedbackController::class, 'viewUserFeedback'])->middleware('isUserOrAuthor');
Route::view("/addFeedback", "addFeedback")->middleware('isUserOrAuthor');
Route::post("addFeedback", [FeedbackController::class, 'addFeedback'])->middleware('isUserOrAuthor');
Route::get("deleteFeedback/{id}", [FeedbackController::class, "deleteFeedback"])->middleware('can:isAdmin');
Route::get("updateFeedback/{id}", [FeedbackController::class, 'updateFeedback'])->middleware('can:isAdmin');

//District Controller
Route::get('/district', [DistrictController::class, 'index'])->middleware('can:isAdmin');
Route::view("/addDistrict", "addDistrict")->middleware('can:isAdmin');
Route::post("addDistrict", [DistrictController::class, 'addDistrict'])->middleware('can:isAdmin');
Route::get("deleteDistrict/{id}", [DistrictController::class, "deleteDistrict"])->middleware('can:isAdmin');
Route::get("updateDistrict/{id}", [DistrictController::class, 'showUpdate'])->middleware('can:isAdmin');
Route::post("updateDistrict/{id}", [DistrictController::class, 'updateDistrict'])->middleware('can:isAdmin');
```

2.3.3 Controllers

CategoryController

```php
<?php

namespace App\Http\Controllers;

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

class CategoryController extends Controller
{
    public function index(){
        $data = Category::all();
        return view("category", ['categories' => $data]);
    }

    public function addCategory(Request $req){
        $req->validate([
            'name' => 'required | unique:categories',
        ]);

        $category = new Category;
        $category->name = $req->name;
        $category->save();
        return redirect("category");
    }

    public function deleteCategory($id){
        $data = Category::find($id);
        $data->delete();
        return redirect("category");
    }

    public function showUpdate($id){
        $data = Category::find($id);
        return view("updateCategory", ['data' => $data]);
    }

    public function UpdateCategory(Request $req){
        $data = Category::find($req->id);
        $req->validate([
            'name' => 'required',
        ]);

        $data->name = $req->name;
        $data->save();
        return redirect("category");
    }
}
```

DistrictController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\District;

class DistrictController extends Controller
{
    public function index(){
        $data = District::all();
        return view("district", ['districts' => $data]);
    }

    public function addDistrict(Request $req){
        $req->validate([
            'name' => 'required | unique:districts',
        ]);

        $district = new District;
        $district->name = $req->name;
        $district->save();
        return redirect("district");
    }

    public function deleteDistrict($id){
        $data = District::find($id);
        $data->delete();
        return redirect("district");
    }

    public function showUpdate($id){
        $data = District::find($id);
        return view("updateDistrict", ['data' => $data]);
    }

    public function UpdateDistrict(Request $req){
        $data = District::find($req->id);
        $req->validate([
            'name' => 'required',
        ]);

        $data->name = $req->name;
        $data->save();
        return redirect("district");
    }
}
```

FeedbackController

UserController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\Feedback;
use App\Models\User;

class FeedbackController extends Controller
{
    public function index(){
        $feedbacks = Feedback::all();
        $users = User::all();
        return view("feedback", ['feedbacks' => $feedbacks, 'users' => $users]);
    }

    public function viewUserFeedback(){
        $id = Auth::id();

        $feedbacks = Feedback::where('user_id', $id)->get();

        return view("viewUserFeedback", ['feedbacks' => $feedbacks]);
    }

    public function addFeedback(Request $req){
        $req->validate([
            'email' => 'required | email',
            'contact' => 'required | min:10',
            'message' => 'required | max:191',
        ]);

        $feedback = new Feedback;
        $feedback->email = $req->email;
        $feedback->contact = $req->contact;
        $feedback->message = $req->message;
        $feedback->status = 0;
        $feedback->user_id = Auth::id();
        $feedback->created_at = date('Y-m-d H:i:s');
        $feedback->save();
        return redirect("viewUserFeedback");
    }

    public function deleteFeedback($id){
        $data = Feedback::find($id);
        $data->delete();
        return redirect("feedback");
    }

    public function updateFeedback($id){
        $data = Feedback::find($id);
        $data->status = 1;
        $data->updated_at = date('Y-m-d H:i:s');
        $data->save();
        return redirect("feedback");
    }
}
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index(){
        $data = User::all();
        return view("user", ['users' => $data]);
    }

    public function addUser(Request $req){
        $req->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => 'required | min:6',
            'confirmPassword' => 'required|same:password',
        ]);

