

Project Eve: A Group Report

Colm Cahalane, Eimear Crotty, Darragh King, Mervyn Galvin, Evan Smith

1. Introduction

1.1 Benefits of this project to potential users

1.2 Our goals for the project \ Educational merit

1.3 Potential Improvements

1.3.1 Our product and its features

1.3.2 Our approach to management

2. Project Overview

2.1 Our product specification

2.2 How it compares to other existing solutions

2.2.1 Examples of Existing Alternatives

2.2.2 Desirable features

2.2.3 Missing features

3. Justifying our approach

3.1 Our management style

3.1.1 Kanban

3.1.2 15 minute standups

3.1.3 Laissez faire + Evan as project manager

3.1.4 Project divided into requirements + optional requirements

3.1.5 Group coding sessions

3.2 Our technical stack

3.2.1 Programming Language - PHP

3.2.2 Framework - Laravel

MVC

The Eloquent ORM

Authentication

Migrations & Seeders

Artisan: A command line management tool

Testing Framework

Request Routing & Middleware

Security

Blade

Resource Management & Storage

Caching

3.2.3 Composer - Extending Laravel with Useful Packages

SimpleSoftware's Simple-QRCode

Eluceo iCal

Google API Client

Stripe PHP SDK

[Translation by SteveBauman](#)

[GoogIMapper](#)

[LaravelFacebookSDK](#)

[3.2.4 Design Framework - Materialize](#)

[The benefits of Material Design](#)

[The advantages of Materialize as a framework](#)

[3.2.5 Virtualized environments in VirtualBox and Vagrant](#)

[3.2.6 Database Layer - MySQL](#)

[3.2.7 Client-side assets: The Gulp approach](#)

[Why use LESS?](#)

[Why minify and merge JS?](#)

[Laravel Elixir](#)

[3.2.8 .editorconfig](#)

[The need for common coding style](#)

[Personal preferences in coding environment](#)

[Sublime Text](#)

[Package Control](#)

[Other capabilities](#)

[PHPStorm](#)

[3.2.9 Knockout.js](#)

[4. Requirements](#)

[Functional](#)

[Non-Functional](#)

[Usability](#)

[Learnable](#)

[Efficient](#)

[Memorable](#)

[Satisfactory](#)

[Efficiency](#)

[Maintainability](#)

[Open Source](#)

[Platform](#)

[Response Time](#)

[Security](#)

[Domain](#)

[Easy To Maintain and Learn OS](#)

[Scalability](#)

[Open Source](#)

[Packages](#)

[4.1 Understanding Use Cases](#)

[4.1.1 How We Found Our Three Use Case Groups](#)

[4.1.2 Administrators](#)

[4.1.3 Event Staff](#)

[4.1.4 Attendees](#)

[4.2 Narration of Proposed System](#)

[4.2.1 Events](#)

[4.2.2 Locations](#)

[4.2.3 News](#)

[4.2.4 Media](#)

[4.2.5 Admin](#)

[4.2.6 Staff](#)

[4.2.7 Users](#)

[5. Planning](#)

[5.1 Allocation of Roles](#)

[5.1.1 The Kanban Philosophy](#)

[5.1.2 The Weekly Standup](#)

[5.1.3 Taiga.io & Allocation](#)

[5.1.4 Team Leader - Kanban Sensei](#)

[5.1.5 Laissez-Faire?](#)

[5.2 The first steps - plan, teach, practice, research](#)

[5.3 Planning Structures](#)

[5.3.1 Milestone Structure](#)

[Milestone 1 - Minimum Viable Product](#)

[Milestone 2 - Payments & Stripe Integration](#)

[Milestone 3 - Google Places & Partners V2](#)

[Milestone 4 - Integrating social media](#)

[Milestone 5 - Optimising Events](#)

[5.4 Risks](#)

[5.3.1 Security](#)

[Laravel & Security](#)

[5.3.2 Time Management](#)

[5.3.3 Unification of Software](#)

[Problems with Git](#)

[How .editorconfig helps](#)

[6. Design](#)

[6.1 GUI](#)

[6.1.1 Google Material Design](#)

[6.1.2 Materialize - a CSS Framework implementation of Material](#)

[6.1.3 LESS Variables and their advantages](#)

[6.2 Database Structure & ER Diagram](#)

[6.3 UML Diagrams](#)

[6.4 Form Fields and Validation](#)

[6.4.1 Laravel Validator Syntax](#)

[6.4.2 Our Fields and Validators:](#)

[Users \(registration\)](#)

[Users \(profile\):](#)

[Company:](#)

[Events:](#)

[Locations:](#)

[News:](#)

[Partners:](#)

[7. Software Deployment](#)

[7.1 Target Environment](#)

[7.2 Deployment of Components](#)

[7.3 Client/Server Configuration](#)

[7.4 Installation \(Production\)](#)

[7.5 Updating](#)

[7.6 Uninstallation](#)

[8. Testing](#)

[8.1 Our Testing Suite](#)

[8.1.1 Staging server - Test by use](#)

[8.2 Test Data](#)

[8.3 Test Results](#)

[8.4 Log of bug reports](#)

[9. The User Manual](#)

[9.1 Server Administrator Manual](#)

[9.1.1 Deployment](#)

[9.1.2 Installation](#)

[9.1.3 Support](#)

[9.1.4 Stripe](#)

[9.2 Site Administrator Manual](#)

[9.2.1 Events](#)

[9.2.2 Locations](#)

[9.2.3 Partners](#)

[9.2.4 Media](#)

[9.2.5 Users](#)

[9.2.6 Tickets](#)

[9.3 Event Staff Manual](#)

[The Scanned Tickets System](#)

[Manually approving tickets](#)

[9.4 User Manual](#)

[10. Evaluation](#)

[10.1 Product](#)

[10.2 Development activity](#)

[10.3 Software used](#)

[10.3.1 Communication](#)

[10.3.2 Code Collaboration](#)

[10.3.3 Task Management](#)

[10.4 What we'd do differently](#)

[10.4.1 What we'd prefer](#)

[10.4.2 What we'd change](#)

[10.5 Conclusions](#)

[Appendix B: Log Of Bug System](#)

[Final Tally](#)

[Open Issues](#)

[#6 Incorrect login, displays errors on main page instead of modal](#)

[#12 Breadcrumbs on every page](#)

[#14 PDF Tickets](#)

[#22 Sign maps key for better security](#)

[Closed Issues](#)

[#1 Obfuscate ticket number in link](#)

[#2 Key constraint restricts image upload](#)

[#3 Search function for users](#)

[#5 Print ticket maximises when 'cancel' is pressed](#)

[#7 Empty message when nothing is found](#)

[#8 No way to get to your past events](#)

[#9 Need to add a way for users to add their photos and media to events](#)

[#10 tagline not stored when creating or updating event](#)

[#11 need to create some sort of button for creating event from index page](#)

[#13 QR Code is different every time I reload the page](#)

[#15 if incorrect creation of event, returns to main page with errors instead of create page](#)

[#16 Events are unviewable](#)

[#17 Creating new location from modal results in 500 error](#)

[#18 Undefined route error after logout](#)

[#19 Can't delete partner that is associated with events](#)

[#20 media/upload route undefined error when logged in as regular user](#)

[#21 Viewing Tickets should be encrypted](#)

[#24 No edit staff page](#)

[#25 payment for event not working](#)

[#26 event creation not working](#)

[#27 'cancel' icon changes to 'C'](#)

[#28 any user allowed to begin to create event](#)

[#29 Partner ID is required](#)

[#30 not able to read pdf document as photo](#)

[#31 create location modal not working on event edit page](#)

[#32 upper limit of distance between partner and event.](#)

[#33 event creation allowed without start, end date or feature image](#)

- [#34 negative value allowed for ticket price](#)
- [#35 Location Modal: Errors not clearing + Prominent](#)
- [#36 Events Creation: Doesn't check valid time](#)
- [#37 Partners: Error when creating a partner](#)
- [#38 Locations](#)
- [#39 Admin Page: Add staff button goes nowhere](#)

[Appendix D: Open Source Software Licences](#)

[Laravel](#)
[SimpleQRCode by SimpleSoftware](#)
[GoogIMapper by Brad Cornford](#)
[Translation by Steve Bauman](#)
[Laravel Facebook SDK by SammyK](#)
[Guzzle](#)
[Stripe-PHP by Stripe](#)
[Google ApiClient by Pulkit Jalan](#)
[iCal by Eluceo](#)
[jQuery & jQuery UI](#)
[Dropzone.js](#)
[Magnific Popup](#)
[Clockpicker.js](#)
[Facebook JS SDK](#)
[Google Maps JS SDK](#)
[Knockout.js](#)
[Materialize](#)

1. Introduction

1.1 Benefits of this project to potential users

Project Eve is an all-in-one solution to event organisation for kinds of groups and companies. Eve can be set up in minutes, is easy for users, admins and staff to use and is extremely secure.

1.2 Our goals for the project \ Educational merit

Throughout the project we aimed to learn about:

- working in a group
- collaborating on code through github
- how to use the Laravel framework
- how to use tools like composer, git and vagrant
- how to handle common errors in our tools
- how to utilize MaterializeCSS to it's fullest extent
- and how to write and debug javascript

1.3 Potential Improvements

1.3.1 Our product and its features

Ideally, our product would've included the ability to book hotels in the vicinity of the event without requiring the user to research for themselves. This could've been implemented in various ways. When we looked at TripAdvisor's API, we realized that we wouldn't be able to get our foot in the door: the requirements were far too restrictive and waiting to (maybe) be approved to access the API was not an option for such a time-sensitive project. We could have used the HotelsCombined API but honestly, it wasn't even considered as we hadn't really heard about it and didn't know it existed until late into the project.

Something that we neglected throughout the development process was usability testing. We, unlike some other groups, weren't able to readily identify what visual aspects of the project would be difficult for a new user to wrap their head around. Also, the on-boarding process for staff was completely an afterthought and as a result, the only guidance they have is the documentation and the knowledge of the administration staff.

Finally, with the project specification being somewhat vague, we couldn't afford to make our goals too ambitious, and risk wasting time in ambitious features. This may have been a mistake, as it may have left us unable to implement all the features we wished to.

1.3.2 Our approach to management

We found using Kanban to be the most natural way for us to organise and assign our work, but if the team were much bigger (like in enterprise) this likely wouldn't work. Group coding sessions were incredibly useful when they took place. Unfortunately, due to external forces like exams and sickness, these didn't happen as often as we would have liked. In retrospect, we could have organised more than one coding session a week but given that we were already meeting informally almost every day, this felt excessive.

For most of us, implementing these features were unlike anything we'd done before. Therefore, it was difficult to assess whether we were over or underestimating our expected finish time. This is most evidenced by the difference in Stripe integration's estimate and actual completion time. This did, however, often allow us to allocate more time to testing the new features that proved to be invaluable in getting a stable version of the product out in time.

2. Project Overview

2.1 Our product specification

The project specification we were given allowed for some flexibility in its interpretation. The core requirements were to create a working website that could promote events and provide a means by which attendees could sign up for said events.

The project was broken down into core and optional requirements. We took these requirements and broke them into achievable goals which we labeled milestones. These milestones were used to break development of the site into workable chunks. Using laravel as our framework along with additional useful and intuitive technologies we managed to accomplish almost all of our original goals, as well as some additions that were made to our plan along the way.

Our personalised specification involved creating an adaptable, multipurpose event site, with a social media aspect to it. Our adapted specification was driven heavily by the philosophy that functionality should take precedence over additional features. That being that everything should work as intended and the site should be as optimised as possible.

Our rough primary objective (A minimal viable product)

- Install/landing page.
- A viewable website where events can be advertised.
- User profiles.
- Ability to register for events via user profiles, with confirmation email.
- Payments using stripe.

- Printable tickets.
- Integration with various calendars.
- Reception desk interface to allow for tickets to be scanned and viewed.
- Information pack for visitors.
- Ability to contact event organisers or developers.

Our secondary and additional objectives.

- Admin panel for control.
- Partners - Integrate interactive maps into the website using Google Maps/Places.
- Social media integration. (Sign up using facebook)
- Addition user features. (Search for friends)

Final objective (Optimisation)

- Gather information from the internet. (Web-Crawling)

2.2 How it compares to other existing solutions

2.2.1 Examples of Existing Alternatives

There is quite a substantial amount of already-existing event websites.

Some examples include:

- "Eventbrite" which allows you to create your own events and get tickets to popular events.
- "Ticketmaster" which is chiefly for obtaining tickets to already popular events such as concerts, comedy events, etc.
- "dublinevents" which is closest in likelihood to our own project specification. However, this site is less popular than the aforementioned two and is apparently lacking in some features such as a user profile.

2.2.2 Desirable features

- Eventbrite allows any user to quickly and effectively create an event for any occasion
 - We modeled some of our site using this idea - an event can be created on the site for any kind of occasion.
 - However, only registered event organisers with the correct permissions have the capability to create and update events.
- One unique feature of our site, in comparison to the examples given, is a public user profile. This gives our site a slight social media feel to it. We felt it necessary to include such a feature for our project as it was our interpretation that if you were to go to an event and meet someone whom you forgot to exchange contact details with prior to the event end, you could simply visit our site and either search for their name or look up the attendees of the event. You could then find their profile and obtain information necessary to contact the user.
- An additional feature involved allowing users to upload pictures related to the event to the event's page. This furthered the social aspect of our site.

2.2.3 Missing features

Some proposed features of the site were scrapped during the development process, either due to time constraints or some other restriction.

Some of these features included developing some method by which our site could gather relevant information from the web. This information could be used to:

- assist the event organiser in determining a suitable venue.
- help visitors find the best modes of transport/accommodation available for the event, etc.

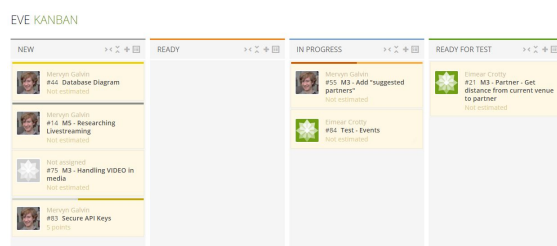
3. Justifying our approach

3.1 Our management style

3.1.1 Kanban

Due to the overall structure of the project specification, dividing the project into mandatory and optional requirements, we wanted a system by which we could ensure that we could change our approach on a week-by-week basis depending on progress on each set of requirements. Waterfall would not work here, as the design would change over time based on what was possible for us.

The first thing that we did was order the tasks into individual Milestones based on their viability and priority. While this gave us an overall strategy, we still had to figure out how to balance tasks between our team. It was from this need that we decided on a **Kanban** structure. Kanban is a technique used in *lean* environments, where an initial Minimal Viable Product (MVP) is developed and expanded upon as requirements change. We saw the mandatory/optional requirements split as similar to lean, so we decided to go towards that.



Kanban is sometimes described as chaotic. The near-flat management structure, the use of a visual board for task organisation, and the rapidly changing task assignments are too complicated for many major businesses. But in a team of five with a dedicated *Kanban Sensei*, we didn't encounter these problems.

The name *kanban* (かんばん) actually translates to “visual board”, referring to the visual board that's used in assignments. During our first week, we didn't use a board, following a more traditional Agile approach. We moved towards Kanban after discovering the Taiga.io utility for designing these boards. We found that the utility offered a clean, at-a-glance overview of the project, allowed us to have ordered, sorted discussions, and allow us to stay updated by email if we ever had to miss a meeting.

