UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

TEST

January 2021 Trimester

|  |  |  |  |
| --- | --- | --- | --- |
| Name | TAN YING YAO | Q1: | Total Marks:  /50 |
| ID | 1703648 | Q2: |

**Course Outcomes.**

In this assessment, you are assessed based on the following **course outcomes**:

|  |  |
| --- | --- |
| **CO1** | Build interactive and database-driven web applications using a web application framework and model-view-controller (MVC) software architecture. |
| **CO2** | Build web applications with user authentication and authorization using cookies, session and role-based access control (RBAC). |

**Assessment Contribution**

This assignment contributes **20%** to the overall assessment for this course.

**Date**

The date and time for this assessment is **Thursday, 11th March 2021, 8:30am – 10am**.

Instruction: Answer all questions in the space provided. You have ONE (1) hour to complete this test.

|  |  |
| --- | --- |
| **Question 1** | **[25 marks]** |
| 1. Developing a Laravel web application for an automotive sale industry requires three models namely, Vehicle Manufacturer, Vehicle Company, Vehicle Category and Vehicle Model. A vehicle manufacturer belongs to exactly one company and may have one or more vehicle categories. Many companies may have the same vehicle category. One Vehicle Model can belong to only one vehicle manufacturer, one vehicle company and one vehicle category. Define **ONE (1)** relationship assignment method/function for each Vehicle Manufacturer, Vehicle Company, Vehicle Category and Vehicle Model model classes. (8 marks) 2. Define the validation rules to validate user input for the Vehicle Model model of the above mentioned web application in (a) based on the requirements specified in Table 1.1 below. (9 marks)  |  |  | | --- | --- | | Attribute | Rules | | model\_number | Required, unique, contain 3-10 uppercase alphabets (A-Z) in front following with 0-3 digits or numbers | | model\_manufacturer | Required, maximum length of 20 characters | | model\_category | Required, minimum 5 and maximum 10 characters | | company\_contact | Required, contain 2-3 digits or numbers following a dash “-” then 6-8 digits digits or numbers | | company\_email | Required, must be a valid email address |   Table 1.1: Required validation rules for Vehicle Model model.   1. Define the database schema of database migration table for the Vehicle Model of the above mentioned web application in (a) based on the attributes specified in Table 1.1. (6 marks) 2. Based on your understanding or own experience, explain the advantage of using database migration in Laravel framework. (2 marks) | |
| **Question 2** | **[25 marks]** |
| 1. Explain **ONE (1)** difference between gates and policies with **ONE (1)** justification to your project leader of the reason in case you choose to use both or just one of the RBAC approach. (5 marks) 2. Define appropriate gates **and** relevant policies for following user actions and relevant model: 3. The user is allowed to create new Vehicle Model. (4 marks) 4. The user is allowed to edit and delete own Vehicle Company. (8 marks) 5. The user is allowed to create and delete any Vehicle Categories. (8 marks) | |

**Student’s Answers:**

1. A)

Vehicle Manufacturer – one-to-many

Vehicle Company – one-to-many

Vehicle Category – many-to-many

Vehicle Model – one-to-one

Class VehicleManufacturer extend model()

{  
Public function Manufacturer()

{

Return $this -> hasMany(Vehicle::class)

}}

Class VehicleCompany extend model()

{

Public function Company()

{

return $this -> hasMany(Vehicle::class)

}

}

Class VehicleCategory extend model()

{

Public function Category()

{

Return $this -> belongsToMany(Vehicle::class)

}

}

Class VehicleModel extend model()

{

Public function Model()

{

Return $this -> belongsTo(Vehicle::class)

}

}

B)

Public function Validator()

{

Return [

‘model\_number’ => [‘required’,’unique:number’,’regex:/^[A-Z]{3,10}[0-9]{0,3}$/’],

‘model\_manufacturer’ => ‘required|max:20’,

‘model\_category’ => ‘required|min:5|max:10’,

‘company\_contact’ => ‘required|’regex:(^[0-9]{2,3}-[0-9]{6-8})$’,

‘company\_email’ => ‘required|email:rfc,dns’,

];

}

C)

Schema::create(‘vehicle, function (Model $table)  
{

$table->string(‘model\_number’)->unique();

$table->string(‘model\_manufacturer’);

$table->string(‘model\_category’);

$table->string(‘company\_contact’);

$table->string(‘company\_email’);

}

D)

It can serve as a type of version control system for database as the schema can be easily modify and updated with artisan commands. It can also serve as a backup as it can recreate a database easily.

A)

Gates and policies mainly differ in which gates are most applicable to actions which are related to model or resource and policies are used to authorize an action for a particular model or resource. There is no need to choose between the both of them as it is not mutually exclusive. Both can be used to great effect as they can be integrated to ensure a secure role-base access control application can be created.

B)

**The user is allowed to create new Vehicle Model.**

//gate function

Public function createModel()

{

Gate::define(‘create-model’, function(User $user)

{return true;

});

}

//policies function

Public function createModel(Request $request)

{

If ($request->user()->cannot(‘createModel’, Post::class))

{

Abort(403);

}

}

**The user is allowed to edit and delete own Vehicle Company.**

//gate

Public function editCompany()

{

Gate::define(‘edit-company’, function(User $user, Company $company)

{return $user->id === $company->user\_id;

});

}

Public function deleteCompany()

{

Gate::define(‘delete-company’, function(User $user)

{return true;

});

}

//policies

Public function editCompany(Request $request, Post $post){

$this->authorize(‘editCompany’, $post);

}

Public function deleteCompany(Request $request, Post $post){

$this->authorize(‘deleteCompany’, $post);

}

**The user is allowed to create and delete any Vehicle Categories.**

Public function createCategories()

{

Gate::define(‘create-categories’, function(User $user)

{return true;

});

}

Public function deleteCategories()

{

Gate::define(‘delete-categories’, function(User $user)

{return true;

});

}

//policies

Public function createCategories(Request $request){

$this->authorize(‘createCategories, Post::class);

}  
Public function deleteCategories(Request $request){

$this->authorize(‘deleteCategories, Post::class);

}