

# TM470 EMA

AN EVENTS MANAGEMENT SYSTEM FOR A TRAINING COURSE PROVIDER

BLAIR FLEMING (K170784X)

## Contents

Introduction .....	4
Supporting material .....	5
Source code.....	5
Application link .....	5
Problem description.....	6
Nature and context of the problem.....	6
Proposed solution or recommendations .....	8
Proposed solution .....	8
Project output .....	9
Project planning .....	10
Analysis of likely impact.....	11
Benefits to workflow.....	11
Legal, social, ethical and professional issues .....	12
Equality, diversity and inclusion concerns .....	13
Account of related literature .....	14
Domain modelling.....	14
Requirements and analysis .....	15
Design, implementation, and testing.....	16
Account of project work and its outcome .....	17
Domain modelling.....	17
Domain definitions.....	17
Domain structural model .....	18
Business processes and rules.....	20
Requirements.....	21
Functional requirements.....	22
Non-functional requirements .....	26
Additional requirements.....	27
Analysis .....	28
Analysis structural model.....	28
Loop invariants.....	29
Object diagram.....	30
Design.....	31
Choosing a framework .....	31

Navigability.....	32
Design structural model.....	33
Implementation .....	36
Initial iteration.....	36
Second iteration.....	37
Third iteration .....	38
Final iteration.....	39
Testing.....	41
Verification.....	41
Validation .....	44
Review of current stage of project work .....	51
Current stage of project work.....	51
Remaining project work to be completed .....	52
Recommendations for project continuation .....	53
Review of project management.....	54
Project lifecycle model.....	54
Project artefacts.....	55
Approach for future projects .....	56
Review of personal development .....	57
What I have learned.....	57
What challenges I faced .....	58
What I would do differently.....	59
References .....	60
Appendices.....	62
Appendix 1 – Project Glossary .....	62
Appendix 2 - Risk Register.....	65
Appendix 3 – Requirements.....	67
Appendix 3a - Use cases.....	67
Appendix 3b - Use case scenarios.....	68
Appendix 3c - User stories .....	78
Appendix 4 - Volere Templates.....	79
Appendix 4a - Functional requirements .....	79
Appendix 4b - Non-functional requirements.....	88
Appendix 5 - Implementation classes.....	92
Appendix 6 - Testing .....	93
Appendix 6a - Verification Table.....	93

Appendix 6b - Validation Table (Functional requirements).....	97
Appendix 6c - Validation Table (Non-functional requirements).....	100

## Introduction

This project, titled *An Events Management System for a Training Course Provider*, explores the problem of an ineffective workflow in a real-world system and how the development of a software system can help to solve that problem. The project uses a problem statement and supplementary domain descriptions and rules to provide context to the problem and establish a foundation for modelling the problem domain.

The domain is then refined through a process of requirements elicitation where interviews with project stakeholders take place, and a more thorough understanding of the workflow is established. The output of this process is analysed to produce a logical model that describes the entities and their relationships within the domain and the processes and rules that govern functionality and constraint.

Through the design and implementation process, the model will transform into a working software system using standard design patterns and methods of reuse, and modern practical tools. The resultant system will then be tested against the original problem statement and set of requirements to verify that the individual components of the system meet their specification, and that the completed solution is valid.

The goal of this project is to develop a software system that provides a high-quality solution to the problem outlined in the problem statement, and that a structured method of project management and delivery is followed and catalogued. In achieving this goal, many legal, social, and ethical concerns associated with the client organisation will be considered, and the impact of a successful solution will be analysed.

## Supporting material

### Source code

The source code for the application developed and implemented in this project is stored in a GitHub repository and is available via the link below:

**Repo:** <https://github.com/blair3003/beacon>

### Application link

The application is hosted on the Amazon AWS cloud service and can be accessed via the link and login details below:

**App:** <http://bob-master.eba-xtpajqn5.eu-west-2.elasticbeanstalk.com>

**Email:** [admin@beacon.com](mailto:admin@beacon.com)

**Password:** password

## Problem description

### Nature and context of the problem

A problem statement for the project is provided in **Box 1** below.

The British Emergency and Ambulatory Care Organisation (BEACON) is an institution with a well-established reputation for delivering high quality pre-hospital emergency care training to health professionals in remote and rural Britain. Demand for places on their many prestigious training courses, such as the Remote Emergency Care Course affectionately known as the 'Recce', has increased so much that the organisation can no longer effectively manage its workflow using traditional pen and paper methods of administration.

Currently, the organisation is reliant on historical data recorded in physical folders and stored in the rapidly shrinking storage facilities on the premises. Administrative staff are required to scour these records to find previous course information such as the date and venue of an event, the list of trainees that attended the event, and the trainers involved in the course delivery. This information is then used to plan future events, balance trainer schedules, and track trainee eligibility and renewal status for certain courses.

The complexities involved in preparing and delivering training courses have become so great that the organisation must consider downsizing their offerings to complete course deliveries on time. This is in direct conflict with the desire from the board of directors to expand the portfolio of available courses and deliver training to a greater number of people. The pressure to scale up an already inefficient process of event planning and delivery is having a profound impact on the administrative workload.

Ineffective course administration is also having a negative effect on the quality of the training provided. Trainers are often mistakenly being added to courses with conflicting time slots leading to a shortage of trainers on courses. Trainees without the appropriate level of experience are also mistakenly being added to courses that they are not qualified for resulting in wasted time and resources. Dissatisfaction of both trainers and trainees is evidenced in the increasingly negative feedback produced at the end of every event.

Administrative staff are growing increasingly frustrated and overwhelmed by the complicated and time-consuming process of course administration. The organisation is beginning to see high employee turnover as staff members are leaving in increasing numbers and new staff are not convinced by the current process enough to want to stay for any length of time. This has led the board to take the decision not to create any new positions within the team until the staffing issue has been resolved.

The combination of inefficient course administration, a worsening experience for course participants and trainers, and pressure from the board to increase course delivery despite their reluctance to expand the admin team, is impacting on the organisation's reputation

as a leader in high quality emergency care training. Potential customers have started looking for alternative training providers, and the organisation's once loyal base of trainers is gradually being picked off by competitors.

The organisation has explored an off-the-shelf software solution in the past called Administrate (2022) however they found it too difficult to shoe-horn their complex administrative needs into this generic events management system. They also found the interaction with the user interface for creating and managing events too dissimilar to their current offline workflow. The board took further issue with paying for features of the software that were not being used.

The decision was therefore made by BEACON's IT Lead, Donald, to approach an independent contractor to design and implement a dedicated software solution that better meets the needs of the organisation.

**Box 1** Problem statement.



## Proposed solution or recommendations

### Proposed solution

The solution to this problem will be the output of a development project that aims to deliver a software-based approach to training course administration that supports the needs of BEACON and the many stakeholders involved in the project.

A direct measure of the success of the project will be the extent to which the software solution satisfies the requirements established throughout the project lifecycle. Success may also be measured indirectly through the change in effectiveness of event administration and the quality of training provided over time after the solution is deployed.

The stakeholders of the proposed solution include the immediate staff members of BEACON and its board of directors. Other stakeholders include the trainers and trainees that participate in events and, by extension, the many hospitals and patients that stand to benefit from an improvement to the training pipeline of health professionals.

Only BEACON administrators should have direct access to the system and as such, the functional and quality requirements will be derived from these stakeholders. A consideration of the impact of the solution on the other stakeholders will be made, however, as they may be impacted by the change in the event administration process.

The additional data required to realise the proposed solution will include data definitions and examples of the various entities described in the problem statement. A description of the processes that take place during effective event administration, and the rules that govern them, will also be necessary.

Supporting information will be collected during the requirements phase of the project and refined throughout the project lifecycle. Visualisations will be produced at various levels of abstraction to describe the structural model of the system and support communication with stakeholders.

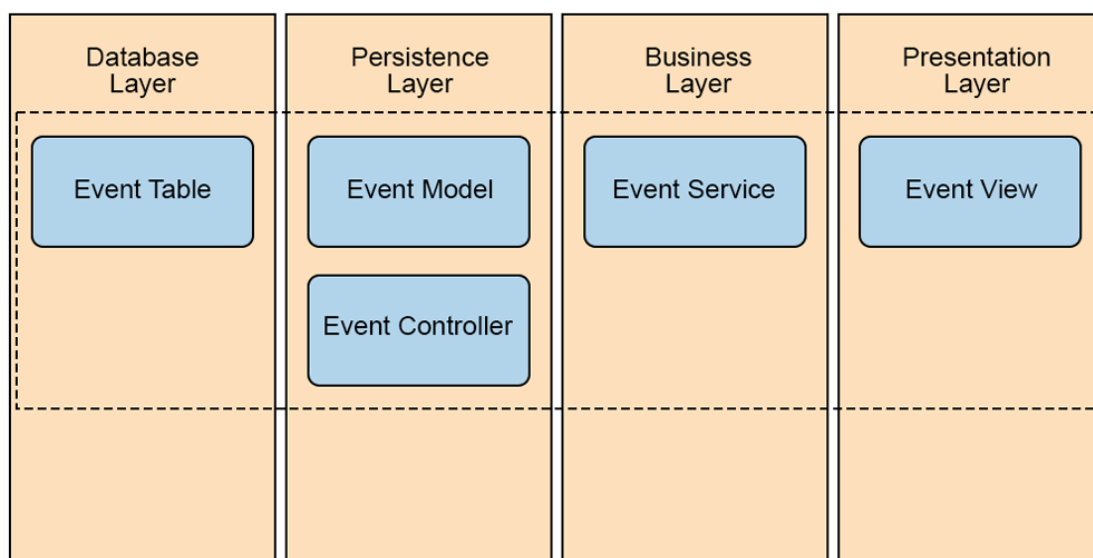
## Project output

This project is well-suited to deliver a database-driven application that provides the digital storage and fast retrieval of records in a computer-based system. The system will use a layered architecture described by Richards (2015), shown in **Figure 2**, upon which a walking skeleton (Dolfing, 2018) can be developed. The walking skeleton will contain the minimum number of bespoke components within the architecture required to deliver a viable solution.

The architecture includes a database layer where records of each entity will be stored in corresponding tables, and a persistence layer where entities are described and accessed through models and controllers. A business layer will contain logic operations to support the management of entities and enforce business rules and constraints, and a presentation layer will provide views of the system and the means of user interaction.

Defining a layered architecture in this way allows for the separation of concerns and assignment of distinct responsibilities to each layer. It also allows for a suitable framework to be chosen based on these architectural decisions. Choosing a framework early in the development process will provide a foundation for design choices and help keep the system flexible and amenable to change.

Laravel (2022a) is an open-source framework that supports the development of web-based database-driven applications using a model-view-controller design pattern (MDN Web Docs, 2022a). By using a framework such as Laravel, a bespoke solution can be produced that reuses a common set of pre-built and rigorously tested tools and components, leading to a faster development time and reduction of project risk.



**Figure 2** Walking skeleton.

The progress of this project is managed through a Project Schedule, shown in Figure 3, that highlights the key phases of the project and the important tasks for completion. Also shown in the Project Schedule is the overlap between the different phases of the project and the deadlines for major assignments.

<https://docs.google.com/spreadsheets/d/1C3sIPY5v9Fx1V9Tw7NAOC1PwXp-DpxheWKVZQodo5Gc/edit?usp=sharing>

[illegible]

**Figure 3** Project Schedule.

## Analysis of likely impact

### Benefits to workflow

Barwell, Hoad & Stewart (2020), in a whitepaper on the hazards of technology adoption, suggest some key considerations that should be made when introducing new technological solutions into a business to maximise the benefit of the technology and reduce the potential for “unintended and unwanted consequences”.

They suggest that before introducing a new system, the workflow should be well understood, and inefficiencies identified. The new system should then provide a solution that helps to solve these issues optimally.

In the case of BEACON, their organisation operates a mostly pen and paper method of administration. The inefficiencies of this environment that a software solution would look to eliminate include the slow pace of event administration, the logistical burden of searching for and retrieving records and the physical demand for storage space which is required to preserve records.

A software solution would offer a modern computer-based workflow that would ideally reduce the amount of time required to plan, organise, and deliver a training course event. The software would provide BEACON administrators simple methods of fast record retrieval without the need to leave their desk. The time savings gained would free up capacity to allow more focus on innovation and other company priorities.

A database would provide a digital means of storing trainee and event records as well as other relevant data, thereby eliminating the need for on-site storage and greatly diminishing the threat of damage or theft of data. Existing storage facilities could then be repurposed for other means such as housing training course equipment or additional office space.

## Legal, social, ethical and professional issues

The impact of new technology on the people and processes of a workplace is unavoidable (Barwell, Hoad & Stewart, 2020). A consideration must therefore be made to the legal, social, ethical, and professional issues that arise due to a change in way things are done. BEACON administrators must be made aware of their new responsibilities with regards to the new system workflow and they must adapt to the changes if it is to succeed.

The most important legal issue that administrators must be aware of is the need to abide by the laws regarding the role of a data controller as outlined in the Data Protection Act (2018). These laws stipulate that BEACON must get consent to store personal data and use it for an established purpose and have adequate facilities in-place to protect the data which they will manage with the new system.

Failure of administrators, and the organisation, to comply with the Data Protection Act can lead to significant financial penalties applied to BEACON, therefore it is essential they appoint a Data Protection Officer to oversee the process and ensure compliance.

A change in workflow because of the new software solution will influence more than just the efficiency of BEACON's event administration process. BEACON training courses are directly responsible for keeping healthcare professionals up to date on their ability to provide life-saving treatment to vulnerable people, therefore a system that indirectly affects this outcome should have this social and ethical responsibility factored into its development.

The new technology will also have an impact on the professional capabilities of the staff employed as administrators at BEACON. A transition from pen-and-paper to a computer-based workflow will require a base-level of technological competency followed by dedicated training to use the required software. Staff who are unable to meet the demands of the new system may create a new challenge for BEACON, and in extreme circumstances may need to be let go.

## Equality, diversity and inclusion concerns

During project development, the personal contribution of stakeholders and the impact their input can have on influencing the final design of the software solution should be recognised. It will be important to consider all contributions equally and avoid biases that may inadvertently exclude certain groups from experiencing the finished product in full.

From a technical viewpoint, it will be necessary to incorporate accessibility features into the software solution that conform with the Web Content Accessibility Guidelines (WCAG) international standard (W3C, 2018). These features should ensure that users with disabilities such as blindness and deafness are not excluded from using the system.

During the requirements elicitation phase of the project, there will be interaction with stakeholders with a variety of different protected characteristics such as age, gender and race. It is essential that each stakeholder's input is considered independently from these characteristics and as such, it may be prudent to capture information anonymously.

It will be important to provide a medium for stakeholders to voice honest criticisms of their current workflow without fear of reprisal from senior members of staff who may take offence to certain statements made about themselves or of the company. The anonymous data capture method should further help to alleviate this issue.

Finally, it will be important to consider unconscious bias that may be exhibited during the development of the software solution. Examples of when this could arise include making design decisions based on personal preference over what is best for the target users; discounting system requirements requested by "less important" stakeholders; and discarding contributions based on age or gender.

## Account of related literature

### Domain modelling

The use of the standardised visualisation technique, the Unified Modelling Language (UML), as described by Grässle, Baumann and Baumann (2005) will feature heavily in this project. UML will provide a framework to produce representations of the various abstractions of the system at different stages of the software development.

These visualisations, which will include structural models and activity, interaction, and use-case diagrams, will be used to support communication with stakeholders and allow them to better understand the complexity of the system and offer feedback which can contribute to future iterations of the software solution. Each visualisation will also provide a level of traceability which can help to identify changes in the system which led to improvement or deterioration of the system, and therefore support replication in future projects or resolve disputes related to dissatisfaction with the outcome of the software solution's deliverables.

This approach is well suited to the project due to the number of stakeholders with varying levels of technical competency. UML in this case will provide a medium for all stakeholders to better understand the system as it progresses. Visualisations produced using UML will have the added benefit of being easily understood by other software developers, therefore giving BEACON the flexibility to re-assign the entire project or certain parts such as design and implementation, to a third-party should they wish to do so.

Scott Ambler (2020) warns of the trade-offs that can come with using UML visualisations; each diagram runs a significant risk of becoming overly complex, thus becoming detrimental to the purpose of the visualisation. It is important, therefore, to keep visualisations as simple as possible and as few, in keeping with an agile approach to the given way of working.

## Requirements and analysis

System requirements, both functional and non-functional, will be gathered using several techniques described by Laplante (2013) including a semi-structured interview and user story creation. The output of these exercises will produce activity and use-case diagrams, and user stories, which should describe the main processes and rules of the business. These approaches are well-suited to this project due to capabilities of the different stakeholders. The IT Lead is clearly the most proficient and well-versed in the intricacies of the current workflow, therefore a one-to-one interview is most appropriate, whereas the administrative team have a wealth of practical experience from which user stories can be extracted.

Having identified the main business processes and rules and captured the key activities and common interactions of the existing workflow, a set of requirements can be produced and recorded using the Volere approach as described by Robertson and Robertson (2012). Information such as the requirement description, related use case or user story, and fit criterion will all be included on individual Volere templates for each requirement. These templates are appropriate to use in this project as a means of describing what the solution needs to accomplish to solve the problem.

In addition to the requirements generated in collaboration with BEACON stakeholders, it will be important to consider additional features that may require inclusion in the software solution. Such requirements should come from professional experience but also additional sources that make recommendations on commonly overlooked aspects of software design. A consideration of requirements based on professional experience might include the necessity for a mobile-first approach to user interface design, as almost 60% of global internet traffic now comes from mobile users (Ceci, 2022).

Returning to Barwell, Hoad & Stewart (2020), on the topic of the human aspects of migrating from paper-based to computer systems, they warn of the threat of adoption hesitancy due to new ways of working. It would be wise therefore, to consider requirements that support the existing workflow, specifically during the transition to the new system. Such requirements might include the need to incorporate a file storage system to allow the traditional paperwork to be scanned and uploaded as a stop gap while administrative staff adjust to the new system.

The complete set of requirements produced during the requirements gathering phase will govern how the software solution begins to take shape. In conjunction with the output of the domain modelling phase, the requirements will provide a strong foundation for analysis of the system and, eventually, validation of the completed solution to ensure the needs of the client have been met.



## Design, implementation, and testing

As part of the design and implementation phases, a consideration of the target framework will take place to ensure appropriate design decisions are made. It is vital that a framework is chosen early in the design phase to maintain a flexible system within the confines of a vigorously tested and industry approved environment. Laravel should afford this luxury due to the wide selection of reusable components within the framework that provide many different avenues from which software can be developed.