Kanban gave us agility. As the project developed, we gained a better sense of which team members were suited to individual tasks, but we were able to assign these tasks without tying the team down to individual roles. For instance, we had one member of the team with a particular flair for visual design and layout. We were able to assign her many of the design-oriented tasks, but we also kept her involved in writing code, and working on much of the User backend. In another structure, the value of this would've been lost.

3.1.2 15 minute standups

We used the 15 minute standup format for our weekly meetings every Monday. What this meant was that at every meeting, the team members would explain the work that they'd done in the week before, and their aims and goals for the week ahead. We'd do all of this with the aid of the Git commit log, and the Taiga.io Kanban board. Following this process, the group, led by Evan, would decide the tasks for the following week, the priorities associated with them, and who would work on these tasks (either alone or collaboratively).

The standup format perfectly matched up with the overall Kanban philosophy of the group. It allowed the group to be "in sync" - everyone on the group was aware of what progress was being made by who in the group, so we wouldn't worry about duplication of work, miscommunication, or people being assigned to tasks they were uncomfortable with.

3.1.3 Laissez faire + Evan as project manager

The primary goal of this project was for each of us to learn more about professional software development, but we each had different goals.

- Evan had already worked on Laravel projects in the past, and was more interested in learning about management and group dynamics, as well as tackling new and more interesting features like internationalisation.
- Colm had worked in Laravel before but had a specific interest in integrations and APIs, and wanted to develop in these areas.
- Darragh hadn't worked in an MVC framework before, and wanted to improve his skills in JavaScript as well.
- Eimear was interested in learning about industry practices and MVC as well as working with integrations.
- Mervyn had an interest in database modelling, but also wanted to experiment with 3rd party integrations and APIs.

The use of a laissez-faire structure, where team members would discuss the things that they'd like to do on a week by week basis, allowed for a natural structure to emerge within the team and for motivation within the team to remain high throughout the project.

However, in a purely laissez-faire system, there's no guarantee that the project's goals will be achieved. For instance, everyone had a general interest in learning about the integrations and APIs. The use of Evan as a project manager, setting the

main strategic goals on a week-by-week basis, allowed for balance. We would work towards our core goals on a week-by-week basis, and ensure that all required tasks were assigned, but generally speaking everyone was capable of learning about the project on their own terms.

3.1.4 Project divided into requirements + optional requirements

In order to ensure quality of product delivered, we wanted to ensure that at any point after the early stage of the product, it would be possible to develop a functioning piece of software. In this sense, it would make sense to develop a solid foundation first, and have all requirements in the product specification included in this foundation. Upon completion of this, we could start working on advanced functionality.

We believe this was a necessary step for our project. It gave us the necessary flexibility to work on ambitious features, but it allowed us to change our direction depending on our available time and resources.

3.1.5 Group coding sessions

Group coding sessions were one of the best decisions that we've made. It allowed us to resolve any issues that the team had, work faster than we would if there were any distractions, and allow the team to work together and check in mid-week. We also were able to educate each other and work collaboratively on resolving tough bugs or programming problems. It gave us a major advantage that we simply wouldn't have had otherwise, and more code was written during the Thursday 2pm/3pm block, which we assigned to group coding, than during any other time of day according to our Github statistics.

The only downside was that due to unexpected circumstances and challenges relating to the semester, we were not able to reliably hold this session every week. Also not every team member had access to a laptop to work on, and it was remarkably difficult to find a room to work from given that labs were crowded and all lecture rooms tended to be in use at that time.

3.2 Our technical stack

3.2.1 Programming Language - PHP

The PHP programming language has been a controversial choice among programmers who identify faults with its procedural style, security, and lack of strict typing. However, more recent versions of the programming language give it strong object-orientation, stronger password hashing, increased speed, and the option to use static typing. Though there are certainly still some design faults and inconsistencies with PHP as a language, many of these are controlled and avoided by our use of a framework.

We use PHP in spite of some of its failures because it gives us access to the environment necessary for rapid development. All five members of our development

team were already familiar with PHP. PHP is portable, well-supported and easy to understand. It integrates excellently with our favourite development environments and can easily be packaged into a Vagrant box or deployed online to an Azure server. Community support, including the Laravel framework and the number of Composer packages available for use make the choice to use PHP a fairly obvious one.

Though Python is a far more accessible language and less prone to errors, web frameworks for Java such as Django are not quite as accessible or oriented to rapid development. Though Java is a far better implementation of object-oriented programming, it is much more difficult to implement a web application in Java. Our group didn't have the Javascript experience necessary to attempt a Node.JS app, and we simply didn't see the benefits of Ruby on Rails to consider using it as our backend language.

3.2.2 Framework - Laravel

Laravel is a popular PHP framework that's gathered significant interest for how accommodating it is to rapid development. Some of our group had experimented with Laravel projects in the past, including a project built for Netsoc called "Lowdown".

Laravel has a particularly excellent documentation resource as well as a set of tutorial videos (Laracasts) available online for free. We believed that this was a vital step as it allowed us all to be up to scratch on Laravel development as quickly as possible.

Laravel has a beautiful approach to syntax, file management, and has automatically included a series of powerful features. We'll discuss them on a case by case basis here.

MVC

Laravel is an implementation of the Model-View-Controller architecture. What this means is that we divide the applications code into:

- **Model** - code that represents how data is represented and stored within the application.
- **View** - code that represents how data is displayed to users.
- **Controller** - code that represents how data is retrieved, modified or otherwise acted upon by users.

In Laravel, this distinction is created as:

- **Models** define the objects the application deal with. They depend on **Migrations** which define the database representation of these objects. In the Model, we define which fields in the object are mass-assignable, which should be private and not exposed to users in the API, and any mutator methods that define how attributes are accessed if they don't just simply follow the contents of the database row.
- **Views** are HTML documents extended by *Laravel Blade*, a templating engine. This allows us to define the ways in which data are returned to the user.

- **Controllers** are PHP classes that contain methods. In Laravel, “*routes*” are defined that pass HTTP request data into methods within these controllers. Data processing is performed within these methods and a view or other result is returned.

This very clear, very efficient approach to MVC made this framework a perfect choice for our team of mixed backgrounds and abilities.

The Eloquent ORM

ORM means Object-Relational Mapping, and refers to the mapping of Laravel models to the data that are stored in the database. The Eloquent ORM means that we have access to data without actually writing SQL queries.

For instance, to access a particular instance of the User model, we can type:

```
$user = User::find($userId);
```

We can query for users that fit specific parameters:

```
$user = User::where(name, 'Colm')
    ->where(city, 'Cork')
    ->get();
```

We can easily access attributes of these items:

```
$userName = $user->name;
```

Or get related entities and their attributes:

```
$eventLocation = $ticket->event->location;
```

We can update models and save their information to the database easily as well. We can even define, in Models, methods that process data before it's saved to the database.

```
$user->name = 'Darragh';
$user->save();
```

Laravel Models allow for representation of one-to-one, one-to-many and many-to-many relationships with ease.

This clear syntax gives Laravel an immediate benefit over any other framework we could work with. It's remarkably easy to pick up and learn the syntax, and makes the controller methods readable, maintainable and functional.

Authentication

Laravel already includes a default Auth package and User model. As such by default we can create users by POST request to /register and login users by sending POST requests to /login. But we also have access to the Auth functions to implement and include in other functions. In this way we have a completely portable and secure Authentication package without having to create and expand upon our own in development. For instance when we started using the Facebook API for Facebook login, it was remarkably simple to include the authentication features into our Facebook controller.

```
$user = User::createOrUpdateGraphNode($fbData);  
Auth::login($user);
```

Migrations & Seeders

Although we settled on MySQL, Laravel is database-agnostic and will work on any variety of platforms. Migrations are files that define the database structure of the application. They act in a versioned manner: to make changes to the database, simply create a new migration class file with an `up()` method (which executes tasks on the database such as creating or altering tables) and a `down()` method (which should undo the changes so that any changes to the database can be rolled back). The syntax that Laravel defines for migrations is similar to but not the same as the MySQL `CREATE TABLE` syntax. For instance, you can create columns like so:

```
$table->integer('quantity');
```

Which would create a column called “quantity” in the table, with type Integer. However, if we were to create a column like so:

```
$table->json('attributes');
```

It would create a text-type column called “attributes” in MySQL, but a json-type column in PostgreSQL which supports json-type columns. *However*, Laravel will still perform the validation and serialization steps necessary to use the data contained in this row in the same manner regardless of the database backend that we use.

This system works with the Artisan command line management tool and ensures that we have the most optimal database layout.

Seeders are another database system that exists in Laravel. These are simply scripts that populate a database. We use a variety of seeders to rapidly populate our website with content so that we can test the site’s functionality.

Artisan: A command line management tool

At the root of a Laravel project is an executable PHP script that is to be launched from the command line called `artisan`. Artisan makes it remarkably simple to perform a lot of tasks in accordance with best practices. Here are some examples:

- `php artisan make:controller UserController`
 - Makes a new Controller file for users with the required class structure already implemented.
- `php artisan migrate`
 - Executes any new Migration files.
- `php artisan db:seed`
 - Seeds the database using the existing Seeder files.
- `php artisan migrate:rollback`
 - Undo the most recently executed migrations.
- `php artisan migrate:refresh --seed`
 - Roll back and then immediately execute all migrations, then seed the database.
- `php artisan clear-compiled`
 - Clear the compiled autoload file.
- `php artisan cache:clear`

- Clear any data from Laravel's cache.

The functionality and ease-of-use offered by Artisan, and the fact that we can programmatically call these functions in shell scripts or cron jobs, makes this feature a fundamental aspect of why we chose Laravel over another web development Framework.

Testing Framework

Laravel provides a testing framework that can be executed from Artisan. This includes PHPUnit tests for unit testing, but also an API for making HTTP requests to the server as a user would and testing the responses to see if the data they contain matches the expected results.

Laravel also provides the Factory pattern for creating large numbers of test users or other models and uses a package called Faker that can generate this data automatically.

These features allowed us efficient ways to test the site automatically as we made changes to it.

Laravel also adds a number of useful features to PHP including smart handling of error reports that include a java-style stack trace, allowing the causes of error to be rapidly identified.

Request Routing & Middleware

Laravel allows you to route requests to functions easily using a single routes.php file that allows for passing this request to the required controller function. As well as this, it allows us to filter requests through generic "middleware" functions that perform some functions such as verifying CSRF protection, gating some routes to only administrator users, verifying authentication and cookies, etc. This setup gives us unparalleled flexibility in implementing security and verification at various steps of the request lifecycle.

Security

Laravel uses the Crypt encryption library that implements the AES-256-CBC cipher for all password hashing and also allows us to encrypt request/response parameters. We use the Crypt library to encrypt ticket information so that only our application is capable of producing verifiable tickets. We can simply use `Crypt::encrypt($t)` and `Crypt::decrypt($encryptedT)` to serialize/deserialize and convert the ticket.

Blade

Blade provides us with a syntax for writing views in HTML extended with some PHP functionality. This means that we can create individual components of views and include/extend them, meaning that we have a standard App layout, that can be extended upon by the various different types of views.

From there, we can use PHP loops, conditionals and references to bring content into views.

Another excellent element of Blade is that it sanitises all output so that no malicious scripts can be injected into the page. We can also use PHP expressions, so we can implement features like translation into the page.

Laravel also has some excellent features relating to HTML and Forms. For instance, Laravel provides a Form facade that allows us to populate a form easily, but also, if we need to redirect back to that form, we can show the errors that Laravel's validator returns and highlight the form fields that need closer attention.

Here is an example of a Blade template:

```
@extends('layouts.app')
@section('body-class') create-event @endsection
@section('title') {{_t('Create an Event')}} @endsection

@section('content')
<!-- Create a new event title -->
<div class="row">
    <!-- Open a new form, allowing for file uploads -->
    {!! Form::open( ['route' => 'events.store', 'files' => true] ) !!}

    <div class="row">
        <!-- Input title -->
        <div class="input-field col m6 s12">
            {!! Form::label('title',_t('Event Title')) !!}
            {!! Form::text('title') !!}
        </div>
        <!-- Input tagline -->
        <div class="input-field col m6 s12">
            {!! Form::label('tagline',_t('Event Tagline')) !!}
            {!! Form::text('tagline') !!}
        </div>
        <!-- Input description -->
        <div class="input-field col m12 s12">
            <h5>{{_t('Description:')}}</h5>
            <div class="editable content" id="description">
                <p>
                    {{_t('Start typing your description here!')}}
                </p>
            </div>
        </div>
    </div>

    <div class="row">
        <!-- Select location or create a new location -->
        <div class="input-field col s6">
            <select name="location_id" id="location-select">
                <option value="" disabled selected>{{_t('Choose
location')}}</option>
                @foreach($locations as $location)
                    <option value="{{ $location->id }}">{{ $location->name
                @endforeach
            </select>
        </div>
    </div>
</div>
```

```

        <option value="-1" >{{_t('Create New Location')}}</option>
    </select>
    <label>{{_t('Location Select')}}</label>
</div>
<!-- Choose a partner for the event -->
<div class="input-field col s6">
    <select multiple name="partner_id[]" id="partner-select">
        <option value="" disabled selected>{{_t('Choose
partner')}}</option>
        @foreach($partners as $partner)
            <option value="{{ $partner->id }}">{{ $partner->name }}</option>
        @endforeach
    </select>
    <label>{{_t('Partner Select')}}</label>
</div>
</div>
{!! Form::close() !!}
</div>
@include("forms.locationmodal")
@endsection

```

Notice the use of statements like `@foreach` and `@extends('layout.app')`, bringing logic and control flow into the system.

The availability of the Blade templating tool, its flexibility, and the ease of learning it, makes it an invaluable reason to use Laravel.

Resource Management & Storage

Laravel has a good approach to storage, providing nice wrappers for file uploads, and referencing uploaded files in other sections of the app. What's more, we can configure the Laravel environment to store these assets on cloud storage such as Amazon S3 or Rackspace. It holds a "public" folder for static assets, and provides the Laravel Elixir system for handling static assets in Gulp.

Caching

Laravel provides a clean caching system that works on a few different levels. We can access a key-value store using the statements `Cache::set($key, $value[, $storageDuration])` and `Cache::get($key, $default)`. Blade templates are processed and cached views are generated from the content they develop. Although these caches are stored in the filesystem by default, we worked a way to store the cache values in memory for fast access using Redis, which Laravel easily interfaces with. Redis is a simple, fast key-value store that runs in memory, and easily adapts to changing requirements such as available memory and volume of requests.

3.2.3 Composer - Extending Laravel with Useful Packages

A second reason why we chose PHP and Laravel is Composer. Composer is a package manager for PHP libraries and classes that generates an auto-load file so the packages we use can easily be included within the project. We chose a number

of Composer packages to extend upon Laravel's functionality and provide us with assistance in delivering all the functionality that is required.

SimpleSoftware's Simple-QRCode

SimpleQRCode generates QR Codes in SVG format. By wrapping this functionality into the Ticket model and creating a `qr()` method, we can output the generated QR code into ticket pages using the call `{!! $ticket->qr() !!}`. As a result, this package is essential to our implementation of ticket codes.

Eluceo iCal

The iCal package allows us to translate datetime, and location data into Calendar files that can be read and understood by Google Calendar, Apple iCal, and other providers. This is a remarkably useful feature and is easily implemented.

Google API Client

The Google API Client package provides easy communication with many of Google's APIs (such as Maps and Places) on the server-side. We use it to query for distance and other similar features. It dramatically simplified our process as Google APIs can be at times difficult to deal with.

