



Quiz Manager Submission: **Development Diary**

Prepared by: Linna Trieu, Associate Software Engineer at Paddle

Qualification: Level 4 Software Development Apprenticeship with Makers Academy

Prepared for: BCS, Digital Industries Apprenticeship

Date: September 2020

DEVELOPMENT DIARY

Overview

- This document will serve as a development diary for the construction phase of the Quiz Manager project. This will contain informal dialogue and narrative as I progressed through the development of the website, including progress updates to granular detail of code snippets as I debug issues with the website.
- The structure of the development diary will change intermittently and will likely not be consistent as it is not a structured document.

Dates covered

Please note that this development diary predominantly covers the construction phase of the project. It may make reference to the following days of my project:

- Day 1 - 08/09/20
- Day 2 - 09/09/20
- Day 3 - 10/09/20
- Day 4 - 11/09/20
- Day 5 - 14/09/20
- Day 6 - 15/09/20
- Day 7 - 16/09/20

DAY 2 - 09/09/20

Lots to do to kick off the Application.

Firstly I need to get the scaffolding and infrastructure up first. In particular, set up:

1. Laravel Scaffold ✓

- Set up the Laravel generic framework, out of the box MVC template.
- Commit all vendor files to the project, around 8,000 files (the project has to be run offline)

2. Laravel Built-In Authentication ✓

- Decided to use built-in auth as I have experience with this in my job as an engineer at Paddle. Laravel offers authentication out of the box as part of the framework.

3. Bootstrap Library ✓

- I had to download the dependency using composer and then move the bootstrap files into the public resources directory. This is where public assets should be located in Laravel best practice.
- Did this using a script in the composer.json file, to automate the process.

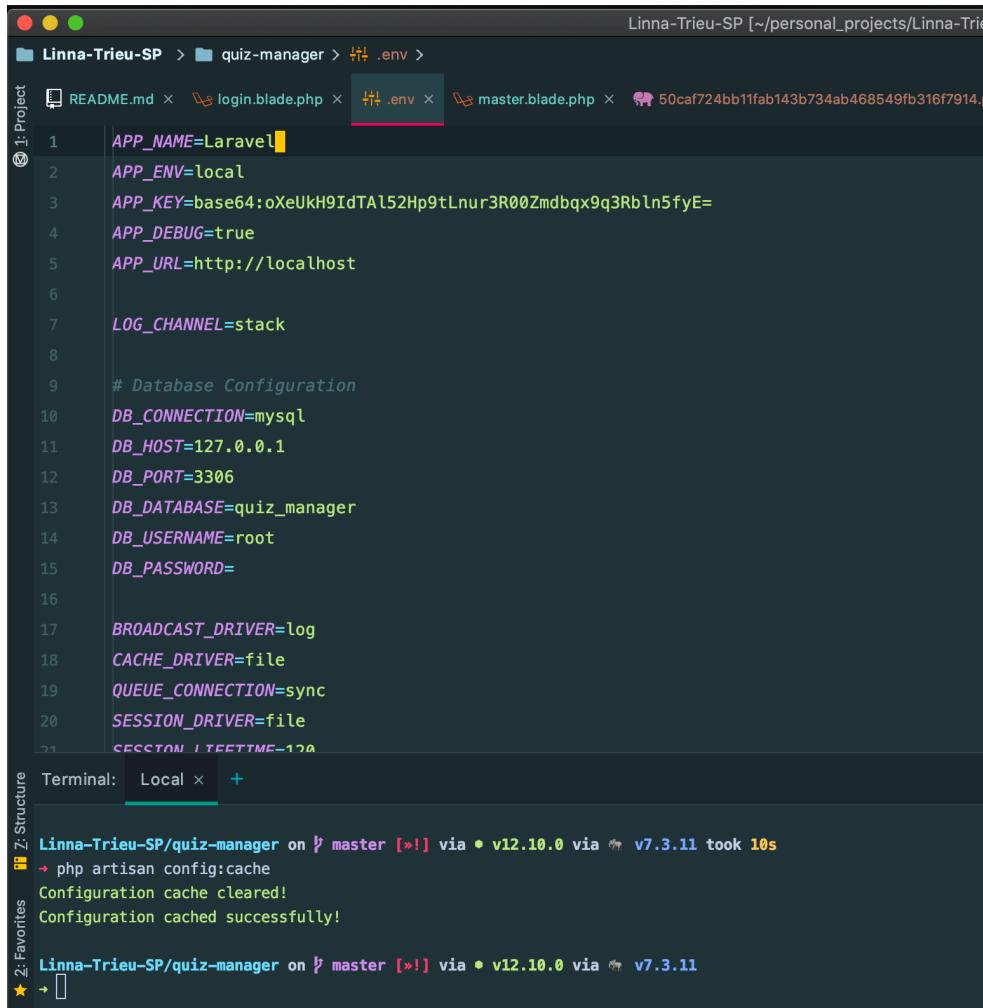
4. DB Config ✓

- Created a local database called 'quiz_manager' in mySQL client
- Set up the app config to connect the application to the new database

5. Making generic laravel scaffold bespoke ✓

- I am renaming the generic template content for Quiz Manager - eg. update the HTML title, app name, favicon, and navbar.
- Issues replacing the `app.name` across the website. I have changed the env variables but nothing has changed on the front end. Why?
- Resolved: needed to look for further environment variables for the app name! Missed APP_NAME

- I also needed to clear the cache after changing environment variables, in order to see the changes updated on the front end!



The screenshot shows a terminal window with the following content:

```

Linna-Trieu-SP [~/personal_projects/Linna-Trieu-SP]
Linna-Trieu-SP > quiz-manager > .env >
1: Project 2: Structure 3: Favorites
  README.md x login.blade.php x .env x master.blade.php x 50caf724bb11fab143b734ab468549fb316f7914.php
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:oXeUkH9IdTAl52Hp9tLnur3R00Zmdbqx9q3Rbln5fyE=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 # Database Configuration
10 DB_CONNECTION=mysql
11 DB_HOST=127.0.0.1
12 DB_PORT=3306
13 DB_DATABASE=quiz_manager
14 DB_USERNAME=root
15 DB_PASSWORD=
16
17 BROADCAST_DRIVER=log
18 CACHE_DRIVER=file
19 QUEUE_CONNECTION=sync
20 SESSION_DRIVER=file
21 SESSION_LIFETIME=120

Terminal: Local x +
```

Output from the terminal:

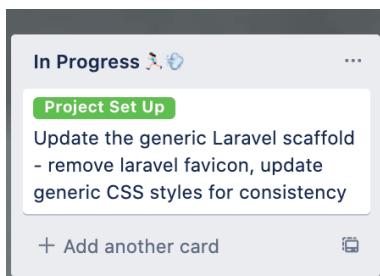
```

Linna-Trieu-SP/quiz-manager on ✘ master [!] via • v12.10.0 via ⚡ v7.3.11 took 10s
→ php artisan config:cache
Configuration cache cleared!
Configuration cached successfully!

Linna-Trieu-SP/quiz-manager on ✘ master [!] via • v12.10.0 via ⚡ v7.3.11
★ →
```

6. CSS Styles Consistency and Favicon

- I couldn't figure out how to do this, so I timeboxed an investigation, though did not finish. I moved the ticket back to the bottom of the 'To-Do' column, so that if time permits at end of day I will look at it again. Moving onto the next Product feature....
- Need consistency in CSS stylesheets
- Need to remove the auth favicon on login page

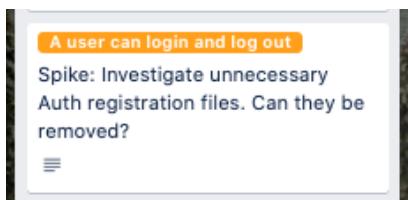


FEATURE: A user can login and log out

- I removed the User Registration aspects as this is out of scope for Quiz Manager.
- Disabled the registration, password reset, and email verification routes
- Run `php artisan route:list` to see a list of routes active

Linna-Trieu-SP/quiz-manager on ✓ auth-login-and-logout [!] via • v12.10.0 via 🌐 v7.3.11					
+ php artisan route:list					
Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	api/user		Closure	api
					auth:api
	GET HEAD	home	home	App\Http\Controllers\HomeController@index	web
					auth
	GET HEAD	login	login	App\Http\Controllers\Auth\LoginController@showLoginForm	web
					guest
	POST	login		App\Http\Controllers\Auth\LoginController@login	web
					guest
	POST	logout	logout	App\Http\Controllers\Auth\LoginController@logout	web
	GET HEAD	password/confirm	password.confirm	App\Http\Controllers\Auth\ConfirmPasswordController@showConfirmForm	web
					auth
	POST	password/confirm		App\Http\Controllers\Auth\ConfirmPasswordController@confirm	web
					auth

- Considering if I should I remove the unnecessary files (incl. view and controllers)? Presumably in a later version of the application, registration will eventually be used on the Quiz Manager; so perhaps I shouldn't delete the files; and I can leave them in a 'disabled' state. I created a ticket in the backlog to investigate, and I moved



- Set up the Eloquent model class - User ✓
 - The Eloquent model class outlines the data stored in the User table associated
 - I decided to use the Enum data type for the user permission level
 - In the code, I created another class called UserPermission which represents the enum

- **What are Enums and Why use them for User Permission Levels?**

- Enums are lists of constants like unchangeable variables.
- It is best practice! When you need a predefined list of values which do represent some kind of numeric or textual data, you should typically use an enum.
- You should always use enums when a variable (especially a method parameter) can only take one out of a small set of possible values.
- In Quiz Manager, I have a pre-defined list of possible user permission levels “{Restrict, View, Edit}”. This is the ideal use case for an enum and best practice!
- Using enums (instead of, say, a listing of strings) improve compile time, performance and reduce errors from passing in invalid arguments or constants (type hinting) .

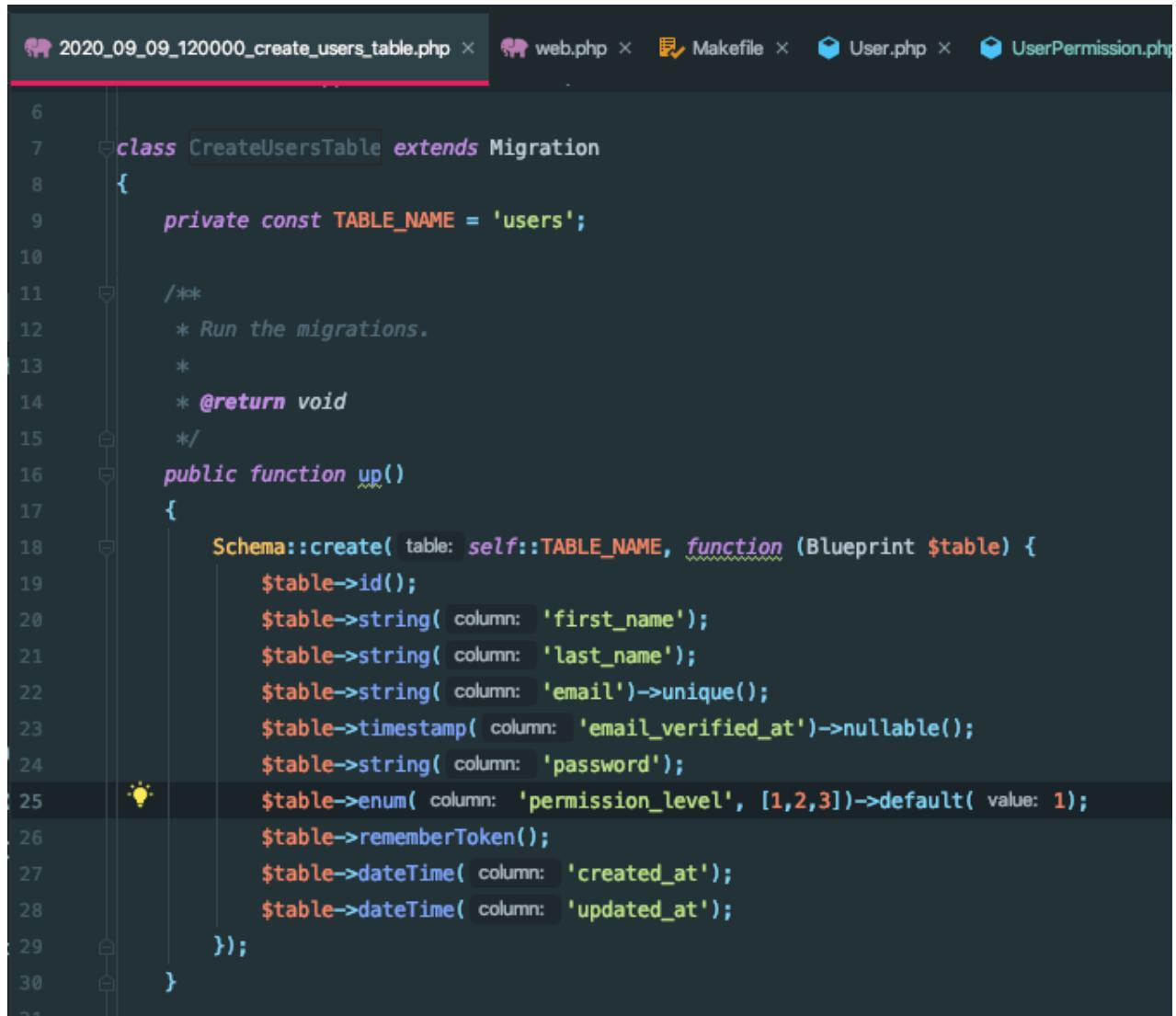
```
<?php

use \MyCLabs\Enum\Enum;

/**
 * UserPermission Enum
 * @method static self PERMISSION_RESTRICT()
 * @method static self PERMISSION_VIEW()
 * @method static self PERMISSION_EDIT()
 */

class UserPermission extends Enum
{
    public const PERMISSION_RESTRICT = 1;
    public const PERMISSION_VIEW = 2;
    public const PERMISSION_EDIT = 3;
}
```