        $user = new User;;
        $user->name = $req->name;
        $user->email = $req->email;
        $user->password = Hash::make($req->password);
        $user->role = $req->role;
        $user->save();
        return redirect("user");
    }

    public function deleteUser($id){
        $data = User::find($id);
        $data->delete();
        return redirect("user");
    }

    public function showUser($id){
        $data = User::find($id);
        return view("updateUser", ['data' => $data]);
    }

    public function UpdateUser(Request $req){
        $data = User::find($req->id);

        $req->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255'],
        ]);

        $data->name = $req->name;
        $data->email = $req->email;
        $data->role = $req->role;
        $data->save();
        return redirect("user");
    }
}
```

PostController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\Category;
use App\Models\District;
use App\Models\Post;
use App\Models\User;

class PostController extends Controller
{
    public function index(){
        $categories = Category::all();
        $districts = District::all();
        $posts = Post::all();
        $users = User::all();
        return view("post", ['posts' => $posts, 'categories' => $categories,
        'users' => $users, 'districts' => $districts]);
    }

    public function allPost(){
        $categories = Category::all();
        $districts = District::all();
        $posts = Post::all();
        $users = User::all();
        return view("allPost", ['posts' => $posts, 'categories' => $categories
        'users' => $users, 'districts' => $districts]);
    }

    public function myPost(){
        $id = Auth::id();
        $categories = Category::all();
        $districts = District::all();
        $posts = Post::where('user_id', $id)->get();
        return view("myPost", ['posts' => $posts, 'categories' => $categories,
        'districts' => $districts]);
    }

    public function singlePost($id){
        $post = Post::find($id);
        $categories = Category::all();
        $districts = District::all();
        $users = User::all();

        return view("singlePost", ['post' => $post, 'categories' => $categories,
        'districts' => $districts, 'users' => $users]);
    }

    public function showAddPost(){
        $categories = Category::orderBy('name')->get();
        $districts = District::orderBy('name')->get();
        return view("addPost", ['categories' => $categories,
        'districts' => $districts]);
    }

    public function addPost(Request $req){
        $req->validate([
            'title' => 'required | max:191',
            'description' => 'required | max:191',
            'post_img' => 'required'
        ]);

        $post = new Post;
        $post->title = $req->title;
        $post->description = $req->description;
        $post->category_id = $req->category_id;
        $post->district_id = $req->district_id;
        $post->post_img = $req->post_img;
        $post->post_date = date('Y-m-d');;
        $post->user_id = Auth::id();
        $post->save();
        return redirect("myPost");
    }

    public function deletePost($id){
        $data = Post::find($id);
        $data->delete();
        return redirect("post");
    }

    public function showPost($id){
        $data = Post::find($id);
        $categories = Category::orderBy('name')->get();
        $districts = District::orderBy('name')->get();
        return view("updatePost", ['data' => $data, 'categories' => $categories,
        'districts' => $districts]);
    }

    public function UpdatePost(Request $req){
        $post = Post::find($req->id);

        $req->validate([
            'title' => 'required | max:191',
            'description' => 'required | max:191',
        ]);

        $post->title = $req->title;
        $post->description = $req->description;
        $post->category_id = $req->category_id;
        $post->district_id = $req->district_id;
        $post->post_img = $req->post_img;
        $post->post_date = date('Y-m-d');;
        $post->user_id = Auth::id();
        $post->save();
        return redirect("myPost");
    }
}
```

2.3.4 Models

Admin

Author

```php
<?php

namespace App\Models;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class Admin extends Authenticatable
{
    use Notifiable;
    protected $guard = 'admin';
    protected $fillable = [
        'name', 'email', 'password',
    ];
    protected $hidden = [
        'password', 'remember_token',
    ];
}