Stripe PHP SDK

Stripe provides a SDK that wraps many of its payment processing API calls into PHP functions. We chose this implementation (over more feature-filled utilities like Laravel Cashier or Cartalyst for Laravel 5) because it allowed us to implement Stripe payments in our project with just two calls, and also gave us a wide range of exceptions to catch if errors arise (Card rejected, Stripe errors etc.). All in all, for use of Stripe PHP SDK, we were able to implement payments in our project with only 54 extra lines of code.

Translation by SteveBauman

This Translation package generates a database cache of translation strings based on information returned from the Google Translate API. This is hugely valuable as it allows us to simply wrap strings and have them returned in the user's preferred language straight away, without us having to translate strings ourselves. It gives us a quick, simple implementation of translation.

A Laravel Middleware class is provided which handles checking the user's locale. From there, we only need to wrap strings in the `_t($string)` function throughout the app and its views and the string will be translated accordingly. As well as its simplicity, we chose this particular package due to the sheer number of languages available within it.



GooglMapper

GooglMapper provides a Laravel syntax for outputting Google Maps in views, by providing a Facade. When installed and included `{!! Mapper::map($latitude, $longitude) !!}` will include in a view a map with a marker on that individual point. Further methods exist for placing more map markers on that map, or drawing areas - making it an important tool in generating our information packs and event pages.

LaravelFacebookSDK

Laravel comes with a Facebook Authentication SDK called Laravel Socialite. However, this alone is not enough as we wished to make further requests to the Facebook SDK to allow users to view which of their friends are attending events. Hence, we decided to use the more feature-filled LaravelFacebookSDK instead, which acts as a wrapper to Facebook's own PHP SDK, but allows us to use it like a Laravel Facade.

This SDK also integrates with Facebook's own Javascript SDK. This allows us to send an authentication token to a route that hooks into our FacebookController and authorises a user from their Facebook token. If their account doesn't exist in Laravel, we create a new user account, and populate it with data from Facebook. In this way, anyone can log into the system with just one click, if they have an account with us or not.

From there, there's a three-line process to get their friends list:

```
$response = $fb->get('/me/friends?limit=5000&fields=id');
$friendData = $response->getDecodedBody()["data"];
$friendIds = array_flatten($friendData);
```

The ease of implementation of social media features made this package essential.

3.2.4 Design Framework - Materialize

The benefits of Material Design

Google announced Material Design as the basis of Android version 5.0 (Lollipop). It's a design system based on the metaphor of overlapping materials, designed as a cross-platform design system that is equally designed for and responsive to touch and keyboard/mouse interaction. This means that the sparse, minimal elements of flat design systems are augmented by hierarchical design, animation, motion, the use of shadow and bold colour.¹ The goal is fluid design that responds to user interaction.

We chose the Material style due to its growing popularity on the web and because it provided us with a valuable set of guidelines to follow to ensure a consistent user experience across different devices.

The advantages of Materialize as a framework

MaterializeCSS² is a CSS framework, meaning that it is CSS code that we can include directly into the project to provide us with a starting point for our product design. MaterializeCSS includes implementations of Material Design concepts such as responsive typography, colour palettes, components such as cards, galleries, collections, dropdown lists, modal dialog boxes etc. It provides useful interactive Javascript elements, it provides animations on interactions such as button presses and loading-bar elements etc. It provides Material-compliant styling for default HTML components such as form fields. It even provides helper classes for parallax-scrolling elements and vertically-aligned elements.

Most importantly by far however is its implementation of the responsive grid system. A `<div class="row">` can include `<div class="col ...">` elements of varying size that are organised horizontally according to their size. A row is twelve columns wide.

Columns are given size depending on screen size: small screens (e.g. mobile phones), medium screens (e.g. tablet devices) and large screens (typically a desktop computer).

As such, a `col s12 m6 l3` element would take up the full row on mobile devices, half the screen on tablets, and quarter of the screen on desktop browsers. This gives us immense flexibility in designing responsive layouts that simply work across all devices.

We chose Materialize as a way to ensure that we could comply with our design guidelines, but also because it provides a simpler, cleaner syntax for grids and components than competitors such as Bootstrap. Further to this, the design of

¹ <https://www.google.com/design/spec/material-design/>

² <http://materializecss.com>

components in Materialize is simply more aesthetically pleasing, and Materialize is actually a smaller CSS/JS framework with less unnecessary bloat.

3.2.5 Virtualized environments in VirtualBox and Vagrant

As our deployment environment is a Ubuntu box running a typical LAMP³ stack, we want to emulate this environment throughout our testing and development stages to ensure that no production-specific bugs appear. Generally speaking, this is difficult. While we can emulate the production environment with a virtual machine running in Virtualbox, synchronising the changes as the developer makes them is not a trivial process.

Vagrant simplifies this process. At the root of the project folder is a Vagrantfile, which includes the necessary instructions to create and provision a virtual machine, as well as synchronising the files between host and development machine. Each developer in the project therefore can run `vagrant up` (assuming they have Virtualbox and Vagrant both installed) and run the web application in a perfectly identical development environment, including the LAMP stack. If we need to execute Artisan commands, most will work both inside and outside the box, but we can also use `vagrant ssh` to execute commands on the virtual machine if we need to run commands which alter the database.

3.2.6 Database Layer - MySQL

MySQL is an industry standard relational database management system. We chose MySQL for the security it offers, the immediate compatibility with Laravel and the wide variety of tools we could use to interface with the database directly where needed, such as Navicat, DataGrip or HeidiSQL. With all members of the group having a familiarity with this system through CS1106⁴ and CS2501⁵ it was an obvious choice.

3.2.7 Client-side assets: The Gulp approach

Why use LESS?

LESS is a pre-processor language for CSS, extending the features of CSS by adding variables, macros, includes and the option for writing code in a nested manner. This makes our CSS code easier to write, understand, collect and maintain. The use of global variables in LESS gives us the ability to set style standards throughout the app.

Pure CSS <pre>.card{ background:#212121; color:#ffff8 }</pre>	LESS <pre>.card{ background:@off-black; color:@off-white;</pre>
-----------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------

³ Linux, Apache, MySQL, PHP

⁴ Introduction to Relational Database Systems

⁵ Database Design and Administration

<pre>.card h1,h2,h3,h4,h5,h6{ font-family: Montserrat, Lato, Arial, Helvetica Neue, sans-serif; } .card p{ Font-family: Roboto, Lato, Arial, Helvetica Neue, sans-serif; }</pre>	<pre>h1,h2,h3,h4,h5,h6{ font-family: @headings; } p{ font-family: @body-text; }</pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------

Why minify and merge JS?

To take advantage of reducing the number of HTTP requests, gain the advantage of caching, and as such gain speedup for the client-side application, it makes sense for us to reduce our app's JavaScript to as few files as possible, and to reduce the length of variable names and eliminate whitespace so that the file size is reduced as much as possible. However, actually writing minified JS by hand is impossible, and writing our code in a single JavaScript file will lead to inevitable merge conflicts and lost code. The solution therefore is to have a folder which holds the Javascript for the app spread across different files, and have a process by which the Javascript is collected, merged, minified, and output in a publicly accessible folder on the server.

Laravel Elixir

The final piece of the issue is Laravel. It comes with Elixir, which is a wrapper for setting up Gulp, a build system. Elixir "mixes" JavaScript and LESS files and outputs the merged files. We get all of the benefits of well-versioned, maintainable client-side assets and faster client-side performance. With Gulp installed, we can simply run `gulp watch` and the public assets will update as the source files are changed.

3.2.8 .editorconfig

The need for common coding style

A group project will require the team to operate in accordance with common coding rules to ensure predictable, maintainable, comprehensible code. This will include details such as tabs versus indentation for spacing, naming conventions, and the text case used for variables and functions (snake_case versus camelCase versus PascalCase). We agreed some rules at the start of the project, but these must be enforced.

`.editorconfig` is a file that sits at the root of the project and when the project is added to the user's text editor of choice, it applies the settings necessary to make sure that these rules are abided by.

```
# Example of an editorconfig file
```

```
# editorconfig.org
root = true

[*]
end_of_line = lf
insert_final_newline = true
charset = utf-8
indent_style = tab
trim_trailing_whitespace = true

[composer.json]
indent_style = space
indent_size = 2

[*.{md,bat}]
end_of_line = crlf
```

.editorconfig is supported by all IDEs and Sublime Text.

Personal preferences in coding environment

Generally in a group it's not possible to ensure that all team members will use the exact same environment for writing code. Each individual has preferences for the editor that they'd like to use, and different reasons for doing so.

Sublime Text

Sublime Text is used by most members of our team. It's a lightweight, extensible text editor with a major focus on providing a distraction-free coding environment.

Package Control

The popular Package Control extension for Sublime Text adds a package repository filled with useful extensions for the editor. Examples include Autocomplete Everywhere, BracketHighlighter, ColorHighlighter, Emmet, PHPLint, Laravel Blade Syntax Highlighting, and even a spell-checker.

Other capabilities

Sublime Text's core features like multi-line editing, Vintage Mode (the option to use Vi-style navigation in Sublime text), the Goto Anywhere menu for locating any file within the project and the code mini-map (an at-a-glance look at the file's structure) give it a major advantage for some team members.

PHPStorm

PHPStorm is the JetBrains IDE for PHP development and includes out-of-box support for Laravel Blade Syntax. As a fully-featured IDE it includes code-linting, autocomplete, git integration, gulp integration, Vagrant controls, a terminal and various refactoring tools. Some of our group prefer it for its design and general approach but it has a "busier" appearance and lacks features like Vintage Mode and the robust multi-line editing support found in Sublime Text.

3.2.9 Knockout.js

Knockout.js is a front-end framework built on the Model-View-ViewModel (MVVM) allowing user-facing views to update based on changes of observable variables. This is used in our project as the support for the self-updating Scanned Tickets page. The View-Model would update the list of tickets every three seconds, and this would be reflected in the view.

Knockout Views are written in a way that is very similar to the Blade syntax, where comments and bound elements are written up in HTML syntax, and this is read and understood by the Knockout library.

This is an example of a Knockout-ready layout.

```
<!-- ko foreach: tickets -->
  <li class="collection-item" data-bind="attr: {'id':id}">
    <div>
      <strong data-bind="text:user.name"></strong>
      <small data-bind="text:event.title"></small>
      <div class="secondary-content">
        <a data-bind="click:$parent.print" href="#"
          target="_blank">
          <i class="fa fa-print teal-text tooltipped"
            alt="{{_t('Print Ticket')}}"
            data-tooltip="{{_t('Print Ticket')}}"
            data-position="bottom"></i>
        </a>
        <a href="#" data-bind="click:$parent.markPrinted">
          <i class="fa fa-check teal-text tooltipped"
            alt="{{_t('Mark Done')}}"
            data-tooltip="{{_t('Mark Done')}}"
            data-position="bottom"></i>
        </a>
      </div>
    </div>
  </li>
<!-- /ko -->
```

This is an example of a Javascript implementation of the ViewModel that populates that Knockout layout.

```
$(document).ready(function() {
  if($('.staff-home-page').length) {
    function TicketViewModel() {
      var self = this;

      self.tickets = ko.observableArray([]);
    }
  }
});
```

```

self.markPrinted = function() {
    $.getJSON("staff/markPrinted/" + this.id,
        function(data) {
            if(data.success) {
                Materialize.toast("Marked ticket as
                                printed!", 1000);
            }else{
                Materialize.toast("Unknown error.", 1000);
            }
        });
    self.tickets.remove(this);
};

self.update = function() {
    $.getJSON("staff/toPrint", function(data) {
        console.log(data);
        self.tickets(data);
    });
};

self.print = function() {
    var win = window.open("/staff/nameTag/" + this.id,
                          '_blank');
    win.focus();
};

self.update();

setInterval(self.update, 2000);
// Auto-update while this view is open.
}

ko.applyBindings(new TicketViewModel());
}
});

```

4. Requirements

1. Functional

The purpose of this project was to build a website for events.

This website should be able to advertise events and also handle their publicity. The website must also handle registration for an event, meaning confirmation of registration via an email containing the ticket for the event. Two optional extensions include the addition of the event to a calendar for the user and the use of QR codes on the tickets. Registration also includes the generation of name badges to be printed by staff at the event's registration desk. Another feature of registration is the generation of electronic information packs containing information specific to each separate event. These information packs are made available to the attendee (after registration). Optional additions make the information pack both language and interest specific to the attendee and also includes an updated attendee list for the event. Another optional addition to registration includes the automated forwarding of attendee travel details to taxi companies and the forwarding of attendee personal details to local hotels to avoid the attendee having to re-enter their details in another web form.

The website must be able to handle the uploading of results and recordings of events and, subsequently, the access to said results and recordings.

The website must allow website users to alert event organisers of possible clashes with other events and also of any clashes of interests.

Some optional functional requirements include the background analysis of an event for organisers. This includes the optimisation of overall cost for accommodation. This must take into account the expected attendance, the capacities of local hotels and their respective meeting rooms, the capacities of B&Bs in the case that the hotels are overbooked and the possibility of local hosts, if subsequently required.

Another optional functional requirement includes the avoidance of event clashes through scheduling. The avoidance must extend to include local events, related events located nationally and also related international events if the event in question is a national conference itself.

One final optional functional requirement is the use of payment facilities to pay for an event, e.g. Paypal.

2. Non-Functional

The non-functional requirements of the website can be categorized as following:

Usability

Usability, or the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, was very important in the design of our website. The Usability requirements for the website can be further sub-categorized as follows:

Learnable

It should be easy for users of all abilities to accomplish basic tasks such as creating an account and registering for an event within 2 minutes of first encountering the website.

Efficient

For all users, once they have learned how to use the website design, they should, within 5 minutes, be able to perform tasks quickly.

Memorable

If a user returns to the website after a prolonged period of absence (e.g. 1 month), they should be able to re-establish proficiency with the website quickly. This should take them at most 5 minutes.

Satisfactory

The design of the website should render the website very pleasant to use.

Efficiency

The resources consumed by the website should be relatively low. Only one low-end server (1 core, 64GB memory) should be needed with a database of less than 1GB.

Maintainability

The website should be easy to maintain, through use of a manual written especially for the maintainers. Any website faults should be located with low effort on the part of the software engineer through the use of elegant and documented code. The website's useful life should be maximised to be above 5 years.

Open Source

The website's code should be made available online in an open-source style. The license that must be used is the MIT license.

Platform

The software platform of choice is the web browser. All major browsers (both mobile and desktop) must be supported, including, but not limited to, Chrome, Firefox, Safari and Internet Explorer (version 9+).

Response Time

With the exception of requests that require the call to third party software, all requests must be answered within 1 second.

Security

The website must be secure when it comes to payment. No details of payment may be passed on to third parties, excepting the third party software used for securely handling payment. Non-payment details may only be passed on to cooperating companies with the express permission of the website user. This permission must be logged for possible future use.

3. Domain

Easy To Maintain and Learn OS

The operating system we chose had to be easy to learn and maintain for a LAMP stack in order to run our application. As such, we chose Ubuntu LTS 14.04 because it's widely supported and there was a large community around the OS that we could easily learn from and use to implement our software.

Scalability

Although not an explicit requirement of the project, we felt scalability was a mandatory requirement. As such, we chose to deploy our application on a Virtual Machine on the Microsoft Azure cloud platform which allows us change the size and power of our VM at will as well as easily deploy it in a load-balanced cluster that uses only enough resources as it needs any one time. For a company, this would be economical and fulfil the needs to server hundreds of thousands of users if needs be.

