PROJECT REPORT

Aston Events

Name: Bradley Willies

Student ID: 180212217

Link to website entry page: http://180212217.cs2410-web01pvm.aston.ac.uk/

Organiser credentials:

Username: bob.smith@email.com

Password: password

Link to GitHub source code: N/A, I have uploaded a zip file containing the source code.

Key Technologies and Structure of System

I have created my Aston Events website system using the technologies:

- Composer – to manage dependencies within the PHP project.
- Laravel – the PHP framework which creates the structure of the project and includes other technologies with it.
- Eloquent – to handle the object relational mapping of the database with the PHP models.
- MailJet – a web-email service.

See Appendix A which shows the system design for reference. I followed the "model view controller" architecture, which is standard when using Laravel. For the views, I split them into the authentication pages and the event viewing pages which a student would view (without being logged in), and the pages an event organiser could access. I used three controllers; one to handle authentication, one to handle CRUD activities on the events and another to handle any of the student page activities. The models just reflected the database tables I created.

Implemented Functions

Functions	Source Files
S1	resources/views/index.blade.php - mainly the table from line 70 to 88. Http/Controllers/StudentController.php - the index function from line 22 to 37.
S2	resources/views/index.blade.php - the form from line 23 to 68, specifically the div from line 24 to 35. Http/Controllers/StudentController.php - lines 26 to 28 apply the filtering by category.
S3	resources/views/index.blade.php - lines 36 to 57 contain the heading options to sort by and the sorting order (asc or desc). Http/Controllers/StudentController.php - lines 29 to 32 apply the ordering to the query.
S4	resources/views/index.blade.php - line 81 allows the user to click the event. resources/views/event/show.blade.php - lines 9 to 116 display information about the selected event. Lines 95 - 104 create the button used to add interest to the event.

	Http/Controllers/StudentController.php - lines 66 to 77 get the event details to pass to the view. Lines 83 - 89 add interest to the event.
S5	resources/views/auth/register.blade.php - view for registering to become an organiser. resources/views/layouts/app.blade.php - lines 42 - 53 provide the navigation buttons to register or login if the user isn't already logged in. Http/Controllers/Auth/RegisterController.php - creates the user/organiser and validates.
E1	resources/views/auth/login.blade.php - view for logging in. resources/views/layouts/app.blade.php - lines 58 - 68 provide a button to logout.
E2	resources/views/organiser/dashboard.blade.php - line 70 creates a button to take an organiser to the create event page. resources/views/organiser/event/create.blade.php - the view with the form for creating an event. Http/Controllers/Organiser/EventController.php - lines 56 to 72 validate the request with EventRequest then store the new event in the database.
E3	resources/views/organiser/dashboard.blade.php - lines 77 to 93 list all events by curent organiser. Http/Controllers/Organiser/EventController.php - lines 20 to 36 get the details for creating the view.
E4	resources/views/organiser/dashboard.blade.php - line 85 allows an organiser to edit their event. resources/views/organiser/event/show.blade.php - the view with the form for updating the chosen event. Http/Controllers/Organiser/EventController.php - function edit shows the form for editing the event and function update validates and updates the event in the database.

Implemented Security Features

Security Features	**Source Files**
Authentication/Authorisation	resources/views/auth - authentication views. routes/web.php - line 25 uses the auth middleware to authorise access. Lines 29 and 30 use the canEditEvent middleware. Http/Middleware/CanEditEvent.php - middleware to authorise an organiser to modify only their own events.
Form validation	The index view and views in the event and organiser directories contain forms with front-end validation. Http/Requests/EventRequest.php & Http/Requests/ContactFormRequest.php - rules function contains server-side validation.
Handle injections (SQL/HTML)	Http/Controllers/Organiser/EventController.php - Eloquent query builder protects against SQL injection.
Hash password	Http/Controllers/Auth/RegisterController.php - line 71.
Restrict file upload to only images	Http/Requests/EventRequest.php - line 33.
Cross-Site Request Forgery	Any view with a form has @csrf which enables the csrf middleware to validate the request.

Implemented Stretchers

Stretchers	**Source Files**
Tables sorted based on headings	resources/views/index.blade.php - lines 37 to 58. Http/Controllers/StudentController.php - lines 29 to 32 apply the sorting to the query. resources/views/organiser/dashboard.blade.php - lines 33 to 54. Http/Controllers/Organiser/EventController.php - lines 27 to 30 apply the sorting to the query.
Events have multiple pictures	resources/views/organiser/event/create.blade.php - line 62 allows "multiple" files and name[]. Http/Controllers/Organiser/EventController.php - store function. Http/Requests/EventRequest.php - line 33.
Student sends email to event organiser	Mail/ContactMail.php - the Mailable class. resources/views/event/contact.blade.php - contact view. resources/views/event/show.blade.php - lines 107 to 115. Http/Controllers/StudentController.php - showContact and sendContactMail functions. resources/views/mail/contact.blade.php - email template.
Organiser links one event to another event and/or web sources	resources/views/organiser/event/create.blade.php - lines 67 to 86. Http/Requests/EventRequest.php - lines 34 and 35. Models/Event.php - lines 23 and 24. resources/views/event/show.blade.php - lines 50 to 58 link to related event.