```

```php
<?php

namespace App\Models;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
class Author extends Authenticatable
{
    use Notifiable;
    protected $guard = 'author';
    protected $fillable = [
        'name', 'email', 'password',
    ];
    protected $hidden = [
        'password', 'remember_token',
    ];
}

```

Category

```php
<?php

namespace App\Models;

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

class Category extends Model
{
    use HasFactory;

    public $timestamps = false;

    protected $fillable = ['name'];
}

```

District

```php
<?php

namespace App\Models;

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

class District extends Model
{
    use HasFactory;

    public $timestamps = false;

    protected $fillable = ['name'];
}

```

Feedback

```php
<?php

namespace App\Models;

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

class Feedback extends Model
{
    use HasFactory;

    public $table = "feedbacks";

    public $timestamps = false;

    protected $fillable = ['email', 'contact', 'message',
    'user_id', 'status'];
}

```

Post

```php
<?php

namespace App\Models;

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

class Post extends Model
{
    use HasFactory;

    public $timestamps = false;

    protected $fillable = ['title', 'description', 'category_id'
    'user_id', 'district_id', 'post_date', 'post_img'];
}

```

User

```php
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];
```

```php
    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```

2.3.5 Authentication Validation

1. LoginController

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;
use Illuminate\Http\Request;
use Auth;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;
    /**
    * Where to redirect users after login.
    *
    * @var string
    */

    protected $redirectTo = '/home';
    /**
    * Create a new controller instance.
    *
    * @return void
    */

    public function __construct()
    {
        $this->middleware('guest')->except('logout');
        $this->middleware('guest:admin')->except('logout');
        $this->middleware('guest:author')->except('logout');
    }

    public function showAdminLoginForm()
    {
        return view('auth.login', ['url' => 'admin']);
    }

    public function adminLogin(Request $request)
    {
        $this->validate($request, [
            'email' => 'required|email',
            'password' => 'required|min:6'
        ]);
        if (Auth::guard('admin')->attempt(['email' => $request->email, 'password' => $request->password], $request->get('remember'))) {
            return redirect()->intended('/admin');
        }
        return back()->withInput($request->only('email', 'remember'));
    }

    public function showAuthorLoginForm()
    {
        return view('auth.login', ['url' => 'author']);
    }

    public function authorLogin(Request $request)
    {
        $this->validate($request, [
            'email' => 'required|email',
            'password' => 'required|min:6'
        ]);
        if (Auth::guard('author')->attempt(['email' => $request->email,'password' => $request->password], $request->get('remember'))) {
            return redirect()->intended('/author');
        }
        return back()->withInput($request->only('email', 'remember'));
    }
}
```

2.RegisterController

```php
<?php
namespace App\Http\Controllers\Auth;

use App\Models\User;
use App\Models\Admin;
use App\Models\Author;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Illuminate\Foundation\Auth\RegistersUsers;
use Illuminate\Http\Request;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Register Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles the registration of new users as well as their
    | validation and creation. By default this controller uses a trait to
    | provide this functionality without requiring any additional code.
    |
    */

    use RegistersUsers;
    /**
    * Where to redirect users after registration.
    *
    * @var string
    */

    protected $redirectTo = '/home';
    /**
    * Create a new controller instance.
    *
    * @return void
    */

    public function __construct()
    {
        $this->middleware('guest');
        $this->middleware('guest:admin');
        $this->middleware('guest:author');
    }

    * Get a validator for an incoming registration request.
    *
    * @param array $data
    * @return \Illuminate\Contracts\Validation\Validator
    */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:6', 'confirmed'],
        ]);
    }

    /**
    * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
    */
    public function showAdminRegisterForm()
    {
        return view('auth.register', ['url' => 'admin']);
    }

    /**
    * @return \Illuminate\Contracts\View\Factory|\Illuminate\View\View
    */
    public function showAuthorRegisterForm()
    {
        return view('auth.register', ['url' => 'author']);
    }

    /**
    * @param array $data
    *
    * @return mixed
    */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
        ]);
    }

    /**
    * @param Request $request
    *
    * @return \Illuminate\Http\RedirectResponse
    */
    protected function createAdmin(Request $request)
    {
        $this->validator($request->all())->validate();
        Admin::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);
        return redirect()->intended('login/admin');
    }

    /**
    * @param Request $request
    *
    * @return \Illuminate\Http\RedirectResponse
    */
    protected function createAuthor(Request $request)
    {
        $this->validator($request->all())->validate();
        Author::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);
        return redirect()->intended('login/author');
    }
}
```