Open Source

In line with our previous requirements, the OS itself also had to be open source. Ubuntu was a perfect choice for that and has always championed OSS at its core.

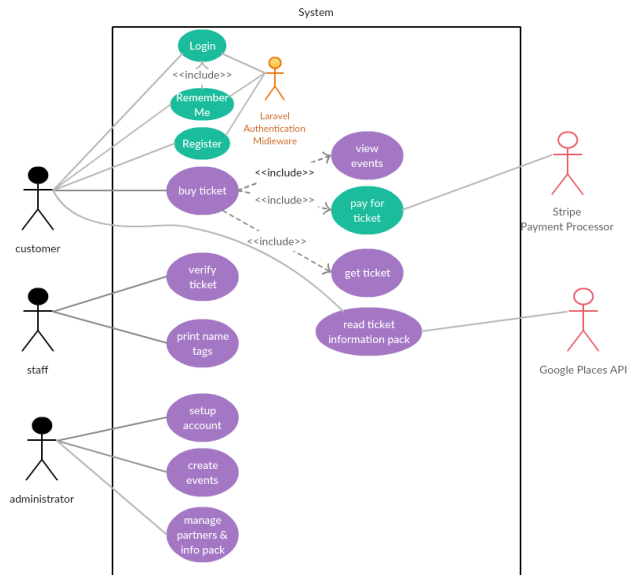
Packages

Ubuntu has a wide array of packages available for the system in order to support any kind of software we wanted to build. It also made it easy to periodically update the system via cronjobs.

4.1 Understanding Use Cases

4.1.1 How We Found Our Three Use Case Groups

We divided our target users into 3 separate groups. The UML diagram above shows the actions associated with each type.



4.1.2 Administrators

The administrators are the people responsible for the purchase, setup and maintaining of the website. They set up an account for the administrators, create events and also manage partners and information packs for events. They also approve media.

4.1.3 Event Staff

The event staff are responsible for the running of events. They scan customer tickets, verify that a ticket is genuine and they also print name tags for the event attendees.

4.1.4 Attendees

These are the people who attend events. They use the website to register an account and search for events. Using this account they can log in, edit their profiles, register for events and purchase tickets. They also can use the information packs provided by the administrators to find out more about the event they are attending.

4.2 Narration of Proposed System

4.2.1 Events

These are the core components of the system. They include, but are not limited to, concerts, conferences and gatherings. All events have a title, a tagline, a description, a start date, an end date, a start time, an end time and a location. Events can also have an image associated with them and also partners, which provide services to attendees of the events.

4.2.2 Locations

Locations are the places at which events and partners are located. Each location has a name, a capacity, a longitude and a latitude. One location can host different events, but an event can only be located at one location.

4.2.3 News

These are posts made on the website. They contain information relevant to an event or some other component of the website. These can be posted by administrators of the website.

4.2.4 Media

These are photographs or videos of an event or partner. These can also be a profile picture for a user account. Media associated with an event can be uploaded by attendees of an event after the event takes place. The media is then set up for approval by an administrator.

4.2.5 Admin

The administrators are the people responsible for the purchase, setup and maintaining of the website. They set up an account for the administrators, create events and also manage partners and information packs for events. They also approve media.

4.2.6 Staff

The event staff are responsible for the running of events. They scan customer tickets, verify that a ticket is genuine and they also print name tags for the event attendees.

4.2.7 Users

These are the people who attend events. They use the website to register an account and search for events. Using this account they can log in, edit their profiles, register for events and purchase tickets. They also can use the information packs provided by the administrators to find out more about the event they are attending.

5. Planning

5.1 Allocation of Roles

5.1.1 The Kanban Philosophy

The methodology we used is Kanban. We considered Waterfall and Agile upon beginning the project, however we felt that due to the size of our group Kanban would suit us best. We were correct as it gave us no issues. Each person in the group was free to create new tasks and post them on our Kanban board. All participants had a full view of what was currently being tackled and what needed to be tackled in the future. No one person was overloaded with tasks. With this methodology, we began with existing roles and it allowed us to make continuous, incremental changes to the system reflecting changes in direction of the design.

5.1.2 The Weekly Standup

He said it's fine to add people

5.1.3 Taiga.io & Allocation

This was the software we used to help us allocate tasks. It is an online Kanban board, giving us a centralized way of seeing all tasks, both complete and incomplete. Any member is able to create a task and any member can assign any other team member to a task. This task can be in one of many phases, namely; new, ready, in progress, ready for test, done and archived.

5.1.4 Team Leader - Kanban Sensei

Our team leader was Evan Smith. He was the one who guided us through the project. He did not have one role, but, like the rest of us, he completed many different tasks. He helped the team members who had little or no experience with Laravel to get up to speed. He also was at hand for the rest of the project to help sort out any errors that were occurring. Evan ensured that there was no hierarchy in our small group that would interfere with development of the website.

5.1.5 Laissez-Faire?

We adopted a 'laissez faire' approach, where all team member were encouraged to tackle any areas of the project without intervention. While we did have a 'team leader', he was there to guide us as a team instead of assigning tasks. He oversaw the project as a whole.

5.2 The first steps - plan, teach, practice, research

1. The first thing we did as a team was summarise the requirements. We divided them into essential requirements and optional requirements and ordered them as such, intending to tackle the former before the latter. Based on the

requirements, we constructed a 'Milestone Structure' as explained in the next section.

2. Our second task was to teach the members of our group with little or no experience in Laravel about the basics; models, views and controllers. We did this both by discussing as a team the different components and also by watching online tutorials given by the Laravel development team.
3. Next, the less experienced team members put their new knowledge to the test by creating models, views and controllers and finding their way around the basic workflow.
4. Finally, the whole team researched the project by studying and using current event software (e.g. Eventbrite). We also read up on relevant topics using Laravel's online documentation pages. In total, then above first steps took us one week.

5.3 Planning Structures

5.3.1 Milestone Structure

To help give structure to the development of our project, we divided the project into milestones. Completing each milestone resulted in one full feature being added onto the website. Each milestone was given a time it would take to be completed, allowing us to see very early on what features were viable and what ones weren't. This meant we began development on the project with a very good idea of what the resulting website would look like and how long it would take to complete it.

Milestone 1 - Minimum Viable Product

This first milestone was the completion of a minimal viable product. It encapsulated all of the 'essential requirement' as specified in the project specification document. This milestone was expected to take us 3 weeks and was completed in 3 weeks.

Milestone 2 - Payments & Stripe Integration

The second milestone was payment integration. This included the integration of Stripe to allow ticket sales on the website. This milestone was expected to take the full team 1 week and was completed by one team member in 2 days. The time left over was used to clean up the website and for testing.

Milestone 3 - Google Places & Partners V2

The third milestone was the implementation of an updated version of partners. It involved the integration of the Google Maps API and the Google Places API to allow for interactive maps and richer information about partner companies. This milestone was expected to take us 2 weeks and was completed in 2 weeks.

Milestone 4 - Integrating social media

The fourth milestone was social media integration. This involved using the Facebook API to allow the authentication of users through Facebook. It also allows users to see who of their Facebook friends are attending an event. This milestone was expected to take us 2 weeks and was completed in 2 weeks.

Milestone 5 - Optimising Events

The fifth and final milestone was optimisation. This involved the gathering of information to help organisers choose a venue for an event. We knew that we wouldn't be able to reach this milestone in time but we left it in as a possible extension.

5.4 Risks

5.3.1 Security

Laravel & Security

One risk we knew of when developing our project was the security associated with our site. Fortunately, Laravel provides CSRF (Cross-Site Request Forgery) and SQL injection protection right out of the box. Another security risk was the possibility of credit card details being stolen from customers purchasing event tickets. However, as we are using Stripe to handle all of our payments, this was not an issue for our website. This was one of the reasons why we chose Stripe as our payment software. Payment details are handled directly by Stripe and are not stored by our system at all. Some features that made Stripe the clear choice for our payment software include HTTPS, HSTS, strong encryption, and disclosure. Stripe has also been audited by a PCI-certified auditor and is certified to PCI Service Provider Level 1, which is the most stringent level of certification available.

5.3.2 Time Management

Another risk was the possibility of our team running out of time when developing this project. To tackle this issue, we used the milestone system, as mentioned above. We stuck to our predictions as closely as possible and, as a result, we finished ahead of our schedule.

5.3.3 Unification of Software

One other possible risk was the unification of our team members' software. To deal with this problem, we used Git and an EditorConfig file.

Problems with Git

We used Git for version control. This worked very well most of the time and kept all our files synced across our various machines. However, one issue was the fact that we used separate ways of using Git. Some team members used the command line, while others used Desktop clients. The command line worked very well and gave no errors, while it took a few days to learn. The desktop clients, on the other hand, were mostly very intuitive but one desktop client in particular, git-cola, gave our team lots of strife. On one occasion it reverted all of one team member's commits. This was eventually corrected but it caused a lot of undue stress. However, the benefits of using GitHub and version control in general far outweighed the disadvantages.

How .editorconfig helps

To define and ensure a standard style across our range of editors and IDEs, an EditorConfig file was used. This file was needed as some of our members used Sublime Text as their editor, while others used PHPStorm. The file itself was very easy to read and worked well with GitHub. It benefited the development process a huge amount.

6. Design

6.1 GUI

(see also [3.2.4 Design Framework - Materialize](#))

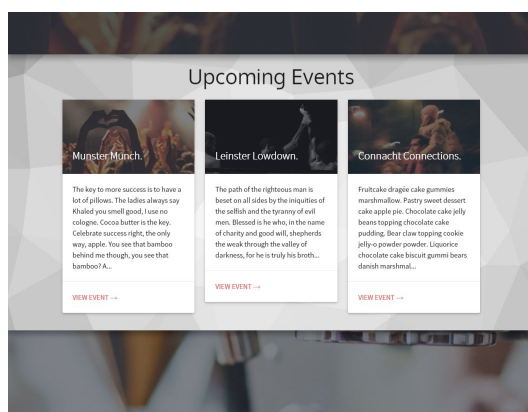
Set within the context of the browser, the goal for the GUI design of the Eve project was to create an intuitive, responsive design that follows the expected flow of modern web applications, but would feel like a native application on mobile devices. In examining modern trends in design, we found Google's Material Design rules as a publicly supported design movement with a large number of available resources, guides, and sample code.

6.1.1 Google Material Design

Material Design is a GUI design style developed by Google as a new standard for cross-platform design, which was initially developed for their internal use but has also been made available for anyone to co-opt and expand on.

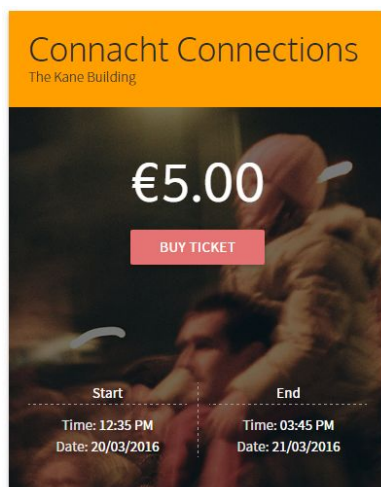


The goals of Material Design are to provide a cross-platform “design language” that follows up on the classical principles of good design, but works universally across different devices and input styles.



It's founded upon the metaphor of layered material, the use of light and shadow to divide items and indicate what can be interacted with. It relies on the elements of print design: typography, grids, use of bold colour and imagery, but combines these with the use of motion to imply user action.

In Eve we use these principles. Buttons are brightly coloured and contrast against the page; they cast shadows to give them a sense of depth and follow the “material” metaphor and when pressed, respond with a wave pattern instantaneously.



The use of shadows and parallax scrolling throughout the app, particularly on the home-page and event pages, give the sense of motion to all elements of the application, allowing individual elements to pop out from their surroundings, and providing clear pathways to user interaction. In this regard, we have built a system that allows to intuitively understand our application the first time they see it.

The use of colour and imagery is something that we're also proud of. By using imagery from events

throughout the application, we provide a bold, immersive experience. However, we use strong, bright colours from the Material palette in order to provide contrast and to highlight interactive elements.

We use the Roboto font in different weights to deliver a consistent, readable body of text across various different contexts. We give a certain level of transparency to allow it to blend in more evenly with its backgrounds. Notably, we avoid the use of pure white or pure black throughout the app, instead using off-white shades, which are easier to read on bright screens.



Our use of material design concepts also extends to providing a responsive environment. We re-order grids, switch our navigation styles, and adjust text styles and sizes to best suit the viewing environment.

Name	Darragh King
E-Mail Address	darragh@king.com
Username	Dark420
Password	
Confirm Password	

We also use a form styling that fits the Material aesthetic; responding with motion and colour to each interaction from users.

6.1.2 Materialize - a CSS Framework implementation of Material

Of course, implementing Material design from scratch would be a difficult process. We decided to use a CSS framework to simplify our design process, but we didn't want to have to rewrite this to suit our goals of using material design.

It provides a smart grid system (as we've discussed in section 3.2.4), a simple system of setting colours in the Material palette using CSS classes (using simply the colour name and classes "lighten-x" or "darken-x" with x as any number from 1-4), clever styling for tables and forms, a series of helper classes for media, layout,

parallax scrolling, typography, and shadow, and a series of useful components that we can implement throughout the application.

Here, we take a look at the code necessary to generate the below form taking into account the row/column structure and the use of alternate styling on material-style input fields:



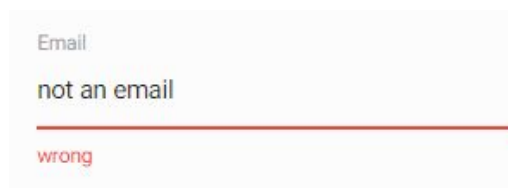
First Name

Placeholder

Last Name

Password

```
<div class="row">
  <form class="col s12">
    <div class="row">
      <div class="input-field col s6">
        <input placeholder="Placeholder" id="first_name" type="text" class="validate">
        <label for="first_name">First Name</label>
      </div>
      <div class="input-field col s6">
        <input id="last_name" type="text" class="validate">
        <label for="last_name">Last Name</label>
      </div>
    </div>
    <div class="row">
      <div class="input-field col s12">
        <input id="password" type="password" class="validate">
        <label for="password">Password</label>
      </div>
    </div>
  </form>
</div>
```



Email

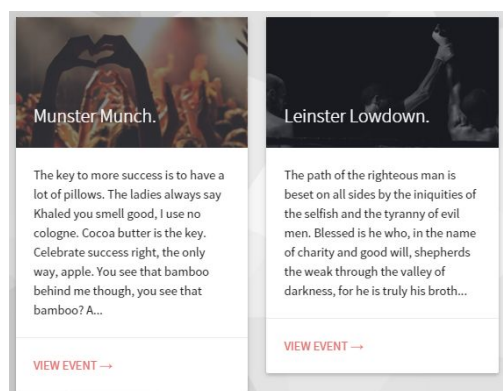
not an email

wrong

As you can see, “input-field” columns have inputs with responsive, animated labels, and the use of a “validate” class allows for use of colour to imply whether or not an input is valid. This also will prevent a form from being submitted.