- In the database migration, I set the type as ‘enum’.
- This field has a default value of 1 (RESTRICT). I defaulted to 1 as this is the lowest permission level for a known user, and therefore this is conservative.



```
6
7     class CreateUsersTable extends Migration
8     {
9         private const TABLE_NAME = 'users';
10
11        /**
12         * Run the migrations.
13         *
14         * @return void
15         */
16        public function up()
17        {
18            Schema::create( table: self::TABLE_NAME, function (Blueprint $table) {
19                $table->id();
20                $table->string( column: 'first_name');
21                $table->string( column: 'last_name');
22                $table->string( column: 'email')->unique();
23                $table->timestamp( column: 'email_verified_at')->nullable();
24                $table->string( column: 'password');
25                $table->enum( column: 'permission_level', [1,2,3])->default( value: 1);
26                $table->rememberToken();
27                $table->dateTime( column: 'created_at');
28                $table->dateTime( column: 'updated_at');
29            });
30        }
31    }
```

Set up User Factory

- I created a factory class for users.
- Following the Laravel OOP Factory Pattern - which is to create objects like a factory by exposing some kind of interface to the outside world like createXYZ.
- With this, I can define a pattern to quickly generate fake models and build “dummy” data for use in seed data and in automated testing!
- I am using Faker for my factory class. This is a PHP library that generates fake data. By default the third-party package is included in composer dependencies - a reliable, and industry-used package)

```
9  *-----*
10 |-----|
11 | Model Factories
12 |-----|
13 |
14 | This directory should contain each of the model factory definitions for
15 | your application. Factories provide a convenient way to generate new
16 | model instances for testing / seeding your application's database.
17 |
18 */
```

```
20 $factory->define( class: User::class, function (Faker $faker) {
21     return [
22         'first_name' => $faker->firstName,
23         'last_name' => $faker->lastName,
24         'email' => $faker->unique()->safeEmail,
25         'email_verified_at' => now(),
26         'password' => '$2y$10$92IXUNpkjO0rQ5byMi.Ye4oKoEa3RqOllC/.og/at2.uheWG/igi',
27         'remember_token' => Str::random( length: 10),
28         'permission_level' => random_int(1, 3),
29     ];
30 });
31 */
```

Run User seeder

- Generated db data by running a seeder command. This created an arbitrary amount of users => 10 in the users table.

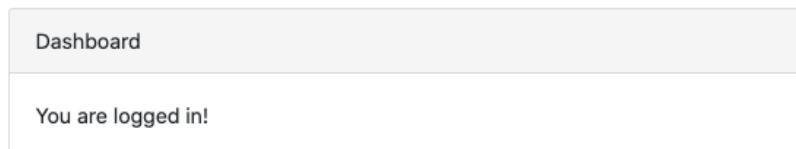
Created the UserController

- Added an Exception handler in case a user tries to be registered.

Testing: A User can login and log out

- Through manual testing - I discovered I cannot log out of the Website. Once logged in, there is no option listed in the navbar.

Quiz Manager

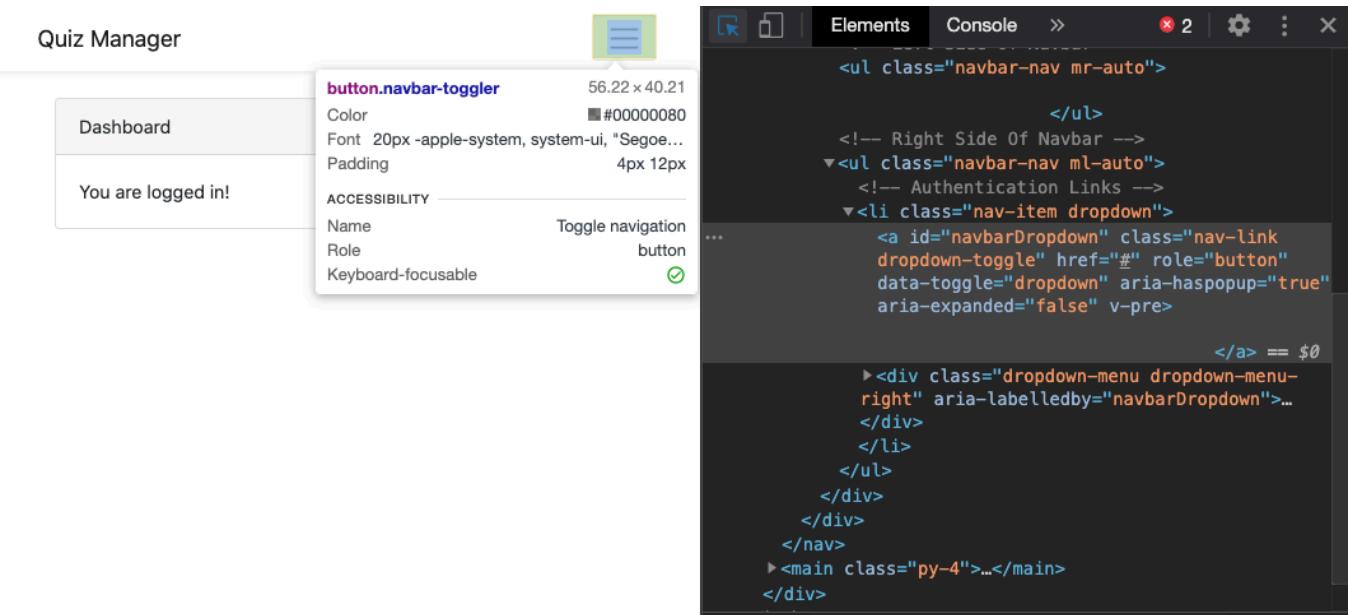


Debugging

HYPOTHESIS APPROACH - This could be due to a few potential issues. Let's go through them (in order of what is most likely the problem):

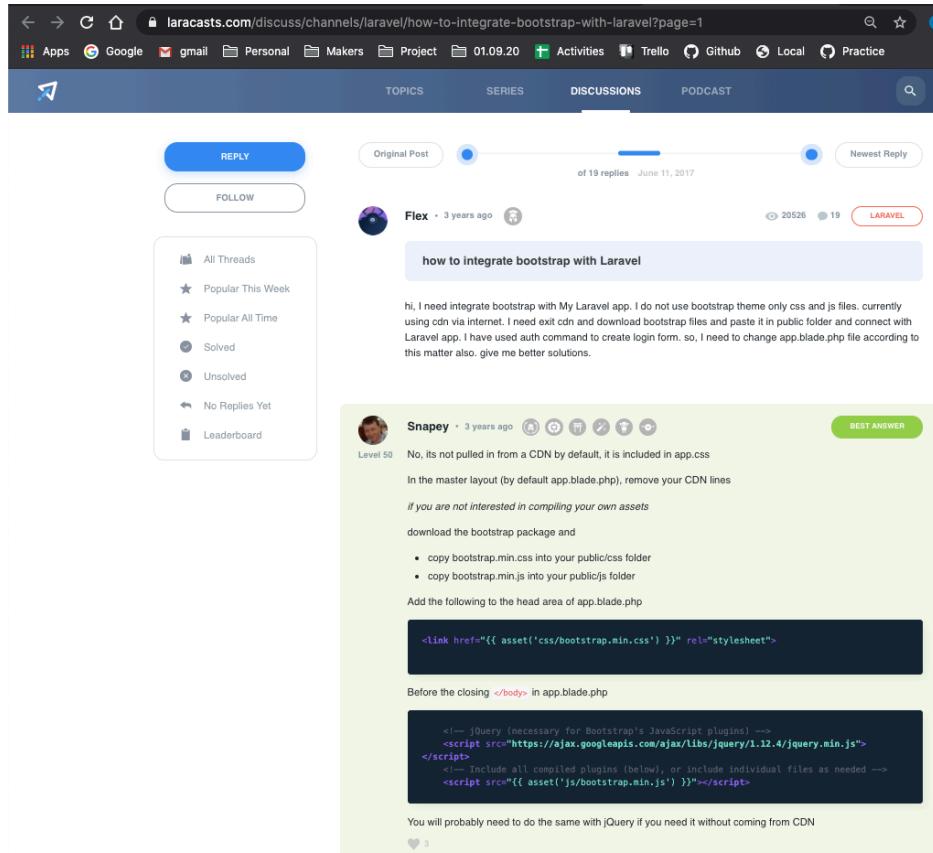
1. The Navbar toggler does not work. The logout option is there... But perhaps I just can't expand the navbar dropdown list to see it.
2. There is something wrong with the Auth logout route - maybe it does not exist? Did something happen in one of the Auth controllers?
3. The logout hyperlink is not appearing due to a HTML issue on the front end
4. Not an exhaustive list of hypotheses. Search for more if none of the above hypotheses are found true.

Outcome: Hypothesis #1 = TRUE ✓ This was identified as cause of the Logout issue.



Investigating Hypothesis #1 - Process

- There is no interactivity with the HTML element called 'button navbar toggler'.
- Why? This led me to think it could be a JavaScript issue; also it is the first time I've looked into JS on this project scaffold so far, so it could be likely not implemented correctly.... let's look into this further.
- I identified where the Bootstrap JavaScript files are referenced in the master blade layouts file. It seems to include the Bootstrap JS file, and looks ok - no errors in the syntax.
- But I realised the HTML file was missing a reference to the jQuery library (I had assumed it was included by default)!
- Determined through internet investigation and troubleshooting
- I added a reference to the jQuery CDN (see line 81) = FIXED the issue 🎉 The drop down list now appears when the navbar toggler is selected!
- As I have now determined the cause of the issue - I can now refactor!
- I cannot reference the jQuery CDN, as it hosted externally on cloud. I need to download so I can reference files locally! 🤸
- Added a Trello ticket to the backlog so I can move on to higher priority tickets!! 🚀



The screenshot shows a forum post on laracasts.com. The post is titled "how to integrate bootstrap with Laravel". The original poster, Flex, asks for help integrating Bootstrap into their Laravel application. They mention not using a Bootstrap theme and instead using local files. Snapy, another user, provides a detailed answer. Snapy suggests removing CDN links from the master layout (app.blade.php) and instead copying Bootstrap's CSS and JS files to the public folder. They provide code snippets for adding the CSS link and including the JS script. The answer is marked as the best answer.

```

<link href="{{ asset('css/bootstrap.min.css') }}" rel="stylesheet">
<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js">
</script>
<!-- Include all compiled plugins (below), or include individual files as needed -->
<script src="{{ asset('js/bootstrap.min.js') }}></script>

```

```

Linna-Trieu-SP [~/personal_projects/Linna-Trieu-SP] - .../quiz-manager/resources/views/layouts/master.blade.php
quiz-manager > resources > views > layouts > master.blade.php > ADD CONFIGURATION... > Git: 
Project master.blade.php × DatabaseSeeder.php × UserSeeder.php × register.blade.php × RegisterController.php × README.md ×
1: 1; 70
70      </ul>
71      </div>
72    </div>
73  </nav>
74
75  <main class="py-4">
76    @yield('content')
77  </main>
78</div>
79
80  <!-- jQuery (necessary for Bootstrap's JavaScript plugins) --&gt;
81  &lt;script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"&gt;&lt;/script&gt;
82  <!-- Bootstrap's JavaScript library --&gt;
83  &lt;script type="text/javascript" src="{{ asset('bootstrap/js/bootstrap.min.js') }}&gt;&lt;/script&gt;
84
85&lt;/body&gt;
86&lt;/html&gt;
87
</pre>

```

Quiz Manager

Dashboard

You are logged in!

Logout

Elements

```

...
  ▼<div class="dropdown-menu dropdown-menu-right show" aria-labelledby="navbarDropdown"> == $0
    ▶<a class="dropdown-item" href="http://127.0.0.1:8000/logout" onclick="event.preventDefault(); document.getElementById('logout-form').submit();">...

```

Login Feature Tests

- Created file ‘LoginTest.php’
- Enable DB config for tests in phpunit.xml
 - Using an sqlite in-memory database (rationale explained in file ref: 3.1. *Testing Document*.
- Before writing any test code, I plan out the testing approach by pseudo-coding. What are the aspects of functionality I want to validate and test are working correctly? For example, these are the ‘Login’ test features I thought I could test out.

Testing login features

- user can view the login form
- user cannot view the login form when already signed in to the website
- user can login with correct user credentials
- user cannot login with a non-existent email
- user cannot login with incorrect user credentials
- user can logout, when already signed in to the website
- user has the remember-me cookie, if elects to be remembered

PHP and Testing Conventions

- Per PHPUnit manual and PSR-12 standard: “*Method names MUST be declared in camelCase*”.
- I have followed Test naming conventions in naming my tests in camelCase.
- There was an issue with one Feature test failing, which I was not expecting to fail... why?
I researched the unexpected behaviour on stack overflow - reason: the config was cached.