2.3.6 Gate Implementation

1. Register in AuthServiceProvider.php

```php
public function boot()
{
    $this->registerPolicies();

    // define an administrator user role
    Gate::define('isAdmin', function ($user) {
        return $user->role == 'admin';
    });

    // define an author user role
    Gate::define('isAuthor', function ($user) {
        return $user->role == 'author';
    });

    // define a user role
    Gate::define('isUser', function ($user) {
        return $user->role == 'user';
    });
}
```

2.Use in Blade Template

a. home.blade.php

```blade
@can('isAdmin')
@include('header')
@else
@include('userHeader')
@endcan
```

b. authorUser.blade.php

```blade
@can("isAuthor")
<li>
    <a href="/addPost">Add Post</a>
</li>
<li>
    <a href="/myPost">My Post</a>
</li>
@endcan
```

3.Use in Route

```
//Post Controller
Route::get('/post', [PostController::class, 'index'])->middleware('can:isAdmin');
Route::get('/allPost', [PostController::class, 'allPost'])->middleware('isUserOrAuthor');
Route::get('/singlePost/{id}', [PostController::class, 'singlePost'])->middleware('isUserOrAuthor');
Route::get('/myPost', [PostController::class, 'myPost'])->middleware('can:isAuthor');
Route::get("/addPost", [PostController::class, 'showAddPost'])->middleware('can:isAuthor');
Route::post("addPost", [PostController::class, 'addPost'])->middleware('can:isAuthor');
Route::get("deletePost/{id}", [PostController::class, "deletePost"])->middleware('can:isAdmin');
Route::get("updatePost/{id}", [PostController::class, 'showPost'])->middleware('can:isAuthor');
Route::post("updatePost/{id}", [PostController::class, 'updatePost'])->middleware('can:isAuthor');
```

2.3.7 Session Implementation

1. Register a session (HomeController.php)

```
public function index(Request $req)
{
    $name = Auth::user()->name;
    $id = Auth::id();

    $req->session()->put(['user' => $name, 'id' => $id]);

    return view('home');
}
```

2. Use the session

a. home.blade.php

```
<div class="col-md-12">
    <h2>Hello and welcome back, {{session('user')}}</h2> <br><br>
</div>
```

b. allPost.blade.php

```
<div class="col-md-2">
    <a class='add-new' href='singlePost/{{$post['id']}}'>Read More</a> <br><br>
    @if($post['user_id'] == session('id'))
    <a class="add-new" href="updatePost/{{$post['id']}}">Update Post</a>
    @endif
</div>
```

References

Charles, N.C. (2023). *Laravel MVC Architecture Explained*. [online] Medium. Available at: https://medium.com/@nnadichime04/laravel-mvc-architecture-explained-21e783dbbd14 [Accessed 8 Mar. 2024].

Clark, J., 2024. *Top 10 backend frameworks in 2024: Which one is the best?* [online] Back4App Blog. Available at: <https://blog.back4app.com/backend-frameworks/> [Accessed 19 Apr. 2024].

Johns, R., 2024. *10 best web development frameworks in 2024 [updated]*. [online] Hackr.io. Available at: <https://hackr.io/blog/web-development-frameworks> [Accessed 19 Apr. 2024].

ICStudio, 2024. *17 benefits of Laravel Framework.* [online] Веб-студія розробки програмного забезпечення - IC Studio, Україна. Available at: <https://icstudio.online/en/post/17-benefits-laravel-framework> [Accessed 19 Apr. 2024].

Konarski, M., 2024. *The Future of Web Application Development: Artificial Intelligence, Generative AI, Cybersecurity, and technological adaptation.* [online] iMakeable. Available at: <https://imakeable.com/en/blog/artificial-intelligence-generative-ai-machine-learning-in-web-application-development> [Accessed 19 Apr. 2024].