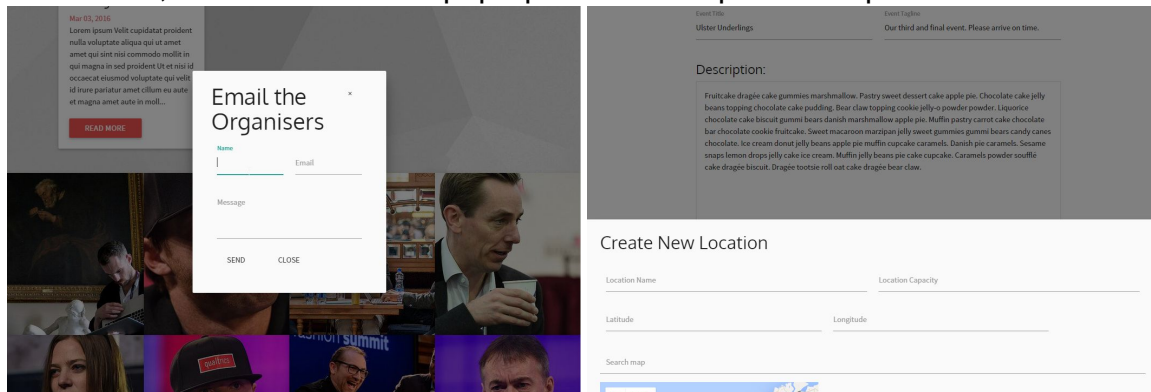
Further to this, Materialize also provides an icon set that is taken from Google’s own operating system icon set, giving us access to recognisable, useful symbols to use throughout the application.



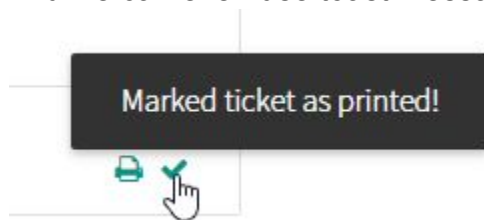
We make common and frequent use of *Cards*, a Materialize component that exists in Google’s initial description of Material Design; a “sheet of material” that provides an entry point to a larger block of information. We use them on the

homepage to refer to events; on the event page to refer to tickets, partners, and attendees. It gives an structure and a clarity throughout the application onto how to access related items.

We also get a huge number of elements with embedded Javascript for interaction. For instance, we can use modal pop-up windows to provide important functions:



And we can even use toast messages to signify successful actions:



6.1.3 LESS Variables and their advantages

We can mark up our preferred font stacks, colours, and other reusable items in LESS variables so that we can reuse them throughout the application. The goal here is that we have consistent rules for styling throughout the app.

6.2 Database Structure & ER Diagram

Laravel's Migration system (see section 3.3.2. "Migrations & Seeders") means that the database tables that exist within the system map almost one-to-one with our Models in Laravel.

The Models that we have defined are:

- Event (*Has one location, has many partners, has many tickets*)
- Location (*Can belong to Event, can belong to Partner*)
- Media (*Belongs to User, belongs to Event*)
- News
- Partner (*Belongs to many Events, has one Location*)
- Setting
- Ticket (*Belongs to one User, belongs to one Event*)
- User (*Can have many media files, can have many tickets*)

We also will need these for the Translation package:

- Locales
- Translations

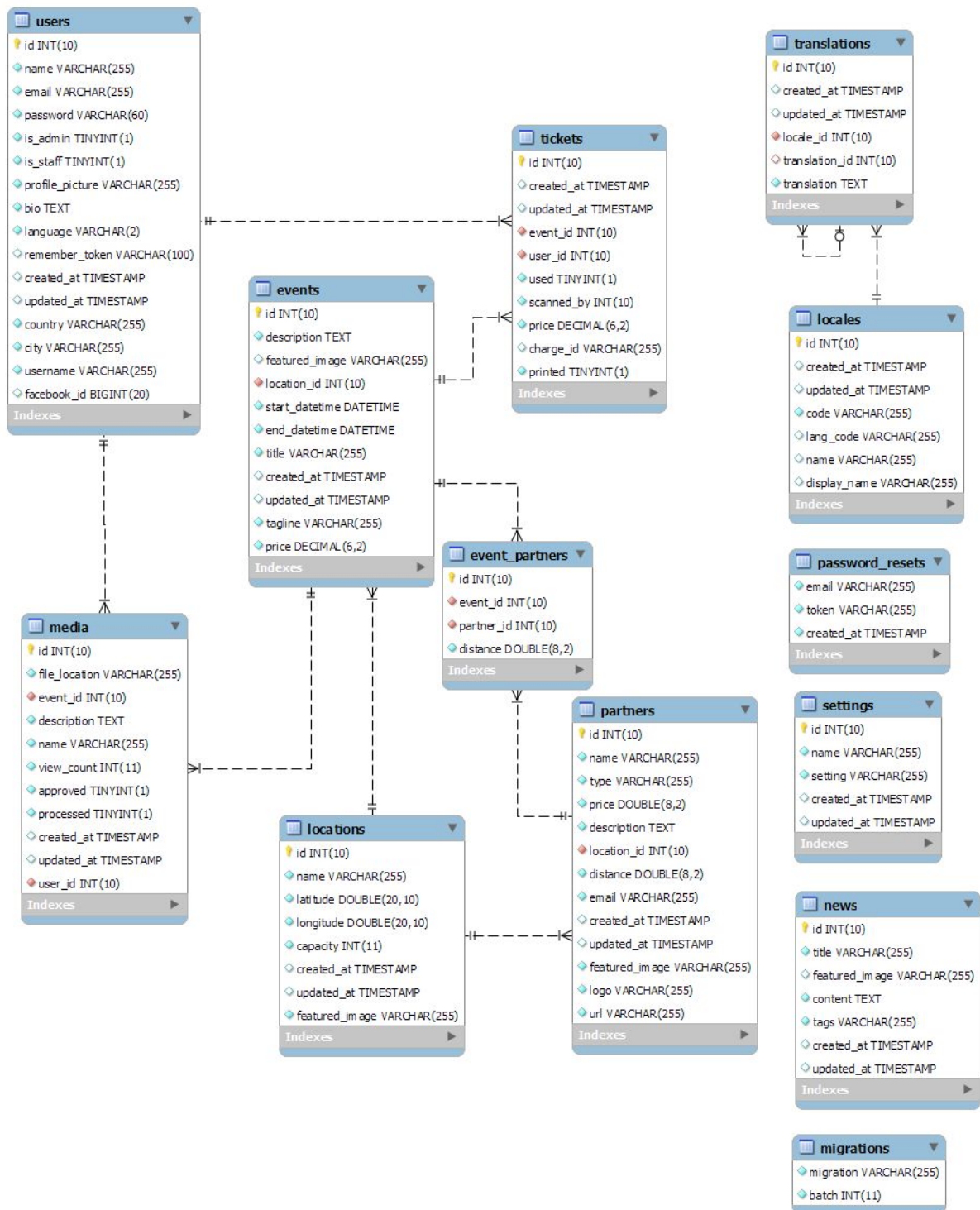
We'll need these for Laravel:

- Migrations *(to have a permanent record of the current database state)*
- Password Resets

And finally, as Partners-Events is a many-to-many relationship, we need a pivot table:

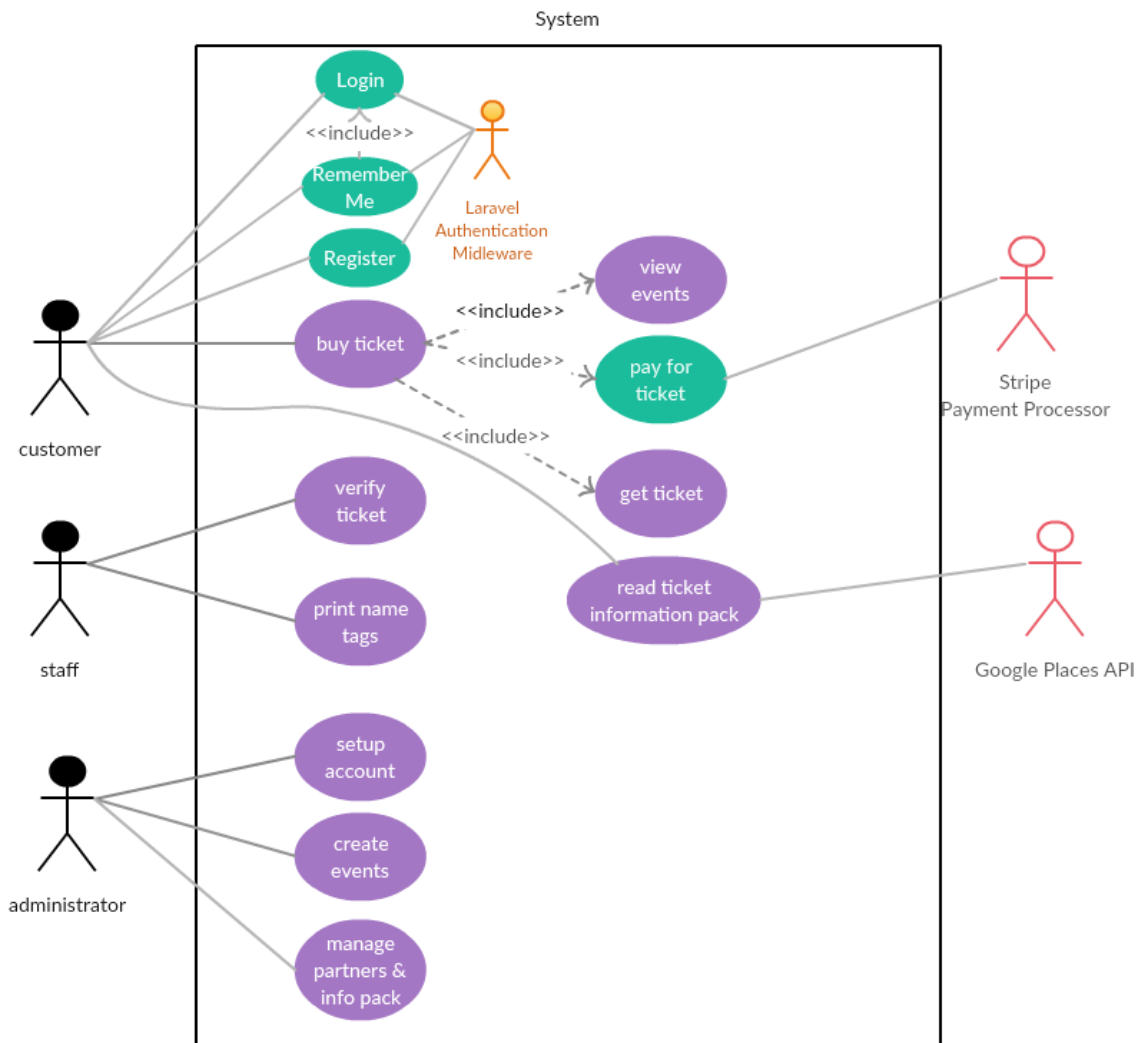
- Partners-Events *(Maps partners to their events)*

From this, we can determine their attributes, and develop an entity-relationship diagram, as shown below in this diagram which we built in MySQL Workbench. This diagram also shows the attributes held by each of these models, in the form of the database columns within the diagram.



We believe this is a sufficiently normalised and that redundancy is eliminated. The database layout we've chosen is easily editable and expandable through use of Laravel migrations, and yet it supports our feature set entirely. We consider this clean, simple database design to be one of the positive aspects of building the app in Laravel.

6.3 UML Diagrams



This UML/Use Case Diagram identifies three distinct classes of users, the activities that they need to perform, and the external factors that they have to deal with. It clearly identifies the objects (Event, Ticket, Location, Partner) that form the interactions the user performs as well.

6.4 Form Fields and Validation

6.4.1 Laravel Validator Syntax

Laravel provides a simple Validator class, allowing us to validate data before we attempt a database insert of the data. This allows us to redirect back to the submission form with errors. While Laravel will sanitise any inputs to the database, and raise an error if there's a type mismatch, it's important to use the Validator to ensure that the user is made aware of any issues with their inputs.

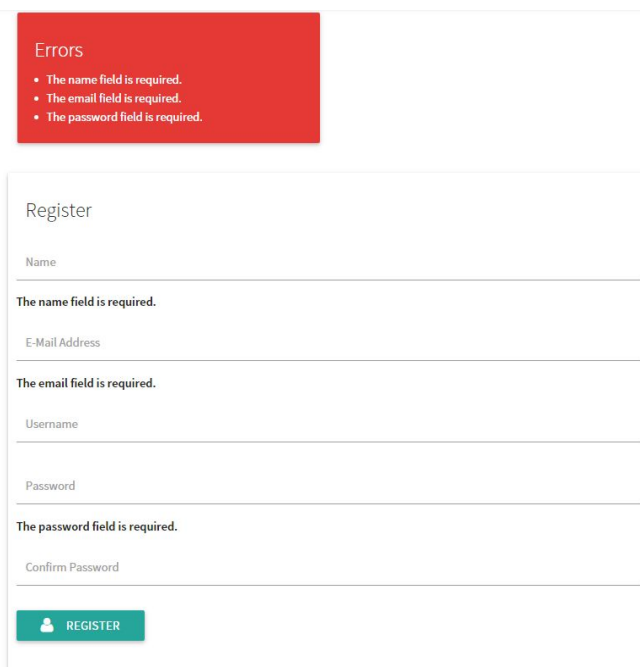
Here is an example of how the Laravel Validator works in code:

```
// Limit the data we're working with to just these fields to
// prevent injection
$data = $request->only( [
    'name',
    'latitude',
    'longitude',
    'capacity',
    'featured_image'
]);

// Validate all input
$validator = Validator::make( $data, [
    'name' => 'required',
    'latitude' => 'required|numeric|between:-85.05,85.05',
    'longitude' => 'required|numeric|between:-180,180',
    'capacity' => 'required|numeric',
    'featured_image' => 'image|sometimes'
]);

if( $validator->fails( ) ){
    // If validation fails, redirect back to
    // registration form with errors
    return Redirect::back( )
        ->withErrors( $validator )
        ->withInput( );
}
```

`return Redirect::back() ->withErrors($errors)` is used in Laravel to return the user to the submission form they used with the full details of any errors that they have encountered.



The screenshot shows a web application interface. At the top left, there is a red error box titled "Errors" containing three bullet points: "The name field is required.", "The email field is required.", and "The password field is required." Below this, there is a "Register" form. The form has five input fields: "Name", "E-Mail Address", "Username", "Password", and "Confirm Password". Each field has a corresponding error message below it: "The name field is required.", "The email field is required.", and "The password field is required." At the bottom of the form, there is a green button with a user icon and the text "REGISTER".

This gives us a powerful way to implement validation of inputs at all points in the application. The option to display errors is included in the general application layout and as such can appear on any page.

The validators aren't the only checks that we run. For instance, we have application middleware that filters requests, and will redirect back with errors if a user attempts to access content that goes beyond their permissions, or use a user account that has not yet been assigned a username.

6.4.2 Our Fields and Validators:

In our form fields, there is a direct match between database column names and field names.