LINNA TRIEU: DEVELOPMENT DIARY

The screenshot shows a developer's workspace with two main windows. On the left is a code editor displaying PHP test code for a login feature. The code uses Laravel's Artisan facade to create a user, post to the login route with correct credentials, and assert a redirect to the home page. A terminal window below the editor shows the command run: php artisan test --filter testUserCanLoginWithCorrectCredentials. The output indicates 1 failed test. On the right is a web browser window showing a Stack Overflow question about PHPUnit expected status code issues. The question asks why tests fail when configuration files are cached. It includes an explanation from David Laruey, a snippet of code showing config:clear being called via Artisan, and a link to a GitHub issue for Laravel framework. The browser also shows the developer's profile and activity.

```
public function testUserCanLoginWithCorrectCredentials(): void
{
    $user = factory(\App\Models\User::class)->create([
        'password' => Hash::make('password')
    ]);

    $response = $this->post('/login', [
        'email' => $user->email,
        'password' => $password,
        'first_name' => $user->first_name,
        'last_name' => $user->last_name,
    ]);

    $response->assertRedirect('/home');
    $this->assertAuthenticatedAs($user);
}
```

Terminal: Local x

```
Linna-Trieu-SP ~/quiz-manager on M auth-login-and-logout [1] via wsl2:10.0 via ~ v7.3.11
make test ARGS="--filter testUserCanLoginWithCorrectCredentials"
php artisan test --filter testUserCanLoginWithCorrectCredentials

FAIL Tests\Feature\LoginTest
  × user can login with correct credentials

Tests: 1 failed

Response status code [500] is not a redirect status code. Failed asserting that false is true.

at tests/Feature/Auth/LoginTest.php:49
  45|         'last_name' => $user->last_name,
  46|
  47|     ]);
  48|
> 49|     $response->assertRedirect('/home');
  50| }
```

Stack Overflow Products Public Search... Home PUBLIC Stack Overflow Tags Users FIND A JOB Jobs Companies TEAMS Paddle

1 This works, except in cases where a Middleware is actually part of your tests. — David Laruey Mar 26 '19 at 15:25 add a comment

Solution: When you cached your configuration files you can resolve this issue by running `php artisan config:clear`.

Explanation: The reason why this can resolve the issue is that PHPUnit will use the cached configuration values instead of the variables defined in your testing environment. As a result, the `APP_ENV` is not set to testing, and the `VerifyCsrfTokenMiddleware` will throw a `TokenMismatchException` (Status code 419).

It is recommended not to cache your configuration in your development environment. When you need to cache your configuration in the environment where you are also running unit tests you could clear your cache manually in your `TestCase.php`:

```
use Illuminate\Support\Facades\Artisan;
public function createApplication()
{
    ...
    Artisan::call('config:clear')
    ...
}
```

Example based on <https://github.com/laravel/framework/issues/13374#issuecomment-239600163>

Read more about configuration caching in this [blog post](#) or the [Laravel documentation](#).

share edit follow edited Jun 19 at 9:22 answered Jan 16 '18 at 12:41 picator 4,563 3 15 27

FEATURE: A user can view the list of quizzes

- **Create Quiz table**

- Migration to create new Quiz table.
- It's good practice to ensure consistency in naming conventions.
 - I made the decision to keep singular names for my Objects, instead of plurals. E.g. Quiz instead of Quizzes, User instead of Users
 - This should be consistent in naming convention across the code, between model objects and the db table name!
 - Ensures maintainability and easier to code with - make a code decision and stick with it.

- **Added Quiz model**

- Added Quiz factory
- Added Quiz seeders
- Create Quiz controller
- For some reason php artisan cli tool (or the Application) does not recognise my model Quiz as a valid model..... concerning. Moving on from the issue (but noting this) as I am not 100% if this is an issue yet.

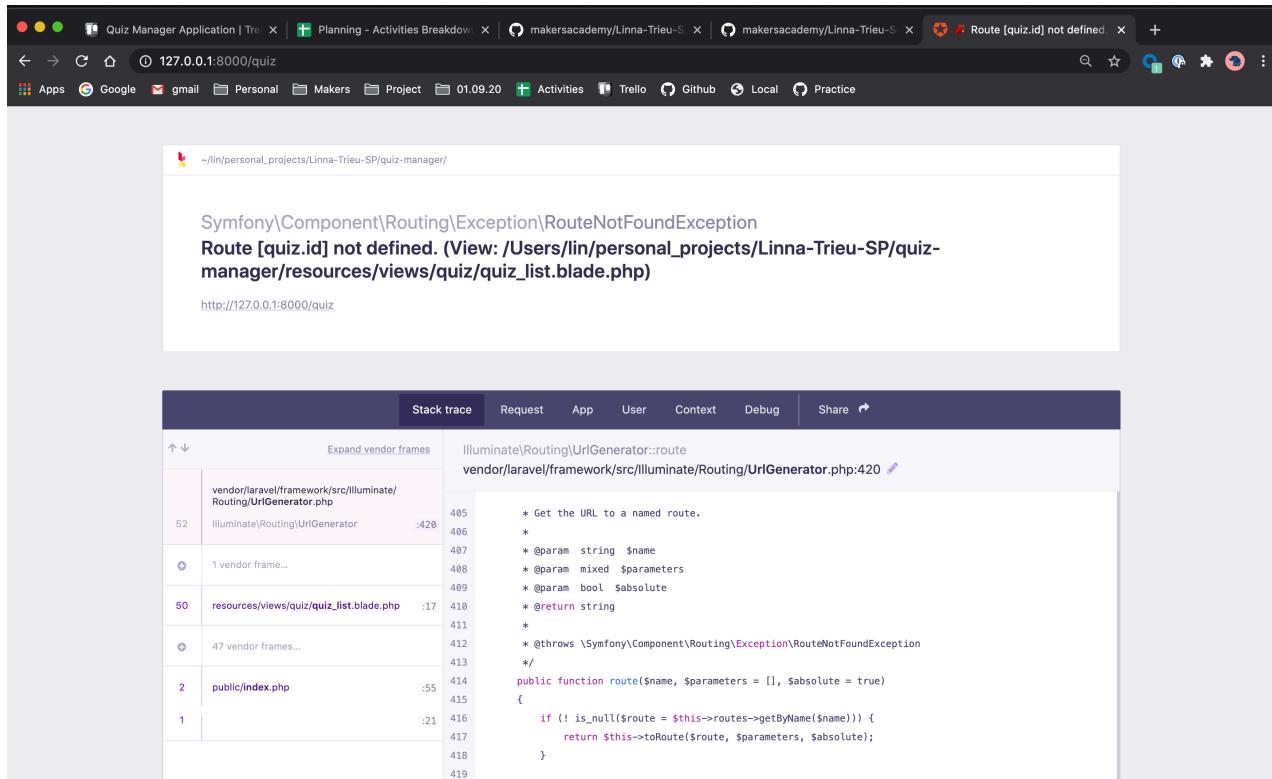
```
Linna-Trieu-SP/quiz-manager on 7 view-list-of-quizzes [!] via • v12.10.0 via ⚡ v7.3.11
→ php artisan make:controller QuizController --resource --model=Quiz

A App\Quiz model does not exist. Do you want to generate it? (yes/no) [yes]:
> ^C

Linna-Trieu-SP/quiz-manager on 7 view-list-of-quizzes [!] via • v12.10.0 via ⚡ v7.3.11 took 3s
→ php artisan make:controller QuizController --resource
Controller created successfully.
```

- Created the Quiz List page with BLADE 🎉
- Added new HTTP GET route for the Quiz List page

- Troubleshooting the connection with view-route-controller...issue with the variable name. Think it is just a typo.



Symfony\Component\Routing\Exception\RouteNotFoundException
Route [quiz.id] not defined. (View: /Users/lin/personal_projects/Linna-Trieu-SP/quiz-manager/resources/views/quiz/quiz_list.blade.php)
http://127.0.0.1:8000/quiz

```

Stack trace Request App User Context Debug Share ↗
Expand vendor frames
vendor/laravel/framework/src/Illuminate/Routing/UrlGenerator.php:420
Illuminate\Routing\UrlGenerator::route
vendor/laravel/framework/src/Illuminate/Routing/UrlGenerator.php:420
    * Get the URL to a named route.
    *
    * @param string $name
    * @param mixed $parameters
    * @param bool $absolute
    * @return string
    *
    * @throws \Symfony\Component\Routing\Exception\RouteNotFoundException
    */
public function route($name, $parameters = [], $absolute = true)
{
    if (! is_null($route = $this->rout...

```

- Fixed 🎉 I can now see a View of the Quiz List page!!

- End of day 2 progress!! ✓
- Website so far

The screenshot shows a web browser window titled "Quiz Manager". The address bar displays the URL "127.0.0.1:8000/quiz". The main content area is titled "Quizzes" and contains five items listed in blue: "Quiz 1 - cumque corporis praesentium", "Quiz 2 - ipsum enim animi", "Quiz 3 - cum cum et", "Quiz 4 - ducimus incident cupiditate", and "Quiz 5 - et et inventore". A blue button labeled "Add a new Quiz" is located at the top right of the content area.

START OF DAY 3 - 10/09/20

- **Agile Ways of Working - Sprint Planning**
 - * Reviewed the Trello board, pulled new tickets from backlog into the current sprint, and identified the overarching Goals & Objectives for the Sprint.
 - * Thoughts & Observations: there is lots of tickets in the ‘Ready for Testing’ column. Immediate focus is to move tickets from ‘left to right’ of the board as quickly as possible.
 - * Will spend time this morning focused on testing and moving tickets into ‘Done’
 - * Then will move onto the tickets currently In Progress, and finally pick up new tickets in the To Do column.
-
- Testing 🚧
 - **Feature: A user can log-in and log-out**
 - Bugs identified:
 1. Logout button on the landing page “/“ does not work. When selected the user is taken to the correct endpoint “logout” but shown an error page. => this should be fixed, blocking issue

Quiz Manager

QUIZZES [LOGOUT](#)

```
<html>
  > <head>...</head>
  > <body>
    ><div class="flex-center position-ref full-height">
      ><div class="content">
        ><div class="title m-b-md">
          > Quiz
          > Manager
        </div>
        ><div class="links">
          ><a href="http://127.0.0.1:8000/quiz">
            > Quizzes </a>
          ><a href="http://127.0.0.1:8000/logout">
            > Logout </a> == $0
          </div>
        </div>
      </div>
    </body>
</html>
```

html body div.flex-center.position-ref.full-height div.content div

Styles Computed Event Listeners DOM Breakpoints Properties

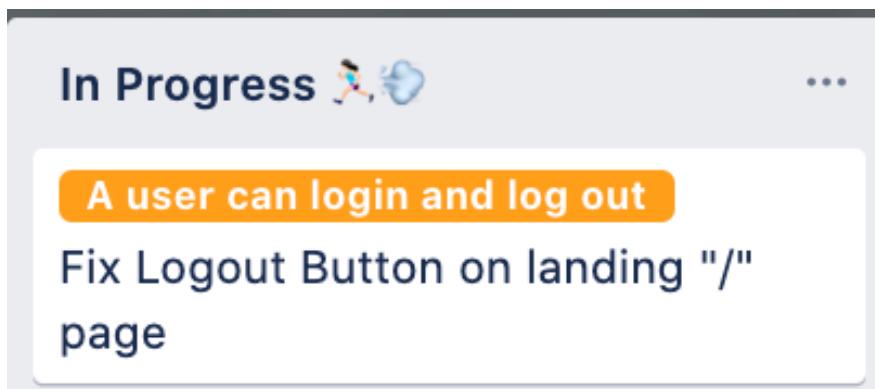
Filter

```
element.style { }
}
.links > a {
  color: #636b6f;
  padding: 0 25px;
```

Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException
The GET method is not supported for this route. Supported methods: POST.

<http://127.0.0.1:8000/logout>

		Stack trace	Request	App	User	Context	Debug	Share ↗	
↑ ↓		Expand vendor frames							
		Illuminate\Routing\AbstractRouteCollection::methodNotAllowed vendor/laravel/framework/src/Illuminate/Routing/AbstractRouteCollection.php:117 ↗							
26	vendor/laravel/framework/src/Illuminate/Routing/AbstractRouteCollection.php	:117	102						
25	Illuminate\Routing\AbstractRouteCollection	:103	103	\$this->methodNotAllowed(\$methods, \$request->method());					
24	Illuminate\Routing\AbstractRouteCollection	:40	104	}					
	21 vendor frames...		105	/**					
2	public/index.php	:55	106	* Throw a method not allowed HTTP exception.					
1		:21	107	*					
			108	* @param array \$others					
			109	* @param string \$method ↗					
			110	* @return void					
			111	*/					
			112	* @throws \Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException					
			113	*/					
			114	protected function methodNotAllowed(array \$others, \$method)					
			115	{					
			116	throw new MethodNotAllowedHttpException(
			117	\$others,					
			118	sprintf(
			119	'The %s method is not supported for this route. Supported methods: %s.',					
			120	\$method,					
			121	implode(', ', \$others)					
			122)					
			123);					
			124	}					
			125	}					



- Fixed Issue and moved ticket into 'Done'.
 - I googled the error message
 - I submitted a GET request on the 'logout' route instead of a POST request.
 - To fix, I used jQuery to submit an on click event for when the logout button is clicked, make a POST action to the logout route
- Fixed Logout functionality on the landing page

```
@auth
    <a href="{{ url('quiz') }}> View Quizzes </a>
    <a href="{{ route('logout') }}" onclick="event.preventDefault();document.getElementById('logout-form').submit();">{{ __('Logout') }}</a>
    <form id="logout-form" action="{{ route('logout') }}" method="POST" class="d-none">@csrf</form>
@endauth
</div>
</div>
</div>
```

2. On other pages, the Logout option is functional and works. But it is difficult for a user to find - bad UX design. The Logout option is within a navbar drop down list - but this provides no value and can confuse the UX. Not immediately apparent where a user can go to logout themselves out.
 - This should be fixed, but is not a priority as the logout button is still working. Front end and UX needs to be updated. Created a new ticket in backlog.
- **Outcome: Feature “A user can log-in and log-out” = Done** ✓  

QUIZ MANAGER

[Add a new Quiz](#)

Quizzes
Quiz 1 - aut amet est
Quiz 2 - eos qui dolorum
Quiz 3 - sint assumenda eligendi
Quiz 4 - nostrum illum nam
Quiz 5 - deserunt qui et

QUIZ MANAGER

The screenshot shows a web-based application titled "QUIZ MANAGER". At the top right is a "LOGOUT" button. Below it is a blue button labeled "Add a new Quiz". The main content area is titled "Quizzes" and contains a list of five items:

- Quiz 1 - aut amet est
- Quiz 2 - eos qui dolorum
- Quiz 3 - sint assumenda eligendi
- Quiz 4 - nostrum illum nam
- Quiz 5 - deserunt qui et

The screenshot shows a backlog item with the title "Backlog 🎵". To the right of the title is a three-dot menu icon. Below the title is a yellow button containing the text "A user can login and log out". The main text of the backlog item reads: "UI: Move the Logout button into the navbar so it is visible, remove navbar toggler." A vertical line on the right side separates this backlog item from the rest of the page.