The Laravel core documentation (Laravel, 2022a) as well as the Laravel Best Practices repository (GitHub, 2022a) provides an essential guide to the framework, including the appropriate way to represent models, views, and controllers, and define other functionality including routing and database interaction. This information will be necessary in this project due to the complexity of the target framework and the necessity to deliver a high-quality solution to a professional standard.

The tracer bullet technique, as described by Hunt and Thomas (2019), will be used to create an initial minimum viable product that successfully implements a core requirement of the solution. Through a single piece of functionality which will highlight the required interface of the framework and the format in which the logical model of the system needs to adapt to, the walking skeleton should be realised. It will also provide a foundation to incorporate new functionality and 'flesh-out' the system. This approach is well-suited to the project due to the uncertainty around the development environment and should therefore provide a safe and steady process for making progress on the software solution.

Testing of individual components of the Laravel solution will follow a unit testing approach described by McReary (2019). McReary provides a guide on how to create an effective suite of tests in the Laravel framework and how to appropriately use fake data and assertions to verify method specifications. Automated tests will be produced to verify the functionality of controllers and services and will be routinely run after new functionality is added as part of a wider system regression testing process to ensure the system as whole is unaffected by minor changes to components. The completed software solution will be tested using a general validation approach - by comparing the system to the recorded requirements.

## Account of project work and its outcome

### Domain modelling

#### Domain definitions

As part of the initial discussion regarding the solicitation of a dedicated software solution, BEACON's IT Lead Donald provided a set of descriptions for the main entities present in the domain. This information, shown in **Figure 4**, was used in support of the problem statement to draft an initial structural model of the domain that described the conceptual classes and their associations.

The information also provided a foundation for discussion to take place during the planned interview as part of the Requirements phase.

A **trainee** is a person who attends a training course and is the recipient of training. On successful completion of a training course, a trainee is awarded a certificate.

A **trainer** is a person employed to support the delivery of a training course. To be eligible to act as a trainer on a specific training course, a trainer must have a valid certificate from that course.

A **training course** is a program of lessons delivered by a team of trainers to teach trainees a particular skill. Training courses have a specified duration, capacity, and certification period.

An **event** is an instance of a training course delivery. Events occur on set dates at a particular venue. Eligible trainees and trainers are assigned to an event based on the requirements of the training course.

**Figure 4** Domain definitions.

## Domain structural model

### *Conceptual classes*

The main entities of the domain that were strong candidates for inclusion as conceptual classes in the structural model included trainees, trainers, courses, and events as these corresponded to tangible yet distinct abstractions of real-world phenomena which could be accurately described in a potential software system.

Most of these entities translated directly into distinct classes, however it was important to consider the Don't-Repeat-Yourself (DRY) principle as described by Thomas and Hunt (2019). Trainers were effectively trainees with additional privileges; therefore, the Trainer class was considered a specialisation of Trainee. This meant avoiding duplicate behaviour and attributes as the Trainer subclass would now inherit these properties from the Trainee superclass.

Additional entities considered for inclusion were venues and certificates. Although a Venue could be stored as a static property of an event, it was possible that additional information about venues may need to be stored, and any changes to a venue such as its name be propagated to all the events dependent on it. A Certificate class would also be necessary to provide a dedicated method of representing course completion and trainer validity.

### *Relationships*

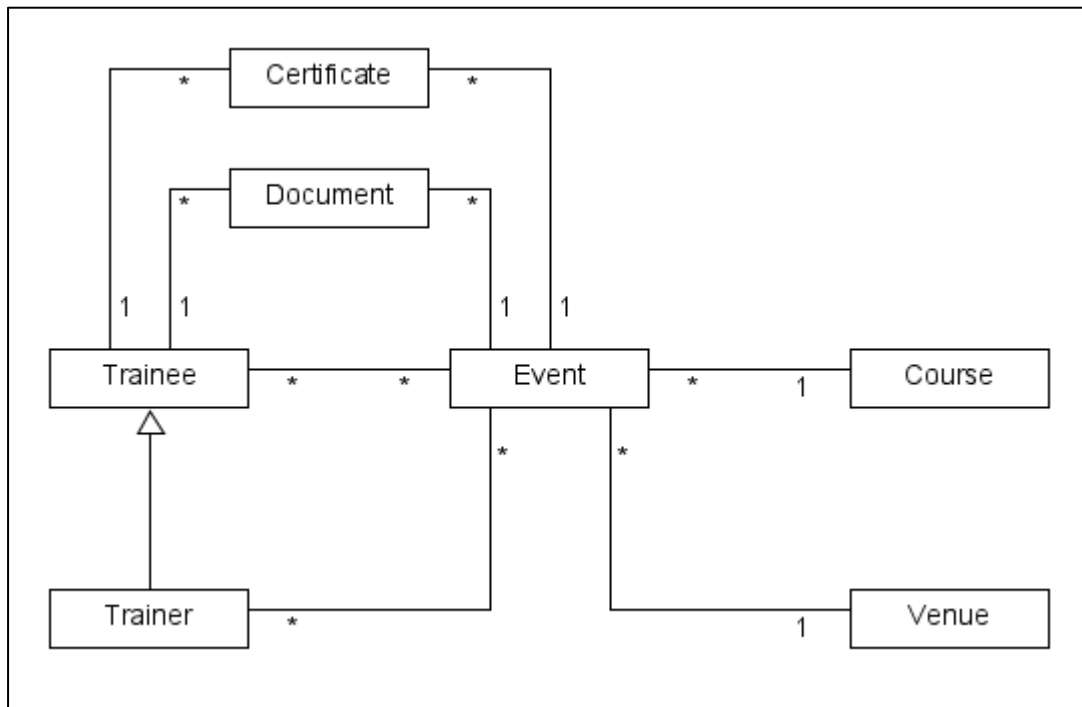
An event is composed of a course, venue and collection of trainees, trainers, and certificates. The Event class would therefore need a relationship defined for each of the other classes in the structural model. A one-to-many relationship between Course and Event, and Venue and Event was sensible since an event could only be associated with one of each of these classes but they themselves could be associated with many different events.

Many-to-many relationships between Trainee and Event, and Trainer and Event were sensible as an event could have many different trainers and trainees, and they each could be associated with many different events. One-to-many relationships between Event and Certificate, and Trainee and Certificate would define how an event or trainee could have many certificates, but each certificate would be associated with a single event and trainee.

A later iteration of the software solution determined that a Document class would also be appropriate for inclusion in the structural model. A document would be related to either a specific event or trainee instance, therefore one-to-many relationships with these other conceptual classes would be appropriate to establish.

### Completed model

The completed domain structural model that illustrates the conceptual classes, including their associations, multiplicities and constraints is shown in **Figure 5**. This model was used to communicate the potential system structure to IT Lead Donald, and his confirmation of its validity ensured a stable foundation for analysis going forward.



**Figure 5** Domain structural model.

## Business processes and rules

In addition to the domain description, IT Lead Donald provided an overview of the main business processes and rules that existed in the current system. The business processes, shown in **Figure 6**, describe potential functionality of the software solution.

New events are created when a training course is confirmed to take place on a certain date.

Trainees can register for the event by submitting a registration form. Forms are stored along with contact information in a trainee record.

Trainees are added to an event as soon as they are confirmed as eligible and have made payment.

The venue and trainers are added to the event as soon as their availability is confirmed.

Trainees are awarded a certificate if they successfully complete the training course.

Trainees can become trainers if they successfully complete a course and fit other criteria.

**Figure 6** Business processes.

The business rules, shown in **Figure 7**, describe potential constraints of the software solution.

The number of trainees on an event cannot exceed the course limit.

A trainee cannot also be a trainer on the same event (and vice versa).

Trainees and trainers cannot be added to multiple events that occur at the same time.

A venue cannot be selected for multiple events that occur at the same time.

A trainer must have a certificate from a previous event with the same course, and that certificate must still be valid.

**Figure 7** Business rules.

The combination of business processes and rules provided by Donald was used to create an initial set of system requirements which would be explored and expanded in the next phase of the project.

## Requirements

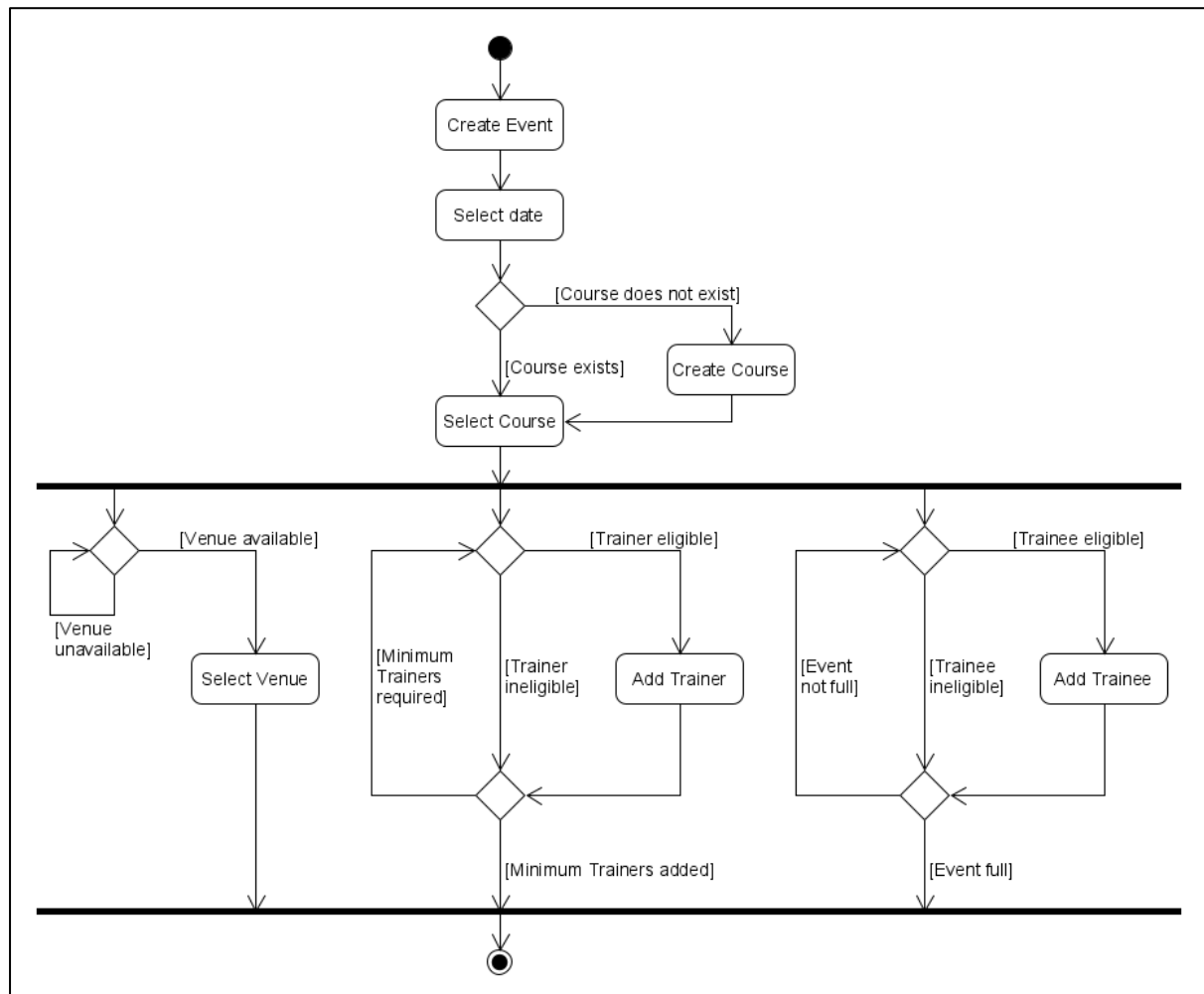
A semi-structured interview took place with BEACON's IT Lead Donald, during the week beginning March 1. This interview, which took place on-site at BEACON headquarters, provided an opportunity to identify a set of use cases from which functional requirements could be derived. The structured component of the interview required Donald to respond to questions related to the current workflow, whereas the unstructured aspect allowed for unprompted questioning that delved deeper into specific aspects of the workflow.

A follow up exercise took place with BEACON's administrative team, during the week beginning April 25 to identify non-functional requirements from a collection of user stories. This exercise was conducted online using software that allowed for anonymous submissions, and the results were reviewed in collaboration with Donald virtually.

## Functional requirements

### Activity diagram

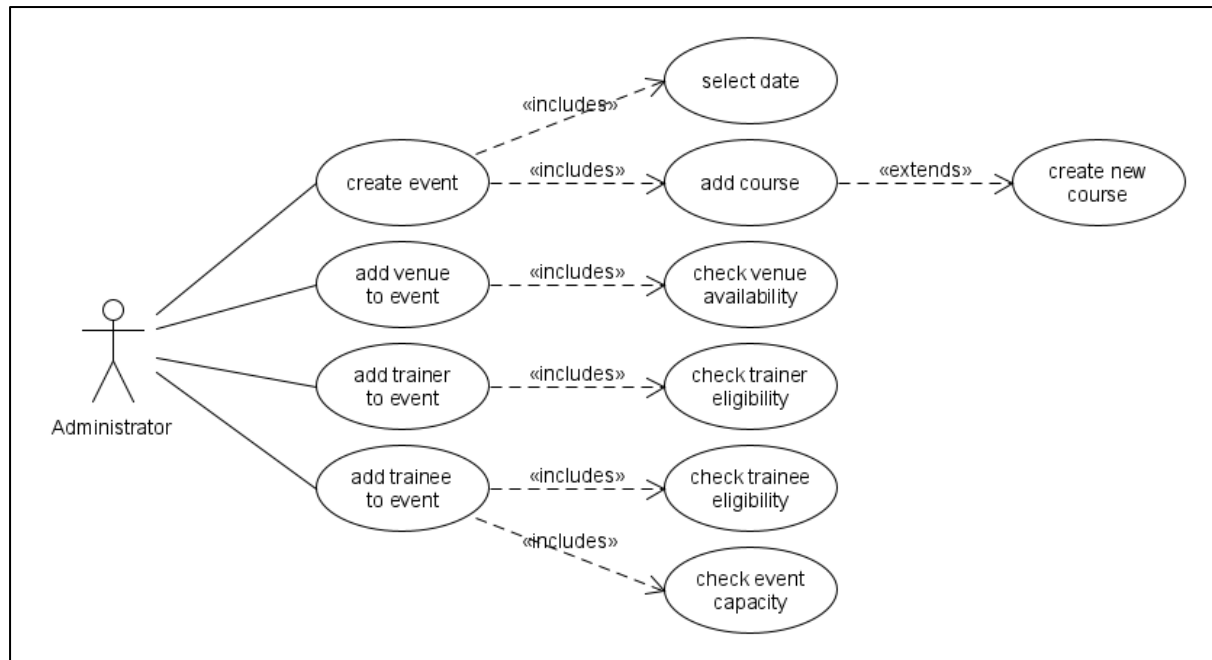
The task of determining functional requirements started by defining a single business process. The creation of an event was chosen for this purpose and an activity diagram, shown in **Figure 8**, was produced using standard UML notation to describe the process.



**Figure 8** Activity diagram for the creation of an event.

## Use case diagram

The actions in the activity diagram were transformed into use cases and expanded into a use case diagram, as shown in **Figure 9**. A single actor Administrator was included in the diagram to represent the main user of the system and their relationship with the use cases.



**Figure 9** Use case diagram for the creation of an event.



### Use case scenario

Each use case was given an identifier and a set of pre- and post-conditions to which the use case must satisfy. The identifier uniquely identified the use case to ensure that no duplicates would be made, and repetitions could be merged. A main success scenario was then drawn up for each use case, with a list of numbered steps necessary to complete the scenario, as shown in **Figure 10**.

<b>Identifier and name</b>	UC1 Create Event
<b>Initiator</b>	Administrator
<b>Goal</b>	A new event is created
<b>Precondition</b>	None
<b>Postcondition</b>	An event will have been created for a training course over the specified dates.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"><li>1 The administrator chooses to create a new event.</li><li>2 The administrator selects the desired course.</li><li>3 The administrator selects the start and end dates.</li><li>4 The system creates an event and gives it an identifier.</li><li>5 The system reveals the identifier to the administrator.</li><li>6 The system allows the administrator to add trainers, trainees and a venue to the event.</li></ol>
<b>Extensions</b>	<ol style="list-style-type: none"><li>2.a course does not exist<ol style="list-style-type: none"><li>2.a.1 The system offers alternative courses or the option to create a new course.</li><li>2.a.2 The administrator selects an alternative course or creates a new course.</li></ol></li></ol>

**Figure 10** Use case scenario for UC1 Create Event.

### *Volere template*

Each step of a use case scenario described a potential functional requirement. After removing ambiguity and duplication, each requirement was recorded on a Volere template, along with a suitable fit criterion and reference to the relevant use case(s), as shown in **Figure 11**.

Requirement #: FR1	Requirement Type: Functional	UC #: UC1
Description: The system shall enable the user to add an event.		
Rationale: Events must be added to the system so they can be managed effectively.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A new event has been stored in the system.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

**Figure 11** Volere template for a functional requirement.

## Non-functional requirements

A more direct approach was taken to obtain the non-functional requirements of the system. Members of BEACON's administrative team were asked to anonymously submit user stories in the format of:

*'As an Administrator I want [feature] so that [reason]'.*

The submissions were then reviewed and approved in collaboration with IT Lead Donald, and a set of non-functional requirements were derived. The requirements were then recorded on Volere templates along with the type of non-functional requirement, as shown in **Figure 12**.

<b>Requirement #:</b> NFR1	<b>Requirement Type:</b> Non-functional (Operational and environmental)	<b>UC #:</b> US1
<b>Description:</b> The system shall be accessible by the user remotely.		
<b>Rationale:</b> Users should be able to use the system from home and out of office.		
<b>Originator:</b> BEACON administrative team		
<b>Fit Criterion:</b> The system can be accessed from outside the BEACON internal network.		
<b>Customer Satisfaction:</b> 5	<b>Customer Dissatisfaction:</b> 3	
<b>Priority:</b> Medium	<b>Conflicts:</b> None	
<b>Supporting Materials:</b> None		
<b>History:</b> Created May 4 2022		

**Figure 12** Volere template for a non-functional requirement.

## Additional requirements

The business rules and processes originally identified as part of the initial discussion with IT Lead Donald in the Domain Modelling phase, shown in **Figures 6** and **7**, also needed to be assessed and recorded as requirements of the appropriate type.

The processes related to the creation of events and other entities, and adding trainees, trainers and a venue to an event, were already represented by suitable functional requirements. Corresponding functional requirements for awarding certificates and promoting trainees to trainers were also already included. What was determined to be appropriate as additional requirements, however, was the ability to search for different entities as well as upload documents related to events or trainees.