Users (registration)

```
'name' => 'required|max:255',
'email' => 'required|email|max:255|unique:users',
'password' => 'required|confirmed|min:6',
```

Users (profile):

```
'bio' => 'required',
'password' => 'sometimes|confirmed|min:6', // to update
'profile_picture' => 'sometimes|image',
'language' => 'required|exists:locales,code',
'city' => 'required|max:255',
'country' => 'required|max:255'
```

Company:

```
'company_name' => 'required|max:255',
'description' => 'required',
'company_logo' => 'sometimes|image|max:10240' // 10mb
```

Events:

```
'title' => 'required|max:255',
'tagline' => 'required|max:255',
'description' => 'required',
'start_date' => 'required|date',
'end_date' => 'required|date',
'featured_image' => 'image|sometimes',
'start_time' => 'required',
'end_time' => 'required',
'location_id' => 'required|exists:locations,id',
'price' => 'required|numeric|min:0'
```

Locations:

```
'name' => 'required|max:255',
'latitude' => 'required|numeric|between:-85.05,85.05',
'longitude' => 'required|numeric|between:-180,180',
'capacity' => 'required|numeric|min:0',
'featured_image' => 'image|sometimes'
```

News:

```
'mce_0' => 'required', // Title
'mce_1' => 'required', // Content
'featured_image' => 'image|sometimes'
```

The “mce_” values here are translated to the actual column names later as they’re used by the TinyMCE Javascript library to allow for What-You-See-Is-What-You-Get editing in the form.

Partners:

```
'name' => 'required|max:255',  
'picture' => 'required|image',  
'type' => 'required|max:255',  
'price' => 'required|numeric|min:0',  
'description' => 'required|max:255',  
'location_id' => 'required|exists:locations,id',  
'email' => 'required|email|max:255',  
'logo' => 'required|image',  
'url' => 'required|url'
```

7. Software Deployment

7.1 Target Environment

Ubuntu LTS 14.04

- PHP 5.6
 - php5-curl
 - php5-mysql
 - Composer
 - Laravel 5.2
- MySQL (5.5.46)
- NodeJS (0.10.25)
 - NPM (1.3.10)
 - Gulp (3.9.0)
 - Laravel-Elixir

7.2 Deployment of Components

1. Provisioned Microsoft Azure VM
 - We provisioned a virtual machine on the Microsoft Azure platform to run our software and maintain a demonstration environment
2. Git
 - Git was an easy way to both version control our code and deploy it when we were ready. To deploy the software, we simply cloned the git repository from Github to get the files onto our server safely
3. Composer
 - Composer managed all of the PHP/Laravel package dependencies for our project
4. Gulp/NPM
 - NPM was a dependency manager for our NodeJS requirements and Gulp automatically processed LESS into CSS as well as minified all of our javascript into a single file

5. Vagrant

- Vagrant was a VM manager for VirtualBox that allowed us to easily duplicate our production environment locally for testing and development throughout the project

7.3 Client/Server Configuration

Any client can interact with the web application via any modern browser or mobile browser. There exists no setup beyond the HTTP requests performed by the browser.

7.4 Installation (Production)

1) Clone the repository into /var/www

```
mkdir -p /var/www
cd /var/www
git clone https://github.com/Evan Smith/Eve.git .
```

2) Then run our installation script. This will download everything necessary and install it on the system for you.

```
sudo ./install.sh
```

You will be prompted to choose a password for MySQL, remember this for the next step.

3) Create the database for the application. You'll be prompted to enter your password

```
echo "create database projecteve" | mysql -u username -p
```

4) In the /var/www directory, edit the .env file following the below example

```
APP_ENV=production
APP_DEBUG=false
APP_KEY=random_32_characters

DB_HOST=localhost
DB_DATABASE=projecteve
DB_USERNAME=root
DB_PASSWORD=PUT_YOUR_MYSQL_PASSWORD_HERE

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

# Fill in your mail server details below. SendGrid.com is
used in the application demo
MAIL_DRIVER=smtp
MAIL_HOST=localhost
MAIL_PORT=1025
```

```
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
```

```
# All of the Maps APIs must be activated for the
following key
```

```
GOOGLE_API_KEY=YOUR_GOOGLE_API_KEY
```

```
STRIPE_SECRET=YOUR_STRIPE_SECRET_KEY
STRIPE_KEY=YOUR_STRIPE_KEY
```

```
FACEBOOK_APP_ID=YOUR_FB_APP_ID
FACEBOOK_APP_SECRET=YOUR_FB_APP_SECRET_KEY
```

5) Navigate to the web address of your server and begin the front-end installation process in which you can rebrand the application and fill out some basic details about your company.

7.5 Updating

1) Navigate to /var/www, pull from the repository and run a composer update

```
cd /var/www
git pull
composer update
php artisan migrate
gulp
```

7.6 Uninstallation

To uninstall the application, simply delete the files and remove the database.

```
rm -r /var/www
echo "drop database projecteve" | mysql -u username -p
```

8. Testing

8.1 Our Testing Suite

8.1.1 Staging server - Test by use

- At the end of each milestone, we pulled a version of our software to a staging server in order to test it in “production”
- At different times throughout the project, we dedicated specific time to testing the site on the staging server and logging bugs in our issue tracker on github

- In the final week before presentations, we dedicated the first half of the week for all of us to try and explicitly break everything on the site - as a user, as an admin and as a passer-by
 - We then alternated days where some people would fix bugs and the others would lodge/test others
 - By the end of the week, we had eliminated all of our reported bugs and were only left with optional enhancements

8.2 Test Data

For every form input on our site, we tried a wide variety of types of data

1. **Valid values:** correct values that output the desired result
2. **Invalid values:** where the value is obviously incorrect, does not match the required format or is the wrong type of data
3. **Edge cases:** data which would test the boundaries of what our forms could handle (EG: Our maps forms can only handle longitude/latitude to a precision of 10 degrees so we tried putting in values of 11 decimal places or more as well as whole numbers to see what would happen)
4. **Stupid cases:** We attempted to tackle the application from the point of view of someone who's never seen it before. As such, we tried to see how any of our instructions or prompts could be misconstrued. Often, we reworded things into simpler english once the form was tested - these were simply aesthetic and usability tests

8.3 Test Results

Most of our testing resulted in bug reports and issues. When an issue was resolved, the date/timeframe was logged and automatically appended to the issue. Please refer to Appendix B for a full list of all of our bug reports and when they were solved.

8.4 Log of bug reports

Please refer to Appendix B

9. The User Manual

9.1 Server Administrator Manual

9.1.1 Deployment

- Set up Azure LAMP stack
- Git clone to /var/www/html
 - `mkdir -p /var/www`
 - `cd /var/www`
 - `git clone https://github.com/Evan_Smith/Eve.git`
- Generate a .env file
 - In the /var/www directory, edit the .env file following the below example


```

APP_ENV=production
APP_DEBUG=false
APP_KEY=random_32_characters

DB_HOST=localhost
DB_DATABASE=projecteve
DB_USERNAME=root
DB_PASSWORD=PUT_YOUR_MYSQL_PASSWORD_HERE

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

# Fill in your mail server details
below. SendGrid.com is used in the application
demo
MAIL_DRIVER=smtp
MAIL_HOST=localhost
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

# All of the Maps APIs must be
activated for the following key
GOOGLE_API_KEY=YOUR_GOOGLE_API_KEY

STRIPE_SECRET=YOUR_STRIPE_SECRET_KEY
STRIPE_KEY=YOUR_STRIPE_KEY

FACEBOOK_APP_ID=YOUR_FB_APP_ID
FACEBOOK_APP_SECRET=YOUR_FB_APP_SECRET_KEY

```

- Navigate to server's address in web browser
- Note on updates through update script
 - cd var/www
 - ./update.sh
- Logwatch

9.1.2 Installation

- Providing your company details
 - Fill in your company details as indicated on the form.
- Setting yourself up as a user
 - See User Manual
- Creating your very first event
 - See Site Administrator manual
- Providing company details
 - Provide all company details requested by form.
- Going live

- Pull to your live server
 - Don't forget to change your site from development to production prior to going live.

9.1.3 Support

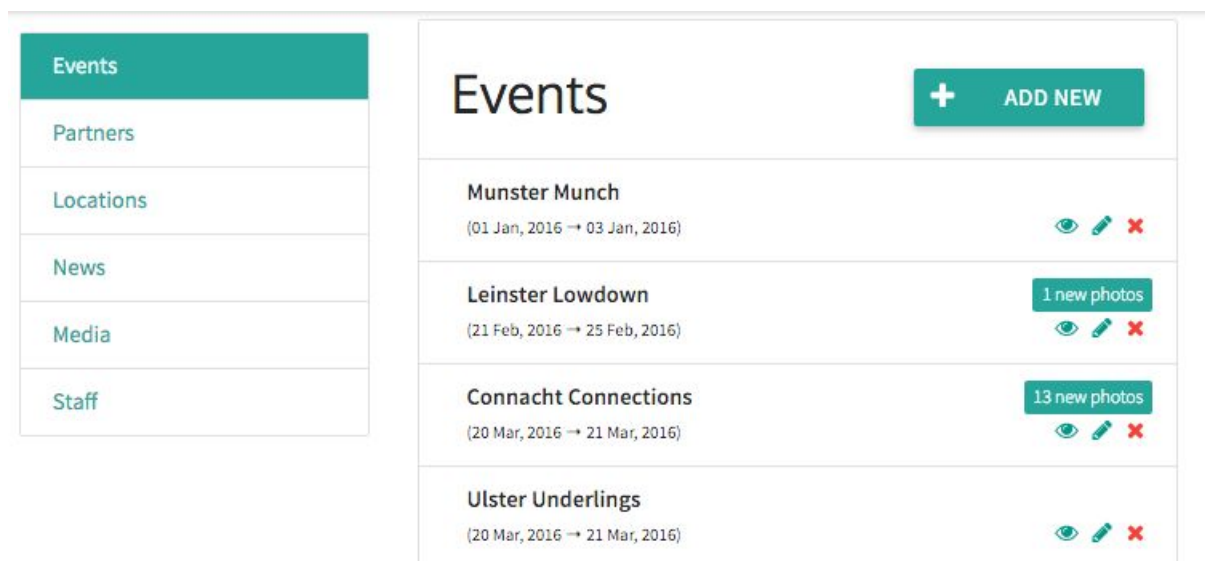
- Works on all *modern browsers* (IE8+)
- Cookies & javascript needed

9.1.4 Stripe

- Navigate to the Stripe website and create an account for your company. Then use that account on your site.

9.2 Site Administrator Manual

Site Administrators have access to the admin panel from which they can control a large amount of the site.



Any errors made by an admin in this section will result in an auto-scroll to the top of the page where any and all errors will be highlighted in red and explained clearly.

Note: Begin sections 9.2.1 - 9.2.4 by navigating to the admin panel. To do so click the “admin” button at the top of the page.

9.2.1 Events

- You will see a small nav bar on the left of the screen.
- If you are not already at the top of the page, you can click Events in this nav panel to view the events section of the admin panel.

- Here you will see an “ADD NEW” button and a “VIEW ALL EVENTS” button. Click the “ADD NEW” button to create your first event.
- Fill in all the relevant information here such as the event title, tagline, description, etc.
- If you are choosing a location that is not yet defined, enter in all the relevant information on the popup modal for a new location and search for your location on the embedded map.
- Once you have finished filling in the form, click create event at the bottom of the page.
- You will be redirected to the events page where you will see your newly created event.
- Click the admin panel again and navigate to the event section. You should see your event here.
 - Accompanying your event you should see three icons.
 - A red x which can be clicked to delete an event.
 - A pencil to edit the event.
 - An eye symbol which can be clicked to view the event.
 - Begin by clicking the eye symbol on any event.
 - This bring you to the events page. You can view any information relevant to the event here, including a list of attendees, info about the event etc.
 - You should see an edit event button at the top of the page. This is equivalent to the pencil on the admin panel.
 - Click it.
 - After clicking the edit event button you will see a form exactly the same as the one for creating events, however it will contain the pre-filled information of the event. Here you can change any information of your event.
 - Navigate back to the Admin panel.
 - The final (and optional) part of this section is clicking the red x. This will permanently delete your event and any pictures associated with it.

9.2.2 Locations

- If you are not already at the locations section of the page you can click Locations in the nav panel to view the locations section of the admin panel.
- You will be presented with a list of existing locations.
- Click the “ADD NEW” button to create a new location.
- This will bring you to a page containing a form that contains an embedded map.
- Enter a location, title and capacity. Unless you know the exact coordinates of your location, search for it using the search map field.
- Once you have entered in all of the required information and given your location a picture, click create location.
- Once created, you will be redirected to the edit location page if you wish to make any last minute changes.

- If not, congratulations! You have created your first location. You can return to the admin panel if you ever wish to remove it.
- You can click “ALL” at the end of this page to finish this section. This will bring you to a page of all of the current locations.
- Here you can click on the pencil icon next to any location and edit it. You can also return to the admin panel to edit a location.

9.2.3 Partners

- If you are not already at the Partners section of the page you can click Partners in the nav panel to view the Partners section of the admin panel.
- Here you will see an “ADD NEW” button and a “VIEW ALL PARTNERS” button. Click the “ADD NEW” button to create your first partner.
- Fill in all the relevant information here such as the partner name, average price, description, etc.
- If you are choosing a location that is not yet defined, enter in all the relevant information on the popup modal for a new location and search for your location on the embedded map.
- Once you have finished filling in the form, click “ADD PARTNER” at the bottom of the page.
- You will be redirected to the partners page where you will see your newly created partner.
- Click the admin panel again and navigate to the partner section. You should see your new partner here.
 - Accompanying your partner you should see three icons.
 - A red x which can be clicked to delete an partner.
 - A pencil to edit the partner.
 - An eye symbol which can be clicked to view the partner.
 - Begin by clicking the eye symbol on any partner.
 - This bring you to the partner’s page. You can view any information relevant to the partner here, including a list of attendees, info about the partner, etc.
 - You should see an edit partner button at the top of the page. This is equivalent to the pencil on the admin panel.
 - Click it.
 - After clicking the edit partner button you will see a form exactly the same as the one for creating events. However, it will contain the pre-filled information of the partner. Here you can change any information of your partner.
 - Navigate back to the Admin panel.
 - The final (and optional) part of this section is clicking the red x. This will permanently delete your partner.

9.2.4 Media

- If you are not already at the media section of the page you can click Media in the nav panel to view the partners section of the admin panel.
- The Media section is simply for approving various media and filtering out unwanted or prohibited media.
- To handle this section simply click either a tick or an x on any of the submitted media.
 - Clicking the tick approves the media
 - Clicking the x deletes the media

9.2.5 Users

This handles managing users' permissions and users' profiles.

1. Managing users' permissions:

- This section is different to the preceding sections in how it is managed. You are unable to manage this aspect of the site via the admin panel. The purpose of this part of the manual is to demonstrate how to promote/demote users to staff/administrators and alter their permissions for the site.
- In order to promote a user to a staff member or admin, simply search for their profile using the inbuilt search function.
- Once you arrive at their profile you will see three buttons.
- The two buttons from the right are what we are concerned with in this part of the manual for user management.
- Depending on the user's permissions you will see both demote staff/promote staff & demote admin/make admin.
- To change this user's status simply click one of these two buttons.

2. Managing users' profiles:

- To manage users' profiles, search for the user you wish to deal with using the inbuilt search function.
- Once you find their public profile you will observe an edit profile button if you have admin privileges.
- To manage this users' profile and make changes to it click this button and fill in the edit profile form as instructed.

9.2.6 Tickets

As an admin you are permitted to view the staff panel as well as the admin panel.

Here you can view all scanned tickets and perform any of the actions that a staff member potentially could.

9.3 Event Staff Manual

The Scanned Tickets System

- Ask the attendee to present you with their ticket.
- Using your QR scanner scan their QR code
- Observe the staff panel to ensure that the scanned ticket pops up as approved.
- Print off the attendee's badge for the event.
- Done

Manually approving tickets

- If a user forgets their ticket or for some reason cannot present their QR code, you can navigate to the event page and view the list of attendees to confirm their identity before printing them a name badge.