- **Feature: A user can view answers to questions**

- This feature requires planning and coding decisions: The logic is more complex with this feature. Previous to this the website has been static, and only using HTML and CSS on the front-end.
- Option: Server side code or Client Side Code ?

- **Desired Behaviour**

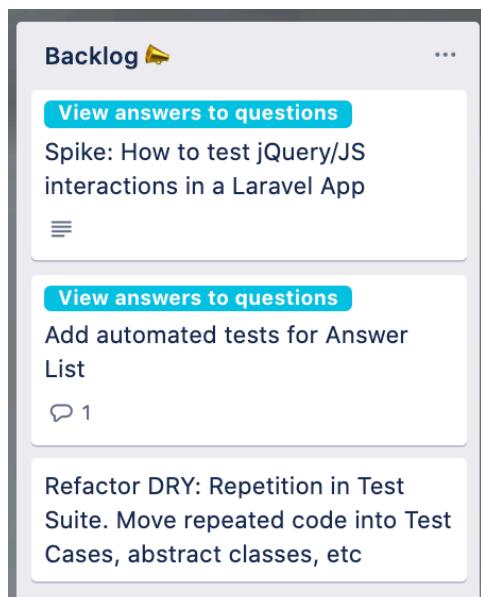
- NOTE: this is not a matter of security. As all edit and view users have ‘permission’ to see answers anyway. This is merely a matter of the default behaviour (for either user!) They both can access the answers so it is not a issue to have the answer still visible in the DOM (ie. if the page is inspected - the user will be able to see the answer key). Thus jQuery option seems good to proceed with!
- With this, I need to handle logic that deals with ‘events’ and interactivity. i.e. by default the answers to the quiz will not be displayed on Quiz Manager. However if a user wishes to see the answer (and have the relevant permissions to do so - though this is not part of the MVP scope yet), they can choose to reveal the answers.
- Based on my initial wireframes my initial design decision was to hide answer by default with a button ‘Reveal Answers’. If the button was clicked by a user, the answers would display on the same page, immediately below its corresponding question.

- **Outcome:**

- This requires interactivity, and therefore JavaScript. In my opinion, the easiest way to achieve the desired behaviour is using jQuery, a popular JS library. As I have previous experience with jQuery and under immediate project time constraints, using a javascript library (instead of vanilla javascript), would likely be faster to write code for.
- With jQuery I handled the logic to hide the answers by default, when a user visits the ‘/quiz’ endpoint. But when requested, the answers are displayed.
- I could have otherwise proceeded with a number of other JavaScript libraries.
- Was there a SERVER-SIDE CODE option?
- > Future feature (if more time): Added a button which shows the Options? But would you want to hide the options by default? Perhaps not.

- **Tests**

- Couldn't figure out how to test jQuery / JS functionality - particularly as this feature is dependent on the user interactivity and the hide/show function.
- I created a new ticket to spike (investigate) how to test these types of features. It is likely that PHPUnit is not enough, I may need to find a PHP Test Suite Library that tests browser interactions, for example using Selenium Driver[. Dusk?](<https://laravel.com/docs/7.x/dusk>)
- I performed Manual testing on the browser to verify the work done. I am happy the Product Feature goal has been met.
- Automated testing needs to be added.
- To move on and proceed with developing the next product feature, I created the Spike ticket and added a note the Automated Test ticket is blocked by the spike. Moved these tickets from 'In Progress' into the 'Backlog' column, and moved on.



- **MVP => DONE ✓**

DAY 4 - 11/09/20

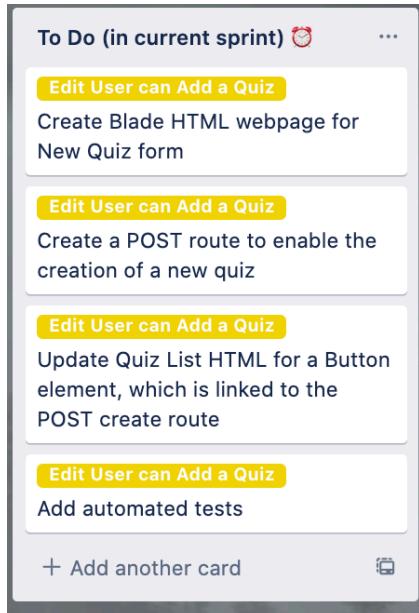
Feature: Implement restricted user permissions, so that a restricted user can only see quiz questions

- Add Seed Data for Users
- The application is configured with three test accounts, with varying levels of permissions.
- Where does the logic sit to determine what can/can't be showed to a restricted user - BE/FE, Server/Client, MVC?
- Logically it seems sensible to have this logic sit in the Back-End, specifically the controller.
- With SOLID and separation of concerns best practice, the front end ('View' in MVC) should not be responsible for too much. This logic should sit in the back-end to determine the user permission (C in MVC).
- Additionally handling this logic on the client-side would be of higher security concern. It would be easy to manipulate the client-side logic, and bypass the user permissions.

- Considerations:
 - Implement user permission system/control flow in the back end controllers.
 - The views/front end should not handle any user permission logic. The filtered information (post control flow, logic handling) should be only passed on to the views (for it to display).

Feature: An edit user can add a quiz

- Planned the user story, breaking down the feature into logical tickets.



- I am following RESTful best practice with naming conventions of resources. e.g. laravel docs & REST API



Laravel

- [Prologue](#)
- [Getting Started](#)
- [Architecture Concepts](#)
- [The Basics](#)
- [Routing](#)
- [Middleware](#)
- [CSRF Protection](#)
- [● Controllers](#)
- [Requests](#)
- [Responses](#)
- [Views](#)
- [Session](#)
- ...

of these actions, including notes informing you of the HTTP verbs and URIs they handle.

Actions Handled By Resource Controller			
Verb	URI	Action	Route Name
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

Specifying The Resource Model

If you are using route model binding and would like the resource controller's methods

- Added a new route to create a Quiz = `/quiz/create`
- Created a new blade HTML template for the new quiz form.
- Added a button on the existing Quiz List page which would link to the new quiz form.
- Created a QuizController method ‘create’ to handling the creation of a new Quiz and save into the database. Once created, the user should be re-directed to the Quiz List page, with their new quiz displayed.
- Added auth checks as only an Edit user has the permissions to add a quiz
- Added automated tests to validate this.

Feature ✓

Feature: An edit user can delete a quiz

- Planned the product feature, breaking down the feature into logical tickets.

Context/background

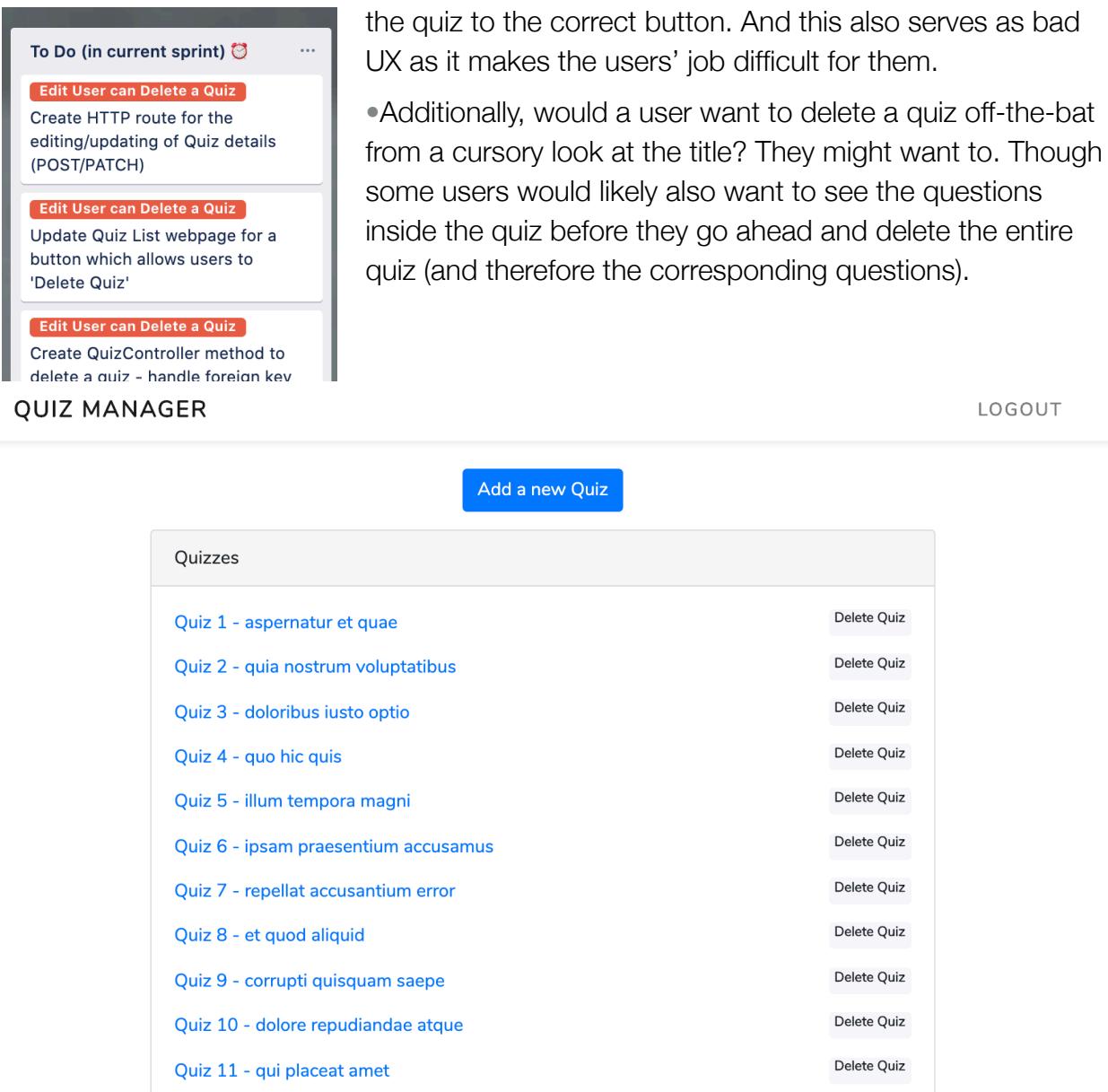
- As declared in my initial project design doc (business requirements and assumptions section), I made the assumption that when a user deletes a quiz, this also means they delete all of its corresponding questions and answers. This assumption seemed reasonable as I also made the assumption that every question belonged to a quiz (i.e. a single question couldn't exist by itself, it was always part of a quiz).
- This business assumption was already implemented during the database design phase of the project. In constructing the database and tables, the Questions table has a foreign key with the Quiz table in its quiz_id field, as a question has a many-to-one relationship with a quiz.

Options

- Can delete a resource in many ways.
- Using jQuery as an onClick event or using the HTTP method verbs (delete), etc. Elected to go with HTTP verb in order to keep it simple as possible (and consistent with the other CRUD actions so far).

1. Implement front end design as with the initial wireframes

- Updated the Quiz List webpage for a button alongside the quiz itself to delete.
- The design of this UI gave me some doubt as for the UX. A user could accidentally click the button for an incorrect quiz, if they could not visually align the quiz to the correct button. And this also serves as bad UX as it makes the users' job difficult for them.
 - Additionally, would a user want to delete a quiz off-the-bat from a cursory look at the title? They might want to. Though some users would likely also want to see the questions inside the quiz before they go ahead and delete the entire quiz (and therefore the corresponding questions).



The screenshot shows a web-based application titled "QUIZ MANAGER". At the top left is a sidebar labeled "To Do (in current sprint)" with three items:

- Edit User can Delete a Quiz**: Create HTTP route for the editing/updating of Quiz details (POST/PATCH)
- Edit User can Delete a Quiz**: Update Quiz List webpage for a button which allows users to 'Delete Quiz'
- Edit User can Delete a Quiz**: Create QuizController method to delete a quiz - handle foreign key

At the top right are "LOGOUT" and "QUIZ MANAGER" buttons. The main content area has a header "Add a new Quiz" and a table titled "Quizzes" containing 11 rows of quiz titles and "Delete Quiz" buttons:

Quizzes	
Quiz 1 - aspernatur et quae	Delete Quiz
Quiz 2 - quia nostrum voluptatibus	Delete Quiz
Quiz 3 - doloribus iusto optio	Delete Quiz
Quiz 4 - quo hic quis	Delete Quiz
Quiz 5 - illum tempora magni	Delete Quiz
Quiz 6 - ipsam praesentium accusamus	Delete Quiz
Quiz 7 - repellat accusantium error	Delete Quiz
Quiz 8 - et quod aliquid	Delete Quiz
Quiz 9 - corrupti quisquam saepe	Delete Quiz
Quiz 10 - dolore repudiandae atque	Delete Quiz
Quiz 11 - qui placeat amet	Delete Quiz

2. I created a new ‘DELETE’ route which was invoked when a user selected the ‘Delete Quiz’ button. This triggered a QuizController method ‘destroy’ to remove the quiz from the db.

