**<?php***/\*  
|--------------------------------------------------------------------------  
| Application Routes  
|--------------------------------------------------------------------------  
|  
| Here is where you can register all of the routes for an application.  
| It's a breeze. Simply tell Laravel the URIs it should respond to  
| and give it the Closure to execute when that URI is requested.  
|  
\*/*Route::get('login', 'LoginController@get');  
Route::post('login', 'LoginController@post');  
Route::group(**array**('before'=>'auth'), **function**(){  
 Route::get('/', **function**(){**return** Redirect::to('test');});  
 Route::get('test', 'TestController@get');  
 Route::post('test', 'TestController@post');  
 Route::get('result', 'ResultController@get');  
 Route::post('result', 'ResultController@post');  
 *//only for admin* Route::group(**array**('prefix' => 'admin', 'before' => 'admin'), **function**(){  
 Route::get('/', 'AdminController@get');  
 Route::post('/', 'AdminController@post');  
  
 Route::get('test-options', 'TestOptionsController@get');  
 Route::get('chapter-rate', 'ChapterRateController@get');  
 *//bind model for chapters* Route::model('chapter', 'Chapter');  
 Route::get('chapters', 'ChaptersController@get');  
 Route::get('chapters/{chapter}', 'ChaptersController@getQuestions');  
  
 Route::post('test-options', 'TestOptionsController@post');  
 Route::post('chapter-rate', 'ChapterRateController@post');  
 Route::post('chapters', 'ChaptersController@post');  
 Route::post('chapters/{chapter}', 'ChaptersController@postQuestionChange');  
  
 *//modify database: delete-all* Route::get('delete-all/contestant', **function**(){Contestant::truncate();});  
 });  
});  
Route::get('log-out', 'LogOutController@get');

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 1/1/2016  
 \* Time: 9:00 PM  
 \*/***class** LoginController **extends** ProjectController{  
 **const** *MESSAGE\_KEY* = "OF1sc1gV";  
 *//handle get request from 'login'* **public function** get(){  
 *//push 'messages' to view* $messages = $this->messageController->getMessages($this::MESSAGE\_KEY);  
 **return** View::make('login',**array**(  
 'messages' => $messages,  
 ));  
 }  
 *//handle post request from 'login'* **public function** post(){  
 *//step 1: validate input-data* $validate\_data = Input::only('contestant\_id', 'keystone');  
 $validate\_rules = **array**(  
 'contestant\_id' => 'required|integer',  
 'keystone' => 'required|min:8'  
 );  
 $validator = Validator::make($validate\_data, $validate\_rules);  
 **if**($validator->fails()){  
 $validate\_messages = $validator->messages()->toArray();  
 $this->messageController->send($validate\_messages, $this::MESSAGE\_KEY);  
 **return** Redirect::to('login');  
 }  
 *//step 2: check empty collection from 'contestant\_id', bcs it may not exist* $contestant = Contestant::find(Input::get('contestant\_id'));  
 **if**(!$contestant){  
 $this->messageController->send(**array**('contestant\_id' => ['contestant\_id:wrong']), $this::MESSAGE\_KEY);  
 **return** Redirect::to('login');  
 }  
 *//step 3: compare hashed-value, if equal, allow login  
 //what we get after find is a 'collection', not a Contestant's instance, so fetch it, first()* **if**(Hash::check(Input::get('keystone'), $contestant->keystone)){  
 Auth::login($contestant);  
 **if**($contestant->id == 1){  
 *//admin after 'login' refer go to 'admin' page* **return** Redirect::to('admin');  
 }**else**{  
 *//contestant after 'login' refer goto 'test' page* **return** Redirect::to('test');  
 }  
 }**else**{  
 $this->messageController->send(**array**('keystone' => ['keystone:wrong']), $this::MESSAGE\_KEY);  
 }  
 *//as a fall-back, return to login* **return** Redirect::to('login');  
 }  
}