The rules that determined the validity of trainees and trainers to be added to events were then explored and certain rules translated to functional requirements. The rule that prevented events from going over their trainee capacity was deemed sound as well as the rule that suggested trainers could not be added to an event if they were already a trainee on the same event. Additionally, the rule that required trainers to have a valid certificate for an appropriate course was also represented by a functional requirement.

The rule that suggested trainees and trainers should not be added to multiple events that occur at the same time was not represented, however. Nor was the rule to restrict venues from hosting multiple events at the same time. It was considered feasible that an event may exist in parallel with other events at the same venue and allow trainees to participate simultaneously with other events. As such, it was deemed unnecessary to restrict the system to such an extent as to prevent this occurrence.

The complete collection of requirements (**Appendix 3**) and Volere templates (**Appendix 4**) produced during the Requirements phase are given in the appendices section.

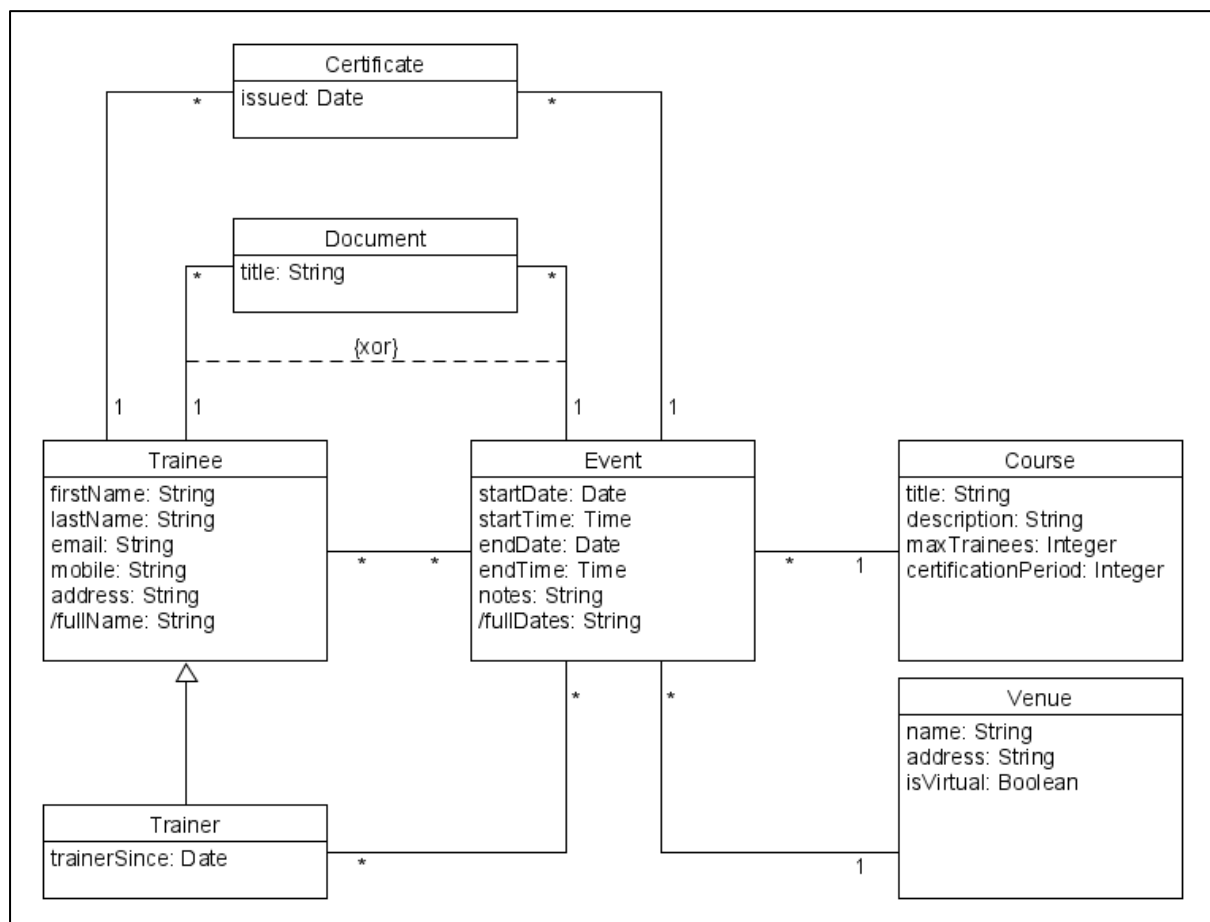
## Analysis

### Analysis structural model

Having gathered a comprehensive set of system requirements, the domain model was expanded to include suitable class attributes, as shown in **Figure 13**. Also included in this structural model is an exclusive-or constraint placed on the Document class. It was determined necessary to include this constraint as a document should be related to either a trainee or event but not both, at any given time.

Derived attributes were added to certain classes to represent information that was likely to be accessed regularly and calculated based on existing attributes. These derived attributes included `/fullName` - a concatenation of a trainees first and last name, and `/fullDates` - a more useful and complete representation of the date range of an event. Since these derivations are dependent on core attributes that are unlikely to be modified, there is a low risk of the derived attributes becoming invalid and as such, the overhead of maintaining them is negligible.

Some attributes such as `isVirtual` of the Venue class suggest potential subclassing may be appropriate, however a decision was made to leave this option open for a future iteration to implement as necessary.



**Figure 13** Analysis structural model.

## Loop invariants

The structural model contained loops which, if left unconstrained, could threaten the integrity of the model. Invariants were therefore created and expressed using UML's Object Constraint Language (OCL) to constrain certain associations.

An invariant was expressed for the loop Certificate, Event and Trainee such that; the trainee associated with a certificate must be included in the list of trainees for the event associated with the certificate. In other words, a trainee must be on an event to get a certificate for said event.

**context** Certificate **inv**:

```
self.event->trainees->includes(self.trainee)
```

An invariant was expressed for the loop Trainee, Trainer and Event such that; the list of events associated with a trainer cannot contain any of the same events associated with the corresponding trainee. In other words, a trainer cannot also be a trainee on the same event.

**context** Trainer **inv**:

```
(self.events->intersection(self.trainee->events))->isEmpty()
```

An invariant was expressed for the loop Document, Trainee and Event such that; a document must be associated with either a trainee or an event, but not both. In other words, a document belongs to a trainee or an event exclusively.

**context** Document **inv**:

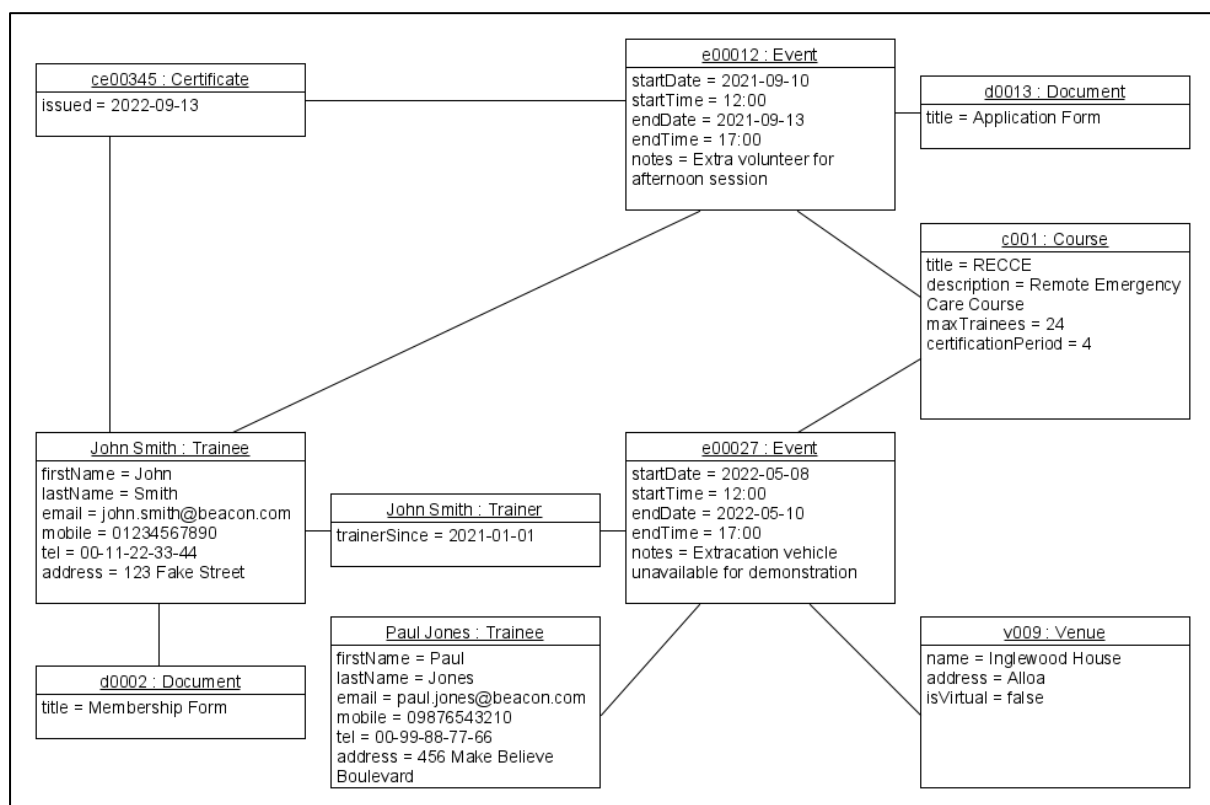
```
self.trainee->notEmpty() xor self.event->notEmpty()
```

## Object diagram

To help determine whether the structural model was sufficiently defined and constrained, an object diagram, shown in **Figure 14**, was produced to represent a possible configuration of the system at a given moment.

The object diagram shows examples of relationships that would be valid in a working solution. For example, the Trainee *John Smith* is associated with the Event *e00012*, and has obtained the Certificate *ce00345*. *John Smith* is also a Trainer on the Event *e00027* which shares the same Course, *c001*, as Event *e00012*.

Although not enforced by the model, the business rule that suggests a Trainer should have a valid certificate for a Course to be eligible to train on Events of that Course is respected by the diagram.



**Figure 14** Object diagram.

## Design

### Choosing a framework

The design phase of the project lifecycle supported the transition of the logical model of the system into a working implementation in a chosen software environment. It was therefore important to consider how the various system classes, including their attributes, associations and constraints would translate into the Laravel framework.

Laravel combines MVC architecture with a RESTful approach (MDN Web Docs, 2022b) to provide services based on web routes. Each route has a corresponding controller method that performs an action in response to a request type. For example, when a user makes a *GET* request to the events index route the EventController's `index()` method is called, which gathers a collection of event models and returns these to the user as part of a view. The RESTful approach is appropriate for this project because the application will be deployed on the web and resources will be transferred via client and server interactions.

The benefits of using an MVC approach to this project is that the application can be kept well-organised with distinct responsibilities applied to each individual component of the design. Models will be responsible for managing the information related to a single object, whereas controllers will be responsible for the behaviour of the models and their reaction to different system requests. Views will primarily focus on providing the visual interface for the user to interact and make requests.

A controller in Laravel could be considered a GRASP expert as it takes full responsibility for the management of corresponding entities, from providing different views of a collection of models, to creating, updating, and destroying single models. Controllers can also make use of supporting classes to manage models such as *form request classes* for validation and *service classes* to apply business logic.



## Navigability

In a working software solution, it should be possible to navigate from a given Event to the various other entities associated with it, and access information regarding these entities from the starting point of the Event. Navigating from an Event to its associated Trainers, Trainees, Certificates and Documents should all result in a collection, whereas the Course and Venue should provide a single result.

For Certificates and Documents, it should be possible to navigate back to the given Event directly, whereas for Trainees, Trainers, Courses and Venues - this should result in a collection of Events. Traversal between Trainers, Certificates or Documents and their associated Trainee should be possible, however in the case of Certificates and Documents, navigating from the Trainee should result in a collection.

Navigation between associated objects in Laravel is enabled by defining the relationship between each model within a dedicated method in the corresponding model classes. The Event model class for example, would have an `course()` method which defines a one-to-many relationship with the Course class and would return a single associated course when called.

## Design structural model

### *Representing classes*

Each of the main system classes could be represented in the framework through dedicated models, views, and controllers, as well as corresponding tables in the database. For example, an Event would have an Event model; an EventController; views for displaying the Event index, a single Event, the creation of an Event and editing of an Event; and an Event database table which stores the data for every single Event record.

Due to the many-to-many relationship between Events and Trainees, and Events and Trainers, the database would require additional pivot tables to store instances of these associations. The EventTrainee and EventTrainer entities were produced for this purpose. These classes would not require dedicated models; however it was decided that the inclusion of controller classes would be appropriate to manage the process of adding trainers and trainees to events.

These pivot tables would store a relationship between an Event and a Trainee (or Trainer) which otherwise could not be represented in the existing database tables without breaking the standard rules of normalisation. An Event record could not contain the list of Trainees in the same way a Trainee record could not contain the list of Events – it would either require additional table columns for each association, the number of which cannot be known beforehand, or a single column for every association which would result in a non-atomic value. Delegating the responsibility of managing the many-to-many relationships to dedicated pivot classes with corresponding database tables solves this issue.

### *System operations to retrieve associations and attributes*

The Laravel framework allows associated objects to be accessed through a dedicated system operation. This operation is a method with a name corresponding to the association, such as `Trainers()` in the Event model class, and contains a single line of code that returns the corresponding object or collection of objects.

Derived attributes are also represented using a system operation, with the naming convention `get<Name>Attribute()`, and instructions for retrieving the required data defined in the method body.

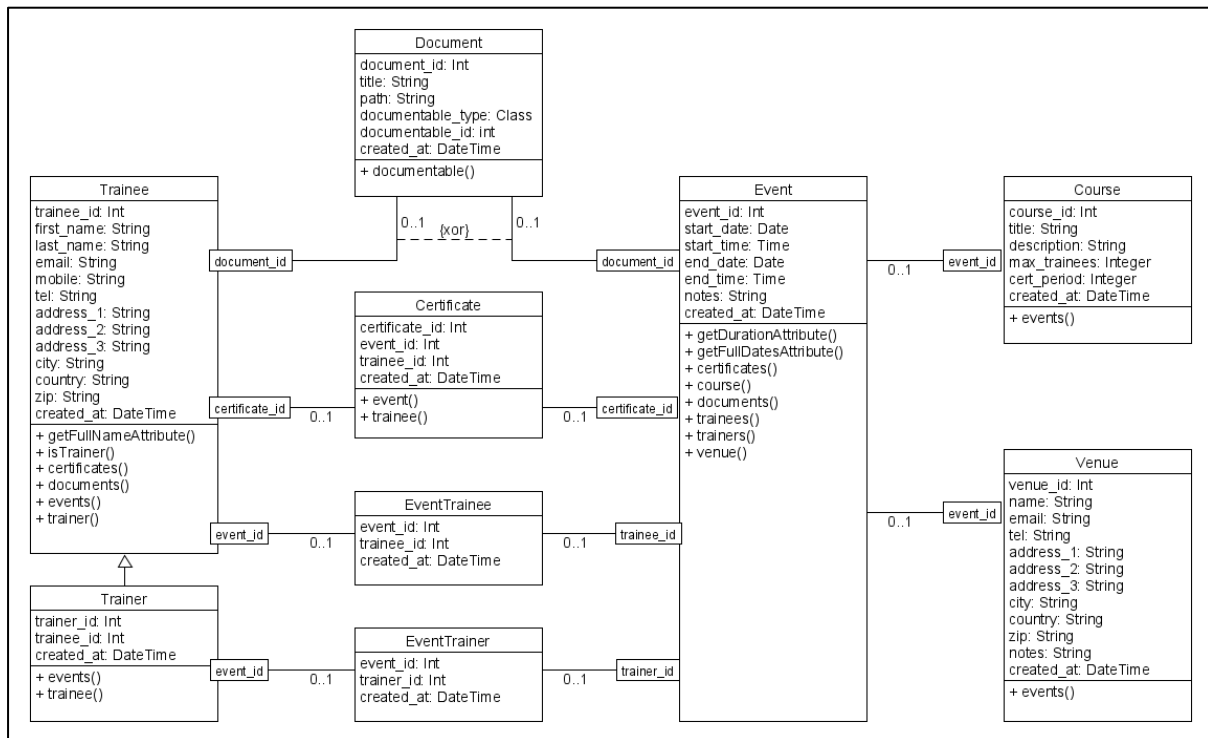
The operation to retrieve the associated object of a Document is a special case because the associated class is not pre-defined. The relationship between Document and either Event or Trainee is polymorphic therefore, per Laravel convention, a `documentable()` method was included which identified the associated object class from the `documentable_type` column of the documents table in the database. A polymorphic relation allows a class to be related to multiple other classes using a single association (Laravel, 2022c).

### *Qualified associations*

To further improve the readability of structural model visualisation, and to ensure the assumption that certain attributes uniquely identify their respective class instances, qualified associations were used in place of the one-to-many associations between each class. For example, by replacing the association between Event and Course with 0..1 at the Event end and *event\_id* at the Course end this could now be read as: *for any given combination of a course and an event\_id, there will be either one associated event or none.*

## Updated model

An updated structural model with updated attributes, qualified associations, system operations, and naming conventions appropriate to the framework, is shown in **Figure 15**. This model was used to confirm the design structure of the system with IT Lead Donald and provide a template from which to implement the software solution.



**Figure 15** Design structural model.

## Implementation

### Initial iteration

An initial iteration of the software solution was implemented using the tracer bullet approach. The Event, Course and Venue classes were translated into models with corresponding tables in the database and populated with dummy data using the PHP Faker library (GitHub, 2022b). Dedicated controllers and views were then created to provide the functionality to create and display a list of events.

This simple iteration, shown in **Figure 16**, provided an initial working solution that could be used as a foundation to slowly develop and incorporate additional functionality. In this iteration, a list of Events was displayed on an index page, with information related to their associations including Course and Venue, as well as other attributes, pulled from the database. This rudimentary implementation was enough to demonstrate the progress of the project to BEACON stakeholders and reassure them of the capability of the software solution.

