Iteration: 1 (23/02 - 01/03)

Alex Taylor - amt22@aber.ac.uk

March 16, 2017

Version 0.7 (Draft)

Contents

1 Story: Admins can create quizzes via the website

1.1 Analysis - Breakdown of Tasks

This story is quite large and should be broken down into several substories:

- (3) Admins can log into a backend
- (2) They are presented a list of their quizzes
- (5) They can create a new quiz in the backend
- (3) They can edit an existing quiz they own

These have been added to the story list document.

2 Story: Admins can log into the backend

2.1 Analysis - Breakdown of Tasks

- Create users table
- Add login page
- Add register page

2.2 Design

Design was limited due to the automated builder. However it added some view files and a default HomeController for a basic homepage.

2.3 Implementation

Implementing this was far easier than originally anticipated, Laravel comes with the needed tables out of the box and has a command to run that sets up simple auth for users: *php artisan make:auth*

2.4 Testing

Because the login and auth is handled by Laravel by default, testing it was deemed to not be a priority. However, three simple tests were written to ensure it never breaks due to future changes. Dusk Tests:

1. Test to ensure the application redirects to /login is the user is not already logged in
2. Test for logging in with a user in the database
3. Test for registering a new user

3 Story: They are presented a list of their quizzes

3.1 Analysis - Breakdown of Tasks

- Make the homepage the QuizController rather than the HomeController
- Check and get the user who is logged in
- Display a list of quizzes for that user

3.2 Design

The HomeController was removed in this period of work and the QuizController used in its place.

3.3 Implementation

Quite an easy amount of work, changing the controller was a simple change to the routes and then updating the quiz view file to use the same layout as the original home views. To get the user, a helper function is provided: *auth()->user()* which gets the user object. Obtaining the id from this is simple and then using an Eloquent ORM call it is easy to find all the quizzes owned by that user.

3.4 Testing

Dusk tests for this story:

1. Test to see if the /home page lists the quizzes as it would on the /quizzes page
2. Test to see if a quiz that belongs to a user is present on the page
3. Test to see if a quiz that belongs to another user is not present on your page whilst your own is

These tests highlighted a problem with the testing framework however. Chrome is used as the Remote Web Driver for running these application tests in. For each test a new migration is made within the test database, thereby wiping the data created within each test. An unintended side effect however is that every new user created starts at id=1 in the users table (a user has to be created for all these tests.) The Chrome driver seems to remember that the user of id=1 logged in, in the previous test and therefore skips the auth step. This means the test order is messed up due to the test trying to log in even though it is already logged in. The solution to this is to create each new user in the next id record, whilst this is somewhat convoluted, it seems to work.

Potential future tests: Use sessions have two users log in and see/ not see the relevant quizzes.

4 Non-Story Work

4.1 Database Work

Some initial work that is needed for almost all the stories is having a working database set up to store the users, quizzes, questions etc. Seeing as the amount of work to setup all the tables and their relationships would not take long, it was deicded that this could be done all at once at the start of the iteration.

While creating these tables it was possible to create the controllers needed within the application at the same time using: *php artisan make:model *name* -mc*

4.2 Seed Data

Because of the amount of changes to the database that were being made, the tables were repeatedly wiped and seeding some data was needed. To do this some seeders were generated with *php artisan make: seed *name**. These were created under database/seeds/ and simply required creating new objects of the desired Model and adding the various fields as parameters of the objects. These objects are then saved to the database. These seeders can be run when a migration is called such that the data is replaced as soon as its lost.

4.3 Layout Changes

The initial *make:auth* command created a default home page with a menu bar and some basic styling. This styling was created with Bootstrap and looks quite nice so the basic styling has been kept. This layout was modified somewhat to add some menu options that persist across pages. This layout is then used by all the backend pages created by extending it.