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 07-Jan-16  
 \* Time: 9:36 PM  
 \*/***class** TestController **extends** ProjectController{  
 **const** *MESSAGE\_KEY* = "OFwQq7ob";  
 *//handle get-request from 'test'* **public function** get(){  
 $messages = $this->messageController->getMessages($this::MESSAGE\_KEY);  
 *//contestant take test only one time, so  
 //if !admin && has-result >>> redirect to 'result'  
 //(admin not blocked, need to review test-page)* $contestant = Auth::user();  
 **if**($contestant->id != 1 && $contestant->result != ""){  
 **return** Redirect::to('result');  
 }  
 *//load random-question from database* $chapters = Chapter::all();  
 $random\_questions = **array**();  
 **foreach**($chapters **as** $chapter) {  
 *//each chapter, load random-questions by chapter-rate  
 //then store in $random\_questions* **if**($chapter->rate != 0){*//make sure chapter has question* $random\_questions[] = $chapter->getQuestions->random($chapter->rate);  
 }  
 }  
 *//get timer from test option* $test\_options = TestOption::all();  
 $timer = $test\_options[0];  
 **return** View::make('test', **array**(  
 'random\_questions' => $random\_questions,  
 'timer' => $timer,  
 'messages' => $messages  
 ));  
 }  
 *//handle post-request from 'test'* **public function** post(){  
 *//receive chose-options from contestant* $answers\_contestant = array\_values(Input::all());  
 $result = 0;  
 **for**($i = 0; $i < count($answers\_contestant); $i++){  
 $option = Option::find($answers\_contestant[$i]);  
 *//if the option is right, result++* **if**($option->is\_right == '1'){  
 $result += 1;  
 }  
 }  
 *//save result from this contestant* $contestant = Auth::user();  
 $contestant->result = $result;  
 $contestant->save();  
 *//show result on 'result'-page* **return** Redirect::to('result');  
 }  
}

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 07-Jan-16  
 \* Time: 9:14 PM  
 \*/***class** ResultController **extends** ProjectController{  
 **const** *MESSAGE\_KEY* = "6wXNciDn";  
 *//handle get-request from 'result'* **public function** get(){  
 $contestant = Auth::user();  
 *//push 'messages' to view* $messages = $this->messageController->getMessages($this::MESSAGE\_KEY);  
 **return** View::make('result', **array**(  
 'contestant' => $contestant,  
 'messages' => $messages  
 ));  
 }  
 *//handle post-request from 'result'* **public function** post(){  
 $contestant\_email = Input::get('contestant\_email');  
 $contestant = Auth::user();  
 $contestant->email = $contestant\_email;  
 $contestant->save();  
 **return** Redirect::to('result');  
 }  
}

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 08-Jan-16  
 \* Time: 12:10 PM  
 \*/***abstract class** ProjectController **extends** BaseController{  
 **protected** $messageController;  
 **public function** \_\_construct(){  
 $this->messageController = **new** MessageController();  
 }  
}

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 1/4/2016  
 \* Time: 4:11 PM  
 \*/***class** MessageController{  
 **public function** \_\_construct(){  
 }  
 **public function** send($messages, $message\_key){  
 Session::put($message\_key, $messages);  
 }  
 **public function** getMessages($message\_key){  
 **if**(Session::has($message\_key)){  
 **return** Session::pull($message\_key);  
 }  
 **return array**();  
 }  
}**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 12/30/2015  
 \* Time: 2:50 PM  
 \*/***class** AdminController **extends** ProjectController{  
 **const** *MESSAGE\_KEY* = "I7fSIL5A";  
 *//handle get-request from 'admin'* **public function** get(){  
 *//check contestant is admin  
 //admin has id == 1* $contestant = Auth::user();  
 **if**($contestant->id != "1"){  
 **return** Redirect::to('test');  
 }  
 *//go to admin-page  
 //push 'messages' to view* $messages = $this->messageController->getMessages($this::MESSAGE\_KEY);  
 **return** View::make('admin', **array**(  
 'messages' => $messages  
 ));  
 }  
 **public function** post(){  
 *//admin-navigation* $admin\_navigation = **new** AdminNavigation();  
 **if**($admin\_navigation->isNavigate()){  
 **return** $admin\_navigation->goToN();  
 }  
 **return false**;  
 }  
}

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 12/30/2015  
 \* Time: 2:38 PM  
 \*/***class** TestOptionsController **extends** ProjectController{  
 **const** *MESSAGE\_KEY* = "30dwbOfr";  
 **public function** get(){  
 $test\_options = TestOption::all();  
 *//push 'messages' to view* $messages = $this->messageController->getMessages($this::MESSAGE\_KEY);  
 **return** View::make('admin-test-options', **array**(  
 'test\_options' => $test\_options,  
 'messages' => $messages  
 ));  
 }  
  
 **public function** post(){  
 *//admin-navigation* $admin\_navigation = **new** AdminNavigation();  
 **if**($admin\_navigation->isNavigate()){  
 **return** $admin\_navigation->goToN();  
 }  
 *//message-notification* $messages = **array**();  
 *//at this time, just one option in test-options, so* $test\_option = TestOption::first();*//'timer'* **if**(Input::has($test\_option->key)){  
 $test\_option->value = Input::get($test\_option->key);  
 $test\_option->save();  
 $messages[$test\_option->key] = $test\_option->key.':saved';  
 $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
 **return** Redirect::to('result');  
 }  
}