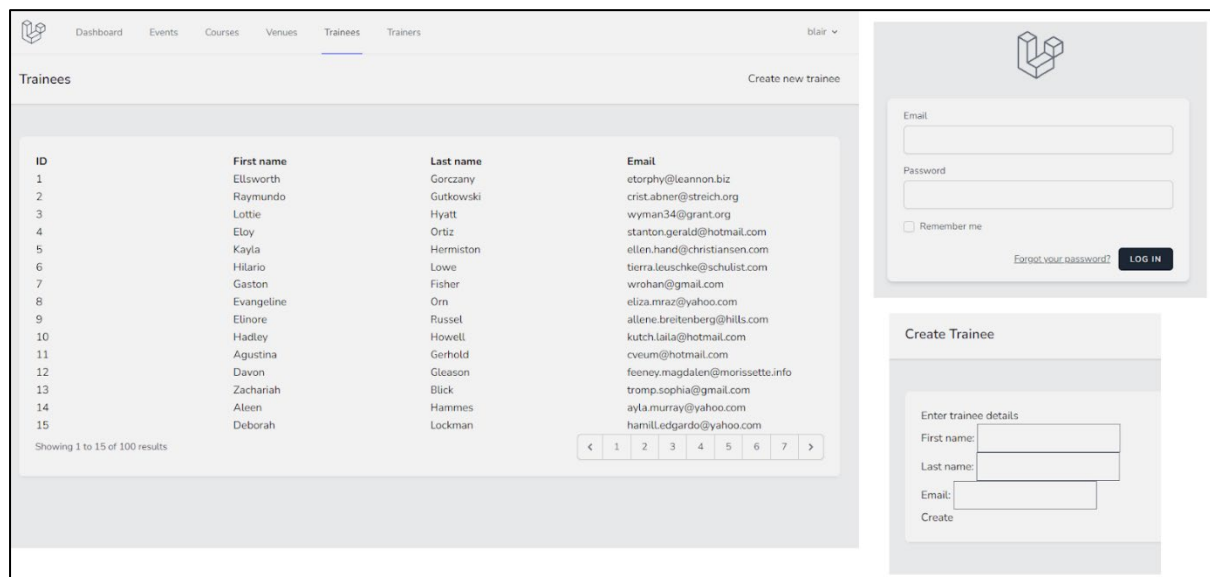
Events					
Course	Venue	Start Date	Start Time	End Date	End Time
Managed asymmetric orchestration	Bode-Lesch	2022-10-17	23:39:00	2022-10-18	23:50:58
Realigned optimizing interface	Wunsch, Ullrich and Bauch	2021-07-19	12:42:54	2021-07-21	03:59:59
Managed asymmetric orchestration	Bosco, Braun and Windler	2023-05-03	15:05:02	2023-05-04	03:46:18
Right-sized mobile installation	Wunsch, Ullrich and Bauch	2022-08-14	14:56:27	2022-08-14	15:15:45
Versatile dedicated capability	Bode-Lesch	2021-06-27	04:55:39	2021-06-27	05:18:33
Versatile explicit knowledgebase	Bode-Lesch	2023-05-02	17:37:56	2023-05-03	19:59:36
Managed leadingedge success	Bosco, Braun and Windler	2022-06-17	12:52:50	2022-06-18	06:00:56
Realigned optimizing interface	Bode-Lesch	2021-11-04	01:39:22	2021-11-05	07:56:11
Right-sized mobile installation	Prosacco, Champlin and Swaniawski	2022-05-08	13:48:17	2022-05-09	03:12:02
Managed asymmetric orchestration	Prosacco, Champlin and Swaniawski	2022-01-03	19:20:05	2022-01-04	20:56:09
Progressive homogeneous data-warehouse	Prosacco, Champlin and Swaniawski	2022-11-14	10:54:40	2022-11-15	19:12:04
Organic disintermediate help-desk	Prosacco, Champlin and Swaniawski	2021-12-25	21:29:37	2021-12-27	10:39:17
Cloned multimedia installation	Bosco, Braun and Windler	2022-11-12	19:59:06	2022-11-14	17:39:07
Cloned multimedia installation	Bode-Lesch	2023-02-26	21:12:55	2023-02-27	01:07:44
Cloned multimedia installation	Bosco, Braun and Windler	2022-04-14	06:45:54	2022-04-15	05:12:54
Right-sized mobile installation	Wunsch, Ullrich and Bauch	2023-01-02	01:35:25	2023-01-03	20:48:24
Managed asymmetric orchestration	Bode-Lesch	2023-02-20	22:32:04	2023-02-22	16:16:10
Managed asymmetric orchestration	Wunsch, Ullrich and Bauch	2021-12-06	04:30:19	2021-12-07	00:51:25
Managed asymmetric orchestration	Prosacco, Champlin and Swaniawski	2022-10-18	16:25:35	2022-10-20	02:32:02
Managed leadingedge success	Wunsch, Ullrich and Bauch	2021-07-15	03:25:28	2021-07-16	19:03:02
Realigned optimizing interface	Bosco, Braun and Windler	2021-10-01	11:54:19	2021-10-02	20:30:19
Realigned optimizing interface	Prosacco, Champlin and Swaniawski	2023-04-30	06:08:17	2023-04-30	11:32:53
Advanced bifurcated database	Prosacco, Champlin and Swaniawski	2022-05-25	21:29:17	2022-05-27	09:32:27

**Figure 16** Initial iteration using trace bullet approach.

## Second iteration

Building on the success of the initial iteration, additional models, views, and controllers were created for each of the other system classes according to the structural model, and the corresponding database tables were populated with more dummy data. Basic navigation, authentication and pagination functionality was also incorporated into the solution at this stage - all of which were reusable components provided by the Laravel framework.

This second iteration, shown in **Figure 17**, provided the ability to log in to the system, create, show, update and delete individual records for each class, and display the full list of records of each class on a single index page. A user could now log into the system using the login form, then access the index pages of each of main system classes. These pages would show a paginated list of all the records on the system for the given class, including certain associations and attributes. The user could also explore individual records and edit or delete them as required or create new ones.



**Figure 17** Second iteration with expanded functionality.

## Third iteration

The next iteration refined the solution by introducing form request classes to handle validation and service classes to support complex logical operations. Form request classes such as the `EventStoreRequest` class were necessary to ensure that only safe and valid data could be entered into the database. They also ensured that any associations linked to the object were viable, for example, when choosing a Course for an Event, the Course must exist in the database.

Service classes such as the `EventService` class contained the logic that determined whether an object could then be created given the validated form inputs. The `EventService` class, for example, contained the `addTrainee()` method which would throw exceptions in the cases of Trainees being present on the event already as a Trainee or Trainer, or the Event being at full capacity. If none of the exception cases were triggered, the Trainee would be added as normal.

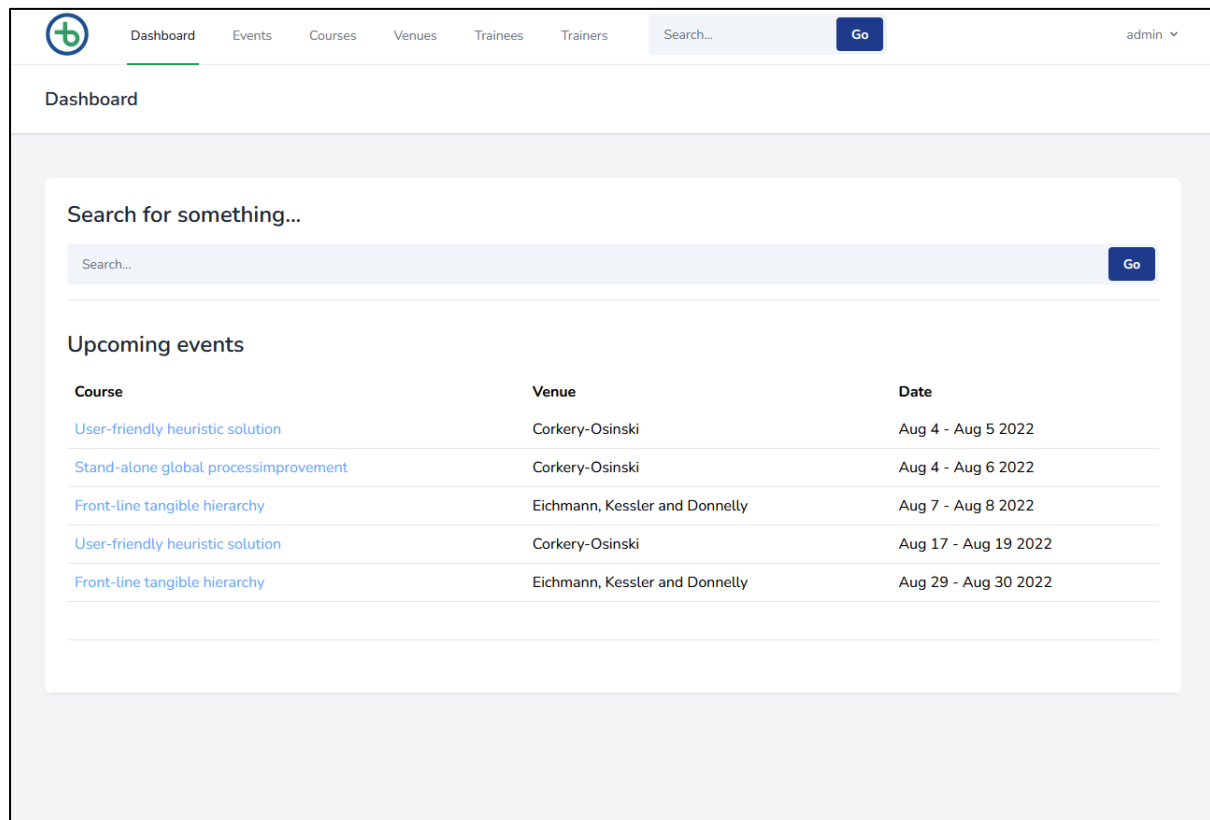
This functionality was initially developed in the controller classes but was moved to dedicated classes to satisfy the single responsibility principle as described in the Laravel Best Practices guide (GitHub, 2022a, *Business logic should be in service class*).

This third iteration, shown in **Figure 18**, applied constraints to the system to ensure business rules were adhered to and only appropriate data would be added to the database.

**Figure 18** Third iteration with validation, logic, and business rules.

## Final iteration

The final iteration, shown in **Figure 19**, incorporated the BEACON brand guidelines into the front-end design of the user interface. A simple dashboard and search mechanism was also created to provide a quick view of upcoming courses and access to other entities. These design improvements vastly improved the usability of the system and satisfied the quality requirements regarding the look and feel of the software solution.



**Figure 19** Final iteration with design improvements and dashboard.



The Event show page, shown in **Figure 20**, was implemented with interactive links to all the objects associated with the event, including course and venue details, and a trainee, trainer, and document list. Buttons for awarding certificates as well as adding and removing trainees, trainers and documents were also included on the page.

Down-sized fresh-thinking GraphicInterface, Jul 23 - Jul 25 2023

[Edit](#) [Delete](#) [Back](#)

### Event details

**Course:** [Down-sized fresh-thinking GraphicInterface](#)  
**Venue:** [Smith, Hyatt and McGlynn](#)  
**Start Date:** Jul 23, 2023  
**Start Time:** 2:52 pm  
**End Date:** Jul 25, 2023  
**End Time:** 2:48 am  
**Notes:** Et et iure quo iusto fugit consequatur nostrum. Consequatur quas id consequatur consectetur. Corporis aut est autem nam.

### Trainees (5/5)

[Add Trainee](#)

Name	Email	Certificate	
<a href="#">Cortney Anderson</a>	<a href="mailto:ofelia.rutherford@yahoo.com">ofelia.rutherford@yahoo.com</a>	<a href="#">Award certificate</a>	<a href="#">Remove Trainee</a>
<a href="#">Saul Barton</a>	<a href="mailto:beryl.rosenbaum@hotmail.com">beryl.rosenbaum@hotmail.com</a>	<a href="#">Award certificate</a>	<a href="#">Remove Trainee</a>
<a href="#">Tatum Bayer</a>	<a href="mailto:elfrieda36@hotmail.com">elfrieda36@hotmail.com</a>	<a href="#">Award certificate</a>	<a href="#">Remove Trainee</a>
<a href="#">Dejah Bednar</a>	<a href="mailto:whane@thompson.com">whane@thompson.com</a>	<a href="#">Award certificate</a>	<a href="#">Remove Trainee</a>
<a href="#">Darby Bednar</a>	<a href="mailto:bosco.lazaro@kreiger.com">bosco.lazaro@kreiger.com</a>	<a href="#">Award certificate</a>	<a href="#">Remove Trainee</a>

### Trainers (0)

[Add Trainer](#)

No trainers added yet.

### Documents

[Browse...](#) No file selected. [Add document](#)

No documents yet.

**Figure 20** Event show page.

A full list of bespoke classes including models, controllers, services, and requests that were produced for this project is given in the appendices section (**Appendix 5**).

## Testing

The software solution was tested using a set of different techniques to check that the system functioned as intended and satisfied the needs of the BEACON organisation.

The verification process focused on ensuring the individual components of functionality within the software solution worked correctly and returned the expected outputs from the given input. Unit tests were run individually to verify system operations, and collectively as part of wider regression testing.

The validation process focused on ensuring the software solution met the functional and quality requirements established at the start of the project. Each requirement was compared to the final working system, and the validity was determined and recorded in a validation table.

## Verification

A test class was created for each controller in the software solution using the Laravel HTTP Tests suite (Laravel 2022b). Each test class contained a series of test cases which ran a test on a corresponding method in the controller. Laravel includes a wrapper for PHPUnit, the standard testing framework for PHP applications (Bergmann, 2007), which provides an outcome of each test and the analysis of failed tests.

A test case sends a request to a specific route and makes assertions about the response. These assertions confirm whether the target method behaves appropriately to a given stimulus. If an assertion fails in each test method, then the test as a whole is considered a failure. Due to the rigidity of the system requiring an authenticated user to make changes, such a user is created at the start of each test case. Dummy data is also created in certain cases to populate the database before testing.

In the test shown in **Figure 21**, an authenticated user is created and, acting as this user, the Events 'create' route is requested. The method then asserts that an appropriate response code (200) is received, and the correct view is returned.

```
public function test_events_create_route_returns_correct_view()
{
    $user = $this->createUser();

    $response = $this->actingAs($user)->get(route('events.create'));
    $response->assertStatus(200);
    $response->assertViewIs('events.create');
}
```

**Figure 21** Test method to check the correct view is returned.

Test classes were chosen to correspond with the controller classes which contained the operations for creating, viewing, updating, and deleting the various entities that existed in the system. It was determined unnecessary to create distinct test classes for the service and request classes as these were considered helper classes used by controllers to apply input validation and business logic.

Test cases were chosen based on a few factors including the expected resources returned from all possible routes, the form validation rules defined in form request classes and business rules defined in service classes.

The test classes were run independently to verify individual controller method functionality, and collectively to verify the complete system functionality. A verification table, a preview of which is shown in **Figure 22**, was produced to record the outcome of the verification process. The table includes a description of the test case, the test class it belongs to and its outcome.

EventTraineeControllerTest	test_event_trainee_store_route_only_stores_trainees_not_already_trainers	Pass
EventTraineeControllerTest	test_event_trainee_store_route_only_stores_trainees_within_course_limit	Pass
EventTraineeControllerTest	test_event_trainee_destroy_route_removes_event_trainee	Pass

**Figure 22** Verification table.

Testing led to the discovery of niche cases where certain objects were allowed to be created, bypassing the rules and validation constraints currently in place on the system. This included allowing events to be created where the end time came before the start time if the start and end dates were the same. The output of that test is shown in **Figure 23**.

```
FAIL Tests\Feature\Http\Controllers\EventControllerTest
  events store route only stores end dates after start

---

* Tests\Feature\Http\Controllers\EventControllerTest > events store route only stores end dates after start
Session is missing expected key [errors].
Failed asserting that false is true.

at C:\Projects\beacon\tests\Feature\Http\Controllers\EventControllerTest.php:136
132▯ 'end_time' => $end->format('H:i')
133▯ ]);
134▯
135▯ $response->assertStatus(302);
136▯ $response->assertSessionHasErrors('end_date');
137▯ }
138▯
139▯ /**
140▯  * Test the events update route.

1 C:\Projects\beacon\vendor\phpunit\phpunit\phpunit:98
  PHPUnit\TextUI\Command::main()

Tests: 1 failed
Time: 1.46s

Start DateTime Object
(
    [date] => 2022-02-05 18:00:44.000000
    [timezone_type] => 3
    [timezone] => UTC
)
End DateTime Object
(
    [date] => 2022-02-05 15:35:15.000000
    [timezone_type] => 3
    [timezone] => UTC
)
```

**Figure 23** A failed test showing how an event was allowed to have a start time later than end time.

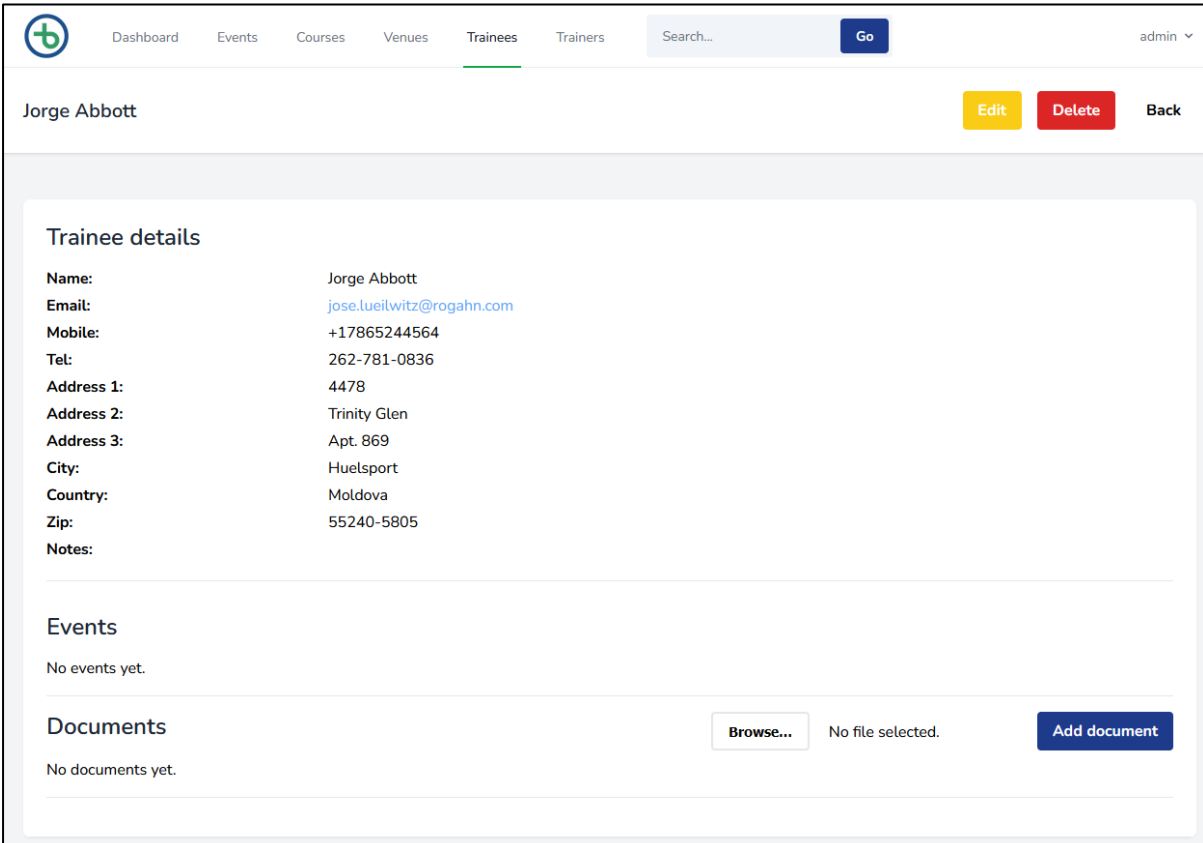
The full table of test classes, case descriptions and outcomes are given in the appendices section (**Appendix 6a**).