9.4 User Manual

- When you first arrive at Eve you will be presented with our home page.
- On this page you will view a number of advertised events and some news items among other information.
- The first step for you as a user to begin attending events and interacting with our site is to register for an account.
- Any errors encountered are self explanatory and easy to deal with.
- Recommended browsers include any modern up to date browser.

1. Registration

1. On the top right of the home page you will see a signup button. Click it.
2. You will then be presented with a signup form. Fill in your correct information.
3. Once you have created an account you will be redirected to an edit user page. Here you can personalise your account with things such as a bio, profile picture, city and language.
4. Once you have entered in your desired information click the submit button at the bottom of the page to complete the registration process.
5. Congratulations! You now have a user account for Eve.

2. View events you are attending

1. After finishing creation of your account you will be brought back to the home page. To the top right of the screen you will see a calendar icon. Click it.

2. You are now viewing the events you are currently attending. It is, of course, empty as you are not yet attending any events (if you are following the manual's steps).
 3. To the top left of the screen below the navigation bar you will see a past events button. Click it.
 4. You are now viewing a page which shows you events that you have attended in the past.
 5. You are now finished this section. To proceed to the next section of the user manual look to the top right of the screen. You will see an icon of a head and shoulders. Click it.
3. View your user profile
1. You are now on your public profile. This profile can be viewed by any other individual on our website. At the bottom of the page you will see an empty section with a message telling you that you are currently not attending any events. We will discuss this later.
 2. We will now attempt to edit our user profile. To do this scroll to the top of your user profile, which you should still be on, and click the "Edit/Edit Profile" button.
 3. You will be brought to the same page you saw at setup. Here you can change any of your own details whenever you want. Once finished, navigate to the bottom of the page and click submit.
 4. Once you have clicked submit you will be brought back to your user profile. Navigate back down to the end of the page and click the button "see events" and continue to the next section of the manual.
4. Register for an Event
1. You are now viewing all of the currently planned events. Choose an event that you wish to attend and click the "view event" button.
 2. You are now viewing the page for that event. At the top right of this page you will see a button that says "upload photos". To upload photos of the event or anything relevant to the event please click this button and fill in the relevant information. Any photos submitted will be submitted for review to a site admin.
 3. Once you have either finished uploading photos or decided you don't want to upload photos, continue down towards the end of the page. You should observe a panel that gives you the option to buy a ticket. Click "purchase ticket" and fill in all of your relevant credit card details. If you have any issues with this step please check your details to ensure they are correct. If issues persist, navigate to the bottom of any page and click "contact us" to get assistance from a site admin.
 4. Once you have successfully purchased a ticket you will be given an option to "view information pack" and also "view ticket". Click either of these buttons.
 5. Both pages will contain a QR code of your ticket. The ticket will contain minimal information relevant to your ticket. The information pack will contain a map of the event location and partners associated with the event as well as some other useful information.
 6. Click the apple icon on the top left of the page to return to the home page and continue on to the next and final section.

5. Find your friends.

1. After any event, you may have met some individuals you want to stay in contact with. You can find these individuals on our site. There are two methods of achieving this.
2. The first of these options is to return to your past events page, click on the event and you will be able to view a list of attendees of the event.
3. The second option is if you are not using a tablet or smartphone you can click on the search bar at the top left of the page and search for them by name.

You now know how to use our site as a user. If you have any difficulties whilst using our product please navigate to the bottom of any page on our site and click “contact us” to send us an email.

10. Evaluation

10.1 Product

Overall, we're happy with the application. We're happy with its design, how easy it is for people to set-up and use. We like to think that it can do everything we set out for it to do.

If we were to do this again, we wouldn't change much but we'd have the benefit of experience and all of us would be hitting the ground running on this.

If given more time, we would like to allow for hotel booking through the application and allow for event organisers to select a venue for an event straight from the application. This would make it the total "all-in-one" application it could

10.2 Development activity

Development was pretty slow to start. Most of us had little to no experience with MVC frameworks. As we all got up to speed, we were able to quickly and effectively work on producing a quality application.

We were easily most efficient during group coding sessions (typically held on a Thursday afternoon) where we could bounce ideas off one-another and get another set of eyes on bugs that cropped up.

Throughout the development process we checked in with each-other at a weekly stand-up meeting. Here, we gave a summary of what we worked on during the week and what we'd like to work on in the coming week. We were also able to identify what sections of the development process weren't being given enough attention. Typically, the person with the least amount of work would volunteer to take up those tasks.

Communication was vital to our progress throughout developing the software. Whether it be for solving problems with our code or getting feedback on changes, because we were in near-constant contact, this process was effectively streamlined to the greatest degree.

10.3 Software used

10.3.1 Communication

For communication, we used Facebook Messenger. This was simple to use and we all had an account already so there was little set-up required, just to create a new room for all of us.

However, this had its problems. Because Messenger is so linked with the social networking platform, it might've been easy to get distracted. It was also impractical to share code snippets over Messenger.

In retrospect, we should have used something like Slack so as to separate the social and work aspect of communication. Slack also allows for topic-specific channels. This would mean that technical problems could be dealt with in one room, while debugging or styling decisions could be done in another room. They wouldn't interfere and clog up each-other's room but would still be visible to everyone. Most usefully, Slack can be integrated with GitHub. This means we could reference issues, files and commits from within Slack and not have to use external links to GitHub.

10.3.2 Code Collaboration

GitHub was our method of code collaboration and issue tracking. Git's use is ubiquitous across the software development industry for these purposes. Therefore, we found it vital that we all had some experience using it.

GitHub's issue tracking made it easy for us to keep an up to date list of identified issues, bugs and missing features. It also allowed us to have a centralized point where all information on a particular issue could be found. Typically, the person who identified the issue would self-assign the issue and eventually close it out. However if there was an issue that someone found in someone else's code, they could tag that person to notify them, alerting the person to the issue immediately.

Issues could be closed directly from commits. In a way, this means that the commits were self-documenting in that you could read the commit message and see which issue it closed out, thereby understanding the purpose of the commit.

Software we used to push and pull with GitHub included git, GitHub Desktop and Git-cola. Git is the traditional command-line utility and has very little overhead. It's simple to use for anyone with some experience in working with the command line. The one major way it falls short of being the most ideal tool is the way it handles merges. Because it's run entirely from the command-line, it's difficult for it to handle merges in an intuitive manner, this is where GUIs have an advantage. Git would fail on pushes where there is a merge conflict and require you to manually merge the files, showing the differences in between equals symbols. This often made merging tedious as you'd have to find the files that had conflicts and merge them manually. Many of our commit messages were simply "Merge Conflict" and often made it difficult to see what had been changed.

GitHub Desktop is GitHub's own solution to working with repositories on GitHub. Overall, this is fine for tracking changes, pulling and pushing but doesn't add much other than a UI to Git. Therefore it suffers from a lot of the same problems as Git, most evidently dealing with merges.

10.3.3 Task Management

As for tracking our progress and who was assigned to what, we used Taiga.io's kanban board. Here we could see what was still to be done, what tasks hadn't been assigned and pick up tasks that we'd like to work on.

Kanban worked great for us as a management-development style and without a doubt, Taiga worked without a hitch for us. Taiga's integration with GitHub was a plus but at times seemed excessive as when issues were closed in GitHub, Taiga required us to manually close issues there too.

Overall though, Taiga was the ideal kanban solution for us.

10.4 What we'd do differently

10.4.1 What we'd prefer

It would have been a great advantage to have the planning process done in advance. Being given the project specification sooner would have allowed us to draw up our milestones and timeline before the allotted development period. Additionally, knowing we could pick our teams beforehand would've helped us in being able to brief our entire team on the framework we were planning to use and to learn how to work with it if we didn't have experience with it.

Finding space to work in was difficult. All the teams were practically fighting for space in the labs. And with other classes using the labs, it wasn't easy to collaborate at times. Being able to book rooms or space to work in would certainly remedy this.

Finally, something we found lacking was structure and clarity in the project specification and feedback we received for our presentations during the project. This vagueness in the specification for the software and presentations was most evident in the range of solutions that were given by other teams and what they presented in the final presentation.

10.4.2 What we'd change

It's clear from looking at how much was committed during them that group coding sessions were our most effective times and if we were to do this again we would definitely schedule more of them.

Also, getting up to speed with Laravel took some of us up to two weeks, which means that two weeks of development were lost for some of us. Anyone who took longer to grasp Laravel also had to catch up with understanding the code-base which saw significant changes in the first few weeks. To remedy this we'd spend a week or so getting everyone on the same page, making sure that we all knew how to work with the Laravel framework

10.5 Conclusions

In conclusion this project was incredibly valuable from the perspective of what we learned from it: collaboration, frameworks and standards. As computer science students we enjoy challenges and the process that goes into solving them. There were times of frustration, like when we were setting up Vagrant or that one time we had three merge conflicts in a row but it's all part of the experience and we learned how to deal with these problems in a professional manner. The experience we gained will certainly be of value to us in the industry.

Project Eve is a piece of software we are all proud to have our name on. We put blood, sweat and tears into getting it up to a standard we could be happy with and are glad to say it's something we would have no problem putting on our respective CVs.

Appendix B: Log Of Bug System

All of the following logs were taken from the system on Tuesday March 15th.

Final Tally

In the end, we have:

- **4 open** issues
 - 3 tagged as *optional enhancements*
 - 1 tagged as a *minor bug*
- **33 closed** issues
 - 12 tagged as *optional enhancements*
 - 21 tagged as *bugs*

Open Issues

#6 Incorrect login, displays errors on main page instead of modal

The error messages are displayed in a big red box on top of the hero image instead of being displayed above the login form. Maybe we should look at making `handleLogin` return to the login view instead if details are incorrect.

- Evan Smith added the issue on Feb 10
- Evan Smith added the **bug** label on Feb 10

#12 Breadcrumbs on every page

- Evan Smith added the issue on Feb 10
- Evan Smith added the **enhancement** label on Feb 10

#14 PDF Tickets

And maybe PDF some other views to simplify the Electronic Info Packs

<https://github.com/barryvdh/laravel-snappy>

- Evan Smith added the issue on Feb 10
- Mervyn Galvin added the **enhancement** label on Feb 11

#22 Sign maps key for better security

https://developers.google.com/maps/documentation/static-maps/get-api-key?hl=en_US#dig-sig-key

Pretty easy stuff, I'm just putting this here to remind me in the morning

- Mervyn Galvin added the issue on Feb 10
- Mervyn Galvin self-assigned this 18 days ago
- Mervyn Galvin added the **enhancement** label 17 days ago

Closed Issues

#1 Obfuscate ticket number in link

- Mervyn Galvin added the enhancement label on Feb 8
- Mervyn Galvin added a commit that referenced this issue on Feb 11
- Mervyn Galvin closed this on Feb 11

#2 Key constraint restricts image upload

There's a foreign key constraint on partners.picture >(references)> media.id

There're 2 ways of approaching this:

remove the constraint and allow partner image upload on partner creation/update
don't allow image upload on partner creation/update and require the user to select the image from a list/something like that

I prefer the first option but I'll get you input on this

- Mervyn Galvin added a commit that referenced this issue on Feb 9
- Mervyn Galvin added a commit that referenced this issue on Feb 9
- Mervyn Galvin closed this on Feb 9

#3 Search function for users

A simple search function to begin with along the lines of

```
User::where('name', 'LIKE', '%'.$query.'%')->get();
```

Later we can look at FullText search

- Evan Smith added the enhancement label on Feb 9
- Darragh King was assigned by Evan Smith on Feb 9
- Mervyn Galvin closed this 18 days ago

#5 Print ticket maximises when 'cancel' is pressed

When viewing an event you are attending and you press the 'Print' button, the dialog for printing shows up as expected. However, if you click cancel, the dialog window closes and the ticket maximises on the screen so that you no longer can see the event.

- Eimear Crotty added the bug label on Feb 10

Mervyn Galvin:

I'll change this to open the printable ticket in a new window then open the dialog

- Mervyn Galvin added a commit that closed this issue on Feb 10
- Mervyn Galvin closed this in eefbbd8 on Feb 10

#7 Empty message when nothing is found

When viewing a page that's supposed to have content from somewhere, we should have a "you're not attending any events soon, maybe you want to check out some other events" message or something to make sure they/we know there's not a problem with the data, just that there is no data to show.

The following pages need messages:

1. /user/profile
 2. /user/myEvents
- Evan Smith added enhancement bug labels on Feb 10
 - Eimear Crotty self-assigned this on Feb 10
 - Eimear Crotty added a commit that closed this issue on Feb 10
 - Eimear Crotty closed this in 485ef5a on Feb 10

#8 No way to get to your past events

We need a menu icon or a link somewhere to get to them.

- Evan Smith added enhancement bug labels on Feb 10
- Darragh King added a commit that closed this issue on Feb 14
- Darragh King closed this in 8096a86 on Feb 14

#9 Need to add a way for users to add their photos and media to events

Currently no way for normal attendees to add their photos and videos to the events, we should probably fix that.

- Evan Smith added the enhancement label on Feb 10
- Evan Smith self-assigned this on Feb 10
- Evan Smith added the bug label on Feb 10
- Evan Smith closed this 29 days ago

#10 tagline not stored when creating or updating event

- Eimear Crotty added the bug label on Feb 10
- Eimear Crotty self-assigned this on Feb 10
- Eimear Crotty closed this on Feb 10

#11 need to create some sort of button for creating event from index page

- Eimear Crotty added the enhancement label on Feb 10
- Eimear Crotty self-assigned this on Feb 10
- Eimear Crotty added a commit that closed this issue on Feb 10
- Eimear Crotty closed this in a04bb87 on Feb 10

#13 QR Code is different every time I reload the page

Maybe it's to do with the encryption but every time I reload the page, it generates me a new QR code.

@Colm Cahalane can you confirm that this is okay?

- Evan Smith added the bug label on Feb 10
- Colm Cahalane was assigned by Evan Smith on Feb 10
- Colm Cahalane removed the bug label on Feb 11
- Colm Cahalane closed this on Feb 14

#15 if incorrect creation of event, returns to main page with errors instead of create page

- Eimear Crotty added bug enhancement labels on Feb 11
- Eimear Crotty self-assigned this on Feb 11
- Eimear Crotty added a commit that closed this issue on Feb 11
- Eimear Crotty closed this in 6ba76c0 on Feb 11

#16 Events are unviewable

- Mervyn Galvin added the bug label on Feb 11
- Mervyn Galvin self-assigned this on Feb 11
- Evan Smith added a commit that closed this issue on Feb 11
- Evan Smith closed this in 6e06d7c on Feb 11

#17 Creating new location from modal results in 500 error

Also, if you get more than one set of errors (i.e. submit incorrectly filled form more than once) the errors just stack and don't get replaced

- Evan Smith added the bug label 29 days ago

Colm Cahalane:

This is because featured image isn't included there. I'll do the work and have it included in that modal.