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 12/30/2015  
 \* Time: 2:41 PM  
 \*/***class** ChapterRateController **extends** ProjectController{  
 **const** *MESSAGE\_KEY* = "sqvbTPJC";  
 **public function** get(){  
 $chapters = Chapter::all();  
 *//push 'messages' to view* $messages = $this->messageController->getMessages($this::MESSAGE\_KEY);  
 **return** View::make('admin-chapter-rate', **array**(  
 'chapters' => $chapters,  
 'messages' => $messages  
 ));  
 }  
 **public function** post(){  
 *//page-navigation* $admin\_navigation = **new** AdminNavigation();  
 **if**($admin\_navigation->isNavigate()){  
 **return** $admin\_navigation->goToN();  
 }  
 *//message-notification* $messages = **array**();  
 **if**(Input::has('chapter\_rate')){  
 $chapters = Chapter::all();  
 *//get chapter-data, create chapter-rules for validation* $chapter\_rate\_data = **array**();  
 $chapter\_rate\_rules = **array**();  
 **foreach**($chapters **as** $chapter){  
 $max\_rate = $chapter->getQuestions->count();  
 $chapter\_rate\_data[$chapter->id] = Input::get($chapter->id);  
 $chapter\_rate\_rules[$chapter->id] = 'integer|max:'.$max\_rate;  
 }  
 *//validate if chapter-rate <= max\_rate* $validator = Validator::make($chapter\_rate\_data, $chapter\_rate\_rules);  
 **if**($validator->fails()){  
 $validate\_messages = $validator->messages()->toArray();  
 $this->messageController->send($validate\_messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
 *//save chapter-rate to database* $i = -1;  
 **foreach**($chapters **as** $chapter){  
 $chapter->rate = Input::get($chapter->id);  
 $chapter->save();  
 $messages['chapter\_rate['.(++$i).']'] = 'chapter-'.$chapter->id.'-chapter\_rate:saved';  
 }  
 *//return back, show confirm from database* $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
 *//as a fallback* $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::to('admin/chapter-rate');  
 }  
}