- However when the button was selected an error occurred: “*integrity constraint violation foreign key constraint*”

127.0.0.1:8000/quiz/1

~/lin/personal_projects/Linna-Trieu-SP/quiz-manager/

Illuminate\Database\QueryException
SQLSTATE[23000]: Integrity constraint violation: 1451 Cannot delete or update a parent row: a foreign key constraint fails
(`quiz_manager`.`question`, CONSTRAINT `question_quiz_id_foreign` FOREIGN KEY (`quiz_id`) REFERENCES `quiz` (`id`)) (SQL: delete from `quiz` where `id` = 1)

<http://127.0.0.1:8000/quiz/1>

		Stack trace	Request	App	User	Context	Debug	Share	
↑ ↓		Expand vendor frames	Illuminate\Database\Connection::runQueryCallback vendor/laravel/framework/src/Illuminate/Database/Connection.php:671						
55	Illuminate\Database\Connection	:671	656 657	<pre>* @throws \Illuminate\Database\QueryException */ protected function runQueryCallback(\$query, \$bindings, Closure \$callback) { // To execute the statement, we'll simply call the callback, which // will then handle the SQL command via PDO command. Then we'll bind // the values to the command and return the result. }</pre>					
54	Illuminate\Database\Connection	:631	658 659						
53	Illuminate\Database\Connection	:496	660 661						

- 3. Debugging....And a fix! 🎉
- I researched online for resources to debug. Example screenshot below.

How to delete a foreign key constraint in Laravel?

Asked 1 year, 5 months ago Active 1 year, 5 months ago Viewed 3k times

I'm working on a project in which I'm adding data in two tables **User & Registration** in one form like this

```
public function storeStudent(Request $request)
{
    $created_user = User::create([
        'name' => $request->name,
        'email' => $request->email,
        'password' => bcrypt($request->password),
        'parent_id' => $request->parent_id,
        'role_id' => $request->role_id,
        'gender'=> $request->gender,
        'date_of_birth'=> $request->date_of_birth,
        'cnic'=>$request->cnic,
        'religion'=>$request->religion,
        'skills'=>$request->skills,
        'physical'=>$request->physical,
        'emergency_name'=>$request->emergency_name,
        'phone_no'=>$request->phone_no,
        'medical_info'=>$request->medical_info,
        'family_dr'=>$request->family_dr,
        'address'=>$request->address,
    ]);

    $registration = Registration::create([
        //reg_id is id for registration table
        'user_id' => $created_user->id,
        'class_id' => $request->class_id,
        'section_id' => $request->section_id,
    ]);
}
```

Now I want to delete the data I'm bad with syntaxes. I don't know what should I do to delete the data I'm trying

1 Answer

Active Oldest Votes



```
$table->integer('user_id')->unsigned();
$table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
```

2

Using `->onDelete('cascade')` works for me :)



share edit follow



add a comment

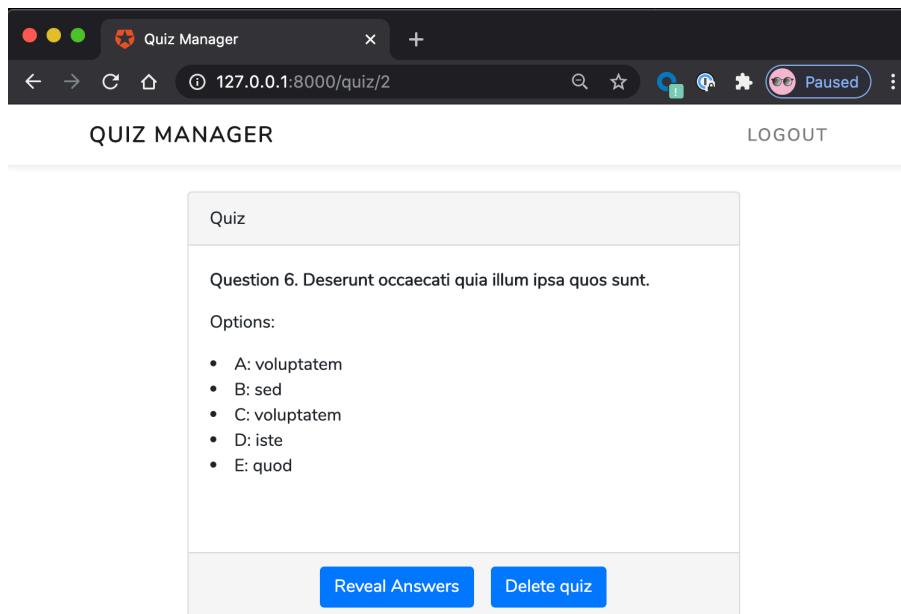
answered Apr 14 '19 at 18:39

Abdul Rehman
99 ● 1 ● 13

- I updated the migration so that the foreign key deleted on cascade!
- Manually tested and the Quiz deleted!
- I checked the db to confirm the questions also deleted => the feature works!  

Now that the Delete feature is functional => time for refactoring and cleanup

- I don't like the Delete Quiz option on the Quiz List page. I think a better UX would be to move the button on the individual Quiz page.
- In this next iteration, I will move the Delete functionality to the Quiz page and see how it feels from a UX perspective.
- The functionality should not have changed at all.
- Experimented with design option and I preferred this design.
 - Decided to implement this approach instead, with the Delete Quiz option only available on the specific quiz page itself. e.g. at endpoint '/quiz/2' instead of '/quiz'. I removed the previous Delete code button from the Quiz List webpage.



- Added automated tests
 - Relied on the `assertDatabase` contains
 - Not sure if using the DB in a feature test is a code smell or best practice? Included for now as it seemed like a good way to validate if the delete was working or not.
- Feature complete   Moving on

DAY 4, 14/09/20

- Re-slicing the remaining User Stories as I learn more about the implementation

Project Sprints / Slicing  ...

- An edit user can add a question
- An edit user can update a question
- An edit user can delete a question
- An edit user can add an answer
- An edit user can update an answer
- An edit user can delete an answer

- There are 6 remaining user stories
- But with the current implementation, it doesn't make sense for me to use this arbitrary split between questions and answers.
- The user's functionality on editing either a question or an answer will likely be done in the same manner from a front end and back end perspective.
- I decided to re-evaluate these slices and compress them into 3 remaining slices.

Project Sprints / Slicing  ...

- An edit user can add a question or answer
- An edit user can update a question or answer to change its text
- An edit user can delete a question or answer

 PROJECT COMPLETE 

- In the daily sprint planning session, I will create the tickets required in the backlog for these slices.
 - Slow morning today. Yikes.... I need to get a kick on with the actual coding.
 - This morning I've spent a lot of time in planning. I assessed the priorities of the 'Tech Debt' column and decide what tickets (if any) are blocking.
 - Based on this, I decided to bring in a number of tickets into the current sprint as they were critical from a user requirement perspective. The remaining tickets were assessed as nice to have, so stayed in the tech debt column to be revisited if there's time.
 - * Next - I continued with the planning theme to assess the next (and final 3) product features. These final slices needed to be planned and broken down into distinct tickets.
 - * On the questions and answers side I am a bit unfamiliar with these so I need to revisit the User Requirements doc.
-
- Right.... I've gone through the user requirements doc. I have some concerns on implementation now...Especially on the re-indexation.
 - How can I re-index the answers?
 - Answers are currently part of the questions table field, so it'll be difficult to index these columns based on if someone deleted option D in the question...yikes. This might be something I might have to miss from the brief.
 - But at least re-reviewing the Requirements Doc has allowed me to prioritise a ticket from the Tech Debt column. I've now brought it into 'To Do'. Before this, I had put off the indexation as a thing to do later.

- Example ticket in the Tech Debt column: This one is a collection of bugs to fix on the front end noted through other manual testing as things to fix or consider.

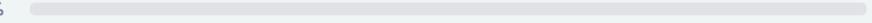
FIX: Update the Question List page in Blade HTML !

in list [In Progress](#)  

Description

Add a more detailed description...

Checklist

0% 

Ensure the title of the quiz is displayed at the top of page

Ensure the questions in each quiz are numbered. (Note you will not be able to use the DB table id - as this is not indexed by quiz.)

Add a Navbar Link so that a user can navigate back to All Quizzes List easily (at the moment the user can only press back on their browser).... Or maybe the logo Quiz Manager in navbar should route to the All Quizzes page....hmm have a think. Either of these options are fine 🎉😊

[Add an item](#)

Activity

Show Details

LT Write a comment...

ADD TO CARD

 Members

 Labels

 Checklist

 Due Date

 Attachment

 Cover

POWER-UPS

+ Add Power-up

ACTIONS

→ Move

 Copy

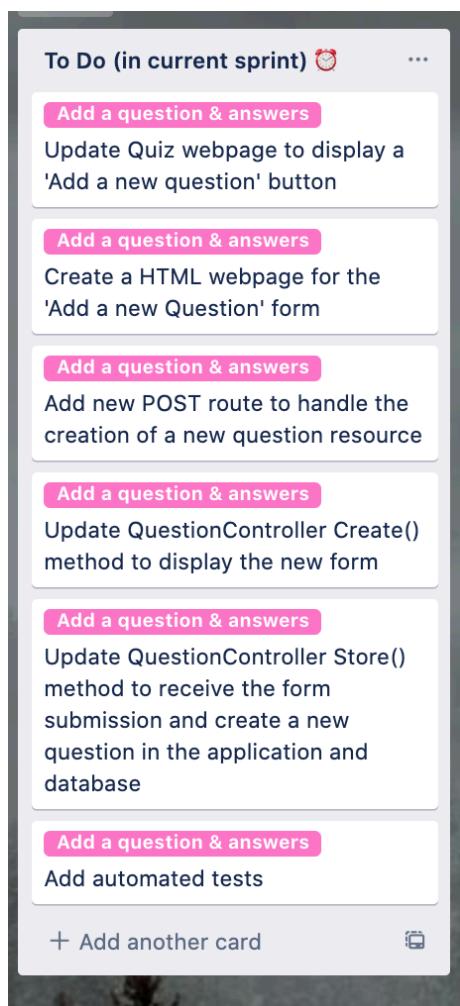
 Make Template

 Watch

 Archive

 Share

- Other than this, from a re-read of the Brief and User Requirements, no other surprises. The CRUD actions on questions and answers seem ok - and I've broken this down into distinct tickets and added on the JIRA board to track. Ready to get on with the coding, finally (it's midday and I haven't started any coding yet)....
- Still a bit of a UI concern as to handle the creation of a quiz with the handling of a question/answer (the two will have to be separate screens to click through). But I just have to park that worry and get on with it. Otherwise I really won't finish this project in the time available.
- Not limited to these JIRA tickets, but it gives a good structure for me to start with:



- **Tech Debt and Issues**

- **Feature: The quizzes are listed**

- I don't know much about the listing of quizzes or questions
- While reading through the project brief, I was unclear about the following:
 - "1. From the brief; "Add and delete questions at any point in the numerical sequence of a quiz (which may cause the questions to be re-numbered)"
 - The brief was vague and did not stipulate how it should be done - so I can choose however I want to implement this product feature.
 - The database IDs will be set by the DB and won't change, so they can't be used for ordering purposes.
 - Presumably each question has a specific index and order.
 - If the order is important, it'll need to be stored in the DB as a separate property presumably.

- **Starting with the Quiz List**

- Initially starting with a simple option...can I do this indexation on the front end?
- Using HTML ordered lists and lists tags ! Does this work....

The screenshot shows a web-based Quiz Manager application. At the top, there is a dark header bar with navigation icons (back, forward, search, etc.) and the URL '127.0.0.1:8000/quiz'. Below the header, the page title 'QUIZ MANAGER' is centered above a blue button labeled 'Add a new Quiz'. To the right of the title are links for 'QUIZZES' and 'LOGOUT'. The main content area is titled 'Quizzes' and contains a numbered list of 10 items, each representing a quiz with a link:

1. Quiz 1 - molestias velit maiores
2. Quiz 2 - et amet facilis
3. Quiz 3 - ducimus rem et
4. Quiz 4 - quia architecto odio
5. Quiz 5 - alias tempore aliquam
6. Quiz 6 - est omnis dolores
7. Quiz 7 - magnam tempore rerum
8. Quiz 8 - pariatur ratione molestiae
9. Quiz 9 - asperiores eveniet officia
10. Quiz 10 - nobis saepe esse

- Looks like Success! So I'll look into this more if this actually works for my purpose and refactor....
- Removed the database id from the view - the user does not need to know the DB id of the quiz

- **Feature: The questions are listed**

- Going through design iterations now for UX.... changes:
 - Indexing the questions listed in a quiz
 - Removed the database id of question from the view
 - Removed the 'options' text - seemed explicit already so no need to state for the user.
 - Had to make the multiple-choice options as an unordered list (hence the default bullet points have changed to an outline). The indentation looks off.... something to hold and possible fix later.
- Before & After:

The image shows two side-by-side screenshots of a web-based Quiz Manager application. Both screenshots are identical in terms of the browser interface, including the title bar, tabs, and navigation buttons.