## Validation

The system requirements defined during the Requirements phase of the project were compared with the final working software solution to determine the validity of the product. For each requirement, evidence in the form of a verified test, a screenshot of a view of the working system, or other metric was used to confirm satisfaction of the fit criterion and implementation of the requirement in the software solution.

### Functional requirements

The functional requirements related to use cases *UC15 - 18 - viewing objects* - are easily validated using screenshots of the working system. **Figure 24** shows how the fit criterion for FR26 is satisfied with a view of a Trainee. This requirement is further validated by the fact that the verification process asserted that the trainee *show* route worked correctly.



The screenshot displays a web application interface for managing trainees. The top navigation bar includes links for Dashboard, Events, Courses, Venues, Trainees (active), and Trainers. A search bar and a 'Go' button are also present. The user 'admin' is logged in. The main content area shows the details for 'Jorge Abbott'. The details section includes fields for Name, Email, Mobile, Tel, Address 1, Address 2, Address 3, City, Country, Zip, and Notes. Below the details are sections for 'Events' and 'Documents'. The 'Events' section shows 'No events yet.' and the 'Documents' section shows 'No documents yet.' with a 'Browse...' button and an 'Add document' button.

Trainee details	
Name:	Jorge Abbott
Email:	jose.lueilwitz@rogahn.com
Mobile:	+17865244564
Tel:	262-781-0836
Address 1:	4478
Address 2:	Trinity Glen
Address 3:	Apt. 869
City:	Huelsport
Country:	Moldova
Zip:	55240-5805
Notes:	

Events

No events yet.

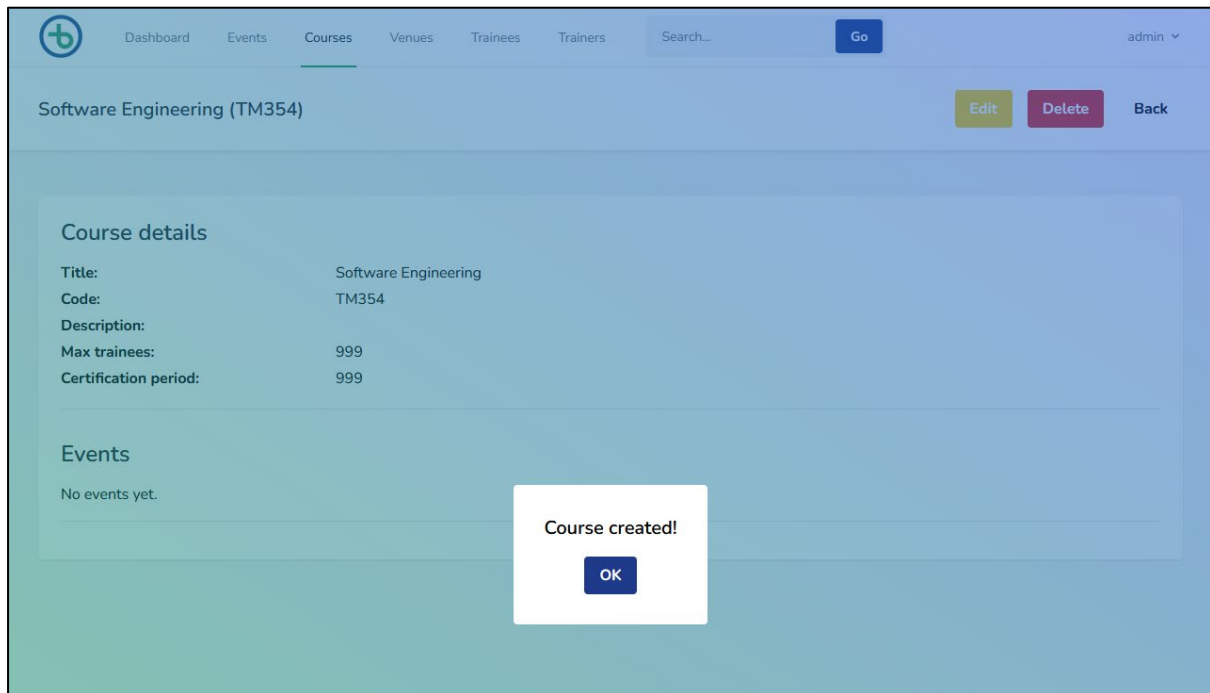
Documents

No documents yet.

Browse... No file selected. Add document

**Figure 24** A view of a trainee will be provided.

The functional requirements related to use cases *UC1, 6, 8, 10, and 11 - the creation of objects* - are also easily validated using screenshots of the working system and confirming correct operation of the system through the previously established verification process. **Figure 25** shows a confirmation that a course has been added to the system, and will be visible on the courses index page, therefore satisfying the fit criterion of FR12.



**Figure 25** A new course has been stored in the system.

The functional requirements related to use cases *UC3 and 4 - the addition of trainees and trainers to events* - can be validated with screenshots of both the events page and trainee page and the outcome of the corresponding verification tests. **Figure 26** shows an event with a list of associated trainees and trainers and the user interface mechanism for adding and removing them.

Software Engineering, Aug 1 - Aug 31 2022

EditDeleteBack

Event details

Course:

Software Engineering

Venue:

Open University

Start Date:

Aug 1, 2022

Start Time:

12:00 pm

End Date:

Aug 31, 2022

End Time:

12:00 pm

Notes:

Trainees (4/999)

Add Trainee

Name	Email	Certificate	
Jorge Abbott	jose.lueilwitz@rogahn.com	Award certificate	Remove Trainee
Dorris Anderson	jaylin62@gmail.com	Award certificate	Remove Trainee
Annabelle Aufderhar	deanna.kemmer@gmail.com	Award certificate	Remove Trainee
Blair Fleming	blair@beacon.com	Award certificate	Remove Trainee

Trainers (2)

Add Trainer

Name	Email	
Savion Bogan	runolfsdottir.herminia@nikolaus.info	Remove Trainer
Camron Cormier	darien54@gmail.com	Remove Trainer

**Figure 26** The system shall enable a user to add a trainee or trainer to an event.

A trainee page with the corresponding event is shown in **Figure 27**.

The screenshot shows a web interface for a trainee profile. At the top, the name "Blair Fleming" is displayed on the left, and "Edit", "Delete", and "Back" buttons are on the right. Below this is a "Trainee details" section with fields for Name, Email, Mobile, Tel, Address 1, Address 2, Address 3, City, Country, Zip, and Notes. The "Email" field is populated with "blair@beacon.com". Below the details is an "Events" section containing a table with columns for Course, Venue, and Date. One event is listed: "Software Engineering" at "Open University" from "Aug 1 - Aug 31 2022".

Course	Venue	Date
Software Engineering	Open University	Aug 1 - Aug 31 2022

**Figure 27** A trainee will be linked to an event.

The functional requirements related to use cases *UC13 and 14 - awarding certificates* – can be validated with screenshots of the system highlighting the mechanism working on an events page. Certificates can be awarded or revoked for each trainee, as shown in **Figure 28**, therefore satisfying the fit criteria of FR22 and 24.

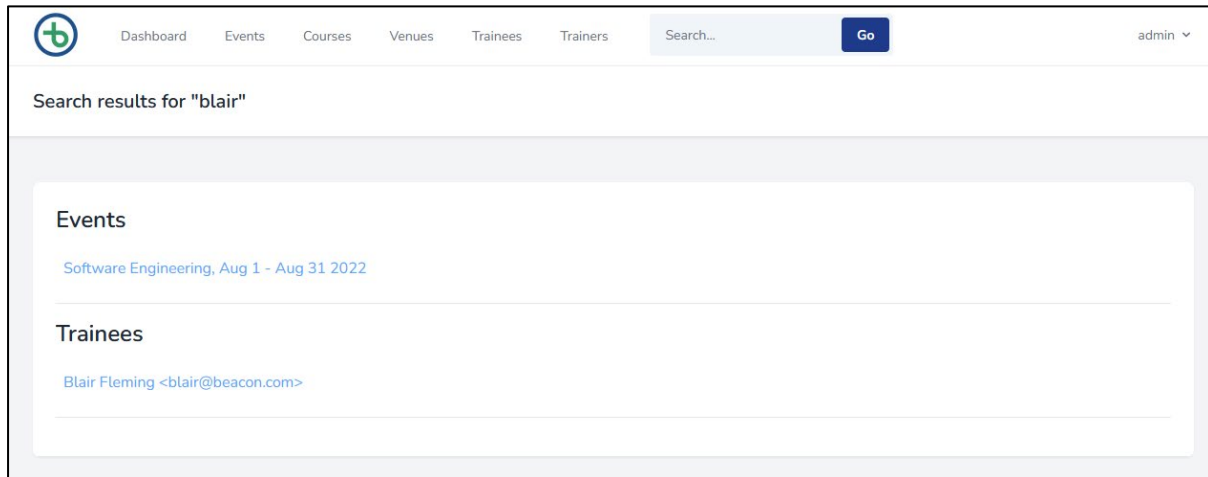
The screenshot shows a "Trainees (4/999)" page with an "Add Trainee" button. Below is a table listing trainees and their certificate status.

Name	Email	Certificate	
Jorge Abbott	jose.lueilwitz@rogahn.com	Award certificate	Remove Trainee
Dorris Anderson	jaylin62@gmail.com	Award certificate	Remove Trainee
Annabelle Aufderhar	deanna.kemmer@gmail.com	Revoke certificate	Remove Trainee
Blair Fleming	blair@beacon.com	Revoke certificate	Remove Trainee

**Figure 28** A certificate will be created for a trainee on an event.



The functional requirements related to the use case *UC19 – searching* – can be validated with screenshots of the system displaying search results for a given search term, as shown in **Figure 29**, therefore satisfying the fit criteria of FR29 and 30.



**Figure 29** Search results.

A decision was made earlier to discard the business rule that required venues to be limited to one event per day, therefore FR5 did not need to be validated.

The full table of functional requirements, fit criteria and outcomes are given in the appendices section (**Appendix 6b**).

### Non-functional requirements

The fit criterion of non-functional requirement NFR1 is dependent on the deployment configuration of the software solution, however the system is optimised to work in a cloud-based environment such as Amazon Web Services and made accessible from the public internet. The chosen deployment environment will also be responsible for the outcome of NFR8 since the security of the system outside of basic user authentication will be deferred to the host.

The solution has incorporated the colour palette and company logo into the design of the user interface and uses BEACON terminology such as events courses and trainees to distinguish the various entities within the system. The fit criterion of requirements NFR2 and NFR9 can therefore be considered met.

Requirements NFR3 and NFR4 will require a period of user adoption testing to fully confirm their fit criterion is met, and similarly NFR10 will require BEACON representatives to spend time with the software, analysing its content and security features to ensure it meets their approval from a legal standpoint.

NFR7 will require the establishment of a maintenance agreement with BEACON to establish the requirements of ongoing monthly support, after which the fit criterion will be met.

NFR5 and NFR6 can use performance metrics to determine whether their fit criterion is met. The performance scenarios shown **Figures 30** and **31** describe how the fit criterion of these requirements can be met.

<i>Part of scenario</i>	<i>Type of value</i>	<i>Actual value</i>
<i>Source</i>	<i>external</i>	<i>web browser, admin user</i>
<i>Stimulus</i>	<i>stochastic event</i>	<i>page request</i>
<i>Artefact</i>	<i>system</i>	<i>events management system</i>
<i>Environment</i>	<i>normal</i>	<i>browser and internet connection working normally</i>
<i>Response</i>	<i>process stimuli</i>	<i>return a page of data</i>
<i>Response measure</i>	<i>latency</i>	<i>page fully loaded within 3 seconds for 99% of requests</i>

**Figure 30** Performance scenario for NFR5.

<i>Part of scenario</i>	<i>Type of value</i>	<i>Actual value</i>
<i>Source</i>	<i>internal</i>	<i>events management system</i>
<i>Stimulus</i>	<i>stochastic event</i>	<i>query to database</i>
<i>Artefact</i>	<i>system</i>	<i>database engine</i>
<i>Environment</i>	<i>normal</i>	<i>database server working normally</i>
<i>Response</i>	<i>process stimuli</i>	<i>add new record to corresponding table</i>
<i>Response measure</i>	<i>latency</i>	<i>new record added to database within 2 seconds of query</i>

**Figure 31** Performance scenario for NFR6.

## Review of current stage of project work

### Current stage of project work

An initial review of the project has explored the problem statement and suggested a possible solution using an established software development approach. The benefits of this solution to the BEACON stakeholders and its broader societal impact have been considered, and recommendations for tackling legal and other challenges have been made.

The project has effectively modelled the domain of the BEACON event management workflow and identified key classes and relationships from which functionality and constraints can be drawn. Interactions with BEACON stakeholders have elicited a strong set of requirements that should influence the development of the software solution, and ultimately decide whether the completed solution meets the needs of the stakeholders.

A design for the software solution has been produced and implemented, using the Laravel framework as a foundation. Appropriate resources have been created including models, views and controllers as well as supporting service and request classes, and a database suitably configured with corresponding tables for each of the main entities. A suite of test classes has been written to test the main controller classes of the system, with suitable test cases to check the most common paths through the system.

A collection of artefacts has been produced throughout the project, including structural models and diagrams, as well as project management resources such as the project journal and schedule. These artefacts have continuously been updated and been subject to several iterations as the project has progressed.

## Remaining project work to be completed

The project has reached a stage where the software solution is ready to be handed over to the BEACON organisation for approval. The approval process should include acceptance testing which will involve an internal review of the functionality and constraints of the system to ensure they meet the functional and quality requirements established and refined throughout the project.

Performance testing of the system should be carried out by BEACON's IT Lead and should include the performance scenarios suggested in the validation section of this report as well as any additional scenarios that would be pertinent to such testing. BEACON should also consider integration testing to make sure the software solution works in conjunction with their existing IT systems, and user testing to identify any usability challenges.

The software itself should be hosted on a publicly accessible web server, using a third-party provider or through BEACON's own external network, and appropriate protective measures such as firewalls and security certificates should be installed. BEACON will also need to consider domain registration and other web hosting costs that will be necessary to make the system live in a full production environment.

BEACON's legal representatives should examine the software solution to ensure it meets the standards of Data Protection Act as well as BEACON's own internal policies on data storage and management. Any faults in the system or breaches of policy should be identified and reported so that appropriate measures can be taken to resolve such issues.

## Recommendations for project continuation

If this project was to continue, additional activities could be included to help support the evolution of the system and improve the quality of the software solution. The testing phase could be extended to include integration and user testing as well as the results of the acceptance testing, and the feedback provided from these activities could inform new iterations of the software solution and identify missing features or constraints.

An additional deployment phase could also be added to the end of the project to support the installation of the software into a live environment. A staging environment would provide an opportunity for testing system performance and security and allow administrative staff to begin training on how to use the new system. A final production environment would allow the software solution to be actively used and fully incorporated into the BEACON workflow.

As the software solution is finalised and the system goes live for BEACON administrators to use, a consideration for ongoing maintenance and support should be made. NFR7 identified the requirement to have monthly updates which would include bug fixes and security patches. The finer details of the required work should be agreed upon and recorded in a maintenance contract and settled internally or with a third-party IT service provider.

## Review of project management

### Project lifecycle model

The lifecycle model I chose for this project was an iterative waterfall. Each stage of the waterfall corresponded to a phase of the software engineering process, from domain modelling, to requirements, analysis, design, implementation and finally testing. The project progressed through these phases in a logical order with a significant amount of work being completed in each phase before moving onto the next.

The output of each phase fed directly into the next and, as such, supported the development of the software solution from the ground up, with a solid foundation that was strengthened as the project went on. The output also fed back into the previous phase to encourage new iterations based on new information or unforeseen issues raised through the development process.

The result of this process was a well-formed software solution with each aspect of the system considered extensively and reviewed and improved in conjunction with the rest of the system. The linear approach of the lifecycle model meant that each phase could be allocated a specific amount of development time with an appropriate amount of overlap between each phase for feedback driven iterations. This was particularly useful for this project due to the time constraint of assignment deadlines.

The project schedule was also very easy to visualise due to the blocks of time allocated to each phase. A Gantt chart was produced which clearly identified the major tasks of the project, their allocated time block, and the overlap between each phase. This aspect of project planning was made simple due to the structured nature of the iterative waterfall.

## Project artefacts

The project was supported by the production of visual artefacts in the form of figures, tables, and charts. These artefacts were vital to the capture and dissemination of information related to the management and development of the project. They were used to analyse the project and assess the viability of different ideas, as well as plan the development process and record the project outcomes.

The most important artefact in this project was the Project Journal which was used to record informal notes regarding the state of the project, possible avenues of exploration to improve the software solution, different development ideas and concepts that could be incorporated into the design and personal feelings and attitudes towards the project. This document was updated regularly and was essential in not only the development of the software solution but also the post-project review.

Other important artefacts in the project included: the Project Schedule which was created as a Gantt chart based on the iterative waterfall model lifecycle, with all the major tasks and assignment deadlines visible; the Glossary which included all the terminology and definitions specific to this project; and a Risk Register which included an assessment of possible risks to project completion and recommendations for mitigation.



## Approach for future projects

Having completed the project, several recommendations for future project management can now be made. A particular hindrance to the swift completion of the project has been the volume of documentation produced, particularly during the requirements phase. Despite the obvious advantage that a collection of resources pertinent to the outcomes of the project can have on the development process, the time spent producing such material could have been used to develop actual working software.

The abundance of use case scenarios, requirements recorded on Volere templates, and testing tables, as evidenced by the large appendices section of this report, provide a lot of fine detail into the process of decision making and outcome generation, however they do not contribute directly to the production of a tangible software solution that the customer BEACON can begin using.

A more suitable approach for a similar software development project would be to follow an agile methodology that encourages working software over comprehensive documentation. Such a process would favour minimal documentation and promote the creation of a product as a priority. This process would also encourage more collaboration between the developer and customer to define requirements in a working system as opposed to a theoretical model.