Database Schema

See Appendix B for the ERD describing the database schema.

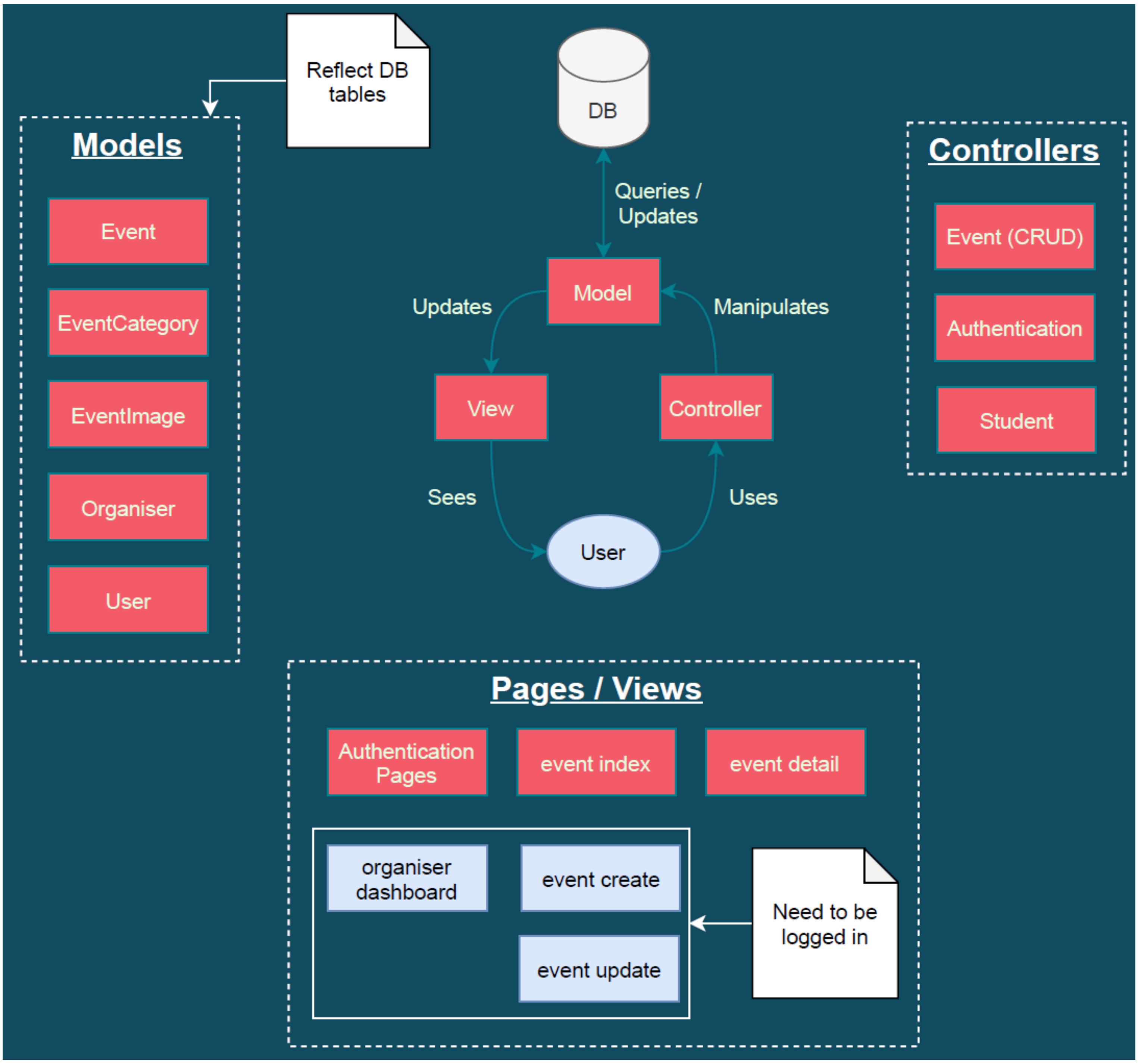
The users table is setup automatically with Laravel authentication, so it makes sense to use this to store new users, such as registered organisers. All authentication details are kept in the users table, while all details specific to an organiser are kept in the organisers table.

The related_event_id field in the events table optionally references another event's id.