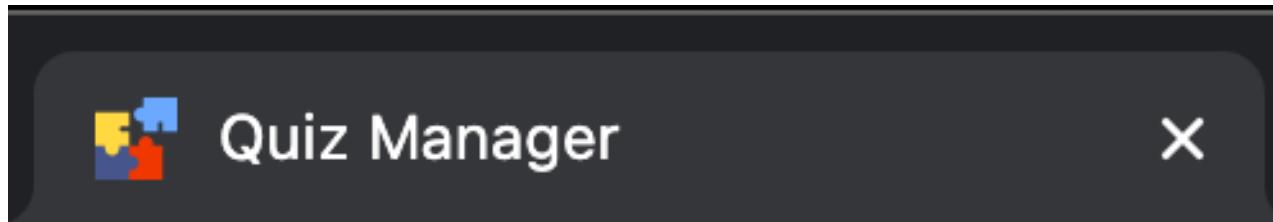
Screenshot 1 (Left): This screenshot shows a single question card. The question text is "Question 5. Atque inventore magnam dignissimos ut." Below the question, under the heading "Options:", is a bulleted list: A: suscipit, B: officis, C: iusto, D: eveniet, E: ipsam. At the bottom of the card are two buttons: "Reveal Answers" and "Delete quiz".

Screenshot 2 (Right): This screenshot shows the same question card, but the question text has been moved to the top of the card. Below it, the question number "1." is present. The "Options:" section and the list of answers remain the same. At the bottom of the card are two buttons: "Reveal Answers" and "Delete quiz".

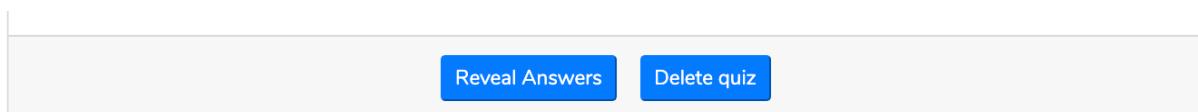
- For now, the indexation product feature is fixed on quizzes and questions  So I'm moving this ticket into Done.
 - Happy with this solution as it was very fast to implement and simple. It did not involve any complex logic, which I believe is good as it keeps the application uncomplicated (if it doesn't need to be, the implementation should be as simple as possible for future engineers and development).
 - I also discovered that the HTML 'ol' tag has a type attribute - which allows me to also order the list in other methods - ie. for the answers list, I have been able to list by alphabet 
 - Not sure about tests in this area; as not sure if the indexation is an important feature; or how I would test this. Arguably I think I don't need tests as this ordering purely relies on HTML and the ordered list `` element. I am happy to rely on the HTML ol element to correctly order the items, upon CRUD actions.
-
- **Feature done** 

Feature: Update the Favicon for the Application

- Not essential but current favicon was a generic one provided by the Laravel Scaffold
- Easier to update the favicon image than to remove it, so I found an appropriate (open source) image from online and saved the image in the public resources folder.
- Relinked the favicon in the master blade template as a puzzle piece icon - for use as the Quiz Manager logo (for now).



- **Inconsistency in UX and behaviour.** The 'Reveal Answers' option button was different to the 'Delete Quiz' button.
- The reveal answer button did not behave like a button. When the cursor was hovered over it, it became obvious the difference in behaviour between the two buttons in the footer.
- To be consistent, I investigated and noticed that one was an anchor tag and the other was a button tag. I converted both to be button tags for consistency.
- * As the quiz object has been passed from the Controller to the View, I can access the quiz['title] instead of the blade template. Both now look and behave the same way for the end user.



- **Observations:**

- Phew. That took much longer than anticipated.... It's 3PM on day 4, and I haven't worked on product features yet all day.
- I spent all day working on 'tech debt' tickets ; though I made good progress through a lot and crossed lots of items off (that needed to be done in any case). But I am a bit worried to have so much product features leftover....my aim was to have finished all product features asap, so I could have tomorrow for documentation. Doesn't look like it's going to happen at this rate but I will re-assess at the end of the day what that means for my plan.
- Need to spend the next 3 hours getting smashing through the last product features/user stories.
- Tickets that have been moved from 'TechDebt' into Done today...

Done ✓

...

FIX: Update the Question List page in Blade HTML !

3/3

FIX: Implement the List indexation of Quizzes and Questions in Quiz Manager -> upon CRUD actions, the resources should re-index (starting at 1) and auto-increment.

3/3

Refactor generic Laravel scaffold elements - remove laravel favicon, update CSS stylesheets for consistency, remove navbar toggle

3/3

When a user logs out, redirect user to the Login page

2

Download jQuery library as local resource to remove reference to CDN.

2

Delete files: unnecessary Auth registration views, controllers and routes

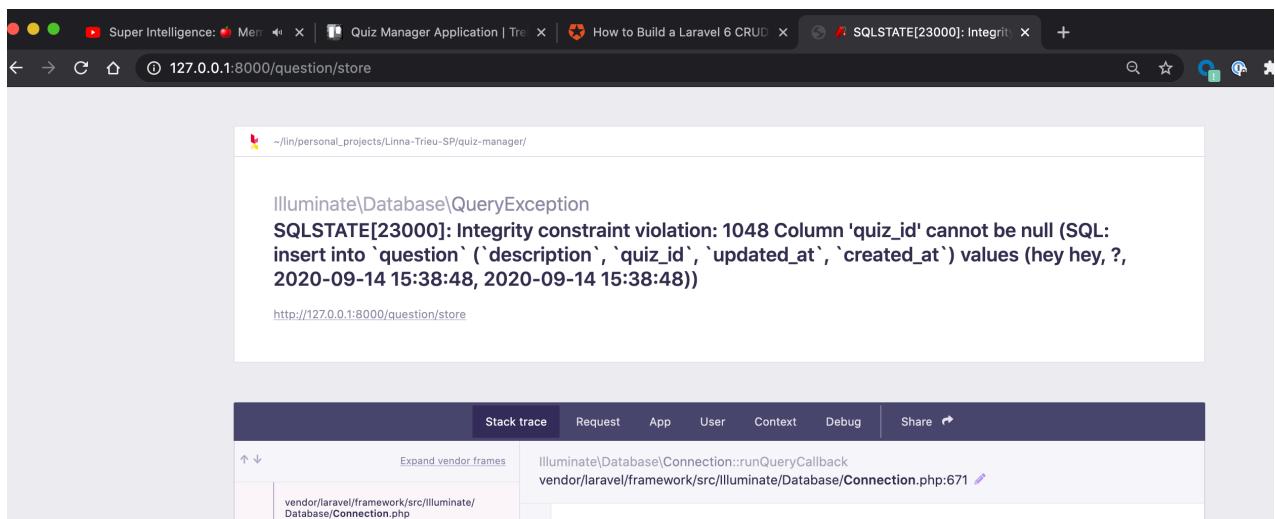
1

Delete 2 files: HomeController and home.blade.php

1

Feature: An edit user can add a question and its answers

- **Context:** I made the assumption that a user cannot add a question without accompanying answers. Thus it stands to logic that a user must submit a question with its corresponding answers in order to create a question (in the application and db), otherwise it will fail.
- I added a 'Create Question' button on the Quiz webpage
- Ugh - running into foreign key issues and passing data back and forth between the controllers and views.
- I am trying to
 - Send quiz_id from Quiz page to controller@store method
 - Send quiz_id from controller@store method to Question page
- For some reason, somewhere in the MVC the chain breaks and the quiz id isn't being set/ passed along thus the database is throwing an error that the FK can't be null.... Why?! I need to go back to research and look into Laravel docs about passing info across two different controller and two views .



- In Logs:
 - “[2020-09-14 15:38:48] local.ERROR: SQLSTATE[23000]: Integrity constraint violation: 1048 Column 'quiz_id' cannot be null (SQL: insert into `question` (`description`, `quiz_id`, `updated_at`, `created_at`) values (hey hey, ?, 2020-09-14 15:38:48, 2020-09-14 15:38:48))”

Debugging nightmare 😱 feeling in a pinch as I need to hack this and move on.

1. Check the application's routes - they look fine
 2. Give dummy data to the missing FK. Validate that is the cause of the issue -> confirmed: it is. When quiz_id is given a value, the error is resolved.
 3. Add logs to get an idea of the missing quiz_id data and why it's happening. quiz is a parameter passed into the store function, so log the argument to see what is actually coming back.
 - Aha! Logs prove that the Quiz object being passed in is empty and null
 - This means that my `store` method is not the cause of the issue.
 - The cause is upstream as the parameter \$quiz object is empty.
 - Yikes.... I'm still going and it's the last 30 mins of the day. Really haven't gotten far with this which is frustrating.

4. At what point in the chain does the Quiz object get lost?

- Create method looks good
 - Why is the quiz object not getting passed to the store method?

The screenshot shows a terminal window with the following details:

- Terminal title: Linna-Trieu-SP [~/personal_projects/Linna-Trieu-SP] - .../quiz-manager/storage/logs/laravel.log
- File path: quiz_list.blade.php, NewQuizTest.php, QuizController.php, web.php, QuestionController.php, quiz.blade.php, new_question.blade.php, laravel.log
- Log message: Log format not recognized. The file size (4.82 MB) exceeds configured limit (2.56 MB). Code insight features are not available.
- Log entries:
 - [2020-09-14 16:32:15] local.DEBUG: Quiz at QuestionController@create["id":4,"title":"est nemo veritatis","icon":"default_icon.jpg","created_at":"2020-09-14T15:00:00"]
 - [2020-09-14 16:32:15] local.DEBUG: Quiz inside the user check["id":4,"title":"est nemo veritatis","icon":"default_icon.jpg","created_at":"2020-09-14T15:17:52.000Z"]
 - [2020-09-14 16:32:30] local.DEBUG: Quiz at QuestionController@store[]\n
 - [2020-09-14 16:32:30] local.ERROR: SQLSTATE[23000]: Integrity constraint violation: 1048 Column 'quiz_id' cannot be null (SQL: insert into `question` (`description`, `quiz_id`) values ('', null))

- Oh no.... The cause was a mis-naming of the variable \$quiz_id being passed in from the view to the controller as \$quiz.... A very silly error. Glad I spotted it. But frustrated that it happened in the first place.
- Damn. It's still not solved the error. The request object in the Store method (question controller is still NULL, an empty object) 
- I need to move on otherwise I'll be stuck on this for the rest of the project, and I've already spent too long on this problem.
- I will stop and come back to this with a fresh pair of eyes.

- I figured out the issue was that there I wasn't sending enough arguments to the Store function. The store method expected both request and a quiz object parameters.
- However only one was working. I added a parameter variable in the route to explicitly pass in the quiz id.... fixed. Very hacky and bad naming from a RESTful perspective (inconsistent, not clear about resources, or what is being created).
- BUT it works. And something to refactor in in another ticket. I need to get the entire feature up, running and tested - as I'm quite behind today's schedule... yikes.

```
45
46     // Create a New Question
47     //Route::get('/question/{quiz_id}/create', [
48     Route::get( uri: '/question/create/{quiz}', [
49         'uses' => 'QuestionController@create',
50         'as' => 'question.create'
51     ]);
52     Route::post( uri: '/question/{quiz_id}', [
53         'uses' => 'QuestionController@store',
54         'as' => 'question.store',
55     ]);
56
```

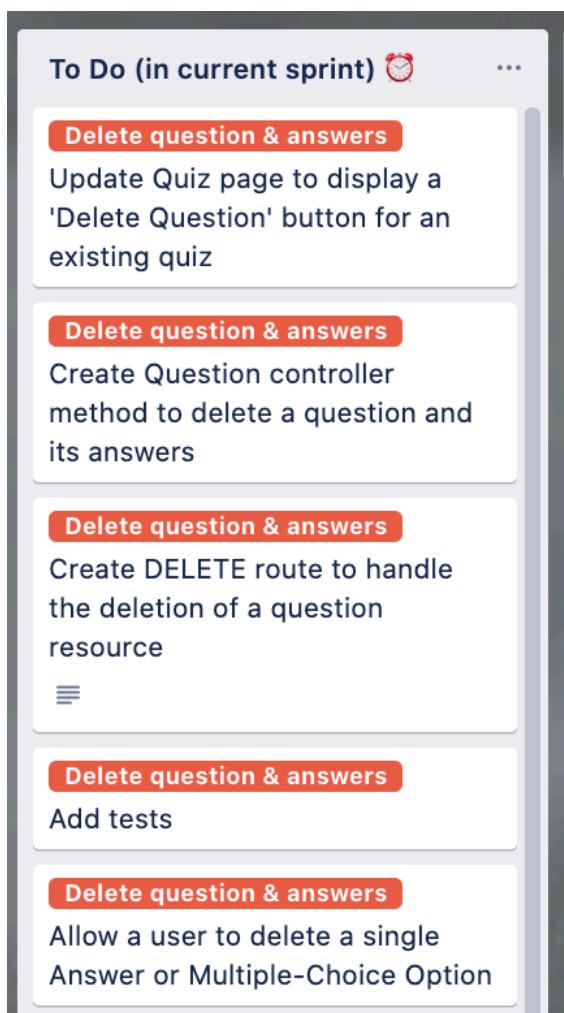
- In the interest of time, I finished the new question form with a very basic HTML design.
- All Question fields are free input boxes for the user to manually type. I've added some validation where I can e.g. input pattern regex validation on the answer key field, and making options A-C required, D-E not required.
- I would like to go back and improve the design of this - few points for improvement:
 - Hate the free input fields - esp for the answer key field. Maybe something like a radio button would do better, and the user can tick the radio button for one of the multiple choice options, instead of manually typing a letter.
 - Option D and E are not required fields but in this form they are given as much prominence as the other A-C required fields. Blur them out by default, and maybe give the user to expand with more options if they select 'add another option' etc?
 - A user can submit a form with an option_e without option_d populated. This looks odd... is there anyway I can prevent that?
 - A user can submit a form with a faulty answer key. By accident a user can submit an answer key of 'E', even if there was no option_e provided in the question. Validation required here - seems like an easy and likely mistake for users to make 😬
- Overall though...phew! Finally I have a working Create Question and Answer form which saves the new question and its answers in the application and the database 🎉 At least that's progress. Better than nothing. There are flaws, that can be refactored and worked on later.