An agile approach might also put less importance on the linear nature of the waterfall lifecycle model and allow development across all phases of the project simultaneously therefore allowing feedback driven iterations across the entirety of the project spectrum.

## Review of personal development

### What I have learned

This project has given me the opportunity to develop a working software solution from the ground up, using the software development process outlined in TM354. I have taken the concepts and techniques described throughout the module and applied them to a real system based on a problem statement from a customer.

The experience has exposed me to many different practical aspects of software development including identifying system requirements, designing a system based on processes and rules, and implementing a solution using an existing framework. Each of these aspects came with their own set of challenges and I had to make many decisions which I could not have foreseen before starting the project.

The most influential part of the project, in terms of improving my knowledge and understanding of the software development process, was the design and implementation phases. During these phases I effectively put into practice the theoretical aspects of a potential system and observed the results in a working solution. I also had to consider the strengths and limitations of the chosen framework and how to appropriately build the solution within that environment.

The testing phase of the project also contributed significantly to my understanding of the domain particularly the enforcement of constraints. Tests on system operations, such as adding trainees to events, highlighted anomalies which were previously unaccounted for and would require additional constraints to be incorporated into the system. This experience also taught me those certain constraints do not need to be enforced, and in fact often should be relaxed to allow for a more flexible system.

## What challenges I faced

One of the major challenges I faced during this project was identifying and prioritising requirements. It was relatively straightforward to arrange to meet BEACON stakeholders and discuss their priorities for the system, but the difficult part came when trying to establish discrete requirements based on a logical process.

The process I chose for the requirements gathering task involved creating use case scenarios and user stories and extracting individual requirements from these artefacts. This task proved to be time consuming and repetitive, and led to a lot of documentation being produced. I also felt the need to restrict the total number of requirements due to the intense effort required to identify and record a single requirement.

Another challenge I faced was the translation from the logical model produced through domain modelling and analysis to the practical model designed specifically for a target framework. It was important to fully understand the interface of the chosen framework Laravel to effectively implement the different features of the system, and this required extensive research and trial and error to get right.

A final challenge worth noting came in the testing phase when deciding what exactly to test and the extent to which testing should be carried out. Creating test classes and test cases for corresponding controller methods proved to be time consuming, therefore it was essential to focus only on testing the aspects of the system bespoke to this project such as event creation and adding trainees and trainers to an event.

## What I would do differently

If I were to complete this project again, or one similar, I would consider making a few alterations to my software development process. I would firstly take a more agile approach to the production of documentation and focus more time and effort on developing a working solution. System requirements would be kept concise and the process of validating the software solution kept simpler.

I would also consider a development approach that encourages more collaboration with the customer. Techniques that might compliment such an approach might include the use of prototypes which would have been particularly useful during the design of the user interface. This was an aspect of the software development process which I hadn't really considered until late in the project and I had no real guidance on how this should be presented other than a vague description of the BEACON brand guidelines.

I felt my software solution lacked a little bit of cohesion particularly around the required functionality of the system. The core elements of entity creation and modification were included, but I also tacked on some additional functionality, such as search and file upload, without much consideration of the system. These units of functionality were developed in response to a single requirement and the time effort required to realise them was excessive compared to other core features.

In a future project I would consider defining a scope which outlines the planned functionality for at least an initial solution. I would then combine additional functionality which is not essential to the core operation of the system into a second iteration to be arranged and completed later down the line. This way a highly cohesive system can be maintained, and the threat of growing arms and legs kept low!

[10701 words]

## References

Administrate (2022) *Administrate*. Available at: <https://www.getadministrate.com/> (Accessed: April 30 2022)

Ambler, S. and Lines, M. (2020) *Choose your WoW: A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working*. Chicago: Project Management Institute. Available at: [https://library-search.open.ac.uk/permalink/44OPN\\_INST/j6vapu/cdi\\_proquest\\_ebookcentral\\_EBC6177765](https://library-search.open.ac.uk/permalink/44OPN_INST/j6vapu/cdi_proquest_ebookcentral_EBC6177765) (Accessed: August 9 2022)

Barwell, A., Stewart, D., Hoad, R. (2020) *Technology Adoption Hazards*. Available at: <https://www.qinetiq.com/en/insights/technology-adoption-hazards> (Accessed: June 22 2022)

Bergmann, S (2007) *PHPUnit - The PHP Testing Framework*. Available at: <https://phpunit.de/> (Accessed: August 02 2022)

Ceci, L (2022) *Mobile internet usage worldwide - statistics & facts*. Available at: <https://www.statista.com/topics/779/mobile-internet/> (Accessed: August 24 2022)

Data Protection Act (2018) *Data Protection Act 2018*. Available at: <https://www.gov.uk/government/collections/data-protection-act-2018> (Accessed: April 28 2022)

Dolfing, H (2018) *Start Your Project With a Walking Skeleton*. Available at: <https://www.henricodolfing.com/2018/04/start-your-project-with-walking-skeleton.html> (Accessed: April 29 2022)

GitHub (2022a) *Laravel Best Practices*. Available at: <https://github.com/alexeymezenin/laravel-best-practices> (Accessed: June 22 2022)

GitHub (2022b) *Faker*. Available at: <https://github.com/fzaninotto/Faker> (Accessed: July 29 2022)

Grässle, P., Baumann, H. and Baumann, P. (2005) *UML 2.0 in action a project-based tutorial*. 1st edition. Birmingham, U.K: Packt Pub. Available at: [https://library-search.open.ac.uk/permalink/44OPN\\_INST/la9sg5/alma9952608896102316](https://library-search.open.ac.uk/permalink/44OPN_INST/la9sg5/alma9952608896102316) (Accessed: April 30 2022)

Hunt, A. and Thomas, D. (2019) *The Pragmatic Programmer: your journey to mastery, 20th Anniversary Edition*. 2nd Edition. Addison-Wesley Professional. Available at: [https://library-search.open.ac.uk/permalink/44OPN\\_INST/j6vapu/cdi\\_proquest\\_miscellaneous\\_2468684960](https://library-search.open.ac.uk/permalink/44OPN_INST/j6vapu/cdi_proquest_miscellaneous_2468684960) (Accessed: May 2 2022)

Laplante, P.A. (2013) *Requirements engineering for Software and systems*. 2nd edition. Boca Raton, FL: Auerbach Publications, an imprint of Taylor and Francis. Available at: [https://library-search.open.ac.uk/permalink/44OPN\\_INST/la9sg5/alma9952703976402316](https://library-search.open.ac.uk/permalink/44OPN_INST/la9sg5/alma9952703976402316) (Accessed: April 30 2022)

Laravel (2022a) *Documentation*. Available at: <https://laravel.com/docs/9.x> (Accessed: June 29 2022)

Laravel (2022b) *Http Tests*. Available at: <https://laravel.com/docs/9.x/http-tests> (Accessed: June 29 2022)

Laravel (2022c) *Eloquent: Relationships*. Available at: <https://laravel.com/docs/9.x/eloquent-relationships> (Accessed: August 23 2022)

MDN Web Docs (2022a) *MVC*. Available at: <https://developer.mozilla.org/en-US/docs/Glossary/MVC> (Accessed: April 29 2022)

MDN Web Docs (2022b) *REST*. Available at: <https://developer.mozilla.org/en-US/docs/Glossary/REST> (Accessed: August 24 2022)

McReary, J (2019) *Start testing your Laravel applications*. Available at: <https://jasonmccreary.me/articles/start-testing-laravel/> (Accessed: June 22 2022)

Richards, M (2015) *Software Architecture Patterns*. Available at: <https://get.oreilly.com/rs/107-FMS-070/images/Software-Architecture-Patterns.pdf> (Accessed: April 29 2022)

Robertson, J.C. and Robertson, S. (2006) *Mastering the requirements process*. Addison-Wesley Professional. Available at: [https://library-search.open.ac.uk/permalink/44OPN\\_INST/j6vapu/cdi\\_askewsholts\\_vlebooks\\_9780132565431](https://library-search.open.ac.uk/permalink/44OPN_INST/j6vapu/cdi_askewsholts_vlebooks_9780132565431) (Accessed: May 2 2022)

W3C (2018) *Web Content Accessibility Guidelines (WCAG) 2.1*. Available at: <https://www.w3.org/TR/WCAG21/> (Accessed: August 9 2022)

## Appendices

### Appendix 1 – Project Glossary

Term	Definition
<b>Activity diagram</b>	A visualisation of a business process, including the actions and the decisions that take place.
<b>Architecture</b>	The framework for components to combine to produce a working software solution.
<b>Assertion</b>	A statement used as part of a test to verify functionality works correctly.
<b>Business Process</b>	A series of activities that takes place to complete an important function within the system.
<b>Business Rule</b>	A constraint placed on a process that must be adhered to.
<b>Certificate</b>	An award given to a trainee on successful completion of an event.
<b>Controller</b>	A class used to determine which models and/or routes to return given a specific request.
<b>Course</b>	A template for a specific type of training.
<b>Data</b>	Information that represents value to the system and requires storage in the software solution.
<b>Database</b>	The virtual storage system of data in the software solution.
<b>Domain</b>	The conceptual environment in which the system exists, including entities, actors and their relationships.
<b>Event</b>	An occurrence of the delivery of training for a group of individuals on a set date at a set location.
<b>EventService</b>	A dedicated class for managing the complex logic of an Event object.
<b>EventTrainee</b>	A pivot model used to represent the many-to-many relationship between Event and Trainee.
<b>EventTrainer</b>	A pivot model used to represent the many-to-many relationship between Event and Trainer.
<b>GRASP</b>	General Responsibility Assignment Software Patterns
<b>GRASP Expert</b>	A GRASP pattern used to identify the appropriate way to distribute responsibility related to knowledge of objects within a system.
<b>Impact (Risk)</b>	The extent to which the risk will affect the project.
<b>Iteration</b>	A version of a project artefact or deliverable that replaces and improves upon an existing version.

<b><i>Laravel</i></b>	An open-source framework for developing database-driven applications.
<b><i>Lifecycle</i></b>	The project timeline, management, and development from conception to completion.
<b><i>Likelihood (Risk)</i></b>	The chance of the risk occurring in the project.
<b><i>Loop invariant</i></b>	A constraint placed on a structural model to preserve its integrity.
<b><i>Model</i></b>	A system class used to represent an entity in the software solution.
<b><i>MVC</i></b>	Model View Controller
<b><i>Regression test</i></b>	A collection of tests run together to ensure the system continues to function correctly after minor changes.
<b><i>Requirement</i></b>	A feature that the system needs, and that the software solution must satisfy.
<b><i>REST</i></b>	Representational State Transfer
<b><i>Routes</i></b>	Web locations that correspond to different parts of the software solution that the user can interact with.
<b><i>Sequence diagram</i></b>	A visualisation of a series of method calls to access and return data.
<b><i>Severity (Risk)</i></b>	The importance of the risk to the project in terms of impact and likelihood.
<b><i>Test class</i></b>	A dedicated class used to store a series of test methods that correspond to the methods of a corresponding class in the software solution.
<b><i>Test method</i></b>	A function used to test a corresponding method in the software solution.
<b><i>Tracer bullet</i></b>	An implementation approach that encourages the development of a single piece of functionality, the success of which can be expanded and built upon in subsequent iterations.
<b><i>Trainee</i></b>	An individual that receives training.
<b><i>Trainer</i></b>	An individual that provides training.
<b><i>UML</i></b>	Unified Modelling Language; a standardised technique of describing system views.
<b><i>Unit test</i></b>	A test method used to verify a piece of software functionality.
<b><i>Use case</i></b>	A specific task that describes how a user might interact with the system.
<b><i>Use case diagram</i></b>	A visualisation of the different use cases relevant to certain actors and their various alterations based on certain behaviour.
<b><i>User-interface</i></b>	The means of interacting with the system to carry out tasks.



<b><i>Validation (requirements)</i></b>	A means of checking the software solution meets the established system requirements and the needs of the client.
<b><i>Validation (user input)</i></b>	A front-end security method of checking the user input fields of a submitted form conforms to the data requirements of the database before being sent to storage.
<b><i>Venue</i></b>	The location for a training course event to take place.
<b><i>Verification</i></b>	A process of determining whether the developed functionality of the software solution works as intended.
<b><i>View</i></b>	A visual representation of the software solution, provided as a HTML document populated with model data and user interaction elements.
<b><i>Volere template</i></b>	A method of recording requirements and establishing their fit criterion.
<b><i>Walking skeleton</i></b>	A minimum viable interpretation of the proposed software architecture.

## Appendix 2 - Risk Register

Risk	Mitigation	Likelihood	Impact	Severity
<b><i>Assignments and exam prep for other OU modules siphon project time.</i></b>	Appropriate and considered project planning.	Low	Low	Low
<b><i>Functional requirements are complex and difficult to implement effectively.</i></b>	Create simple and distinct requirements with clear definitions and fit criterions.	Low	Medium	Medium
<b><i>Quality requirements lack consensus or are too vague.</i></b>	Clarify requirement definition and purpose and consider only requirements with quantifiable measurements.	Low	Medium	Medium
<b><i>Local copy of codebase is lost or becomes inaccessible.</i></b>	Use a third-party code repository such as GitHub and make regular off-site backups using cloud services.	Low	High	Medium
<b><i>A significant number of iterations are required for each stage of the project lifecycle.</i></b>	Incorporate expectation of multiple iterations into the project plan.	Medium	Low	Medium
<b><i>Project assignments require more time to complete than is initially planned.</i></b>	Extend project phases as necessary and reallocate time from other modules. Consult with Tutor if necessary.	Medium	Medium	Medium
<b><i>Ineffective project planning may lead to repetition of tasks and discovery of unforeseen challenges.</i></b>	Use forward planning and liberal use of Project Journal to predict possible challenges and incorporate possible delays into project plan.	Medium	High	High
<b><i>Help and support from the tutor is required throughout the project.</i></b>	Keep in regular correspondence with Tutor as necessary.	High	Low	Medium
<b><i>Knowledge gap in Laravel implementation and other</i></b>	Use online resources such as the Laravel Best Practices	High	Medium	High

<b><i>supporting technologies leads to delay in project completion.</i></b>	repository to support the development and implementation of the project.			
<b><i>Project progress is not documented effectively and frequently, leading to lost progress and inability to review and reflect accordingly.</i></b>	Make regular backups of solution document and assignments and store versions in different locations. Make liberal use of Project Journal and record date of entries.	High	High	High
<b><i>Stakeholder availability is not consistent resulting in the inability to capture a full set of requirements.</i></b>	Requirements phase can be extended until the necessary quantity of requirements is captured.	Medium	Medium	Medium
<b><i>Stakeholders are concerned for their professional reputation therefore are unwilling to give an honest critique of the organisation and its failings.</i></b>	Contributions from stakeholders can be made anonymous and stakeholders reassured that their input will only be shared appropriately.	Low	Medium	Medium
<b><i>Tutor feedback will be unavailable from TMA03 due to the planned project completion date taking place shortly after the TMA03 submission date.</i></b>	Planned project completion date moved back to provide additional implementation time.	High	Medium	High

## Appendix 3 – Requirements

### Appendix 3a - Use cases

1. UC1 Create Event
2. UC2 Select Venue For Event
3. UC3 Add Trainee To Event
4. UC4 Add Trainer To Event
5. UC5 Update Event Details
6. UC6 Create Course
7. UC7 Update Course Details
8. UC8 Create Trainee
9. UC9 Update Trainee Details
10. UC10 Create Trainer
11. UC11 Create Venue
12. UC12 Update Venue Details
13. UC13 Award Certificate
14. UC14 Revoke Certificate
15. UC15 View Event
16. UC16 View Trainee
17. UC17 View Course
18. UC18 View Venue
19. UC19 Search For Something
20. UC20 Upload Document

## Appendix 3b - Use case scenarios

<b>Identifier and name</b>	UC1 Create Event
<b>Initiator</b>	Administrator
<b>Goal</b>	A new event is created
<b>Precondition</b>	None
<b>Postcondition</b>	An event will have been created for a training course over the specified dates.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"><li>1 The administrator chooses to create a new event.</li><li>2 The administrator selects the desired course.</li><li>3 The administrator selects the start and end dates.</li><li>4 The system creates an event and gives it an identifier.</li><li>5 The system reveals the identifier to the administrator.</li><li>6 The system allows the administrator to add trainers, trainees and a venue to the event.</li></ol>
<b>Extensions</b>	<ol style="list-style-type: none"><li>2.a course does not exist<ol style="list-style-type: none"><li>2.a.1 The system offers alternative courses or the option to create a new course.</li><li>2.a.2 The administrator selects an alternative course or creates a new course.</li></ol></li></ol>

<b>Identifier and name</b>	UC2 Select Venue For Event
<b>Initiator</b>	Administrator
<b>Goal</b>	A venue is added to an event.
<b>Precondition</b>	An event exists.
<b>Postcondition</b>	A venue will be linked to an event.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"><li>1 The administrator chooses to select a venue for an event.</li><li>2 The administrator selects the event.</li><li>3 The administrator selects the desired venue.</li><li>4 The system checks the venue is available for the dates of the event.</li><li>5 The system links to the venue to the event.</li></ol>
<b>Extensions</b>	<ol style="list-style-type: none"><li>3.a venue does not exist<ol style="list-style-type: none"><li>3.a.1 The administrator selects an alternative venue or creates a new venue.</li></ol></li><li>4.a venue unavailable<ol style="list-style-type: none"><li>4.a.1 The administrator selects an alternative venue or changes the dates of the event.</li></ol></li></ol>

<b>Identifier and name</b>	UC3 Add Trainee to Event
<b>Initiator</b>	Administrator
<b>Goal</b>	A trainee is added to an a event.
<b>Precondition</b>	An event exists.
<b>Postcondition</b>	A trainee will be linked to an event.
<b>Assumptions</b>	The expected initiator is a member of the administration team.

**Main success scenario**

- 1 The administrator chooses to add a trainee to an event.
- 2 The administrator selects the event.
- 3 The administrator selects the desired trainee.
- 4 The system checks the trainee is eligible for the event.
- 5 The system links to the trainee to the event.

**Extensions**

- 3.a trainee does not exist
  - 3.a.1 The administrator creates a new trainee.
- 4.a trainee ineligible
  - 4.a.1 The administrator cancels the process.

<b>Identifier and name</b>	UC4 Add Trainer to Event
<b>Initiator</b>	Administrator
<b>Goal</b>	A trainer is added to an a event.
<b>Precondition</b>	An event exists.
<b>Postcondition</b>	A trainer will be linked to an event.
<b>Assumptions</b>	The expected initiator is a member of the administration team.

**Main success scenario**

- 1 The administrator chooses to add a trainer to an event.
- 2 The administrator selects the event.
- 3 The administrator selects the desired trainer.
- 4 The system checks the trainer is eligible for the event.
- 5 The system links to the trainer to the event.