Kvernadze, L., 2023. *The role of Artificial Intelligence (AI) in web app development in 2023.* [online] LinkedIn. Available at: <https://www.linkedin.com/pulse/role-artificial-intelligence-ai-web-app-development-2021-kvernadze/> [Accessed 19 Apr. 2024].

Team, Z., 2024. *3 undeniable characteristics of modern web apps.* [online] Zartis. Available at: <https://www.zartis.com/3-undeniable-characteristics-of-modern-web-apps/> [Accessed 19 Apr. 2024].

Team, T.E. (2019). *The 7 Most Important Software Design Patterns*. [online] Medium. Available at: https://learningdaily.dev/the-7-most-important-software-design-patterns-d60e546afb0e.

Shalinda, N. (2023). *Design Pattern in Software Development*. [online] Medium. Available at: https://medium.com/@sjmnaveenshalinda/design-pattern-in-software-development-9d38e8f91b0b.

Assessment

Criteria

Part 1 (40%):

COs	Criteria	Poor (0)	Needs Improvement (1-20)	Satisfactory (21-60)	Good (61-80)	Excellent (81-100)
CO1	Analogy of Web Frameworks (15 marks)	No comparison or irrelevant information.	Inaccurate or incomplete comparison, lacks understanding of strengths/weaknesses.	Basic comparison, limited strengths/weaknesses, no personal perspective.	Clear comparison, reasonable strengths/weaknesses, some personal opinion.	Comprehensive comparison of frameworks, clear strengths/weaknesses, insightful personal perspective.
CO1	Software Design Patterns (15 marks)	No discription of software design patterns or irrelevant information.	Misunderstanding of patterns, inaccurate benefits/drawbacks, or irrelevant information.	Basic explanation of patterns, some understanding of benefits/drawbacks, no discussion of alternatives.	Good explanation of patterns, reasonable benefits/drawbacks, limited discussion of alternatives.	In-depth explanation of patterns, clear benefits/drawbacks, discussion of alternatives.

CO1	Quality of report writing (5 marks)	Difficult to understand, disorganized, significant errors, inaccurate or irrelevant information.	Unclear, disorganized, numerous errors, incorrect technical terms.	Organized, some errors, inaccuracies in technical terms.	Clear and concise, few errors, mostly accurate technical terms.	Well-organized document, free of errors, uses technical terms correctly.
CO1	Visuals and Citations (5 marks)	Irrelevant or misleading visuals, incorrect or missing citations.	No visuals or citations.	Few or irrelevant visuals, inconsistent or missing citations.	Some visuals or citations used, inconsistent or unclear.	Relevant and informative visuals enhance understanding, consistent citation style.

Part 2 (60%):

	COs	Criteria	Poor (0)	Needs Improvement (1-20)	Satisfactory (21-60)	Good (61-80)	Excellent (81-100)
A	CO1	View/Controller /Model (20 marks)	No or non-View/Controller /Model.	Incomplete view/controller/model or missing key features.	Basic view/controller/model, limited organization.	Well-defined view/controller/model, following some best practices.	Well-defined view/controller/model, following best practices.
	CO1	Database migration (5 marks)	No database migration.	Incomplete database migration or missing key features.	Basic database migration, limited organization.	Well-defined database migration, following some best practices.	Well-defined database migration, following best practices.
B	CO2	User Authentication (10 marks)	No authentication.	Missing authentication features or insecure authentication mechanisms.	Limited authentication, insecure authentication, or unclear implementation.	Multiple user types, basic authentication, some authentication implementation.	Distinct user types, appropriate authentication, secure authentication mechanism.
	CO2	User Roles and Permissions (15 marks)	No roles or permissions.	Missing role-based features or insecure access control mechanisms.	Limited roles or permissions, insecure access control, or unclear implementation.	Multiple roles, basic permissions, some access control implementation.	Distinct roles, appropriate permissions, secure access control using policies/middleware.

CO2	Code Quality and Documentation (10 marks)	Unusable code or no documentation.	Poorly written code, difficult to understand, missing or inaccurate documentation.	Unorganized code, limited readability, incomplete documentation.	Mostly readable code, some organization issues, basic documentation information with presentation of app screens.	Clean, well-organized code, clear documentation with presentation of app screens.