- Eugh. End of day. That was a manic, unplanned day. Felt like a lot of scrambling. I hit a roadblock and spent way too long troubleshooting. Not sure what I could have done to avoid that or do better. I will reflect and come back to it tomorrow.
- Realised I have forgotten about the logic required in adding new questions, along with a new quiz. In fairness, this feature does work; but there is painful user experience involved in doing so. A user needs to create a quiz. Gets redirected to homepage, Then needs to find and select the quiz before inputting the quiz questions one-by-one. I forgot about this user journey...added a ticket to the Trello board for now. Not sure on the prioritisation of it. I'm very tempted to leave it for this iteration, and forget about it until the remaining features have been implemented. But this is a very important point for the next iteration of the New Question feature... 😱
- I have another 1-2 days left on the total project. But I have two full user stories remaining (edit user can delete or edit questions and answers). Time for the crunch period now... things are going to the wire.
- A re-prioritisation exercise has to be done first thing tomorrow morning in the Sprint Planning Session.

DAY 6, 15/09/20 (THE PENULTIMATE DAY)

- Phew. There is a lot to get on with today....
- First: on the remaining tickets for the New Question feature - none of them are blocking. But they are causing me a headache in troubleshooting, and I don't want to get bogged down the entire morning trying to resolve those issues. As they are not critically important, I'm going to move them to the bottom of the To-Do column. I will start the day on adding the remaining product features.

Feature: An edit user can delete a question or an answer



- Broke the product feature into distinct Trello tickets.
 - Deleted the question - seems straightforward, and I could follow the style I used earlier to delete a question. As stated with Business Assumptions, when a user deletes a question, the question and all of its associated answers are also deleted from the application and database.
 - Have done some refactoring to change the quiz id parameters to the quiz object itself. This removes an entire DB call in the function itself, e.g. destroy. Where possible it makes sense to just pass in the object, rather than a specified id. In destroy, I was passing in the ID to then make an eloquent call to the db itself to find the object. In this example, that makes no sense - is detrimental for performance and inefficient. Avoid calls where you can, though it doesn't matter so much at this small size application, but for scaling up, it will.
 - Refactoring - before and after

- As a result, I can remove the Eloquent ‘find’ call on line 91, and 73 below.

```

v 7 quiz-manager/app/Http/Controllers/QuestionController.php ...
@@ -5,6 +5,7 @@
5  use App\Models\Question;
6  use App\Models\Quiz;
7  use App\Models\UserPermission;
8  use Illuminate\Http\RedirectResponse;
9  use Illuminate\Http\Request;
10 use Illuminate\Http\Response;
@@ -83,12 +84,12 @@ public function store(Request $request): RedirectResponse
83      /**
84       * Remove the specified resource from storage.
85       *
86      - * @param int $questionId
87      * @return RedirectResponse
88      */
89      - public function destroy(int $questionId): RedirectResponse
90      {
91          $question = Question::find($questionId);
92          $quizId = $question->quiz_id;
93          $permissionLevel = Auth::user()->permission_level;
94      }
95

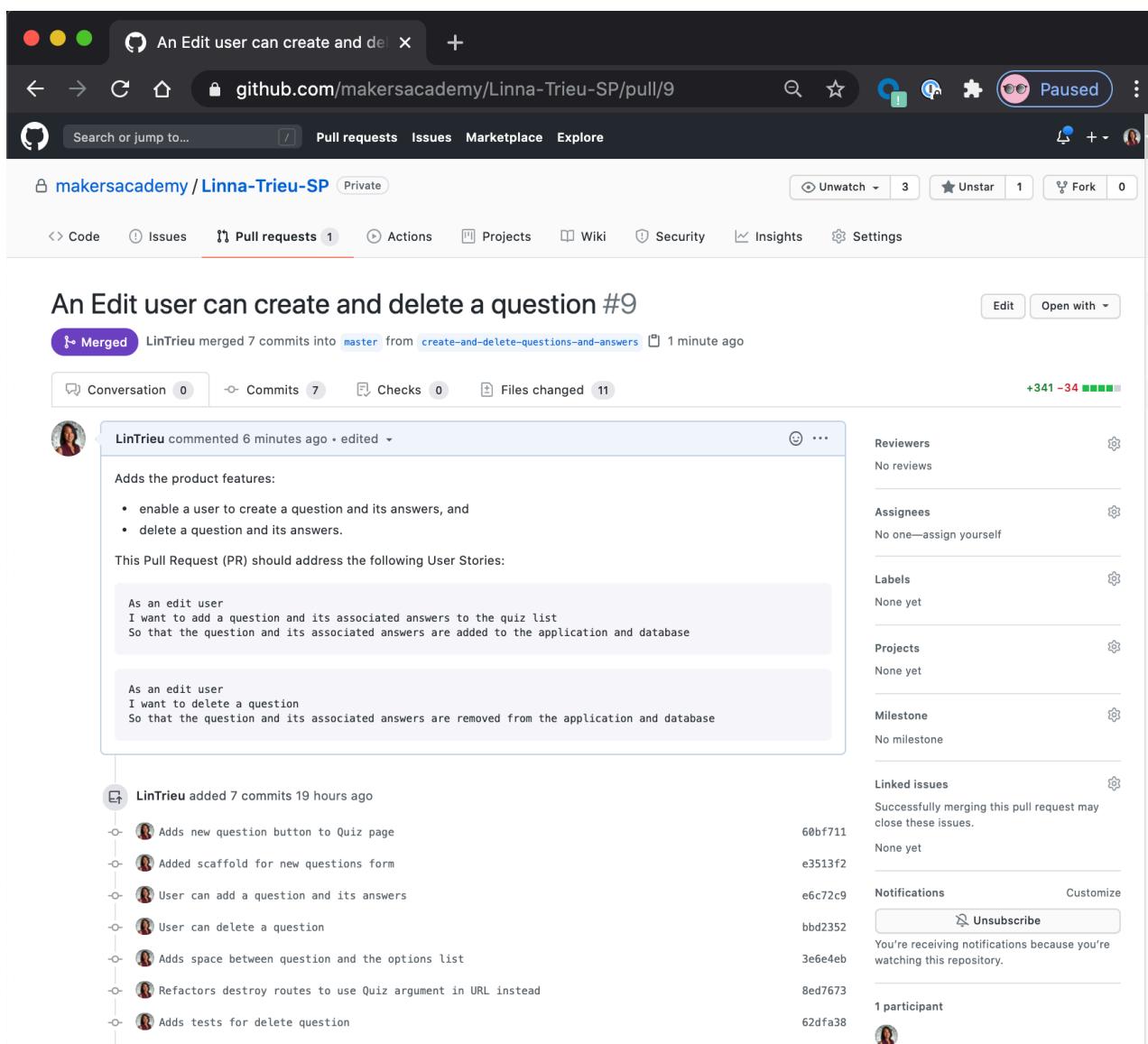
```

```

v 7 quiz-manager/app/Http/Controllers/QuizController.php ...
@@ -4,6 +4,7 @@
4
5  use App\Models\Quiz;
6  use App\Models\UserPermission;
7  use Illuminate\Http\RedirectResponse;
8  use Illuminate\Http\Request;
9  use Illuminate\Http\Response;
@@ -65,12 +66,12 @@ public function store(Request $request): RedirectResponse
65      /**
66       * Delete the specified Quiz from database.
67       *
68      - * @param int $quizId
69      * @return RedirectResponse
70      */
71      - public function destroy(int $quizId): RedirectResponse
72      {
73          $quiz = Quiz::find($quizId);
74          $permissionLevel = Auth::user()->permission_level;
75          if ($permissionLevel == UserPermission::PERMISSION_EDIT) {
76
77      }

```

- Haven't dealt with if a user wants to delete a single answer yet though....so I have added a ticket and moved into the To Do column. When picking up this ticket, re-visit the User Requirements.
- New Ticket: "Allow a user to delete a single Answer or Multiple-Choice Option"
- Closed this feature. Created a PR and tested the features manually and in isolation. As testing met the acceptance criteria and user stories, merged the branch into Master, and moved the tickets from 'Ready for Testing' into the 'Done' column.



The screenshot shows a GitHub Pull Request page for a repository named 'makersacademy/Linna-Trieu-SP'. The pull request is titled '#9 An Edit user can create and delete a question' and has been merged. The commit history shows 7 commits from LinTrieu, with the most recent being 19 hours ago. The pull request details section includes a comment from LinTrieu about product features and user stories, and a note about closing issues. The right sidebar provides options for reviewers, assignees, labels, projects, milestones, linked issues, and notifications.

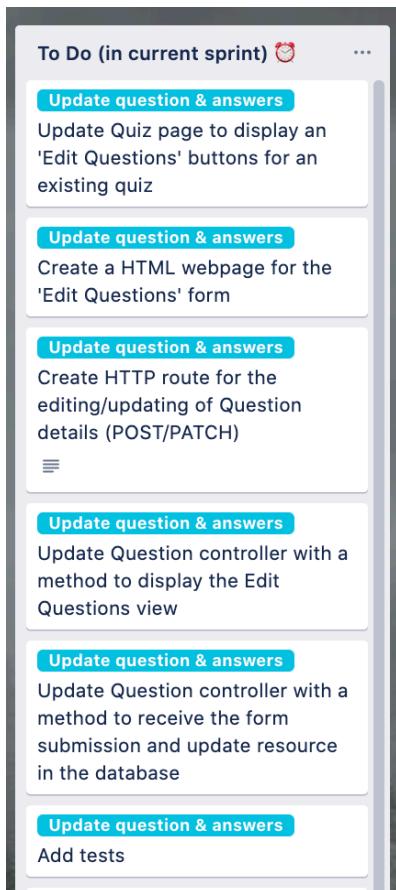
Pull Request Details:

- Merged:** LinTrieu merged 7 commits into `master` from `create-and-delete-questions-and-answers` 1 minute ago
- Conversation:** 0
- Commits:** 7
- Checks:** 0
- Files changed:** 11
- Reviewers:** No reviews
- Assignees:** No one—assign yourself
- Labels:** None yet
- Projects:** None yet
- Milestone:** No milestone
- Linked issues:** Successfully merging this pull request may close these issues. None yet
- Notifications:** Customize. You're receiving notifications because you're watching this repository.
- Participants:** 1 participant (LinTrieu)

Commit History:

- LinTrieu added 7 commits 19 hours ago
- ↳ Adds new question button to Quiz page (60bf711)
- ↳ Added scaffold for new questions form (e3513f2)
- ↳ User can add a question and its answers (e6c72c9)
- ↳ User can delete a question (bbd2352)
- ↳ Adds space between question and the options list (3e6e4eb)
- ↳ Refactors destroy routes to use Quiz argument in URL instead (8ed7673)
- ↳ Adds tests for delete question (62dfa38)

- **Feature: An edit user can update question and answers**
- Broke the product feature into distinct tickets and prioritised in order.



- **1. Firstly deal with 'edit' – Show the form to edit an existing question**
 - Button on the Quiz page ✓
 - Route to get the controller method edit ✓
 - Edit method to display the form to edit ✓
 - Done with not too much issues. Though the design of the View (Quiz Page) is not ideal... it'll do for now.

- Created a new View for the Edit Question form, though I could have re-used the New Question view and used conditionals. For now, it seemed faster to keep the views separate and distinct as to clearly indicate what view is displayed to the End User; instead of involving blade conditionals.
- **2. Next deal with ‘update’ - Update the edited question in the database**
- Created the form with the previous questions so the end user knows exactly what part of the Question they are updating. I was concerned about how to do that beforehand, but with some research I found the Laravel blade ‘old’ helper method. The old method lets me display the current value of the fields to the user and I passed in the question object so I could access all parts of the questions required for the Edit Question Form. = quick win, thankfully 
- **3. Hmm... weirdly the form is allowing me to submit but it isn't actually updating**
when I am re-directed to the list of Questions. And in checking the DB, the edited values aren't actually saving to the DB either. Looking into this, though it must be a bug in the logic that sits in the QuestionController@Update method
- Still can't seem to see why... hmm. A bit frustrating.
- After lunch I think I need to research, look up this online and try to debug! 
- Solved ✓
 - With a fresh pair of eyes, I realised I needed to access the new input fields of the question. For some reason I was accidentally accessing the previous (i.e. old) version of the question object.
 - Now retrieving and saving the new values in the Edit Question form.
- Very relieved to have completed the first iteration of that feature. - happy to have done that.
- Now going through a round of manual testing as I know I will have missed a lot of edge cases and validation is required.
- Getting distracted with design choices... bad habit. For example improving the new question form. At the moment it is too similar to the edit questions form and it is easy to get the two mixed up.
- So I changed the HTML form ‘placeholder’ text to the ‘small’ element instead, which lists the information below the input box, rather than a placeholder in itself (which is the behaviour for the Edit Questions form).