**Extensions**

- 3.a trainer does not exist
  - 3.a.1 The administrator creates a new trainer.
- 4.a trainer ineligible
  - 4.a.1 The administrator cancels the process.

<b>Identifier and name</b>	UC5 Update Event Details
<b>Initiator</b>	Administrator
<b>Goal</b>	An event is updated.
<b>Precondition</b>	An event exists.
<b>Postcondition</b>	The properties of an event will be updated.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	
	<ol style="list-style-type: none"> <li>1 The administrator chooses to update an event.</li> <li>2 The administrator changes the course.</li> <li>3 The administrator changes the start and end dates.</li> <li>4 The system updates the properties of the event.</li> </ol>
<b>Extensions</b>	
	<ol style="list-style-type: none"> <li>2.a trainers are not eligible for course <ol style="list-style-type: none"> <li>2.a.1 The administrator removes the trainers and changes the course.</li> </ol> </li> <li>3.a trainees or trainers unavailable on new dates <ol style="list-style-type: none"> <li>3.a.1 The administrator removes the trainees or trainers and changes the dates.</li> </ol> </li> </ol>

<b>Identifier and name</b>	UC6 Create Course
<b>Initiator</b>	Administrator
<b>Goal</b>	A new course is created.
<b>Precondition</b>	None
<b>Postcondition</b>	A course will have been created.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	
	<ol style="list-style-type: none"> <li>1 The administrator chooses to create a new course.</li> <li>2 The administrator enters the course details.</li> <li>3 The system creates a course and gives it an identifier.</li> <li>4 The system reveals the identifier to the administrator.</li> </ol>
<b>Extensions</b>	
	<ol style="list-style-type: none"> <li>2.a course title exists <ol style="list-style-type: none"> <li>2.a.1 The administrator chooses a different title.</li> </ol> </li> </ol>

<b>Identifier and name</b>	UC7 Update Course Details
<b>Initiator</b>	Administrator
<b>Goal</b>	A course is updated.
<b>Precondition</b>	A course exists.
<b>Postcondition</b>	The properties of a course will be updated.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to update a course.</li> <li>2 The administrator changes the course details.</li> <li>3 The system updates the properties of the course.</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2.a course title exists <ol style="list-style-type: none"> <li>2.a.1 The administrator chooses a different title.</li> </ol> </li> </ol>

<b>Identifier and name</b>	UC8 Create Trainee
<b>Initiator</b>	Administrator
<b>Goal</b>	A new trainee is created
<b>Precondition</b>	None
<b>Postcondition</b>	A trainee will have been created.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to create a new trainee.</li> <li>2 The administrator enters the trainee details.</li> <li>3 The system creates an event and gives it an identifier.</li> <li>4 The system reveals the identifier to the administrator.</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2.a trainee email exists <ol style="list-style-type: none"> <li>2.a.1 The administrator enters a different email address for the trainee.</li> </ol> </li> </ol>



<b>Identifier and name</b>	UC9 Update Trainee Details
<b>Initiator</b>	Administrator
<b>Goal</b>	A trainee is updated.
<b>Precondition</b>	A trainee exists.
<b>Postcondition</b>	The properties of a trainee will be updated.
<b>Assumptions</b>	The expected initiator is a member of the administration team.

**Main success scenario**

- 1 The administrator chooses to update a trainee.
- 2 The administrator changes the trainee details.
- 3 The system updates the properties of the trainee.

**Extensions**

- 2.a trainee email exists
- 2.a.1 The administrator enters a different email address for the trainee.

<b>Identifier and name</b>	UC10 Create Trainer
<b>Initiator</b>	Administrator
<b>Goal</b>	A new trainer is created
<b>Precondition</b>	A trainee exists.
<b>Postcondition</b>	A trainer will have been created and linked to a trainee.
<b>Assumptions</b>	The expected initiator is a member of the administration team.

**Main success scenario**

- 1 The administrator chooses to create a new trainer.
- 2 The administrator selects the desired trainee.
- 3 The system creates a trainer.
- 4 The system links the trainer to a trainee.

<b>Identifier and name</b>	UC11 Create Venue
<b>Initiator</b>	Administrator
<b>Goal</b>	A new venue is created.
<b>Precondition</b>	None
<b>Postcondition</b>	A venue will have been created.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to create a new venue.</li> <li>2 The administrator enters the venue details.</li> <li>3 The system creates a venue and gives it an identifier.</li> <li>4 The system reveals the identifier to the administrator.</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2.a venue name exists <ol style="list-style-type: none"> <li>2.a.1 The administrator chooses a different name.</li> </ol> </li> </ol>

<b>Identifier and name</b>	UC12 Update Venue Details
<b>Initiator</b>	Administrator
<b>Goal</b>	A venue is updated.
<b>Precondition</b>	A venue exists.
<b>Postcondition</b>	The properties of a venue will be updated.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to update a venue.</li> <li>2 The administrator changes the venue details.</li> <li>3 The system updates the properties of the venue.</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2.a venue name exists <ol style="list-style-type: none"> <li>2.a.1 The administrator chooses a different name.</li> </ol> </li> </ol>

<b>Identifier and name</b>	UC13 Award Certificate
<b>Initiator</b>	Administrator
<b>Goal</b>	A new certificate is created
<b>Precondition</b>	An trainee exists and is linked to an event.
<b>Postcondition</b>	A certificate will have been created and linked to a trainee and event.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to award a certificate.</li> <li>2 The administrator selects the desired event.</li> <li>3 The administrator selects the desired trainee.</li> <li>4 The system creates a certificate.</li> <li>5 The system links the certificate to the trainee and event.</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2.a event not ended</li> <li>2.a.1 The administrator cancels the process.</li> </ol>

<b>Identifier and name</b>	UC14 Revoke Certificate
<b>Initiator</b>	Administrator
<b>Goal</b>	A certificate is destroyed
<b>Precondition</b>	A certificate exists and is linked to a trainee and event.
<b>Postcondition</b>	A certificate will have been destroyed and links to trainee and event removed.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to revoke a certificate.</li> <li>2 The administrator selects the desired event.</li> <li>3 The administrator selects the desired trainee.</li> <li>4 The system removes the links to the trainee and event.</li> <li>5 The system removes the certificate.</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>3.a trainer requires certificate for other event</li> <li>3.a.1 The administrator removes the trainer from the other event before revoking certificate.</li> </ol>

<b>Identifier and name</b>	UC15 View Event
<b>Initiator</b>	Administrator
<b>Goal</b>	An event is displayed.
<b>Precondition</b>	An event exists.
<b>Postcondition</b>	A view of the event will have been returned.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to view an event.</li> <li>2 The administrator selects the desired event.</li> <li>3 The system retrieves the event data.</li> <li>4 The system retrieves associated trainees, trainers, course and venue data.</li> <li>5 The system returns a view with the retrieved data.</li> </ol>

<b>Identifier and name</b>	UC16 View Trainee
<b>Initiator</b>	Administrator
<b>Goal</b>	A trainee is displayed.
<b>Precondition</b>	A trainee exists.
<b>Postcondition</b>	A view of the trainee will have been returned.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to view a trainee.</li> <li>2 The administrator selects the desired trainee.</li> <li>3 The system retrieves the trainee data.</li> <li>4 The system retrieves associated trainer and events data.</li> <li>5 The system returns a view with the retrieved data.</li> </ol>

<b>Identifier and name</b>	UC17 View Course
<b>Initiator</b>	Administrator
<b>Goal</b>	A course is displayed.
<b>Precondition</b>	A course exists.
<b>Postcondition</b>	A view of the course will have been returned.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to view a course.</li> <li>2 The administrator selects the desired course.</li> <li>3 The system retrieves the course data.</li> <li>4 The system retrieves associated events data.</li> <li>5 The system returns a view with the retrieved data.</li> </ol>

<b>Identifier and name</b>	UC18 View Venue
<b>Initiator</b>	Administrator
<b>Goal</b>	A venue is displayed.
<b>Precondition</b>	A venue exists.
<b>Postcondition</b>	A view of the venue will have been returned.
<b>Assumptions</b>	The expected initiator is a member of the administration team.

**Main success scenario**

- 1 The administrator chooses to view a venue.
- 2 The administrator selects the desired venue.
- 3 The system retrieves the venue data.
- 4 The system retrieves associated events data.
- 5 The system returns a view with the retrieved data.

<b>Identifier and name</b>	UC19 Search For Something
<b>Initiator</b>	Administrator
<b>Goal</b>	A list of search results is displayed.
<b>Precondition</b>	None
<b>Postcondition</b>	A collection of entities related to the search term will have been returned.
<b>Assumptions</b>	The expected initiator is a member of the administration team.

**Main success scenario**

- 1 The administrator chooses to search for something.
- 2 The administrator enters a search term into the search box.
- 3 The system retrieves data related to the search term.
- 4 The system returns a view with the retrieved data.

**Extensions**

- 3.a no results
  - 3.a.1 The system suggests alternative search terms.

<b>Identifier and name</b>	UC20 Upload Document
<b>Initiator</b>	Administrator
<b>Goal</b>	A document is uploaded to the system.
<b>Precondition</b>	An event or trainee exists.
<b>Postcondition</b>	A document will be saved to system storage and linked to an event or trainee.
<b>Assumptions</b>	The expected initiator is a member of the administration team.
<b>Main success scenario</b>	<ol style="list-style-type: none"> <li>1 The administrator chooses to upload a document.</li> <li>2 The administrator selects the desired event or trainee.</li> <li>3 The administrator selects a document from local storage.</li> <li>4 The system links the document to the event or trainee.</li> <li>5 The system saves the document to system storage.</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>3.a file type invalid <ol style="list-style-type: none"> <li>3.a.1 The administrator selects an alternative document.</li> </ol> </li> <li>3.b file size too large <ol style="list-style-type: none"> <li>3.b.1 The administrator selects an alternative document.</li> </ol> </li> </ol>

## Appendix 3c - User stories

1. US1 - As an Administrator I want the system to be available remotely so that I can work from home or on an event.
2. US2 - As an Administrator I want the system to have company branding so that it feels like it belongs to the team.
3. US3 - As an Administrator I want the system to be user friendly so that I don't have to spend much time learning how to use it.
4. US4 - As an Administrator I want the system to show all relevant event information on one page so that I don't have to open new pages to find related info.
5. US5 - As an Administrator I want the system to find events and trainees fast so that I don't have to waste time waiting for the system to respond.
6. US6 - As an Administrator I want the system to streamline the event creation process so that I can get more work done in a shorter amount of time.
7. US7 - As an Administrator I want the system to be updated regularly so that bugs do not affect the workflow.
8. US8 - As an Administrator I want the system to be safe from unauthorised access so that data is not stolen or deleted.
9. US9 - As an Administrator I want the system to use language and terms relevant to BEACON so that there is a common understanding of the system.
10. US10 - As an Administrator I want the system to adequately protect personal data so that we reduce the risk of violating the data protection act.

## Appendix 4 - Volere Templates

### Appendix 4a - Functional requirements

Requirement #: FR1	Requirement Type: Functional	UC #: UC1
Description: The system shall enable the user to add an event.		
Rationale: Events must be added to the system so they can be managed effectively.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A new event has been stored in the system.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR2	Requirement Type: Functional	UC #: UC1
Description: The system shall enable the user to select a course for an event.		
Rationale: Events must have an associated course.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A course will be linked to an event.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR3	Requirement Type: Functional	UC #: UC1
Description: The system shall enable the user to select a date range for an event.		
Rationale: Events must have a start and end date.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: An event will have a set start and end date.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR4	Requirement Type: Functional	UC #: UC2
Description: The system shall enable the user to select a venue for an event.		
Rationale: Events must take place at a specific venue.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A venue will be linked to an event.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		



Requirement #: FR5	Requirement Type: Functional	UC #: UC2
Description: The system shall check whether a venue is available on a given date.		
Rationale: Venues cannot be used for more than one event on the same day.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the venue's availability will be provided.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

<b>Requirement #:</b> FR6	<b>Requirement Type:</b> Functional	<b>UC #:</b> UC3
<b>Description:</b> The system shall enable the user to add a trainee to an event.		
<b>Rationale:</b> Events must have trainees.		
<b>Originator:</b> BEACON IT Lead, Donald		
<b>Fit Criterion:</b> A trainee will be linked to an event.		
<b>Customer Satisfaction:</b> 5	<b>Customer Dissatisfaction:</b> 5	
<b>Priority:</b> High	<b>Conflicts:</b> None	
<b>Supporting Materials:</b> None		
<b>History:</b> Created Apr 14 2022		

Requirement #: FR7	Requirement Type: Functional	UC #: UC3
Description: The system shall check the eligibility of a trainee for an event.		
Rationale: A trainee cannot be on multiple events simultaneously as a trainee or trainer.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the trainee's eligibility will be provided.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

<b>Requirement #:</b> FR8	<b>Requirement Type:</b> Functional	<b>UC #:</b> UC4
<b>Description:</b> The system shall enable the user to add a trainer to an event.		
<b>Rationale:</b> Events must have trainers.		
<b>Originator:</b> BEACON IT Lead, Donald		
<b>Fit Criterion:</b> A trainer will be linked to an event.		
<b>Customer Satisfaction:</b> 5	<b>Customer Dissatisfaction:</b> 5	
<b>Priority:</b> High	<b>Conflicts:</b> None	
<b>Supporting Materials:</b> None		
<b>History:</b> Created Apr 14 2022		

Requirement #: FR9	Requirement Type: Functional	UC #: UC4
Description: The system shall check the eligibility of a trainer for an event.		
Rationale: A trainer cannot be on multiple events simultaneously as a trainee or trainer and must have a valid certificate for the associated course.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the trainer's eligibility will be provided.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR10	Requirement Type: Functional	UC #: UC5
Description: The system shall enable the user to update an event.		
Rationale: Events must be updated to reflect changes to the event or fix errors.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: An existing event will be updated in the system.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR11	Requirement Type: Functional	UC #: UC5
Description: The system shall check the updated event dates are suitable for the registered trainers and trainees.		
Rationale: Events cannot be changed to dates where trainers or trainees are unavailable.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the suitability of the updated dates will be provided.		
Customer Satisfaction: 3	Customer Dissatisfaction: 3	
Priority: Medium	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR12	Requirement Type: Functional	UC #: UC6
Description: The system shall enable the user to add a new course.		
Rationale: Courses must exist on the system so that events can be created.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A new course has been stored in the system.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR13	Requirement Type: Functional	UC #: UC6
Description: The system shall check the course details do not exist already.		
Rationale: Duplicate courses cannot exist on the system.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the unique course details is provided.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR14	Requirement Type: Functional	UC #: UC7
Description: The system shall enable the user to update a course.		
Rationale: Courses must be updated to reflect changes to the course or fix errors.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: An existing course will be updated in the system.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR15	Requirement Type: Functional	UC #: UC8
Description: The system shall enable the user to create a trainee.		
Rationale: Trainees must be created so they can be added to events or made trainers.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A new trainee has been stored in the system.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR16	Requirement Type: Functional	UC #: UC8
Description: The system shall check the trainee details do not exist already.		
Rationale: Duplicate trainees cannot exist on the system.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the unique trainee details is provided.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR17	Requirement Type: Functional	UC #: UC9
Description: The system shall enable the user to update a trainee.		
Rationale: Trainees must be updated to reflect changes to the trainee or fix errors.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: An existing trainee will be updated in the system.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR18	Requirement Type: Functional	UC #: UC10
Description: The system shall enable the user to create a trainer from a trainee.		
Rationale: Trainers must be created so they can be added to events.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A new trainer has been stored in the system and linked to an existing trainee.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR19	Requirement Type: Functional	UC #: UC11
Description: The system shall enable the user to add a new venue.		
Rationale: Venues must be created so they can be added to events.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A new venue has been stored in the system.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR20	Requirement Type: Functional	UC #: UC11
Description: The system shall check the venue details do not exist already.		
Rationale: Duplicate venues cannot exist on the system.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the unique venue details is provided.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR21	Requirement Type: Functional	UC #: UC12
Description: The system shall enable the user to update a venue.		
Rationale: Venues must be updated to reflect changes to the venue or fix errors.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: An existing venue will be updated in the system.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR22	Requirement Type: Functional	UC #: UC13
Description: The system shall enable the user to award a certificate to a trainee on an event.		
Rationale: Trainees must be awarded certificates to mark their achievement and allow them to become tainers for the course.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A certificate will be created for a trainee on an event.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR23	Requirement Type: Functional	UC #: UC13
Description: The system shall check the event has ended before creating a certificate.		
Rationale: A trainee cannot be awarded a certificate if the event has not ended yet.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the event having ended will be provided.		
Customer Satisfaction: 4	Customer Dissatisfaction: 4	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR24	Requirement Type: Functional	UC #: UC14
Description: The system shall enable the user to revoke a certificate from a trainee on an event.		
Rationale: Certificates must be revokable if an error was made.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A certificate will be removed from the system.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		



Requirement #: FR25	Requirement Type: Functional	UC #: UC15
Description: The system shall enable the user to view an event.		
Rationale: Events must be visible to users so they can be managed effectively.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A view of an event will be provided.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR26	Requirement Type: Functional	UC #: UC16
Description: The system shall enable the user to view a trainee.		
Rationale: Trainees must be visible to users so they can be managed effectively.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A view of a trainee will be provided.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR27	Requirement Type: Functional	UC #: UC17
Description: The system shall enable the user to view a course.		
Rationale: Courses must be visible to users so they can be managed effectively.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A view of a course will be provided.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR28	Requirement Type: Functional	UC #: UC18
Description: The system shall enable the user to view a venue.		
Rationale: Venues must be visible to users so they can be managed effectively.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A view of a venue will be provided.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR29	Requirement Type: Functional	UC #: UC19
Description: The system shall provide a search mechanism to find stored content.		
Rationale: Users require a method of finding information related to a search term.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A list of search results will be provided.		
Customer Satisfaction: 3	Customer Dissatisfaction: 3	
Priority: Low	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR30	Requirement Type: Functional	UC #: UC19
Description: The system shall provide alternative search results for an unsuccessful query.		
Rationale: Users may be using ineffective search terms for their query.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A list of alternative search results will be provided.		
Customer Satisfaction: 3	Customer Dissatisfaction: 3	
Priority: Low	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR31	Requirement Type: Functional	UC #: UC20
Description: The system shall enable the user to upload a document.		
Rationale: Documents may contain additional information about events or trainees that is not currently captured by the system.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A document is stored on the system and linked to an event or trainee.		
Customer Satisfaction: 3	Customer Dissatisfaction: 3	
Priority: Low	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

