**LARAVEL**

**Part 2**

**User authentication**

* *php artisan make:auth*

Enter the above command in the terminal. Select ‘yes’ when asked to create a layout with user authentication.

Auth controllers are under http>controllers>auth

Auth views are under resources>views>auth

To add a foreign key:

* php artisan make:migrate add\_user\_id\_to\_posts (call it something explicit like this)
* Find the migrations script and do the following:

Public function up()

{

*//where posts is the table name, and user\_id is column name*

Schema::table(‘posts’, function($table) {

$table->integer(‘user\_id’);

});

}

Public function down()

{

Schema::table(‘posts’, function($table) {

$table->dropColumn(‘user\_id);

});

}

* php artisan migrate

Once this has been completed, in the PostsController, under the store() method – add the user\_id:

$post->user\_id = auth()->user()->id;

*//this gets the currently logged in user*

**Model relationships**

To create a relationship between users and posts table:

In the models:

//in post model:

Public function user() {

Return $this->belongsTo(‘App\user’);

}

//in user model:

Public function posts() {

Return $this->hasMany(‘App\post’);

// 1 to many relationship because a user can have more than 1 post

}

//in the dashboard controller (or homeController, if name was changed on creation):

Public function index()

{

$user\_id = auth()->user()->id;

$user = User::find($user\_id);

Return view(‘dashboard’)->with(‘posts’, $user->posts);

//as there is now a relationship, $user->posts can be used

}

//At the top of the script, make sure to add: use App\user;

//in dashboard view:

Create a foreach loop through user-specific created posts. You can also add edit and delete buttons in this section also.

As there is now a relationship, you can use the user() method created in the posts model to access user fields. E.g. in index:

<p>Created By: {{$post->user->name}}</p>

//where $post is the var passed to script, user is the method which links/relates to user model/table, and name is column name

**Access control**

If you look in the dashboard controller (or home controller if name unchanged), which was created when making authentication, it has a method called *\_\_construct()* which has the following code:

$this->middleware(‘auth’);

This basically does not allow access to anyone who is not signed in. This method can be copied and pasted in to the posts controller. You would likely want to add exceptions because guests should be able to view the posts. To do this…

//PostsController

Public function \_\_construct()

{

$this->middleware(‘auth’, [‘except’ => [‘index’, ‘show’] ]);

//where index and show are the views you want to allow guests to view

}

As we don’t want guest to see admin buttons like edit and delete, in the views that these buttons are present, you can put a conditional to not show to guests:

@if(!Auth::guest()) //if user is not a guest, display buttons

//code

@endif

You also wouldn’t want users to edit/delete posts that other users created. So under the @if(!Auth::guest()) conditional, put another:

@if(Auth::user()->id == $post->user\_id)

//if the currently logged in user\_id is equal to the user\_id of the user who created the post

What still isn’t right, is that users can still type in the URL to edit another users post. To correct this, in the PostsController, edit the code in the methods. For example, in the edit() method:

Public function edit($id)

{

$post = Post::find($id);

//check for correct user

If(auth()->user()->id !== $post->user\_id) {

Return redirect(‘/posts’)->with(‘error’, ‘unauthorised page’);

}

Return view(‘posts.edit’)->with(‘post’, $post);

}

//make sure to do the same for the destroy() method

**File uploading**

In the create view – add a file upload:

{{ Form::file(‘cover\_image’) }}

//in the Form::open() line, add the following code (after post but still within array):

,’enctype’=>’multipart/form-data’]) }}

//create a migration to add a column for the file uploads

Php artisan make:migration add\_cover\_image\_to\_posts

//in the migration file:

Public function up()

{

Schema::table(‘posts’, function($table) {

$table->string(‘cover\_image’);

});

}

Public function down()

{

Schema::table(‘posts’, function($table) {

$table->dropColumn(‘cover\_image’);

});

}

//now run:

Php artisan migrate

//this column will now be in the posts table – check via phpmyadmin

Make sure to edit the PostsController method to validate the file upload and store the image:

$this->validate($request, [

‘title’ => ‘required’,

‘body’ => ‘required’,

‘cover\_image’ => ‘image | nullable | max:1999’

*//where image is file type, nullable = optional, 1999 is file size (just under 2MB)*

]);

//handle file upload

If($request->hasFile(‘cover\_image’)) {

//get filename with extension

$filenameWithExt = $request->file(‘cover\_image’)->getClientOriginalName();

//get just filename

$filename = pathinfo($fileNameWithExt, PATHINFO\_FILENAME);

//get extension

$extension = $request->file(‘cover\_image’)->getClientOriginalExtension();

//filename to store

$fileNameToStore = $filename . ’\_’ . time() . ’.’ . $extension

*//where time() is timestamp to make a unique filename, to prevent been overwritten*

}else{

$fileNameToStore = ‘noimage.jpg’; //default image

}

//upload image

$path = $request->file(‘cover\_image’)->storeAs(‘public/cover\_images’, $fileNameToStore);

//if not already created, this will initially create a directory public/cover\_images which will be located: storage>app>public

//create post

$post->cover\_image = $fileNameToStore; //to add to database

Currently, this folder is not accessible through the web browser so you need to create a symlink. In terminal:

php artisan storage:link

A storage folder should now be added in public/

This image can now be accessed in the view like:

<img src=”/storage/cover\_images/{{$post->cover\_image}}”/>

In the destroy() method in PostsController, you will need to delete the image from the storage folder, as well as delete the data from the column name

If($post->cover\_image != ‘noimage.jpg’) { //if cover\_image is not the default image

Storage::delete(‘public/cover\_images/’ . $post->cover\_image);

}

//At top of script, make sure to add: use Illuminate\support\facades\storage

You would also need to delete an image from the folder when user is update their post with a new image.

**Quiz**

**Part 1**

1. Add user authentication
2. Add a foreign key to the posts table which references user\_id from the user table
3. Add the user\_id to the store method in the PostsController for when user creates a post
4. Create a relationship between the post and user table
5. Display the user who created each post under each post

**Part 2**

1. Add access control to posts, including exceptions to guest can view posts
2. Hide the edit and delete buttons from guests
3. Hide the edit and delete buttons from users if the post was not created by them
4. Prevent users from typing in url to edit/delete other users posts

**Part 3**

1. Add a file upload input in the create view
2. Create a new migration to add a column for uploaded images making sure to edit the migrations file before running the migration
3. Validate the uploaded image in the PostsController
4. Handle the file upload
5. Upload the image in to the correct directory
6. Store the image path in the database table
7. Create link between the storage folder so accessible from browser
8. How would you display the uploaded image in a view
9. If a post is deleted or edited, remove the image from the storage folder