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 12/30/2015  
 \* Time: 2:44 PM  
 \*/***class** ChaptersController **extends** ProjectController{  
 **const** *MESSAGE\_KEY* = "lWln8RGZ";  
 **public function** get(){  
 $chapters = Chapter::all();  
 *//push 'messages' to view* $messages = $this->messageController->getMessages($this::MESSAGE\_KEY);  
 **return** View::make('admin-chapters', **array**(  
 'chapters' => $chapters,  
 'messages' => $messages  
 ));  
 }  
 **public function** getQuestions(Chapter $chapter){  
 $total\_chapter = Chapter::all()->count();  
 $questions = $chapter->getQuestions;  
 *//push 'messages' to view* $messages = $this->messageController->getMessages($this::MESSAGE\_KEY);  
 **return** View::make('admin-chapter-questions', **array**(  
 'chapter' => $chapter,  
 'questions' => $questions,  
 'total\_chapter' => $total\_chapter,  
 'messages' => $messages  
 ));  
 }  
 **public function** post(){  
 *//message-notification* $messages = **array**();  
 *//handle navigation* $admin\_navigation = **new** AdminNavigation();  
 **if**($admin\_navigation->isNavigate()){  
 **return** $admin\_navigation->goToN();  
 }  
 *//chapters-change  
 //delete-chapter* **if**(Input::has('delete\_chapter')){  
 $chapter = Chapter::find(Input::get('chapter\_id'));  
 $questions = $chapter->getQuestions;  
 **if**(count($questions) > 0){  
 $messages['delete\_chapter'] = 'chapter-'.$chapter->id.':has questions';  
 }**else**{  
 $chapter->delete();  
 $messages['delete\_chapter'] = 'chapter-'.$chapter->id.':deleted';  
 }  
 }  
 *//new-chapter* **if**(Input::has('new\_chapter')){  
 $chapter = **new** Chapter;  
 *//delete + auto\_increment >>> modidify chapter-id not continuous  
 //manually change chapter-id* $last\_chapter = Chapter::all()->last();  
 $chapter->id = $last\_chapter->id + 1;  
 $chapter->text = Input::get('chapter\_text');  
 $chapter->rate = 0;  
 $chapter->save();  
 *//messages-notification* $messages['new\_chapter'] = 'chapter-'.$chapter->id.'-'.$chapter->text.':saved';  
 }  
 $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
 **public function** postQuestionChange(){  
 *//message-notification* $messages = **array**();  
 *//handle navigation* $admin\_navigation = **new** AdminNavigation();  
 **if**($admin\_navigation->isNavigate()){  
 **return** $admin\_navigation->goToN();  
 }  
 *//handle chapter-text change  
 //redirect after changed* **if**(Input::has('chapter\_change')){  
 $chapter = Chapter::find(Input::get('chapter\_change'));  
 $chapter->text = Input::get('chapter\_text');  
 $chapter->save();  
 $messages['chapter\_change\_text'] = 'chapter-'.$chapter->id.':saved';  
 $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
 *//handle delete-question  
 //redirect after deleted, no need to modify other inputs* **if**(Input::has('delete\_question')){  
 $question = Question::find(Input::get('delete\_question'));  
 $store\_question\_id = $question->id;  
 $question->delete();  
 $messages['delete\_question'] = 'question-'.$store\_question\_id.':deleted';  
 $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
 *//handle change on a question (both this one, and it's options)  
 //redirect after all-changes saved* **if**(Input::has('question\_change')){  
 $question = Question::find(Input::get('question\_change'));  
 *//question-change, change question-text* **if**(Input::has('question\_text')){  
 $question->text = Input::get('question\_text');  
 $question->save();  
 $messages['question\_change\_text'] = 'question-'.$question->id.':saved';  
 }  
 *//question-change, change question-chapter\_id* **if**(Input::has('chapter\_id')){  
 $question->chapter\_id = Input::get('chapter\_id');  
 $question->save();  
 $new\_chapter = $question->getChapter;  
 $messages['question\_change\_chapter\_id'] = 'question-'.$question->id.':now belongs to chapter-'.$new\_chapter->id;  
 }  
 *//options-change* **if**(Input::has('options')) {  
 $options = Input::get('options');  
 *//save options-change* $i = -1;  
 **foreach**($options **as** $option\_id => $option\_text){  
 $option = Option::find($option\_id);  
 $option->text = $option\_text;  
 *//reset all option-is\_right = 0  
 //is\_right set again with input-is\_right checked* $option->is\_right = 0;  
 $option->save();  
 $messages['options\_change['.(++$i).']'] = 'option-'.$option->id.':saved';  
 }  
 *//modify option-is\_right* **if**(Input::has('is\_right')){  
 $option = Option::find(Input::get('is\_right'));  
 *//this option set is\_right = 1* $option->is\_right = 1;  
 $option->save();  
 $messages['options\_change\_is\_right'] = 'option-'.$option->id.'-is\_right:saved';  
 }  
 }  
 *//send message-notification* $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
 *//new-question  
 //redirect after create new-one* **if**(Input::has('new\_question')){  
 *//save new question* $question = **new** Question;  
 *//delete + auto\_increment >>> modify question-id not continuous  
 //manually change question-id* $last\_question = Question::all()->last();  
 $question->id = $last\_question->id + 1;  
 $question->text = Input::get('question\_text');  
 $question->chapter\_id = Input::get('chapter\_id');  
 $question->save();  
 $question\_id = $question->id;  
 $messages['new\_question'] = 'question-'.$question->id.':saved';  
 *//save new options* $options\_text = Input::get('options');  
 $created\_options = **array**();  
 **for**($i = 0; $i < 4; $i++){  
 $option = **new** Option;  
 $option->text = $options\_text[$i];  
 $option->question\_id = $question\_id;  
 $option->is\_right = 0;  
 $option->save();  
 *//store in array new-option in $created\_options, to add is\_right on which* $created\_options[$i] = $option;  
 $messages['option['.$i.']'] = 'option-'.$option->id.':saved';  
 }  
 **if**(Input::has('is\_right')){  
 $right\_option = Input::get('is\_right');  
 *//get option from store-$created\_options, which selected is\_right* $option = $created\_options[$right\_option];  
 $option->is\_right = 1;  
 $option->save();  
 $messages['option\_is\_right'] = 'option-'.$option->id.'-is\_right:saved';  
 }  
 *//send message-notification* $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
 *//as a fallback  
 //send message-notification* $this->messageController->send($messages, $this::MESSAGE\_KEY);  
 **return** Redirect::back();  
 }  
}

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Hoang Anh  
 \* Date: 07-Jan-16  
 \* Time: 9:33 PM  
 \*/***class** LogOutController **extends** Controller{  
 *//handle get-request from 'log-out'* **public function** get(){  
 **if**(Auth::check()){  
 Auth::logout();  
 **echo** '<h1>you-have-logged-out</h1>';  
 }**else**{  
 **echo** '<h1>you-havent-log-in</h1>';  
 }  
 }  
}