Requirement #: FR32	Requirement Type: Functional	UC #: UC20
Description: The system shall prevent users from uploading documents that are too large or of incompatible file type.		
Rationale: The file storage system has a limited capacity and only specific file types and sizes can be accepted.		
Originator: BEACON IT Lead, Donald		
Fit Criterion: A confirmation of the documents suitability for storage is provided.		
Customer Satisfaction: 3	Customer Dissatisfaction: 3	
Priority: Low	Conflicts: None	
Supporting Materials: None		
History: Created Apr 14 2022		

<b>Requirement #:</b> FR33	<b>Requirement Type:</b> Functional	<b>UC #:</b> UC3
<b>Description:</b> The system shall prevent trainees from being added to an event that has reached its trainee capacity limit.		
<b>Rationale:</b> Events should not have more trainees than the corresponding course can support.		
<b>Originator:</b> BEACON IT Lead, Donald		
<b>Fit Criterion:</b> An error message that prevents the event going over capacity should be displayed.		
<b>Customer Satisfaction:</b> 3	<b>Customer Dissatisfaction:</b> 3	
<b>Priority:</b> Low	<b>Conflicts:</b> None	
<b>Supporting Materials:</b> None		
<b>History:</b> Created July 26 2022		

<b>Requirement #:</b> FR34	<b>Requirement Type:</b> Functional	<b>UC #:</b> UC3
<b>Description:</b> The system shall prevent trainees from being added to an event where they already exist as a trainer.		
<b>Rationale:</b> Events should not have trainees who are also trainers on the same event.		
<b>Originator:</b> BEACON IT Lead, Donald		
<b>Fit Criterion:</b> An error message that prevents the trainee from being added should be displayed.		
<b>Customer Satisfaction:</b> 3	<b>Customer Dissatisfaction:</b> 3	
<b>Priority:</b> Low	<b>Conflicts:</b> None	
<b>Supporting Materials:</b> None		
<b>History:</b> Created July 26 2022		

<b>Requirement #:</b> FR35	<b>Requirement Type:</b> Functional	<b>UC #:</b> UC4
<b>Description:</b> The system shall prevent trainers from being added to an event where they already exist as a trainee.		
<b>Rationale:</b> Events should not have trainers who are also trainees on the same event.		
<b>Originator:</b> BEACON IT Lead, Donald		
<b>Fit Criterion:</b> An error message that prevents the trainer from being added should be displayed.		
<b>Customer Satisfaction:</b> 3	<b>Customer Dissatisfaction:</b> 3	
<b>Priority:</b> Low	<b>Conflicts:</b> None	
<b>Supporting Materials:</b> None		
<b>History:</b> Created July 26 2022		

<b>Requirement #:</b> FR36	<b>Requirement Type:</b> Functional	<b>UC #:</b> UC4
<b>Description:</b> The system shall prevent trainers from being added to an event where they do not have a valid certificate for the corresponding course.		
<b>Rationale:</b> Trainers should be fully certificated before they can train on an event.		
<b>Originator:</b> BEACON IT Lead, Donald		
<b>Fit Criterion:</b> An error message that prevents the trainer from being added should be displayed.		
<b>Customer Satisfaction:</b> 3	<b>Customer Dissatisfaction:</b> 3	
<b>Priority:</b> Low	<b>Conflicts:</b> None	
<b>Supporting Materials:</b> None		
<b>History:</b> Created July 26 2022		



## Appendix 4b - Non-functional requirements

Requirement #: NFR1	Requirement Type: Non-functional (Operational and environmental)	UC #: US1
Description: The system shall be accessible by the user remotely.		
Rationale: Users should be able to use the system from home and out of office.		
Originator: BEACON administrative team		
Fit Criterion: The system can be accessed from outside the BEACON internal network.		
Customer Satisfaction: 5	Customer Dissatisfaction: 3	
Priority: Medium	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

Requirement #: NFR2	Requirement Type: Non-functional (Look-and-feel)	UC #: US2
Description: The system shall conform to the BEACON corporate branding guidelines.		
Rationale: The system should look like it belongs to BEACON users and is a part of the organisation's workflow.		
Originator: BEACON administrative team		
Fit Criterion: The system shares the same colours and logo as other organisational assets		
Customer Satisfaction: 5	Customer Dissatisfaction: 2	
Priority: Low	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

Requirement #: NFR3	Requirement Type: Non-functional (Usability and humanity)	UC #: US3
Description: The system shall be easy for users to interact with.		
Rationale: Users should not have to spend significant time learning and interacting with a system that is supposed to reduce their expended effort.		
Originator: BEACON administrative team		
Fit Criterion: All members of the admin team should be able to learn how to use the system in under 1 hour, and create an event in less than 1 minute.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

<b>Requirement #:</b> NFR4	<b>Requirement Type:</b> Non-functional (Usability and humanity)	<b>UC #:</b> US4
<b>Description:</b> The system shall make it easy for users to find information associated with the entity they are currently viewing.		
<b>Rationale:</b> Users should not have to navigate through the system and open separate pages to find related information.		
<b>Originator:</b> BEACON administrative team		
<b>Fit Criterion:</b> A member of the admin team can find an associated record in less than 5 seconds.		
<b>Customer Satisfaction:</b> 3	<b>Customer Dissatisfaction:</b> 3	
<b>Priority:</b> Medium	<b>Conflicts:</b> None	
<b>Supporting Materials:</b> An example of an associated record would be a trainee on an event.		
<b>History:</b> Created May 4 2022		

Requirement #: NFR5	Requirement Type: Non-functional (Performance)	UC #: US5
Description: The system shall be quick to respond to user requests.		
Rationale: The system should be able to handle all requests and respond quickly.		
Originator: BEACON administrative team		
Fit Criterion: The system shall provide a response to 99% of user requests within 3 seconds.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: Medium	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

Requirement #: NFR6	Requirement Type: Non-functional (Performance)	UC #: US6
Description: The system shall create new records with minimum delay.		
Rationale: The system should create new records rapidly to allow users to make efficient use of the system.		
Originator: BEACON administrative team		
Fit Criterion: The system shall create new records in less than 2 seconds.		
Customer Satisfaction: 5	Customer Dissatisfaction: 3	
Priority: Medium	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

Requirement #: NFR7	Requirement Type: Non-functional (Maintainability and support)	UC #: US7
Description: The system shall be updated regularly.		
Rationale: The system should have bug fixes and security patches applied to keep the system safe, secure and running efficiently.		
Originator: BEACON administrative team		
Fit Criterion: The system shall be updated once per month.		
Customer Satisfaction: 5	Customer Dissatisfaction: 3	
Priority: Low	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

Requirement #: NFR8	Requirement Type: Non-functional (Security)	UC #: US8
Description: The system shall prevent unauthorised access to sensitive data.		
Rationale: The system stores sensitive data such as trainee contact information therefore should only be accessible by authorised BEACON personnel.		
Originator: BEACON administrative team		
Fit Criterion: The system shall be protected from unauthorised access 100% of the time.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

Requirement #: NFR9	Requirement Type: Non-functional (Cultural)	UC #: US9
Description: The system shall use terminology familiar to BEACON stakeholders.		
Rationale: Entities with different names from their BEACON counterparts can cause confusion and miscommunication.		
Originator: BEACON administrative team		
Fit Criterion: The system shall provide a like-for-like representation for 9 out of 10 BEACON terms.		
Customer Satisfaction: 5	Customer Dissatisfaction: 3	
Priority: Low	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

Requirement #: NFR10	Requirement Type: Non-functional (Legal)	UC #: US10
Description: The system shall operate in accordance with British data protection law.		
Rationale: Breaches of the UK Data Protection Act can lead to significant legal repercussions.		
Originator: BEACON administrative team		
Fit Criterion: The system shall be approved by BEACON's legal representative.		
Customer Satisfaction: 5	Customer Dissatisfaction: 5	
Priority: High	Conflicts: None	
Supporting Materials: None		
History: Created May 4 2022		

## Appendix 5 - Implementation classes

### Models

- Certificate
- Course
- Document
- Event
- Trainee
- Trainer
- Venue

### Controllers

- CertificateController
- CourseController
- DocumentController
- EventController
- EventTraineeController
- EventTrainerController
- TraineeController
- TrainerController
- VenueController

### Services

- DocumentService
- EventService

### Requests

- CertificateStoreRequest
- CourseStoreRequest
- CourseUpdateRequest
- DocumentStoreRequest
- EventStoreRequest
- EventTraineeStoreRequest
- EventTrainerStoreRequest
- EventUpdateRequest
- TraineeStoreRequest
- TraineeUpdateRequest
- TrainerStoreRequest
- VenueStoreRequest
- VenueUpdateRequest

## Appendix 6 - Testing

### Appendix 6a - Verification Table

Test Class	Test Case	Outcome
<b>CertificateControllerTest</b>	test_certificates_store_route_stores_certificate	Pass
<b>CertificateControllerTest</b>	test_certificates_destroy_route_removes_certificate	Pass
<b>CourseControllerTest</b>	test_courses_index_route_returns_index_view	Pass
<b>CourseControllerTest</b>	test_courses_create_route_returns_create_view	Pass
<b>CourseControllerTest</b>	test_courses_show_route_returns_show_view	Pass
<b>CourseControllerTest</b>	test_courses_edit_route_returns_edit_view	Pass
<b>CourseControllerTest</b>	test_courses_store_route_stores_course	Pass
<b>CourseControllerTest</b>	test_courses_store_route_only_stores_unique_titles	Pass
<b>CourseControllerTest</b>	test_courses_store_route_only_stores_unique_codes	Pass
<b>CourseControllerTest</b>	test_courses_update_route_updates_course	Pass
<b>CourseControllerTest</b>	test_courses_update_route_only_updates_unique_titles	Pass
<b>CourseControllerTest</b>	test_courses_update_route_only_updates_unique_codes	Pass
<b>CourseControllerTest</b>	test_courses_destroy_route_removes_course	Pass
<b>CourseControllerTest</b>	test_courses_destroy_route_only_removes_courses_without_events	Pass
<b>DocumentControllerTest</b>	test_documents_store_route_stores_document_for_event	Pass
<b>DocumentControllerTest</b>	test_documents_store_route_stores_document_for_trainee	Pass

<b>DocumentControllerTest</b>	test_documents_store_route_only_stores_unique_document_titles	Pass
<b>DocumentControllerTest</b>	test_documents_destroy_route_removes_document	Fail*
<b>EventControllerTest</b>	test_events_index_route_returns_index_view	Pass
<b>EventControllerTest</b>	test_events_create_route_returns_create_view	Pass
<b>EventControllerTest</b>	test_events_show_route_returns_show_view	Pass
<b>EventControllerTest</b>	test_events_edit_route_returns_edit_view	Pass
<b>EventControllerTest</b>	test_events_store_route_stores_event	Pass
<b>EventControllerTest</b>	test_events_store_route_only_stores_end_dates_after_start	Fail**
<b>EventControllerTest</b>	test_events_update_route_updates_event	Pass
<b>EventControllerTest</b>	test_events_update_route_only_updates_end_dates_after_start	Fail**
<b>EventControllerTest</b>	test_events_destroy_route_removes_event	Pass
<b>EventTraineeControllerTest</b>	test_event_trainee_create_route_returns_create_view	Pass
<b>EventTraineeControllerTest</b>	test_event_trainee_store_route_stores_event_trainee	Pass
<b>EventTraineeControllerTest</b>	test_event_trainee_store_route_only_stores_unique_trainees	Pass
<b>EventTraineeControllerTest</b>	test_event_trainee_store_route_only_stores_trainees_not_already_trainers	Pass
<b>EventTraineeControllerTest</b>	test_event_trainee_store_route_only_stores_trainees_within_course_limit	Pass
<b>EventTraineeControllerTest</b>	test_event_trainee_destroy_route_removes_event_trainee	Pass
<b>EventTraineeControllerTest</b>	test_event_trainee_destroy_route_does_not_remove_event_trainee_not_on_event	Pass

EventTrainerControllerTest	test_event_trainer_create_route_returns_create_view	Pass
EventTrainerControllerTest	test_event_trainer_store_route_stores_event_trainer	Pass
EventTrainerControllerTest	test_event_trainer_store_route_only_stores_unique_trainers	Pass
EventTrainerControllerTest	test_event_trainer_store_route_only_stores_trainees_not_already_trainees	Pass
EventTrainerControllerTest	test_event_trainer_store_route_only_stores_trainees_with_certification	Pass
EventTrainerControllerTest	test_event_trainer_store_route_only_stores_fully_certified_trainers	Pass
EventTrainerControllerTest	test_event_trainer_destroy_route_removes_event_trainer	Pass
EventTrainerControllerTest	test_event_trainer_destroy_route_does_not_remove_event_trainer_not_on_event	Pass
TraineeControllerTest	test_trainees_index_route_returns_index_view	Pass
TraineeControllerTest	test_trainees_create_route_returns_create_view	Pass
TraineeControllerTest	test_trainees_show_route_returns_show_view	Pass
TraineeControllerTest	test_trainees_edit_route_returns_edit_view	Pass
TraineeControllerTest	test_trainees_store_route_stores_trainee	Pass
TraineeControllerTest	test_trainees_store_route_only_stores_unique_emails	Pass
TraineeControllerTest	test_trainees_update_route_updates_trainee	Pass
TraineeControllerTest	test_trainees_update_route_only_updates_unique_emails	Pass
TraineeControllerTest	test_trainees_destroy_route_removes_trainee	Pass
TrainerControllerTest	test_trainers_index_route_returns_index_view	Pass



TrainerControllerTest	test_trainers_create_route_returns_create_view	Pass
TrainerControllerTest	test_trainers_show_route_returns_trainee_show_view	Pass
TrainerControllerTest	test_trainers_edit_route_returns_trainee_edit_view	Pass
TrainerControllerTest	test_trainers_store_route_stores_trainer	Pass
TrainerControllerTest	test_trainers_store_route_only_stores_unique_trainees	Pass
TrainerControllerTest	test_trainers_destroy_route_removes_trainer	Pass
VenueControllerTest	test_venues_index_route_returns_index_view	Pass
VenueControllerTest	test_venues_create_route_returns_create_view	Pass
VenueControllerTest	test_venues_show_route_returns_show_view	Pass
VenueControllerTest	test_venues_edit_route_returns_edit_view	Pass
VenueControllerTest	test_venues_store_route_stores_venue	Pass
VenueControllerTest	test_venues_store_route_only_stores_unique_names	Pass
VenueControllerTest	test_venues_update_route_updates_venue	Pass
VenueControllerTest	test_venues_update_route_only_updates_unique_names	Pass
VenueControllerTest	test_venues_destroy_route_removes_venue	Pass

*\*Failed test led to the discovery that documents, although removed from the database, were not being removed from file storage upon deletion.*

*\*\*Failed tests led to the discovery of the situation where events starting and finishing on the same day can have an end time occurring before the start time.*

Appendix 6b - Validation Table (Functional requirements)

Requirement	Fit Criterion	Satisfied?
<b>FR1</b>	A new event has been stored in the system.	Yes
<b>FR2</b>	A course will be linked to an event.	Yes
<b>FR3</b>	An event will have a set start and end date.	Yes
<b>FR4</b>	A venue will be linked to an event.	Yes
<b>FR5</b>	A confirmation of the venue's availability will be provided.	No*
<b>FR6</b>	An event will have a list of trainees.	Yes
<b>FR7</b>	A confirmation of the trainee's eligibility will be provided.	Yes
<b>FR8</b>	An event will have a list of trainers.	Yes
<b>FR9</b>	A confirmation of the trainer's eligibility will be provided.	Yes
<b>FR10</b>	An existing event will be updated in the system.	Yes
<b>FR11</b>	A confirmation of the suitability of the updated dates will be provided.	Yes
<b>FR12</b>	A new course has been stored in the system.	Yes
<b>FR13</b>	A confirmation of the unique course details is provided.	Yes
<b>FR14</b>	An existing course will be updated in the system.	Yes
<b>FR15</b>	A new trainee has been stored in the system.	Yes
<b>FR16</b>	A confirmation of the unique trainee details is provided.	Yes
<b>FR17</b>	An existing trainee will be updated in the system.	Yes

<b>FR18</b>	A new trainer has been stored in the system and linked to an existing trainee.	Yes
<b>FR19</b>	A new venue has been stored in the system.	Yes
<b>FR20</b>	A confirmation of the unique venue details is provided.	Yes
<b>FR21</b>	An existing venue will be updated in the system.	Yes
<b>FR22</b>	A certificate will be created for a trainee on an event.	Yes
<b>FR23</b>	A confirmation of the event having ended will be provided.	Yes
<b>FR24</b>	A certificate will be removed from the system.	Yes
<b>FR25</b>	A view of an event will be provided.	Yes
<b>FR26</b>	A view of a trainee will be provided.	Yes
<b>FR27</b>	A view of a course will be provided.	Yes
<b>FR28</b>	A view of a venue will be provided.	Yes
<b>FR29</b>	A list of search results will be provided.	Yes
<b>FR30</b>	A list of alternative search results will be provided.	Yes
<b>FR31</b>	A document is stored on the system and linked to an event or trainee.	Yes
<b>FR32</b>	A confirmation of the documents suitability for storage is provided.	Yes
<b>FR33</b>	An error message that prevents the event going over capacity should be displayed.	Yes
<b>FR34</b>	An error message that prevents the trainee from being added should be displayed.	Yes
<b>FR35</b>	An error message that prevents the trainer from being added should be displayed.	Yes

<b>FR36</b>	An error message that prevents the trainer from being added should be displayed.	Yes
-------------	--	-----

*\*Requirement no longer needs to be satisfied due the relaxation of rules around venue availability.*

## Appendix 6c - Validation Table (Non-functional requirements)

Requirement	Fit Criterion	Satisfied?
<b>NFR1</b>	The system can be accessed from outside the BEACON internal network.	Deferred*
<b>NFR2</b>	The system shares the same colours and logo as other organisational assets.	Yes
<b>NFR3</b>	All members of the admin team should be able to learn how to use the system in under 1 hour, and create an event in less than 1 minute.	Deferred*
<b>NFR4</b>	A member of the admin team can find an associated record in less than 5 seconds.	Deferred*
<b>NFR5</b>	The system shall provide a response to 99% of user requests within 3 seconds.	Yes
<b>NFR6</b>	The system shall create new records in less than 2 seconds.	Yes
<b>NFR7</b>	The system shall be updated once per month.	Deferred*
<b>NFR8</b>	The system shall be protected from unauthorised access 100% of the time.	Deferred*
<b>NFR9</b>	The system shall provide a like-for-like representation for 9 out of 10 BEACON terms.	Yes
<b>NFR10</b>	The system shall be approved by BEACON's legal representative.	Deferred*

*\*Deferred requirements require further analysis with the client to determine satisfaction.*