UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

TEST

January 2021 Trimester

Name	[redacted]	Q1: 23	Total Marks:
ID	[redacted]	Q2: 22½	45½/50

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**.

TEST - UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT
Instruction: Answer all questions in the space provided. You have ONE (1) hour to complete this test.

Question 1 **[25 marks]**

a) 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)

b) 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.

c) 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)

d) Based on your understanding or own experience, explain the advantage of using database migration in Laravel framework. (2 marks)

Question 2	**[25 marks]**

a) 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)

b) Define appropriate gates **and** relevant policies for following user actions and relevant model:

(i) The user is allowed to create new Vehicle Model. (4 marks)

(ii) The user is allowed to edit and delete own Vehicle Company. (8 marks)

(iii) The user is allowed to create and delete any Vehicle Categories. (8 marks)

TEST - UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

Student's Answers:

8

1 (a)

```
class VehicleManufacturer extends Model {
        public function company() {
                return $this->belongsTo(VehicleCompany::class);
        }
        public function categories() {
                return $this->hasMany(VehicleCategory::class);
        }
}
class VehicleCompany extends Model {
        public function categories() {
                return $this->hasMany(VehicleCategory::class);
        }
}
class VehicleCategory extends Model {
        public function companies() {
                return $this->belongsToMany(VehicleCompany::class);
        }
}
class VehicleModel extends Model {
        public function manufacturer() {
                return $this->belongsTo(VehicleManufacturer::class);
        }
        public function company() {
                return $this->belongsTo(VehicleCompany::class);
        }
        public function category() {
                return $this->belongsTo(VahicleCategory::class);
        }
}
```

1/2

1/2

1/2

1/2

1 (b)

```
public function rules() {
        return [
            'model_number' => ['required', 'unique:VehicleModel',
                'regex:/^([A-Z]{3,10})(\d{3})$/'],
            'model_manufacturer' => 'required|max:20',
            'model_category' => 'required|min:5|max:10',
            'company_contact' => ['required', 'regex:/ ^(\d{2,3})\-(\d{7,8})$/'],
            'company_email' => 'required|email'
        ];
    }
```

1 (c)

```
Schema::create('vehicleModel', function (Blueprint $table){
        $table->id();
        $table->string('model_number')->unique();
        $table->string('model_manufacturer');
        $table->string('model_category');
        $table->string('company_contact');
        $table->string('company_email');
        $table->timestamps();
    })
```

1 (d)

The advantage of using database migration is we can declare the database schema in the code and let Laravel framework to create the database table automatically. Through this method, we can match the database schema and our class properties easily since both of them are in the code and we do no need to check the database manually to match the schema with the class properties.

Great!

TEST - UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

2 (a)

The difference of gates and policies is gates are guarding based on the actions of the users while policies are guarding based on the models or actions on model. Therefore, the gates need to be called manually while the policies are automatically used when certain functions of the models are called.

I would choose to use both gates and policies since they are acting on different scope which are actions and models. There are cases when we need to use actions which are not related to model to guard the action while there are times when we can use policies instead of gates to guard model easily.

2 (b) (i)

```
Gate::define('create-vehicle-model',  [VehicleModelPolicy::class, 'create']);

class VehicleModelPolicy {
        public function create(User $user) {
                return true;
        };
}
```

(ii)

```
Gate::define('edit-vehicle-company,  [VehicleCompanyPolicy::class, 'update']);
Gate::define('delete-vehicle-company,  [VehicleCompanyPolicy::class, 'delete']);

class VehicleCompanyPolicy {
        public function update(User $user, VehicleCompany $company) {
                return $user->id === $company->owner_id;
        };
        public function delete(User $user, VehicleCompany $company) {
                return $user->id === $company->owner_id;
        };
}
```

(iii)
```
Gate::define('create-vehicle-category,  [VehicleCategoryPolicy::class, 'create']);
Gate::define('delete-vehicle-category,  [VehicleCategoryPolicy::class, 'delete']);

class VehicleCategoryPolicy {
    public function create(User $user) {
        return true;
    };
    public function delete(User $user) {
        return true;
    };
}
```