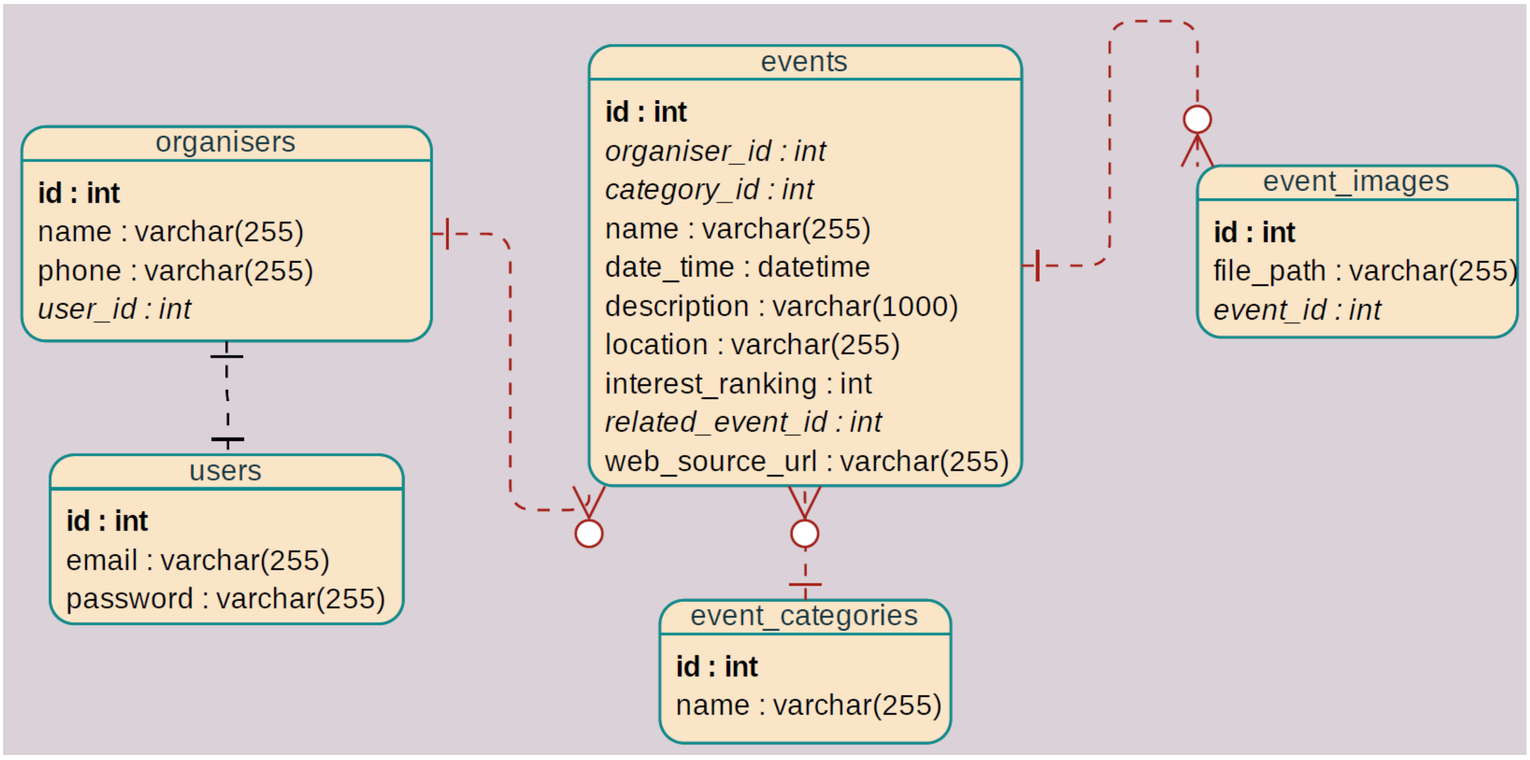
Points to Note When Running the System

- In order to create an event you are required to provide at least one image, along with other event details.
- The Contact form to send a message to an event organiser is live and uses MailJet with Laravel. It will send an email to the event organiser's email address. To test this; register as an organiser using your own email -> create an event -> go to the contact page for that event. When you send a message you should receive an email.

Appendix A: System Design



Appendix B: Entity Relationship Diagram


organisers
id : int
name : varchar(255)
phone : varchar(255)
user_id : int
users
id : int
email : varchar(255)
password : varchar(255)
events
id : int
organiser_id : int
category_id : int
name : varchar(255)
date_time : datetime
description : varchar(1000)
location : varchar(255)
interest_ranking : int
related_event_id : int
web_source_url : varchar(255)
event_categories
id : int
name : varchar(255)
event_images
id : int
file_path : varchar(255)
event_id : int