QUIZ MANAGER

QUIZZES

Create a New Question

Question description
E.g 'What is the capital of the UK?'

Input the multiple-choice options for your question (*minimum of three*)

Option A	<input type="text"/> Manchester
Option B	<input type="text"/> London
Option C	<input type="text"/> Glasgow
Option D	<input type="text"/>
Option E	<input type="text"/>

Correct answer Input an answer key: A, B, C, D or E

Submit Question

- Also updating the Edit Quiz form from a UX perspective. Each of the views currently are very bare and it is not immediately clear to the end user what they are supposed to do. I am adding text to each form to clarify and provide instructions to the end user....
- Finally adding some tests for the update feature:
- AH. Falling behind today and haven't touched any documentation today.... For some reason my feature tests just aren't working.
- It seems related to an error using Faker and Factory - a recurring error even after creating an object (e.g. quiz) using the factory, a Database error fails the test as the DB does not recognise that the object has been created.... 'NOT NULL constraint'
- E.g.

The screenshot shows the PhpStorm IDE interface with several tabs open. The main code editor tab contains a test class named `EditAQuestionTest.php`. The terminal tab shows the execution of the test, which fails with the following output:

```

Edit Aquestion (Tests\Feature\EditAQuestion)
x Edit user can update a question 109 ms
  | expected status code 302 but received 500
  | failed asserting that 500 is identical to 500.
  |
  | /Users/lin/personal_projects/Linna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Testing/TestResponse.php:186
  |
  | FAILURES!
  Tests: 0, Assertions: 0, Failures: 1.
make: *** [phpunit] Error 1

```

The log panel on the right shows a detailed stack trace of the error:

```

Log format not recognized
The file size (0.93 MB) exceeds configured limit (2.56 MB). Code insight features are not available.

OR: SQLSTATE[23000]: Integrity constraint violation: 19 NOT NULL constraint failed: question.description (SQL: update "question" set "title" = 'Geography', "description" = 'What is the Capital of the UK?', "quiz_id" => $quiz->id, "id" => 1,)

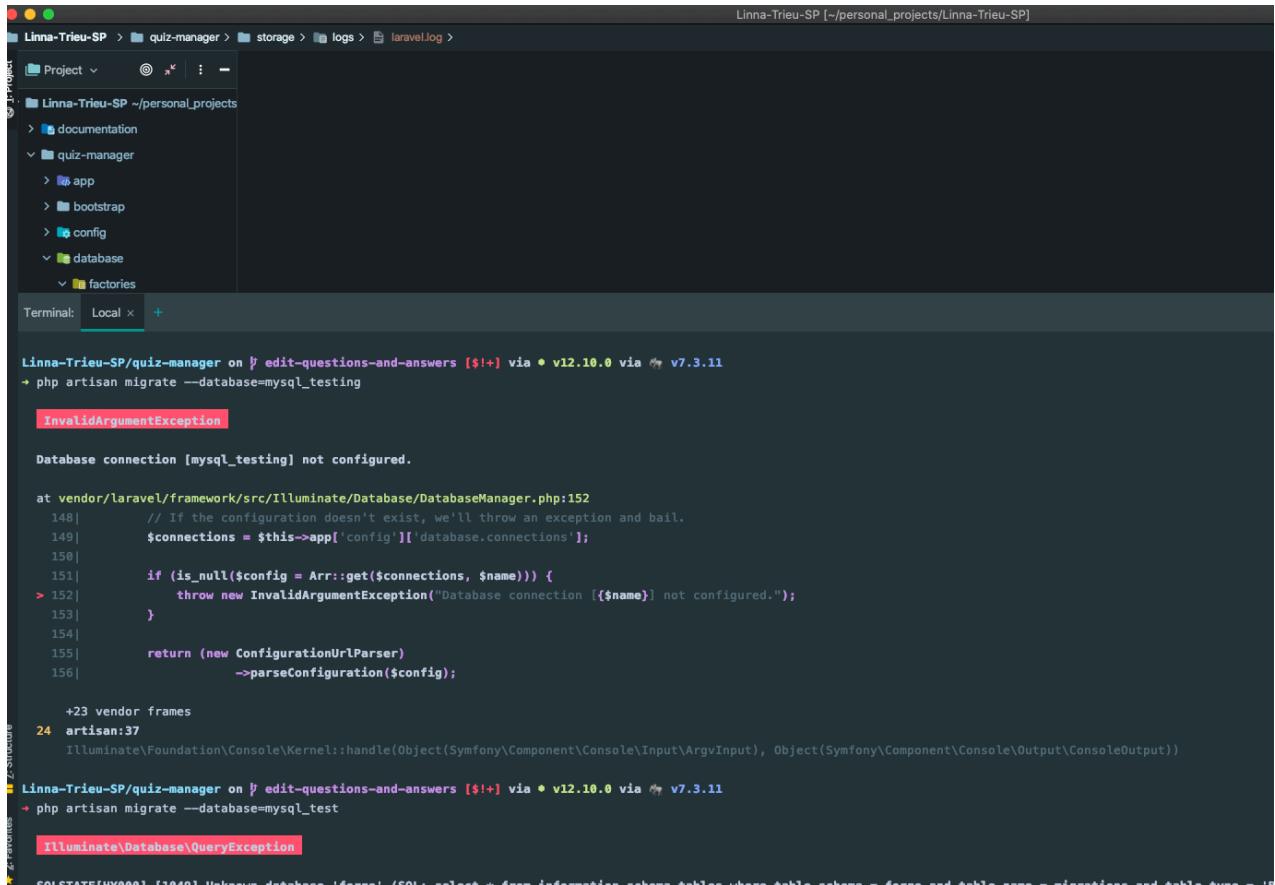
Line 34407: OR: SQLSTATE[23000]: Integrity constraint violation: 19 NOT NULL constraint failed: question.description (SQL: update "question" set "title" = 'Geography', "description" = 'What is the Capital of the UK?', "quiz_id" => $quiz->id, "id" => 1,)

Line 34408:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Connection.php(631): Illuminate\Database\Connection->runQuery()
Line 34409:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Connection.php(496): Illuminate\Database\Connection->runQuery()
Line 34410:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Connection.php(429): Illuminate\Database\Connection->aff
Line 34411:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Query\Builder.php(286): Illuminate\Database\Connection->
Line 34412:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Eloquent\Builder.php(866): Illuminate\Database\Eloquent\Connection->
Line 34413:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Eloquent\Builder.php(866): Illuminate\Database\Eloquent\Connection->
Line 34414:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Eloquent\Builder.php(866): Illuminate\Database\Eloquent\Connection->
Line 34415:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Eloquent\Builder.php(866): Illuminate\Database\Eloquent\Connection->
Line 34416:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Eloquent\Builder.php(866): Illuminate\Database\Eloquent\Connection->
Line 34417:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Database/Eloquent\Model.php(888): Illuminate\Database\Eloquent\Model->save()
Line 34418:     inna-Trieu-SP/quiz-manager/app/Http/Controllers/QuestionController.php(138): Illuminate\Database\Eloquent\Model->save()
Line 34419:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Routing\Controller.php(54): call_user_func_array(Array, Array)
Line 34420:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Routing\ControllerDispatcher.php(43): Illuminate\Routing\Control
Line 34421:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Routing\Route.php(239): Illuminate\Routing\ControllerDispatcher->
Line 34422:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Routing/Route.php(196): Illuminate\Routing\Route->runController()
Line 34423:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Routing/Router.php(685): Illuminate\Routing\Route->run()
Line 34424:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Pipeline.php(128): Illuminate\Routing\Router->Illuminate
Line 34425:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Routing\Middleware/SubstituteBindings.php(41): Illuminate\PipeLine
Line 34426:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Pipeline.php(167): Illuminate\Routing\Middleware\Substit
Line 34427:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Auth\Middleware/Authenticate.php(44): Illuminate\PipeLine\Vi
Line 34428:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Foundation\Http\Middleware\VerifyCsrfToken.php(77): Illuminate\Pl
Line 34429:     inna-Trieu-SP/quiz-manager/vendor/laravel/framework/src/Illuminate/Pipeline.php(167): Illuminate\Foundation\Http\Middleware\Verifi

```

- Looking into why factory/faker isn't working in this test....Spending a lot of time researching and debugging this...Eating up a lot of time.
- TIMEBOX: MOVE ON AT 4PM IF STILL NO SOLUTION 😭
- AH - it seems to be a limitation of the database driver. -sqlite
- Really frustrating but through trawling through online forums, it seems to be an edge case issue for some individuals
- I need to create a fresh DB for test purposes

- Went deep deep into debugging random things - infra , db, testing
- Wasn't able to resolve the issue and I had to move on 😞



```

Linna-Trieu-SP [~/personal_projects/Linna-Trieu-SP]
Linna-Trieu-SP > quiz-manager > storage > logs > laravel.log >

Project: Linna-Trieu-SP ~/personal_projects
  documentation
  + quiz-manager
    app
    bootstrap
    config
    + database
      factories

Terminal: Local × +
```

```

Linna-Trieu-SP/quiz-manager on ⚡ edit-questions-and-answers [$!+] via • v12.10.0 via ⚡ v7.3.11
+ php artisan migrate --database=mysql_testing

InvalidArgumentException

Database connection [mysql_testing] not configured.

at vendor/laravel/framework/src/Illuminate/Database/DatabaseManager.php:152
148|     // If the configuration doesn't exist, we'll throw an exception and bail.
149|     $connections = $this->app['config'][database.connections];
150|
151|     if (is_null($config = Arr::get($connections, $name))) {
152|         throw new InvalidArgumentException("Database connection [{$name}] not configured.");
153|     }
154|
155|     return (new ConfigurationUrlParser)
156|         ->parseConfiguration($config);

+23 vendor frames
24 artisan:37
 Illuminate\Foundation\Console\Kernel::handle(Object(Symfony\Component\Console\Input\ArgvInput), Object(Symfony\Component\Console\Output\ConsoleOutput))

Linna-Trieu-SP/quiz-manager on ⚡ edit-questions-and-answers [$!+] via • v12.10.0 via ⚡ v7.3.11
+ php artisan migrate --database=mysql_test

Illuminate\Database\QueryException

SQLSTATE[HY000] (1049) Unknown database 'forge' (SQL: select * from information_schema.tables where table_schema = 'forge' and table_name = 'migrations' and table_type = '')


```

- I abandoned the plan and the specific Feature test.- not sure why it wasn't working. Kept the current testing structure with an in memory sqlite database.
- I created a branch which contains the experimentation.... But had to leave behind to get on with THE FINAL DAY OF PROJECT. Though that was a disappointing waste of 2 hours.
- Branch: `test-database-experiment`
- OUTCOME: As a result of the lack of feature tests in one area (Feature: Editing a Question & Answer) I will need to document this in the Test Documentation, and as a result, perform more manual testing in that particular area to address the lack of automated tests.

- **Experimenting with styles...**

QUIZ MANAGER

QUIZZES LOGOUT

Quiz: aut quibusdam officiis

1. Laborum eos nostrum culpa est animi.? Edit Delete

Options:

- A. veritatis
- B. suscipit
- C. consequatur
- D. enim
- E. sed

2. Voluptatem quis voluptatibus praesentium facilis consequatur possimus.? Edit Delete

Options:

- A. qui
- B. nihil
- C. cupiditate
- D. sit
- E. quo

Reveal Answers Add Question Delete Quiz

QUIZ MANAGER

QUIZZES LOGOUT

Quiz: aut quibusdam officiis

1. Laborum eos nostrum culpa est animi.? Edit Delete

Options:

- A. veritatis
- B. suscipit
- C. consequatur
- D. enim
- E. sed

2. Voluptatem quis voluptatibus praesentium facilis consequatur possimus.? Edit Delete

Options:

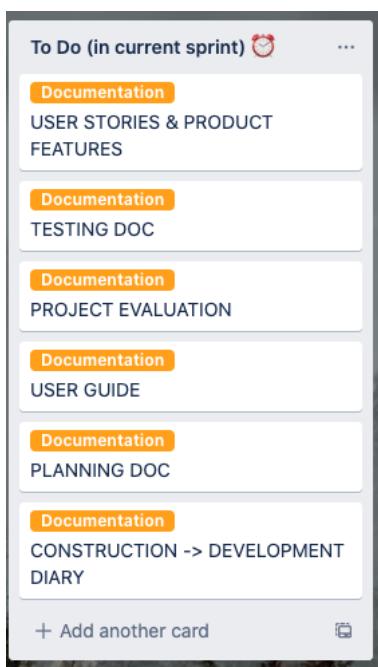
- A. qui
- B. nihil
- C. cupiditate
- D. sit
- E. quo

Reveal Answers Add Question Delete Quiz

- A tiring day, but done and achieved a lot over the course of this development diary. I am happy to have fulfilled all the user requirements and user stories for the Quiz Manager in very tight time constraints.
- Not much more I can do. I should implement a code freeze from this point 

16/09/20 - FINAL PROJET DAY

- OK, last few changes to made before a code freeze.
 - Need to make sure everything is available for offline use (so that the folder can be sent as a zip file and be run in isolation - without reference to external website links)
 - Need to include and commit all vendor files to Git
 - Do manual testing
 - Implement quick wins in terms of product bugs and missing features
- Then I need to implement a code freeze and move onto the documentation side: Still using the trello board to track deliverables.



- As for the development diary, this is the final entry as the development phase of the Quiz Manager project has finished.
- For the remainder of my time on the project, I will now move onto documenting the post-implementation documents.