- Colm Cahalane added a commit that closed this issue 28 days ago
- Colm Cahalane closed this in da8a9ef 28 days ago

#18 Undefined route error after logout

- Evan Smith added the bug label 29 days ago
- Evan Smith added a commit that closed this issue 29 days ago
- Evan Smith closed this in a46c93e 29 days ago

#19 Can't delete partner that is associated with events

Do a check to see if the partner is partnered before allowing to delete

- Mervyn Galvin self-assigned this 29 days ago
- Evan Smith added the bug label 29 days ago
- Mervyn Galvin added a commit that closed this issue 27 days ago
- Mervyn Galvin closed this in d9c3b1c 27 days ago
- Mervyn Galvin added a commit that referenced this issue 27 days ago

#20 media/upload route undefined error when logged in as regular user

- Evan Smith added a commit that closed this issue 29 days ago
- Evan Smith closed this in 9357915 29 days ago

#21 Viewing Tickets should be encrypted

- Evan Smith added the enhancement label 27 days ago
- Mervyn Galvin closed this 19 days ago

#24 No edit staff page

When going through the admin index page I noticed there's an edit staff button but no function/view for it.

- Mervyn Galvin added bug enhancement labels 16 days ago

Colm Cahalane:

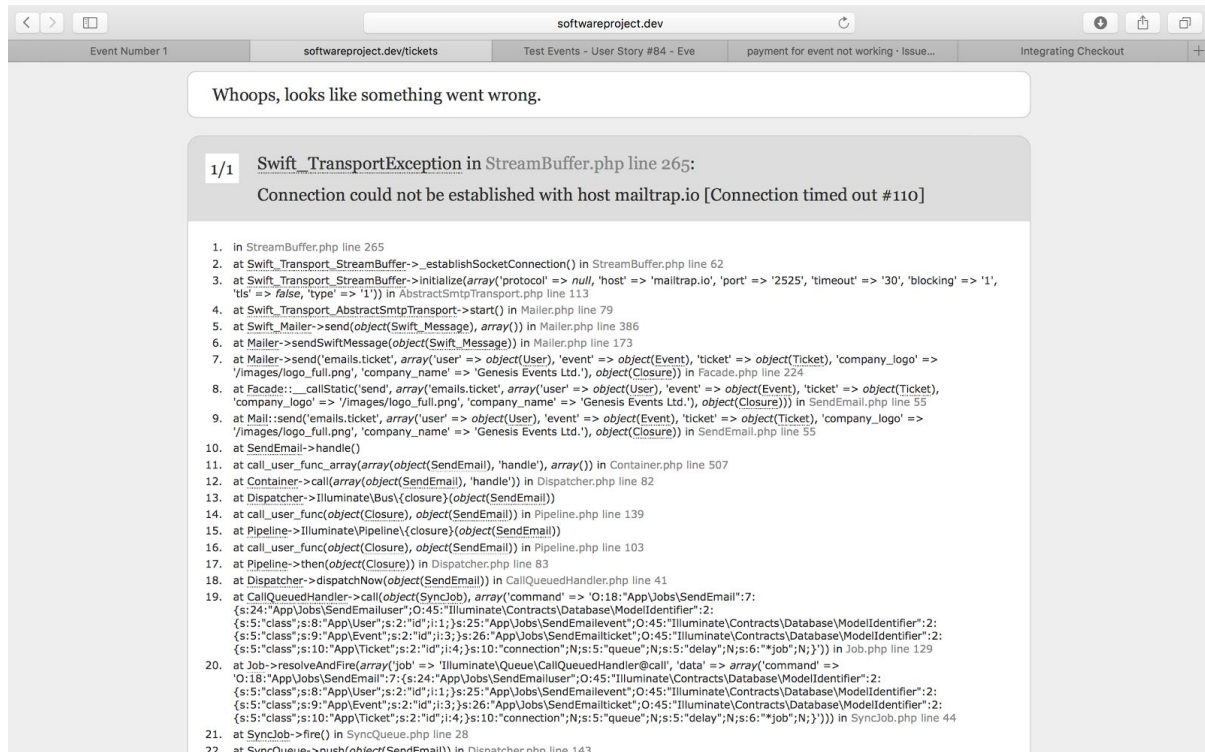
Resolved this in a quick way. Each user profile page has options to promote/demote users. I've edited the section to match.

- Colm Cahalane closed this 12 days ago

#25 payment for event not working

@Colm Cahalane When I try to pay for an event ticket, the window won't close. As far as I can see the details I have input are correct. I tried the same details on the Stripe API page and they work there.

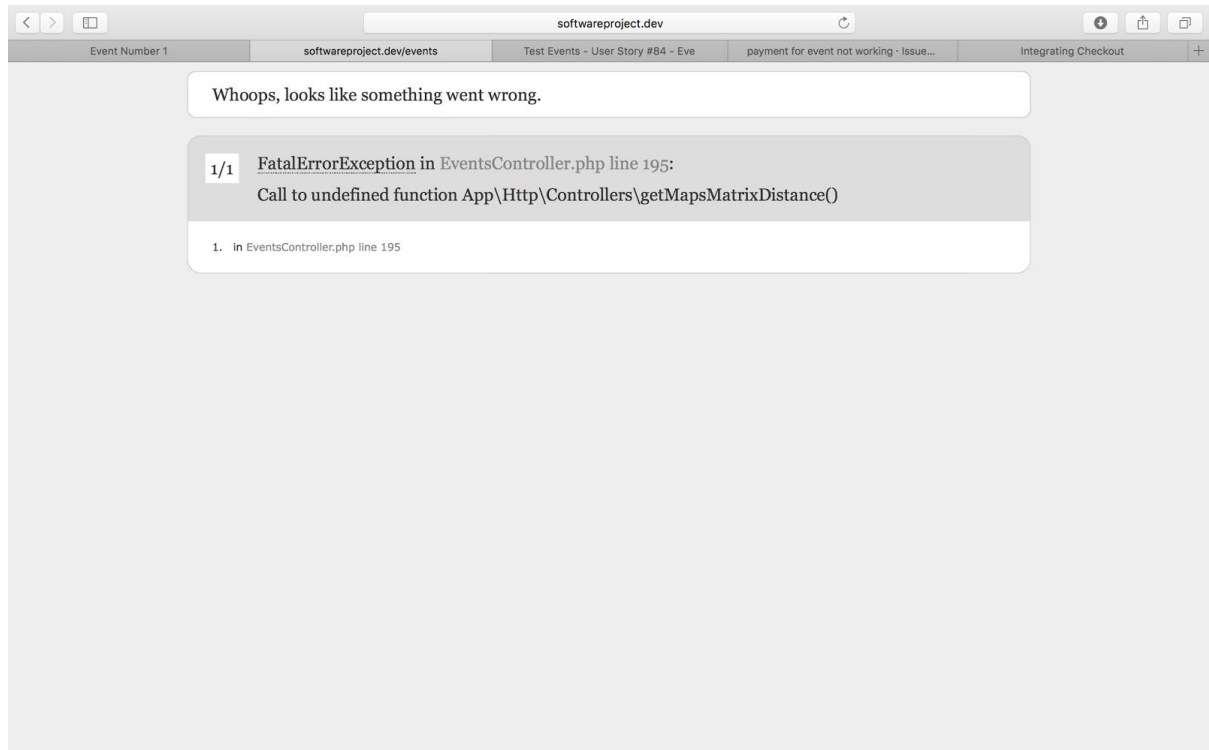
Actually, I've just managed to get the window to close. This involved pressing enter a few times after entering my phone number. The payment seemed to go through (the button went green) but then a window full of errors came (see the attached photo).



- Eimear Crotty added the bug label 14 days ago
- Mervyn Galvin:
- This has to do with the VM not having a mail server set up. This works in production (i.e. when connected to an SMTP server)
- Mervyn Galvin closed this 14 days ago

#26 event creation not working

@Mervyn Galvin in line 195 of the function 'store' in 'EventsController.php', the function 'getMapsMatrixDistance' is called. Apparently it's undefined? This happens when I try to create a new event from scratch. I've attached the error message to help with this.



The same thing happens when I try to edit an event and store it, it's on line 338 in EventsController.php.

#27 'cancel' icon changes to 'C'

This happens when you click on an event card in events.index to view more of the event, without actually going to the event page itself.

- Eimear Crotty added the bug label 12 days ago
- Mervyn Galvin closed this 12 days ago

#28 any user allowed to begin to create event

Any user can begin to create and event, i.e. they see and can fill in the event creation form. It is only when they press submit that they are told they don't have permission to 'edit events'.

- Eimear Crotty added the enhancement label 12 days ago
- Mervyn Galvin added the bug label 12 days ago
- Mervyn Galvin self-assigned this 12 days ago
- Mervyn Galvin added a commit that closed this issue 12 days ago
- Mervyn Galvin closed this in 4c4b7f4 12 days ago

#29 Partner ID is required

- Mervyn Galvin added the bug label 12 days ago
- Mervyn Galvin self-assigned this 12 days ago
- Eimear Crotty added a commit that closed this issue 8 days ago
- Eimear Crotty closed this in 521e37c 8 days ago

#30 not able to read pdf document as photo

I know it shouldn't be able to do this. At the moment it comes up with a Laravel 'NotReadableException' error, instead of returning to the event edit page with a 'file must be either a .jpg or a .png file'. I don't know if this is possible, so I'm marking this as an enhancement instead of a bug.

- Eimear Crotty added the enhancement label 10 days ago
- Evan Smith added a commit that closed this issue 8 days ago
- Evan Smith closed this in e2d465c 8 days ago

#31 create location modal not working on event edit page

When I select 'Create New Location' and enter valid details into the form, I hit 'Create Event'. The modal doesn't close and the 'new location' is not reflected in the list of locations in the drop down menu.

- Eimear Crotty added the bug label 10 days ago
- Eimear Crotty added a commit that closed this issue 8 days ago
- Eimear Crotty closed this in 71bce2d 8 days ago

#32 upper limit of distance between partner and event.

The upper limit of the distance between a partner and an event is 999999.99m. If the distance is greater than this, it is shown as 999999.99m in the information pack. This should not be an issue if all of our partners are in Ireland, but may be something to consider in the future.

- Eimear Crotty added the enhancement label 10 days ago
- Eimear Crotty closed this 8 days ago

#33 event creation allowed without start, end date or feature image

Start and end date time defaults to Wed, Nov 30, -0001 12:00 AM.

- Eimear Crotty self-assigned this 9 days ago
- Eimear Crotty added the bug label 9 days ago
- Eimear Crotty added a commit that closed this issue 9 days ago
- Eimear Crotty closed this in 6b62108 9 days ago

#34 negative value allowed for ticket price

- Eimear Crotty added the bug label 9 days ago
- Eimear Crotty self-assigned this 9 days ago
- Eimear Crotty added a commit that closed this issue 9 days ago
- Eimear Crotty closed this in 16ce0df 9 days ago

#35 Location Modal: Errors not clearing + Prominent

After getting an error in the location modal, previous errors should be cleared.

Errors are also not prominent enough as I only discovered this bug because I had click the button several times, assuming nothing had happened. Red box and scroll up maybe?

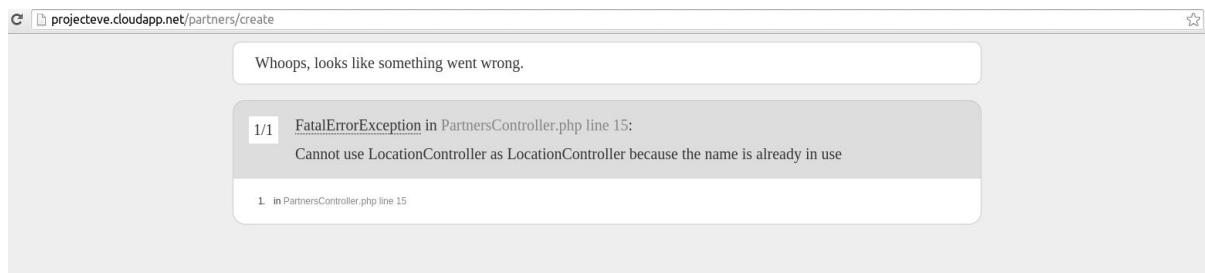
- Eimear Crotty added a commit that closed this issue 7 days ago
- Eimear Crotty closed this in bef33b9 7 days ago

#36 Events Creation: Doesn't check valid time

Entered in "hdijasgugfiuasguif878739824uy" as the message and it was accepted. We should disable that input.

- Eimear Crotty added a commit that closed this issue 8 days ago
- Eimear Crotty closed this in 704c9b8 8 days ago

#37 Partners: Error when creating a partner



- Eimear Crotty added a commit that closed this issue 7 days ago
- Eimear Crotty closed this in d1208a8 7 days ago

#38 Locations

Entering gibberish into the location creation page's longitude and latitude fields results in some weird coordinates.

Maybe add disabled to the input field?

- Eimear Crotty added a commit that closed this issue 8 days ago
- Eimear Crotty closed this in 73d13b7 8 days ago

#39 Admin Page: Add staff button goes nowhere

We should probably link to a search page or something for users? Or just remove the button altogether?

- Eimear Crotty added a commit that closed this issue 7 days ago
- Eimear Crotty closed this in b078a66 7 days ago

Appendix D: Open Source Software Licences

The Eve project was made possible through the use of Open-Source software. Here we include credit to the original developers of this code and the licence by which we used their code.

Laravel

The Laravel framework is open-sourced software licensed under the MIT license.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SimpleQRCode by SimpleSoftware

The MIT License (MIT)

Copyright (c) 2014 Simple Software LLC www.simplesoftware.io

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES

OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

GoogIMapper by Brad Cornford

The MIT License (MIT)

Copyright (c) 2014 Bradley Cornford <me@bradleycornford.co.uk>

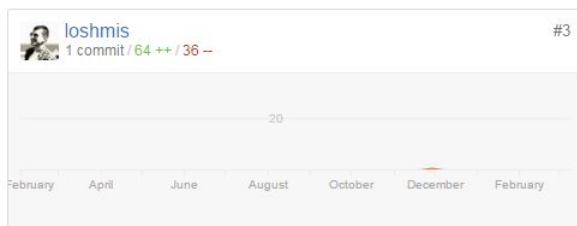
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Translation by Steve Bauman

Evan Smith (TheJokersThief), of the Eve Project team, is a contributor to the Translation package.



Further to this, Translation is a wrapper to Levan Velijanashvili's Google Translate PHP package, which is in turn distributed under the MIT licence.

Copyright (c) 2013 Levan Velijanashvili

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Laravel Facebook SDK by SammyK

The MIT License (MIT)

Copyright (c) 2014 Sammy Kaye Powers <sammyk@sammykmedia.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Guzzle

The MIT License (MIT)

Copyright (c) 2011-2016 Michael Dowling, <https://github.com/mtdowling>
<mtdowling@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Stripe-PHP by Stripe

The MIT License

Copyright (c) 2010-2015 Stripe

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Google ApiClient by Pulkit Jalan

The MIT License

Copyright (c) 2014 Pulkit Jalan

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND

NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

iCal by Eluceo

This package is released under the MIT license.

Copyright (c) 2012-2015 Markus Poerschke

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

jQuery & jQuery UI

jQuery Foundation projects are released under the MIT license.

Copyright (c) 2016 The jQuery Foundation

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES

OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Dropzone.js

Copyright (c) 2012, Matias Meno

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Magnific Popup

Copyright (c) 2014-2016 Dmitry Semenov, <http://dimsemenov.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING

FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Clockpicker.js

Copyright (c) 2014 Wang Shenwei

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Facebook JS SDK

Copyright 2016 Facebook

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Google Maps JS SDK

Copyright 2016 Google

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Knockout.js

The MIT License (MIT) - <http://www.opensource.org/licenses/mit-license.php>

Copyright (c) Steven Sanderson, the Knockout.js team, and other contributors
<http://knockoutjs.com/>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Materialize

The MIT License (MIT)

Copyright (c) 2014-2016 Materialize

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.